Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

INDUSTRIA INGENIARITZA TEKNIKOKO ATALA
SECCIÓN INGENIERÍA TÉCNICA INDUSTRIAL

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

2015 / 2016

T³: Time To Train

Memoria

DATOS DE LA ALUMNA O DEL ALUMNO	DATOS DEL DIRECTOR O DE LA DIRECTORA
NOMBRE Oskar	NOMBRE Mikel
APELLIDOS Gomes Hidalgo	APELLIDOS Villamañe Gironés
DNI 20221127X	DEPARTAMENTO Lenguajes y Sistemas Informáticos
FDO.:	FDO.:
FECHA:	FECHA:

Índice de contenidos

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

Cada vez más gente se dispone a mejorar su forma física realizando ejercicios anaeróbicos debido a los excesos que comete a lo largo del año o simplemente por querer mejorar su aspecto. Al mismo tiempo, mucha gente decide entrenar en sus propias casas o bien se inscribe en gimnasios sin tener unas nociones básicas sobre los entrenamientos a realizar. Dicho desconocimiento puede provocar lesiones musculares, aparte de ser un impedimento en el desarrollo de la musculatura.

Más allá de la cantidad de peso levantada durante los entrenamientos, el motivo por el cual no se desarrollan los músculos suele ser fruto tanto de una mala rutina de ejercicios como de una dieta inadecuada. Es muy común que el sujeto que está entrenando (sobre todo si es principiante) finalice las series de ejercicios cuando su capacidad física no da más de sí, lo que puede acabar convirtiéndose en una costumbre.

Para proporcionar al lector nociones básicas sobre entrenamientos de ejercicios anaeróbicos, se podría decir que una sesión de entrenamiento se divide en una lista de ejercicios que se realizan durante el transcurso de la misma. Generalmente cada uno de estos ejercicios se repiten un número reducido de veces (3 o 4), denominándose “series”. A su vez, cada serie de ejercicios ha de realizarse un número más elevado de veces, las llamadas “repeticiones”, en las cuales se suelen establecer unas marcas de tiempo para realizar los movimientos. Muchas veces, cuando la persona principiante que está entrenando eleva su esfuerzo físico al límite, tiende a realizar repeticiones más rápidas, las cuales son desfavorables, ya que dan la sensación de haber cumplido con la meta, pero al realizar un menor esfuerzo físico realmente no podrá progresar.

Para asesorar en los entrenamientos y realizarlos correctamente, se ha decidido desarrollar una aplicación para móviles Android denominada *Time To Train*, en la cual los usuarios podrán generar plantillas en las cuales configurarán las rutinas de ejercicios, especificando todos aquellos parámetros que son imprescindibles en una buena tabla de ejercicios: ilustración del ejercicio a realizar, cuántas series de cada ejercicio, número de repeticiones por serie, tiempos de descanso, etc. Todo ello mientras el reloj de la aplicación lleva la cuenta de los tiempos que se han establecido.

Con ello se pretende facilitar el entrenamiento del usuario, mientras a los más novatos les ayuda el contemplar una imagen del ejercicio a realizar, los más expertos pueden proceder a realizar los ejercicios sin tener que estar pendientes de configurar su cronómetro cada vez que finalizan una serie. Además se permitirá la opción de almacenar las plantillas generadas en un servidor de tal manera que se puedan compartir los entrenamientos y así desarrollar una aplicación que pueda competir con las existentes en el mercado actualmente.

A continuación se expondrán los motivos por los cuales se han escogido los dispositivos móviles y tablets como las plataformas objetivo sobre las que lanzar la aplicación.

El primero de ellos es la comodidad que ofrecen cuando se usan sus aplicaciones. *Time To Train* va a ser una aplicación en la que se espera que la mayor parte del tiempo sea de navegación (acceder a listados, ver detalles, iniciar entrenamiento, etc.), y es ahí donde estos dispositivos de reducido tamaño ven potenciadas sus posibilidades. Gracias al poder tocar cualquier parte de la pantalla la navegación es instantánea, además de poder utilizar la aplicación allá donde se desee transportar el dispositivo, función de la que carecen sus competidores (ordenadores y portátiles) dado que si se quiere entrenar en el gimnasio, por ejemplo, no quedaría otra opción más que llevar el dispositivo al lugar de entrenamiento, siendo algo incómodo o imposible de realizar. Al tratarse de smartphones y tablets la conectividad no es un problema, dado que todas disponen de conexión inalámbrica, ya sea mediante WIFI o mediante tarifa de datos.

Para finalizar se definirá porqué se ha escogido Android como OS de lanzamiento. Dicha decisión ha sido tomada entre varios factores. El más importante es que el OS de Google es el más extendido, alzándose con un 84,7% de cuota de mercado (1), además se disponía de un móvil Samsung Galaxy S con OS Android Gingerbread 2.3.6 y una tablet Woxter con OS Android Jelly Bean 4.2.2 para poder realizar las pruebas necesarias. El lanzar una aplicación sobre un OS tan extendido podría suponer una ventaja de cara a recuperar la inversión realizada en el transcurso del desarrollo. Además el lenguaje sobre el que se desarrollan las aplicaciones es Java, y aunque haya sido utilizado en la carrera para diferentes tareas, desarrollar para dispositivos Android es algo totalmente desconocido para el alumno y supone un reto, lo cual es una idea muy atractiva.

1.1. Definiciones, acrónimos y abreviaturas

- **ABD:** Administración de Bases de Datos. Asignatura cursada en tercer curso de carrera.
- **ADT:** Herramientas de desarrollo de Android (del inglés Android Developer Tools).
- **AJAX:** Asynchronous Javascript And Xml.
- **API:** Application Programming Interface. Es una interfaz diseñada para comunicar dos plataformas diferentes y que es utilizada por el software como capa de abstracción.
- **BD:** Base de Datos.
- **CSS:** Hojas de estilo en cascada (del inglés Cascading Style Sheets) para gestionar la apariencia de una página web.
- **DAS:** Desarrollo Avanzado de Software. Asignatura optativa cursada para ampliar conocimientos en el desarrollo para Android.
- **DAWE:** Desarrollo de Aplicaciones Web Enriquecidas. Asignatura optativa que profundiza en el uso de HTML5, Javascript y APIs para ofrecer aplicaciones web atractivas.
- **eCPM:** Costo por mil efectivo. Hace referencia a las ganancias que reporta la publicidad por cada 1000 impresiones que se realizan.
- **ECTS:** European Credit Transfer and accumulation System. Sistema de créditos europeo implantado en los nuevos grados donde se realiza un 50% de trabajo extra de manera personal.
- **EDA:** Estructura de Datos y Algoritmos. Asignatura de segundo curso de carrera donde se estudia la complejidad algorítmica y cálculo de tiempos en algoritmos.
- **EDT:** Esquema de Descomposición del Trabajo.
- **HCI:** Interacción Persona-Computador (del inglés Human-Computer Interaction).
- **HTML:** HyperText Markup Language. Lenguaje de marcas para diseño web.
- **IDE:** Entorno de desarrollo Integrado (del inglés IDE).
- **iOS:** sistema operativo de Apple (iPhone OS).
- **IS:** Ingeniería del Software. Asignatura de 2º curso en el cual se estudian patrones de diseño y normativas a tener en cuenta de cara al diseño de proyectos complejos.
- **JSON:** JavaScript Object Notation. Es un formato de texto que a través de una nomenclatura representa objetos que pueden ser utilizados en diferentes lenguajes de programación.
- **MD5:** Message-Digest Algorithm 5. Algoritmo de encriptación de datos y archivos utilizado en este proyecto para encriptar las contraseñas de los usuarios.
- **OS:** Sistema Operativo (Operating System).
- **PHP:** Acrónimo recursivo al igual que GNU, en este caso significa: PHP HyperText Pre-processor.
- **RAM:** Random Access Memory. Memoria interna de tipo volátil. Muy rápida para realizar accesos a los datos guardados en ella, pero como se indica, se pierde al apagar el dispositivo.
- **SGSSI:** Sistemas de Gestión de Seguridad y Sistemas de Información. Asignatura 3er curso en la cual se tratan temas de seguridad tanto web como en sistemas operativos.
- **TFG:** Trabajo de Fin de Grado.

- **UI:** User Interface. Representación gráfica de los elementos que se pueden mostrar en pantalla y con los que el usuario puede interactuar. Forma parte de cualquier sistema en el cual un usuario y un dispositivo se comunican entre sí y ofrecen reacciones.
- **URL:** Uniform Resource Locator o localizador de recursos uniforme es una formato de caracteres que designa la localización de unos recursos. En nuestro caso, la dirección en la que se encuentra una página web.
- **XAMPP:** Acrónimo donde X indica la plataforma y el resto Apache Mysql Php Perl.
- **XML:** eXtensible Markup Language.

2. PLANTEAMIENTO INICIAL

En este apartado se detallarán los objetivos a completar durante el proyecto, la arquitectura a utilizar, sus herramientas y estimaciones tanto para el tiempo necesario para desarrollar *Time To* Train como los gastos monetarios que incurriría este proceso.

2.1. Objetivos

El principal objetivo de este proyecto es desarrollar una aplicación para dispositivos móviles Android que permita al usuario tener un ayudante a la hora de realizar ejercicios anaeróbicos. Entre los puntos más importantes hay que destacar:

- Aprender a desarrollar aplicaciones para dispositivos Android.
- Recordar e incrementar los conocimientos de desarrollo Web que se han adquirido a lo largo de la carrera e interconectar esta tecnología con el desarrollo móvil anteriormente descrito.
- Definir y desarrollar un TFG que tenga una utilidad real para un grupo de gente, en este caso, las personas que realizan ejercicios físicos anaeróbicos.
- Poner al alumno en una situación nueva, en la que tendrá que hacer uso de todos los conocimientos adquiridos durante la carrera y demostrar si es capaz de asimilar conceptos nuevos en el desarrollo en un área diferente como son los dispositivos móviles.

2.2. Arquitectura

La arquitectura utilizada para desarrollar esta aplicación está basada en un esquema cliente-servidor (Ver Ilustración 1), la cual ha sido adaptada a partir de dos imágenes (27) y (28). Las funcionalidades propias de la aplicación las gestionará el móvil (parte cliente) y el almacenamiento de los datos para que otros usuarios puedan acceder a ellos se ejecutarán en un servidor a través de código PHP.

Aunque inicialmente se valoró la opción de no incluir almacenamiento de datos en un servidor y ejecutarlo todo en local, finalmente se adoptó la arquitectura cliente-servidor almacenando toda la información en una base de datos MySQL, ya que enriquece el proyecto, haciéndolo más completo, aparte de no perder los datos si el usuario desinstala la aplicación de su dispositivo.

Además tanto la página web desarrollada para poder incluir nuevos ejercicios en la aplicación sin recurrir a la modificación del código como las peticiones realizadas por el dispositivo móvil a la base remota se realizarán utilizando AJAX, una técnica que permite llamadas a los archivos PHP esperando una respuesta de ellos. De esta manera se pueden mostrar los datos sin recurrir a recargar la página, lo cual proporciona un acabado más elegante y profesional. Las respuestas del servidor serán cadenas de caracteres que utilizarán el estándar JSON, formato elegido por su facilidad a la hora de generar el objeto con toda la información a enviar o bien extraer la información recibida.

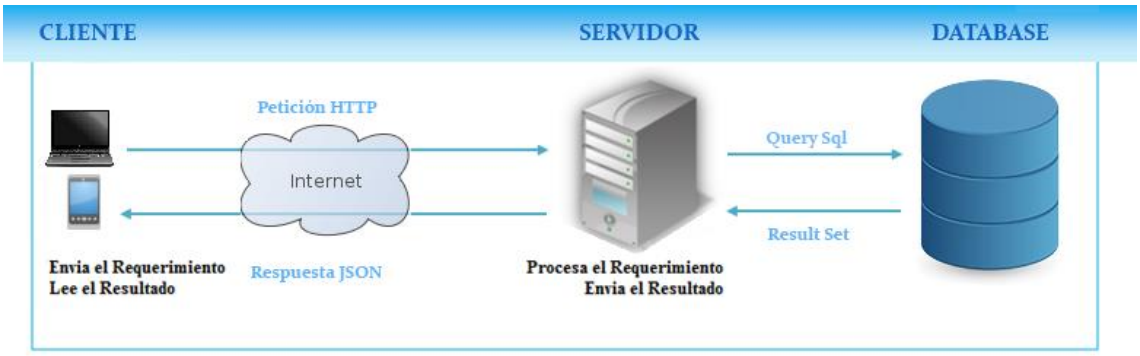


Ilustración 1: Esquema de arquitectura

2.3. Alcance

Para facilitar la planificación del TFG se ha optado por seguir el método de desarrollo por prototipos, para los cuales se ha dividido todo el proceso de desarrollo en paquetes individuales que añaden funcionalidades al proyecto gradualmente hasta su finalización.

Se ha generado una estructura de descomposición de trabajo EDT (Ver Ilustración 2), para así identificar fácilmente las diferentes fases por las que pasará el desarrollo del proyecto.

De esta manera se ayuda al desarrollador al poder dividir el proyecto por paquetes o prototipos e ir añadiéndole mayores funcionalidades según avanza el desarrollo, además de poder detectar a tiempo errores y solventarlos antes de que estos se tornen difíciles de modificar.

Las diferentes fases que se han identificado son:

- **Gestión:** Fase principal del TFG, en ella se realiza un boceto del proyecto que se quiere desarrollar en el cual se determinan las tareas más significativas. También hay que informarse de las características de Android, informándose sobre las diferentes posibilidades para su diseño y documentación.

- **Análisis:** Se definirán las diferentes funcionalidades que se quieran implementar en el futuro en la aplicación y la manera de afrontarlas. También se enumeran los recursos que serán necesarios durante el transcurso del proyecto.
- **Diseño:** En esta fase se procederá a determinar tanto la estructura con la que se va a implementar la aplicación como la parte gráfica de la misma, para la cual se utilizará la herramienta Cacoo y así ofrecer diseños preliminares con una relación calidad/coste más que razonable. De esta manera, en caso de desechar diseños, el impacto es mucho menor ya que su coste en tiempo es muy inferior sobre todo en las etapas iniciales.
- **Formación:** El alumno procederá a adquirir todos los conocimientos necesarios sobre el uso de las herramientas que va a necesitar y las cuales ya han sido definidas. Durante este proceso será necesario tanto leer manuales como poner en práctica lo aprendido.
- **Implementación:** Utilizando un desarrollo de trabajo basado en prototipos se han diferenciado los siguientes entregables de manera que se avance sobre seguro.

    - Prototipo 1: Web/Android: Login y registro.
    - Prototipo 2: Web: Gestión de ejercicios y gestión de cuenta.
    - Prototipo 3: Android: Generar plantilla.
    - Prototipo 4: Web/Android: Almacenar plantilla en BD
    - Prototipo 5: Web/Android: Gestión de plantillas y cuenta.
    - Prototipo 6: Web/Android: Funcionalidades del buscador.
    - Prototipo 7: Web/Android: Añadir plantillas a favoritos.
    - Prototipo 8: Android: Función Play: Añadir cronómetro.
    - Prototipo 9: Android: Función Play: Añadir indicador sonoro.

- **Pruebas:** Dado que cada prototipo que se vaya desarrollando en la fase de implementación va a tener que ser completamente funcional y sin posibilidad de errores, esta fase habrá que intercalarla con la anterior. El objetivo es servir como fase de testeo ante cualquier caso de prueba que se pueda dar. Esta fase es muy importante, ya que cuanto más completas sean las pruebas, menor será la probabilidad de que se cometan errores.
- **Documentación:** Aunque la memoria del TFG es un documento que se ha de ir completando a medida que se van alcanzando los distintos objetivos impuestos en la planificación, el último paso es finalizarla y proceder a la preparación de la defensa del proyecto.

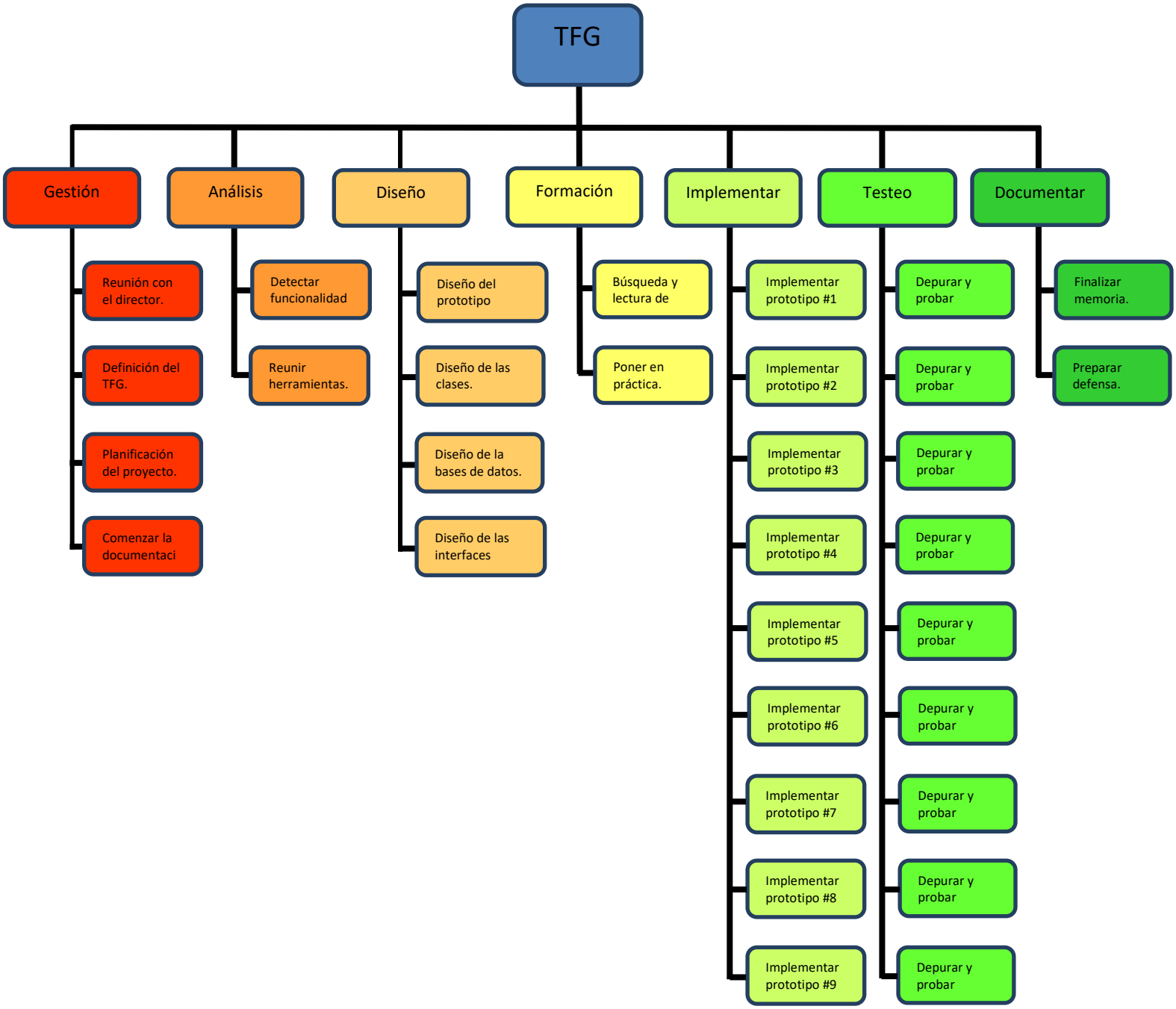


Ilustración 2: Diagrama EDT

Tareas

A continuación se detallan las tareas que conformarán todo el proceso de desarrollo y que han sido expuestas en el EDT anterior.

1- Alcance

Paquete de trabajo 1.1: Reunión con el director
Descripción Reunirse con el director de proyecto para valorar la viabilidad del proyecto a realizar. Duración 1 hora. Salidas/Entregables Esbozo de lo que será el proyecto. Recursos necesarios Tener en mente unas ideas sobre las que cimentar el proyecto. Precedencias Ninguna.

Paquete de trabajo 1.2: Definición del TFG
Descripción Establecer aquellos objetivos que deberá alcanzar el TFG para cubrir las necesidades que se hayan concretado con el director. Duración 3 horas. Salidas/Entregables Descripción de los objetivos del proyecto. Recursos necesarios Hoja de proposición de TFG cumplimentada, entregada y aceptada. Precedencias Paquete de trabajo 1.1.

Paquete de trabajo 1.3: Planificación del proyecto
Descripción Identificar las tareas que deberán llevarse a cabo durante el desarrollo del proyecto y establecer un orden para llevarlas a cabo. Duración 3 horas. Salidas/Entregables Lista de tareas identificadas. Recursos necesarios Tener claro las fases sobre las que pasará el desarrollo. Precedencias Paquete de trabajo 1.2.

Paquete de trabajo 1.4: Comenzar documentación
Descripción Iniciar el proceso de documentación sobre todos los informes y pruebas que se vayan realizando. Duración 10 horas. Salidas/Entregables Documento de memoria inicial. Recursos necesarios Viabilidad del trabajo aceptada. Precedencias Paquetes de trabajo 1.2 y 1.3.

2- Análisis

Paquete de trabajo 2.1: Detectar funcionalidades
Descripción Definir las funcionalidades que se van a implementar en la aplicación. Duración 10 horas. Salidas/Entregables Lista de funcionalidades que incorporará el proyecto con documentación sobre cómo afrontarlas. Recursos necesarios Tener claro qué debe hacer el proyecto exactamente. Precedencias Paquete de trabajo 1.2.

Paquete de trabajo 2.2: Reunir herramientas necesarias
Descripción Informarse acerca de qué herramientas pueden resultar útiles para documentar y desarrollar el proyecto, así como compararlas con otras herramientas similares. **Duración** 5 horas **Salidas/Entregables** Listado de herramientas que serán utilizadas durante todo el proyecto. **Recursos necesarios** Haber determinado las características inherentes a este proyecto: funcionalidades, entorno de desarrollo, almacenamiento en BD, etc. **Precedencias** Paquete de trabajo 2.1.

3- Diseño

Paquete de trabajo 3.1: Diseño del prototipo digital
Descripción Utilizar la herramienta Cacoo para diseñar un prototipo inicial de lo que será el aspecto de la aplicación para determinar aspectos importantes en cuanto a la usabilidad. **Duración** 8 horas. **Salidas/Entregables** Prototipo digital a valorar. **Recursos necesarios** Conocer el manejo de la plataforma Cacoo así como tener cuenta de usuario. **Precedencias** Paquete de trabajo 2.1

Paquete de trabajo 3.2: Diseño de las clases

Descripción

Una vez se tiene en mente cómo se mostrarán los datos del usuario, determinar las clases que serán necesarias para mostrar los datos por pantalla y añadirlas a la memoria.

Duración

6 horas.

Salidas/Entregables

Diagrama con las clases que serán utilizadas para implementar. Memoria actualizada.

Recursos necesarios

Haber generado un modelo de dominio a partir del cual identificaremos las clases necesarias.

Precedencias

Paquete de trabajo 2.1

Paquete de trabajo 3.3: Diseño de la base de datos

Descripción

Definir las tablas en las que se almacenarán los datos de la aplicación a partir del modelo de dominio generado en la fase de diseño, para así tener en mente las referencias que existan entre usuarios, cuentas y plantillas. El apartado de Análisis y diseño de la memoria debe actualizarse.

Duración

5 horas.

Salidas/Entregables

Modelo relacional que represente la base de datos. Memoria actualizada.

Recursos necesarios

Tener claro los datos a almacenar.

Precedencias

Paquete de trabajo 2.1

Paquete de trabajo 3.4: Diseño de las interfaces definitivas

Descripción

Diseñar las interfaces que se mostrarán al usuario en su manejo por la aplicación.

Duración

10 horas.

Salidas/Entregables

Conjunto de interfaces a utilizar.

Recursos necesarios

Haber valorado las alternativas propuestas durante el prototipo digital y poseer las herramientas necesarias para diseñarlas de manera definitiva.

Precedencias

Paquete de trabajo 3.1

4- Formación

Paquete de trabajo 4.1: Búsqueda y lectura de manuales
Descripción Buscar los manuales necesarios para aprender a manejar el lenguaje de programación Android y proceder a asimilar los conocimientos que exponen. Duración 8 horas. Salidas/Entregables Conocimientos teóricos de utilización del nuevo lenguaje adquiridos. Recursos necesarios Tener claro los nuevos lenguajes a aprender, manuales de desarrollo para OS Android, manuales de desarrollo Web, etc. Precedencias Haber finalizado la fase de diseño.

Paquete de trabajo 4.2: Realizar prácticas
Descripción Proceso en el cual se ponen en práctica todos los conocimientos adquiridos al leer los manuales de desarrollo en los nuevos lenguajes de programación. Duración 22 horas Salidas/Entregables Conocimientos puestos a prueba tras la realización de los tutoriales del manual. Recursos necesarios Entorno de desarrollo (IDE) de los nuevos lenguajes aprendidos. Precedencias Paquete de trabajo 4.1

5- Implementación

Paquete de trabajo 5.1: Prototipo 1- Login y Registro

Descripción

Implementar la parte inicial de los prototipos Web y Android que permita al usuario iniciar sesión en la aplicación y/o registrarse. Actualizar el apartado de desarrollo de la memoria.

Duración

25 horas.

Salidas/Entregables

Código con la implementación. Memoria actualizada.

Recursos necesarios

Tener definido el prototipo.

Precedencias

Paquetes anteriores finalizados con éxito.

Paquete de trabajo 5.2: Prototipo 2- Gestión Web

Descripción

Añadir las funcionalidades web referentes a la gestión tanto de ejercicios como de la cuenta de usuario. Actualizar el apartado de desarrollo de la memoria.

Duración

20 horas.

Salidas/Entregables

Código con la implementación. Memoria actualizada.

Recursos necesarios

Tener definido el prototipo.

Precedencias

Paquetes anteriores finalizados con éxito.

Paquete de trabajo 5.3: Prototipo 3- Generar plantilla.

Descripción

Implementar el prototipo que permita crear plantillas, añadiendo o modificando los ejercicios de la misma durante su creación. Actualizar el apartado de desarrollo de la memoria.

Duración

40 horas.

Salidas/Entregables

Código con la implementación. Memoria actualizada.

Recursos necesarios

Tener definido el prototipo.

Precedencias

Paquetes anteriores finalizados con éxito.

Paquete de trabajo 5.4: Prototipo 4- Almacenar plantilla en BD

Descripción
Añadir la funcionalidad de almacenar en la base de datos las plantillas que se van generando. Actualizar el apartado de desarrollo de la memoria.

Duración
20 horas.

Salidas/Entregables
Código con la implementación. Memoria actualizada.

Recursos necesarios
Tener definido el prototipo.

Precedencias
Paquetes anteriores finalizados con éxito.

Paquete de trabajo 5.5: Prototipo 5- Gestión de plantillas y cuenta.

Descripción
Implementación del prototipo que permite realizar modificaciones en las plantillas que hayan sido generadas por el propio usuario, ya sea modificándolas o eliminándolas. Además también se incluirá la gestión de cuentas vía dispositivo móvil. Actualizar el apartado de desarrollo de la memoria.

Duración
30 horas.

Salidas/Entregables
Código con la implementación. Memoria actualizada.

Recursos necesarios
Tener definido el prototipo.

Precedencias
Paquetes anteriores finalizados con éxito.

Paquete de trabajo 5.6: Prototipo 6- Funcionalidades del buscador

Descripción
Implementación del prototipo que permite realizar búsquedas de plantillas, ejercicios o usuarios. Actualizar el apartado de desarrollo de la memoria.

Duración
40 horas.

Salidas/Entregables
Código con la implementación. Memoria actualizada.

Recursos necesarios
Tener definido el prototipo.

Precedencias
Paquetes anteriores finalizados con éxito.

Paquete de trabajo 5.7: Prototipo 7- Añadir plantillas a favoritos

Descripción
Permitir que el usuario pueda agregar a favoritos plantillas de otros usuarios o bien eliminarlas de favoritos. Actualizar el apartado de desarrollo de la memoria.

Duración
30 horas.

Salidas/Entregables
Código con la implementación. Memoria actualizada.

Recursos necesarios
Tener definido el prototipo.

Precedencias
Paquetes anteriores finalizados con éxito.

Paquete de trabajo 5.8: Prototipo 8- Añadir cronómetro de seguimiento

Descripción
Implementación de la funcionalidad que activa el cronómetro durante las diferentes fases de los entrenamientos. Actualizar el apartado de desarrollo de la memoria.

Duración
20 horas.

Salidas/Entregables
Código con la implementación. Memoria actualizada.

Recursos necesarios
Tener definido el prototipo.

Precedencias
Paquetes anteriores finalizados con éxito.

Paquete de trabajo 5.9: Prototipo 9- Añadir indicador sonoro

Descripción
Implementar un indicador sonoro el cual avisa a la persona que está entrenando de los tiempos de descanso y comienzo del siguiente ejercicio. Actualizar el apartado de desarrollo de la memoria.

Duración
20 horas.

Salidas/Entregables
Código con la implementación. Memoria actualizada.

Recursos necesarios
Tener definido el prototipo.

Precedencias
Paquetes anteriores finalizados con éxito.

6- Testeo y corrección

Paquete de trabajo 6.1: Testeo y corrección del Prototipo 1

Descripción

Establecer un plan de pruebas para probar el prototipo correspondiente y depurar el código con el fin de solucionar los posibles errores. Actualizar el apartado de pruebas unitarias.

Duración

5 horas.

Salidas/Entregables

Prototipo 5.1 totalmente funcional. Memoria actualizada.

Recursos necesarios

Plan de pruebas y código del prototipo 1 finalizado.

Precedencias

Paquete de trabajo 5.1.

Paquete de trabajo 6.2: Testeo y corrección del Prototipo 2

Descripción

Establecer un plan de pruebas para probar el prototipo correspondiente y depurar el código con el fin de solucionar los posibles errores. Actualizar el apartado de pruebas unitarias.

Duración

5 horas.

Salidas/Entregables

Prototipo 5.2 totalmente funcional. Memoria actualizada.

Recursos necesarios

Plan de pruebas y código del prototipo 2 finalizado.

Precedencias

Paquete de trabajo 5.2.

Paquete de trabajo 6.3: Testeo y corrección del Prototipo 3

Descripción

Establecer un plan de pruebas para probar el prototipo correspondiente y depurar el código con el fin de solucionar los posibles errores. Actualizar el apartado de pruebas unitarias.

Duración

5 horas.

Salidas/Entregables

Prototipo 5.3 totalmente funcional. Memoria actualizada.

Recursos necesarios

Plan de pruebas y código del prototipo 3 finalizado.

Precedencias

Paquete de trabajo 5.3.

Paquete de trabajo 6.4: Testeo y corrección del Prototipo 4

Descripción

Establecer un plan de pruebas para probar el prototipo correspondiente y depurar el código con el fin de solucionar los posibles errores. Actualizar el apartado de pruebas unitarias.

Duración

5 horas.

Salidas/Entregables

Prototipo 5.4 totalmente funcional. Memoria actualizada.

Recursos necesarios

Plan de pruebas y código del prototipo 4 finalizado.

Precedencias

Paquete de trabajo 5.4.

Paquete de trabajo 6.5: Testeo y corrección del Prototipo 5

Descripción

Establecer un plan de pruebas para probar el prototipo correspondiente y depurar el código con el fin de solucionar los posibles errores. Actualizar el apartado de pruebas unitarias.

Duración

5 horas.

Salidas/Entregables

Prototipo 5.5 totalmente funcional. Memoria actualizada.

Recursos necesarios

Plan de pruebas y código del prototipo 5 finalizado.

Precedencias

Paquete de trabajo 5.5.

Paquete de trabajo 6.6: Testeo y corrección del Prototipo 6

Descripción

Establecer un plan de pruebas para probar el prototipo correspondiente y depurar el código con el fin de solucionar los posibles errores. Actualizar el apartado de pruebas unitarias.

Duración

5 horas.

Salidas/Entregables

Prototipo 5.6 totalmente funcional. Memoria actualizada.

Recursos necesarios

Plan de pruebas y código del prototipo 6 finalizado.

Precedencias

Paquete de trabajo 5.6.

Paquete de trabajo 6.7: Testeo y corrección del Prototipo 7

Descripción
Establecer un plan de pruebas para probar el prototipo correspondiente y depurar el código con el fin de solucionar los posibles errores. Actualizar el apartado de pruebas unitarias.

Duración
5 horas.

Salidas/Entregables
Prototipo 5.7 totalmente funcional. Memoria actualizada.

Recursos necesarios
Plan de pruebas y código del prototipo 7 finalizado.

Precedencias
Paquete de trabajo 5.7.

Paquete de trabajo 6.8: Testeo y corrección del Prototipo 8

Descripción
Establecer un plan de pruebas para probar el prototipo correspondiente y depurar el código con el fin de solucionar los posibles errores. Actualizar el apartado de pruebas unitarias.

Duración
5 horas.

Salidas/Entregables
Prototipo 5.8 totalmente funcional. Memoria actualizada.

Recursos necesarios
Plan de pruebas y código del prototipo 8 finalizado.

Precedencias
Paquete de trabajo 5.8.

Paquete de trabajo 6.9: Testeo y corrección del Prototipo 9

Descripción
Establecer un plan de pruebas para probar el prototipo correspondiente y depurar el código con el fin de solucionar los posibles errores. Actualizar el apartado de pruebas unitarias.

Duración
5 horas.

Salidas/Entregables
Prototipo 5.9 totalmente funcional. Memoria actualizada.

Recursos necesarios
Plan de pruebas y código del prototipo 9 finalizado.

Precedencias
Paquete de trabajo 5.9

7- Documentación

Paquete de trabajo 7.1: Finalizar memoria
Descripción Finalizar el documento de la memoria. En él se incluirán conclusiones, la diferencia entre la estimación inicial y la real y diferentes anotaciones o modificaciones relacionadas con el desarrollo de la aplicación. **Duración** 30 horas. **Salidas/Entregables** Documento de memoria del TFG. **Recursos necesarios** Haber finalizado el desarrollo de la aplicación móvil y la experiencia obtenida a lo largo del mismo. **Precedencias** Paquetes de trabajo 5.X y 6.X.

Paquete de trabajo 7.2: Preparar defensa
Descripción Desarrollar la guía con la cual se defenderá el proyecto y ensayar su presentación. **Duración** 10 horas. **Salidas/Entregables** Presentación para realizar la defensa del TFG. **Recursos necesarios** Memoria finalizada. Solicitud de defensa proyecto aceptada. **Precedencias** Paquete de trabajo 7.1.

2.4. Planificación temporal

Ahora que se han identificado y enumerado las fases a través de las cuales avanzará el desarrollo del proyecto, es necesario mostrarlas de manera más simple y comprensible para el lector.

Para estimar el tiempo que va a llevar finalizar el proyecto se fijó un límite en cuanto a la fecha para entregarlo, y a partir de ello calcular el tiempo que se dedicaría al mismo en cada una de las fases aproximadamente. Dado que el TFG se considera una asignatura optativa de 12 créditos y, por extensión de lo que se considera un crédito ECTS = 10 horas lectivas + 15 horas trabajo en casa, se estimó que un proyecto tiene una duración media de 300 horas, aunque diversas fuentes (profesor Héctor de la Presa, por ejemplo) indican que numerosos alumnos exceden ese margen de tiempo por mucho siendo un tiempo muy común un desarrollo en torno a 550 o 600 horas.

Considerando el uso de un entorno de programación desconocido, funciones nuevas para desarrollar en dispositivos Android y posibles problemas que surgirán a lo largo del mismo, se ha calculado que el tiempo que llevaría realizar el proyecto de la siguiente manera:

Tabla 1: Estimación de tiempos 1/2

Tarea	Duración estimada (horas)
1- Gestión	**17**
1.1- Reunión con el director	1
1.2- Definición del TFG	3
1.3- Planificación del proyecto	3
1.4- Comenzar documentación	10
2- Análisis	**15**
2.1- Detectar funcionalidades	10
2.2- Reunir herramientas necesarias	5
3- Diseño	**29**
3.1- Diseño del prototipo digital	8
3.2- Diseño de las clases	6
3.3- Diseño de la base de datos	5
3.4- Diseño de las interfaces definitivas	10
4- Formación	**30**
4.1- Búsqueda y lectura de manuales	8
4.2- Realizar prácticas	22

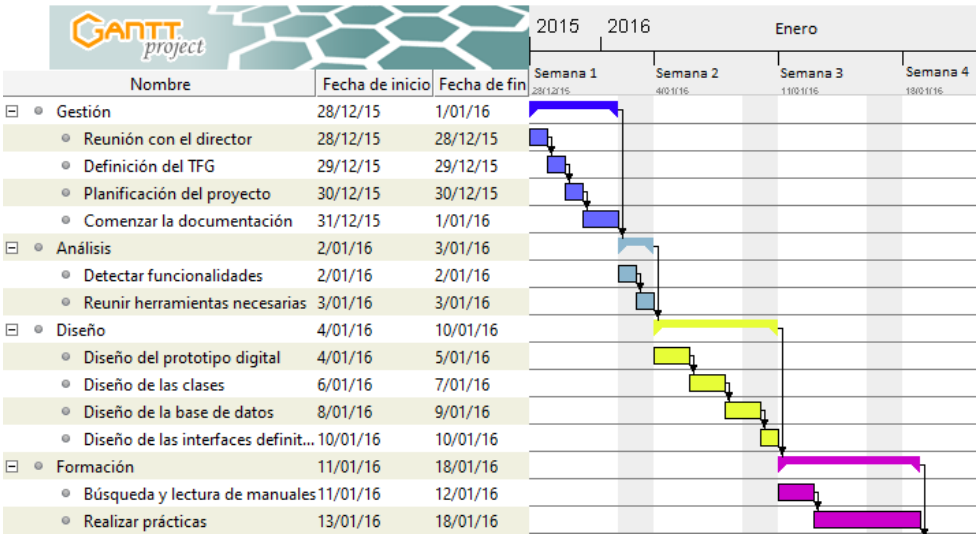
Tabla 2: Estimación de tiempos 2/2

Tarea	Duración estimada (horas)
5- Implementación	**245**
5.1- Implementación del prototipo 1	25
5.2- Implementación del prototipo 2	20
5.3- Implementación del prototipo 3	40
5.4- Implementación del prototipo 4	20
5.5- Implementación del prototipo 5	30
5.6- Implementación del prototipo 6	40
5.7- Implementación del prototipo 7	30
5.8- Implementación del prototipo 8	20
5.9- Implementación del prototipo 9	20
6- Testeo y corrección	**45**
6.1- Testeo del prototipo 1	5
6.2- Testeo del prototipo 2	5
6.3- Testeo del prototipo 3	5
6.4- Testeo del prototipo 4	5
6.5- Testeo del prototipo 5	5
6.6- Testeo del prototipo 6	5
6.7- Testeo del prototipo 7	5
6.8- Testeo del prototipo 8	5
6.9- Testeo del prototipo 9	5
7- Documentación	**40**
7.1- Finalizar memoria	30
7.2- Preparar defensa	10
Total estimado (horas):	**421**

Finalmente se muestra un diagrama de Gantt (Ver Ilustración 3 e Ilustración 4) que ha sido elaborado considerando que se va a trabajar en el proyecto 7 días de la semana a razón de 5 horas diarias. Además, dado el interés mostrado por adquirir nuevos conocimientos en esta área así como la necesidad de acabarlo antes de un plazo límite se van a invertir tanto fines de semana como festivos en finalizar la aplicación, lo cual indica que una fecha estimada sobre la que el proyecto acabará se sitúa a aproximadamente 84 días desde la fecha de hoy, es decir, en unas 12 semanas el proyecto debería haberse finalizado.

Hay que recalcar que aunque existan tareas que puedan llevarse a cabo en paralelo, al ser un trabajo individual se van a realizar secuencialmente y así trabajar siempre sobre una base sólida y bien realizada para no tener que volver hacia atrás en el desarrollo del TFG y cumplir con el plazo auto-exigido.

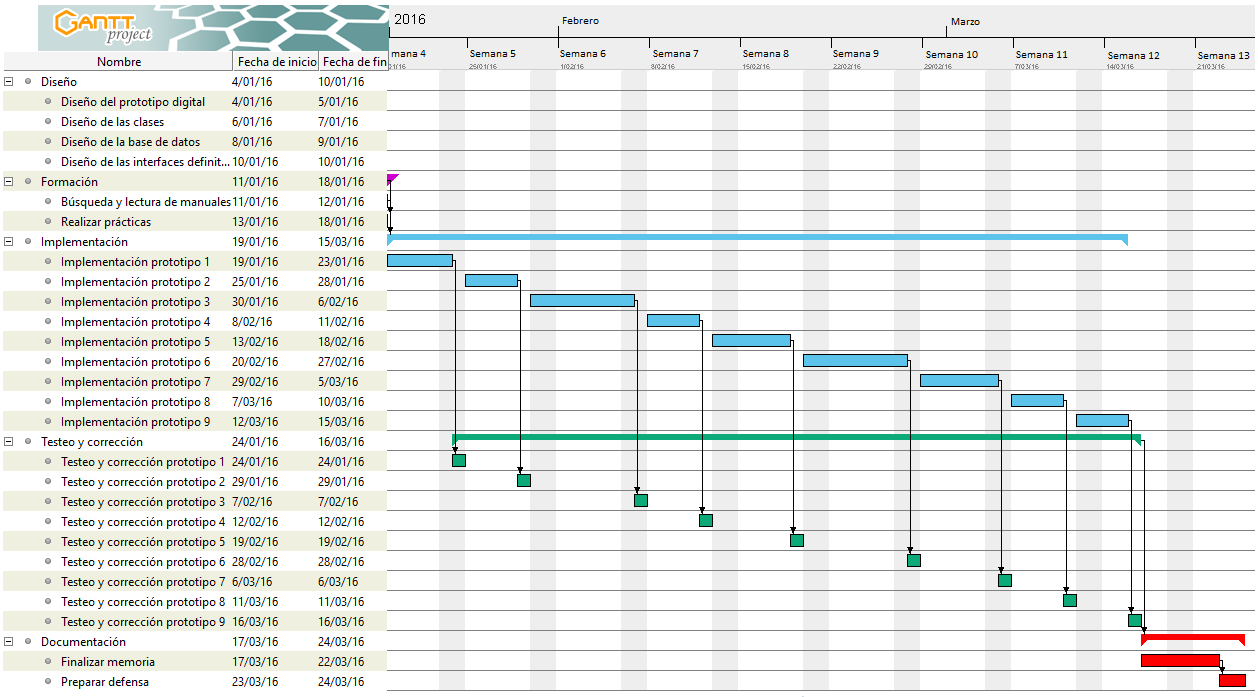
La fase inicial del proyecto se ha estimado como el proceso de análisis, diseño y autoformación para desarrollarlo y viene ilustrado a continuación:



Nombre	Fecha de inicio	Fecha de fin
Gestión	28/12/15	1/01/16
Reunión con el director	28/12/15	28/12/15
Definición del TFG	29/12/15	29/12/15
Planificación del proyecto	30/12/15	30/12/15
Comenzar la documentación	31/12/15	1/01/16
Análisis	2/01/16	3/01/16
Detectar funcionalidades	2/01/16	2/01/16
Reunir herramientas necesarias	3/01/16	3/01/16
Diseño	4/01/16	10/01/16
Diseño del prototipo digital	4/01/16	5/01/16
Diseño de las clases	6/01/16	7/01/16
Diseño de la base de datos	8/01/16	9/01/16
Diseño de las interfaces definit...	10/01/16	10/01/16
Formación	11/01/16	18/01/16
Búsqueda y lectura de manuales	11/01/16	12/01/16
Realizar prácticas	13/01/16	18/01/16

Ilustración 3: Diagrama de GANTT 1/2

Como se explicó anteriormente, las tareas de implementación y comprobación de errores están ligadas, de tal manera que cuando se finaliza la implementación de una nueva funcionalidad se comprueba el funcionamiento de la misma para detectar y solucionar errores, como se puede ver en la Ilustración 4.



Nombre	Fecha de inicio	Fecha de fin
Diseño	4/01/16	10/01/16
Diseño del prototipo digital	4/01/16	5/01/16
Diseño de las clases	6/01/16	7/01/16
Diseño de la base de datos	8/01/16	9/01/16
Diseño de las interfaces definit...	10/01/16	10/01/16
Formación	11/01/16	18/01/16
Búsqueda y lectura de manuales	11/01/16	12/01/16
Realizar prácticas	13/01/16	18/01/16
Implementación	19/01/16	15/03/16
Implementación prototipo 1	19/01/16	23/01/16
Implementación prototipo 2	25/01/16	28/01/16
Implementación prototipo 3	30/01/16	6/02/16
Implementación prototipo 4	8/02/16	11/02/16
Implementación prototipo 5	13/02/16	18/02/16
Implementación prototipo 6	20/02/16	27/02/16
Implementación prototipo 7	29/02/16	5/03/16
Implementación prototipo 8	7/03/16	10/03/16
Implementación prototipo 9	12/03/16	15/03/16
Testeo y corrección	24/01/16	16/03/16
Testeo y corrección prototipo 1	24/01/16	24/01/16
Testeo y corrección prototipo 2	29/01/16	29/01/16
Testeo y corrección prototipo 3	7/02/16	7/02/16
Testeo y corrección prototipo 4	12/02/16	12/02/16
Testeo y corrección prototipo 5	19/02/16	19/02/16
Testeo y corrección prototipo 6	28/02/16	28/02/16
Testeo y corrección prototipo 7	6/03/16	6/03/16
Testeo y corrección prototipo 8	11/03/16	11/03/16
Testeo y corrección prototipo 9	16/03/16	16/03/16
Documentación	17/03/16	24/03/16
Finalizar memoria	17/03/16	22/03/16
Preparar defensa	23/03/16	24/03/16

Ilustración 4: Diagrama de GANTT 2/2

2.5. Herramientas

En este apartado se especificarán las diferentes herramientas que se van a utilizar a lo largo del proceso de desarrollo.

- **Software utilizado:**
  - ***Cacoo:*** Es una herramienta web de diseño de prototipos. Dicha herramienta es muy potente, ya que permite elaborar complejos diseños y dotarlos de cierta funcionalidad, de manera que permite simular un funcionamiento real de una aplicación de software cuya lógica no ha sido implementada. Cacoo ha sido escogida ya que fue utilizada en la asignatura Interacción Persona-Computador y se considera que puede resultar de gran ayuda para el proceso de diseño de interfaces. Dados los conocimientos que se adquirieron durante el proceso de aprendizaje de la asignatura nos permite crear prototipos de la aplicación en un período de tiempo bastante bajo y así escoger la mejor opción, ya que en el momento de comenzar el proyecto la experiencia con el uso de "*layouts*" era muy baja.

Ilustración 5: Logo de Cacoo (2)

  - ***Visual Paradigm 12.0:*** Software esencial para generar los diagramas referentes a la captura de requisitos y análisis en el lenguaje de representación de modelado UML. Con él se realizarán los diseños de los diagramas de casos de uso, modelo de dominio, diagrama de clases, la estructura de las bases de datos y los diagramas de secuencia.

Ilustración 6: Logo de Visual Paradigm

- ***Paquete de ofimática Office 2013:*** Incluye diversas herramientas que se utilizarán para documentar todo el proceso y preparar la presentación. En especial se usará el procesador de textos *Microsoft Word 2013* para realizar este documento, la hoja de cálculo *Microsoft Excel 2013* para la creación de tablas y Microsoft PowerPoint 2013 para preparar la defensa de la presentación.

Ilustración 7: Logo de Microsoft Office 2013 (3)

- ***GanttProject 2.7.1:*** Aplicación que permite la creación de gráficos Gantt para prever la fecha de finalización de un proyecto dado un número de tareas con sus respectivas duraciones y precedencias, en caso de haberlas. Permite exportar el diagrama GANTT en formato .png lo que es idóneo para insertar en cualquier documento.

Ilustración 8: Logo de GanttProject (4)

- ***Android Studio 1.5.1:*** entorno de desarrollo integrado (IDE) desarrollado por Google utilizado tanto en el proceso de aprendizaje realizando tutoriales como en la fase de implementación y depuración del código en busca de errores si los hubiese. En un comienzo se debatió entre usar Eclipse como entorno de desarrollo, pero a fecha de diciembre de 2014 se lanzó la primera versión estable y el plugin ADT de Eclipse dejó de recibir soporte en cuanto a desarrollo. De esta manera Google comenzó a recomendar Android Studio como el entorno de desarrollo más completo al incluir novedades como la herramienta Gradle, el cual es un sistema de compilación basado en Java que

permite introducir tus propios scripts, siendo capaz de compilar de acuerdo a las modificaciones introducidas.

Ilustración 9: Logo de Android Studio (5)

- ***Dropbox:*** Aplicación que permite almacenar cualquier archivo que se quiera en un servidor. Gracias al email de la universidad está permitido que los alumnos almacenen una cantidad de datos superior a la normal, no obstante el almacenamiento base es de 2GB y es más que suficiente para guardar todos los documentos necesarios. El principal uso de esta aplicación es la de poder disponer de los datos en cualquier momento (por ejemplo, demostración de diagramas ante el director) así como la de tener una copia de seguridad en caso de pérdida. En el apartado de gestión de riesgos se detallará más en profundidad.

Ilustración 10: Logo de Dropbox (6)

- ***Paint + GIMP:*** Software gratuito utilizado para realizar modificaciones en ilustraciones o en tareas de diseño que luego utilizará *Time To Train*. En principio se usará Paint para retocar alguna imagen que se vaya a añadir al documento de la memoria dado que es un editor muy básico pero a la vez es muy sencillo de utilizar, lo cual no necesita de un proceso de aprendizaje. En caso de no encontrar las imágenes de dominio público que se necesiten para la aplicación se procederá a utilizar una herramienta más compleja como puede ser GIMP y así crear imágenes propias que caractericen esta aplicación.

Ilustración 11: Logo de GIMP (7)

- ***XAMPP:*** Paquete gratuito de aplicaciones que permitirán trabajar desde el ordenador personal como si se tuviese un servidor real contratado, utilizándose para implementar la funcionalidad de almacenamiento ya que se ha utilizado anteriormente en múltiples ocasiones y se sabe de antemano que permite probar en él un grabado de datos. Dependiendo del sistema operativo que se posea, la primera letra del paquete XAMPP coincidirá con la del OS del sistema, existiendo versiones para Linux, Windows y Mac OS, que coinciden con LAMPP, WAMPP Y MAMPP. El resto de siglas hacen referencia a las diferentes aplicaciones que trae el paquete, las cuales son:
  - Apache: realiza la simulación de un servidor.
  - MySQL: lenguaje para gestión de bases de datos.
  - PHP: lenguaje de programación para gestionar el funcionamiento del servidor.
  - Perl: lenguaje de programación basado en C.

Ilustración 12: Logos de aplicaciones XAMPP. Apache, MySQL y PHP (de izda. a dcha.)

- ***Sublime Text 2:*** Editor de código gratuito multiplataforma en la cual se puede seleccionar el lenguaje en el cual se quiere programar y con ello colorea las funciones para que se identifiquen más rápidamente. Con él se ha desarrollado toda la parte relacionada con el diseño y la lógica de la Web (HTML, JavaScript, CSS y PhP).

Ilustración 13: Logo de Sublime Text 2 (8)

- **Hardware utilizado:**
    - ***Samsung Galaxy S:*** móvil con sistema operativo Android Gingerbread 2.3.6 utilizado para comprobar compatibilidad y funcionamiento.
    - ***Tablet Woxter QX100:*** tablet con sistema operativo Android Jelly Bean 4.2.2 utilizado para comprobar compatibilidad y funcionamiento.
    - ***Ordenador de sobremesa Intel Core2-Quad Q9650:*** Ordenador con sistema operativo Windows 10 64 bits en el que se llevará todo el proceso de desarrollo.

2.6. Gestión de riesgos

En este punto se detallarán el plan de gestión de riesgos asociados a este proyecto, ya que son una de las características inherentes a todo proyecto. Existen dos factores que conlleva todo riesgo, el primero es la incertidumbre, ya que es imposible prever cuándo se producirá ese evento con un impacto negativo para el proyecto o si siquiera se hará efectivo. El control de la incertidumbre está fuera del alcance de cualquier persona. El segundo factor es la pérdida potencial, ya que de hacerse efectivo el riesgo pueden ocurrir consecuencias no deseadas así como pérdidas de material.

Para minimizar la posibilidad de que se hagan efectivos los riesgos se establecen planes de prevención individuales de manera que se consigue reducir la probabilidad de que se produzca. Además en caso de producirse el desastre tiene que haberse definido un plan de contingencia, el cual está pensado para minimizar el impacto negativo que pudiese tener en el desarrollo del proyecto.

Las probabilidades de cada riesgo se asignarán mediante una probabilidad: Baja, media, alta la cual la cual se puede calcular de acuerdo a los baremos establecidos en la siguiente tabla (Ver Tabla 3).

Tabla 3: Probabilidades de riesgos

Tipo de probabilidad	Porcentaje (%)
Probabilidad alta	66,67%-100%
Probabilidad media	33,34%-66,66%
Probabilidad baja	0%-33,33%

En caso de hacerse efectivo algún riesgo afectará negativamente al desarrollo del proyecto retrasando su desarrollo en una cantidad de tiempo que se ha valorado de la siguiente manera:

Tabla 4: Impacto

Impacto	Tiempo perdido
Bajo	menos de 1 día
Medio	1-2 días
Alto	3-4 días
Muy alto	5 o más días

A continuación se listan los diferentes riesgos asociados a este proyecto.

Avería en equipos informáticos

Descripción

Dado el tiempo de vida que posee el equipo informático y el uso que se le ha dado es ligeramente probable que fallen componentes como la tarjeta gráfica, que recientemente ha enviado un mensaje de error en pantalla, así como el disco duro, el cual se nota que falla en algunas ocasiones.

Prevención

Mantenimiento periódico de hardware.

Plan de contingencia

Buscar el punto crítico que originó el fallo y reemplazarlo con piezas nuevas. Restaurar copia de seguridad más reciente en caso de pérdida de información.

Probabilidad

Baja.

Impacto

Muy alto, debido al considerable retraso que reportaría este tipo de fallo y la imposibilidad de continuar trabajando durante ese lapso de tiempo al no disponer de otro equipo mientras se reemplazan las piezas.

Pérdida de la conexión a internet del PC de sobremesa

Descripción
Ocasionalmente Euskaltel sufre fallos en proveer servicio de conexión a internet. Dichos fallos son aleatorios y tanto su impacto como su duración son variables, pudiendo ser desde caídas temporales hasta pérdida permanente de conexión durante 1 día.

Prevención
Guardar una copia de seguridad lo más actualizada posible en el ordenador para continuar el trabajo desde ahí.

Plan de contingencia
Tener descargados todos los documentos necesarios para proceder con el trabajo y no depender de la conexión (por ejemplo: manuales, apuntes, etc.).

Probabilidad
Muy baja.

Impacto
Variable en función del tiempo que se pierda.

Modificación o eliminación errónea

Descripción
El alumno es muy dado a eliminar archivos y realizar modificaciones/formateos, sin contar con que si se borra algo de la carpeta de Dropbox del ordenador mientras el programa está activo, también se modifican los datos de la nube.

Prevención
Llevar con rigor un proceso de copias de seguridad de manera que siempre se tenga al menos una copia de seguridad. Para ello se pueden utilizar memorias externas, servidores gratuitos de almacenamiento como Dropbox o Google Drive siempre y cuando se tenga una especial atención en sincronizar los datos únicamente al finalizar los cambios, cerrando la nube una vez se han actualizado.

Plan de contingencia
Recuperar el archivo más próximo al cambio o eliminación y restaurarlo.

Probabilidad
Media-Baja.

Impacto
Dependiendo del archivo podría ser crítico o insignificante.

Pérdida de dispositivo móvil
Descripción Como el título indica, la posibilidad de perder el dispositivo móvil, ya sea por caída del bolsillo o simplemente por avería. **Prevención** Tener especial atención cuidándolo durante el proceso de creación del proyecto y tener cuidado de no extraviarlo. **Plan de contingencia** Realizar simulaciones usando el AVD de Android Studio. **Probabilidad** Baja. **Impacto** Medio, debido a la lentitud con la que realiza simulaciones en el ordenador, a la larga sería una pérdida de tiempo molesta.

Baja por enfermedad
Descripción Al comenzar el año las temperaturas bajan y el riesgo de contagio es mayor. **Prevención** Tomar medidas de seguridad para no caer contagiado. **Plan de contingencia** Guardar reposo en la medida de lo posible, tomar medicación. **Probabilidad** Media-Alta **Impacto** Alto. Debido a la pérdida de concentración y tener que guardar reposo hasta la mejora del desarrollador.

Problemas en el diseño de algoritmos

Descripción

Aunque el lenguaje de programación es Java y se conoce, existen multitud de clases y funciones que son nuevas y no se conoce su funcionamiento. Esto puede acarrear un mal funcionamiento en el algoritmo diseñado y el desconocimiento de porqué no funciona.

Prevención

Repasar el algoritmo con cuidado y realizar pruebas unitarias de cada elemento que intervenga y reunir manuales orientativos con buenos ejemplos.

Plan de contingencia

Consultar con el profesor.

Probabilidad

Media.

Impacto

Medio.

Cambios en el servicio gratuito Dropbox

Descripción

Actualmente Dropbox es un servicio gratuito que ofrece un límite de espacio en el que almacenar datos. Recientemente ha habido cambios en otros similares (OneDrive) reduciendo el espacio que ofrecían. De hacerse efectivo el mismo cambio en Dropbox podría tener consecuencias negativas.

Prevención

Mantener la copia del servidor y la del ordenador lo más similares posibles.

Plan de contingencia

Proceder al traslado de los datos a otros servidores similares: OneDrive o Google Drive.

Probabilidad

Muy baja.

Impacto

Variable en función de las veces que se sincronicen los datos de la nube con el disco duro del ordenador. En principio debería ser muy bajo o bajo si se realiza diariamente.

Nuevas funcionalidades/Mala estructuración
Descripción Aunque se dedique tiempo a analizar todo lo necesario para llevar a cabo el proceso de diseño e implementación, es probable que se pasen por alto funcionalidades esenciales que se detecten en fases más tardías y eso tenga un impacto negativo. **Prevención** Mantener unos objetivos del proyecto desde el principio y bien definidos, de manera que no puedan existir dudas en el futuro sobre si sería apropiado incluir nuevos elementos al mismo. **Plan de contingencia** Localizar la raíz del problema y realizar las modificaciones necesarias empezando desde la base. **Probabilidad** Variable en función del diseño, pero se estima que su probabilidad sea baja. **Impacto** Alto al tener que rehacer diversos apartados como documentación, diseño, implementación, etc.

Incompatibilidad de versiones
Descripción En ocasiones tanto Android Studio como las APIs disponibles lanzan una nueva versión. En caso de actualizar la versión pueden ocurrir problemas de compatibilidad con elementos antiguos o bien forzar a implementar clases de manera diferente a la forma tradicional, especialmente en Android 6.0 Marshmallow (API lvl23). **Prevención** Revisar los cambios que traen las nuevas versiones y ver si podría entrar en conflicto alguna implementación antigua. **Plan de contingencia** Consultar en Android Developers o StackOverflow. **Probabilidad** Alta. Tanto Android Studio como las APIs lanzan versiones nuevas con relativa frecuencia. **Impacto** Medio.

2.7. Evaluación económica

Para realizar una estimación económica se han diferenciado los gastos en directos e indirectos. El primer tipo proviene de aquellos gastos que se pueden asignar de manera clara ya que se pueden deducir con precisión: horas trabajadas, alquiler de un servidor externo si lo hubiera, etc. El segundo tipo proviene de aquellos gastos los cuales son difíciles de valorar cuanto valor aportan al proyecto realmente. Por ejemplo, de los gastos de luz que se realicen en esta vivienda no se sabe exactamente qué cantidad está destinada a mantener el equipo informático activo para desarrollar el proyecto.

Para realizar la estimación como si de un proyecto con finalidad comercial se tratase, se han calculado los costes de todos los elementos que han incurrido en el desarrollo del mismo y considerando que las licencias no han sido gratuitas.

Los gastos con su tipo asociado que se han detectado son:

Tabla 5: Tipos de gastos

Descripción del gasto	Tipo de gasto
Sueldo de personal	Directo
Amortización de equipos informáticos	Indirecto
Amortización de dispositivo móvil (smartphone)	Indirecto
Amortización de dispositivo móvil (Tablet)	Indirecto
Licencia de Microsoft Office 2013	Indirecto
Licencia Visual Paradigm 12.0	Indirecto
Factura de la luz	Indirecto

Gastos directos

- Sueldo del personal: Se estimará que el sueldo medio de un trabajador es de 30€/hora. Antes de calcular el coste que supondrá el sufragar los costes de personal, hay que deducir el coste asociado al proceso de formación, ya que en el ámbito de desarrollo de software son conocimientos que corren por cuenta del trabajador y no de la empresa. Así pues poseemos los siguientes datos:

Tabla 6: Cálculo de gastos de personal

Descripción	Cantidad
Tiempo de desarrollo total	421 horas
Tiempo de formación	30 horas
Sueldo medio trabajador	30 €/hora

Con esos datos podemos calcular el sueldo real a percibir por el trabajador, el cual se muestra a continuación (Ver Tabla 7):

Tabla 7: Costes de personal

Descripción	Coste(€)
Coste total del proyecto	12.630
Deducciones por formación	900
Coste real a pagar	**11.730**

Gastos indirectos

A continuación se enumeran los diferentes gastos indirectos los cuales analizaremos en la tabla de amortizaciones de gastos indirectos (Ver Tabla 8).

- **Amortización de equipos informáticos:** Se posee un ordenador Intel Core2-Quad con procesador CPU 9650 a 3.00 GHz adquirido en verano de 2008 por un valor de 800€ aproximadamente. Se planea sustituirlo a finales de 2016, con lo cual habrá tenido una duración total de unos 8 años. Se calculará la amortización relativa a las 13 semanas de uso en el proyecto dándole un uso del 60% durante la duración del mismo.
- **Amortización de dispositivo móvil:** Para realizar pruebas sobre si las funcionalidades que se estaban implementando eran compatibles con el OS Android Gingerbread 2.3.6 se utilizó un móvil Samsung Galaxy S I-9000 adquirido en 2011 el cual tenía un precio de venta de 495€ y se pretende reemplazar en 2016. Su uso será únicamente del 20%.
- **Amortización de tablet Woxter:** Dispositivo de tipo tablet utilizado para realizar las pruebas del código. Será más utilizado que el móvil dado que tiene más potencia y no tiene los problemas que tiene el móvil ya que es antiguo y en ocasiones presenta errores de inestabilidad. Dado que el proyecto que se está desarrollando quiere lanzarse en móviles, es obligatorio probarlo en uno, aunque para la mayoría de pruebas simples se usará este dispositivo tablet. El uso que se le dará a este dispositivo es del 40%. Las características son Tablet Woxter QX100 Quad-Core con 1GHz de RAM y OS 4.2.2. Su precio de venta es de 140€ y se prevé que su duración sea de al menos 4 años, ya que su batería ha sido sustituida recientemente.
- **Licencia de Microsoft Office 2013:** paquete de ofimática con un valor de 149€ cuya duración es de 3 años, estando a punto de actualizar a Office 2016. Se estima que el uso del mismo estará vinculado en un 90% al proyecto.
- **Licencia Visual Paradigm 12.0:** posee un valor de 299$, el equivalente a 275,31€. La duración prevista también es de 3 años y su uso será destinado en su totalidad al diseño del proyecto.
- **Factura de luz:** Es un gasto de tipo indirecto un tanto especial, ya que es muy difícil estimar el consumo eléctrico de los dispositivos. Según datos de la OCU *una casa española tiene unos gastos medios de 990 euros al año en energía* (9). Eso nos da un consumo medio de 82,5€ mensuales en electricidad. En principio supondremos que únicamente un 20% del consumo lo produce el equipo.

A continuación se muestran los cálculos de las amortizaciones anteriormente mencionadas.

Tabla 8: Cálculo de amortizaciones de gastos indirectos

Descripción	Coste (€)	Duración	Amort. Anual(€)	Tiempo de uso	Uso (%)	Coste (€)
PC Core2-Quad Q9650 3.00GHz	800	8 años	100	13 semanas	60	15
Samsung Galaxy S	495	4 años	123,75		20	6,19
Tablet Woxter	140		35		40	3,5
Total costes de hardware	-	-	-	-	-	**24,69**
Microsoft Office 2013	149	3 años	49,67	13 semanas	90	11,18
Visual Paradigm 12.0	275,31		91,77		100	22,94
Total costes de licencias	-	-	-	-	-	**34,12**
Luz	-	-	990	13 semanas	20	49,5
Total costes energéticos	-	-	-	-	-	**49,5**

Teniendo en cuenta el valor inicial del producto procedemos a calcular la amortización anual de la siguiente manera: **Coste total/ Duración = Amortización Anual**. A partir de ese dato calculamos cuánto se amortiza durante únicamente el proceso de desarrollo, el cual ha sido estimado en 13 semanas: **Amortización Anual * (13 semanas /52 semanas al año) * %Uso**.

Finalmente mostramos el resumen de gastos viene resumido en la siguiente tabla:

Tabla 9: Resumen de estimación de costes

Tipo de costes	Coste(€)
Personal	11.730
Hardware	24,69
Licencias	34,12
Energía	49,5
Coste total estimado	**11.838,31**

Recuperación de la inversión

Una vez obtenido una cantidad estimada de lo que supondría desarrollar la aplicación es necesario determinar diferentes fuentes de ingresos con las que se podría interactuar para no realizar esfuerzos en vano. En este caso se diferencian tres posibles fuentes de ingresos: publicidad, micropagos y aplicaciones de pago.

- **Publicidad**

También denominada *"**In-App Adversiting**"* es algo muy común en aplicaciones gratuitas las cuales quieren monetizar su contenido. Se basan en mostrar mensajes publicitarios en la aplicación y el desarrollador percibirá una cantidad de ingresos cada vez que los usuarios cliquen sobre el mensaje publicitario. Para indicar las ganancias que se estiman es capaz de producir la aplicación se introduce un concepto nuevo, el eCPM (Costo por Mil Efectivo). En él se valoran el número de impresiones de la publicidad, el número de clicks que se han hecho y con ello se calculan los beneficios con la siguiente fórmula:

(Beneficios/Nº de impresiones)*1.000 = eCPM

De tal manera que si el eCPM es de 4€, significa que por cada 1000 veces que se muestra publicidad se ganan 4€. Existen plataformas que te incluyen la publicidad en tu aplicación, siendo las más usadas para el OS Android AdMob y Mobfox. Solo se necesita una cuenta Google Wallet para ligarla a AdMob, y cada vez que un usuario interactúe con la publicidad reportará una cantidad que oscila entre 0,001 y 0,003€, siendo necesario un volumen considerable de impresiones el hacerla rentable (eCPM de entre 1€ y 3€). A continuación se muestra el cálculo del número de impresiones necesarias suponiendo que publicitamos la aplicación con AdMob:

Tabla 10: Beneficios de publicidad

Beneficios publicidad
Si 1 impresión reporta de media 0,002€. ¿Cuántas impresiones de publicidad son necesarias para obtener un beneficio de 11.838,31€? 0,002€ ---------- 1 impresión 11.838,31€ ---------- x impresiones **x = 11.838,31 * 1 / 0,002 = 5.919.155 impresiones necesarias.**

- **Micropagos**

Son aplicaciones de descarga gratuita, solo que tienen funcionalidades que están limitadas dentro de la aplicación y por las que el usuario tendrá que pagar si quiere desbloquearlas. Cogiendo esa idea y adaptándolo a esta aplicación, se podrían realizar tablas de entrenamiento realizadas por gente experta (por ejemplo, entrenadores personales) e incluirlas en la aplicación bajo un distintivo y a su vez, pedir una cantidad de dinero por ellas si se quieren utilizar. Por poner un ejemplo, se podrían vender plantillas de entrenamiento por 0,25€. El número de plantillas a vender para comenzar a obtener beneficios sería de:

Tabla 11: Beneficios de micropagos

Beneficios micropagos
Vendemos cada plantilla a 0,25€. ¿Número de plantillas que se necesitan vender para hacer frente a los gastos totales de 11.838,31€? 0,25€ ---------- 1 venta 11.838,31€ ---------- x ventas **x = 11.838,31 * 1 / 0,25 = 47.354 ventas necesarias.**

- **Aplicación de pago**

Aplicaciones que se venden en plataformas que las promocionan (Google Play, AppStore, etc.) por un precio que se fije. Para fijar el precio se han comparado otras aplicaciones en diversas plataformas y su precio varía enormemente: gratuitas, 1.99$, 4.99$, 11.99$, etc. Finalmente, dado que otras aplicaciones son capaces de conectarse a redes sociales pero no disponen de la capacidad de indicar los inicios/fines de entrenamiento mediante sonidos se ha establecido un precio razonable de 2,65€ para la aplicación. Hay que tener en cuenta que a veces es mejor disponer de una aplicación a un precio inferior para que el rango de compradores potenciales sea mayor. A continuación se muestra el cálculo del número de veces que tiene que ser comprada la aplicación para que la aplicación comience a generar beneficios:

Tabla 12: Beneficios como aplicación de pago

Beneficios como aplicación de pago
Es necesario comprar licencia de desarrollador de Google, lo cual incrementa el coste otros 25€ adicionales situándose en 11.713,31€ totales. Google Play obtiene un 30% de las ganancias totales al aportar la plataforma de distribución. Precio de venta: 2,65€. Como sólo vamos a obtener un 70% de los beneficios podríamos calcular las ganancias de la siguiente forma: Nº de ventas*2,65 * 0,7 = 11.713,31€. **Nº mínimo de ventas = 6.315 copias vendidas.**

Aclarar que al lanzar la aplicación se consideran que los tres tipos explicados anteriormente son excluyentes: una aplicación de pago no tiene publicidad ni micropagos ya que resultaría invasiva y tendría mala reputación en las plataformas de descarga porque el usuario paga también por evitar esos mensajes.

Una vez estudiadas las tres posibilidades se propone lanzar la aplicación como de pago o de publicidad indistintamente ya que son buenas opciones. El número de personas que realiza ejercicios y busca asesoramiento es cada vez mayor, ya que por temas económicos actualmente muy pocos pueden permitirse entrenadores personales. Siendo 5.900.000 clicks en publicidad un número enorme, hay que pensar que un mismo usuario puede reportar un número considerable de clicks, lo cual se traduce en ganancias:

1 usuario hace 5 clicks diarios = 1.825clicks/año = 3,65€ beneficios/año.

11.838,31€ / 3,65€ por persona = 3.244 personas.

De esta manera hemos calculado que con 3.244 personas haciendo una media de 5 clicks diarios, en un año se habrían cubierto los gastos.

La otra opción resulta un tanto más difícil de realizarse, ya que existen diferentes aplicaciones de entrenamiento muy competitivas aunque no se ha podido encontrar ninguna que marque los plazos de entrenamiento. Así pues, 2,65€ no es un precio muy descabellado sobre todo teniendo en cuenta otras opciones las cuales superaban con creces ese valor.

3. ANTECEDENTES

Aunque cuando nació la idea de crear una aplicación de entrenamientos no se tuvieron en cuenta otras aplicaciones para no condicionar los objetivos de la misma, finalmente se interactuó con otras aplicaciones para ver qué funcionalidades aportaban y así compararlas.

Dicha comparación tenía como objetivo principal detectar posibles mejoras en cuanto a alguna funcionalidad y/o la disposición de los elementos en la interfaz principalmente, para así poder mostrar la información de la forma más eficiente posible.

Existe una gran variedad de aplicaciones que realizan funciones similares a esta y cuyas características más notables son:

- Rutinas establecidas.
- Personalización de rutinas de ejercicio.
- Cronómetro para los tiempos de descanso.
- Calendario para ejercicios.
- Énfasis en las dietas (apartado desechado que no obstante, puede ser una posible mejora de futuro).

Las aplicaciones más destacables que se han encontrado son las siguientes:

JEFIT

Con más de 1 millón de descargas en Google Play, es una aplicación disponible igualmente para iOS, en la que generar tus plantillas de entrenamiento y asignarlas a diferentes días no llevará más de 5 minutos. Con una buena cantidad de ejercicios disponibles distribuidos por grupos musculares, permite la activación de un cronómetro que avisa cuando se finaliza la cuenta atrás, además tiene soporte para hacer publicaciones en redes sociales.

- Pros:
  - Intuitiva.
  - Personalización de rutinas.
  - Calendario de ejercicios.
  - Interfaz minimalista.
- Contras:
  - Solo en inglés.
  - No permite modificar los tiempos en el cronómetro (Ver Ilustración 14, Ilustración 15, Ilustración 16, Ilustración 17).
  - Publicidad.
  - Datos guardados en local.
  - Registro obligatorio.

Ilustración 14: Menú JEFIT

Ilustración 15: Rutina de ejercicios JEFIT

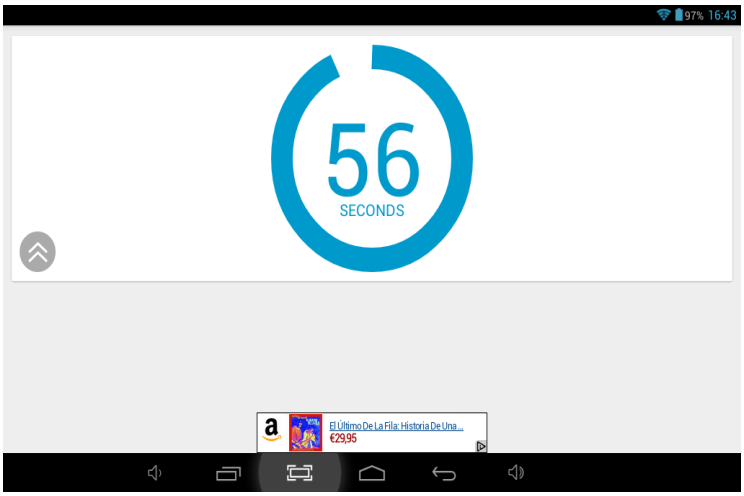


Ilustración 16: Cronómetro JEFIT

Virtuagym

Aplicación disponible para iOS y Android con más de 5 millones de descargas para esta última plataforma. Al inicio permite al usuario comenzar a usar la aplicación sin registrarse y así acceder a cualquiera de las 7 rutinas de entrenamiento que existen por defecto. Si el usuario

se registra accede a múltiples plantillas prediseñadas. La versión gratuita solo permite crear una rutina, teniendo que pagar 6,99$.

- Pros:
  - Personalización de rutinas.
  - Registro opcional.
  - Soporte para nutrición.
  - Asistente de repeticiones por voz.
  - Previsualización del ejercicio.
- Contras:
  - Tiempos de cronómetro fijos (no modificables).
  - Interfaz ligeramente compleja.
  - No permite subir tu plantilla a la red.

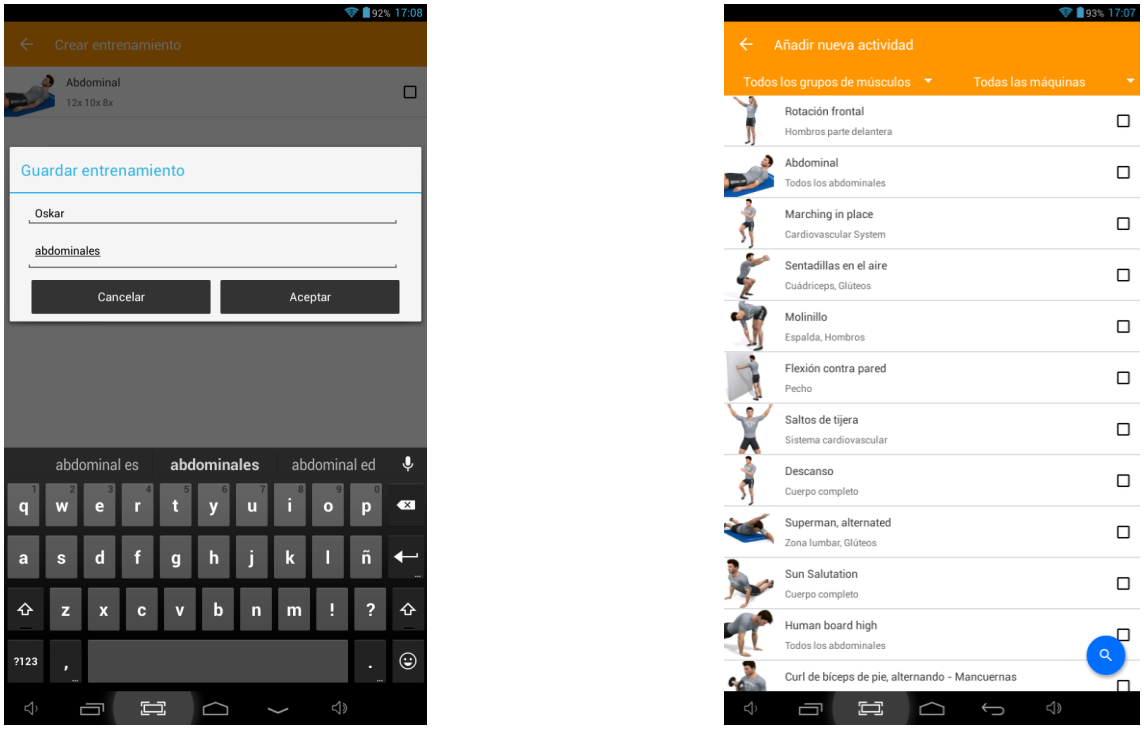


Ilustración 17: Proceso de creación de rutina de ejercicios en VirtuaGym

GYMG Fitness

Se dio con esta aplicación buscando en foros de aplicaciones sobre entrenamiento y la verdad, sorprende ver la cantidad de descargas que dice Google Play haber tenido, más de 1.000.000. La sorpresa viene cuando contiene una pobre personalización de las rutinas por no decir que es inexistente. Ofrece un conjunto de rutinas preestablecidas las cuales se muestran en pantalla de manera eficaz, sabiendo de un solo vistazo qué ejercicios debes realizar durante el entrenamiento, pero no ofrece vídeos de cómo realizarlos si no es en el apartado específico de ejercicios, cosa que durante los entrenamientos sí ofrecen sus competidoras. La cantidad de

publicidad que contiene es alarmante, siendo muy frecuente clicar involuntariamente en un anuncio ya que salen espontáneamente y cada muy poco tiempo.

- Pros:
  - No hace falta registro.
  - Fácil de utilizar.
  - Supuestas rutinas de musculación de famosos.
- Contras:
  - Publicidad excesiva.
  - No permite crear plantillas personalizadas.
  - Navegación limitada.

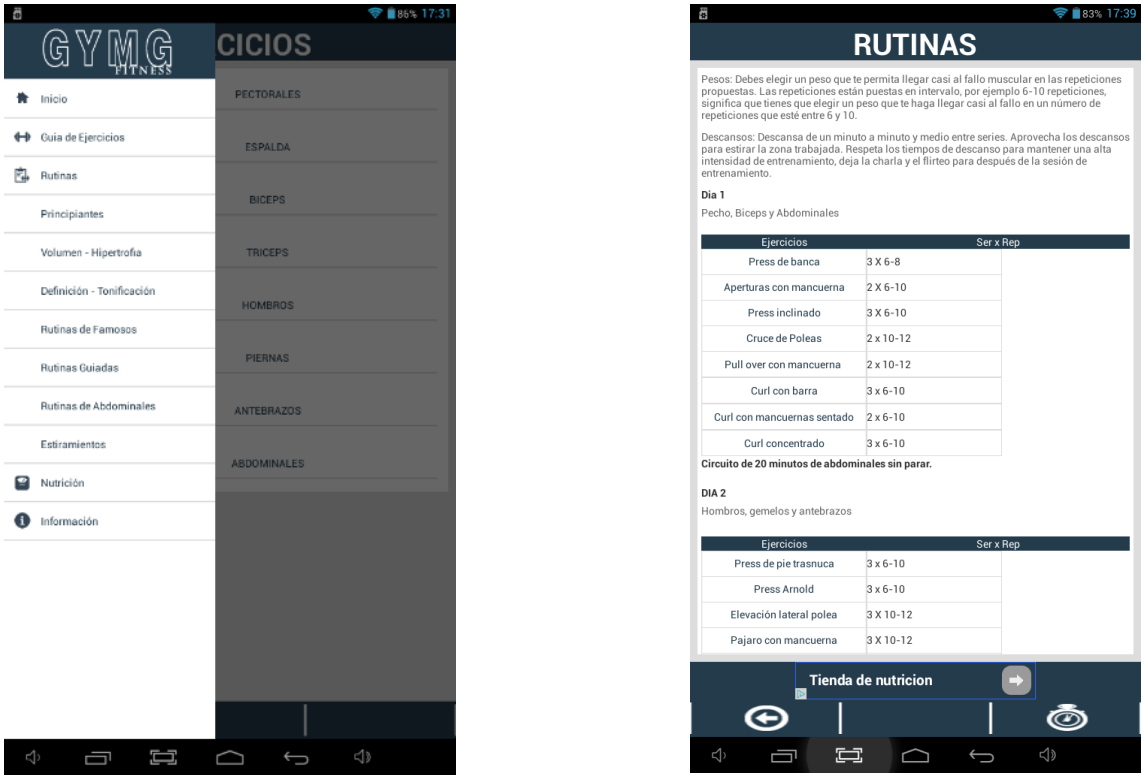


Ilustración 18: Menú principal y vista de rutinas en GYMG

Abdominales/Flexiones en 8 minutos

Principal inspiración a la hora de diseñar la aplicación de lo que se hablará al finalizar el apartado de antecedentes, existiendo al menos una aplicación para ejercicios de abdominales y otra para pectorales. Ofrece la posibilidad de generar un calendario de ejercicio en el cual se puede configurar la aplicación para que notifique al usuario que es la hora de realizar el entrenamiento. Se puede descargar el ejercicio o visualizarlo en línea. Su funcionamiento se basa principalmente en la reproducción de un vídeo en el que se muestra al ejercicio a realizar durante todo el entrenamiento. Marca tanto los ritmos como los segundos de descanso. El principal inconveniente es no poder generar tus plantillas y no poderlas almacenar online.

- Pros:
  - Sin necesidad de registro.
  - Entrenamiento totalmente guiado.
- Contras:
  - Publicidad.
  - No permite crear plantillas personalizadas.
  - Funcionalidades limitadas en la versión gratuita.

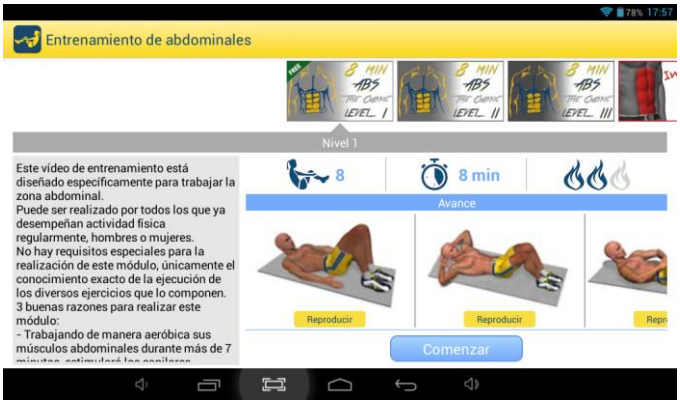


Ilustración 19: Menú "… en 8 minutos"

Ilustración 20: Ejercicio "… en 8 minutos"

Ilustración 21: Descanso "… en 8 minutos"

Para finalizar este apartado hay que mencionar que la inspiración para realizar la aplicación nunca vino sabiendo que existía otra que hiciese lo mismo y si existe alguna es pura coincidencia. Debido a entrenamientos que se realizaron utilizando el vídeo del entrenamiento para abdominales/flexiones se pensó *"¿por qué no generar una aplicación en la que se pueda configurar una plantilla y un cronómetro lleve los tiempos de descanso para así no perder tiempo configurándolos?".* De hecho no existe una que lo haga aún, ya que las que permiten configurar la plantilla y marcan los tiempos, éstos permanecen intocables y las repeticiones son estables, pudiendo sólo añadir y quitar ejercicios, no modificando el número de repeticiones o el tiempo de descanso. Además las cuatro aplicaciones aquí expuestas (las que más se parecen a *Time To Train*) guardan los datos en local, perdiéndose al cambiar el móvil. Por ello esta aplicación es única, ya que la hace diferente del resto de aplicaciones conocidas en su versión gratuita.

4. CAPTURA DE REQUISITOS

En este apartado se detallarán aquellos datos de gran importancia gracias a los cuales se hará un análisis inicial de cómo se va a enfocar el desarrollo del proyecto. El primer apartado recoge los casos de uso así como los actores identificados en este sistema. El segundo muestra el modelo del dominio diseñado, el cual servirá de gran ayuda para diseñar un sistema de bases de datos relacional que permita almacenar los datos requeridos por la aplicación.

4.1. Modelo de casos de uso

Para dividir más cómodamente los casos de uso se ha realizado una división entre aquellos casos de uso relacionados con la gestión de ejercicios y cuentas en el sistema web y otro para todos los casos de uso que se efectúen desde la aplicación móvil.

4.1.1. Casos de uso sistema Web

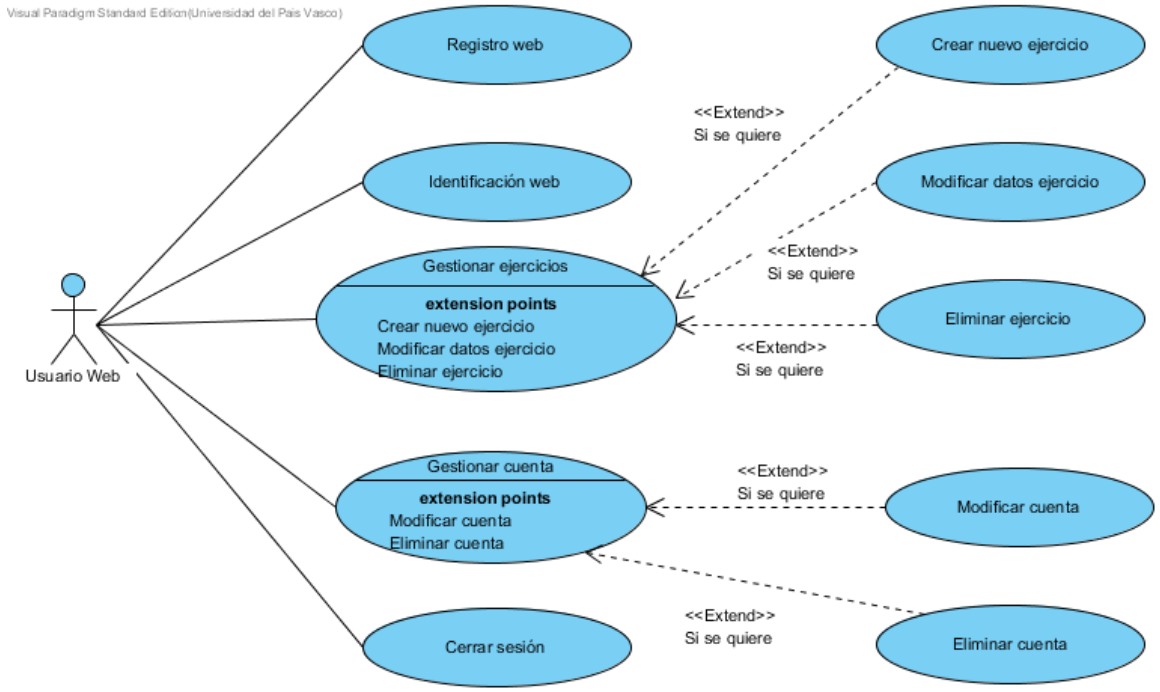


Ilustración 22: Casos de uso: Usuario web

Registro web

Permite al usuario crear una cuenta para acceder al sistema mediante el proceso de inicio de sesión. Dicha cuenta podrá utilizarse para acceder tanto a la aplicación móvil como a la página web, aunque únicamente desde esta última se podrán realizar acciones como la creación de nuevos ejercicios o eliminación de cuenta.

Identificación web

Permite que cualquier usuario que se haya registrado previamente pueda acceder al sistema para realizar gestiones relacionadas con ejercicios y/o su cuenta.

Gestionar ejercicios

Permite que el usuario acceda al área desde la cual puede gestionar ejercicios, ya sea creando nuevos, modificándolos o borrándolos del sistema.

Crear nuevo ejercicio

Da de alta un nuevo tipo de ejercicio en el sistema indicando sus datos, gif del movimiento y una descripción para que los usuarios de la aplicación móvil puedan incluirlo en sus plantillas.

Modificar ejercicio

Permite que el usuario visualice los ejercicios que él mismo haya generado y así realizar modificaciones sobre ellos.

Eliminar ejercicio

Permite que el usuario pueda eliminar del sistema los ejercicios creados por él siempre y cuando no los esté usando ningún usuario en sus plantillas.

Gestionar cuenta

Permite al usuario acceder al área de gestión de cuenta desde la cual podrá decidir si realizar modificaciones en la misma o eliminarla.

Modificar cuenta

El usuario podrá realizar gestiones en su cuenta de usuario al permitirle modificar los datos de la misma.

Eliminar cuenta

Permite que el usuario elimine sus datos del sistema al dar de baja su cuenta.

Cerrar sesión

Cierra la sesión activa en la web y deja el sistema preparado para iniciar sesión con otra cuenta.

4.1.2. Casos de uso aplicación móvil

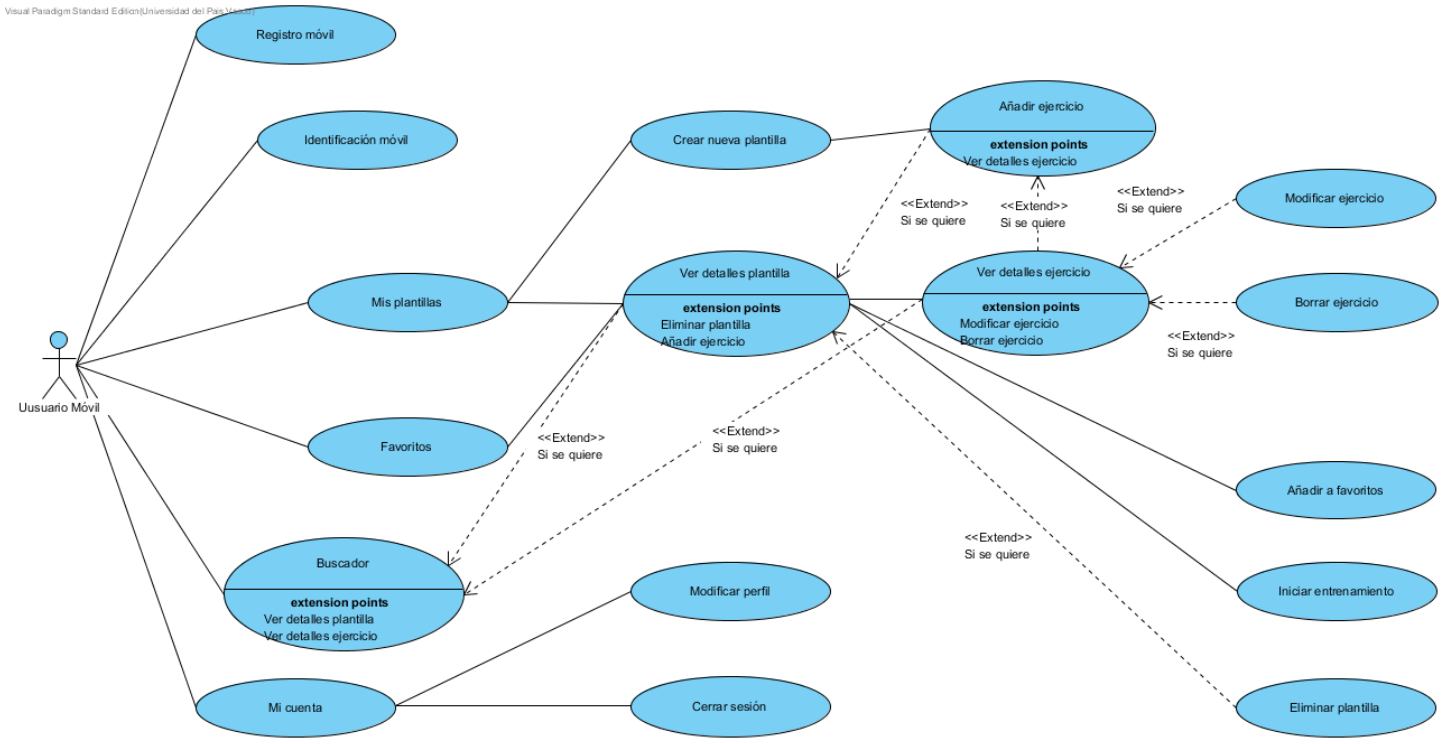


Ilustración 23: Casos de uso: Usuario móvil

Registro móvil

Permite que los usuarios del dispositivo móvil creen una cuenta con la que posteriormente puedan acceder a la aplicación y utilizarla.

Identificación móvil

Permite al usuario móvil iniciar sesión en su aplicación móvil siempre y cuando ya haya registrado una cuenta vía web o dispositivo móvil. Tras esto el usuario podrá utilizar la aplicación tanto para entrenar y visualizar datos online como para crear sus propios planes de entrenamiento y que otros usuarios los puedan visualizar.

Mis plantillas

Permite que el usuario que haya iniciado sesión acceda al menú de aquellos planes de entrenamiento que él mismo haya definido, sobre los cuales podrá realizar acciones como modificación o eliminación.

Crear nueva plantilla

Permite que el usuario que haya iniciado sesión comience la creación de una nueva plantilla de entrenamiento añadiéndole nuevos ejercicios a la misma. Para salvar los datos el usuario añadirá una serie de *tags*, una descripción y el nombre de la plantilla con la cual se guardará en el sistema.

Añadir ejercicio

Permite que el usuario añada un nuevo ejercicio a la plantilla actual siempre y cuando el usuario esté identificado y la plantilla en la que añadirá el ejercicio haya sido diseñada por él.

Ver detalles plantilla

Permite que el usuario, una vez haya iniciado sesión en la aplicación, pueda ver detalles de la plantilla, lo cual incluye el listado de ejercicios que la componen, opciones como añadir a favoritos, o incluso realizar modificaciones/eliminación de la misma si esta fuera de su propiedad.

Ver detalles ejercicio

Permite al usuario identificado que visualice las características que definen el ejercicio que ha seleccionado. Entre ellas el nombre, la imagen con la posición a adoptar y la descripción del mismo.

Modificar ejercicio

Permite al usuario identificado, tras consultar los detalles de una de sus plantillas, que modifique el número de series, repeticiones o el tiempo de descanso del ejercicio a modificar.

Borrar ejercicio

Permite al usuario que, una vez identificado y tras consultar los detalles de una de sus plantillas, elimine el ejercicio de la plantilla actual.

Eliminar plantilla

Permite al usuario que, una vez identificado pueda eliminar la plantilla actual.

Añadir a favoritos

Permite al usuario que, una vez identificado, pueda añadir a favoritos la plantilla actual. En caso de estar visualizando las plantillas de “favoritos”, esta sería eliminada de la lista.

Iniciar entrenamiento

Permite al usuario, una vez identificado, pueda comenzar el entrenamiento de la plantilla actual.

Favoritos

Tras haber iniciado sesión, permite que el usuario acceda a una lista de aquellas listas que haya seleccionado como favorita. Dicha lista podrá contener tanto plantillas de entrenamiento suyas como de otras personas, y al seleccionarlas podrá acceder a sus detalles así como a diferentes casos de uso en común desde otras áreas.

Buscador

Tras haber iniciado sesión, permite que el usuario pueda realizar búsquedas relacionadas con plantillas y con ejercicios y así poder visualizar la información de los ejercicios en cualquier momento o bien realizar entrenamientos diseñados por otras personas.

Mi cuenta

Permite al usuario, una vez se ha identificado, que pueda acceder al área de gestión de cuentas.

Modificar perfil

Permite al usuario identificado que pueda realizar modificaciones en sus datos personales.

Cerrar sesión

Cierra la sesión de usuario actual y elimina los datos que se almacenan en la parte local del dispositivo móvil para que de esta manera se pueda iniciar sesión con una cuenta diferente y dicho proceso no sea automático.

4.2. Modelo del dominio

El siguiente modelo del dominio muestra qué datos van a ser almacenados en la base de datos y las relaciones que existen entre las entidades que han sido identificadas:

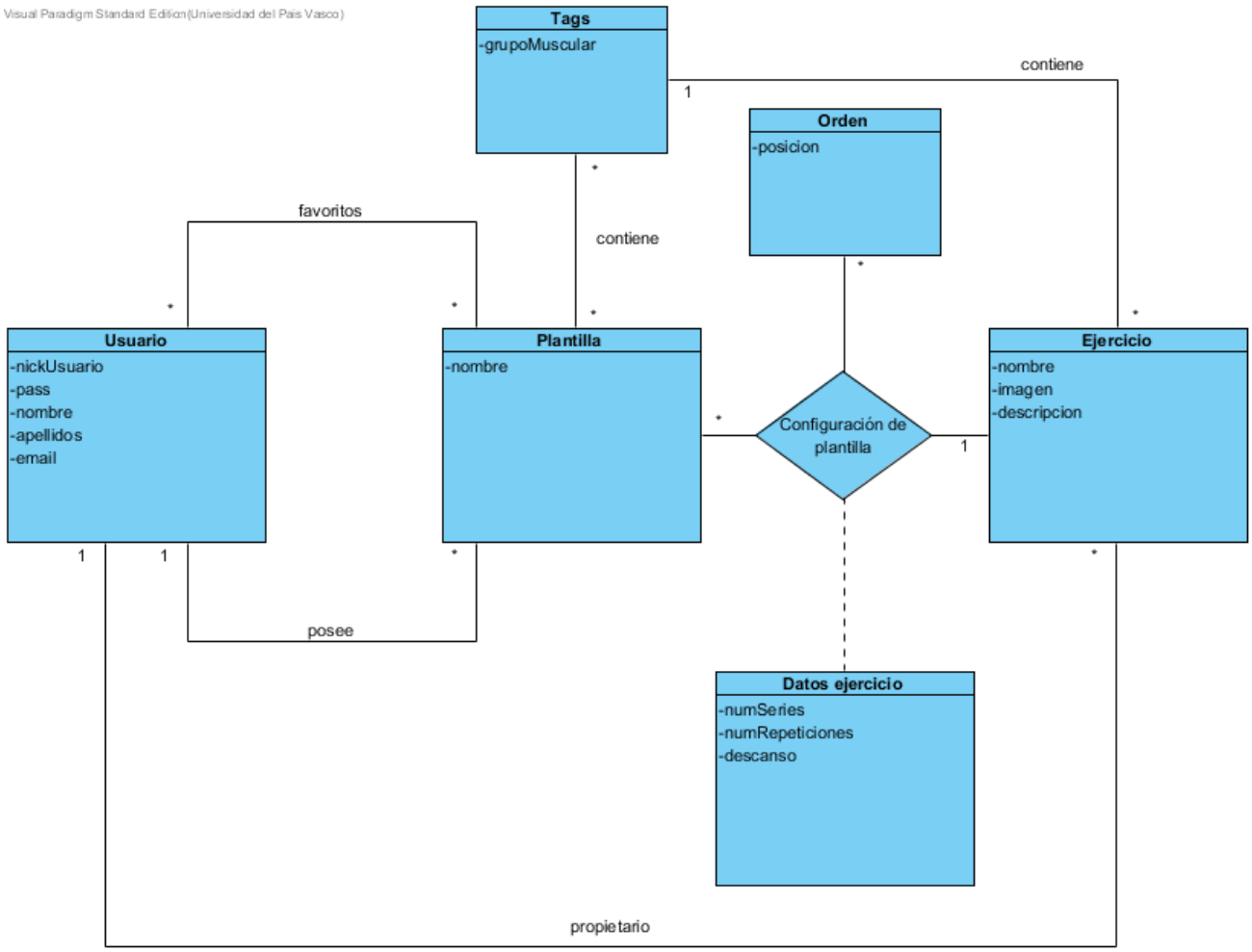


Ilustración 24: Modelo del dominio DB

El siguiente modelo de dominio representa la información que se almacena en el dispositivo (local) para poder realizar peticiones.

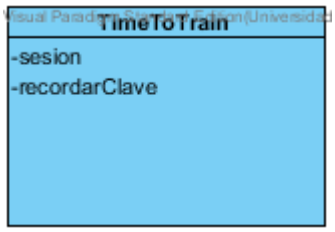


Ilustración 25: Modelo del dominio Local

A continuación se procederá a la explicación del modelo del dominio anterior, para ello se explicará en tres fases: entidades, asociaciones y relaciones.

Entidades

Para poder almacenar y organizar toda la información que va a requerir *Time To Train* se han identificado las siguientes entidades:

Usuario

Almacenaremos los datos que proporcionen los usuarios en el formulario de registro. Principalmente utilizaremos el nick de usuario como identificador único junto con la contraseña para realizar identificaciones, aunque el resto de datos se almacenarán por si en un futuro se decide ampliar la aplicación (mensajes al e-mail).

Plantilla

Representa un plan de entrenamiento generado por un usuario y cuyo principal identificador será el nombre que el usuario le dé a la misma.

Tags

Se identifica por un nombre el cual sirve para describir una de las cinco zonas del cuerpo que se ejercitan al realizar ejercicios durante una sesión.

Ejercicio

Unidad mínima de la que constará un plan de entrenamiento. En ella se describen los atributos básicos de los que consta el ejercicio: nombre, url de la imagen o gif a mostrar y su descripción.

Orden

Contiene la posición que los ejercicios guardan dentro de las plantillas.

Asociaciones

Las asociaciones contienen atributos que no son de una entidad como tal, sino de la combinación de varias de ellas.

Datos ejercicio

Contiene aquellos datos los cuales no pertenecen a la entidad "ejercicio" al variar de una plantilla a otra. Dichos atributos son: el número de series, el número de repeticiones a realizar y el descanso que hará al acabar cada serie de ejercicios.

Relaciones

Sirven para tener referencias entre entidades. Qué usuario ha creado qué tabla o ejercicio y relaciones similares. A continuación se explica la cardinalidad de ellas.

Usuario - Plantilla:

- **Favoritos *…*:** Sirve para almacenar las plantillas que tienen en favoritos los usuarios. Cada usuario puede tener ninguna o múltiples plantillas agregadas en favoritos. Por el otro lado, cada plantilla puede formar parte de la lista de favoritos de ninguno o varios usuarios.
- **Creación 1…*:** Con ello se almacenará el creador de cada una de las plantillas. Un usuario puede utilizar la aplicación para generar ninguna o muchas plantillas. Sin embargo cada plantilla sólo puede haber sido generada por un único usuario.

Usuario - Ejercicio:

- **Creación 1…*:** Esta relación determina el creador de cada ejercicio. Relación similar a la de usuario – plantilla. Cada ejercicio es creado por un único usuario, pero cada usuario puede generar muchos ejercicios.

Plantilla- Tags:

- **Contiene *…*:** Determina el número de Tags que puede contener una plantilla, ya que el usuario podrá, durante la creación de la misma, indicar los grupos musculares que se ejercitan con la plantilla. De esta manera al realizar búsquedas de plantillas los usuarios verán los tags sin necesidad de comprobar cada uno de los ejercicios que contiene. La cardinalidad es * en ambos lados ya que un mismo tag puede formar parte de múltiples plantillas y cada plantilla puede tener varios tags.

Ejercicio- Tags:

- **Tiene *…1:** Al generar un ejercicio nuevo los usuarios indican el grupo muscular al que principalmente afecta, mientras que un mismo tag puede pertenecer o identificar múltiples ejercicios.

Plantilla – Orden – Ejercicio – Datos ejercicio:

- **Relación *…*…1:** A la hora de realizar el modelo de dominio se pensó que por lo general, cuando se realiza un entrenamiento, una vez acabas una serie de ejercicios no la vuelves a repetir más adelante. No obstante el permitir que aparezca más veces le añade diversidad y hace que sea una aplicación más permisiva. Por ello existe esta relación triple con un atributo. Al generar su tabla de ejercicios el usuario podrá configurar varios de los parámetros de cada uno de ellos, por ello es necesario separar estos atributos que varían de la entidad "ejercicio" y añadirlos como un atributo de la relación en lo que se ha denominado "datos ejercicio". De esta manera se almacenará qué ejercicio forma parte de qué plantilla, el orden en que se realizará y los atributos que se acaban de mencionar. La cardinalidad de la relación es "*…1" al poder, una misma configuración de ejercicio, aparecer varias veces dentro de una misma plantilla. Sin embargo en una plantilla en una posición concreta solo puede aparecer un ejercicio.

5. ANÁLISIS Y DISEÑO

En este apartado se mostrará el diseño que tendrá la base de datos utilizada para albergar toda la información así como las clases Java utilizadas en la aplicación móvil.

5.1. Diagrama de base de datos

En este apartado se analizará el diagrama de bases de datos relacional que se ha obtenido gracias a la transformación del modelo de dominio, y, gracias a la cual, se va a almacenar toda la información que manejará *Time To Train*.

Aunque la aplicación se puede dividir en parte web y parte móvil, se utilizará la misma base de datos para dar servicio a estas dos plataformas. Esto es así porque aunque sólo comparten los procesos de registro, identificación y gestión de cuentas, las acciones que se pueden realizar en la parte web (creación y gestión de ejercicios) tienen su repercusión en la parte móvil (creación de plantillas utilizando esos ejercicios).

Como se puede observar, tanto la entidad plantilla como el ejercicio tienen un atributo "nombre", siendo en una de ellas la clave primaria y en la otra entidad un atributo más. La razón de esta elección es que un ejercicio denominado, por ejemplo, curl de bíceps, puede ser únicamente un ejercicio, por ello se ha tomado la decisión de que debía considerarse como la clave primaria, mientras que puede que varios usuarios llamen sus plantillas de la misma manera.

Se han tenido en cuenta otros aspectos, como que un usuario genere ejercicios absurdos y les provea de nombres de ejercicios "oficiales", pero se partirá de la idea de que en el lanzamiento de *Time To Train* contará con todos aquellos ejercicios que se consideren esenciales, (ya sea porque un grupo de gente de confianza estrene la app y generen los ejercicios básicos) de manera que sus nombres no puedan estar en uso anteriormente porque un usuario haya tomado sus nombres.

Finalmente se puede observar con claridad las tablas y las relaciones existentes entre ellas en la Ilustración 26 de la siguiente página.

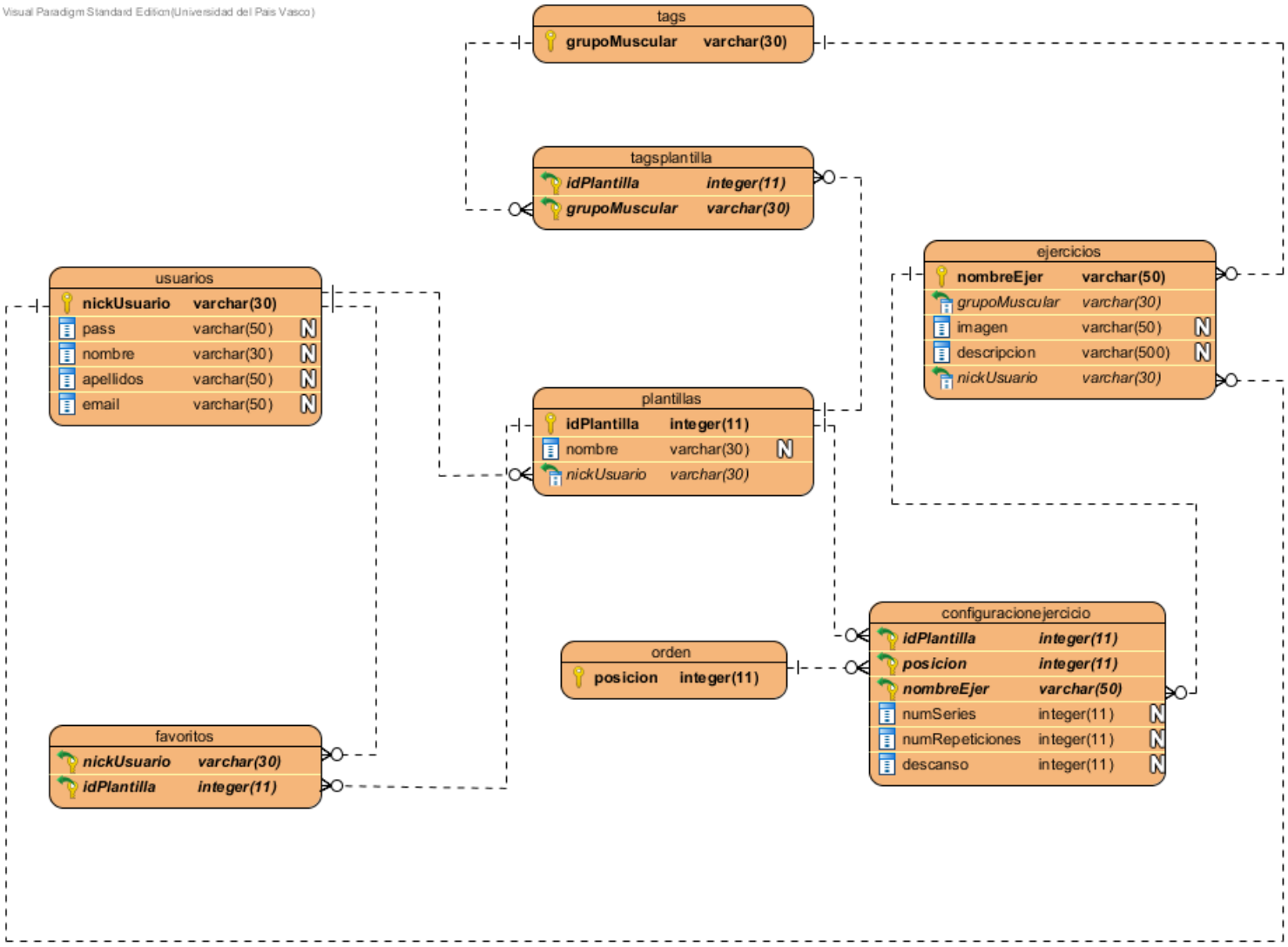


Ilustración 26: Diagrama de bases de datos

5.2. Diagrama clases

De cara a simplificar la comprensión del diagrama de clases se ha decidido dividir en tres diagramas menores con un código de colores para diferenciar los tipos de clases utilizados. El código es el siguiente:

Tabla 13: Diagrama de clases: Código de colores

Tipo de clase	Color
Modelo	
Adaptador / Modificación	
Fragment / DialogFragment	
AsyncTask	
Activity	

Además se ha preparado una introducción para las clases más importantes de Android y así proporcionar los conocimientos básicos al lector para poder entender qué son las Activities, Fragments y AsyncTasks que se van a mencionar a lo largo de los siguientes apartados.

AsyncTask

Es una clase de Android que permite la ejecución de tareas en un segundo plano, de manera que el hilo principal se mantenga lo menos ocupado posible y el dispositivo no se bloquee cuando el usuario desee interactuar con la aplicación. Por lo general las peticiones HTTP a un servidor se realizan por este método ya que requieren de cierto tiempo para establecer la conexión, enviar la petición y recibir la respuesta (inicio de sesión, cargar imágenes, etc).

Activity

Una Activity o actividad es un elemento de Android el cual enlaza una clase java, que gestiona un comportamiento y un fichero .xml que contiene los elementos gráficos a visualizar. Dicho de manera sencilla, una Activity es una de las pantallas del móvil en las se pueden observar elementos gráficos (.xml) con los que interactuar (java).

Android Studio provee de diferentes tipos de actividades (Google maps, navigation drawer, widget, actividad vacía, etc.) algunas de las cuales contienen pequeños segmentos de código ya implementado. A partir de Android 1.4 se generan dos archivos .xml del layout, repartiendo los elementos gráficos en ambos y resultando más lioso comprenderlo, motivo por el que las actividades de este proyecto se generaron como Empty Activity, implementándose desde cero y teniendo un único layout por actividad.

Una aplicación móvil es un sistema que permanece activo en el tiempo, a diferencia de los programas estándares se realizan durante la carrera, asignándose a cada Activity un ciclo de vida que se gestiona por eventos (Ver Ilustración 27). Dicho flujo de eventos comienza cuando se inicia la aplicación y que en función de cómo se gestione tendrá un comportamiento u otro.

Por ejemplo, si se recibe una llamada telefónica durante la ejecución de *Time To Train* se ejecutarán los métodos *.onPause()* y *.onStop()*. Al colgar y devolver el control a la aplicación se ejecutarán los métodos *.onRestart()*, *.onStart()* y *.onResume()*.

Dicho ciclo de vida es importante tenerlo en mente ya que ofrece puntos clave sobre los que implementar segmentos de código para gestionar el comportamiento deseado y no obtener errores durante el proceso. Por ejemplo, lo que muchos usuarios desconocen es que al girar el dispositivo las actividades se eliminan y vuelven a crear, lo que conlleva la pérdida de la información que haya en pantalla ya que el método *.onCreate()* vuelve a inicializar todo. Para evitar eso hay que implementar una serie de métodos que guarden cada uno de los elementos que contiene la actividad justo antes de eliminarla y restablecer el estado tres volver a crearla.

La gestión de todos esos elementos puede ser tremendamente complicada ya que es necesario restablecer el estado de no sólo los elementos visuales, sino también procesos en segundo plano, valores de preferencias, etc. Por ello son muchas las aplicaciones que no permiten girar el móvil, siendo esta la respuesta en muchos de esos casos.

Para un diseño más profesional Google recomienda que en el archivo .java asociado a una Activity se implementen únicamente los métodos para manejar la interfaz gráfica del mismo y no para otros fines. En caso de necesitar una petición de datos a un servidor remoto u operaciones que necesiten de un tiempo determinado para finalizarse es recomendable usar clases externas para ello. De esta manera se obtiene un código más comprensible de cara a futuras modificaciones al dividir la carga de código.

Por último mencionar que el OS Android puede eliminar actividades que estén en la pila de memoria (memory stack) cuando el OS necesite liberar recursos, por lo que hay que tener precaución con los datos que se quieren almacenar y en qué momento se debe hacer.

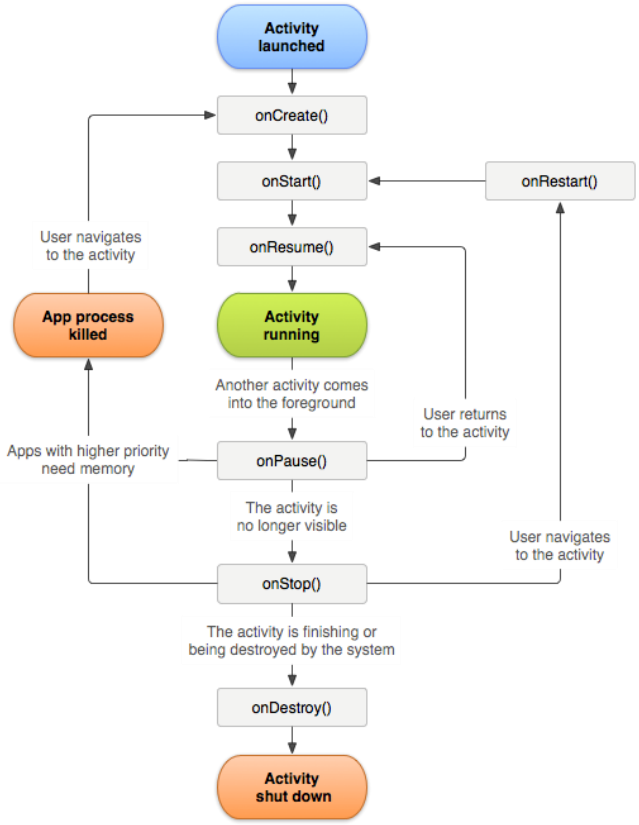


Ilustración 27: Ciclo de vida de una Activity

Fragments

Los fragments son elementos de diseño que, al igual que las actividades tienen su propio ciclo de vida (Ver Ilustración 28), comportamiento e interfaz gráfica, pero que independientemente no pueden ser utilizadas, ya que deben asociarse a una actividad.

Su aparición se produjo con la llegada de la API de nivel 11 (Android 3.0) y las tablets, ya que al poseer pantallas más grandes en las que visualizar el mismo contenido que un móvil resultaba en un mal acabado estético. Gracias a estos elementos se puede dividir la pantalla del dispositivo en tantas partes como fragments se coloquen, siendo lo más extendido el uso de un fragment para mostrar un listado de elementos en la mitad izquierda del Tablet y otro fragment en la mitad derecha para cargar y mostrar los datos del elemento seleccionado sin necesidad de generar una nueva actividad.

Esto es un punto a favor ya que es elimina la generación excesiva de actividades que pueden favorecer que el OS tenga que eliminar alguna para liberar recursos.

El otro punto positivo que tienen los fragments es la reutilización de código, ya que una vez implementada su clase se pueden utilizar en cualquier momento en cualquier clase, ahorrando mucho código y trabajo.

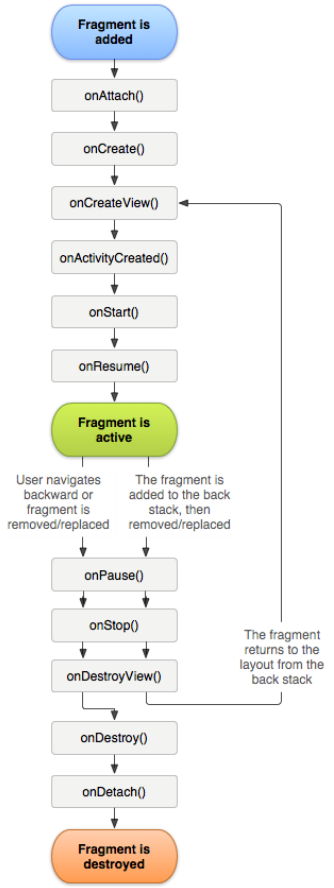


Ilustración 28: Ciclo de vida de Fragments

Diagrama de clases Activities + Modelo

El primer diagrama de clases (Ver Ilustración 29) muestra todas las actividades (interfaces o "pantallas") de las que consta la aplicación y se podría considerar una especie de mapa de navegación, indicando las áreas a las que se puede acceder dentro de la aplicación y su origen. También se han incluido las clases del modelo que se utilizan para realizar sus gestiones.

Para simplificar el nombre de las actividades con respecto al Workspace de trabajo se ha eliminado el sufijo "+Activity" que tienen por defecto.

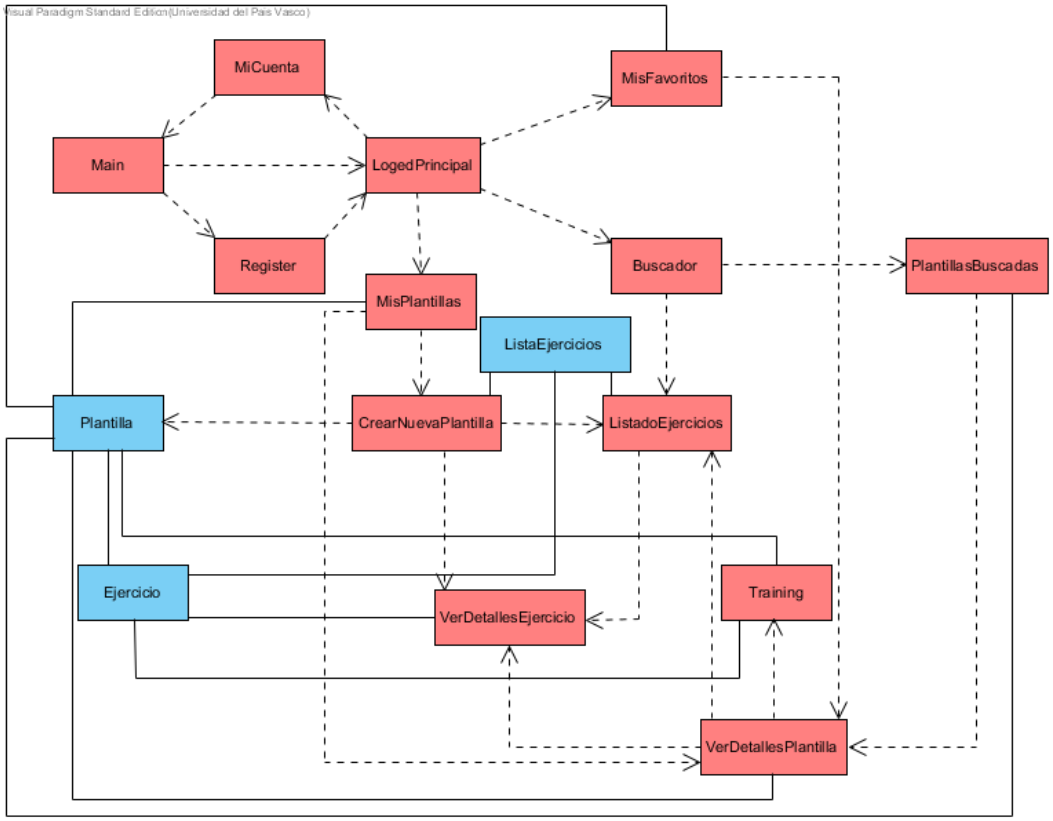


Ilustración 29: Diagrama de clases: Activities + Model Objects

Diagrama de clases Activities + DialogFragments

A lo largo de la aplicación se muestran diferentes ventanas de diálogo las cuales están pensadas para advertir al usuario de eventos de tipo error (conexión, confirmación, cierre de sesión, etc).

Aunque su número no es muy elevado el problema de incluir estas clases en el diagrama completo es que los dialogs de error de conexión se reutilizan todas las actividades que requieran hacer una petición al servidor, por lo cual el número de conexiones entre clases es muy grande, lo que complica su comprensión.

El segundo diagrama de clases (Ver Ilustración 30) muestra el uso de las ventanas de diálogo en cada actividad:

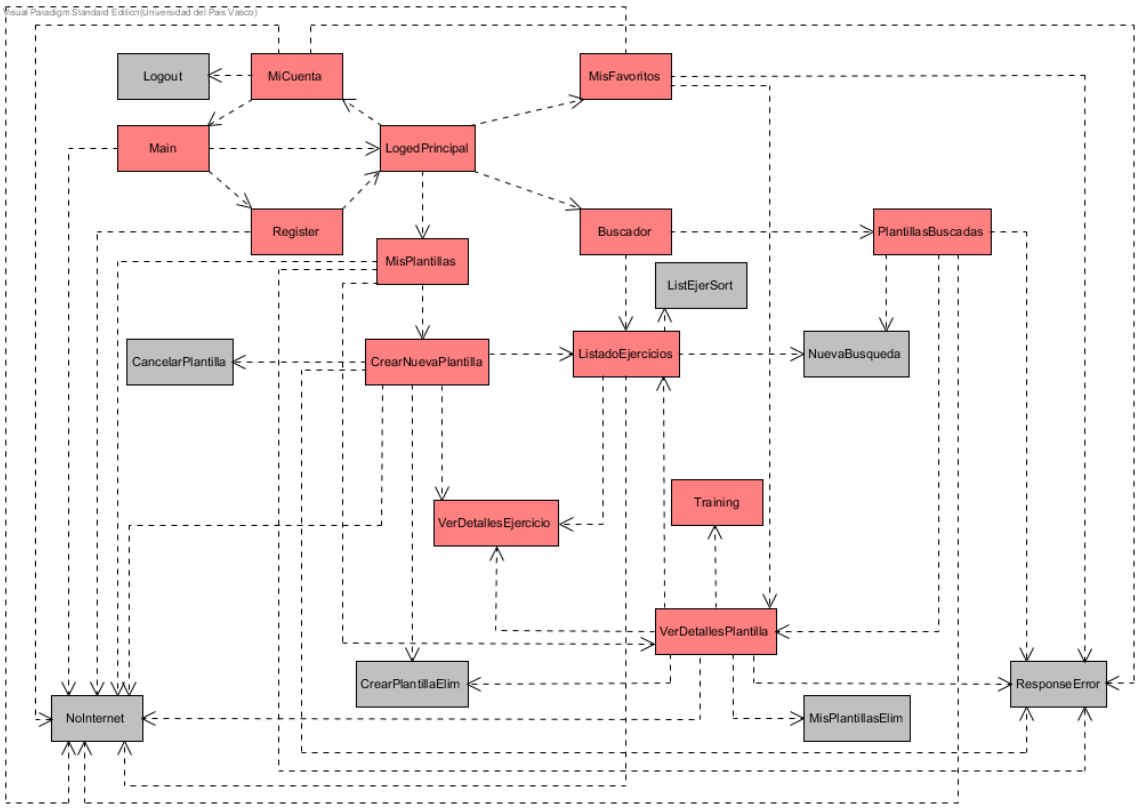


Ilustración 30: Diagrama de clases: Activities + DialogFragments

Diagrama de clases Activities + Adapters + Fragments + AsyncTasks

El último diagrama de clases (Ver Ilustración 31) representa el funcionamiento de la aplicación.

Las relaciones entre actividades (rojo) determinan las opciones posibles de navegación, mientras que las clases asíncronas (verde) realizan peticiones de datos al servidor para poder mostrarlas en elementos de la interfaz a través de adaptadores (amarillo).

Hay que destacar la inclusión de cuatro Fragments (gris), que si bien pertenecen al mismo grupo que los dialogFragments, se han incluido en este diagrama ya que en el caso anterior muestran diálogos informativos, mientras que aquí son porciones reutilizables de código utilizados en diferentes actividades para poder mostrar información dinámicamente sin tener que recargar la actividad actual.

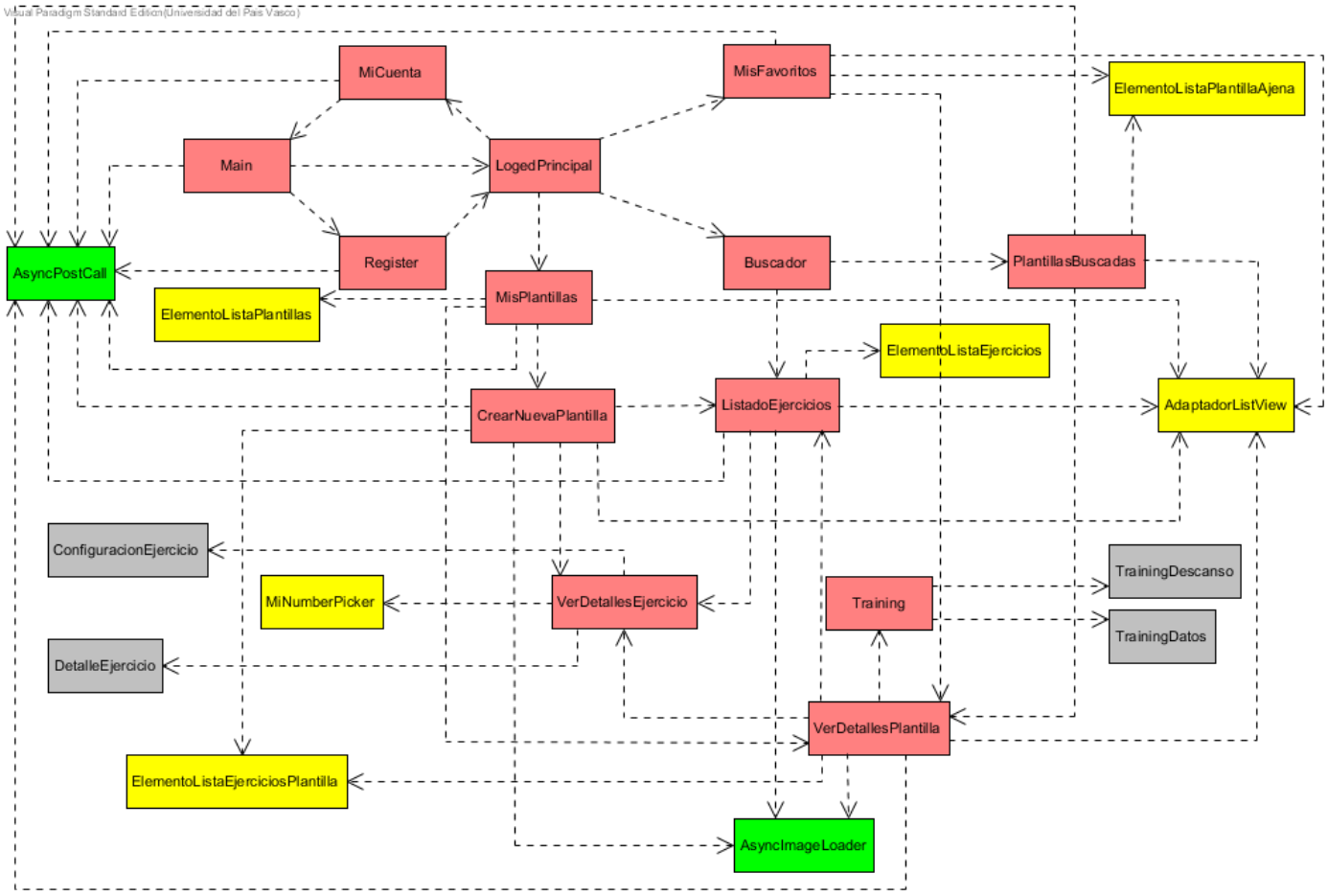


Ilustración 31: Diagrama de clases: Activities + Adapters + Fragments + AsyncTasks

5.3. Detalle de las clases

A continuación se explicará el funcionamiento de cada clase. Aunque las actividades son la parte esencial de cualquier aplicación se van a proceder a explicar en 4º lugar dado que existen métodos implementados que pertenecen a "interfaces" definidas en los primeros bloques a explicar.

5.3.1. AsyncTasks

Son clases que realizan tareas en segundo plano para no bloquear el hilo principal de procesamiento. Extienden de la clase AsyncTask.

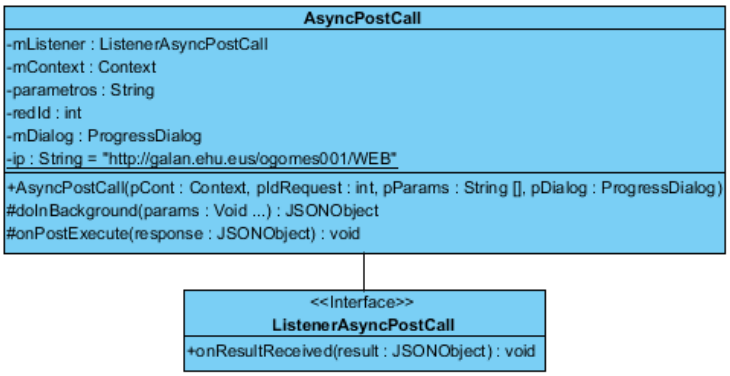


Ilustración 32: Detalle de las clases: AsyncTasks- AsyncPostCall

La clase *AsyncPostCall* realiza peticiones HTTP en segundo plano para no bloquear la ejecución de la aplicación mientras se espera la respuesta. Es capaz de detectar si el dispositivo tiene una conexión inalámbrica válida para poder realizar la petición. En función del parámetro de entrada pIdRequest se configura la llamada para que ésta se realice al fichero PHP oportuno alojado en el servidor http://galan.ehu.eus.

Además define la interface ListenerAsyncPostCall que deberá ser implementada en cada Actividad que haga uso de esta clase ya que de no ser definido un comportamiento para el método *onResultReceived()* de la interface se lanza una excepción avisando de que es imprescindible su implementación.

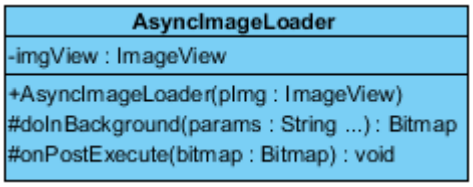


Ilustración 33: Detalle de las clases: AsyncTasks- AsyncImageLoader

Al igual que la clase anterior, la clase *AsyncImageLoader* se encarga de realizar una acción en segundo plano, en este caso la carga de imágenes en los listados de ejercicios o plantillas. Para poder realizar la carga es necesario instanciar la clase indicando el contenedor sobre el cual se va a cargar la imagen, el cual es un ImageView. A continuación el método *doInBackGround()* decodifica la dirección url y obtiene la imagen, insertándola en el contenedor.

5.3.2. DialogFragments

Son clases que definen ventanas de diálogo utilizadas para mostrar mensajes importantes al usuario y que por lo general tienen una repercusión al seleccionar alguna de las opciones disponibles. Extienden de la clase DialogFragment y no se instancian pasándole parámetros a la constructora, sino que se genera un paquete de datos el cual se le asigna al objeto dialogFragment una vez instanciado.

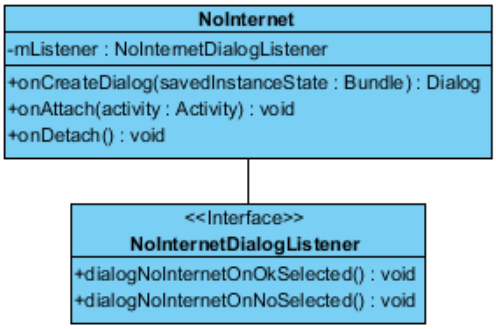


Ilustración 34: Detalle de las clases: DialogFragments- NoInternet

La clase *NoInternet* es la encargada de definir una ventana de diálogo que informa al usuario sobre la ausencia de conexiones inalámbricas en el dispositivo. Define la interface *NoInternetDialogListener* con dos métodos a implementar en las actividades que lo usen, las cuales pueden diferir en función de en qué actividad se lance este mensaje.

Por ejemplo en el inicio de sesión el denegar la activación de datos no tiene repercusión, pero el intentar visualizar el listado de ejercicios sin conexión devuelve a la actividad anterior, ya que sin conexión no hay ejercicios que visualizar.

En caso de aceptar la sugerencia de activación se abriría la configuración de conexiones inalámbricas sin tener que navegar manualmente.

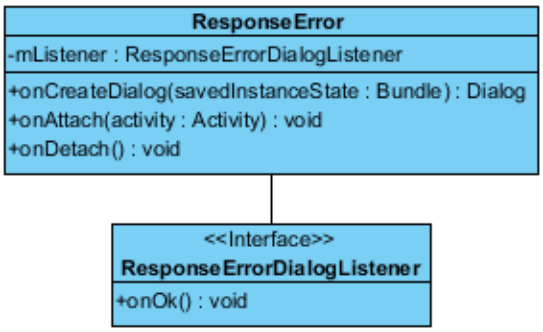


Ilustración 35: Detalle de las clases: DialogFragments- ResponseError

La clase *ResponseError* genera una ventana de diálogo para mostrar errores de conexión que se hayan podido producir al realizar la llamada asíncrona *AsyncPostCall*. Dichos errores pueden incluir la imposibilidad de realizar una conexión con el servidor, superar el tiempo de espera o problemas con la verificación de la identidad del usuario que realiza la petición.

La interface que define, *ResponseErrorDialogListener* tiene como objetivo asignar un comportamiento diferente a este fragment en cada actividad implementada, al igual que en el caso anterior.

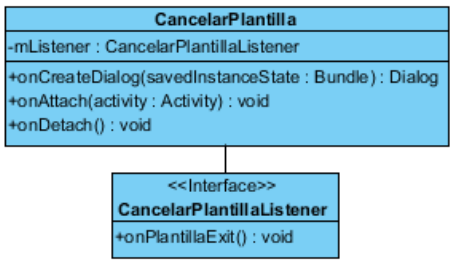


Ilustración 36: Detalle de las clases: DialogFragments- CancelarPlantilla

La clase *CancelarPlantilla* genera una ventana de diálogo que advierte al usuario de la pérdida de la plantilla si se encuentra en mitad de creación y decide salir de la actividad.

Para ello en la actividad CrearNuevaPlantilla se detecta si se han añadido ejercicios o bien si se le ha dado un nombre a la plantilla y, en caso de pulsar “back” se instancia esta clase para generar una ventana de diálogo. En caso de aceptar y pulsar “Salir” se perderá la plantilla al activarse el método de la interfaz *CancelarPlantillaListener*. En caso de pulsar “Cancelar” no

ocurrirá ningún cambio sea cual sea la actividad en la que se esté, motivo por el cual no se ha definido un método para tal caso en la interface.

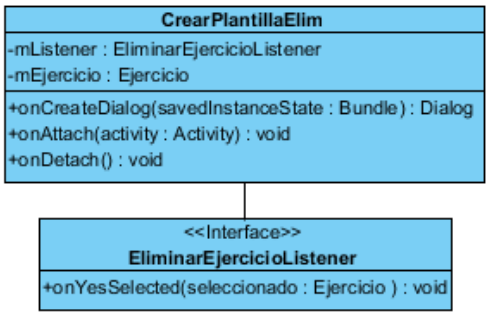


Ilustración 37: Detalle de las clases: DialogFragments- CrearPlantillaElim

La clase *CrearPlantillaElim* gestiona la eliminación de un ejercicio ya añadido a una plantilla ya sea durante la creación o durante la modificación de la plantilla una vez generada, para lo cual advierte al usuario de las consecuencias.

Define la interface *EliminarEjercicioListener* que, en caso de aceptar la eliminación, devuelve el objeto que ha sido seleccionado para su eliminación y proceder a la misma.

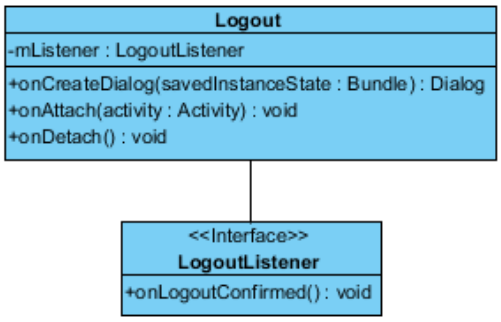


Ilustración 38: Detalle de las clases: DialogFragments- Logout

La clase *Logout* crea una ventana de diálogo para advertir al usuario que está a punto de cerrar la sesión actual. Es utilizada en el apartado de gestión de cuenta.

Para ello es necesario implementar su interface *LogoutListener* en la actividad *MiCuenta* con los métodos necesarios para indicarle al dispositivo que olvide las credenciales y redirija a la actividad principal en caso afirmativo.

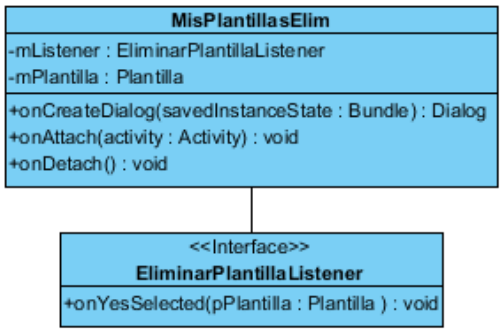


Ilustración 39: Detalle de las clases: DialogFragments- MisPlantillasElim

La clase *MisPlantillasElim* es utilizada para advertir al usuario de la eliminación de la plantilla actual si mientras se están visualizando los datos de la misma se pulsa sobre el icono de eliminación.

Tiene un funcionamiento casi idéntico al DialogFragment de eliminación de ejercicios, salvo que en este tipo de dialog se devuelve en el método *onYesSelected()* de la interface *EliminarPlantillaListener* aquella plantilla que ha sido seleccionada para su eliminación.

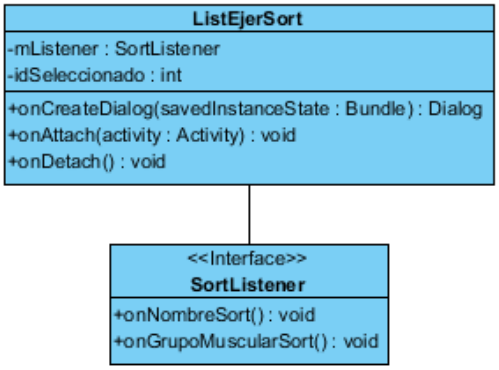


Ilustración 40: Detalle de las clases: DialogFragments- ListEjerSort

La clase *ListEjerSort* genera una ventana de diálogo durante la vista de ejercicios que da al usuario la posibilidad de ordenar el listado ya sea por grupo muscular o por nombre de ejercicio, para lo cual la actividad *ListadoEjercicios* debe implementar la interface *SortListener*.

Al mostrarse el diálogo éste mostrará el criterio de ordenación que se ha adoptado, siendo recordado cada vez que el criterio varíe.

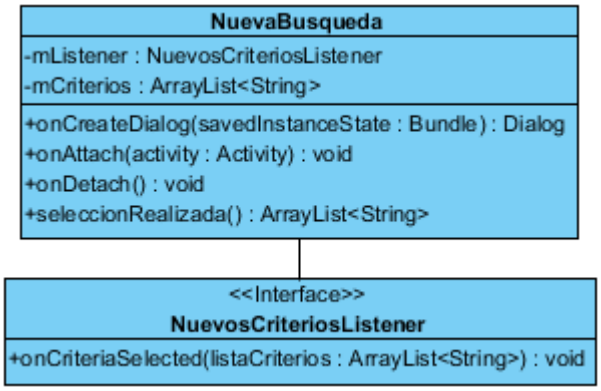


Ilustración 41: Detalle de las clases: DialogFragments- NuevaBusqueda

La clase *NuevaBusqueda* genera una ventana de diálogo con una vista similar al de una actividad. En ella se muestran los grupos musculares por los cuales se puede proceder a realizar una búsqueda y realizar un filtrado en función de los grupos seleccionados.

Al igual que en el dialog *ListEjerSort*, es capaz de recordar la última selección y mostrar activados en el layout aquellos grupos musculares por los cuáles se está realizando el filtrado.

Este DialogFragment contiene métodos propios para poder gestionar la selección realizada y activar el método de su interface si se ha realizado una selección previamente, para lo cual se determina el número de grupos musculares pulsados cargando el atributo mCriterios. Únicamente si mCriterios contiene al menos un grupo muscular se activará el método *onCriteriaSelected()* de la interface *NuevosCriteriosListener*.

5.3.3. Adapters

En este apartado se detallan dos de las clases que ofrecen Android que han sido modificadas para poder mostrar información de elementos utilizando una estética personalizada y no la que ofrece Android por defecto. Dichas clases son el *AdaptadorListView* y el *NumberPicker*.

Por defecto el ListView contiene elementos de tipo String y el acceder a ellos es una tarea sencilla porque simplemente devolviendo el elemento en la posición “i” se obtiene el texto que contiene.

Pero si por el contrario se quieren almacenar elementos que muestran la información de ejercicios o plantillas, deben crearse clases en las que se puedan almacenar esos datos y a la vez obtenerlos, ya que devolviendo el objeto de la posición “i” se devolverá un objeto complejo. Para ello se han definido clases con nomenclatura “Elemento…”, ya que dependiendo del tipo de información que se desee mostrar se utilizará una u otra clase.

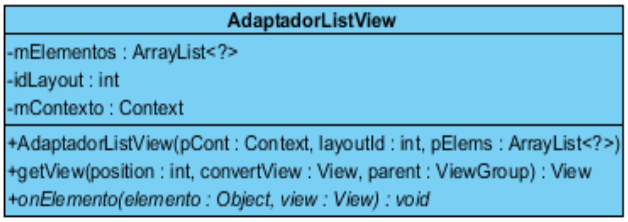


Ilustración 42: Detalle de las clases: Adapters- AdaptadorListView

La clase *AdaptadorListView* ha sido obtenida y modificada de la página web (10) la cual extiende de *BaseAdapter* y permite generar un listado al cual se le puede insertar un elemento genérico <?>, con lo cual es válido para cualquier aplicación.

Para instanciar un elemento de esta clase se necesita el contexto o actividad en el cual se va a insertar, el layout que va a mostrarse por cada uno de los elementos que componen el listado y el listado de elementos (objetos complejos o no) que va contener.

Adicionalmente tras instanciar esta clase se pedirá implementar el método *onElemento()*, en el cual se le debe especificar qué estructura van a tener los elementos que va a albergar el adaptador en ese caso en concreto, utilizando para ello los métodos getters definidos en cada clase "Elemento…".

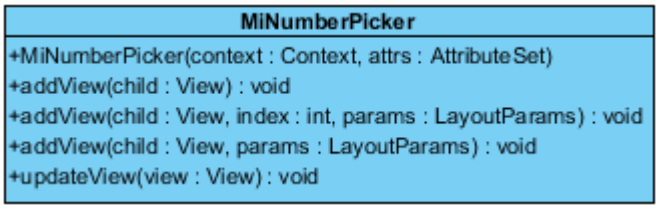


Ilustración 43: Detalle de las clases: Adapters- MiNumberPicker

La clase *MiNumberPicker* es una modificación de la clase Android *NumberPicker*, la cual ha sido modificada para poder ajustar el tamaño del elemento y color de los valores numéricos que muestran por defecto. Ha sido utilizada en el fragment de configuración de ejercicios.

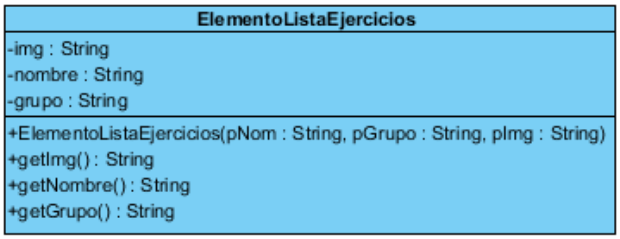


Ilustración 44: Detalle de las clases: Adapters- ElementoListaEjercicios

La clase *ElementoListaEjercicios* ha sido utilizada en la actividad *ListadoEjercicios* para mostrar los datos de cada ejercicio en cada uno de los elementos del listado personalizado. A partir del atributo "img" se obtiene la url de la imagen a cargar, mientras que sus otros dos atributos "nombre" y "grupo" se muestran como información referente al elemento en cuestión.

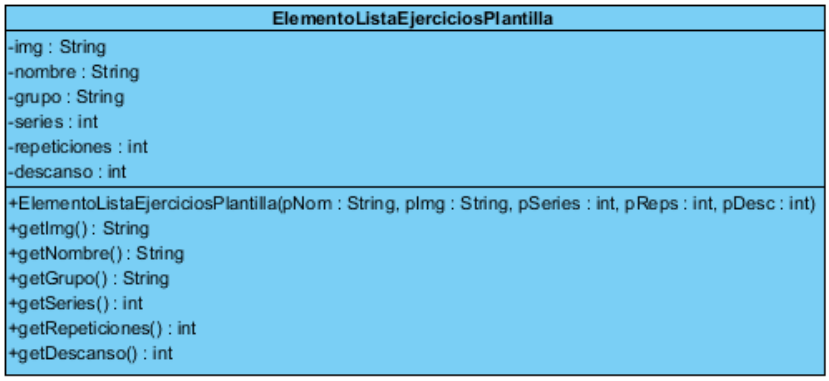


Ilustración 45: Detalle de las clases: Adapters- ElementoListaEjerciciosPlantilla

De funcionamiento similar a la clase anterior, *ElementoListaEjerciciosPlantilla* se encarga de mostrar aquellos datos de un ejercicio que ya se ha configurado y añadido a una plantilla de entrenamiento. Se utiliza tanto durante la creación de una nueva plantilla para ver los ejercicios añadidos y sus configuraciones así como durante la visualización de una plantilla ya configurada.

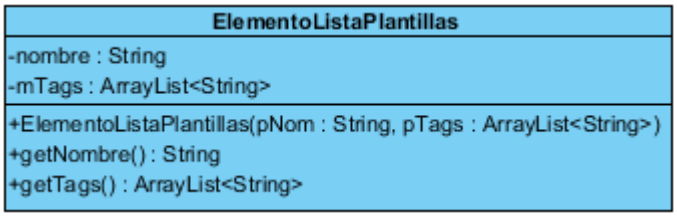


Ilustración 46: Detalle de las clases: Adapters- ElementoListaPlantillas

La clase *ElementoListaPlantillas* ha sido utilizada en la actividad *MisPlantillas* para mostrar en el listado personalizado cada una de las plantillas que el usuario ha creado. En ella únicamente se muestra el nombre de la plantilla así como los grupos musculares que tiene en uso.

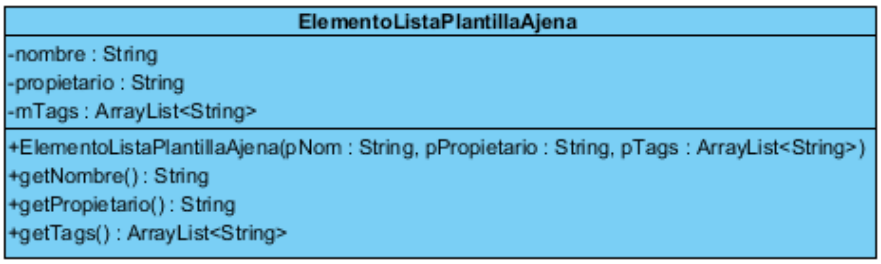


Ilustración 47: Detalle de las clases: Adapters- ElementoListaPlantillaAjena

ElementoListaPlantillasAjena muestra los elementos que forman los listados de las clases *MisFavoritos* o *PlantillasBuscadas*. Es similar a la clase anterior, salvo que en esta ocasión se muestra tanto el nombre del propietario de la plantilla de entrenamiento así como una imagen de una estrella destacando aquellas plantillas que el usuario tenga añadidas a favoritos.

5.3.4. Actividades

Son clases que contienen una interfaz visual o layout con elemento gráficos y además son capaces de responder a las acciones del usuario sobre dichos elementos. Extienden de la clase *AppCompatActivity* y en muchas de ellas se implementan métodos de las interfaces definidas en las clases anteriores, ya sea para reaccionar ante la respuesta del servidor (*AsyncPostCall*) o para mostrar cambios importantes en el sistema (*DialogFragments*).

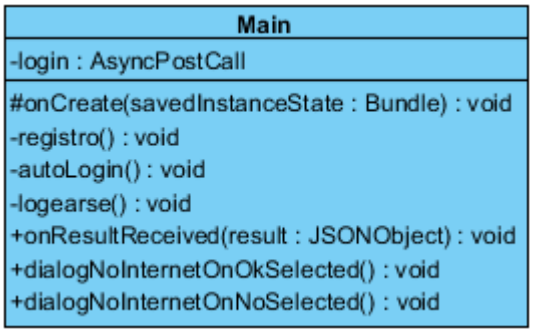


Ilustración 48: Detalle de las clases: Actividades- Main

La actividad *Main* permite al usuario iniciar sesión en la aplicación así como crear una cuenta de usuario en ella. Su funcionamiento da la posibilidad de recordar las credenciales para no tener que volver a introducirlas manualmente la próxima vez que el usuario intente iniciar sesión, de esta manera el funcionamiento es más fluido.

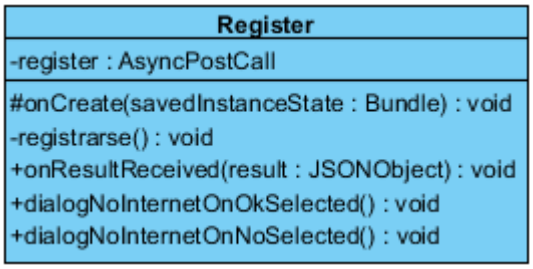


Ilustración 49: Detalle de las clases: Actividades- Register

La actividad *Register* gestiona los datos de registro y realiza una petición HTTP al servidor para generar una nueva tupla en la DB con los datos del usuario en caso de que sean correctos. Si el proceso es correcto se procede a iniciar sesión, y si es fallido se muestra una ventana de diálogo informativa.

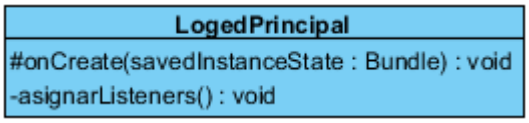


Ilustración 50: Detalle de las clases: Actividades- LogedPrincipal

LogedPrincipal gestiona la actividad o "pantalla" raíz de la aplicación una vez se ha iniciado sesión. Desde ella se puede acceder a las funcionalidades que ofrece *Time To Train* gracias a unos listeners que se les han asignado a los botones disponibles en pantalla.

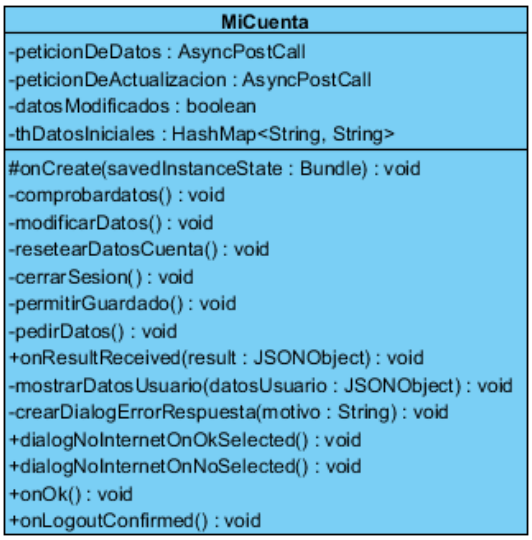


Ilustración 51: Detalle de las clases: Actividades- MiCuenta

La actividad *MiCuenta* permite al usuario modificar sus datos de usuario en cualquier momento y gestiona el cierre de sesión, olvidando las credenciales y redireccionando a la actividad de inicio de sesión *Main*.

Posee la opción de resetear los datos a los originales si se han modificado pero no se han sincronizado aún con el servidor, de esta manera si se cambian los datos accidentalmente se pueden volver a recuperar los últimos datos sincronizados del servidor.

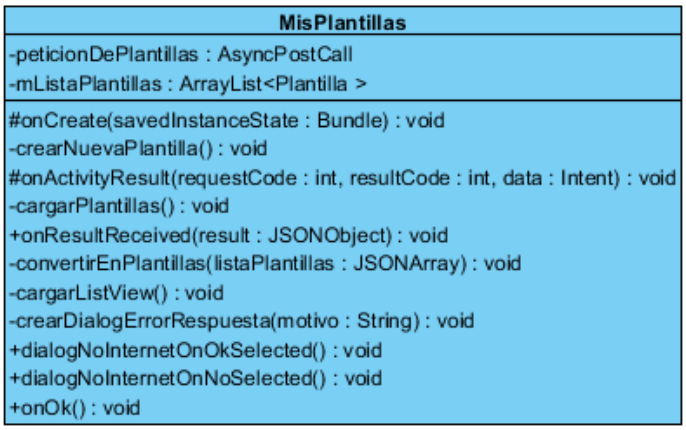


Ilustración 52: Detalle de las clases: Actividades- MisPlantillas

MisPlantillas es una actividad que se encarga de mostrar el listado de plantillas que el usuario actual ha generado. Para ello al acceder a esta actividad se realiza una llamada HTTP al servidor al comienzo y carga el listado en función del objeto JSON devuelto. También permite acceder a las áreas de creación de plantillas o información de cada plantilla y recargar el listado en cualquier momento a través de un botón de recargar listado.

Además cuando se activa el evento *onActivityResult()* se recargarán los datos ya que se puede haber generado un entrenamiento nuevo o modificado alguno antiguo.

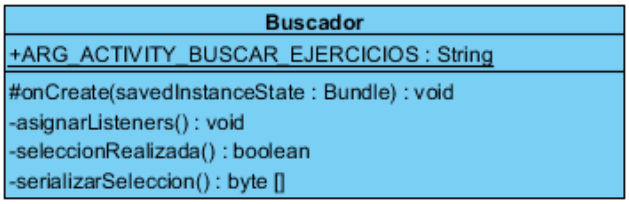


Ilustración 53: Detalle de las clases: Actividades- Buscador

La actividad *Buscador* gestiona las preferencias del usuario de cara a realizar una búsqueda indicando tanto los grupos musculares que buscar así como el elemento a buscar: ejercicios o plantillas de entrenamiento.

Para poder realizar la búsqueda se debe seleccionar al mínimo un elemento y, en caso de hacerlo se accederá a la actividad que realizará la búsqueda, siendo ellas *ListadoEjercicios* o bien *PlantillasBuscadas*.

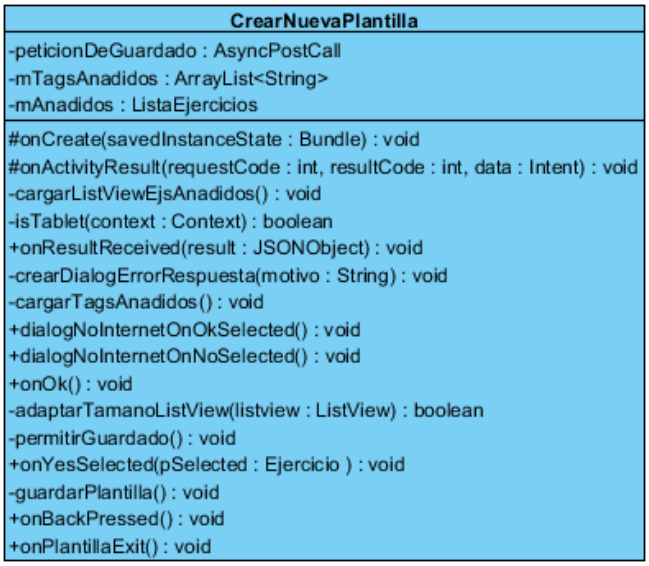


Ilustración 54: Detalle de las clases: Actividades- CrearNuevaPlantilla

La actividad *CrearNuevaPlantilla* gestiona la creación de una nueva plantilla, ya sea permitiendo al usuario escoger ejercicios y añadirlos a la plantilla en curso, eliminando ejercicios ya añadidos o modificando la configuración de los existentes.

Dicha creación se gestiona mediante listeners los cuales detectan que la configuración en curso tiene las propiedades necesarias para ser guardado como plantilla (nombre y listado de ejercicios), activando los iconos necesarios para proceder al guardado en el servidor y mostrando dinámicamente aquellos grupos musculares que la plantilla en curso contiene en todo momento.

También previene si el usuario accidentalmente pulsa salir durante una creación, lanzando el diálogo oportuno.

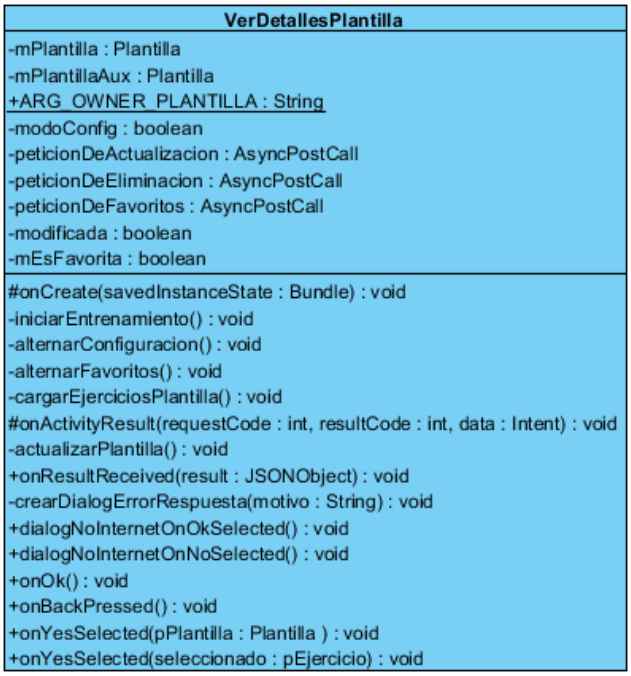


Ilustración 55: Detalle de las clases: Actividades- VerDetallesPlantilla

La actividad *VerDetallesPlantilla* muestra la información de la plantilla seleccionada. Para acceder a esta actividad se inserta como argumento una variable booleana cuyo nombre es igual al atributo estático "ARG_OWNER_PLANTILLA". Si dicha variable es "true" se mostrarán una serie de opciones que permitirán además realizar operaciones de gestión y modificar la plantilla o incluso eliminarla.

También gestiona posibles errores al realizar los guardados en el servidor, para lo cual es capaz de retroceder al estado anterior de la plantilla, el cual se guarda en el atributo mPlantillaAux.

Por último se puede agregar o eliminar de favoritos la plantilla si la configuración lo permite así como acceder a la actividad de entrenamiento.

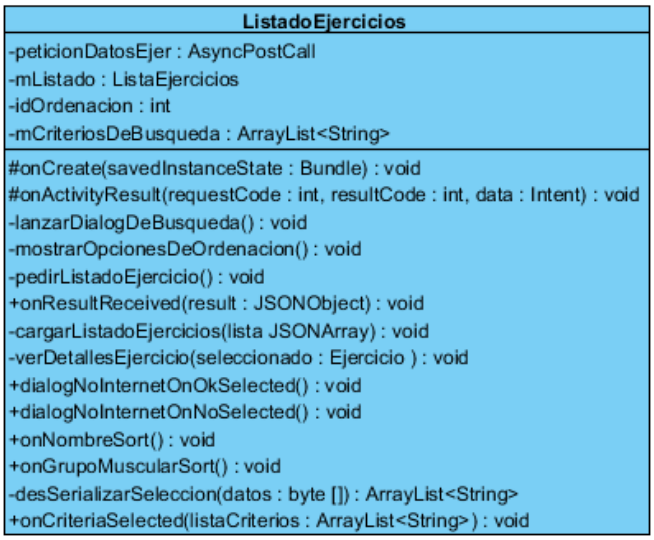


Ilustración 56: Detalle de las clases: Actividades- ListadoEjercicios

La actividad *ListadoEjercicios* se encarga de realizar una llamada al servidor para obtener el listado de ejercicios disponibles que satisfagan los criterios de búsqueda especificados. Posteriormente se genera un listado desde el cual el usuario podrá acceder a cada uno de los ejercicios pulsando sobre ellos.

Dicha actividad permite recargar el listado en cualquier momento, reiniciar la búsqueda cambiando únicamente el criterio actual y ordenar los elementos para que al usuario le sea más fácil encontrar el ejercicio deseado.

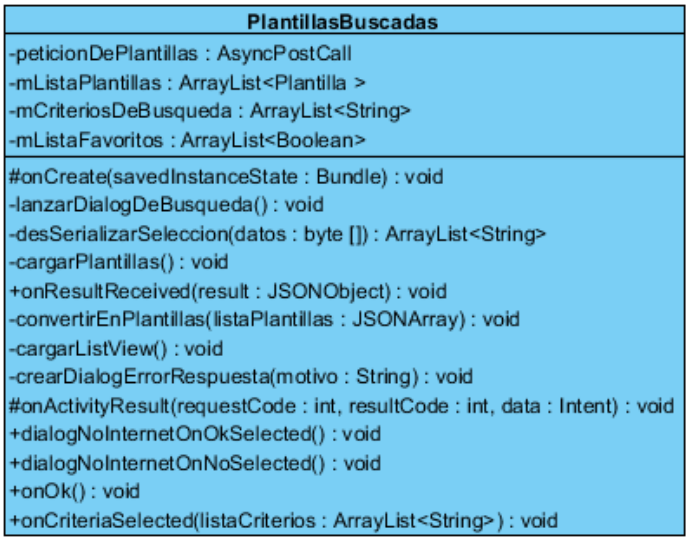


Ilustración 57: Detalle de las clases: Actividades- PlantillasBuscadas

PlantillasBuscadas es una actividad a la cual se accede desde el buscador y, en función de los parámetros de búsqueda realiza la petición de todas las plantillas que satisfacen dicho criterio. Además obtiene si para cada una de las plantilla el usuario las tiene añadidas a favoritos o no,

mostrando un icono familiar de una estrella en cada elemento del listado para diferenciarla de las que no tiene añadidas a favoritos.

Al igual que en la actividad anterior, permite rehacer las búsquedas o acceder a los detalles de los elementos pulsando sobre ellos.

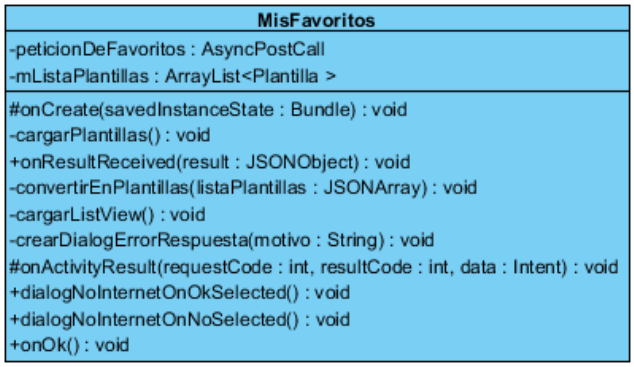


Ilustración 58: Detalle de las clases: Actividades- MisFavoritos

La actividad *MisFavoritos* se encarga de cargar en la actividad las plantillas que el usuario ha seleccionado como favoritas y permite acceder a los detalles de cada una de ellas.

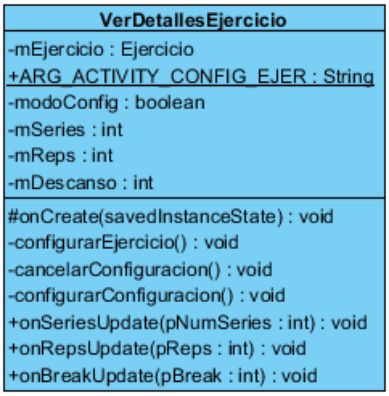


Ilustración 59: Detalle de las clases: Actividades- VerDetallesEjercicio

La actividad *VerDetallesEjercicio* gestiona la información a mostrar de un ejercicio seleccionado. Si al acceder a esta clase se detecta como parámetro el atributo estático "ARG_ACTIVITY_CONFIG_EJER" entonces se podrá visualizar y acceder a la configuración del ejercicio en la plantilla de entrenamiento desde la que se accedió al ejercicio.

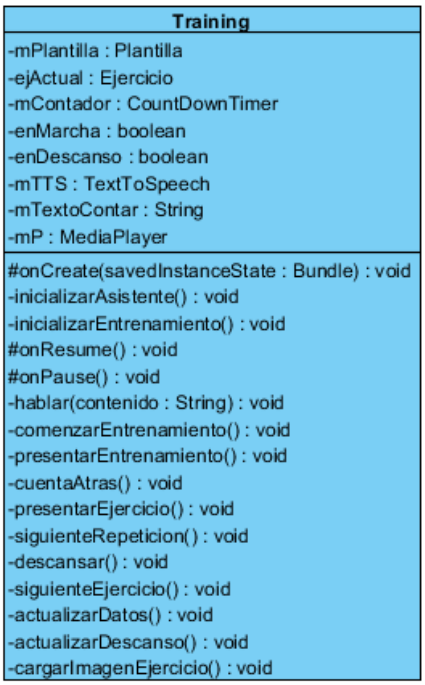


Ilustración 60: Detalle de las clases: Actividades- Training

Training es la actividad que gestiona el asistente de voz para que de las indicaciones a seguir durante la realización del entrenamiento. Pide a la clase del modelo *Plantilla* que proporcione los datos de los ejercicios para poder representarlo en la interfaz y maneja la gestión de tiempos para encadenar los mensajes del asistente.

5.3.5. Fragments

En este apartado se detallan cuatro clases que extienden de Fragment y que se utilizan durante la configuración de un ejercicio y durante el entrenamiento para mostrar los datos sin tener que lanzar otra actividad distinta. En el apartado de desarrollo de explicará en profundidad qué es un fragment.

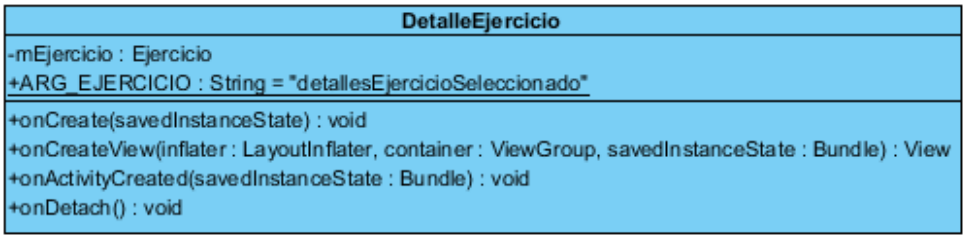


Ilustración 61: Detalle de las clases: Fragments- DetalleEjercicio

DetalleEjercicio es el fragment utilizado en la configuración del ejercicio. Muestra la información básica del ejercicio en el layout para que el usuario pueda ver la descripción que el creador del ejercicio le proporcionó desde la herramienta web diseñada para ello.

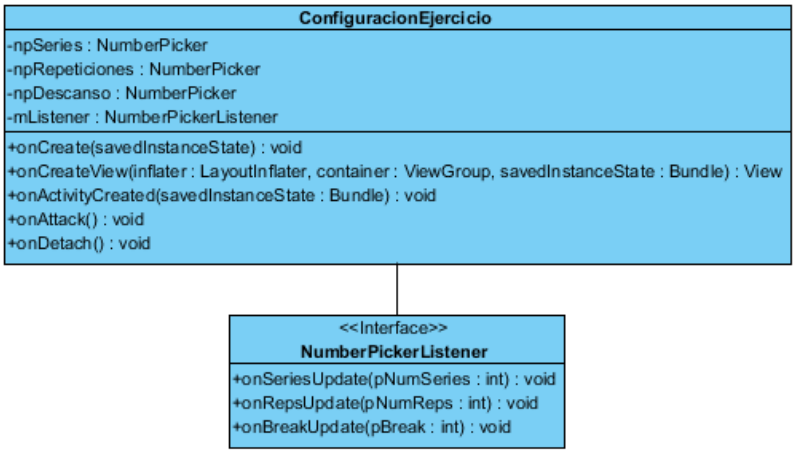


Ilustración 62: Detalle de las clases: Fragments- ConfiguracionEjercicio

El fragment *ConfiguracionEjercicio* contiene varios NumberPickers personalizados para que el usuario proporcione valor al número de repeticiones, series o descanso. Cada vez que estos valores se actualizan se activa la interface *NumberPickerListener* que define esta clase. Dicha interface es implementada en la actividad *VerDetallesEjercicio*.

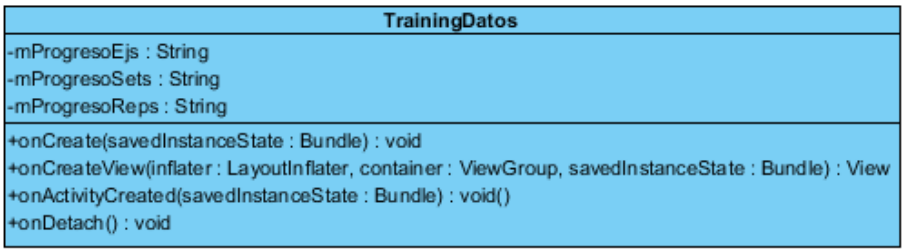


Ilustración 63: Detalle de las clases: Fragments- TrainingDatos

El fragment *TrainingDatos* muestra datos sobre la situación actual del ejercicio cuando se ha comenzado el entrenamiento. A medida que pasa el tiempo la actividad *Training* actualiza este fragment con nueva información para guiar al usuario.

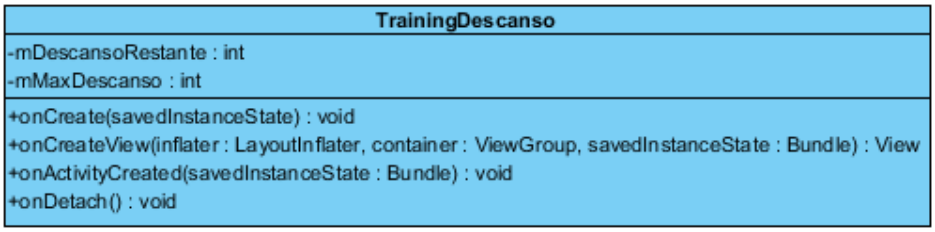


Ilustración 64: Detalle de las clases: Fragments- TrainingDescanso

El fragment *TrainingDescanso* contiene un progressbar y un texto cuyas apariencias han sido modificadas para asemejarse a un cronómetro, el cual indica el tiempo restante de descanso del que el usuario dispone al llegar al total de repeticiones de una serie. Al igual que el fragment anterior, su actualización es periódica.

5.3.6. Modelo

El último apartado contiene las clases del modelo que realizan operaciones de ordenación de elementos, búsquedas, comprobaciones, etc.

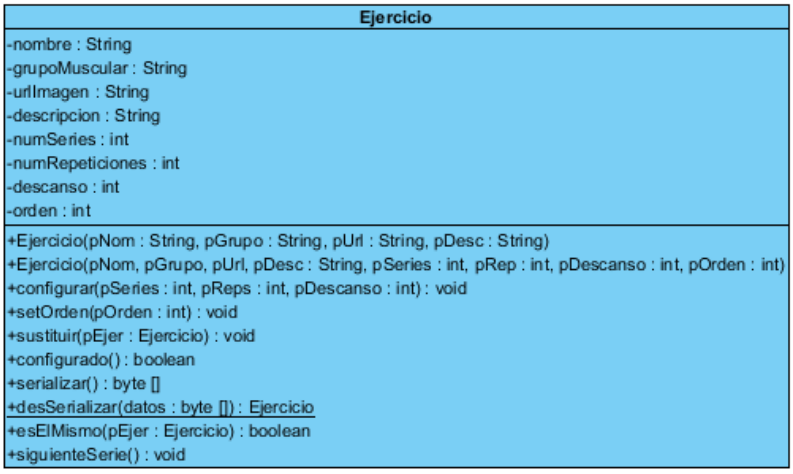


Ilustración 65: Detalle de las clases: Objetos- Ejercicio

Ejercicio es la clase del modelo esencial que define un ejercicio con dos constructoras para ser utilizadas en situaciones diferentes.

La primera es para poder diferenciar de un ejercicio que proviene del listado completo de ejercicios disponible, los cuales no están configurados. La segunda es utilizada cuando se obtiene una plantilla del servidor, la cual contiene un listado de ejercicios que sí se encuentran

configurados. Contiene operaciones de serialización, sustituir los datos o configuración del ejercicio.

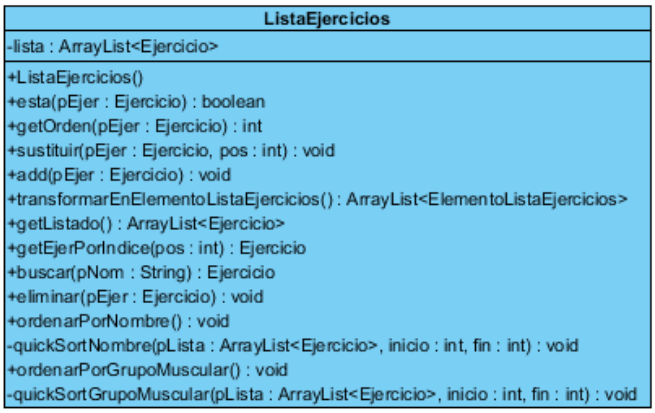


Ilustración 66: Detalle de las clases: Objetos- ListaEjercicios

La clase *ListaEjercicios* se encarga de gestionar el listado de ejercicios devuelto en la actividad *ListadoEjercicios*. Su funcionamiento se basa en aprovechar los métodos de un ArrayList así como añadir unos propios para hacer gestiones como ordenar por nombre o por grupo muscular, para lo cual se realiza una llamada al método *sutituir()* de la clase Ejercicio.

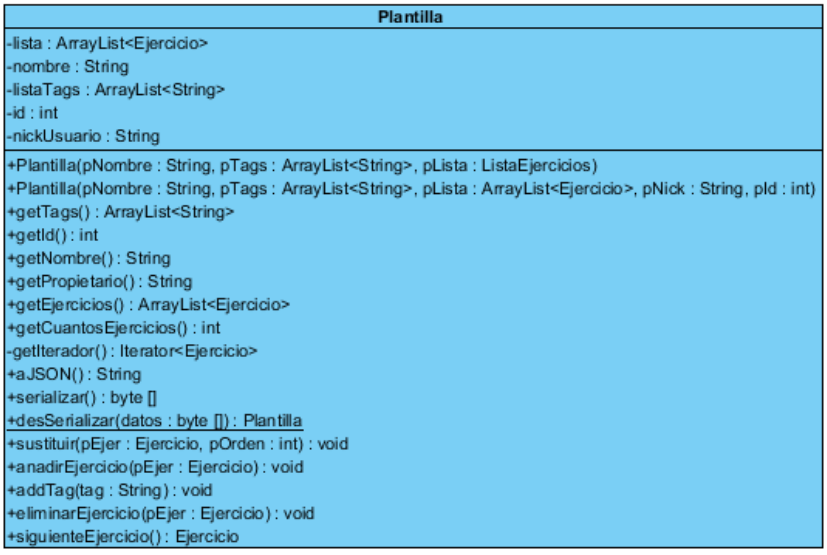


Ilustración 67: Detalle de las clases: Objetos- Plantilla

La clase *Plantilla* contiene toda la información de una plantilla. Al igual que la clase *Ejercicio* contiene dos constructoras. La primera es utilizada cuando la Plantilla aún no se puede considerar como correcta, ya que contiene un listado de ejercicios que podrían no estar configurados y para lo cual hay que ejecutar el algoritmo de *plantillaCorrecta()* para proceder a su guardado en caso de que sea válida. La segunda constructora es utilizada al recuperar los

datos de la DB, ya que se considera que es una plantilla correcta y configurada, además de contener datos como el nick del propietario y el id con el que se almacena la plantilla.

Gestiona los grupos musculares que contiene en todo momento, ya que al añadir un nuevo ejercicio o eliminarlo se actualiza automáticamente.

Por último contiene la lógica necesaria para ir avanzando entre ejercicios cuando se está ejecutando la actividad *Training*.

6. DESARROLLO

A continuación se detallará todo el proceso de desarrollo mencionando las diversas decisiones tomadas de cara a la implementación de la aplicación final y el porqué de su elección. Para que sea más cómoda la lectura se va a realizar una división entre la parte de desarrollo referente a la página web y la aplicación móvil.

6.1. Desarrollo web

Aunque inicialmente no estaba pensado desarrollar este apartado, el director del TFG aconsejó desarrollar una herramienta que permitiera la inclusión de nuevos ejercicios en la aplicación sin recurrir a la modificación del código de proyecto mediante actualizaciones, reforzando el proyecto de esta manera.

La página web desde la que los usuarios podrán generar sus ejercicios y almacenarlos en la aplicación resulta sencilla ya que no era el objetivo primordial de este proyecto, lo que no quiere decir que no se haya dedicado tiempo y dedicación a pulir las funcionalidades de esta herramienta, ya que ha sido necesario un exhaustivo proceso de pruebas para intentar encontrar todos los casos posibles de prueba.

Para que sus funcionalidades no fuesen tan limitadas se ha permitido que los usuarios que accedan a la página web puedan también gestionar sus datos de usuario modificándolos o eliminar su cuenta de usuario.

Para el desarrollo de la misma se ha recurrido al uso de Bootstrap 3, el cual es un conjunto de herramientas o framework desarrollado en HTML, CSS y JavaScript inicialmente para Twitter, que permite dotar a los elementos de una página web convencional de una apariencia más atractiva para el usuario.

De esta manera cada elemento o etiqueta del código HTML puede cambiar su apariencia si se le indica la o las clases de las cuales quieres tomar apariencia o comportamiento, siendo “form-control” una clase utilizada para convertir los campos de un formulario estándar en unos con un contorno redondeado y ligero brillo azulado, como se puede ver en la ilustración.

Ilustración 68: HTML & CSS estándar vs Bootstrap 3

Uno de los mayores inconvenientes a la hora de realizar el diseño de una página web es la visualización que ofrecerá dependiendo de la resolución de pantalla del dispositivo sobre el que se haya accedido.

Para ello Bootstrap ofrece un diseño *responsive* o *sensible*, lo cual permite adaptarse al tamaño de pantalla del dispositivo del que se ha accedido. Aunque también permite dotar de diferentes tamaños a cada elemento en función del dispositivo de acceso. Para lograr este efecto es necesario configurar los atributos de cada elemento de la página web añadiéndole las clases oportunas y configurándolas en un archivo de estilo .CSS.

En este proyecto se ha utilizado tanto un .CSS propio como el de Bootstrap ya que en algún caso se ha requerido de alguna característica específica y, al no encontrarla en Bootstrap se ha optado por diseñar un estilo propio.

De cara a realizar un diseño responsive en función del dispositivo de acceso es necesario tener una serie de conceptos o mecánicas claras ya que sino resulta imposible adaptar la anchura de los elementos. Y es que por cada elemento horizontal, Bootstrap divide en hasta 12 columnas cada uno de esos elementos. La sintaxis utilizada para dotar a un elemento de una anchura específica se ve en la siguiente tabla:

Tabla 14: Bootstrap Grid System

	Smartphones (<768px)	Tablets (>=768px)	Ordenadores (>=992px)	Ordenadores (>=1200px)
Nº de columnas	12	12	12	12
Clase	.col-xs-	.col-sm-	.col-md-	.col-lg-
Max. Columnas	12	12	12	12
Ancho columna	Auto	62px	81px	97px

Hay que destacar que dentro de un mismo elemento podemos incluir diferentes clases del sistema de rejilla mencionado anteriormente, de esa manera se puede configurar la página para que en móvil exista una repartición de 50% y 50% entre dos elementos al dotarles de las clases “col-xs-6” y “col-xs-6” respectivamente y hacer que en un ordenador de sobremesa se reparta un 25% y 75% al añadirles las clases oportunas a las ya existentes, resultando en “col-xs-6 col-md-3” y “col-xs-6 col-md-9”.

En caso de no añadir esta segunda clase el ordenador de sobremesa adoptaría la distribución de los elementos de la clase inferior más directa, en este caso “xs”.

En la siguiente página se muestra un ejemplo de la distribución realizada en una de las áreas de la página web, en la cual se divide en tres áreas y una de ellas a su vez en dos, teniendo una distribución de ~40% ~60% y haciendo que el tamaño de los botones sea diferente en ordenador y smartphone debido a la necesidad de alinearlos lo máximo posible.

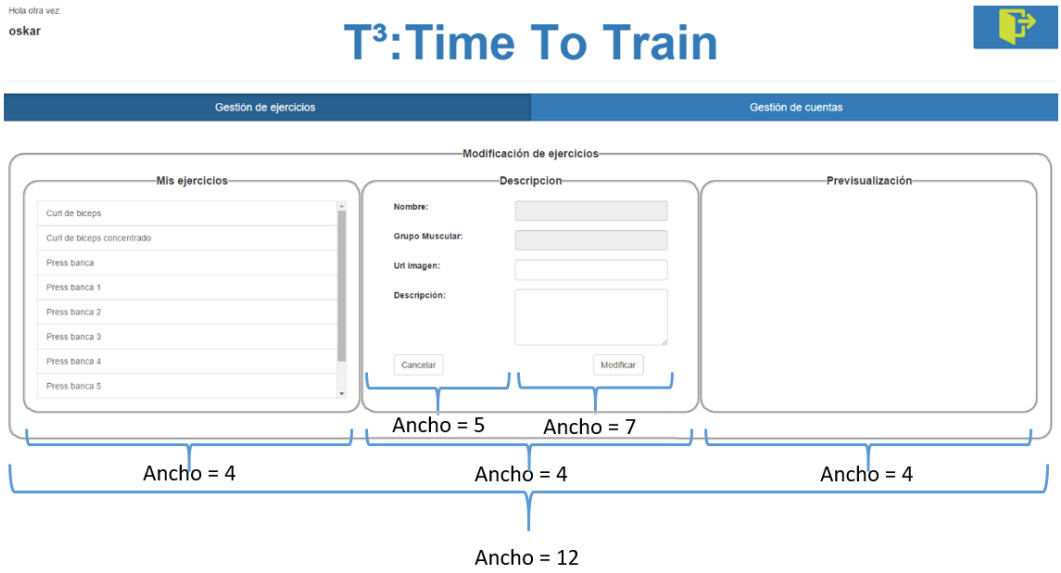


Ilustración 69: Bootstrap Grid System ejemplo interfaz

```
<!--este contenedor albergara las 3 zonas principales: listado, datos y previsualización -->
<div class="container-fuid col-xs-12 vert-offset-top-3">

    <fieldset class="scheduler-border">
        <legend class="scheduler-border">Modificación de ejercicios</legend>

        <!-- Primera area -->

        <fieldset class="scheduler-border col-xs-4">
            <legend class="scheduler-border">Mis ejercicios</legend>
        </fieldset>

        <!-- Segunda area, los datos de los ejercicios al ser pinchados-->
        <fieldset class="scheduler-border col-xs-4">
            <legend class="scheduler-border">Descripcion</legend>

            <div style="position:relative; overflow: auto; height:330px">

            <div class="form-group col-xs-12">
                <label for "nombre" class="col-xs-5 control-label">Nombre:</label>
                <div class="col-xs-7">
                    <input type="text" class="form-control" id="nombre" name="nombre" disabled>
                </div>
            </div>

            <div class="form-group col-xs-12">
            </div>

            <div class="form-group col-xs-12">
            </div>

            <div class="form-group col-xs-12">
            </div>
            <div class="form-group col-xs-12 col-lg-11">
                    <div class="col-xs-6 col-lg-9">
                        <a href="pag_gestion_ejercicios.php" class="btn btn-default">Cancelar</a>
                    </div>

                    <div class="col-xs-6 col-lg-3">
                        <button type="button" class="btn btn-default" onclick="llamadaAjax()" data-dis
                    </div>
                </div>
            </div>
        </fieldset>

        <!-- Tercera area  -->
        <fieldset class="scheduler-border col-xs-4">
            <legend class="scheduler-border">Previsualización</legend>

            <div class="col-xs-12" style="position:relative; overflow: auto; height:330px">
                <img class="img-responsive" id="previsualizacion" src="">
            </div>

        </fieldset>
```

Ilustración 70: Bootstrap Grid System ejemplo código

Las diferentes tablas obtenidas de la transformación del modelo del dominio fueron creadas y añadidas a la base de datos desde el apartado "SQL" de phpmyadmin ya que se encontró más cómodo el crear las tablas de esta manera, al no saber cómo indicar claves extranjeras en el método con interfaz gráfica. Un ejemplo de las tablas creadas tiene la siguiente forma:

```
CREATE TABLE plantillas
(
idPlantilla INT NOT NULL AUTO_INCREMENT,
nombre VARCHAR(30) NOT NULL,
nickUsuario VARCHAR (30) NOT NULL,
PRIMARY KEY (idPlantilla),
FOREIGN KEY (nickUsuario) REFERENCES usuarios (nickUsuario)
)
```

Cabe destacar que aunque en la aplicación no se utilizan el apellido y email del usuario, se guardan por si en el futuro la aplicación se ampliase, de esta manera el diseño estaría preparado para obtener el correo de un usuario y enviarle un mail con una notificación, por ejemplo.

Seguridad

A continuación se procederá a comentar la lógica del funcionamiento que ha sido implementado.

En la asignatura de 3º Sistemas de Gestión de Seguridad y Sistemas de Información (SGSSI) se dieron una serie de pautas a la hora de procesar los datos de un formulario, indicando que lo óptimo es que el botón que se crea para enviar los datos no sea de tipo "submit", sino de tipo button. A continuación, antes de enviar los datos se deben procesar con una serie de funciones implementadas por el desarrollador, y, de esta manera, validar cada uno de los campos.

Actualmente, gracias a HTML5 este paso no es necesario realizarlo, ya que se han incluido una serie de características que facilitan estas validaciones. En cada campo del formulario se puede especificar un patrón que el dato introducido debe cumplir de acuerdo a las expresiones regulares que se decidan indicar, lanzando un error en forma de notificación *popover* en caso de incumplirlas (Ver Ilustración 71).

Ilustración 71: HTML5 validación cliente

Ilustración 72: PHP Validación servidor

Este añadido en HTML5 ahorra una gran cantidad de código de validaciones en una sola línea, lo cual ha sido muy de agradecer.

SQL Injection & XSS

No obstante tras haber cursado SGSSI y haber realizado la prueba opcional Hack-it en la que se incentivaba a los alumnos al ponerles en la piel de un supuesto hacker para mejorar la nota, se cayó en la cuenta de que era posible modificar el código HTML de la página, lo cual permite desactivar estas restricciones impuestas por los patrones y de esta manera, realizar ataques de SQL Injection y XSS.

Los ataques SQL Injection consisten en introducir una serie de caracteres que complementan la sentencia SQL que valida, por ejemplo, un inicio de sesión en el sistema. En la Ilustración 73 se puede ver una comparación de cómo al introducir 'OR '1' = '1 en ambos campos de identificación, las comillas simples hacen que se pueda insertar la igualdad 1=1 en la sentencia, lo cual devuelve TRUE como resultado y permite el inicio de sesión en el sistema.

```
$sqlNick  = "SELECT * FROM usuarios WHERE nickUsuario = '$_POST[nick]' && pass = '$_POST[pass]'";
$sqlEMail = "SELECT * FROM usuarios WHERE email = '$_POST[nick]' && pass = '$_POST[pass]'";

$sqlNick  = "SELECT * FROM usuarios WHERE nickUsuario = '' OR '1'= '1' && pass = '' OR '1'= '1' ";
$sqlInjection = "SELECT * FROM usuarios WHERE email = '' OR '1'= '1' && pass = '' OR '1'= '1' ";
```

Ilustración 73: SQL Injection

Los ataques XSS (cross-site scripting) consisten en permitir que se ejecuten scripts generados por el usuario. Pueden resultar realmente molestos si se almacenan en la base de datos, ya que al acceder a los campos de la tabla se ejecutaría el script que ser inofensivo (un simple alert) o destructivo (ejecutar una sentencia con un delete).

Generalmente estos exploit no serán utilizados ya que el usuario medio desconoce este tipo de artimañas. No obstante lo más recomendable es hacer una validación doble, la primera en el lado del cliente con HTML5 y la segunda en el servidor, donde es imposible manipular los datos y sobre lo cual se hablará más adelante.

Referencia directa insegura a objetos

Para evitar la intrusión en el sistema se ha incluido una pequeña estructura de código (Ver Ilustración 74) para impedir el uso de usuarios no identificados a áreas en las que se requiere identificación (Ver Ilustración 75). Para enviar los datos de un formulario existen dos métodos de envío disponibles GET y POST.

El método GET envía los parámetros insertándolos en la url, por lo que son visibles para el usuario, pudiendo modificarlo y así provocar efectos no deseados al engañar al sistema.

El método utilizado por la web desarrollada es el POST, en el cual los parámetros viajan en segundo plano, ocultándose al ojo del usuario por lo cual su modificación es más complicada, proporcionando un extra de seguridad.

```php
<?php
    session_start();

    if(!isset($_SESSION['usuario'])){
        echo "<script language='Javascript'>alert('Acceso no permitido a usuarios no identificados');</script>";
        header("refresh:0;URL=main.php");
    }
?>
```

Ilustración 74: Referencia directa insegura a objetos 1/2

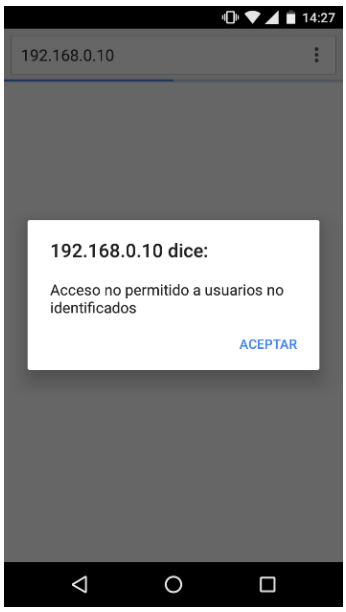


Ilustración 75: Referencia directa insegura a objetos 2/2

Exposición de datos sensibles

Finalmente el último de los elementos de seguridad que se ha incluido en la aplicación web es la de cifrar aquella información que se considera crucial. En este caso se ha recurrido a la criptografía (sistema usado ya en la antigüedad para cifrar la información) y así dotar de cierta seguridad a las contraseñas almacenadas en el sistema al usar el sistema de cifrado MD5.

MD5 forma parte de los llamados “algoritmos de resumen”, el cual muchos de los archivos y documentos oficiales que se ofrecen en la web poseen. El resumen criptográfico no es más que un código de (16 o 32 bits) que es aplicable tanto para cadenas de caracteres como para archivos. De esta manera permite obtener un “String” resultado de cifrar la información en el caso de las contraseñas, o bien un código en el caso de los archivos y así comprobar que el origen del mismo es una fuente de confianza y no ha sido modificado.

Uno de los problemas que conlleva el uso de este tipo de algoritmos es la colisión de resúmenes, esto es, dos cadenas diferentes pueden tener el mismo resumen criptográfico. Para intentar paliar esta posibilidad se ha optado por el uso del algoritmo MD5 de 32 bits frente al de 16bits que también ofrece PHP.

```
//buscamos alguna tupla cuyo usuario y contraseña coincida con el de inicio de sesion
$sqlNick  = "SELECT * FROM usuarios WHERE nickUsuario = '$_POST[nick]' && pass = md5('$_POST[pass]')";
$sqlEMail = "SELECT * FROM usuarios WHERE email = '$_POST[nick]' && pass = md5('$_POST[pass]')";
```

Ilustración 76: Exposición de datos sensibles

Aclarar que el algoritmo MD5 es un algoritmo unidireccional. Esto quiere decir que una vez se encripta la información no es posible traerla de vuelta a su estado original.

A continuación se ofrece un esquema obtenido de la asignatura SGSSI el cual explica el procedimiento cuando se quiere comparar la contraseña introducida con una ya encriptada, el cual utiliza una hash function (también denominada función de reducción, con las cuales se tenía experiencia al haberlas utilizado para implementar HashMaps propias en Java en la asignatura EDA) para obtener un código a partir de la cadena original.

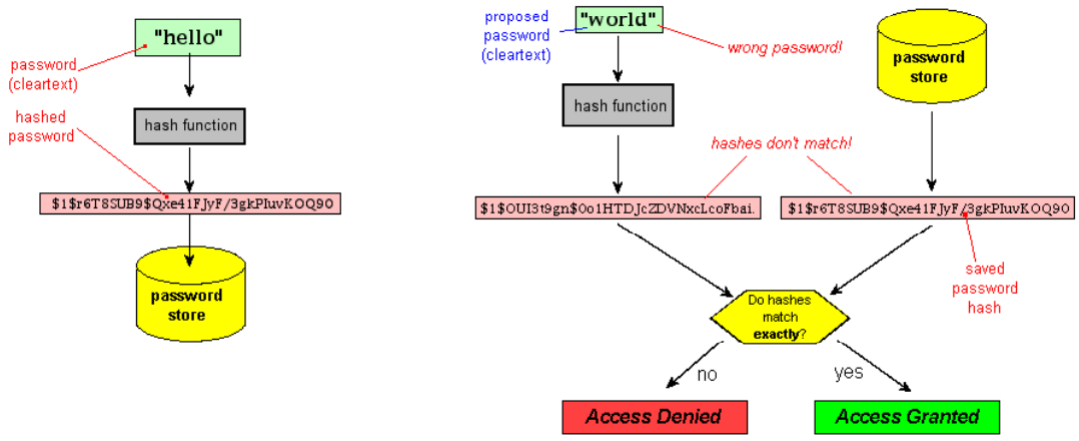


Ilustración 77: Ejemplo de encriptación y verificación

Diseño inicial

Al no existir la asignatura “Sistemas web” en el plan antiguo de ingenierías, hay que destacar que los conocimientos de jquery son bastante básicos y el funcionamiento inicial de la web era algo muy poco profesional, al realizar recargas de página cuando se hacían peticiones PHP.

Inicialmente el funcionamiento del registro, por ejemplo, consistía en una página HTML con dicho formulario y un área cuya clase creada en CSS y denominada “oculto” permitía que una sección se mantuviese en la página sin ocupar espacio. Dicha área iba a ser utilizada para mostrar los posibles mensajes de error que el servidor PHP devolviese al realizar la petición de registro.

Para poder recibir un mensaje de respuesta de vuelta se acudió al profesor Juanan Pereira, antiguo profesor de DAWE, para valorar diferentes maneras de hacer esto posible. Inicialmente se escogió una manera de devolver el mensaje el cual consistía en realizar un header (Location: pagina.php) y añadirle ?atributo al final del location. De esta manera se realiza un redireccionamiento a la página elegida y se puede realizar una función en PHP que detecte que la variable *$_GET['atributo']* está inicializada. De esta manera podría detectar un mensaje de error o de éxito y ocultar/mostrar las secciones ocultas mencionadas anteriormente, aunque se siguieron sopesando otras opciones ya que "ensuciar" la url con atributos no era algo que atrajese al desarrollador.

Con este planteamiento inicial se realizó una primera demostración del prototipo web al director del proyecto Mikel Villamañe, quien aconsejó realizar llamadas AJAX. De esta manera la página no se recargaría, los formularios no se borrarían y no ensuciaría la url con información no importante. Por ello, cada página que lo requiere contiene su función AJAX oportuna para pedir datos al fichero .php correspondiente.

Para no repetir funciones tanto en JavaScript como en PHP se han tenido en cuenta aquellas funciones que han sido utilizadas por varios ficheros y se han incluido en su correspondiente archivo funciones_generales.js o funciones_generales.php.

Diseño final - AJAX

El funcionamiento de las llamadas AJAX implementadas es el siguiente:

Funcionamiento HTML5

Para que se realice una validación de campos de HTML5 es obligatorio que el botón de verificación sea de tipo SUBMIT, el cual, tras realizar la comprobación llamará al .php de action asignado en el formulario. Esta manera de ejecutarse tiene un problema, y es que si se realiza una llamada AJAX en el onClick del botón submit se ejecuta dos veces el archivo .php, una por la llamada AJAX y otra por el action.

Finalmente se encontró una solución para este problema. Los formularios contienen un atributo especial denominado "onsubmit=" al cual se le puede asignar un valor. Si este valor es "False" se realiza la validación del formulario pero no llega a enviarse al archivo del action. Si en la función que realiza la llamada del AJAX se finaliza con un "return false" y se asigna de esta manera el onsubmit="return llamadaAJAX()", entonces se realizaría la petición una única vez.

Funcionamiento JavaScript

Esta sección del funcionamiento de una llamada AJAX es muy sencillo, se basa en recoger los datos del formulario y generar un objeto de tipo XMLHttpRequest, configurando sus

parámetros y enviándolo mediante POST (recordemos que POST es un método de envío de datos entre páginas en el que se mantiene una cierta privacidad al no mostrar los datos en la URL, al contrario que el método GET).

El objeto XMLHttpRequest tiene un método que se activa al recibir el mensaje de respuesta del fichero .php al que se hizo la petición. Una vez recibido, se devuelve un String cuyo contenido determina el éxito o fracaso de la petición. Si el mensaje está vacío se considera un éxito y si contiene texto se considera un error, siendo dicho texto el motivo.

Dependiendo de cuál de los casos se cumpla, se realizará una llamada a la función que se encuentra en el archivo funciones_generales.js el cual muestra u oculta el div correspondiente al mensaje de error asignándole el texto de respuesta de la llamada AJAX.

Funcionamiento PHP

Posiblemente la parte más abrumadora de implementar inicialmente, pero curiosamente la que más sencilla ha resultado.

Tras obtener los datos de conexión que alberga el archivo datos_conexion.php, se realizan las comprobaciones de los datos de acuerdo a las expresiones regulares y, en caso de no cumplirse alguna la validación del formulario, se detiene y devuelve el error el cual será captado por la función AJAX de JavaScript y se mostrará en la página web. Si la validación es correcta el funcionamiento continuaría.

La parte más importante de este apartado es que, aunque a través de la edición de código HTML con la consola del explorador se pueden enviar elementos para perjudicar a otro usuario, por ejemplo, eliminando sus ejercicios desde una cuenta ajena, dicha maniobra sería imposible de llevar a cabo, ya que cada procedimiento se verifica con la sesión que está iniciada en el navegador, y al ser código PHP, es imposible modificarlo en el lado del cliente.

Decisiones tomadas

Algunas de las decisiones más importantes que se han decidido tomar en este apartado es, por ejemplo, la modificación de los datos de un ejercicio o la cuenta de usuario.

Cambio de nick o nombre de ejercicios

Se ha impedido la modificación de tanto el nick de usuario como el nombre y grupo muscular de un ejercicio, cuyos motivos son los siguientes:

En el primer caso se ha tomado dicha decisión por seguir un estándar que parece haber entre páginas web, y es que pensándolo detenidamente, no dejan cambiar el nombre de usuario, pero sí los datos personales.

En el caso de los ejercicios, sólo puede existir un nombre para un ejercicio. Todas las variaciones posibles se pueden incluir añadiendo el tipo de variación al ejercicio. De la misma forma, un ejercicio siempre va a ejercitar un tipo de músculo, es imposible que ejercite otro, ya que de ser así seguramente sea porque la posición ha cambiado ligeramente, pero para este caso bastaría con indicarlo en el nombre del ejercicio al crearlo. Por ejemplo:

Curl de bíceps -> Curl de bíceps sentado -> Curl de bíceps concentrado -> Curl de bíceps…

Codificación de caracteres en BD

Entre los inconvenientes detectados durante el desarrollo de la herramienta web se ha encontrado que la codificación de caracteres UTF-8 que se indica en los documentos HTML5 no son suficientes de cara a almacenar los datos en la base de datos MySQL, y es que al almacenar un nombre con tildes, aun permitiendo tildes en las expresiones regulares, los valores almacenados podían almacenarse con errores en caso de contener palabras acentuadas (José Martínez, por ejemplo).

Para evitar eso, al generar una conexión con la base de datos hay que indicar qué tipo de codificación de caracteres se va a utilizar como se muestra en la siguiente figura. De esta manera se pueden almacenar tildes y caracteres especiales sin errores

```php
<?php

define('NOMBRE_SERVIDOR','localhost');
define('USUARIO','oskar');
define('PASSWORD','timetotrain');
define('DB','timetotrain');

//realizamos la conexion
$conexion = mysqli_connect(NOMBRE_SERVIDOR,USUARIO,PASSWORD,DB);

//aqui indicamos que lo que se vaya a almacenar en la BD va a ser en formato UTF-8, asi guardara acentos y tildes correctamente
mysqli_set_charset($conexion,"utf8");
?>
```

Ilustración 78: Codificación UTF-8 en PHP

Url de la previsualización del ejercicio

Otra de las decisiones tomadas ha sido la de elegir el servidor de almacenamiento de imágenes que se le pedirá al usuario. Es cierto que se podría haber elegido un abanico de servidores almacenando un array de expresiones regulares y preguntar a ver si alguna de ellas satisfacía la url proporcionada por el usuario, pero imgur.com es una plataforma de almacenamiento de imágenes mundialmente conocida que además ha desarrollado un formato de imágenes .gif denominado .gifv, el cual permite que se carguen hasta 10 veces más rápido dichas imágenes. El inconveniente encontrado es que HTML5 aún no tiene soporte para dicho formato, lo cual hace que no se carguen las imágenes en pantalla y, hasta que exista soporte para dicho formato se ha optado por implementar una función en .php que detecta si la url de la imagen proporcionada finaliza en .gif o en .gifv, eliminando el último caracter en dicho caso. De esta manera las imágenes se visualizan correctamente sin importar si su formato es .gif o .gifv.

Al eliminar un ejercicio

Se acudió al director del proyecto Mikel Villamañe para ver por qué decisión decantarse sobre las que ya se tenían en mente. Entre ellas estaban el no dejar eliminar un ejercicio, eliminarlo y tener que modificar las plantillas ajenas, lo cual aparte de resultar engorroso perjudicaría a los usuarios que tuviesen en uso ese ejercicio, o bien eliminarlo únicamente cuando no esté en uso por ningún usuario. Finalmente esta última opción ha sido la escogida ya que de esta manera se podría aliviar la carga en la BD en caso de haber ejercicios ocupando espacio y su implementación es sencilla.

Al eliminar un usuario

Al igual que en el caso anterior, ¿cómo actuar al eliminar la cuenta de usuario? Finalmente se ocurrió generar vía código en MySQL una figura de usuario la cual se ha denominado "*Time To Train*". Lo especial de esta figura es que no posee contraseña para identificarse y además su nick de usuario contiene espacios, por lo que es imposible que un usuario pueda identificarse o crear una similar ya que tanto las validaciones JavaScript como PHP implementadas lanzan errores si intentan usurpar la identidad de esta figura o generar cuentas similares.

Esta figura existe para que, cuando un usuario quiera borrar su cuenta, todos los ejercicios pasen a ser propiedad de *Time To Train* y de esta manera no exista el problema que ocurría en la decisión mencionada anteriormente.

¿Divs con errores y/o Modal?

Finalmente, aunque es una plataforma web, se ha buscado que se pueda acceder desde móviles y su apariencia sea lo más atractiva posible. Por ello en las áreas de eliminación y modificación de ejercicios, al constar de tres secciones (listado, descripción y previsualización del ejercicio seleccionado), se optó por darle un tamaño fijo a cada una de esas tres secciones a través de su estilo, ya que al dejar que se ajustase automáticamente se perdía simetría en el diseño.

Para calcular el tamaño adecuado para cada una de ellas, se comprobó que dándole un tamaño de 320px a cada área, la visualización de la página desde el smartphone de 4", la tablet de 10" y dos monitores de 19" y 26" resultaba correcta.

Tras realizar esta pequeña personalización de tamaño se encontró un problema, el cual aparecía al mostrar el mensaje de error o éxito que acostumbra a usar la página web. Y es que 320px resultaba insuficiente para el cuadro central en el que se mostraba el mensaje, deformando la apariencia.

Finalmente en el área de eliminación y modificación de ejercicios se ha optado por otro tipo de ventana emergente las cuales las proporciona un plugin de Bootstrap denominas *Modal*. De

esta manera la página muestra el error o el éxito de la acción sin perjudicar a la distribución de los elementos.

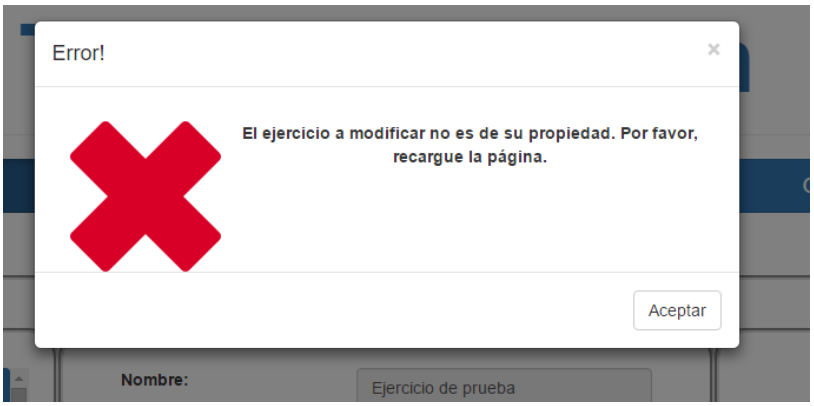


Ilustración 79: Ejemplo ventana Modal de Bootstrap

6.2. Desarrollo Android

En este apartado se comentará el uso que se le ha dado a los diferentes elementos de Android que se han considerado claves para desarrollar la aplicación.

Actividades

Una aplicación suele constar de diferentes actividades y para poder pasar los datos de una a otra se genera un objeto de tipo "Intent" al cual se le puede asignar una serie de argumentos de la misma manera que se almacenan datos en las tablas Hash, esto es, indicándole el par <K,V>, siendo "K" la clave o Key por el que identificarás el valor o Value "V" que almacenarás en el Intent.

En el método *.onCreate()* se pueden recuperar los posibles datos entrantes de una actividad con el método *.getExtras()* indicándole la clave deseada, obteniendo el valor original.

Inicialmente puede parecer un sistema un tanto limitado, ya que no permite pasar objetos complejos, por ejemplo, en *Time To Train* no permite pasar un objeto de tipo ListaEjercicios. Como argumento sólo permite pasarle un String, int, boolean, Charsequence, ArrayList un array de bytes y pocos tipos más.

Dado que pasarle un listado de objetos complejos uno a uno es algo mal visto se valoraron dos opciones.

La primera es que, tomando como referencia la lógica de los archivos .php generados durante el desarrollo de la aplicación web, se pensó en convertir los objetos en objetos de tipo JSON,

ya que de esta manera se convierten en un único String. El inconveniente que se le vio a este método es que al recibir el Intent hay que "parsear" la cadena JSON para rehacer el objeto original.

La segunda opción y la finalmente implementada es mucho más sencilla. Consiste en transformar un objeto en un array de bytes. Dicho proceso se conoce como serialización y para poder realizarlo sobre un objeto es necesario que todos los atributos del mismo sean igualmente serializables. Para ello hay que indicar en la cabecera de la clase que implementa la interfaz Serializable y su paquete: java.io.serializable. Al recoger el dato en la segunda actividad basta con llamar al método de des-serialización implementado en la clase, el cual garantiza que el array de bytes se convierta en el objeto deseado, siendo una ventaja el no necesitar parsear el objeto.

```
public byte[] serializar(){
    try {
        ByteArrayOutputStream bs = new ByteArrayOutputStream();
        ObjectOutputStream os = new ObjectOutputStream(bs);
        os.writeObject(this);
        os.close();
        byte[] bytes = bs.toByteArray();
        return bytes;
    }
    catch(IOException e){
        e.printStackTrace();
    }
    return null;
}

//funcion para que dada una cadena de bytes lo transforme en un objeto
public static Plantilla desSerializar(byte[] datos){
    try{
        ByteArrayInputStream bs= new ByteArrayInputStream(datos);
        ObjectInputStream ois = new ObjectInputStream(bs);
        Plantilla plantilla = (Plantilla)ois.readObject();
        ois.close();
        return plantilla;
    }
    catch (Exception e){
        e.printStackTrace();
    }
    return null;
}
```

Ilustración 80: Ejemplo Serializar objetos

Fragments

En la aplicación se pueden encontrar dos tipos de fragments:

- Fragment.
- DialogFragment.

Aunque sus nombres sean diferentes la mecánica a la hora de implementar es la misma. En primer lugar, los fragments estándar están implementados en, por ejemplo, la actividad de visualización de detalles de un ejercicio durante la configuración, la cual que está compuesta por una Activity y dos fragments. En la actividad de entrenamiento también existen dos fragments que se van actualizando con el progreso del entrenamiento o el descanso restante.

A continuación se muestra un ejemplo de cómo a la actividad "VerDetallesEjercicio" con su layout (imagen izquierda) se le pueden insertar distintos fragments (A o B) en función de si se deseaba ver la descripción o la configuración del ejercicio.

Ilustración 81: Fragments en la APP

Se ha decidido que en el layout de la Activity de configuración de ejercicio se muestren aquellos elementos que, aun cambiando de fragment deben permanecer impasibles, siendo en este caso el gif de la animación, ya que si éste fuera parte de cada fragment la animación se reiniciaría. Cuando se detecta que el usuario ha pulsado el botón de configuración se cambia la mitad inferior de la vista así como la barra de acción superior (ActionBar).

El segundo tipo, los DialogFragments, son elementos diseñados cuya funcionalidad está más extendida durante toda la aplicación, ya que son una serie de ventanas de diálogo que

informan al usuario de determinadas circunstancias (errores de conexión, petición de activación de redes inalámbricas, confirmación de eliminación, ordenación de elementos, etc.

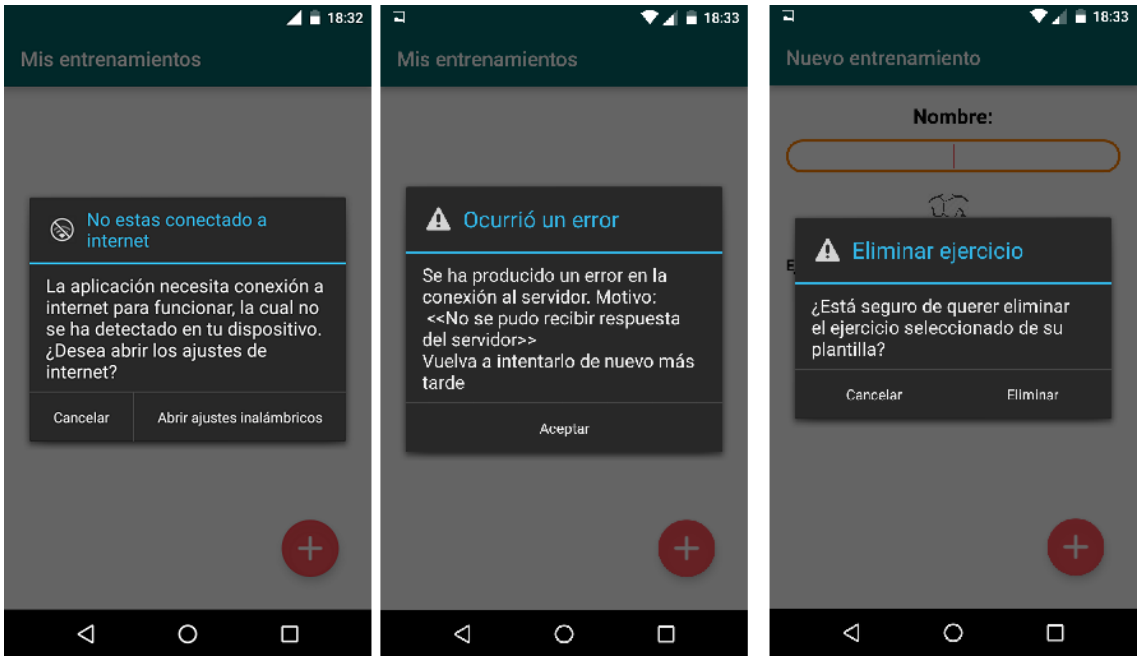


Ilustración 82: DialogFragments

A veces estos elementos reciben peticiones del usuario, pero no tienen control total sobre el sistema, ya que al encontrarse en diferentes partes de la aplicación es posible que en cada una se quiera que se comporte de una manera diferente. Esto se soluciona implementando Interfaces, de las cuales se hablará en el siguiente punto.

Como parte de las buenas prácticas, Google recomienda que el intercambio de mensajes entre Fragments se haga a través de las interfaces, siendo gestionadas desde una Activity, y nunca intentar comunicar fragments directamente entre sí.

Interfaces

Las interfaces son cabeceras de métodos que sirven para que, aquellas clases que vayan a utilizarlas implementen obligatoriamente esos métodos según el funcionamiento que estimen oportuno.

Google recomienda que sean las actividades quienes gestionen los datos de la vista, aunque ésta la esté proporcionando un Fragment. Por ello es necesario que, para recibir el estado de la vista en el momento en que el usuario realiza acciones sobre ella es necesario especificar Listeners, y la manera de obligar a que el usuario los implemente es mediante Interfaces.

A continuación se muestra un ejemplo de interfaz de listener correspondiente a la configuración de un ejercicio durante la configuración o modificación de una plantilla de entrenamiento.

```
public class FragmentConfiguracionEjercicio extends Fragment {
    /**
     * Created by oskar on 07/04/2016.
     */
    private Ejercicio seleccionado;
    private NumberPicker npSeries, npRepeticiones, npDescanso;
    private numberPickerListeners mListener;

    public interface numberPickerListeners{
        void onSeriesUpdate(int pNumSeries);
        void onRepsUpdate(int pReps);
        void onBreakUpdate(int pBreak);
    }
```

Ilustración 83: Interfaces: Declaración de los listeners

En función de la acción del usuario sobre los NumberPickers, se activará un método u otro.

```
npSeries.setOnValueChangedListener(new NumberPicker.OnValueChangeListener() {
    @Override
    public void onValueChange(NumberPicker picker, int oldVal, int newVal) {
        mListener.onSeriesUpdate(newVal);
    }
});

npRepeticiones.setOnValueChangedListener(new NumberPicker.OnValueChangeListener() {
    @Override
    public void onValueChange(NumberPicker picker, int oldVal, int newVal) {
        mListener.onRepsUpdate(newVal);
    }
});

npDescanso.setOnValueChangedListener(new NumberPicker.OnValueChangeListener() {
    @Override
    public void onValueChange(NumberPicker picker, int oldVal, int newVal) {
        mListener.onBreakUpdate(newVal);
    }
});
```

Ilustración 84: Interfaces: Activación de los listeners

Finalmente no hay que olvidarse de qué ocurriría si el desarrollador instancia un Fragment pero se le olvida implementar su interfaz. Para que ello no ocurra se insertan unas líneas de código que verifican que el listener se encuentra implementado en la Activity correspondiente, lanzando un mensaje de advertencia si el listener es null.

```
//nos aseguramos de implementar la interfaz que maneja los numberPickers
@Override
public void onAttach(Activity activity) {
    super.onAttach(activity);
    try {
        mListener = (numberPickerListeners) activity;
    } catch (ClassCastException e) {
        throw new ClassCastException(activity.toString() + " must implement el listener del Fragment de Configuracion de Ejercicios!");
    }
}
```

Ilustración 85: Interfaces: Obligando a implementar los listeners

Si el desarrollador olvida especificar que la actividad va a hacer uso de un tipo específico de fragment con listener definido mediante Interface con siguiente línea de importación (Ver Ilustración 86), entonces la aplicación lanzaría un error por el simple hecho de instanciar el fragment y ejecutar el método *.onAttach()*, mostrando el mensaje definido en el log de Android Studio el cual avisa al usuario del motivo (Ver Ilustración 87).

```
public class VerDetallesEjercicioActivity extends AppCompatActivity implements FragmentConfiguracionEjercicio.numberPickerListeners {
```

Ilustración 86: Interfaces: Cabecera

```
E/AndroidRuntime: FATAL EXCEPTION: main
                  Process: com.example.oskar.timetotrain, PID: 28287
                  java.lang.ClassCastException: com.example.oskar.timetotrain.Activities.VerDetallesEjercicioActivity@fe44b3b must implement el listener del Fragment de Configuracion de Ejercicios!
                      at com.example.oskar.timetotrain.Fragments.FragmentConfiguracionEjercicio.onAttach(FragmentConfiguracionEjercicio.java:122)
```

Ilustración 87: Interfaces: Log de error

Este funcionamiento se extiende para las opciones de las que disponen los DialogFragments, dándole al usuario la opción de cancelar una acción o aceptarla. Aunque sea necesario incluir la cabecera de los fragments no es necesario implementarlos posteriormente, ya que a veces se buscan comportamientos diferentes dependiendo de la actividad en la que se esté. Por ejemplo, es posible que si se produce un error en la conexión en una situación concreta lo más conveniente sea cerrar la actividad para que se reinicie cuando se haya solventado este problema y en otros casos, como en el de tener una plantilla de entrenamiento casi finalizada, conviene no cerrar la actividad ya que se echaría a perder el tiempo que el usuario ha gastado en su configuración.

Adaptador ListViews

Para mostrar listado de elementos en las aplicaciones Android utiliza una clase denominada ListView que si bien su uso es sencillo, su utilidad es bastante limitada ya que los listados que ofrece son simples (Ver Ilustración 88). Su funcionamiento permite asignarle un título y un subtítulo por cada elemento del listado, no siendo suficientes elementos para mostrar la información requerida en cada listado de *Time To Train*. Para poder utilizarlos es necesario

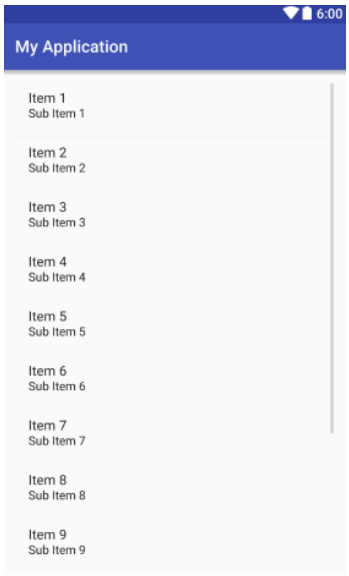


Ilustración 88: ListView básico

Para poder mostrar listados diferentes se investigó el uso de diferentes opciones encontradas por internet, aunque finalmente se decidió utilizar una clase encontrada en la página web [(10)] y que gracias al método abstracto *onElemento()* se podrá implementar fácilmente en aquellas actividades que requieran su uso.

```
public abstract class AdaptadorListView extends BaseAdapter{
    //codigo obtenido de http://jarroba.com/listview-o-listado-en-android/
    /**
     * Created by oskar on 30/03/2016.
     */

    private ArrayList<?> mElementos;
    private int idLayout;
    private Context mContexto;

    public AdaptadorListView(Context pContexto, int R_layout_id, ArrayList<?> pElementos){...}


    @Override
    public int getCount() { return mElementos.size(); }

    @Override
    public Object getItem(int position) { return mElementos.get(position); }

    @Override
    public long getItemId(int position) { return position; }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {...}


    public abstract void onElemento(Object elemento, View view);
}
```

Ilustración 89: AdaptadorListView: Código

Dicha clase tiene la particularidad de poder ser utilizado en cualquier listado que se desee crear ya que acepta cualquier tipo de elemento a mostrar. Para poder utilizarse son necesarios otros dos elementos:

1. Layout para cada elemento del listado.
2. Clase java que representa cada elemento del listado.

El layout muestra la estética que va a tener cada elemento, mientras que la clase java podría considerarse un extractor de información de cada elemento del listado. En un listado simple, al contener Strings es muy sencillo obtener la información de cada elemento, ya que acceder a la posición "n" del ListView proporcionará el contenido de dicho elemento. En cambio, si el tipo de elemento que contiene es un objeto complejo, accediendo a dicha posición "n" no se podrá obtener información que se pueda plasmar en el listado, por ello la necesidad de crear un objeto java que extraiga información.

Una vez se tienen estos elementos se puede proceder a asignar el adaptador al listado que lo requiera.

Para ello hay que indicar en la constructora del adaptador el contexto (**this**) o actividad donde se va a insertar el listado, el layout con la estética que va a adoptar cada elemento (**elemento_listado_ejercicio**) y el tipo de elementos que va a contener (en este caso listado es del tipo ArrayList<ElementoListadoEjercicio>).

Tras hacer esto el método abstracto onElemento se importa automáticamente y tan sólo hay que indicarle a cada elemento del layout de "**elemento_listado_ejercicio"** qué valores o imágenes cargar en cada elemento a través del método *view.findViewById().* Como se puede observar en la siguiente ilustración, se utilizan los métodos getters para acceder a cada elemento y asignarlo a cada uno de los tres elementos del layout.

```
lista.setAdapter(new AdaptadorListView(this, R.layout.elemento_listado_ejercicio, listado) {
    //por cada elemento que le pasemos al adapter hay que vincularlo con los elementos de la interfaz correspondientes
    @Override
    public void onElemento(Object elemento, View view) {
        TextView tvNombre = (TextView) view.findViewById(R.id.tvNom_elemento_listado_ejercicios);
        if (tvNombre != null) {
            String nombreEjer = ((ElementoListaEjercicios) elemento).getNombre();
            tvNombre.setText(nombreEjer);
        }

        TextView tvGrupoMuscular = (TextView) view.findViewById(R.id.tvGrupo_elemento_listado_ejercicios);
        if (tvGrupoMuscular != null) {
            String grupoMuscular = ((ElementoListaEjercicios) elemento).getGrupo();
            String codigo = "tag"+grupoMuscular;
            int idGrupoMuscular = getResources().getIdentifier(codigo, "string",getPackageName());
            tvGrupoMuscular.setText(idGrupoMuscular);
        }

        ImageView img = (ImageView) view.findViewById(R.id.img_elemento_listado_ejercicios);
        if (img != null) {
            String url = ((ElementoListaEjercicios) elemento).getImg();
            new AsyncImageLoader(img).execute(url);
        }
    }
});
```

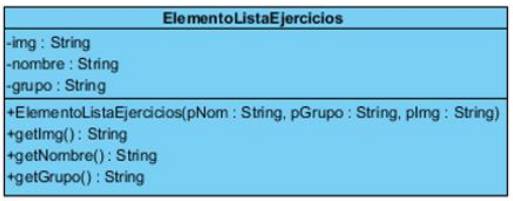


Ilustración 90: AdaptadorListView: Implementación

A continuación se muestran algunos ejemplos de los diferentes listados que se pueden encontrar por la aplicación. De izquierda a derecha son el resultado de: listar ejercicios disponibles, ver detalles de plantilla, mis plantillas y buscador de plantillas. Todos ellos usan el mismo adaptador, variando únicamente el layout y la clase del elemento a mostrar.

Ilustración 91: AdaptadorListView: Ejemplos

AsyncTask

A lo largo de la carrera se han implementado múltiples programas o métodos los cuales realizaban su ejecución de manera secuencial y no han sido hasta las asignaturas ABD y DAWE cuando se pudo probar, aunque fugazmente, que existían otras posibilidades en computación, en este caso, los hilos de ejecución en Java (Threads) y los Web Workers en JavaScript.

Las operaciones relacionadas con la interfaz deberían ser las únicas que se ejecuten en el hilo principal o Main Thread, y generar otros hilos para realizar en un segundo plano el resto de operaciones. Los desarrolladores deberían mantener lo más libre posible el hilo principal, ya que es donde se registra la interacción del usuario con el dispositivo.

Si las búsquedas o procesos de larga duración como una cuenta atrás se realizasen en el mismo hilo principal, entonces la UI no respondería durante esos períodos y sería imposible interactuar con el dispositivo, impidiendo, por ejemplo, cancelar el contador.

Para minimizar el impacto de este tipo de fallos que cometen los desarrolladores en ocasiones, Android ha sido diseñado de tal manera que si se queda bloqueada la UI por un determinado período (el tiempo puede ser de 5, 10 o 15 segundos) se produce un fallo conocido comúnmente como ANR (Application Not Responding) acompañado de un familiar mensaje que seguramente cualquier usuario de Android ha podido ver alguna vez.

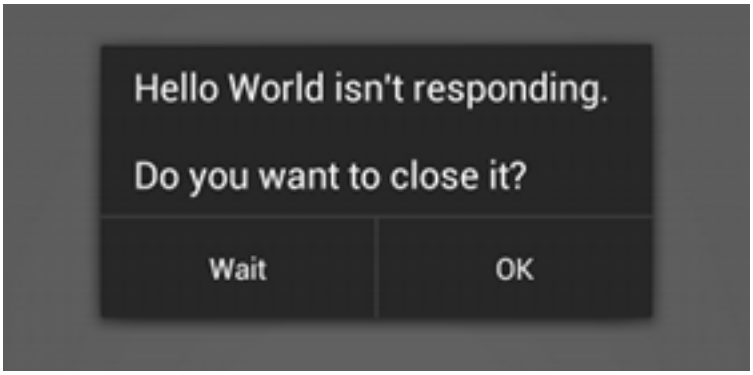


Ilustración 92: Application Not Responding (ANR)

En Android, AsyncTask o tarea asíncrona (Asynchronous Task) es la clase utilizada para permitir procesos paralelos. En *Time To Train* se han generado dos clases que heredan de AsyncTask: AsyncImageLoader y AsyncPostCall.

AsyncImageLoader es utilizada para cargar las imágenes en el adaptador personalizado, ya que se necesita que el listado esté disponible para poder manipularlo aun no habiéndose cargado las imágenes en el mismo, teniendo que realizar las cargas en segundo plano.

AsyncPostCall es utilizada para realizar las conexiones al servidor remoto, pedir los datos y, por medio de una interface, devolver el JSON devuelto del servidor a la clase que la esté implementando. Todos los ficheros .php a los que se realizan llamadas asíncronas devuelven objetos JSON llamados "response" con una estructura concreta con tres atributos. El primero de ellos se denomina "status" y puede contener un 1 o un 0, siendo 0 sinónimo de éxito al realizar la petición y un 1 sinónimo de fallo. En caso de contener un 0 se procede a obtener el valor de su segundo atibuto denominado "msg" y que contiene un mensaje de error codificado (más información en el apartado Strings.xml de desarrollo). Finalmente el tercer atributo, "result" contiene un objeto JSON cuando la petición ha sido correcta y que la aplicación deberá procesar para transformarlo en un objeto comprensible por el sistema.

Action Bar

La ActionBar, AppBar o barra de acción es una barra localizada en la parte superior el cual es un elemento indispensable en cualquier aplicación donde poder situar botones con los que navegar por la aplicación o realizar gestiones en la misma.

En *Time To Train* su uso ha sido indispensable para poder variar el comportamiento de algunas acciones.

Permite situar elementos en ese espacio para no ocupar espacio de la interfaz y obstaculizar la vista.

Por ejemplo, al observar los detalles de una plantilla de la que se es propietario, si el icono de configuración está activado permite eliminar la plantilla, añadir nuevos ejercicios o modificar

los existentes, mientras que si está desactivado se podrán ver la información que los usuarios hayan proporcionado a cada ejercicio así como comenzar el entrenamiento, como se puede ver en la Ilustración 93.

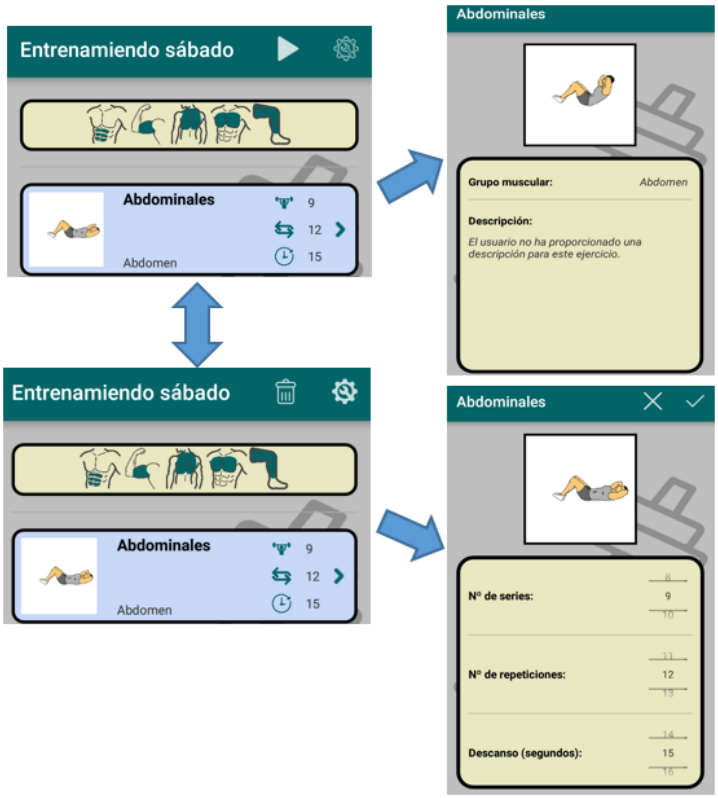


Ilustración 93: Alternando navegación con ActionBar

No obstante hay elementos que siguiendo la estética de otras aplicaciones, porque dado el tamaño de ActionBar es limitado o bien porque necesitan ser ubicadas en un lugar llamativo son ubicadas en la interfaz mediante botones flotantes, como es el caso de añadir ejercicios o comenzar entrenamiento.

Shared preferences

Uno de los problemas contra los que se enfrentan los programas es la volatilidad de sus variables perdiendo sus valores ya sea al cerrar la aplicación o apagando el dispositivo.

Gracias a la clase de Android "SharedPreferences" ese error se soluciona, ya que su funcionamiento consiste en poder almacenar variables en la memoria del dispositivo en forma de <Key, Value>. Dichos valores se almacenan en un fichero que se encuentra en la carpeta */data/data/nombre.del.paquete.de.aplicación/shared_prefs/nombre.xml* y su estructura está basada en etiquetas XML.

Para poder guardar variables es necesario especificar un nombre bajo el cual se almacenarán todas las variables que se quieran (Ver Ilustración 94), manteniéndose ordenadas de esta manera. En la aplicación se ha optado por el nombre "*Time To Train*", así todos los valores que la aplicación necesite se encontrarán en ese "paquete" dentro de la propia clase SharedPreferences.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<string name="sesion">oskar</string>
<boolean name="recordarLogin" value="true" />
</map>
```

Ilustración 94: SharedPreferences XML

En la aplicación únicamente ha sido necesario almacenar el valor del nombre del usuario con el cual ha iniciado sesión y si activó el CheckBox en el inicio de sesión para evitar el inicio de sesión manualmente la próxima vez que inicie la aplicación.

Para realizar cualquier petición asíncrona al servidor se envía el identificador de quién está realizando la petición, de esta manera si el usuario no se encuentra en el sistema o bien no tiene permisos para realizar la acción solicitada se produce un error. Por ejemplo, para enviar una petición de eliminación de plantilla se envía la plantilla seleccionada y el Nick del usuario que se guarda en las preferencias.

Strings.xml

Un elemento indispensable para cualquier aplicación hoy en día es que esté traducida a varios idiomas. Android proporciona una versátil herramienta que permite almacenar textos con sus traducciones en ficheros .xml y así poder visualizarlos cómodamente en un entorno gráfico adaptado a ello.

Para poder hacer uso de esta propiedad primero se debe definir una clave por la que localizar el recurso de texto a almacenar y la traducción que por defecto va a poseer ese texto cuando sea utilizado. A continuación, accediendo al fichero .xml que se almacena en la carpeta /res/values/strings.xml se pueden aportar traducciones a los idiomas que se desee, indicando manualmente las traducciones para todos los lenguajes que se deseen aportar.

Time To Train se encuentra en castellano e inglés, pero el listado de idiomas a los que poder traducir cualquier aplicación supera la cifra de 200 idiomas y dialectos, algunos de ellos se pueden observar en la Ilustración 95.

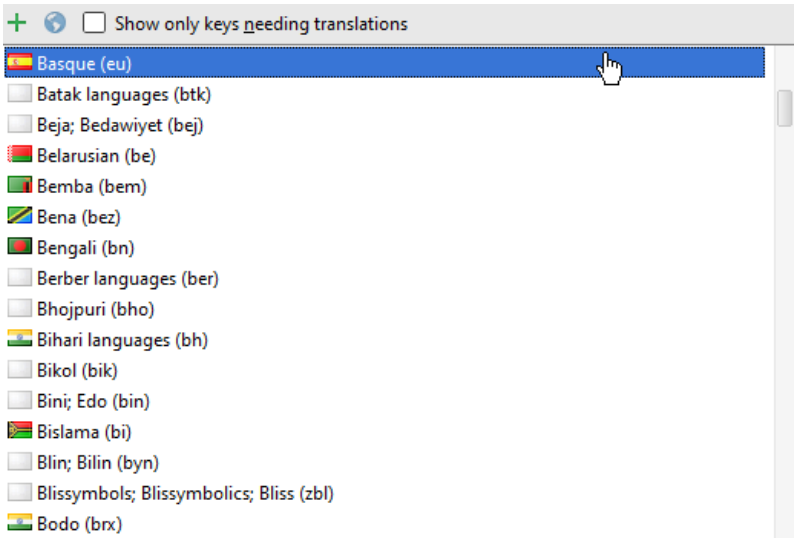


Ilustración 95: Idiomas disponibles

Su funcionamiento se basa en que al iniciarse la aplicación detecta el lenguaje que el dispositivo móvil tiene configurado e intenta conseguir los recursos referentes a ese idioma para que la aplicación los muestre en su idioma. En caso de no estar traducido al idioma del dispositivo se optará por mostrar los mensajes en el idioma por defecto.

Key	Default Value	Untra...	English (en)	Spanish (es)
abBuscador	Buscar entrenamientos y ejercicios	☐	Search training and exercises	Buscar entrenamientos y ejercicios
abFavoritos	Mis favoritos	☐	My favorites	Mis favoritos
abListadoEjercicios	Ejercicios disponibles	☐	Available exercises	Ejercicios disponibles

Ilustración 96: Editor de Traducciones

En las llamadas asíncronas realizadas por la aplicación es posible que en ocasiones ocurran errores ya sea porque los datos proporcionados no cumplen los requisitos de validación, porque el usuario no tiene permisos para realizar peticiones o cualquier otro motivo.

El fichero .php sería el encargado de procesar la petición y devolver un mensaje informando el porqué del error. Si el fichero .php enviase el motivo del error con un texto en concreto el sistema mostraría todos los mensajes del servidor con el idioma original (por ejemplo: “error de inicio de sesión”), lo cual desentonaría si no concordase con el idioma sobre el que se ejecuta la aplicación (por ejemplo: inglés).

Gracias a la herramienta de traducción que ofrece Android es posible traducir los mensajes, y es que en vez de devolver el mensaje en un idioma concreto se devuelve una codificación en el atributo “msg” del objeto JSON sobre el que se ha hablado en el apartado AsyncTasks. El ejemplo de la Ilustración 97 muestra el proceso de validación de una plantilla al ser añadida al sistema. Si el usuario que va guardarla posee un nick que no cumple los requisitos, igual que el nombre de la plantilla o bien la longitud del nombre se produce un error el cual es diferente para cada caso. Nota: después del “else” el proceso de validación continúa pero supone bastante código.

```php
//validamos los parametros de entrada antes de continuar
$resultadoComprobacion = validarFormulario();
if($resultadoComprobacion!="form_correcto"){
    $response['msg'] = $resultadoComprobacion;
}
else{
```

```php
//funciones php para validar los campos con expresiones regulares y evitar SQL injection
function validarFormulario(){

    global $nick,$nombrePlantilla;

    if(!validarNick($nick)){
        //error de nickname
        return "Cod13";
    }
    else{
        if(strlen($nombrePlantilla)>30){
            return "Cod22";
        }
        else{
            if(!validarNombrePlantilla($nombrePlantilla)){
                return "Cod21";
            }
            else{
                return "form_correcto";
            }
        }
    }
}
```

Ilustración 97: Generando código de error a traducir

Para mostrar el mensaje traducido basta con acceder al contenedor de recursos “String” del que dispone la aplicación y mostrar el mensaje en pantalla con la clase Toast.

```java
//cogemos generamos el codigo del error en base al msg que nos llega
String codigoString = "response" + result.getString("msg");
int idString = getResources().getIdentifier(codigoString, "string", getPackageName());
Toast.makeText(getApplicationContext(), getResources().getString(idString), Toast.LENGTH_SHORT).show();
```

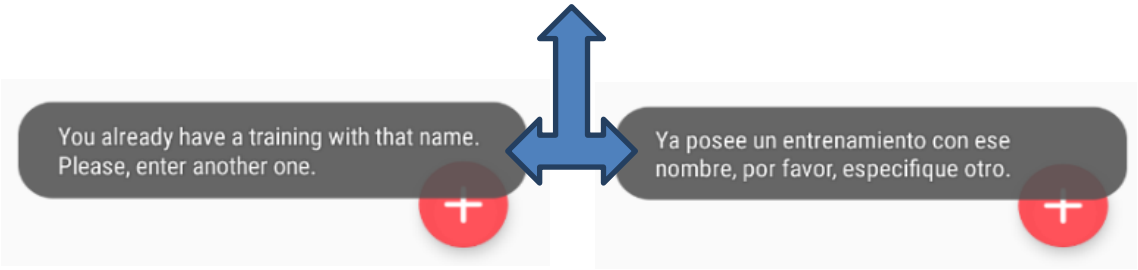


Ilustración 98: Traducción dinámica de los mensajes del servidor

Aspectos de usabilidad

Durante la asignatura HCI se estudiaron una serie de pautas que seguir de cara a favorecer la usabilidad del sistema y que se han intentado seguir de cara a mejorar la apariencia de la aplicación y las cuales se enumeran en este apartado:

- Hay que seguir una estructura y mantenerla a lo largo de la aplicación en la disposición de los botones. Aunque el sistema se encuentre en diferentes actividades lo ideal es que la disposición de los se mantenga. Botones con consecuencias positivas a la derecha y negativas a la izquierda.
- Utilizar un lenguaje o estética que sea familiar para los usuarios. Por ese motivo se ha decidido que el botón de añadir nuevos ejercicios esté en la misma posición que Gmail o Calendar poseen sus respectivos botones (abajo a la derecha).

- El uso de colores en su justa medida. El número de colores distintos recomendados para utilizar en una aplicación está entre 4 y 8. El uso excesivo de ellos tiene malos resultados de cara a no poder distinguir elementos o información importante en pantalla.
- Combinar colores. Existen diferentes técnicas de combinación de colores. En función de los colores a utilizar conviene utilizar lo más alejados posibles en el círculo cromático o bien utilizar colores que tengan una disposición de 120º los unos de los otros, de ahí el uso de Azul verdoso para el ActionBar con el Rojo de los FloatingActionButtons.
- El tamaño de la letra y el uso de mayúsculas. El uso de diferentes letras, tamaños o estilos utilizado como un recurso a la hora de llamar la atención. Se ha evitado en la medida de lo posible el uso de mayúsculas ya que puede resultar molesto o irrespetuoso.
- Proporcionar feedback al usuario sobre el estado del sistema en todo momento. Por cada acción del usuario se proporciona una respuesta, ya sea un éxito al guardar la plantilla o mostrando un mensaje con el error exacto que lo motivó, así como la posible solución para ello.

Decisiones tomadas

La mayoría de las decisiones que se tomaron durante el proyecto fueron para obtener una mejor experiencia de usuario, ya que el enfoque inicial que se hizo al diseñar los primeros prototipos de las interfaces y la usabilidad del sistema podían entorpecer el uso de la aplicación.

Control Spinner

Como se muestra en la figura de casos del caso de uso extendido “Crear Nueva Plantilla” (Ver Ilustración 136), la idea original era proporcionar al usuario un elemento gráfico de tipo dropdown desplegable (en Android denominado Spinner) con el cual podría indicar una serie de grupos musculares que caracterizasen el entrenamiento.

Al cargar la actividad de creación de plantilla el sistema realizaba una petición asíncrona con los grupos musculares disponibles, los cuales se le presentaban al usuario para personalizar su plantilla con unos grupos musculares u otros, guardándose en la base de datos mediante la API diseñada cuando el dispositivo Android lo requiriese.

Tras haber generado numerosas plantillas para realizar pruebas de los diferentes casos de prueba posibles se llegó a la conclusión de que en base lo aprendido en la asignatura HCI (Human-Computer Interaction) se podía apreciar que lastraba enormemente la fluidez con la que el usuario podía crear una tabla de entrenamiento. Además no tenía mucho sentido generar una plantilla con ejercicios que entrenen un grupo muscular (piernas) si se le indica finalmente mediante estos “tags” que el grupo muscular de la plantilla es otro (abdomen o brazo, por ejemplo).

Para dotar de mayor consistencia a la hora de generar una plantilla se eliminaron los elementos visuales de “Control Spinner” y el “ListView” con los grupos musculares añadidos.

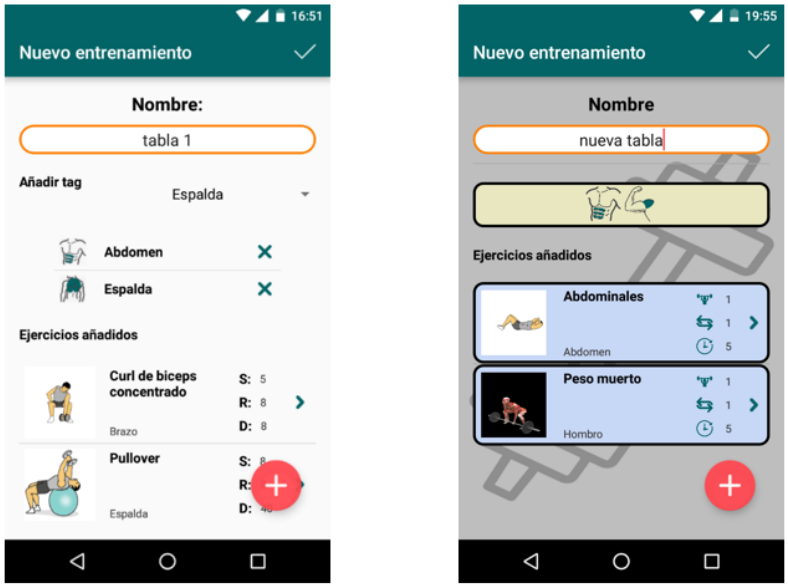


Ilustración 99: Eliminación del control Spinner

El nuevo diseño propuesto consiste en que a medida que el usuario añade o elimina ejercicios de la tabla de entrenamiento actual, un área que contiene unos contenedores de imágenes se cargan dinámicamente con los grupos musculares que tienen la plantilla. Además es indiferente el orden en el que se inserten ejercicios en la tabla, ya que las imágenes se cargan alfabéticamente y gestionando correctamente que se muestre una imagen por grupo muscular y ocultándolas correctamente si ya no quedan ejercicios de un determinado grupo muscular en la plantilla.

SearchView

El diseño inicial la aplicación iba a contar con una barra de búsqueda, de la cual disponen multitud de aplicaciones, aunque esta idea fue finalmente eliminada.

Al igual que en el caso anterior, la usabilidad de este sistema dejaba bastante que desear. Para poner en situación al lector, imagine que se quiere buscar un ejercicio en concreto, ¿no sería más lógico filtrar por grupo muscular que por el propio nombre? Es decir, si no se sabe el nombre exacto a buscar o partes de él, este tipo de búsqueda resulta inútil.

De la misma manera, al buscar entrenamientos ocurre lo mismo. Una plantilla tiene un nombre que un usuario le proporciona, pudiendo ser morfológicamente correctos (por ej: “rutina de entrenamiento lunes”) o bien siendo nombres no reconocibles (por ej: “plant3 v2”), por lo que seguramente no fuese posible encontrarlos mediante este tipo de búsqueda.

El que este sistema tuviese una usabilidad tan limitada fue motivo suficiente para desechar este sistema de búsqueda, ya que se pretende generar una experiencia de usuario ágil, en la que la cantidad de datos introducidos por teclado sea mínima.

Al diseño final basta con indicarle qué grupos musculares se quieren buscar mediante unos toggle buttons personalizados y presionar el tipo de búsqueda a realizar (ejercicios o plantillas de entrenamiento). Como se puede observar resulta un proceso mucho más rápido, su funcionamiento es más intuitivo y de esta manera se puede acceder a cualquier elemento por incoherente que sea su nombre.

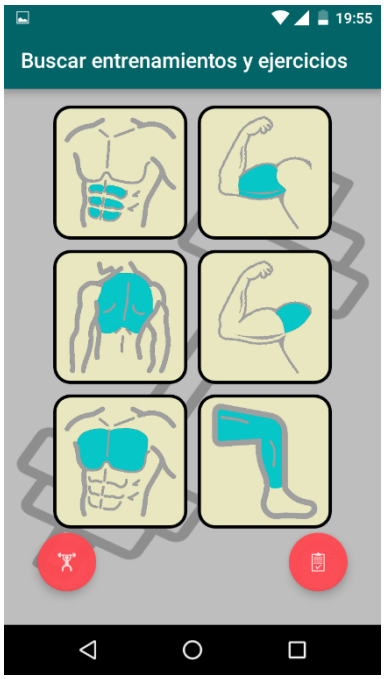


Ilustración 100: Buscador

Rotación de pantalla desactivada

Como se ha mencionado en la introducción sobre Actividades, al girar el dispositivo toda la información que estuviese almacenada se perdía (objetos generados desde que se inició la actividad, información mostrada en pantalla, etc).

Cuando se tiene una actividad como la de registro, inicio de sesión o incluso el listado de plantillas no existe problema, ya que existen dos métodos que se lanzan al girar el móvil (hay que recordar que muchas de las funcionalidades se gestionan por eventos).

En el primero, denominado *.onSaveInstanceState()*, se puede obtener el conjunto de parámetros de la actividad actual e introducir aquellos datos que no quieras perder, y en el segundo método, denominado *.onRestoreInstanceState()*,se pueden reasignar los valores a cada elemento, ya que se lanza después de haber creado la interfaz.

Recuperar los datos introducidos por el usuario es una tarea sencilla, ya que basta con acceder a los elementos de la UI y obtener sus valores. El problema es cuando se quiere guardar el estado de un objeto que se lanza en segundo plano, en este caso las tareas asíncronas AsyncTasks. Al no poder acceder al mismo para detectar su estado hasta que haya finalizado su ejecución, se producían errores si se giraba el dispositivo antes de recibir respuesta del servidor (tiene un timeout establecido de 5 segundos).

Como las aplicaciones deben mantener una cierta consistencia, no es lógico que ciertas actividades tengan su pantalla en posición vertical forzada, pudiendo estar en otras actividades en ambas posiciones, por eso se ha bloqueado el giro automático. Para ello se ha indicado en el archivo manifest.xml los siguientes atributos para cada actividad:

```
<activity android:name=".Activities.MiCuentaActivity"
    android:screenOrientation="portrait" />

<activity android:name=".Activities.TrainingActivity"
    android:screenOrientation="portrait"></activity>
```

Ilustración 101: Bloqueando la orientación del dispositivo

Reutilización

La aplicación ha sido diseñada de tal manera que se reutilice el código de las actividades.

Por ejemplo, la mayoría tiene un comportamiento diferente ya que no tiene nada que ver la actividad de registro con la de listar ejercicios. No obstante se puede acceder a los detalles de una plantilla desde diferentes actividades de la actividad: desde el apartado de mis plantillas, desde el buscador o desde favoritos, teniendo diferentes comportamientos.

Esto no quiere decir que se hayan hecho tres actividades diferentes, lo cual habría sido más cómodo, sino que se ha generado una única actividad “VerDetallesPlantilla” que, en función de los parámetros de entrada que se la hayan proporcionado se va a configurar su apariencia de una forma u otra, ocultando los botones de gestión si se encuentra en el buscador u ocultando los botones de añadir a favoritos si se encuentra en “MisPlantillas”.

El comportamiento de la actividad “VerDetallesEjercicio” sigue tiene un diseño similar, en el cual el botón de la ActionBar “X” puede alternar el fragment o salir de la actividad en función de la actividad de origen.

Shared preferences

Inicialmente se iban a almacenar en las preferencias diferentes datos referentes al usuario para no tener que pasar los valores entre Actividades continuamente pero al final se decidió almacenar únicamente el nick de usuario, el cual es único y no permite ser modificado.

El motivo de esta decisión fue que cuando se modifican los datos desde la página web, el dispositivo móvil debería recibir algún tipo de notificación push, avisándole de que los valores en SharedPreferences tendrían que ser modificados para sincronizarse correctamente con los datos introducidos en la web, pero si, como se ha mencionado anteriormente todas las acciones se realizan enviando el Nick de usuario, el cual es único e inalterable, no tenía sentido gestionar un sistema de notificaciones push el cual podría ahorrarse.

7. VERIFICACIÓN Y EVALUACIÓN

En este apartado se detallarán las pruebas que han sido realizadas para garantizar el buen funcionamiento de ambas aplicaciones.

Cabe recalcar que durante el proceso de implementación, debido a la experiencia obtenida durante la carrera, se han implementado los métodos con la intención de manejar todo tipo de situaciones, gracias a lo cual se ha simplificado enormemente el proceso de verificación y los resultados obtenidos han sido satisfactorios.

7.1. Plan de pruebas aplicación web

A continuación se enumeran las pruebas que se han realizado para comprobar el correcto funcionamiento de la aplicación web.

7.1.1- Registro web

Tabla 15: Pruebas unitarias Web: Registro 1/3

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
1	Varios intentos de registro dejando en blanco cada uno de los campos cada vez para comprobar la validación HTML5.	Validación HTML5 informando del error.	El informe de error de HTML5 se muestra donde se esperaba.	**Correcto.**
2	Intento de registro con nick de usuario con espacios.	Mostrar error vía HTML5. El nick de usuario no puede contener espacios.	Mensaje HTML5 detectando y mostrando el error al usuario.	**Correcto.**

Tabla 16: Pruebas unitarias Web: Registro 2/3

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
3	Introducir contraseñas con mayor o menos tamaño del establecido (8-16).	Informar del problema.	HTML5 muestra un mensaje con el error.	**Correcto.**
4	Contraseñas de registro desiguales.	Mensaje informando del problema.	Borrado de la contraseña secundaria e informe del problema.	**Correcto.**
5	Cuenta de correo introducida no tiene formato xxxx@yyy.zz o .zzz	Mostrar mensaje de error oportuno al usuario.	El mensaje muestra el error sobre el email.	**Correcto.**
6	Comprobar que no se pueden introducir caracteres extraños para evitar SQL injection vía HTML5.	Informar al usuario de que ha introducido caracteres indebidos.	Mensaje mostrado correctamente.	**Correcto.**
7	Varios intentos de registro dejando en blanco cada uno de los campos cada vez para comprobar la validación PHP.	Visualizar un DIV con el mensaje de error y el motivo de la generación del mismo.	Mensaje de error mostrado para cada uno de los campos del formulario.	**Correcto.**
8	Validación de nick de usuario, nombre y apellidos en PHP.	Informar de si dichos campos satisfacen las expresiones regulares de cada uno.	La validación detecta los fallos y los muestra al usuario en un mensaje.	**Correcto.**
9	Validación de contraseñas PHP.	Mostrar mensaje si contraseñas diferentes, vacías o incumplen la longitud establecida.	La validación detecta los fallos y los muestra al usuario en un mensaje.	**Correcto.**
10	Intentar registrar un usuario cuyo nombre de usuario o email ya están en uso.	El sistema no da de alta al usuario.	El sistema avisa al usuario sobre el error con un mensaje para que pueda poner solución a dicho problema.	**Correcto.**

Tabla 17: Pruebas unitarias Web: Registro 3/3

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
11	Registro de un usuario que no existe en la BD.	El usuario queda registrado.	El usuario se almacena en la BD, informa al usuario y redirige a la página principal.	**Correcto.**

- **Test 10:** Las primeras pruebas validan el formulario en el cliente vía HTML5 y JavaScript, pero para evitar posibles ataques en las pruebas marcadas se han desactivado esas formas de validación y se ha realizado un testeo en el servidor vía PHP.

7.1.2- Identificación web

Tabla 18: Pruebas unitarias Web: Identificación 1/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
12	Iniciar sesión sin rellenar alguno de los dos campos.	Impedir proceso e informar al usuario.	HTML5 muestra un mensaje del motivo del fracaso.	**Correcto.**
13	Desactivar validación HTML5 y JS e introducir caracteres no válidos.	Fracaso al iniciar sesión.	PHP ha devuelto el motivo del error y lo muestra en un mensaje.	**Correcto.**
14	Introducir una combinación de usuario y contraseña no válidos.	Fracaso al iniciar sesión.	Fracaso en la identificación y mostrado un mensaje con el motivo.	**Correcto.**
15	Introducir una combinación correo y contraseña no válidos.	Fracaso al iniciar sesión.	Fracaso en la identificación y mostrado un mensaje con el motivo.	**Correcto.**
16	Introducir combinación usuario y contraseña correctos.	Identificar al usuario correctamente.	Identificación exitosa y variable de sesión creada con el nick de usuario como referencia.	**Correcto.**

Tabla 19: Pruebas unitarias Web: Identificación 2/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
17	Introducir combinación de correo y contraseña correctos.	Identificar al usuario correctamente.	Identificación exitosa, buscado el nickUsuario asociado al correo y creada la variable de sesión en referencia a ello.	**Correcto.**

7.1.3- Crear nuevo ejercicio

Tabla 20: Pruebas unitarias Web: Crear nuevo ejercicio 1/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
18	Validación de HTML5 del formulario.	Informar de errores.	Validación de HTML5 muestra el error cometido.	**Correcto.**
19	Desactivar validación HTML5 y JavaScript. Comprobar la validación PHP.	Informar de errores.	La validación de PHP ha resultado exitosa.	**Correcto.**
20	Crear un ejercicio con una URL que no es de imgur, no acaba en .gif o .gifv o está vacía.	Impedir creación del ejercicio e informar del motivo al usuario.	PHP ha detectado que el formato no es válido para mostrarlo en HTML5 y ha informado al usuario.	**Correcto.**
21	Intentar crear un ejercicio cuyo nombre ya está en uso.	No permitir la creación e informar de ello.	La llamada AJAX ha detectado, ha impedido la creación del ejercicio y ha mostrado el error.	**Correcto.**
22	Crear un ejercicio sin descripción pero con los datos rellenados correctamente.	Creación de ejercicio exitosa.	El ejercicio se ha almacenado en la base de datos.	**Correcto.**
23	Crear un ejercicio con todos los datos rellenados correctamente.	Creación de ejercicio exitosa.	El ejercicio se ha almacenado en la base de datos.	**Correcto.**

Tabla 21: Pruebas unitarias Web: Crear nuevo ejercicio 2/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
24	Crear un ejercicio cuya descripción contiene caracteres extraños (cualquiera salvo signos de puntuación normales).	Impedir la creación e informar al usuario del error.	Validación realizada mediante JavaScript y PHP ha detectado el fallo y ha mostrado un error paralizando el proceso de creación de ejercicio.	**Correcto.**
25	Crear un ejercicio cuya descripción excede el límite establecido (200 caracteres).	Impedir la creación e informar al usuario del error.	La validación de PHP y JavaScript detecta la excesiva longitud y ha mostrado un mensaje además de paralizar el proceso.	**Correcto.**

- **Test 24 y 25:** HTML5 no contiene soporte para validar TextArea, por lo cual es imposible asignarle un mensaje de error de tipo .customValidity() al igual que otros elementos de HTML5, así que su validación ha sido realizada únicamente vía JavaScript (cliente) y PHP (servidor).

7.1.4- Modificar ejercicio

Tabla 22: Pruebas unitarias Web: Modificar ejercicio 1/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
26	Modificar la url de la imagen por una incorrecta.	Validación HTML5 muestra error.	Ventana Modal de Bootstrap muestra error.	**Correcto.**
27	Modificar la descripción añadiendo caracteres extraños (signos de puntuación y comillas válidas) o excediendo longitud de 200 caracteres.	Validación JavaScript detecta error e informa al usuario.	Ventana Modal de Bootstrap muestra error.	**Correcto.**

Tabla 23: Pruebas unitarias Web: Modificar ejercicio 2/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
28	Desactivar validación HTML5 e introducir url/descripción incorrectas.	Validación PHP debería paralizar proceso e informar en la llamada AJAX.	Proceso interrumpido y ventana Modal de Bootstrap ha informado del error.	**Correcto.**
29	Modificar url de la imagen por una url válida.	La url de la imagen debería actualizarse.	Ejercicio actualizado y mensaje modal de éxito mostrado.	**Correcto.**
30	Modificar la descripción del ejercicio por una válida.	La descripción del ejercicio debería actualizarse.	Ejercicio actualizado y mensaje modal de éxito mostrado.	**Correcto.**
31	Acceder al innerHTML del nombre del ejercicio desde la consola, modificarlo de acuerdo al ejercicio de otro usuario e intentar actualizarlo.	Impedir el proceso de actualización.	PHP ha detectado que el ejercicio que obtiene como parámetro no está asignado al usuario con el que se ha iniciado sesión, ha bloqueado la actualización e informado de los motivos del mismo.	**Correcto.**

- **Test 31:** Resulta improbable que un usuario común active este caso de prueba, pero se ha considerado que a día de hoy, si se quiere proporcionar un extra de seguridad hay que realizar comprobaciones tan improbables como esta, ya que desafortunadamente es muy común frecuentar lo que se denomina “comunidad tóxica” en muchas plataformas de internet, las cuales buscan perjudicar el funcionamiento de la misma.

7.1.5- Eliminar ejercicio

Tabla 24: Pruebas unitarias Web: Eliminar ejercicio

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
32	Intentar eliminar cualquier ejercicio que esté en uso por otro usuario.	Fracaso en la eliminación.	El sistema no elimina el ejercicio e informa al usuario de que el ejercicio está en uso.	**Correcto.**
33	Intentar eliminar un ejercicio que no está en uso por nadie.	Eliminar el ejercicio de la BD.	Se ha eliminado el ejercicio de la BD e informado al usuario.	**Correcto.**
34	Acceder al innerHTML del nombre del ejercicio desde la consola, modificarlo de acuerdo al ejercicio de otro usuario e intentar eliminarlo.	Fracaso en la eliminación.	PHP ha detectado que el ejercicio que obtiene como parámetro no está asignado al usuario con el que se ha iniciado sesión, ha impedido la eliminación e informado de los motivos del mismo.	**Correcto.**
35	Acceder al innerHTML de todos los campos del ejercicio salvo el nombre desde la consola, modificarlos e intentar eliminarlo.	Eliminar el ejercicio.	El ejercicio se ha eliminado.	**Correcto.**

- **Test 35:** Al eliminar un ejercicio el PHP valida únicamente si el nombre del ejercicio está vinculado a la variable de sesión actual, por lo cual el resto de la información la he considerado irrelevante. Habrá quien opine que es mejor comprobar todos y cada uno de los campos, pero dado que el sistema está desarrollado para que exista únicamente un ejercicio con un nombre específicos no debería haber problema.

7.1.6- Modificar cuenta

Tabla 25: Pruebas unitarias Web: Modificar cuenta 1/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
36	Introducir caracteres inválidos en cada uno de los campos del formulario.	Impedir modificación e informar.	La validación HTML5 ha detectado el error e informa del mismo.	**Correcto.**
37	Modificar uno o varios campos con valores correctos y no introducir contraseña.	Actualizar información.	Actualización de los valores proporcionados y contraseña no modificada.	**Correcto.**
38	Proporcionar unos valores correctos para actualizar contraseñas (y si se quiere también el resto de casillas).	Actualizar información.	Actualización de los valores proporcionados.	**Correcto.**
39	Proporcionar una contraseña cuya longitud está fuera de límites o cuyos caracteres no están permitidos.	Avisar del error.	Validación de HTML5 ha impedido el proceso e informado al usuario.	**Correcto.**
40	Proporcionar contraseñas diferentes.	No actualizar e informar.	Validación de HTML5 ha impedido el proceso e informado al usuario.	**Correcto.**
41	Desactivar validación HTML5 Y JavaScript, proporcionando una contraseña cuya longitud está fuera de límites o cuyos caracteres no están permitidos.	No actualizar e informar.	La validación PHP ha detectado el error mostrándoselo al usuario e interrumpiendo el proceso.	**Correcto.**

Tabla 26: Pruebas unitarias Web: Modificar cuenta 2/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
42	Desactivar validación HTML5 Y JavaScript, proporcionando dos contraseñas diferentes.	No actualizar e informar.	La validación PHP ha detectado el error mostrándoselo al usuario e interrumpiendo el proceso.	**Correcto.**

7.1.7- Eliminar cuenta

Tabla 27: Pruebas unitarias Web: Eliminar cuenta

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
43	Intentar eliminar cuenta sin proporcionar contraseña.	Mostrar mensaje de error.	La validación de HTML5 impide el proceso.	**Correcto.**
44	Introducir una contraseña incorrecta.	Proceso fallido.	PHP detecta que la combinación usuario y contraseña son incorrectas impidiendo el proceso e informando al usuario.	**Correcto.**
45	Proporcionar la contraseña correcta.	Eliminar cuenta.	La cuenta se ha eliminado. Todos los ejercicios son propiedad de una figura llamada "Time To Train" y todo rastro de la cuenta queda eliminado de la BD.	**Correcto.**

7.1.8- Cerrar sesión

Tabla 28: Pruebas unitarias Web: Cerrar sesión

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
46	Pulsar en el icono de cerrar sesión.	Cerrar la sesión actual.	Cerrar la sesión y redirigir a la principal.	**Correcto.**
47	Tras cerrar sesión intentar navegar hacia atrás (backspace o atrás en el navegador)	Impedir proceso.	Al haber eliminado la variable de sesión, un script inicial detecta que no existe e informa al usuario de que no puede acceder a esa zona de la web.	**Correcto.**

7.2. Plan de pruebas aplicación Android

A continuación se enumeran las pruebas que se han realizado para corregir los errores que se han presentado durante el desarrollo de la aplicación Android y su corrección.

7.2.1- Identificación App

Tabla 29: Pruebas unitarias App: Identificación 1/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
48a	Intentar iniciar sesión sin conexión a internet.	Avisar al usuario que no posee conexión y cancelar proceso.	Excepción detectada.	**Incorrecto.**
48b	Intentar iniciar sesión sin conexión a internet.	Avisar al usuario que no posee conexión y cancelar proceso.	Se activa el dialgo correspondiente para advertir al usuario.	**Correcto.**
49	Intentar iniciar sesión con WIFI.	Permitir proceso.	Se ha intentado iniciar sesión.	**Correcto.**

Tabla 30: Pruebas unitarias App: Identificación 2/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
50	Intentar iniciar sesión con tarifa de datos.	Permitir proceso.	Se ha intentado iniciar sesión.	**Correcto.**
51	Iniciar sesión sin rellenar alguno o los dos datos.	Error en el inicio de sesión.	No se procede al inicio de sesión y se avisa al usuario del motivo.	**Correcto.**
52a	Introducir ambos campos (erróneos) e intentar iniciar sesión.	Error en el inicio de sesión.	Excepción detectada.	**Incorrecto.**
52b	Introducir ambos campos (erróneos) e intentar iniciar sesión.	Error en el inicio de sesión.	El sistema no inicia sesión.	**Correcto.**
53a	Realizar una prueba de inicio de sesión enviando datos al servidor y esperando su respuesta.	Mostrar mensaje de vuelta.	Excepción detectada.	**Incorrecto.**
53b	Realizar una prueba de inicio de sesión enviando datos al servidor y esperando su respuesta.	Mostrar mensaje de vuelta.	Se ha recibido el mensaje de vuelta y se ha mostrado.	**Correcto.**
54	Introducir ambos campos (erróneos) e intentar iniciar sesión proporcionando el nick de usuario.	Iniciar sesión guardando el nick de usuario.	Inicio de sesión exitoso. Se ha detectado/guardado el nick de usuario.	**Correcto.**
55	Introducir ambos campos (correctos) e intentar iniciar sesión proporcionando la cuenta de correo.	Iniciar sesión guardando el nick de usuario.	Inicio de sesión exitoso. Se ha detectado/guardado el nick de usuario asociado a la cuenta proporcionada en el inicio de sesión.	**Correcto.**

- **Test 48:** El error residía en cambios realizados en las APIs de Google y los diferentes métodos que van quedando obsoletos (deprecated), los cuales recomiendan dejar de utilizar a la hora de programar. Tras modificar algunos de los métodos asociados a la

detección de conexiones de internet del dispositivo se pudo continuar con el resto de pruebas.

- **Test 52:** El error residía en que tras realizar pruebas en otros proyectos para no "ensuciar" la programación del TFG, se olvidó especificar en el archivo manifest.xml de Google que se necesitan permisos de internet para la aplicación, lo cual lanzaba una excepción.
- **Test 53:** La estructura del JSON devuelto en este caso difería del resto de JSON, por lo cual intentaba acceder a un atributo que no existía y lanzando una excepción. Normalizando todos los JSON se evitó el error.

7.2.2- Registro App

Tabla 31: Pruebas unitarias App: Registro 1/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
56a	Registro con datos incorrectos.	Mensaje de error.	Sin conexión al servidor.	**Incorrecto.**
56b	Registro con datos incorrectos.	Mensaje de error.	Se avisa al usuario de que los campos tienen fallos.	**Correcto.**
57	Registro de usuario dejando vacíos los campos.	Debería impedirse la petición HTTP.	Se han detectado los campos vacíos y se ha mostrado un error en pantalla avisando del motivo.	**Correcto.**
58	Registro introduciendo caracteres especiales o espacios en el nick de usuario.	Registro cancelado.	El registro no se ha llevado a cabo y se muestra al usuario el problema.	**Correcto.**
59	Los dos campos de contraseñas son diferentes.	Registro cancelado.	Se ha cancelado el registro, mostrando un mensaje que indica que las contraseñas son diferentes y el campo de la segunda contraseña se ha borrado.	**Correcto.**

Tabla 32: Pruebas unitarias App: Registro 2/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
60	El nombre y/o apellido contienen tildes.	Validación correcta.	El proceso de registro ha continuado.	**Correcto.**
61	El nombre y/o el apellido contienen espacios o caracteres especiales.	Validación incorrecta.	El registro no se ha llevado a cabo y se muestra al usuario el problema.	**Correcto.**
62a	Todos los datos introducidos son correctos.	Registro correcto y redirección a la actividad principal con sesión guardada.	Se ha cerrado la aplicación.	**Incorrecto.**
62b	Todos los datos introducidos son correctos.	Registro correcto y redirección a la actividad principal con sesión guardada.	Se redirecciona a la actividad principal, pero mantiene la anterior activity en la pila.	**Incorrecto.**
62c	Todos los datos introducidos son correctos.	Registro correcto y redirección a la actividad principal con sesión guardada.	Se han eliminado las actividades de la pila y se ha redireccionado a la actividad principal.	**Correcto.**

- **Test 56:** Un fallo en la asignación de parámetros en el archivo .php hacía que no se devolviese el objeto JSON, por lo cual el dispositivo mostraba siempre un mensaje "No se pudo recibir respuesta del servidor". Finalmente se volvió a rediseñar el web service de registro paso por paso con dos ficheros auxiliares .html y .js para simular más rápidamente las llamadas al web service y depurar más fácilmente los errores que se puedan dar en el futuro en otras peticiones HTTP.
- **Test 62:** Al registrar el usuario la pila de actividades actual (Main->Registro) debería borrarse y guardar las preferencias, pero no se borraban y se cerraba la aplicación en un primer momento. Indagando en StackOverFlow se pudo localizar un código (http://stackoverflow.com/a/20113162) que permitió eliminar las actividades en el backstack y guardar el nombre con el cual se identificaría la sesión actual.

7.2.3- Crear nueva plantilla

Tabla 33: Pruebas unitarias App: Crear nueva plantilla 1/3

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
63a	Crear un nuevo entrenamiento rellenando los 3 atributos de la futura plantilla (nombre, tags y ejercicios).	Debería activarse el botón de guardado (check en la toolbar).	El botón de guardado no se ha activado.	**Incorrecto.**
63b	Crear un nuevo entrenamiento rellenando los 3 atributos de la futura plantilla (nombre, tags y ejercicios).	Debería activarse el botón de guardado (check en la toolbar).	El botón de guardado ahora se muestra cuando es necesario.	**Correcto.**
64	Crear una plantilla sin añadir Tags.	No debería permitir el guardado.	El algoritmo de comprobación de "plantilla correcta" impide el guardado al no mostrar el "check" de guardado.	**Correcto.**
65	Crear una plantilla sin introducir nombre.	No debería permitir el guardado.	El algoritmo de comprobación de "plantilla correcta" impide el guardado al no mostrar el "check" de guardado.	**Correcto.**
66	Crear una plantilla con algún ejercicio sin configurar (descanso, series o repeticiones = -1).	No debería permitir el guardado.	El algoritmo de comprobación de "plantilla correcta" impide el guardado al no mostrar el "check" de guardado.	**Correcto.**
67	Crear una plantilla sin ejercicios.	No debería permitir el guardado.	El algoritmo de comprobación de "plantilla correcta" impide el guardado al no mostrar el "check" de guardado.	**Correcto.**
68	Eliminar un ejercicio de la plantilla a crear.	Debería reflejarse en el listado.	Se ha eliminado correctamente y se ha actualizado el listado.	**Correcto.**

Tabla 34: Pruebas unitarias App: Crear nueva plantilla 2/3

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
69a	Modificar los datos de un ejercicio añadido en la plantilla.	El ejercicio de la plantilla a crear debería modificarse.	La configuración del ejercicio no se ha modificado	**Incorrecto.**
69b	Modificar los datos de un ejercicio añadido en la plantilla.	El ejercicio de la plantilla a crear debería modificarse.	Los datos del ejercicio han sido actualizados.	**Correcto.**
70	Añadir un Tag ya añadido a la plantilla.	El Tag no debería añadirse.	El Tag no se ha añadido.	**Correcto.**
71	Eliminar un Tag de la plantilla.	El Tag debería eliminarse.	El Tag se ha eliminado de la lista de Tags.	**Correcto.**
72	Intentar crear una plantilla con un nombre no válido (SQL Injection)	El ejercicio no debería guardarse.	No se ha guardado y se le ha informado al usuario del motivo.	**Correcto.**
73	Fase de guardado 1: Almacenar datos en la tabla "plantillas".	Debería almacenarse su nombre, el propietario y proporcionar un id automático incremental.	Se han cumplido las expectativas.	**Correcto.**
74	Fase de guardado 2: Almacenar los Tags de la plantilla en la tabla "tagsplantilla".	Deberían almacenarse todos los Tags de la plantilla a guardar.	Los Tags se han almacenado correctamente.	**Correcto.**
75	Fase de guardado 3: Almacenar la configuración de los ejercicios en la tabla "configuración ejercicio".	Deberían almacenarse las repeticiones, las series y los descansos de cada ejercicio.	Se ha generado una entrada de configuración por cada ejercicio.	**Correcto.**
76	Fase final de guardado: Probar a guardar una plantilla completa.	Deberían guardarse los datos en las tablas oportunas.	Todas las tablas se han rellenado con los datos correctamente	**Correcto.**
77	Guardado de plantilla con las redes inalámbricas desconectadas.	Debería mostrarse un mensaje de error.	Se muestra un DialogFragment que le da al usuario la posibilidad de acceder a ajustes de red. La actividad no se cierra.	**Correcto.**

Tabla 35: Pruebas unitarias App: Crear nueva plantilla 3/3

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
78	Guardado de plantilla con el servidor desconectado.	Debería mostrarse un mensaje de error.	Se muestra un DialogFragment con el motivo del error. La actividad no se cierra.	**Correcto.**

- **Test 63:** El algoritmo que comprueba si una plantilla es correcta contenía un fallo de diseño al ejecutar la comprobación de si la lista de ejercicios resultaba válida. Tras su modificación todo funcionó correctamente.
- **Test 69:** El problema radicaba en que el método de la clase Ejercicio que sustituía los datos de un ejercicio por otro fallaba, ya que inicialmente únicamente se usaba para intercambiar nombre, url de la imagen, descripción y el grupo muscular en la lista de ejercicios al realizar una ordenación. Posteriormente se adaptó para que tuviese otros atributos como nº de repeticiones, nº de series, descanso y orden, pero se olvidó sustituir estos valores, por lo cual no se modificaba bien el ejercicio. Tras solucionarlo, identificaba correctamente los ejercicios y su modificación era posible.
- **Test 76:** Para avanzar sobre seguro, la inserción de datos en las tablas se hizo por partes, no teniendo (afortunadamente) errores al haber trabajado con PHP en la primera mitad del proyecto al desarrollar la herramienta web.

7.2.4- Ver listado ejercicios

Tabla 36: Pruebas unitarias App: Ver listado ejercicios 1/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
79	Acceder al listado con las redes inalámbricas desconectadas.	Debería mostrarse un aviso al usuario y finalizar la actividad.	Se muestra un DialogFragment que le da al usuario la posibilidad de acceder a ajustes de red. La actividad se cierra enviando al usuario atrás.	**Correcto.**

Tabla 37: Pruebas unitarias App: Ver listado ejercicios 2/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
80	Acceder al listado de ejercicios con el servidor desconectado.	Debería mostrar un mensaje de "sin respuesta".	Se muestra un DialogFragment que informa al usuario de los errores de conexión y la actividad no se cierra.	**Correcto.**
81	Acceder al listado con ambas partes activas (servidor y cliente).	Debería cargar el listado.	El listado se ha cargado y se muestra correctamente.	**Correcto.**
82a	Ordenar los elementos del listado (orden o grupo muscular).	Deberían ordenarse de acuerdo al criterio.	Algunos ejercicios se han duplicado, otros han desaparecido.	**Incorrecto.**
82b	Ordenar los elementos del listado (orden o grupo muscular).	Deberían ordenarse de acuerdo al criterio.	Sólo se han ordenado los nombres de los ejercicios, el resto de atributos no han sido intercambiados.	**Incorrecto.**
82c	Ordenar los elementos del listado (orden o grupo muscular).	Deberían ordenarse de acuerdo al criterio.	Los datos se han ordenado correctamente.	**Correcto.**
83	Ordenar los elementos del listado por nombre.	Deberían ordenarse por nombre.	Los elementos se han ordenado correctamente.	**Correcto.**
84	Ordenar los elementos del listado por grupo muscular.	Deberían ordenarse por grupo muscular.	Se ordenan únicamente en castellano.	**Aceptable.**
85	Ver los detalles de un ejercicio.	Deberían visualizarse los datos del ejercicio.	Se ha generado una activity para que el usuario visualice los datos.	**Correcto.**

- **Test 79:** El cierre de la actividad se da para que no cargue una interfaz en blanco, sin listado que mostrar.
- **Test 80:** La actividad no se cierra ya que da la posibilidad al usuario de refrescar la página.
- **Test 82:** El algoritmo de ordenación QuickSort implementado sustituía únicamente los nombres de los ejercicios, no los atributos. Tras incluir el método *sustituir()* en la clase *Ejercicio* la ordenación fue satisfactoria.
- **Test 84:** Aunque la aplicación está pensada para desarrollarla en castellano, se ha intentado traducir todo aquello posible al inglés. El mapeo de traducciones realizado

se ejecuta de forma que recibiendo el dato de la DB, se traduce dinámicamente en ejecución al visualizarlo. El problema es que, recibiendo los nombres en castellano, la ordenación se realiza en tal idioma, de manera que, por ejemplo, Chest aparece después de Shoulder, ya que en castellano son Hombro y Pecho respectivamente. Como la traducción ha sido algo secundario que se pensó en incluir durante el desarrollo de la aplicación, su solución será algo que forme parte del proyecto futuro.

7.2.5- Ver detalles de ejercicio

Tabla 38: Pruebas unitarias App: Ver detalles ejercicio 1/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
86	Cargar Gif de la imagen.	Debería mostrarse en el ImageView destinado.	El Gif se carga correctamente.	**Correcto.**
87	Entrar desde la creación de plantillas.	Debería permitir configurar sus atributos para añadirlo a la plantilla actual.	Permite acceder al fragment de configuración.	**Correcto.**
88a	Comprobación de mantenimiento de la configuración durante la creación de plantillas al cambiar el fragment.	Una vez establecidos los valores de series/ repeticiones/ descanso, si se alterna de fragment y se vuelve a la configuración deberían mantenerse los datos.	Al volver al fragment de configuración, los valores (numberPickers) se sitúan en sus valores por defecto y la configuración se pierde.	**Incorrecto.**
88b	Comprobación de mantenimiento de la configuración durante la creación de plantillas al cambiar el fragment.	Una vez establecidos los valores de series/ repeticiones/ descanso, si se alterna de fragment y se vuelve a la configuración deberían mantenerse los datos.	Al instanciar de nuevo el fragment éste es capaz de recordar los datos introducidos por el usuario la última vez.	**Correcto.**
89	Entrar desde los detalles de una plantilla con el "modo configuración" desactivado.	Debería mostrar únicamente el fragment con las características del ejercicio.	Se muestra la descripción técnica del ejercicio.	**Correcto.**

Tabla 39: Pruebas unitarias App: Ver detalles ejercicio 2/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
90	Entrar desde los detalles de una plantilla con el "modo configuración" habilitado.	Debería mostrar los valores con los que el ejercicio se configuró.	Se muestra el fragment con los parámetros de configuración del ejercicio.	**Correcto.**
91	Entrar en los detalles desde el apartado del buscador.	Debería mostrar la descripción técnica.	Sólo se muestra la información base del ejercicio.	**Correcto.**

- **Test 88:** Al inicio el algoritmo planteado era erróneo y tuvo que sobrescribirse un método base de los NumberPickers para que al modificar sus valores (onChange) activasen una interface que avisara a la Activity sobre la que se insertan los fragments y guardase esos valores como atributos y no perderlos.

7.2.6- Ver detalles plantilla

Tabla 40: Pruebas unitarias App: Ver detalles plantilla 1/3

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
92	El modo de configuración solo debería estar habilitado desde "Mis plantillas".	Debería estar habilitado el modo configuración.	El modo configuración de plantilla es visible y es posible alternar entre activado y desactivado.	**Correcto.**
93	Habilitar el añadido de ejercicios.	Al alternar al modo configuración debería habilitarse el Floating Action Button de añadir ejercicios.	El botón de añadido se muestra al usuario correctamente.	**Correcto.**
94	Ver detalles de un ejercicio.	Si la plantilla es nuestra debería permitirse la modificación de datos. Sino únicamente visualizar la descripción.	El fragment mostrado es el adecuado (detalles o configuración).	**Correcto.**

Tabla 41: Pruebas unitarias App: Ver detalles plantilla 2/3

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
95	Borrar una plantilla de la que se es propietario.	Debería eliminarse la plantilla y guardarse los datos en la DB.	La plantilla ya no forma parte del sistema.	**Correcto.**
96	Modificar el nombre de la plantilla.	La nueva estructura de la plantilla debería actualizarse en la DB.	La plantilla se ha actualizado correctamente.	**Correcto.**
97a	Modificar la configuración de algún ejercicio de una plantilla.	La nueva estructura de la plantilla debería actualizarse en la DB.	Se ha bloqueado la aplicación	**Incorrecto.**
97b	Modificar la configuración de algún ejercicio de una plantilla.	La nueva estructura de la plantilla debería actualizarse en la DB.	La plantilla se ha actualizado correctamente.	**Correcto.**
98	Eliminar ejercicios de una de las plantillas.	Debería pedir la confirmación del usuario con un DialogFragment.	Se muestra el dialog correctamente.	**Correcto.**
99a	Eliminar ejercicios de una de las plantillas.	La nueva estructura de la plantilla debería actualizarse en la DB.	Plantilla actualizada correctamente, pero si el ejercicio a eliminar era el único de un grupo muscular la visualización es incorrecta.	**Incorrecto.**
99b	Eliminar ejercicios de una de las plantillas.	La nueva estructura de la plantilla debería actualizarse en la DB.	La plantilla se ha actualizado correctamente en la DB y se visualizan correctamente los tags.	**Correcto.**
100	Añadir un nuevo ejercicio en la plantilla.	La nueva estructura de la plantilla debería actualizarse en la DB.	La plantilla se ha actualizado correctamente.	**Correcto.**
101	Eliminar la plantilla actual.	La plantilla debería eliminarse de la DB y todas sus referencias.	La plantilla se ha eliminado correctamente.	**Correcto.**

Tabla 42: Pruebas unitarias App: Ver detalles plantilla 3/3

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
102	Cargar icono de favoritos.	El icono cargado debería corresponderse con la situación (si el usuario tiene o no la plantilla en favoritos).	El icono se carga en consecuencia correctamente.	**Correcto.**

- **Test 97:** El bloqueo se debía a que el funcionamiento de esta activity está hecho de tal manera que si se intenta modificar el ejercicio en la DB y ocurre un error hay que recuperar el estado anterior de la plantilla mediante una copia la cual se genera al modificar la plantilla. Si la plantilla era eliminada nada más entrar ocurría un error al intentar recuperar un estado el cual estaba sin inicializar. Tras gestionar este error con una variable booleana se pudo corregir el funcionamiento.
- **Test 99:** Este curioso error tuvo fácil detección y solución ya que el funcionamiento es idéntico a la creación de plantillas, en la cual se pueden eliminar ejercicios igualmente y cuya casuística se tuvo en cuenta, el problema fue que se olvidó incorporar 6 líneas de código que solucionan este error. Dicho error consiste en que al visualizar una plantilla de, por ejemplo, una lista de 4 tags de grupos musculares, se cogen 4 elementos ImageView y se pone su propiedad en visible. Al eliminar un tag se recarga la lista, pero esta vez solo carga los 3 elementos necesarios, dejando el 4º aún visible. La solución fue implementar un segundo loop que siguiera en el último elemento con una imagen asignada y que los restantes pusiera su propiedad de visibilidad a Gone (la diferencia entre Gone e Invisible es similar a la de JavaScript entre 'Hidden' y 'None'. Mientras que 'Invisible' y 'Hidden' ocupan un espacio que realmente no se ve, 'Gone' y 'None' simulan que el objeto no existe, variando la distribución de los componentes y logrando mejores resultados.

7.2.7- Buscador

Tabla 43: Pruebas unitarias App: Buscador 1/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
103	Realizar búsqueda con las redes inalámbricas desconectadas.	Debería mostrarse un DialogFragment informando al usuario de la situación permitiéndole activarlas.	Se lanza un dialog para que el usuario active los datos o bien salga de esta actividad.	**Correcto.**
104	Realizar una búsqueda con el servidor desconectado.	Debería informarse al usuario del problema.	Se muestra un dialog informando de la situación y se vuelve a la actividad de búsqueda.	**Correcto.**
105	Buscar ejercicios.	Deberían listarse los ejercicios.	Se muestra el listado con todos los ejercicios de la DB.	**Correcto.**
106	Acceder a los detalles de un ejercicio	Deberían visualizarse las características del ejercicio sin estar habilitados los elementos de configuración.	Al usuario se le muestra únicamente la descripción del ejercicio, sin posibilidad de acceder al menú de configuración.	**Correcto.**
107a	Buscar plantillas.	Deberían visualizarse correctamente los datos de las plantillas.	El propietario mostrado es incorrecto. Se muestra el propio usuario del dispositivo como propietario a pesar de cargarse plantillas ajenas.	**Incorrecto.**
107b	Buscar plantillas.	Deberían visualizarse correctamente los datos de las plantillas.	Los datos visualizados son correctos.	**Correcto.**
108	Buscar plantillas.	Deberían listarse las plantillas que coincidan con los especificados por el usuario.	Se muestran las plantillas que el usuario ha solicitado.	**Correcto.**

Tabla 44: Pruebas unitarias App: Buscador 2/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
109	Acceder a los detalles de la plantilla	Deberían estar habilitados lo botones de favoritos e iniciar entrenamiento únicamente.	Las únicas opciones disponibles son favoritos e iniciar entrenamiento.	**Correcto.**
110a	Añadir a favoritos una plantilla buscada.	La relación usuario - idPlantilla de favoritos debería almacenarse en la DB.	La relación se almacena correctamente en la DB, pero en la activity con el listado no se ha reflejado este cambio.	**Incorrecto.**
110b	Añadir a favoritos una plantilla buscada.	La relación usuario - idPlantilla de favoritos debería almacenarse en la DB.	Se ha almacenado la relación usuario - idPlantilla y en el listado se muestra la estrella de favoritos sobre la plantilla pertinente.	**Correcto.**
111	Cambiar los filtros de búsqueda.	Se debería rehacer la búsqueda mostrando aquellos elementos que satisfagan los filtros.	Se muestran exactamente los elementos solicitados.	**Correcto.**

- **Test 107:** La causa de este fallo es un fallo de diseño en el fichero PHP, en el cual la variable en la cual se almacena el propietario de cada plantilla se guardaba con el nick del usuario demandante de la petición del listado de plantillas, en vez del nick del propietario real de la plantilla.
- **Test 110:** Al añadir la plantilla a favoritos la propia actividad "ver detalles plantilla" sí que respondía a este cambio, pero la actividad anterior, la cual contiene el listado completo de plantillas del buscador, no recibía el aviso. Por ello hubo que modificar el intent a un "startActivityForResult" con el cual se pudo recargar el listado al detectar el cambio.

7.2.8- Favoritos

Tabla 45: Pruebas unitarias App: Favoritos

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
112	Acceder a favoritos con las redes inalámbricas desconectadas.	Debería avisar al usuario sobre la activación de las redes inalámbricas.	Un dialog insta al usuario a activar los datos.	**Correcto.**
113	Acceder a favoritos con el servidor desconectado.	Debería avisar al usuario de los problemas de conexión.	Se lanza un dialog advirtiendo al usuario de problemas en el servidor.	**Correcto.**
114	Cargar listado de favoritos.	Deberían cargarse los listados correctamente.	Se han cargado correctamente todas las plantillas.	**Correcto.**
115	Ver detalles de una plantilla.	Debería generarse una actividad con los detalles de la plantilla para que el usuario pueda iniciar entrenamiento o ver detalles.	El funcionamiento es el deseado.	**Correcto.**
116	Eliminar de favoritos.	Debería desactivarse el icono de favoritos así como recargar el listado de la actividad anterior.	El icono se ha desactivado y el listado ya no contiene la plantilla desechada.	**Correcto.**

7.2.9- Mi cuenta

Tabla 46: Pruebas unitarias App: Mi cuenta 1/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
117	Modificar perfil introduciendo caracteres no válidos en cada uno de los campos EditText.	Impedir modificación e informar al usuario.	No se han actualizado los datos y se ha informado al usuario.	**Correcto.**

Tabla 47: Pruebas unitarias App: Mi cuenta 2/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
118	Modificar perfil introduciendo contraseñas diferentes.	Impedir modificación e informar al usuario.	No se han actualizado los datos y se ha informado al usuario.	**Correcto.**
119	Proporcionar longitudes fuera del rango permitido en alguno de los campos.	Informar al usuario del problema.	No se han actualizado los datos y se ha informado al usuario.	**Correcto.**
120a	Modificar perfil proporcionando valores correctos.	Actualizar información.	La información no se ha actualizado	**Incorrecto.**
120b	Modificar perfil proporcionando valores correctos.	Actualizar información.	La información ha sido actualizada.	**Correcto.**
121	Modificar perfil proporcionando valores correctos.	Actualizar información.	La información se ha actualizado.	**Correcto.**
122	Modificar perfil proporcionando valores correctos, salvo el correo proporcionado, el cual está en uso por otro usuario.	No actualizar e informar de la situación.	La información no se ha actualizado	**Correcto.**
123	Cerrar sesión.	Eliminar las preferencias asociadas al usuario y enviar a la actividad principal.	El funcionamiento es correcto.	**Correcto.**

- **Test 120:** Si se proporcionaba la antigua cuenta de correo el resto de datos no se actualizaba ya que detectaba que un usuario (yo mismo) la tenía en uso. Tras un cambio en el php para obviar que la cuenta es la del propio usuario se pudieron actualizar correctamente los datos.

7.2.10- Iniciar entrenamiento

Tabla 48: Pruebas unitarias App: Iniciar entrenamiento 1/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
124	Probar el ritmo de las repeticiones.	Cada 2 segundos debería sumar 1 repetición al ejercicio.	Las repeticiones tienen el ritmo deseado.	**Correcto.**
125	Comprobar el tiempo de descanso.	Al finalizar las repeticiones debería comenzar el tiempo de descanso.	El descanso se ha activado correctamente.	**Correcto.**
126a	Comprobar aumento de serie.	Al finalizar el descanso debería comenzar la siguiente serie.	El número de serie no se ha mostrado correctamente.	**Incorrecto.**
126b	Comprobar aumento de serie.	Al finalizar el descanso debería comenzar la siguiente serie.	El número de serie aumenta, pero no cambia de ejercicio.	**Incorrecto.**
126c	Comprobar aumento de serie.	Al finalizar el descanso debería comenzar la siguiente serie.	Se actualiza correctamente el número de serie actual y cambia de ejercicio.	**Correcto.**
127	Inicializar siguiente ejercicio.	Al finalizar las series de un ejercicio debería continuar con el siguiente.	El ejercicio se ha iniciado correctamente.	**Correcto.**
128	Pausa de entrenamiento.	Al pulsar el botón de pausa el entrenamiento debería pausarse.	El contador se ha pausado y el asistente ha informado de ello.	**Correcto.**
129a	Reanudación de entrenamiento.	Al pulsar el botón play debería proseguir el entrenamiento.	El entrenamiento ha saltado la serie/descanso.	**Incorrecto.**
129b	Reanudación de entrenamiento.	Al pulsar el botón play debería proseguir el entrenamiento.	El entrenamiento continúa desde el punto pausado.	**Correcto.**

Tabla 49: Pruebas unitarias App: Iniciar entrenamiento 2/2

Id	Descripción	Resultado esperado	Resultado obtenido	Funcionamiento
130	Comprobación de audio: repeticiones.	Las series pares deberían indicarse con un aviso acústico.	El asistente indica correctamente la situación del entrenamiento.	**Correcto.**
131	Comprobación de audio: descanso.	El tiempo de descanso debería ir acompañado de un sonido de cronómetro.	El cronómetro se activa y desactiva si el usuario pausa el entrenamiento.	**Correcto.**
132a	Comprobar estabilidad de la aplicación.	Se ha salido del entrenamiento en tiempo de ejecución.	Se ha producido un fallo y se ha cerrado la aplicación.	**Incorrecto.**
132b	Comprobar estabilidad de la aplicación.	Se ha salido del entrenamiento en tiempo de ejecución.	La aplicación se detiene y evita el error, pudiendo continuar posteriormente.	**Correcto.**
133a	Comprobar disponibilidad del Text To Speech.	El Text To Speech debería volver a estar disponible tras mover la App a segundo plano y recuperarla.	El Text To Speech ha dejado de funcionar.	**Incorrecto.**
133b	Comprobar disponibilidad del Text To Speech.	El Text To Speech debería volver a estar disponible tras mover la App a segundo plano y recuperarla.	El Text To Speech vuelve a funcionar cuando se continúa el entrenamiento tras la pausa.	**Correcto.**
134	Mantener pantalla encendida durante entrenamiento.	La pantalla debería mantenerse encendida.	La pantalla se mantiene activa durante el entrenamiento.	**Correcto.**

- **Test 126:** El método que calcula el incremento de serie estaba mal implementado, haciendo que aumentase el número de la serie actual pero disminuyese el número de series totales, pasando de 1/3 -> 2/2 -> 3/1. Además tampoco ejecutaba el método que da paso al siguiente ejercicio. Tras realizar algunas modificaciones en el algoritmo ya se pudo mostrar correctamente la información.
- **Test 129:** El algoritmo estaba mal definido y hacían falta unos ajustes para determinar en qué fase estaba (en descanso, en marcha, etc.) y actuar en consecuencia. Tras indicar el estado del entrenamiento utilizando variables booleanas el funcionamiento fue el deseado.

- **Test 132:** Al poner la activity en segundo plano la aplicación se cerraba al producirse un error. Dicho error es producido porque la aplicación continúa su ejecución aun no estando visible, por lo que al llegar los contadores al objetivo se intentaba cambiar de fragment, lanzando un error ya que no existe la actividad sobre la que insertarlo. Para solucionarlo, hay que sobreescribir el método *onPause()* para detener el entrenamiento.
- **Test 133:** Al poner la activity en segundo plano el text to speech dejaba de funcionar. Había que instanciar de nuevo la clase para poder seguir utilizándola.

8. CONCLUSIONES Y TRABAJO FUTURO

En el último apartado se expondrán las diferentes conclusiones que se han obtenido durante la realización del proyecto dividida por capítulos.

8.1. Gestión general

En primer lugar mencionar que al comienzo del proyecto en diciembre de 2015 se tomó muy en serio la planificación, dedicándole el número apropiado de horas para poder realizarlo antes del plazo límite, el cual era julio 2016. Para ello se exageró ligeramente el cálculo inicial para forzarme a realizarlo antes de un plazo y tener la seguridad de que iba a acabarlo a tiempo. No obstante dichos planes fueron truncados debido a que me aceptaron en una empresa para comenzar prácticas y era una buena oportunidad, lo cual redujo la jornada que le podía dedicar al proyecto, retrasando enormemente los avances.

Si a ese motivo se le suma el duro trabajo desempeñado en la asignatura optativa "Desarrollo Avanzado de Software" en la cual me matriculé para adquirir posibles conocimientos que me ayudasen a desarrollar más limpiamente el proyecto, la planificación inicial queda totalmente desajustada, retrasándose varios meses por escasez de tiempo para poder dedicarle al TFG.

Y es que a pesar de llevar un control sumamente estricto en el cual diariamente he ido apuntando el número de horas y al área al que se las he dedicado se llegó tarde al plazo de entrega de julio. Aun pudiendo entregarlo para dicha convocatoria preferí retrasarlo dos meses para pulir la memoria, el aspecto gráfico de la aplicación y extender el contrato en prácticas, adquiriendo más experiencia de cara a mi futuro laboral.

1	Día	Tiempo	Tarea
21	13/01/2016	40	Analisis
22	14/01/2016		
23	15/01/2016		
24	16/01/2016	70	Analisis
25	17/01/2016	40	Analisis
26	18/01/2016	30	Formacion
27	19/01/2016	30	Analisis
28	20/01/2016		
29	21/01/2016	250	Documentacion
30	22/01/2016	160	Analisis
31	23/01/2016	120	Documentacion
32	24/01/2016	180	Documentacion
33	25/01/2016		
34	26/01/2016	30	Analisis
35	27/01/2016	120	Documentacion
36	28/01/2016	140	Documentacion
37	29/01/2016	120	Documentacion
38	30/01/2016	230	Documentacion

1	Día	Tiempo	Tarea
166	06/06/2016	150	Documentacion
167	07/06/2016	120	Documentacion
168	08/06/2016	160	Documentacion
169	09/06/2016	180	Documentacion
170	10/06/2016	60	Documentacion
171	11/06/2016	300	Documentacion
172	12/06/2016	120	Documentacion
173	13/06/2016	350	Documentacion
174	14/06/2016	240	Documentacion
175	15/06/2016	220	Documentacion
176	16/06/2016	180	Estilos
177	17/06/2016		
178	18/06/2016	300	Estilos
179	19/06/2016	40	Estilos
180	20/06/2016	240	Estilos/FIN
181	21/06/2016		
182	**Total**	**26025**	433,75
183	**Total (+DAS)**	**29025**	483,75

Ilustración 102: Ejemplo del cómputo diario de horas dedicadas

Nota: aunque en la captura parece que le haya dedicado todos los días tiempo al proyecto existen múltiples días o incluso dos semanas enteras que debido a atender el proyecto de DAS no pude dedicarle tiempo.

En cuanto al tiempo dedicado a la aplicación he de decir que tampoco me he alejado mucho de mis predicciones (~434 horas). El finalizar la memoria ha supuesto un duro trabajo que me ha llevado bastante más tiempo de lo previsto, casi triplicando los cálculos iniciales, de lo cual no me he dado cuenta hasta la hora de hacer cuentas al redactar este apartado.

Además habría que sumarle una fracción del tiempo que se ha dedicado a la asignatura DAS al haber servido para desarrollar el proyecto, dedicándole bastantes horas a la misma y siendo entonces cuando el tiempo estimado se aleja del real. Dado que una asignatura en el plan Bolonia tiene una relación de 1 crédito = 25 o 30 horas de trabajo, se van a asignar un total de 50horas de trabajo al proyecto, cantidad que considero razonable.

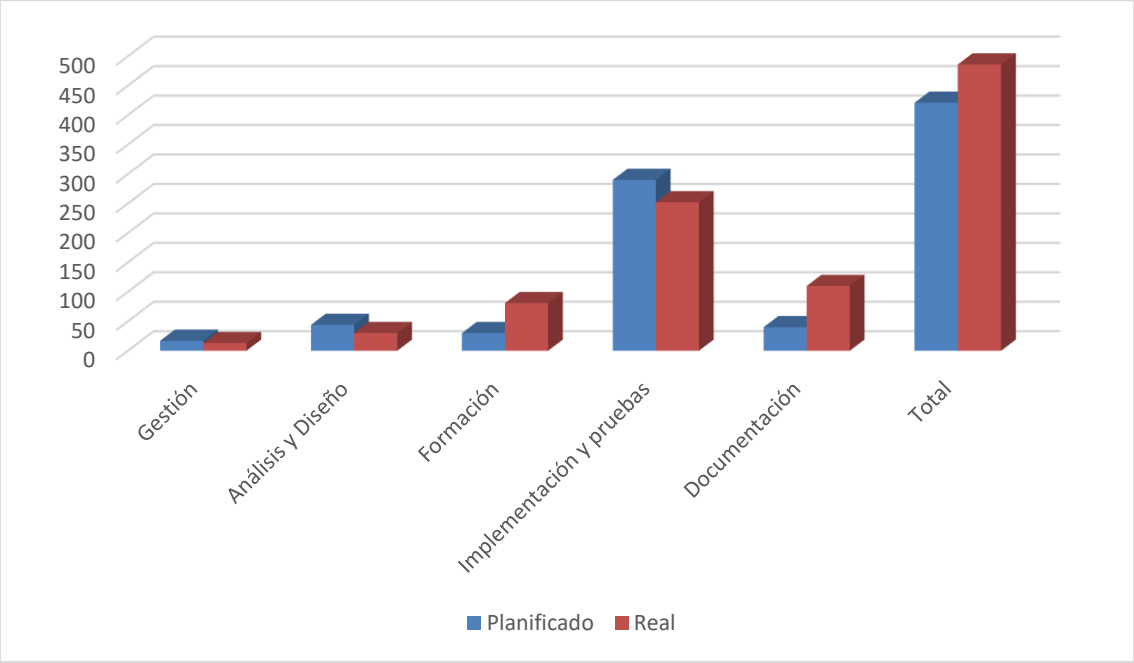


Ilustración 103: Gráfico comparativo: Tiempo planificado vs Tiempo real

El desarrollo de la herramienta Web inicialmente me causó cierto rechazo debido a los pocos conocimientos que se han ido adquiriendo por cuenta propia en el desarrollo de servicios PHP. No obstante a medida que se iba avanzando en la implementación esta resultó más sencilla de lo esperado, quedando satisfecho en cuanto a la exhaustividad con la que se comprueban los datos de cara a que no haya errores.

En cuanto al desarrollo de la aplicación a grandes rasgos me ha resultado más fácil de lo inicialmente previsto. Al haber dedicado un número considerable de horas a la formación (fruto de dedicarle tiempo personal y posteriormente tiempo de DAS) se redujo proporcionalmente el desarrollo de la misma.

Quizá de todo el desarrollo lo que más me ha sorprendido ha sido el tiempo que le he tenido que dedicar a la memoria para realizarla. Debido a experiencias previas en otras asignaturas, nunca he tenido problemas para redactar memorias con aspectos técnicos (EDA, IS) en relativamente poco tiempo y bastante exhaustivas, por ello el tiempo inicial previsto fue mucho menor del final. Finalmente mencionar que las imágenes utilizadas en esta memoria han sido localizadas con la herramienta de Google para buscar imágenes que permiten reutilización y modificación.

8.2. Objetivos

Una vez finalizado el desarrollo de la aplicación toca hacer cuentas y contrastar los resultados obtenidos con los objetivos iniciales.

El primero de ellos era aprender a programar aplicaciones para dispositivos Android con todo lo que ello suponía: aprender las mecánicas de manejo del IDE Android Studio, clases propias de Android, metodologías de estructuración de recursos en una aplicación (carpetas raw, assets, etc. en un lugar específico del paquete del proyecto) y muchas más cosas.

Hace poco comentaba al director del TFG que es sorprendente el miedo con el que afronté inicialmente el proyecto al ver la cantidad de elementos nuevos que tendría que utilizar y no sabía por dónde empezar, pero ahora es echar la mirada atrás y pensar qué cantidad de cosas he aprendido. Con lo cual, a nivel de enriquecimiento personal estoy muy contento y pienso que he cumplido este objetivo.

También se ha conseguido crear una infraestructura que de soporte a las peticiones que la aplicación Android por medio de unos servicios web robustos capaces de filtrar la información recibida y proporcionar a la vez la mejor respuesta posible.

La aplicación tiene una utilidad real y voy a poder utilizarla de aquí en adelante, que uno de los requisitos que me puse a mí mismo al realizar un proyecto. No quería algo que quedase olvidado en un cajón y no poder enseñar nunca. Por ese motivo y por lo que describo en el siguiente punto pienso que he cumplido todos mis objetivos.

8.3. Opinión personal

En este apartado describiré lo que este proyecto ha supuesto para mí. Al haber cursado una FPII de Electrónica pensé que desarrollar un proyecto en una ingeniería tendría las mismas mecánicas y metas, pero a medida que avanzaba en el desarrollo me preguntaba a mí mismo si eran iguales.

Tal vez el concepto que tengo en mente sobre qué es cursar un TFG está equivocado, pero tras finalizar la aplicación creo que la meta de los TFG no es haber desarrollado algo usable o tangible (dependiendo disciplina), sino la experiencia que se lleva uno del proceso en sí. No se trata de "mi aplicación es la mejor" o "a mí me ha quedado más bonita", se trata de "he trabajado duramente, he madurado al buscarme la vida para desarrollarlo y finalmente he podido con ello".

Todas esas horas dedicadas a pensar, diseñar, investigar, buscar la mejor manera posible han tenido su esfuerzo y, echando un vistazo atrás a esos distantes pero a la vez tan cortos 6 meses me doy cuenta de lo satisfecho que estoy.

Satisfecho porque pensé en ocasiones que no iba a ser capaz de desarrollar lo prometido y tener que buscar alternativas para conseguir llevarlo a cabo al tener constancia de aplicaciones mucho más complejas, ya que si una persona puede hacerlo, cualquiera puede.

Tal vez el punto que más me disgusta es el acabado gráfico. El diseño nunca ha sido mi punto fuerte y por ello, aun habiendo tenido presentes los criterios de diseño de Nielsen (ver Aspectos de usabilidad), pienso que es un punto a mejorar en el futuro.

En cuanto a la aplicación pienso que tiene su utilidad, ya que la idea partió de un día entrenando en casa, teniendo que activar constantemente el cronómetro cada vez que finalizaban las repeticiones y comenzaba el descanso. Entonces pensé, tal vez si a esto se le da alguna vuelta valga como proyecto. Estoy seguro que incluso algún amigo se animará a probar la aplicación, al igual que yo, que pienso darle uso ya que esa era mi objetivo. Además no he encontrado otra que te permita generar tus propios ejercicios, con lo cual considero que es algo único y que añadiéndole un par de elementos más podría tener un buen futuro.

8.4. Opiones de amigos

Al finalizar el desarrollo de la aplicación lo primero que hice fue enseñar la aplicación a los amigos y familiares para que me dijeran sus impresiones. En general les gustó la aplicación y no pusieron objeciones a las funcionalidades de la misma pero precisamente lo que buscaba eran críticas constructivas, y esas críticas se obtienen de gente que es deportista activa.

No fue sino hasta que llegó el mes de julio cuando, al reunirme con los amigos del pueblo pude obtener finalmente consejos de mejora de la mano de *Javier Luengo*.

Al igual que el resto de amigos comentó que la aplicación sonaba interesante, pero al practicar ciclismo de montaña asiduamente pudo darme un consejo que creo que realmente vale la pena, y es que sería mucho mejor aún si incluyese un apartado para realizar rutas en bicicleta o corriendo utilizando un GPS y poder compartirlas.

La envergadura de dicha mejora es tan grande que posiblemente pudiera ser otro TFG, pero es sin duda la crítica que más he valorado, ya que ha sido la única que ha aportado una posible línea de mejora en el futuro para *Time To Train*.

8.5. Trabajo futuro

Aun habiendo cumplido todos los requisitos básicos de desarrollo del proyecto expondré aquí unas sugerencias que pienso que bordarían el trabajo, siendo la inclusión de alguno de ellos un verdadero desafío.

En primer lugar se trataría de diseñar una segunda línea de trabajo en la cual el sistema trabajaría en local. *Time To Train* hoy por hoy necesita de conexión de datos para poder funcionar, ya que toda la información se solicita a un servidor. En caso de ir a un lugar apartado por poca conectividad la aplicación no funcionaría.

La idea sería generar un sistema que guardase las plantillas en local, de esta manera se necesitaría internet una única vez para generar las plantillas o realizar búsquedas y agregarlas a favoritos, actualizando tablas de memoria internet con los nuevos cambios. Cuando el usuario vaya a realizar la petición al servidor, si se detecta un problema de conexión ofrecería lo guardado en memoria en vez de hacer la petición al servidor.

Al investigar otras aplicaciones similares para redactar los ANTECEDENTES de la memoria pude comprobar que había aplicaciones que ofrecían dietas, componente esencial para el desarrollo muscular. Por ello el implementar un content provider como puede ser un calendario añadiendo la comida que se tiene que realizar cada día de la semana sería un punto que, en mi opinión, situaría por delante a *Time To Train* frente al resto de aplicaciones.

Por último se pensó en un comienzo en la inclusión de giroscopio/acelerómetro para marcar el ritmo de los ejercicios en vez de que estos fuesen avanzando por tiempo, pero tras darle vueltas no pudo ser incluido ya que el número de ejercicios que pueden aprovecharlo es muy limitado. Pienso que es una idea que podría ser implementada en el futuro pero dada la complejidad de la misma unido a los ejercicios que puedan llegar a utilizarlo hace que sea una propuesta atractiva, pero en ningún caso algo cuyo desarrollo se pueda asegurar.

9. BIBLIOGRAFÍA

(1) Xataka.com
Android e iOS ya están en el 96,4% de los smartphones.
http://www.xataka.com/moviles/idc-android-e-ios-ya-estan-en-el-96-4-de-los-smartphones (accedido el 28/12/2015)

(2) Theneddo
Autor de la Ilustración "logo de Cacoo"
"Cacoo logo" by Theneddo - Own work. Licensed under CC BY-SA 3.0 via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:Cacoo_logo.png#/media/File:Cacoo_logo.png (accedido el 1/1/2016)

(3) AxG
Autor de la Ilustración "Logo de Office 2013"
"Microsoft Office 2013 logo and wordmark" by Original work: Microsoft CorporationThis SVG version: AxG at English Wikipedia - Original: http://www.microsoft.com/office/preview/This SVG version: Own work. Licensed under Public Domain via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:Microsoft_Office_2013_logo_and_wordmark.svg#/media/File:Microsoft_Office_2013_logo_and_wordmark.svg (accedido el 1/1/2016)

(4) GanttProject Developers
Autor de la Ilustración "Logo de GanttProject"
"Ganttprojectlogo" by GanttProject Developers - GanttProject Developers. Licensed under CC BY-SA 3.0 via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:Ganttprojectlogo.png#/media/File:Ganttprojectlogo.png (accedido el 1/1/2016)

(5) Google Inc.
Autor de la Ilustración "Logo de Android Studio"
"Android Studio icon" by Google Inc. - Android.com. Licensed under CC BY 2.5 via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:Android_Studio_icon.svg#/media/File:Android_Studio_icon.svg (accedido el 1/1/2016)

(6) Dropbox
Autor de la Ilustración "Logo de Dropbox"
"Dropbox logo (September 2013)" by Dropbox - https://www.dropbox.com/branding. Licensed under Public Domain via Wikimedia Commons -

https://commons.wikimedia.org/wiki/File:Dropbox_logo_(September_2013).svg#/media/File:Dropbox_logo_(September_2013).svg (accedido el 1/1/2016)

(7) Mascota oficial del proyecto GIMP
Autor de la Ilustración “Logo de GIMP”
"Wilber-gimp" by Mascota oficial del proyecto GIMP - Wilber Construction Kit. Licensed under CC BY-SA 3.0 via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:Wilber-gimp.png#/media/File:Wilber-gimp.png (accedido el 1/1/2016)

(8) Sublime Text 2
Autor de la Ilustración “Logo de Sublime Text 2”
"Sublime Text Logo" by Source (WP:NFCC#4). Licensed under Fair use via Wikipedia - https://en.wikipedia.org/wiki/File:Sublime_Text_Logo.png#/media/File:Sublime_Text_Logo.png (accedido el 17/1/2016)

(9) OCU.org
¿Cuánta energía consume una casa?
http://www.ocu.org/vivienda-y-energia/gas-luz/noticias/cuanta-energia-consume-una-casa-571584 (accedido el 2/1/2016)

(10) Jarroba.com
ListView o Listado en Android
http://jarroba.com/listview-o-listado-en-android/ (accedido el 30/3/2016)

(11) Sshhtt.com
Sound Effects (sonidos gratuitos)
http://www.sshhtt.com/sounds/home (accedido el 30/3/2016)

(12) Flaticon.com
Autor del icono “Aceptar” de la aplicación.
Icons made http://www.flaticon.com/authors/hadrien from http://www.flaticon.com is licensed by http://creativecommons.org/licenses/by/3.0/ Creative Commons BY 3.0 (accedido el 2/4/2016)

(13)Flaticon.com
Autor del icono “Ejercicio” de la aplicación.
Icons made by http://www.freepik.com from http://www.flaticon.com is licensed by http://creativecommons.org/licenses/by/3.0/ Creative Commons BY 3.0 (accedido el 2/5/2016)

(14)Flaticon.com
Autor del icono “Refrescar” de la aplicación.

Icons made by http://www.flaticon.com/authors/gregor-cresnar from http://www.flaticon.com is licensed by http://creativecommons.org/licenses/by/3.0/ Creative Commons BY 3.0 (accedido el 6/4/2016)

(15)Flaticon.com
Autor del icono "Ordenar" de la aplicación.
Icons made by http://www.flaticon.com/authors/dave-gandy from http://www.flaticon.com is licensed by http://creativecommons.org/licenses/by/3.0/ Creative Commons BY 3.0 (accedido el 6/4/2016)

(16)Flaticon.com
Autor del icono "Cancelar" de la aplicación.
Icons made by http://www.flaticon.com/authors/lyolya from http://www.flaticon.com is licensed by http://creativecommons.org/licenses/by/3.0/ Creative Commons BY 3.0 (accedido el 12/4/2016)

(17)Flaticon.com
Autor del icono "Lista" de la aplicación.
Icons made by http://www.freepik.com from http://www.flaticon.com is licensed by http://creativecommons.org/licenses/by/3.0/ Creative Commons BY 3.0 (accedido el 12/4/2016).

(18)Flaticon.com
Autor del icono "Ejercicios" de la aplicación.
Icons made by http://www.flaticon.com/authors/gregor-cresnar from http://www.flaticon.com is licensed by http://creativecommons.org/licenses/by/3.0/ Creative Commons BY 3.0 (accedido el 12/4/2016)

(19)Flaticon.com
Autor del icono "Buscar" de la aplicación.
Icons made by http://www.flaticon.com/authors/gregor-cresnar from http://www.flaticon.com is licensed by http://creativecommons.org/licenses/by/3.0/ Creative Commons BY 3.0 (accedido el 1/5/2016)

(20)Flaticon.com
Autor del icono "Estrella" de la aplicación.
Icons made by http://www.flaticon.com/authors/gregor-cresnar from http://www.flaticon.com is licensed by http://creativecommons.org/licenses/by/3.0/ Creative Commons BY 3.0 (accedido el 7/5/2016)

(21)Flaticon.com
Autor del icono "Logout" de la aplicación.
Icons made by http://www.freepik.com from http://www.flaticon.com is licensed by http://creativecommons.org/licenses/by/3.0/ Creative Commons BY 3.0 (accedido el 17/5/2016)

(22)Flaticon.com
Autor del icono “Play” de la aplicación.
Icons made by http://www.freepik.com from http://www.flaticon.com is licensed by http://creativecommons.org/licenses/by/3.0/ Creative Commons BY 3.0 (accedido el 25/5/2016)

(23)Flaticon.com
Autor del icono “Usuario/contraseña en login” de la aplicación.
Icons made by http://www.flaticon.com/authors/designerz-base from http://www.flaticon.com is licensed by http://creativecommons.org/licenses/by/3.0/ Creative Commons BY 3.0 (accedido el 18/6/2016)

(24)Flaticon.com
Autor del icono “Password en login” de la aplicación.
Icons made by http://www.freepik.com from http://www.flaticon.com is licensed by http://creativecommons.org/licenses/by/3.0/ Creative Commons BY 3.0 (accedido el 18/6/2016)

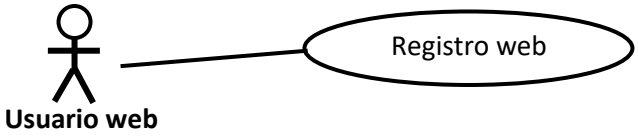
(25)Flaticon.com
Autor del icono “Repetición en elemento_listado_ejercicio_plantilla” de la aplicación.
Icons made by http://www.freepik.com from http://www.flaticon.com is licensed by http://creativecommons.org/licenses/by/3.0/ Creative Commons BY 3.0 (accedido el 20/6/2016)

(26)Flaticon.com
Autor del icono “Descanso en elemento_listado_ejercicio_plantilla” de la aplicación.
Icons made by http://www.freepik.com from http://www.flaticon.com is licensed by http://creativecommons.org/licenses/by/3.0/ Creative Commons BY 3.0 (accedido el 20/6/2016)

(27)Tiago de Jesus Neves
Autor de la Ilustración “Arquitectura Cliente Servidor 1/2”
https://commons.wikimedia.org/wiki/File:Cliente-Servidor.png (accedido el 05/01/2016)

(28)YurikaH
Autor de la Ilustración “Arquitectura Cliente Servidor 2/2”
https://commons.wikimedia.org/wiki/File:Modelo_Cliente-Servidor.png (accedido el 05/01/2016)

ANEXO I:
CASOS DE USO EXTENDIDOS



Nombre: Registro web.

Descripción: Permite al usuario crear una cuenta para acceder al sistema mediante el proceso de inicio de sesión. Dicha cuenta podrá utilizarse para acceder tanto a la aplicación móvil como a la página web, aunque únicamente desde esta última se podrán realizar acciones como la creación de nuevos ejercicios o eliminación de cuenta.

Actores: Usuario web.

Precondiciones: Ninguna

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. El usuario accede a la web, pulsa el botón “Regístrame”.
2. Tras cumplimentar el formulario de registro, el usuario pulsará “Finalizar registro”.
   2.1. [Si al verificar los campos del formulario se encuentran errores]
   Se mostrará al usuario el informe de errores y las posibles causas.
   2.2. [Si al verificar los campos del formulario todo es correcto]
   Se le informará al usuario del éxito del registro y se le redireccionará a la página principal para que pueda comenzar a usar *Time To Train*.

Postcondiciones: Los datos del usuario quedarán almacenados en el sistema para que pueda iniciar sesión más adelante.

Interfaz gráfica:

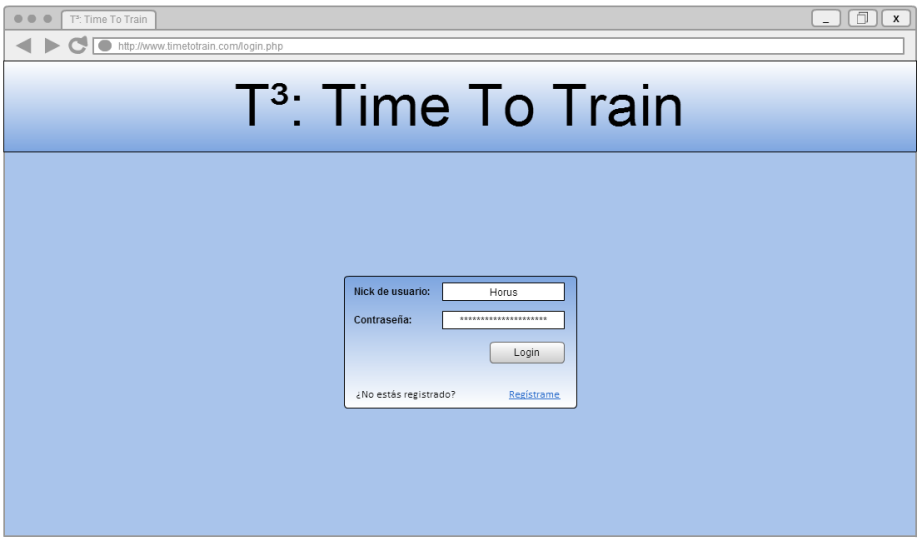


Ilustración 104: Web: Login

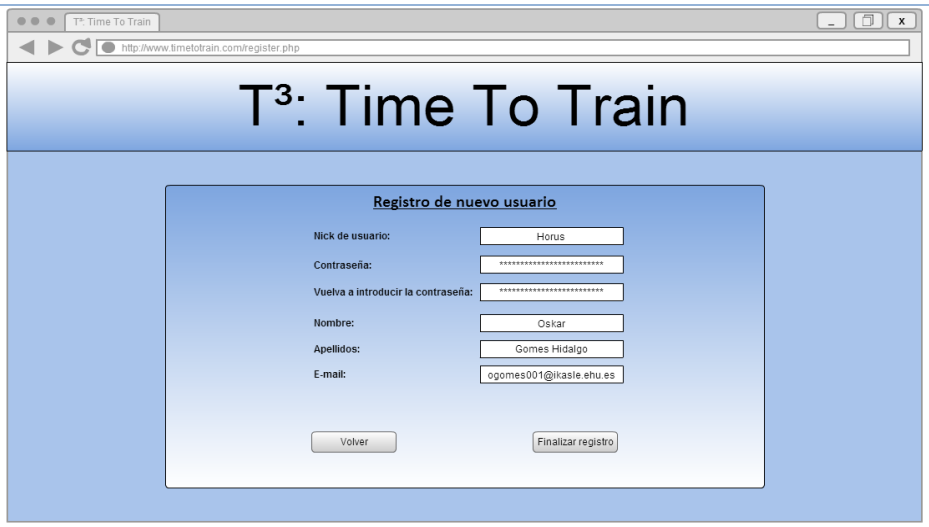


Ilustración 105: Web: Registro

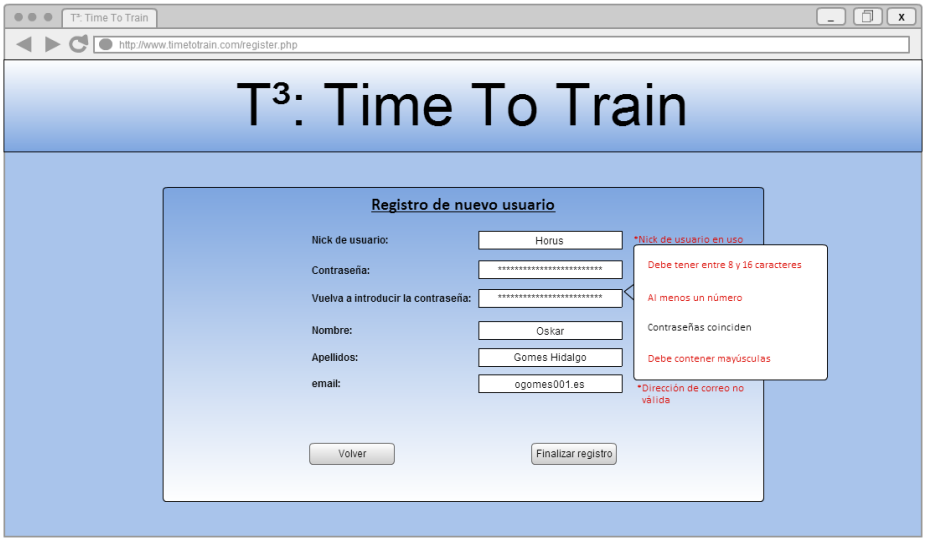


Ilustración 106: Web: Registro incorrecto

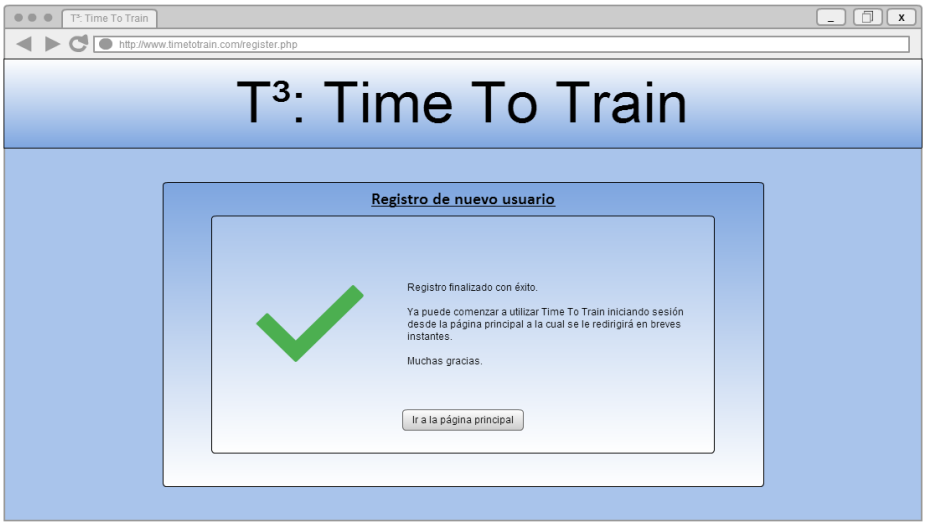
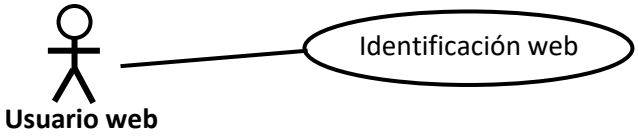


Ilustración 107: Web: Registro correcto



Nombre: Identificación web.

Descripción: Permite que cualquier usuario que se haya registrado previamente pueda acceder al sistema para realizar gestiones relacionadas con ejercicios y/o su cuenta.

Actores: Usuario web.

Precondiciones: Ninguna.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. El usuario introduce los datos de sus credenciales y pulsa "Login".
   1.1. [Si existe una correspondencia en la BD para ese usuario contraseña]
   El usuario accederá a la página principal de la aplicación, desde la que podrá:
   1.1.1. Gestionar sus ejercicios.
   1.1.2. Gestionar su cuenta.
   1.2. [Si no existe ninguna tupla en la BD para ese usuario y contraseña]
   Se mostrará un mensaje de error informando de la causa del mismo.

Postcondiciones: El usuario iniciará sesión en el sistema y accederá a la página principal.

Interfaz gráfica:

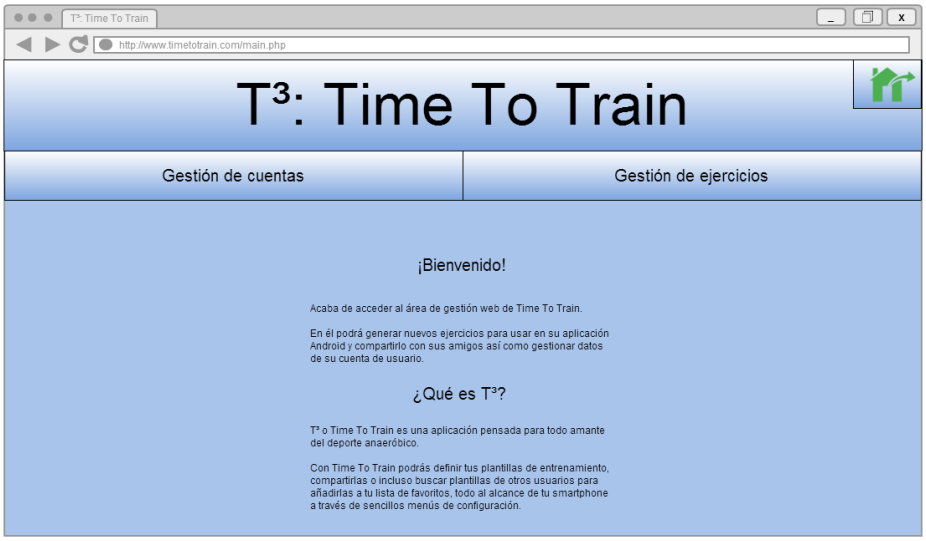


Ilustración 108: Web: Loged principal

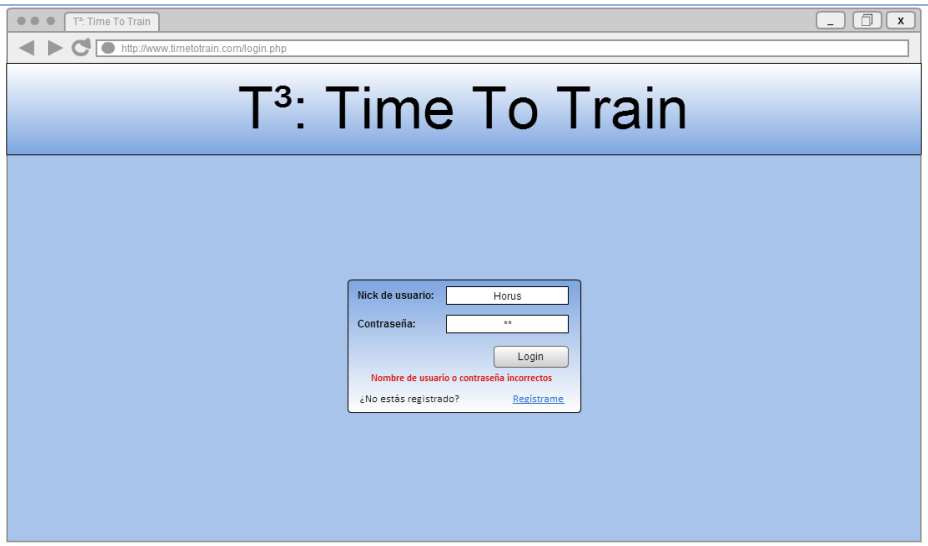
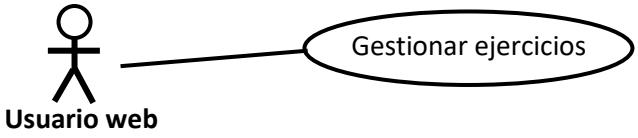


Ilustración 109: Web: Login incorrecto



Nombre: Gestionar ejercicios.

Descripción: Permite que el usuario acceda al área desde la cual puede gestionar ejercicios, ya sea creando nuevos, modificándolos o borrándolos del sistema.

Actores: Usuario web.

Precondiciones: Que el usuario haya iniciado sesión en el sistema.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. El usuario selecciona la opción “Gestión de Ejercicios”.
2. El sistema accede al área de gestión de ejercicios, donde el usuario podrá:
   2.1. Crear nuevos ejercicios.
   2.2. Modificar ejercicios creados anteriormente por él.
   2.3. Eliminar ejercicios creados anteriormente por él.

Postcondiciones: El usuario será dirigido al área de gestión de ejercicios.

Interfaz gráfica:

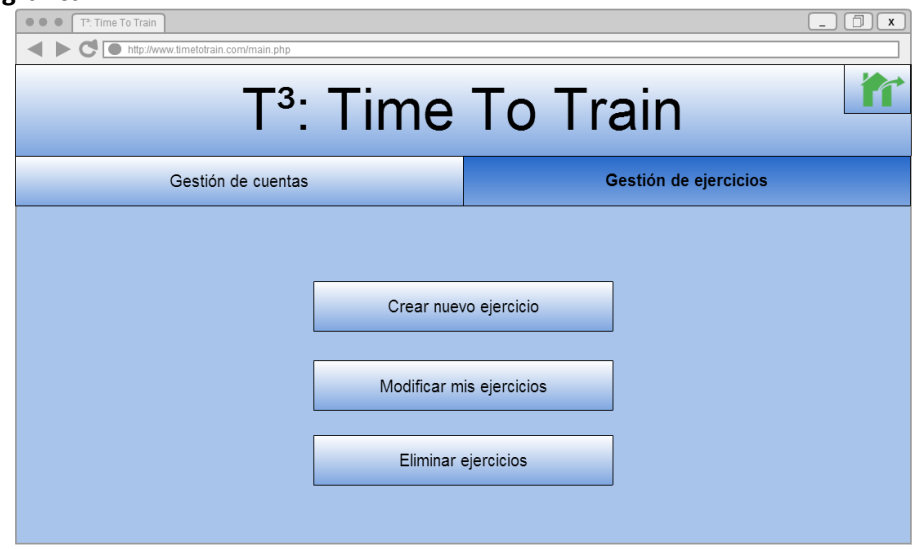
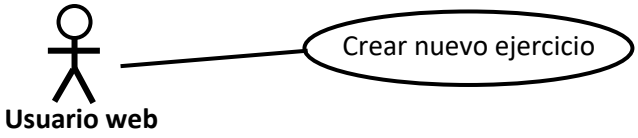


Ilustración 110: Web: Gestión de ejercicios



Nombre: Crear nuevo ejercicio.

Descripción: Da de alta un nuevo tipo de ejercicio en el sistema indicando sus datos, gif del movimiento y una descripción para que los usuarios de la aplicación móvil puedan incluirlo en sus plantillas.

Actores: Usuario web.

Precondiciones: Que el usuario haya accedido al área de gestión de ejercicios.

Requisitos no funcionales: El usuario rellenará los datos y el JavaScript de la página validará el formulario de cara a que no haya errores en los campos (especialmente con la URL de la imagen, la cual se pedirá que sea de www.imgur.com).

Flujo de eventos:

1. El usuario rellena los datos del formulario indicando nombre, descripción, seleccionando un grupo muscular del ejercicio a crear en el dropdown e indicando la url de un gif en imgur.com.
2. A continuación pulsará sobre “Crear ejercicio”.
   2.1. [Si ya existe un ejercicio con el mismo nombre]
   Se mostrará un mensaje avisando de la incidencia y obligando al usuario a que indique otro nombre.
   2.2. [Si la url es incorrecta]
   Se mostrará un mensaje avisando de la incidencia y deberá especificar una url válida.
   2.3. [Si no se ha seleccionado ningún grupo muscular]
   Se mostrará un mensaje de aviso informando de que ha de especificarlo.
   2.4. [Si la validación del formulario resulta correcta]
   Un mensaje de confirmación avisará de que el ejercicio ha sido almacenado correctamente en el sistema y se le redirigirá a “gestión de ejercicios”.

Postcondiciones: El nuevo ejercicio quedará almacenado en el sistema para que otros usuarios podrán utilizarlo en sus plantillas.

Interfaz gráfica:

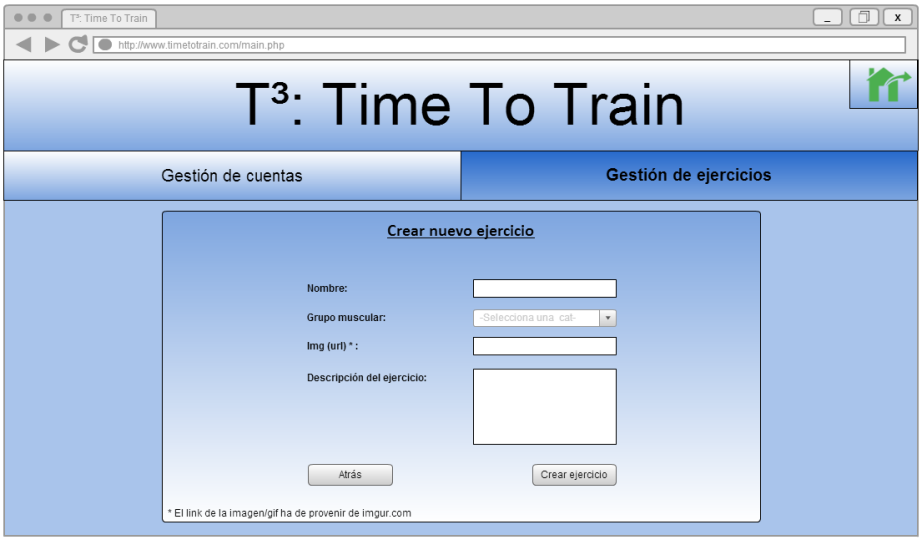


Ilustración 111: Web: Crear nuevo ejercicio

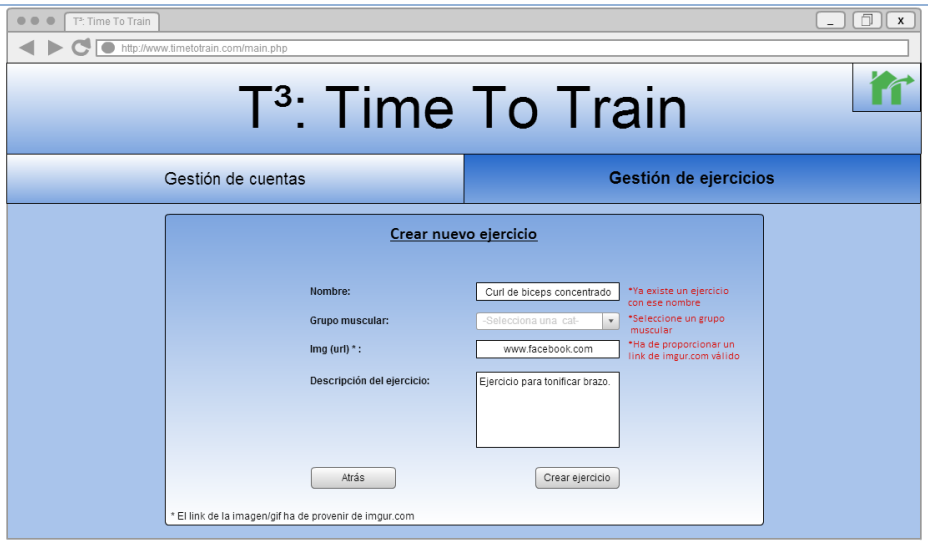


Ilustración 112: Web: Crear nuevo ejercicio error

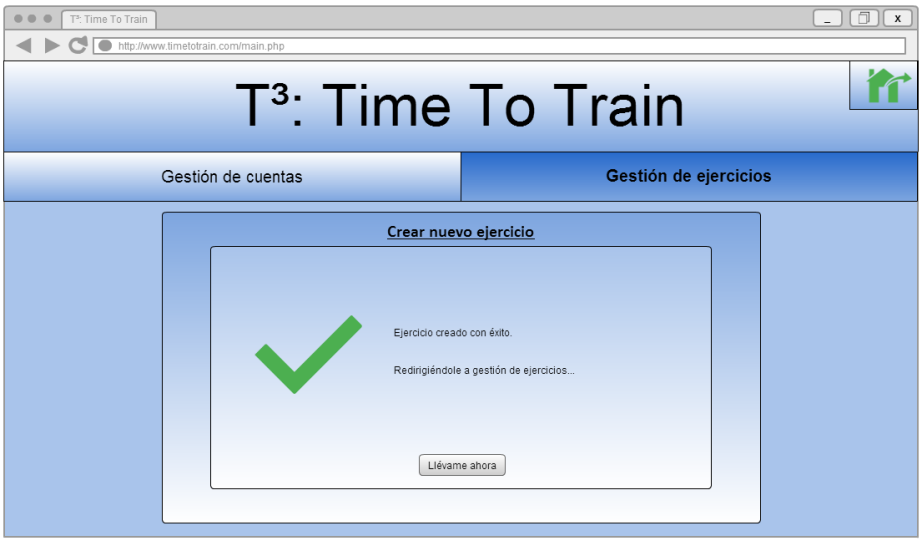
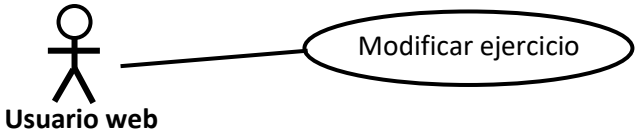


Ilustración 113: Web: Crear nuevo ejercicio éxito



Nombre: Modificar ejercicio.

Descripción: Permite que el usuario visualice los ejercicios que él mismo haya generado y así realizar modificaciones sobre ellos.

Actores: Usuario web.

Precondiciones: Que el usuario haya accedido al área de gestión de ejercicios.

Requisitos no funcionales: Al seleccionar un elemento de la lista de ejercicios del usuario, los datos referentes a ese ejercicio se verán reflejados en la tabla central, donde se podrán modificar.

Flujo de eventos:

1. Se le muestra al usuario un listado de ejercicios creados por él en el marco izquierdo.
2. El usuario pincha sobre uno de los ejercicios.
3. En el marco derecho se cargan los datos que contiene el ejercicio elegido.
   3.1. El usuario realiza modificaciones en los campos
      3.1.1.[Si pulsa “Atrás”]
         Se descartan los cambios realizados y redirige a la gestión de ejercicios.
      3.1.2.[Si pulsa “Guardar cambios”]
         3.1.2.1. [Si la url ha sido modificada y contiene fallos]
            Se avisa al usuario con un mensaje del error cometido.
         3.1.2.2. [Si los campos modificados son correctos]
            Se sustituyen los datos antiguos del ejercicio por los que existan en los campos en el momento de pulsar el botón.

Postcondiciones: Las modificaciones sobre el ejercicio quedarán reflejadas en la BD.

Interfaz gráfica:

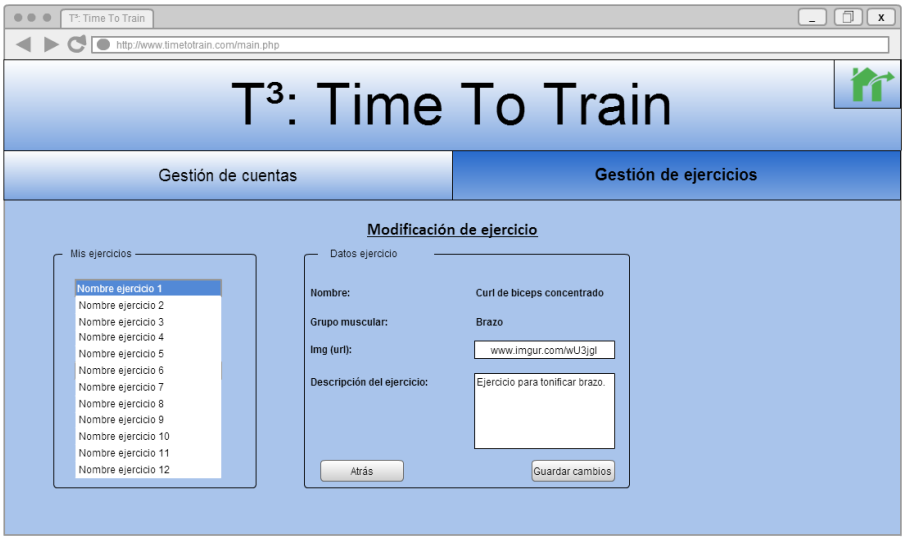


Ilustración 114: Web: Modificar ejercicio

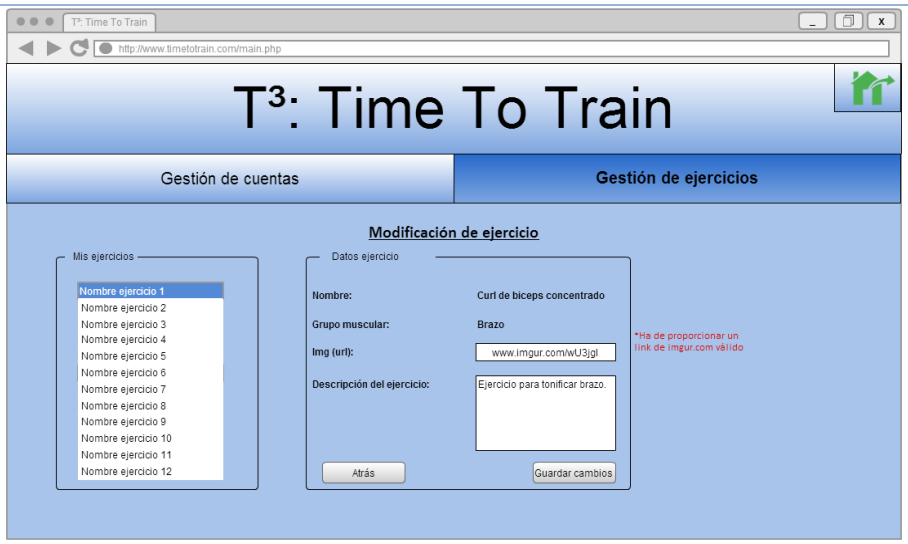


Ilustración 115: Web: Modificar ejercicio error

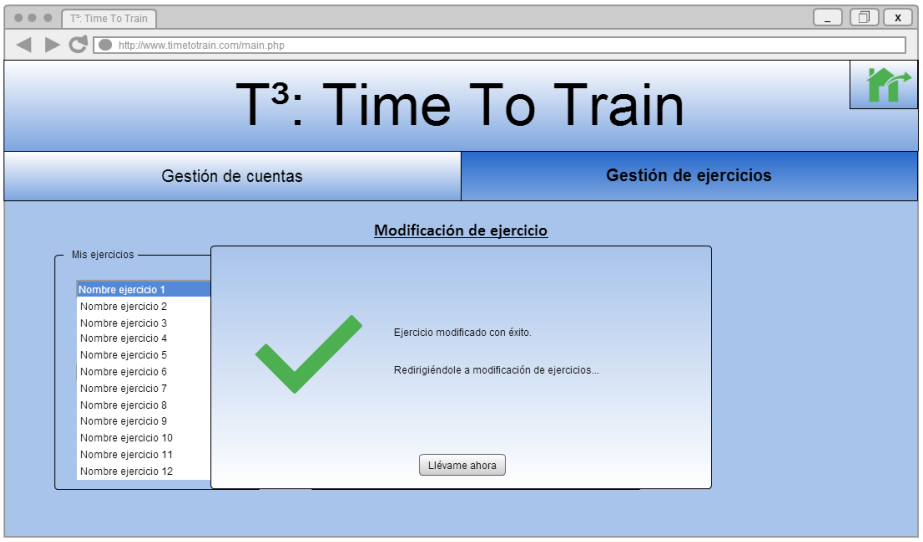
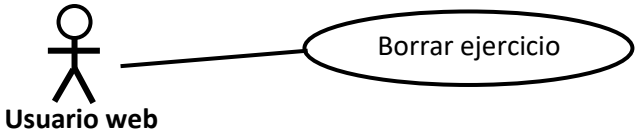


Ilustración 116: Web: Modificar ejercicio éxito



Nombre: Borrar ejercicio.

Descripción: Permite que el usuario pueda eliminar del sistema los ejercicios creados por él.

Actores: Usuario web.

Precondiciones: Que el usuario haya accedido al área de gestión de ejercicios.

Requisitos no funcionales: Tras seleccionar uno de los ejercicios creados por el usuario que aparecen en la lista de la izquierda el usuario verá sus datos para confirmar que es el ejercicio a eliminar. Tras ello pulsará Eliminar y tendrá que confirmar su eliminación. La eliminación se llevará a cabo sólo si ese ejercicio no está siendo utilizado por ningún otro usuario en sus plantillas.

Flujo de eventos:

1. Se le muestra al usuario un listado de ejercicios creados por él en el marco izquierdo.
2. El usuario pincha sobre uno de los ejercicios.
3. En el marco derecho se cargan los datos del ejercicio seleccionado.
4. El usuario pulsa “Eliminar ejercicio”.
   Un mensaje pedirá la confirmación del usuario para verificar que no ha sido una pulsación involuntaria.
   4.1. El usuario pulsa “Eliminar”.
      4.1.1.[Si ese ejercicio está en uso por algún usuario]
         El ejercicio no será borrado y el usuario será avisado.
         4.1.1.1. El usuario pulsará “Aceptar” para volver al área de eliminación de ejercicios.
      4.1.2.[Si el ejercicio no está en uso]
         El ejercicio se borrará del sistema, avisando al usuario del éxito.
         4.1.2.1. El usuario pulsará “Aceptar” para volver al área de eliminación de ejercicios.

Postcondiciones: El ejercicio se eliminará de la base de datos.

Interfaz gráfica:

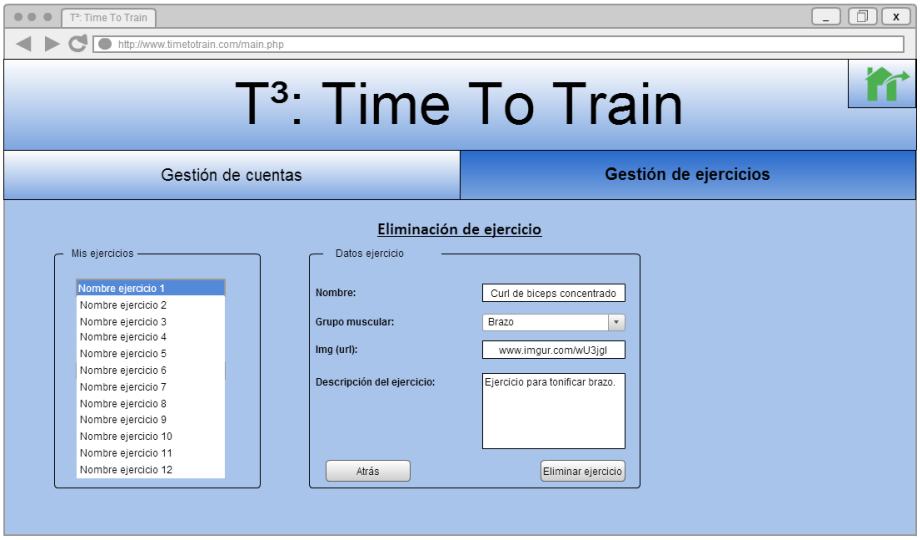


Ilustración 117: Web: Eliminar ejercicio

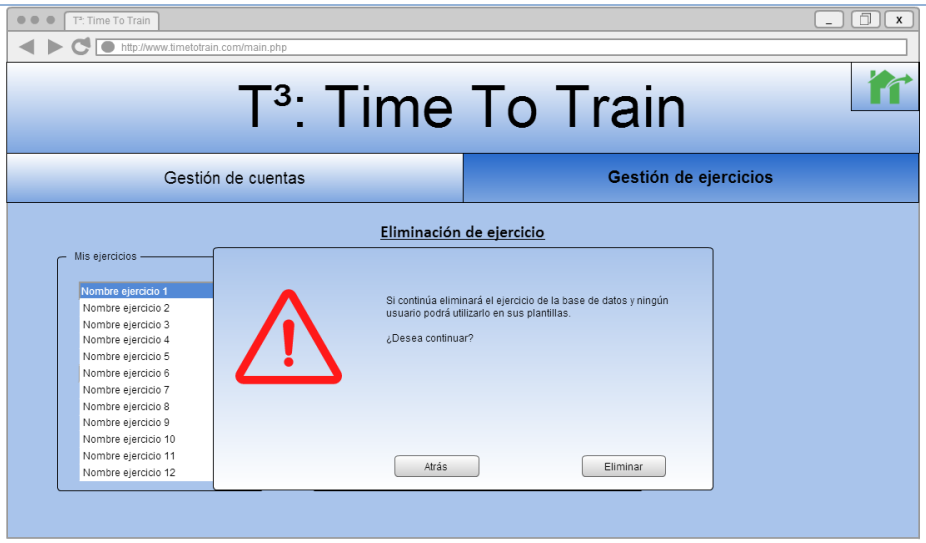


Ilustración 118: Web: Eliminar ejercicio aviso

Ilustración 119: Web: Eliminar ejercicio en uso

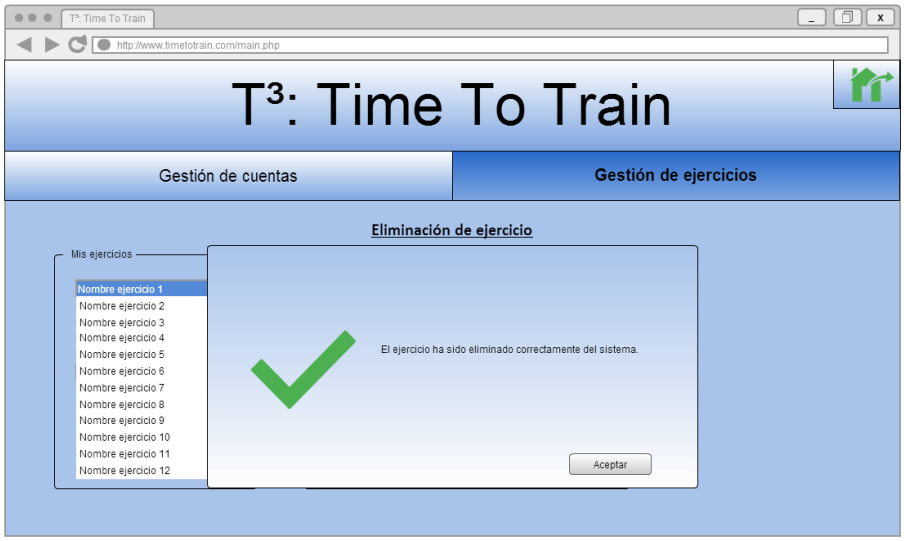
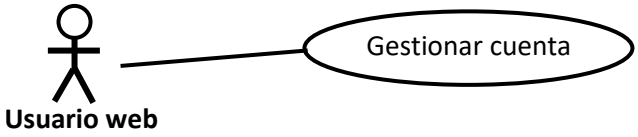


Ilustración 120: Web: Eliminar ejercicio éxito



Nombre: Gestionar cuenta.

Descripción: Permite al usuario acceder al área de gestión de cuenta desde la cual podrá decidir si realizar modificaciones en la misma o eliminarla.

Actores: Usuario web.

Precondiciones: Que el usuario haya accedido al área de gestión de cuentas.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. El usuario selecciona la opción “Gestión de cuentas”.
2. El sistema accede al área de gestión de cuentas, donde el usuario podrá:
   2.1. Modificar los datos de su cuenta.
   2.2. Eliminar su cuenta de usuario.

Postcondiciones: El usuario será dirigido al área de gestión de cuentas.

Interfaz gráfica:

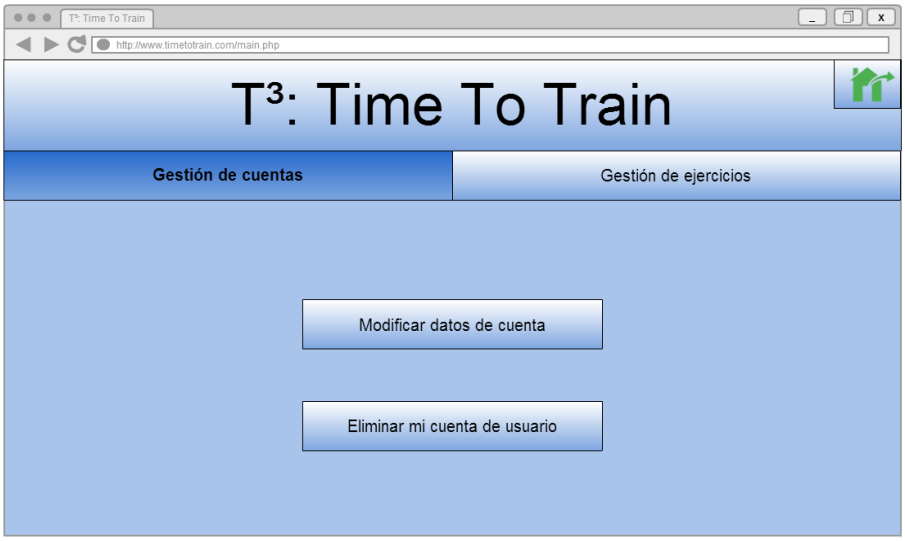
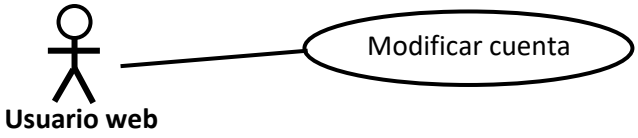


Ilustración 121: Web: Gestión de cuentas



Nombre: Modificar cuenta.

Descripción: El usuario podrá realizar gestiones en su cuenta de usuario al permitirle modificar los datos de la misma.

Actores: Usuario web.

Precondiciones: Que el usuario haya accedido al área de gestión de cuentas.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. Se le muestra al usuario un formulario el cual podrá rellenar con los datos a modificar.
2. El usuario modifica aquellos campos que quiera modificar.
3. A continuación pulsará "Guardar cambios".
4. Un mensaje pide que el usuario introduzca la antigua contraseña para permitir que se realicen cambios en su cuenta.
   4.1. [Si el usuario pulsa "Atrás"]
   Vuelve al área de modificación de cuentas.
   4.2. [Si la contraseña es incorrecta]
   Avisa del error y pide volver a introducirla.
   4.3. [Si la contraseña es válida]
   Se guardan los cambios en la base de datos y avisa de ello.

Postcondiciones: En caso de cumplirse los requisitos no funcionales, los datos del usuario se modificarán en la BD.

Interfaz gráfica:

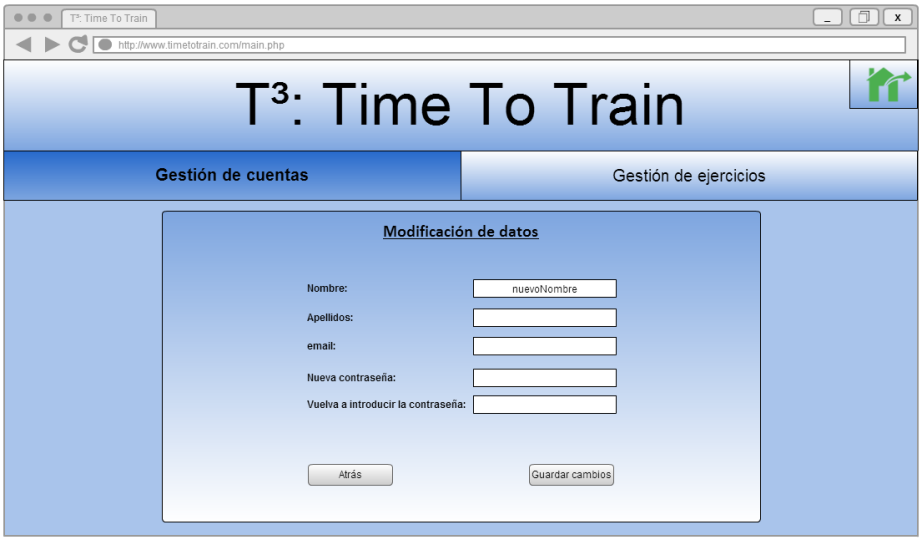


Ilustración 122: Web: Modificar cuenta

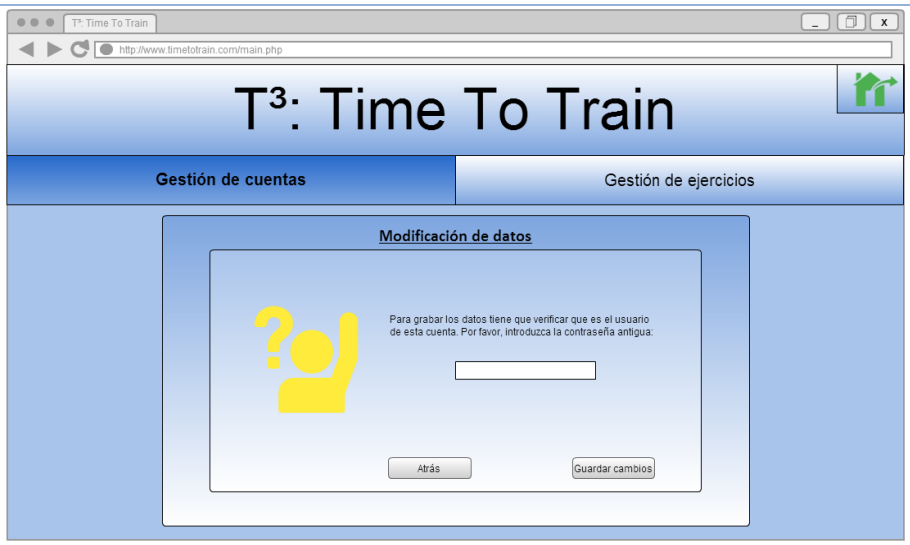


Ilustración 123: Web: Modificar cuenta verificación

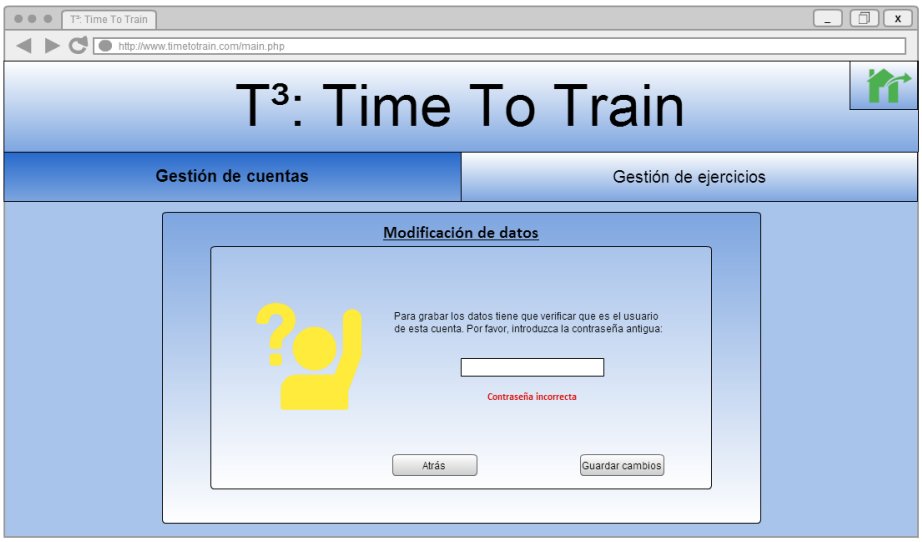


Ilustración 124: Web: Modificar cuenta verificación error

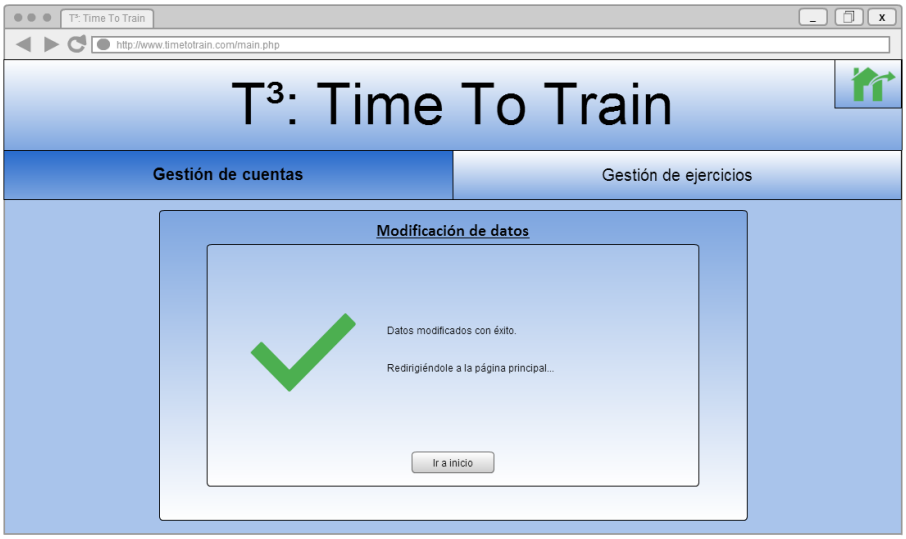
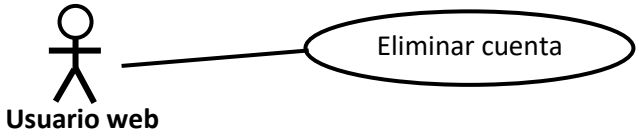


Ilustración 125: Web: Modificar cuenta éxito



Nombre: Eliminar cuenta.

Descripción: Permite que el usuario elimine sus datos del sistema al dar de baja su cuenta.

Actores: Usuario web.

Precondiciones: Que el usuario haya accedido al área de gestión de cuentas.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. Se carga la página en la que un mensaje informará al usuario de que tiene que introducir su contraseña para realizar la eliminación.
2. El usuario introduce la contraseña.
3. A continuación pulsará "Eliminar"
   3.1. [Si la contraseña es incorrecta]
   3.2. Un mensaje avisará al usuario del error cometido.
   3.3. [Si la contraseña es correcta]
   3.4. Se redirigirá a la página principal y la cuenta se eliminará.

Postcondiciones: La cuenta del usuario quedará eliminada del sistema y no podrá seguir iniciando sesión.

Interfaz gráfica:

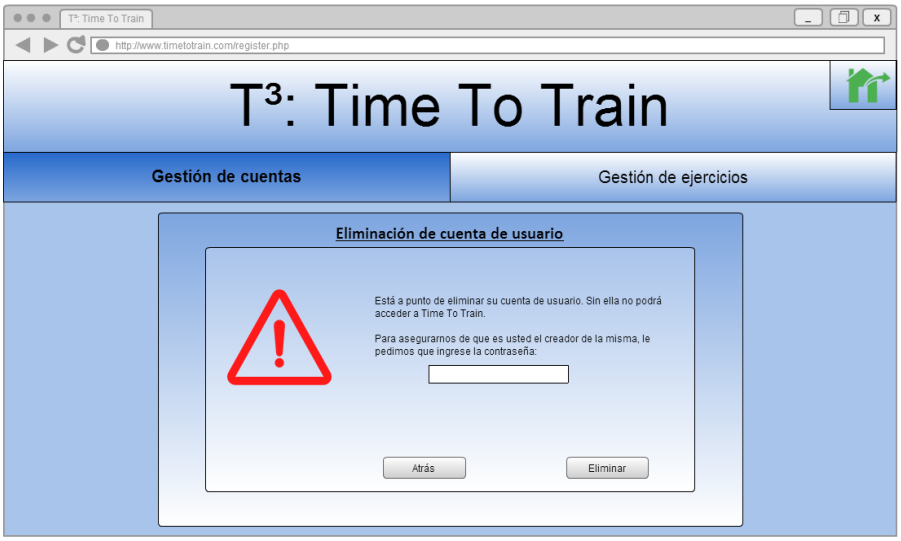


Ilustración 126: Web: Eliminar cuenta verificación

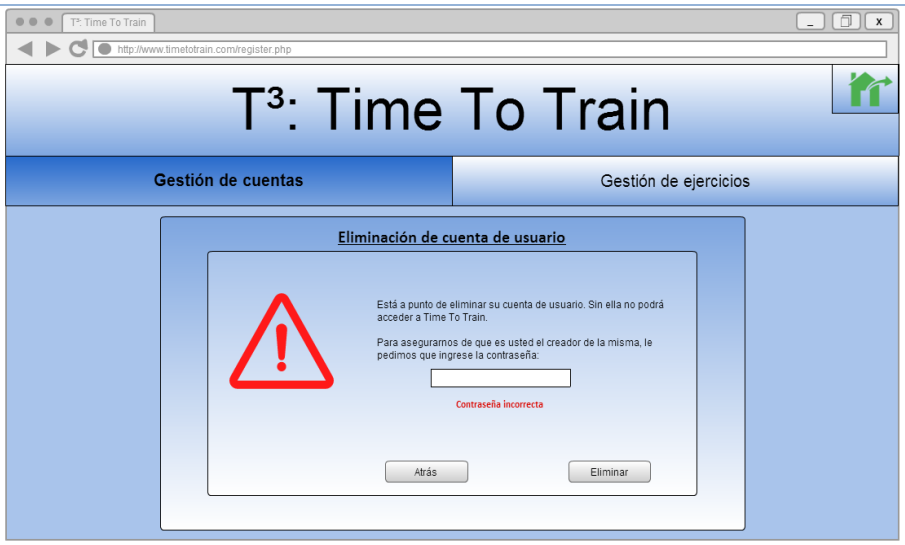
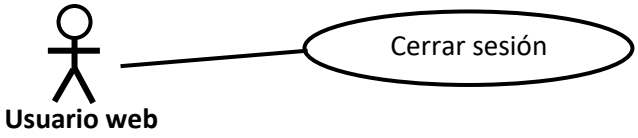
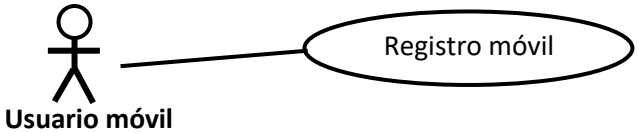


Ilustración 127: Web: Eliminar cuenta verificación error



Nombre: Cerrar sesión.
Descripción: Cierra la sesión activa en la web y deja el sistema preparado para iniciar sesión con otra cuenta.
Actores: Usuario web.
Precondiciones: Que el usuario haya iniciado sesión en el sistema.
Requisitos no funcionales: Ninguno.
Flujo de eventos: 1. El usuario pulsa en el icono situado en la esquina superior derecha. 2. El sistema olvidará las credenciales del usuario y se situará en la página de inicio de sesión.
Postcondiciones: El sistema redirigirá a la página principal y estará listo para entrar con otra cuenta.
Interfaz gráfica:

Ilustración 128: Web: Logout



Nombre: Registro móvil.

Descripción: Permite que los usuarios del dispositivo móvil creen una cuenta con la que posteriormente puedan acceder a la aplicación y utilizarla.

Actores: Usuario móvil.

Precondiciones: Conexión de datos en el móvil activa.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. Se debe acceder a la aplicación y pulsar sobre el botón de registro.
2. El usuario rellena el formulario y pulsa el botón para validar y registrarse.
   2.1. [Si al verificar los campos del formulario se detectan errores]
   Se mostrará un mensaje de error con el motivo del mismo.
   2.2. [Si al verificar los campos del formulario todo es correcto]
   Se finaliza el registro y se guarda su información en el sistema.

Postcondiciones: La cuenta de usuario quedará registrada en el sistema.

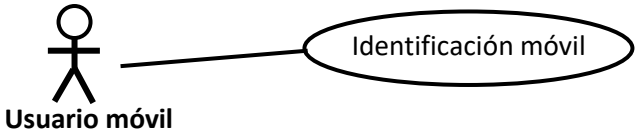
Interfaz gráfica:

Ilustración 129: App: Login

Ilustración 130: App: Registro

Ilustración 131: App: Registro incorrecto nombre

Ilustración 132: App: Registro incorrecto contraseñas



Nombre: Identificación móvil.

Descripción: Permite al usuario móvil iniciar sesión en su aplicación móvil siempre y cuando ya haya registrado una cuenta vía web o dispositivo móvil. Tras esto el usuario podrá utilizar la aplicación tanto para entrenar y visualizar datos online como para crear sus propios planes de entrenamiento y que otros usuarios los puedan visualizar.

Actores: Usuario móvil.

Precondiciones: Conexión de datos en el móvil activa.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. El usuario introduce los datos de inicio de sesión y pulsa el botón “Login”.
   1.1. [Si no existe ninguna cuenta que coincida con las credenciales]
   Se mostrará un mensaje de error informando de la causa del mismo.
   1.2. [Si existe alguna cuenta que concuerde con los datos de inicio de sesión]
   El usuario accederá a la vista principal de la aplicación, desde la que podrá:
   1.2.1. Acceder a sus plantillas y crear nuevas.
   1.2.2. Visualizar las plantillas marcadas como favoritas.
   1.2.3. Realizar búsquedas de plantillas o ejercicios.
   1.2.4. Realizar gestiones en su cuenta.

Postcondiciones: El usuario accederá al menú principal de la aplicación.

Interfaz gráfica:

Ilustración 133: App: Login error

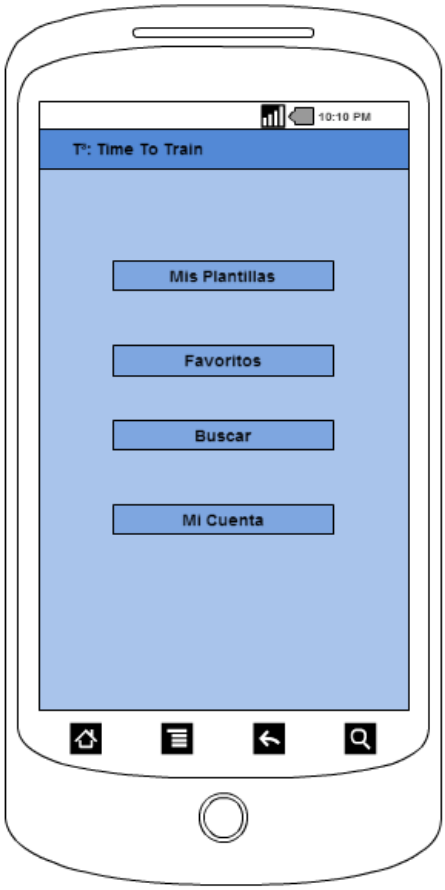
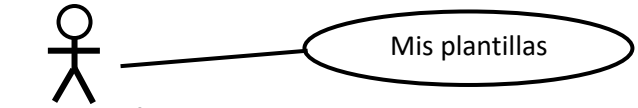


Ilustración 134: App: Loged principal



Nombre: Mis plantillas.

Descripción: Permite que el usuario que haya iniciado sesión acceda al menú de aquellos planes de entrenamiento que él mismo haya definido, sobre los cuales podrá realizar acciones como modificación o eliminación.

Actores: Usuario móvil.

Precondiciones: Haber iniciado sesión en la aplicación y mantener la conexión de internet.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. El usuario seleccionará la opción “Mis plantillas”.
2. El sistema accederá al área de plantillas del usuario desde donde podrá:
   2.1. Acceder a los detalles de las plantillas que él mismo ha generado.
   2.2. Crear nuevas plantillas de entrenamiento.

Postcondiciones: Se mostrará el listado de plantillas generadas por el usuario.

Interfaz gráfica:

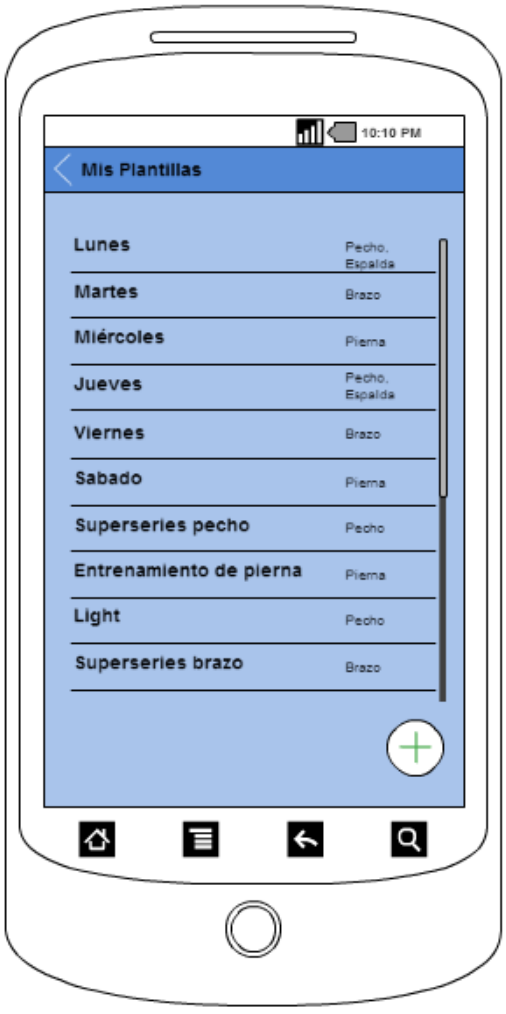
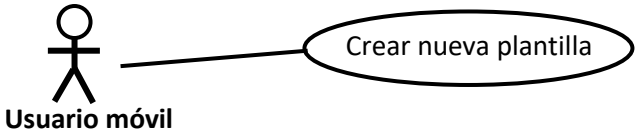


Ilustración 135: App: Mis Plantillas



Nombre: Crear nueva plantilla.

Descripción: Permite que el usuario que haya iniciado sesión comience la creación de una nueva plantilla de entrenamiento añadiéndole nuevos ejercicios a la misma. Para salvar los datos el usuario añadirá una serie de *tags*, una descripción y el nombre de la plantilla con la cual se guardará en el sistema.

Actores: Usuario móvil.

Precondiciones: Haber accedido al área "Mis plantillas" previamente.

Requisitos no funcionales: El número máximo de Tags es de cinco, uno por cada grupo muscular (brazo, hombro, pecho, abdomen y pierna). De esta manera se evitarán errores.

Flujo de eventos:

1. Se deberá pulsar sobre el botón "+" para crear una nueva plantilla.
2. El usuario añadirá los ejercicios que desee a la plantilla actual para dejar a su gusto, así como incluirá los tags que crea que definan la plantilla y tendrá que darle un nombre a la plantilla.
3. A continuación tendrá que validar la configuración pulsando el botón "v" de validación.
   3.1. [Si el nombre de la plantilla está en uso por ese usuario]
   Se avisará al usuario de tal circunstancia para que lo modifique.
   3.2. [Si el nombre de la plantilla es válido]
   Se almacenarán los datos en la BD y se le informará de ello.

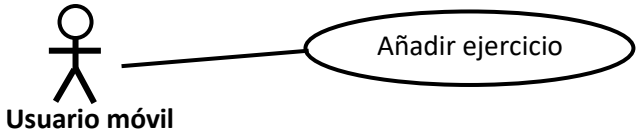
Postcondiciones: La nueva plantilla quedará almacenada en la BD asociada al usuario actual.

Interfaz gráfica:

Ilustración 136: App: Crear plantilla

Ilustración 137: App: Crear plantilla error

Ilustración 138: App: Crear plantilla éxito



Nombre: Añadir ejercicio.

Descripción: Permite que el usuario añada un nuevo ejercicio a la plantilla actual siempre y cuando el usuario esté identificado y la plantilla haya sido diseñada por él.

Actores: Usuario móvil.

Precondiciones: Acceder al área "Mis plantillas" y crear un nuevo entrenamiento.

Requisitos no funcionales: Los valores numéricos a introducir al generar la descripción del ejercicio estarán dentro de un rango establecido para evitar errores.

Flujo de eventos:

1. Durante la creación de una plantilla de entrenamiento el usuario pulsará el indicador "+" para añadir un nuevo ejercicio.
2. Se mostrará una lista que muestra la variedad de ejercicios que contiene actualmente el sistema.
3. Se deberá seleccionar el ejercicio que se quiera incluir en la plantilla actual y pulsará "v".
   3.1. [Si el usuario había seleccionado más de un ejercicio]
   Se mostrará un mensaje de error y pedirá que sólo seleccione uno.
   3.2. [Si sólo había seleccionado un ejercicio]
   El ejercicio se añadirá correctamente a la plantilla actual.

Postcondiciones: El nuevo ejercicio formará parte de la plantilla actual.

Interfaz gráfica:

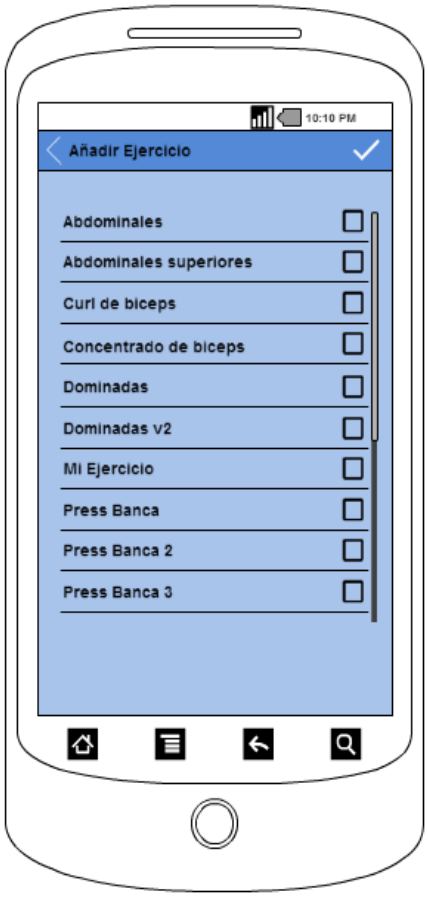


Ilustración 139: App: Añadir ejercicio

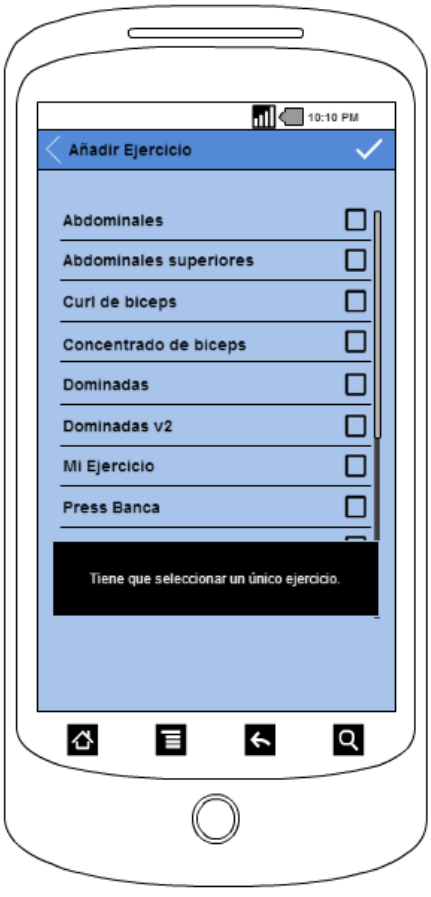
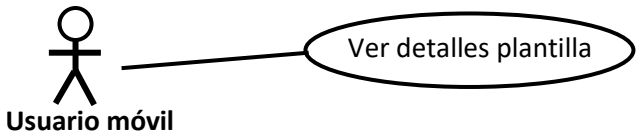


Ilustración 140: App: Añadir ejercicio error nombre



Nombre: Ver detalles plantilla.

Descripción: Permite que el usuario, una vez haya iniciado sesión en la aplicación, pueda ver detalles de la plantilla, lo cual incluye el listado de ejercicios que la componen, opciones como añadir a favoritos, o incluso realizar modificaciones/eliminación de la misma si esta fuera de su propiedad.

Actores: Usuario móvil.

Precondiciones: Estar en Mis plantillas, favoritos o haber buscado plantillas mediante el caso de uso "buscador".

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. Tras pulsar en el nombre de una plantilla de entrenamiento se accederá al listado de los ejercicios que contiene la plantilla desde la que podrá:
   1.1. Ver la descripción de cada ejercicio.
   1.2. Añadirla a favoritos.
   1.3. Iniciar el entrenamiento.
   1.4. [Si accede desde "Favoritos" además podrá]
      1.4.1. Eliminar plantillas de su lista de favoritos al pulsar el botón de la estrella.
   1.5. [Si accede desde "Mis plantillas" además podrá]
      1.5.1. Añadir nuevos ejercicios.
      1.5.2. Eliminar ejercicios.
      1.5.3. Eliminar la plantilla de entrenamiento actual.

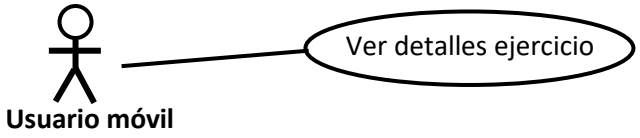
Postcondiciones: Se mostrará el listado de ejercicios que la componen y las diferentes opciones sobre la misma.

Interfaz gráfica:

Ilustración 141: App: Ver detalles plantilla no favorita

Ilustración 142: App: Ver detalles plantilla favorita

Ilustración 143: App: Ver detalles plantilla propia



Nombre: Ver detalles ejercicio.

Descripción: Permite al usuario identificado que visualice las características que definen el ejercicio que ha seleccionado. Entre ellas el nombre, la imagen con la posición a adoptar y la descripción del mismo.

Actores: Usuario móvil.

Precondiciones: Estar visualizando los detalles de una plantilla.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

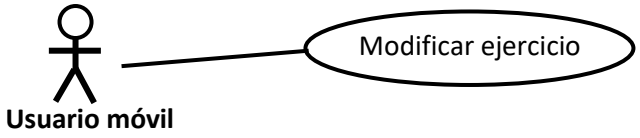
1. Al pulsar sobre el nombre del ejercicio se accederá a la descripción del mismo.
   1.1. [Si se accede a los detalles desde el área de "Mis plantillas"]
   1.2. El usuario podrá además realizar acciones de gestión sobre el mismo como:
      1.2.1.Eliminar el ejercicio de la plantilla actual.
      1.2.2.Realizar modificaciones en el número de series o repeticiones del mismo.

Postcondiciones: Se mostrará al usuario las características del ejercicio seleccionado.

Interfaz gráfica:

Ilustración 144: App: Ver detalles ejercicio plantilla ajena

Ilustración 145: App: Ver detalles ejercicio plantilla propia



Nombre: Modificar ejercicio.

Descripción: Permite al usuario identificado, tras consultar los detalles de una de sus plantillas, modifique los valores de entrenamiento del ejercicio que se quiera.

Actores: Usuario móvil.

Precondiciones: Estar visualizando los detalles del ejercicio de una plantilla de la que se es propietario.

Requisitos no funcionales: El rango de los valores numéricos a introducir estará limitado para evitar fallos.

Flujo de eventos:

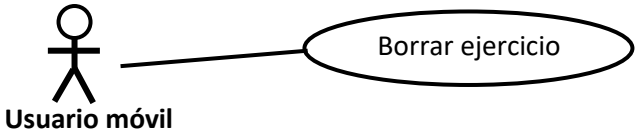
1. El usuario pulsará el icono del engranaje y accederá al área de modificación.
2. A continuación rellenará los campos que quiere modificar en el ejercicio actual y tras ello guardará los cambios pulsando “v”.
   2.1. [Si los campos no son correctos]
   Se mostrará un mensaje al usuario indicando los motivos del mismo.
   2.2. [Si los datos de los campos son correctos]
   Se realizarán las modificaciones en el ejercicio de la plantilla que se estaba visualizando y redirigirá al ejercicio.

Postcondiciones: El ejercicio de la plantilla a la que pertenece quedará modificado.

Interfaz gráfica:

Ilustración 146: App: Modificar ejercicio

Ilustración 147: App: Modificar ejercicio error



Nombre: Borrar ejercicio.

Descripción: Permite al usuario que, una vez identificado y tras consultar los detalles de una de sus plantillas, elimine el ejercicio de la plantilla actual.

Actores: Usuario móvil.

Precondiciones: Estar visualizando los detalles del ejercicio de una plantilla de la que se es propietario.

Requisitos no funcionales: Ninguno.

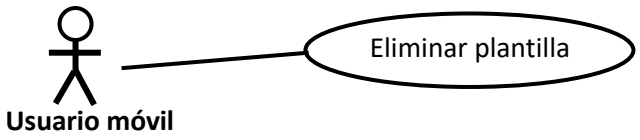
Flujo de eventos:

1. El usuario pulsará el icono del cubo de basura para eliminar el ejercicio.
2. Se mostrará un mensaje de advertencia de eliminación del ejercicio.
   2.1. [Si el usuario cancela la eliminación]
   No se eliminará el ejercicio y se volverá a la descripción del mismo.
   2.2. [Si el usuario confirma la eliminación]
   El ejercicio será eliminado de la plantilla actual y le redirigirá a la plantilla.

Postcondiciones: El ejercicio no formará parte de la plantilla actual.

Interfaz gráfica:

Ilustración 148: Eliminar ejercicio aviso



Nombre: Eliminar plantilla.

Descripción: Permite al usuario que, una vez identificado pueda eliminar la plantilla actual.

Actores: Usuario móvil.

Precondiciones: Estar visualizando los detalles de una plantilla desde el área “Mis Plantillas”.

Requisitos no funcionales: Ninguno.

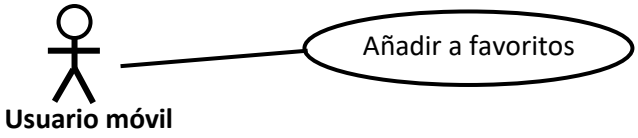
Flujo de eventos:

1. El usuario pulsará sobre el icono del cubo de basura para proceder a eliminar la plantilla.
2. Se mostrará un mensaje de advertencia de eliminación de la plantilla actual.
   2.1. [Si el usuario cancela la eliminación]
   No se eliminará la plantilla actual y se devolverá a la descripción de la plantilla.
   2.2. [Si se confirma la eliminación de la plantilla]
   La plantilla será eliminada del sistema y se redirigirá a la vista principal de “Mis plantillas”.

Postcondiciones: Todo rastro de la plantilla actual quedará eliminado del sistema.

Interfaz gráfica:

Ilustración 149: App: Eliminar plantilla aviso



Nombre: Añadir a favoritos.

Descripción: Permite al usuario que, una vez identificado, pueda añadir a favoritos la plantilla actual. En caso de estar visualizando las plantillas de "favoritos", esta sería eliminada de la lista.

Actores: Usuario móvil.

Precondiciones: Estar visualizando los detalles de una plantilla.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. El usuario pulsará sobre el botón de la estrella.
   1.1. [Si la plantilla ya está en favoritos]
   La plantilla será eliminada de favoritos y un mensaje advertirá al usuario.
   1.2. [Si la plantilla no está añadida a favoritos]
   Un mensaje advertirá al usuario de que la plantilla ha sido añadida a favoritos.

Postcondiciones: La plantilla se añadirá a favoritos.

Interfaz gráfica:

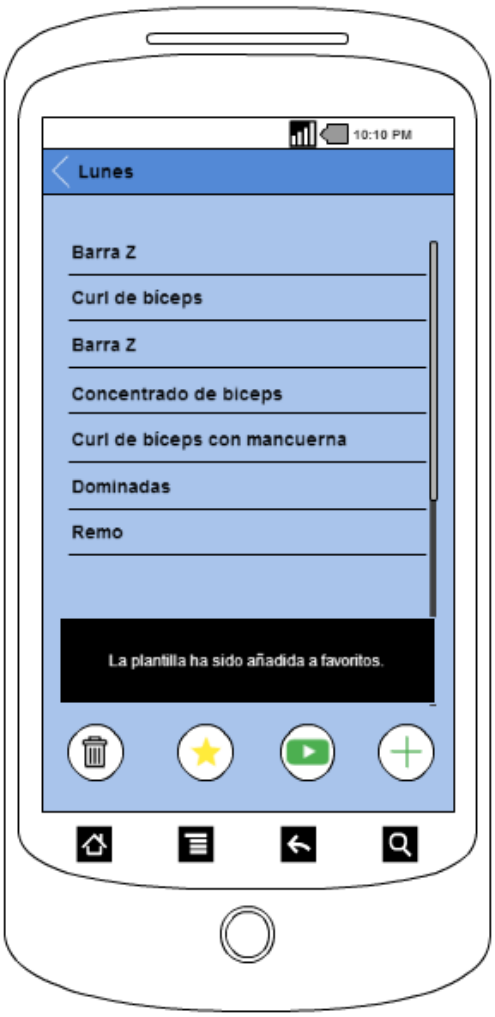
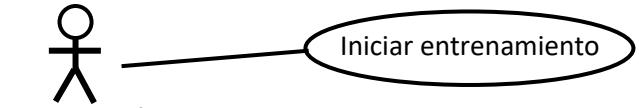


Ilustración 150: App: Añadir favoritos

Ilustración 151: App: Eliminar favoritos



Nombre: Iniciar entrenamiento.

Descripción: Permite al usuario, una vez identificado, pueda comenzar el entrenamiento de la plantilla actual.

Actores: Usuario móvil.

Precondiciones: Estar visualizando los detalles de una plantilla.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. El usuario pulsará sobre el icono de "play" para iniciar el entrenamiento.
2. Se mostrará el ejercicio a realizar con los detalles del mismo y el sistema calculará el ritmo de los ejercicios, avisando al usuario de las transiciones.
   2.1. [Si el usuario pulsa pause]
   La cuenta del ejercicio quedará en pausa a la espera de continuar o cancelar.
   2.1.1.[Si el usuario pulsa "play"]
   El entrenamiento continuará.
   2.2. [Si se finaliza el número de repeticiones de la serie actual]
   Comenzará el tiempo de descanso con el cronómetro y tras ello, dará paso al siguiente ejercicio.

Postcondiciones: Comenzará el contador de tiempos de la actividad así como los avisos acústicos.

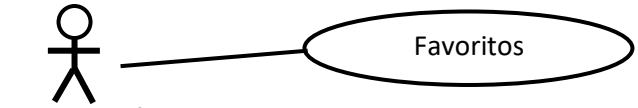
Interfaz gráfica:

Ilustración 152: App: Iniciar entrenamiento Vista ejercicio

Ilustración 153: App: Iniciar entrenamiento Vista cronómetro

Ilustración 154: App: Iniciar entrenamiento Vista ejercicio pausa

Ilustración 155: App: Iniciar entrenamiento Vista cronómetro pausa



Nombre: Favoritos.

Descripción: Tras haber iniciado sesión, permite que el usuario acceda a una lista de aquellas listas que haya seleccionado como favorita. Dicha lista podrá contener tanto plantillas de entrenamiento suyas como de otras personas, y al seleccionarlas podrá acceder a sus detalles así como a diferentes casos de uso en común desde otras áreas.

Actores: Usuario móvil.

Precondiciones: Haber iniciado sesión en la aplicación y mantener la conexión de internet.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. Se deberá seleccionar la opción “Favoritos”.
2. El sistema se posicionará en dicha área, desde la que el usuario podrá:
   2.1. Consultar el listado de plantillas que ha designado como favoritas.
   2.2. Visualizar los detalles de todas y cada una de ellas.

Postcondiciones: Se mostrará un listado con las plantillas favoritas del usuario actual.

Interfaz gráfica:

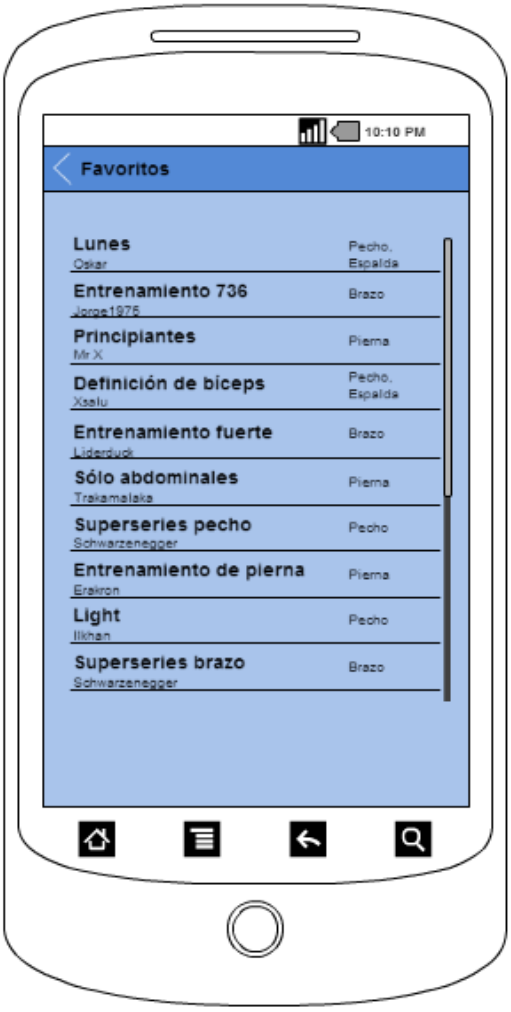
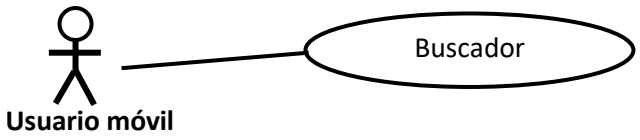


Ilustración 156: App: Favoritos



Nombre: Buscador.

Descripción: Tras haber iniciado sesión, permite que el usuario pueda realizar búsquedas relacionadas con plantillas y con ejercicios y así poder visualizar la información de los ejercicios en cualquier momento o bien realizar entrenamientos diseñados por otras personas.

Actores: Usuario móvil.

Precondiciones: Haber iniciado sesión en la aplicación y mantener la conexión de internet.

Requisitos no funcionales: El tipo de búsqueda a realizar (plantillas, ejercicios) será seleccionado por el usuario.

Flujo de eventos:

1. El usuario selecciona la opción "Buscador".
2. A continuación rellenará los tags que quiera buscar y seleccionará el tipo de búsqueda: plantillas o ejercicios.
3. Finalmente pulsará sobre el icono de la lupa para comenzar la búsqueda.
   - 3.1. [Si el usuario busca plantillas]
     - 3.1.1.[Si no existe ninguna coincidencia con los valores introducidos]
       Se mostrará un mensaje avisando de que no existen resultados.
     - 3.1.2.[Si existen coincidencias en la BD]
       Se le muestra un listado usuario para que pueda consultar detalles varios de las mismas.
   - 3.2. [Si el usuario busca ejercicios]
     Se mostrará un listado de los ejercicios del grupo muscular que haya seleccionado desde el cual podrá consultar los detalles.

Postcondiciones: Se mostrará un listado de elementos cuyas características coincidan con las especificadas en la búsqueda.

Interfaz gráfica:

Ilustración 157: App: Buscador vista principal

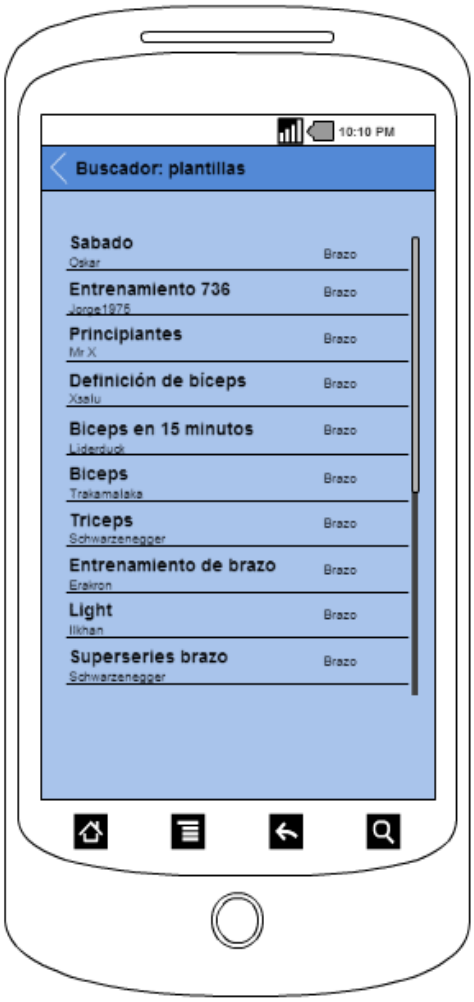


Ilustración 158: App: Buscador de plantillas

Ilustración 159: App: Buscador de plantillas sin resultados

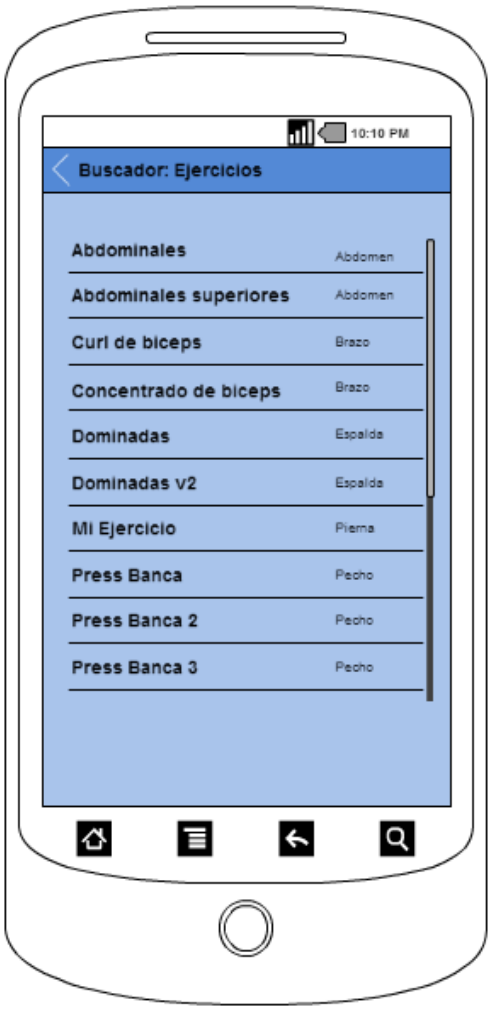
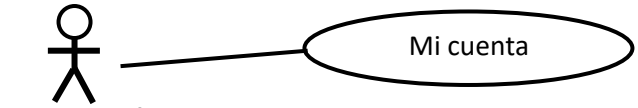


Ilustración 160: App: Buscador de ejercicios



Nombre: Mi cuenta.

Descripción: Permite al usuario, una vez se ha identificado, que pueda acceder al área de gestión de cuentas.

Actores: Usuario móvil.

Precondiciones: Haber iniciado sesión en la aplicación y mantener la conexión de internet.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. Se deberá seleccionar la opción “Mi cuenta”.
2. El sistema accederá al área desde la que el usuario podrá:
   2.1. Realizar modificaciones en su perfil de usuario.
   2.2. Cerrar la sesión en el dispositivo actual.

Postcondiciones: El usuario accederá al área de cuentas.

Interfaz gráfica:

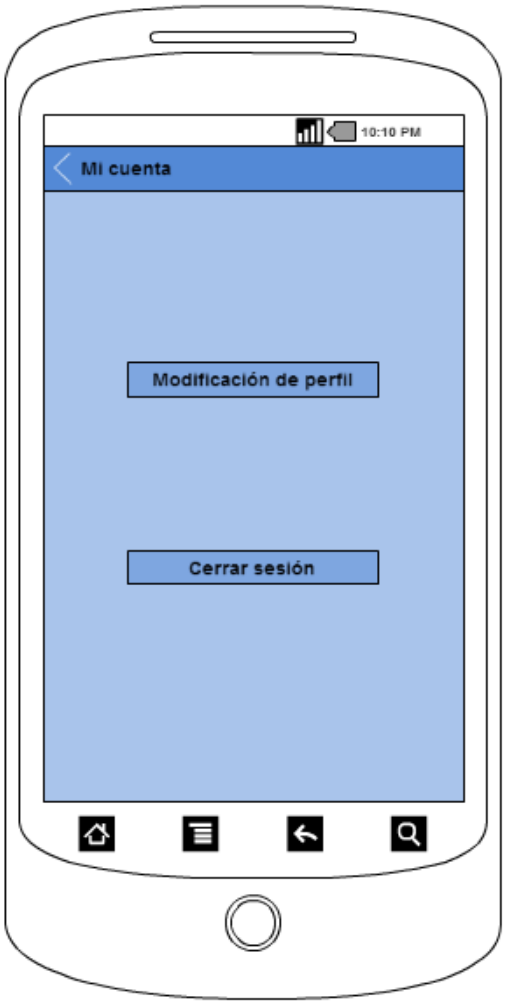
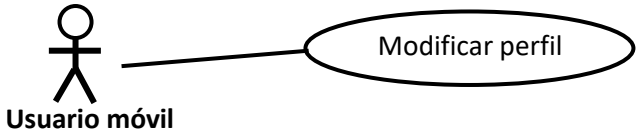


Ilustración 161: App: Mi Cuenta



Nombre: Modificar perfil.

Descripción: Permite al usuario identificado que pueda realizar modificaciones en sus datos personales.

Actores: Usuario móvil.

Precondiciones: Acceder al área de "Mi Cuenta".

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. Primero se pulsará sobre el botón de modificación de perfil y con ello accederá al formulario de modificación.
2. El usuario podrá modificar aquellos campos que considere necesarios y pulsará "v".
   2.1. [Si ha habido algún error al rellenar los campos]
   Se mostrará un mensaje de error del mismo.
   2.2. [Si todo es correcto]
   Se procederá a verificar la identidad del usuario pidiéndole su antigua contraseña por pantalla.
   2.2.1.[Si la contraseña facilitada es errónea]
   Se avisará del fallo y tendrá que volver a escribirla.
   2.2.2.[Si la contraseña facilitada es correcta]
   Los datos se almacenarán en la BD y se avisará al usuario de ello.

Postcondiciones: Los datos modificados serán guardados en el dispositivo.

Interfaz gráfica:

Ilustración 162: App: Modificar perfil

Ilustración 163: App: Modificar perfil error contraseña

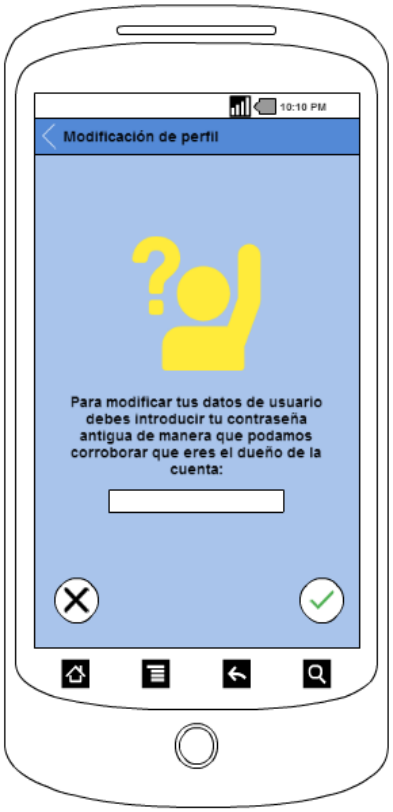


Ilustración 164: App: Modificar perfil verificación

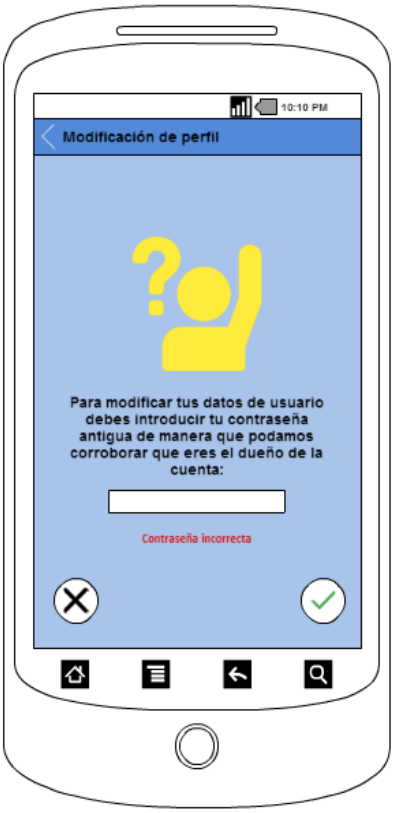
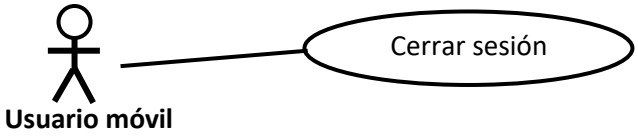


Ilustración 165: App: Modificar perfil verificación error

Ilustración 166: App: Modificar perfil éxito



Nombre: Cerrar sesión.

Descripción: Cierra la sesión de usuario actual y elimina los datos que se almacenan en la parte local del dispositivo móvil para que de esta manera se pueda iniciar sesión con una cuenta diferente y dicho proceso no sea automático.

Actores: Usuario móvil.

Precondiciones: Acceder al área de "Mi Cuenta".

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. Se deberá pulsar sobre la opción "Cerrar sesión".
2. Se avisará de que está a punto de finalizar la sesión actual y pedirá la confirmación del usuario.
3. El usuario pulsará "v" para cerrar sesión en el dispositivo y se eliminarán las credenciales.

Postcondiciones: La sesión activa en el dispositivo se eliminará, saliendo a la página de iniciar sesión.

Interfaz gráfica:

Ilustración 167: App: Cerrar sesión

ANEXO II:
DIAGRAMAS DE SECUENCIA

Aplicación web

Registro web:

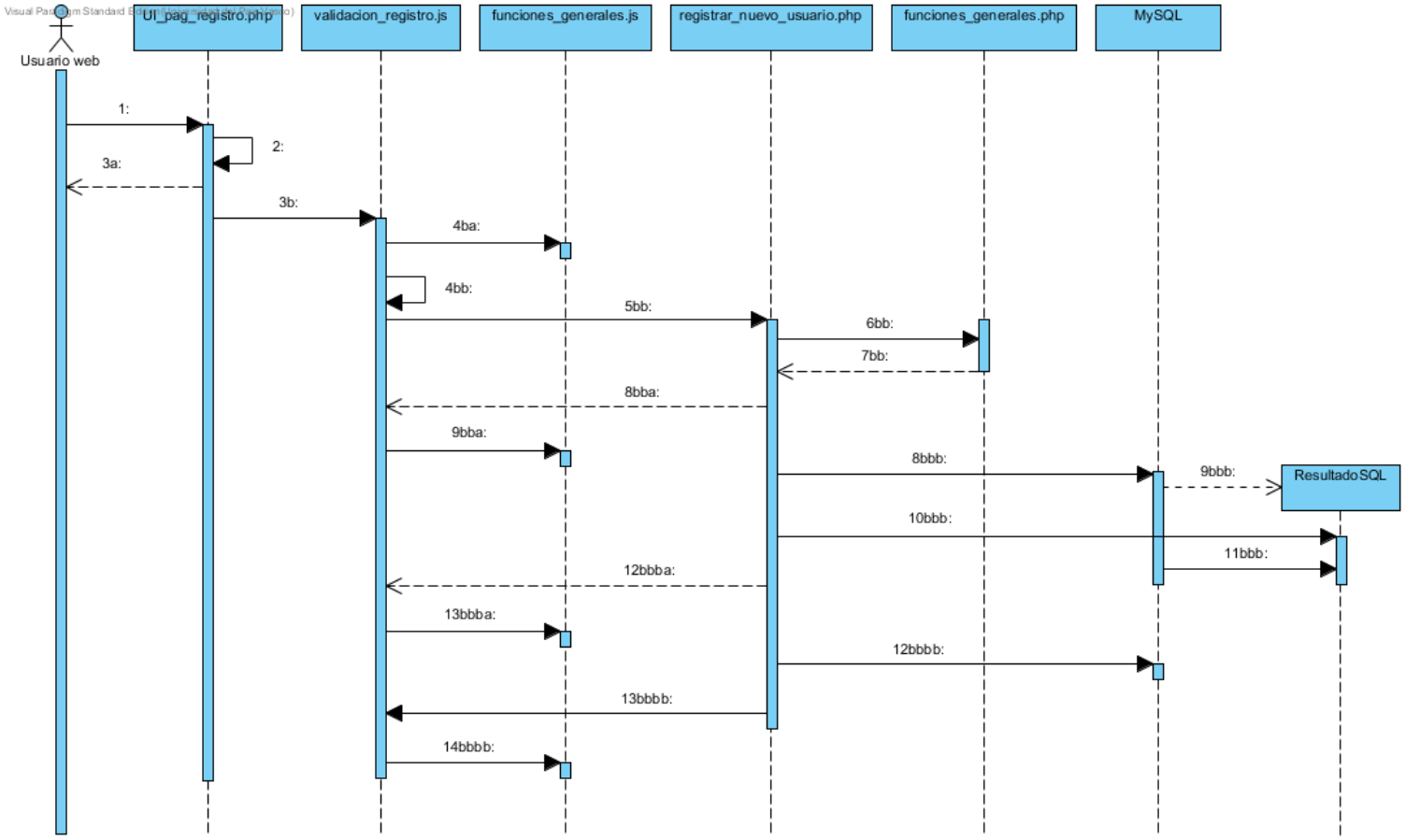


Ilustración 168: Diagrama de secuencia Web: Registro

1- El usuario rellena el formulario de registro y pulsa sobre “Registrarse”.
2- validarRegistroHTML(form);
[Si la validación de HTML detecta errores]
3a- html.setCustomValidation(error);
[Si la validación ha sido correcta]
3b- validarFormulario(form);
[Si la validación de JavaScript falla]
4ba- mostrarDivError(“las contraseñas no coinciden”)
[Si la validación de JS es correcta]
4bb- llamadaAJAX();
5bb- validarFormulario(nick, nombre, apellidos, email, pass, pass2);
6bb- validarFormulario(nick, nombre, apellidos, email, pass, pass2);
7bb- return $validacion;
[Si validación tiene errores]
8bba- return $msgError;
9bba- mostrarDivError($msgError);
[Si validación correcta]
8bbb- execSQL(Consulta1);

```
SELECT * FROM usuarios
WHERE nickUsuario = '$_POST[nick]'
OR email = '$_POST[email]'
```

9bbb- new();
10bbb- mysqli_num_rows();
11bbb- destroy();
[Si la consulta devuelve alguna línea]
12bbba- return $error= “Nick de usuario o correo ya está en uso”
13bbba- mostrarDivError($error);
[Si no ha devuelto líneas]
12bbbb- execSQL(Consulta2);

```
INSERT INTO usuarios (nickUsuario, pass, nombre, apellidos, email)
VALUES ('$_POST[nick]', md5('$_POST[pass]'), '$_POST[nombre]',
'$_POST[apellidos]', '$_POST[email]')
```

13bbbb- return “ “;
14bbbb- mostrarDivÉxito();

Identificación web:

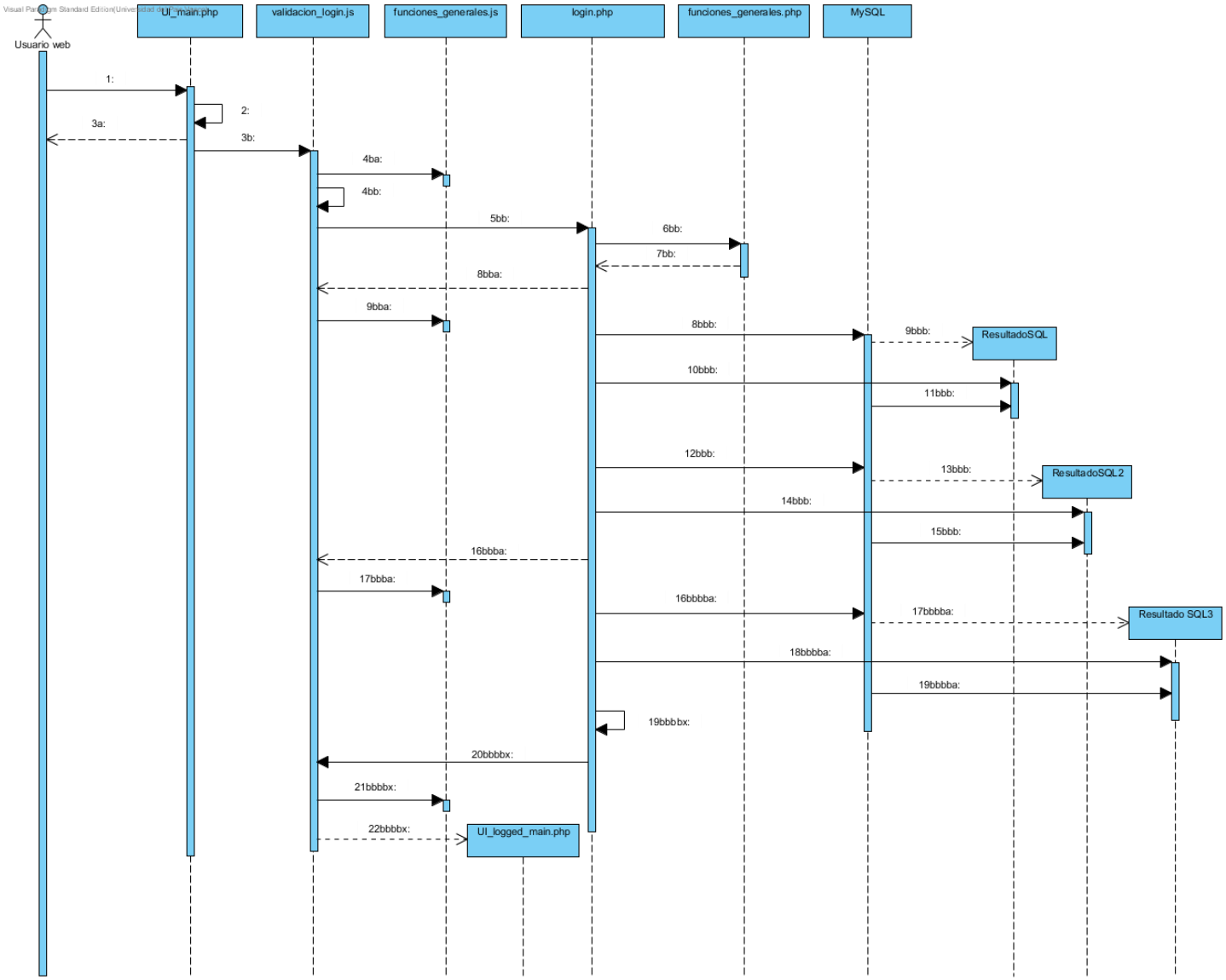


Ilustración 169: Diagrama de secuencia Web: Identificación

1- Tras introducir los datos en el formulario el usuario pulsa en “Iniciar Sesión”.
2- validarLoginHTML(form);
[Si la validación de HTML detecta errores]
3a- html.setCustomValidation(error);
[Si la validación ha sido correcta]
3b- validarLogin (form);
[Si la validación de JavaScript falla]
4ba- mostrarDivError(“las contraseñas no coinciden”)
[Si la validación de JS es correcta]
4bb- llamadaAJAX();
5bb- validarFormulario(nick, pass);
6bb- validarFormulario(nick, pass);
7bb- return $validacion;
[Si validación tiene errores]
8bba- return $msgError;
9bba- mostrarDivError($msgError);
[Si validación correcta]
8bbb- execSQL(Consulta1);

```
SELECT * FROM usuarios
WHERE nickUsuario = '$_POST[nick]'
&& pass = md5('$_POST[pass]')
```

9bbb- new();
10bbb- mysqli_num_rows();
11bbb- destroy();
12bbb- execSQL(Consulta2)

```
SELECT * FROM usuarios
WHERE email = '$_POST[nick]'
&& pass = md5('$_POST[pass]')
```

13bbb- new()
14bbb- mysqli_num_rows()
15bbb- destroy()
[Si ninguna consulta ha devuelto alguna línea]
16bbba- return $error= “Nick de usuario o contraseña incorrectos”
17bbba- mostrarDivError($error);

[Si alguna consulta ha devuelto líneas]

[Si inició sesión introduciendo email]

16bbbba- execSQL(Consulta3);

```
SELECT nickUsuario FROM usuarios
WHERE email= '$_POST[nick]'
```

17bbbba- new();

18bbbba- get(nombreDeUsuario);

19bbbba- destroy();

[Si inició sesión introduciendo el nick de usuario]

19bbbbx- $_SESSION(nickUsuario);

20bbbbx- return " ";

21bbbbx- mostrarDivÉxito();

22bbbbx- header("pag_logged_main.php");

Crear ejercicio:

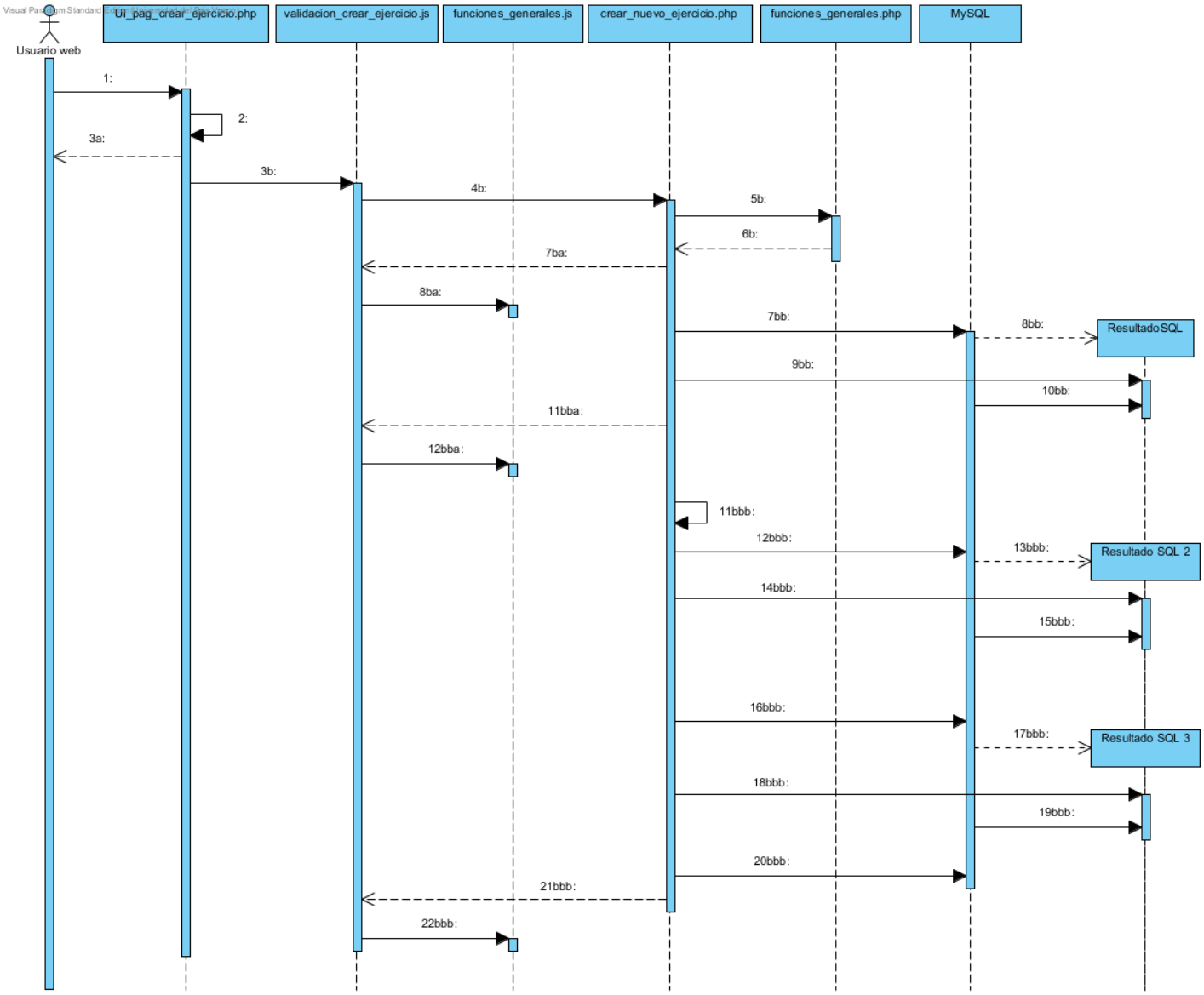


Ilustración 170: Diagrama de secuencia Web: Crear ejercicio

1- El usuario rellena el formulario de creación de ejercicio y pulsa “Guardar ejercicio”.
2- validarRegistroHTML(form)
[Si la validación de HTML detecta errores]
3a- html.setCustomValidation(error);
[Si la validación ha sido correcta]
3b- llamadaAJAX();
4b- validarFormulario(nombre, grupoMuscular, url, descripcion);
5b- validarFormulario(nombre, grupoMuscular, url, descripcion);
6b- return $validacion;
[Si validación tiene errores]
7ba- Return $msgError;
8ba- mostrarDivError($msgError);
[Si validación correcta]
7bb- execSQL(Consulta1);

```
SELECT * FROM ejercicios
WHERE nombreEjer = '$nombreEjer'
```

8bb- new();
9bb- mysqli_num_rows();
10bb- destroy();
[Si la consulta ha devuelto líneas (ejercicio en uso)]
11bba- return $msgError = “nombre de ejercicio en uso”;
12bba- mostrarDivError($msgError);
[Si la consulta no ha devuelto líneas]
11bbb- procesarGif();
12bbb- execSQL(Consulta2);

```
SELECT * FROM tags
WHERE grupoMuscular = '$grupoMuscular'
```

13bbb- new();
14bbb- get(idGrupoMuscular);
15bbb- destroy();
16bbb- execSQL(Consulta3);

```
SELECT * FROM usuarios
WHERE  nickUsuario= '$nickUsuario'
```

17bbb- new();
18bbb- get(idUsuario);
19bbb- destroy();
20bbb- execSQL(Consulta4);

```
INSERT INTO ejercicios (nombreEjer, grupoMuscular, imagen, descripcion, nickUsuario)
VALUES ('$nombreEjer', '$idGrupo', '$imagen', '$descripcion', '$idUsuario')
```

21bbb- return “”;
22bbb- mostrarDivExito();

Modificar ejercicio:

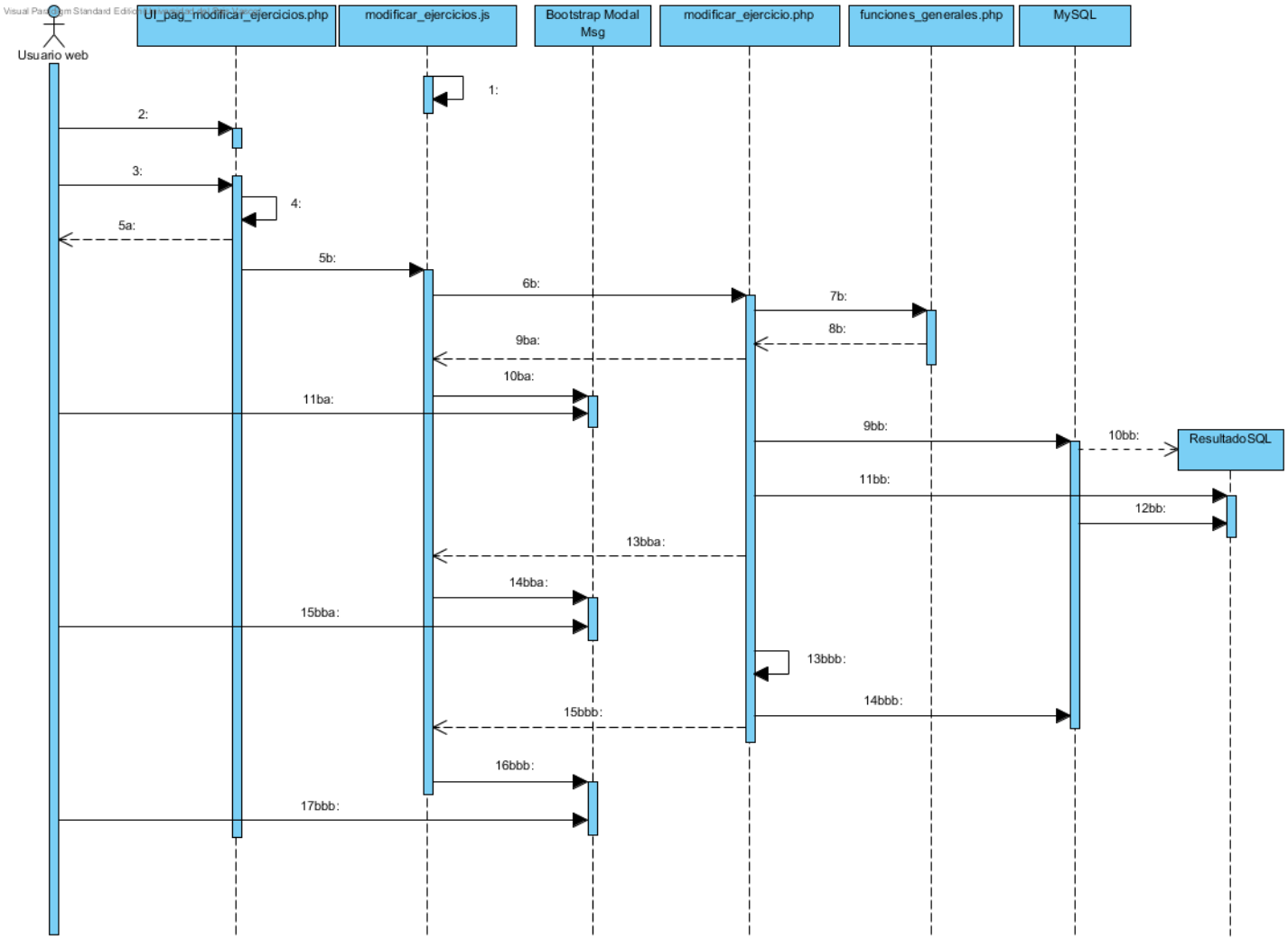


Ilustración 171: Diagrama de secuencia Web: Modificar ejercicio

1- El sistema carga la tabla de ejercicios del usuario (ver diagrama de secuencia “Cargar tabla de ejercicios”).
2- Al pinchar sobre un ejercicio el sistema muestra en pantalla sus datos (ver diagrama de secuencia “Cargar datos ejercicio”).
3- Tras modificar los datos necesarios el usuario pulsa sobre “Guardar ejercicio”.
4- validarFormularioHTML(form)

[Si la validación de HTML detecta errores]

5a- html.setCustomValidation(error);

[Si la validación ha sido correcta]

5b- llamadaAJAX();
6b- validarFormulario(nombreEjer, imagen, descripcion);
7b- validarFormulario(nombreEjer, imagen, descripcion);
8b- return $validacion;

[Si validación tiene errores]

9ba- Return $msgError;
10ba- mostrarModalError($msgError);
11ba- el usuario pulsará sobre “Aceptar” para poder continuar.

[Si validación correcta]

9bb- execSQL(Consulta1);

```
SELECT * FROM ejercicios
WHERE nickUsuario = '$_SESSION[usuario]'
&& nombreEjer='$nombreEjer'
```

10bb- new();
11bb- mysqli_num_rows();
12bb- destroy();

[Si la consulta no devuelve líneas (ejercicio no es de su propiedad)]

13bba- return $msgError = “Ejercicio no es de su propiedad. Recargue la página”;
14bba- mostrarModalError($msgError);
15bbb- el usuario pulsará sobre “Aceptar” para poder continuar.

[Si la consulta ha devuelto líneas]

13bbb- procesarGif();
14bbb- execSQL(Consulta2);

```
UPDATE ejercicios
SET imagen='$imagen', descripcion='$descripcion'
WHERE nombreEjer='$nombreEjer'
&& nickUsuario='$_SESSION[usuario]'
```

15bbb- echo “”;
16bbb- mostrarModalExito();
17bbb- el usuario pulsará sobre “Aceptar” para poder continuar.

Eliminar ejercicio:

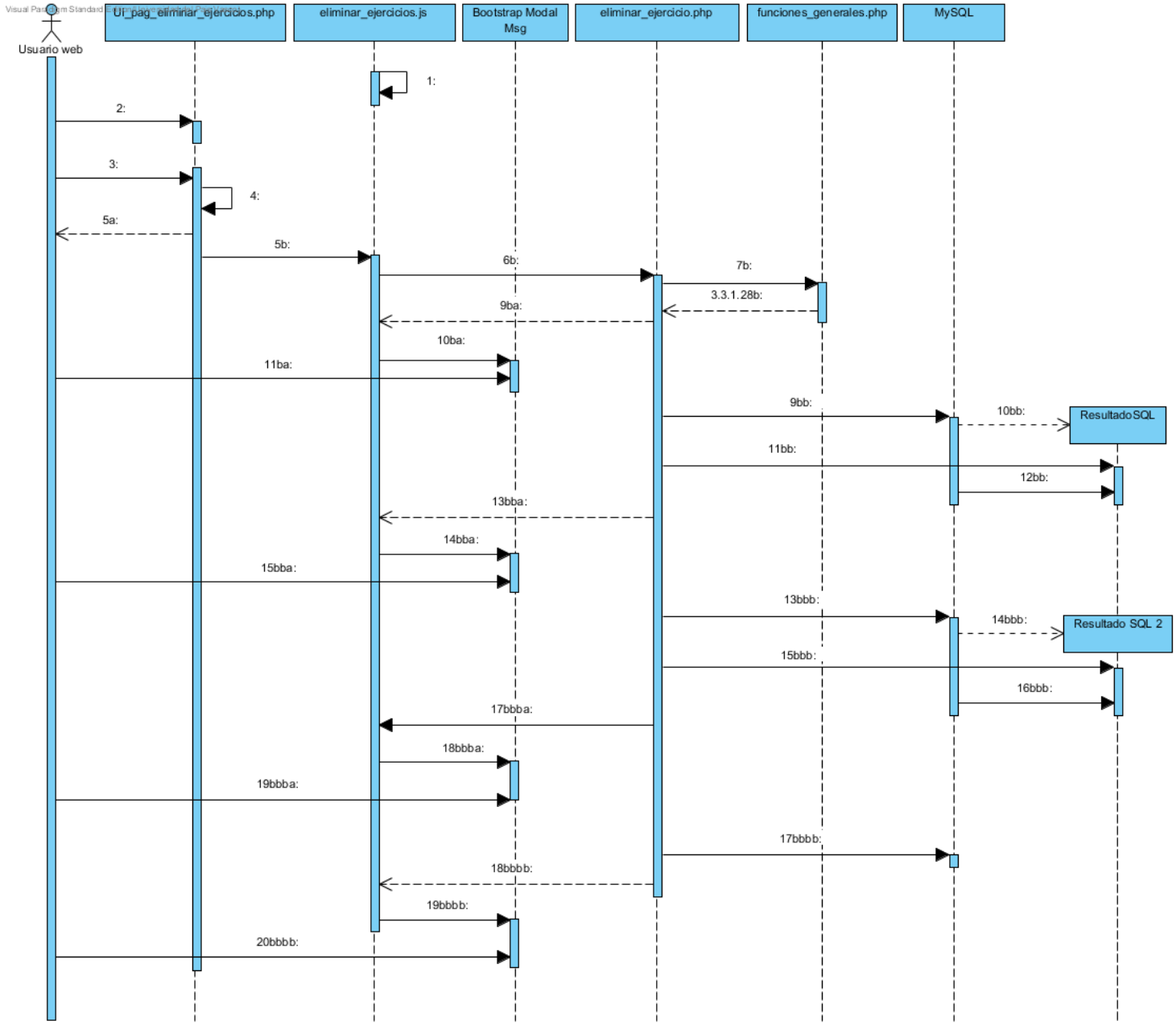


Ilustración 172: Diagrama de secuencia Web: Eliminar ejercicio

1- El sistema carga la tabla de ejercicios del usuario (ver diagrama de secuencia “Cargar tabla de ejercicios”).
2- Al pinchar sobre un ejercicio el sistema muestra en pantalla sus datos (ver diagrama de secuencia “Cargar datos ejercicio”).
3- El usuario pulsará sobre “Eliminar”.
4- validarFormularioHTML(form);
[Si la validación de HTML detecta errores]
5a- html.setCustomValidation(error);
[Si la validación ha sido correcta]
5b- llamadaAJAX();
6b- validarFormulario(nombreEjer);
7b- validarFormulario(nombreEjer);
8b- return $validacion;
[Si validación tiene errores]
9ba- Return $msgError;
10ba- mostrarModalError($msgError);
11ba- el usuario pulsará sobre “Aceptar” para poder continuar.
[Si validación correcta]
9bb- execSQL(Consulta1);

```
SELECT * FROM ejercicios
WHERE nickUsuario= '$_SESSION[usuario]'
&& nombreEjer= '$nombreEjer'
```

10bb- new();
11bb- mysqli_num_rows();
12bb- destroy();
[Si la consulta no devuelve líneas (ejercicio no es de su propiedad)]
13bba- return $msgError = “Ejercicio no encontrado. Recargue la página”;
14bba- mostrarModalError($msgError);
15bba- el usuario pulsará sobre “Aceptar” para poder continuar.
[Si la consulta ha devuelto líneas]
13bbb- execSQL(Consulta2);

```
SELECT * FROM configuracionejercicio
WHERE nombreEjer= '$nombreEjer'
```

14bbb- new();
15bbb- mysqli_num_rows();
16bbb- destroy();

[Si la consulta anterior ha devuelto líneas (ejercicio en uso)]

17bbba- return $msgError =”Ejercicio en uso por otras personas. Imposible eliminar ejercicio”.
18bbba- mostrarModalError($msgError);
19bbba- el usuario pulsará sobre “Aceptar” para poder continuar.

[Si la consulta anterior no ha devuelto líneas]

17bbbb- execSQL(Consulta3);

```
DELETE FROM ejercicios
WHERE nombreEjer = '$nombreEjer'
```

18bbbb- return “”;
19bbbb- mostrarModalExito();
20bbbb-el usuario pulsará sobre “Aceptar” para poder continuar

Cargar tabla de ejercicios:

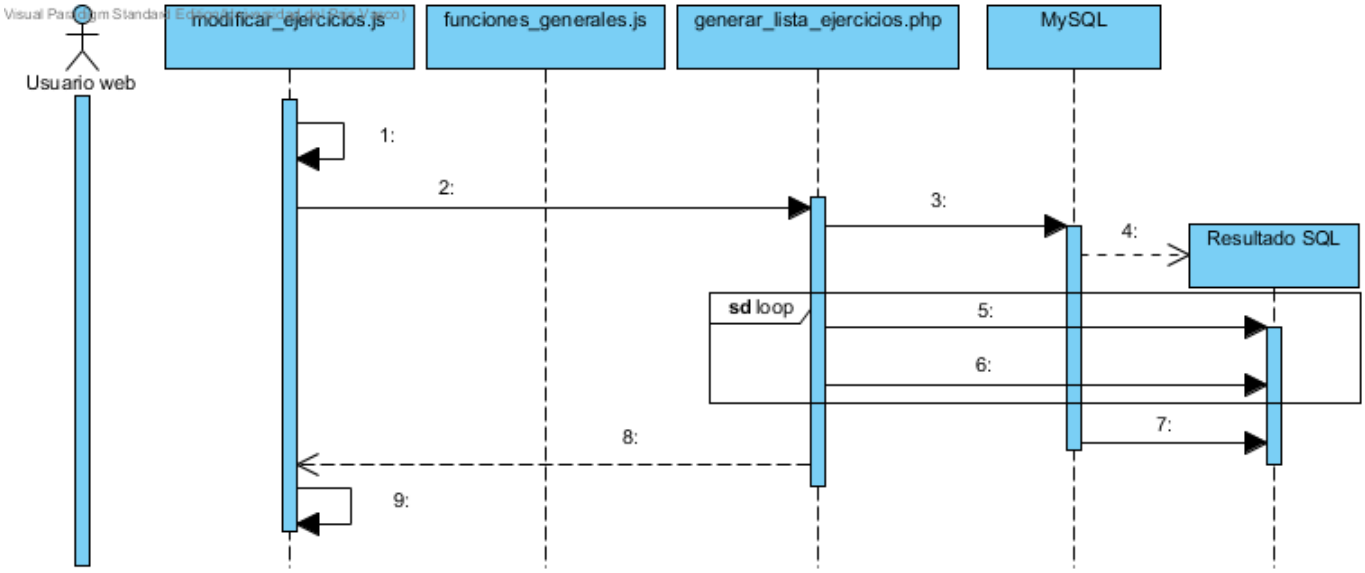


Ilustración 173: Diagrama de secuencia Web: Cargar tabla de ejercicios

La ejecución de este algoritmo comienza al cargar la página con la función onload de javaScript. Está presente al eliminar ejercicios y modificarlos. Para no repetir el mismo diagrama, el presentado es el correspondiente a "modificar ejercicios".

1- cargarTablaEjercicios();
2- cargarTablaEjerciciosAJAX();
3- execSQL(Consulta1);

```
SELECT * FROM ejercicios
WHERE nickUsuario = '$_SESSION[usuario]'
```

4- new();
[Por cada tupla devuelta se genera código HTML desde PHP para crear la tabla]
5- next();
6- get('NombreEjer')
[Fin loop]
7- destroy();
8- return $codigoHTML
9- tabla.innerHTML = $codigoHTML;

Cargar datos de ejercicio:

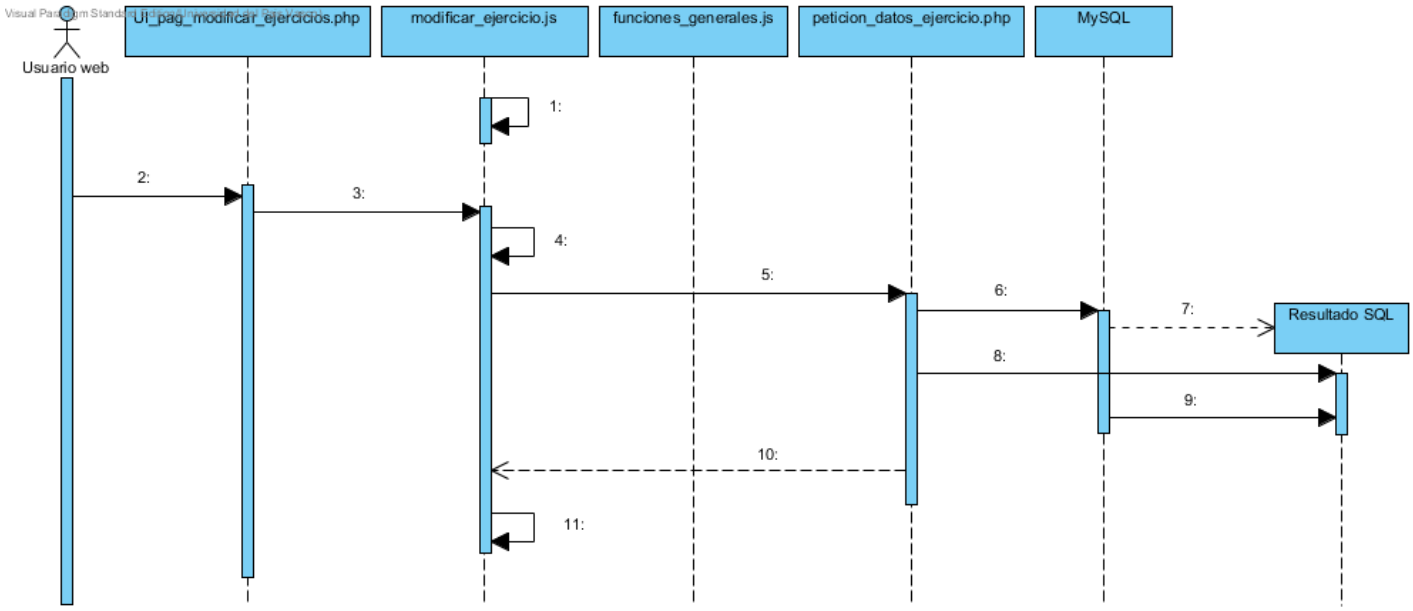


Ilustración 174: Diagrama de secuencia Web: Cargar datos ejercicio

Al igual que el diagrama anterior, este diagrama se aplica tanto al eliminar como al modificar un ejercicio. La versión presentada corresponde a "modificar ejercicios".

1- cargarTablaEjercicios();
2- El usuario selecciona un ejercicio de su listado en la UI correspondiente.
3- cargarEjercicio();
4- modificarEsteticaUI();
5- peticiónDatosEjerAJAX();
6- execSQL(Consulta1)

```
SELECT * FROM ejercicios
WHERE nombreEjer='$nombreEjer'
&& nickUsuario='$propietario'
```

7- new();
8- getDatosEjer();
9- destroy();
10- return JSONdatosEjercicio;
11- mostrarDatosEjercicio(JSONdatosEjercicio);

Modificar cuenta:

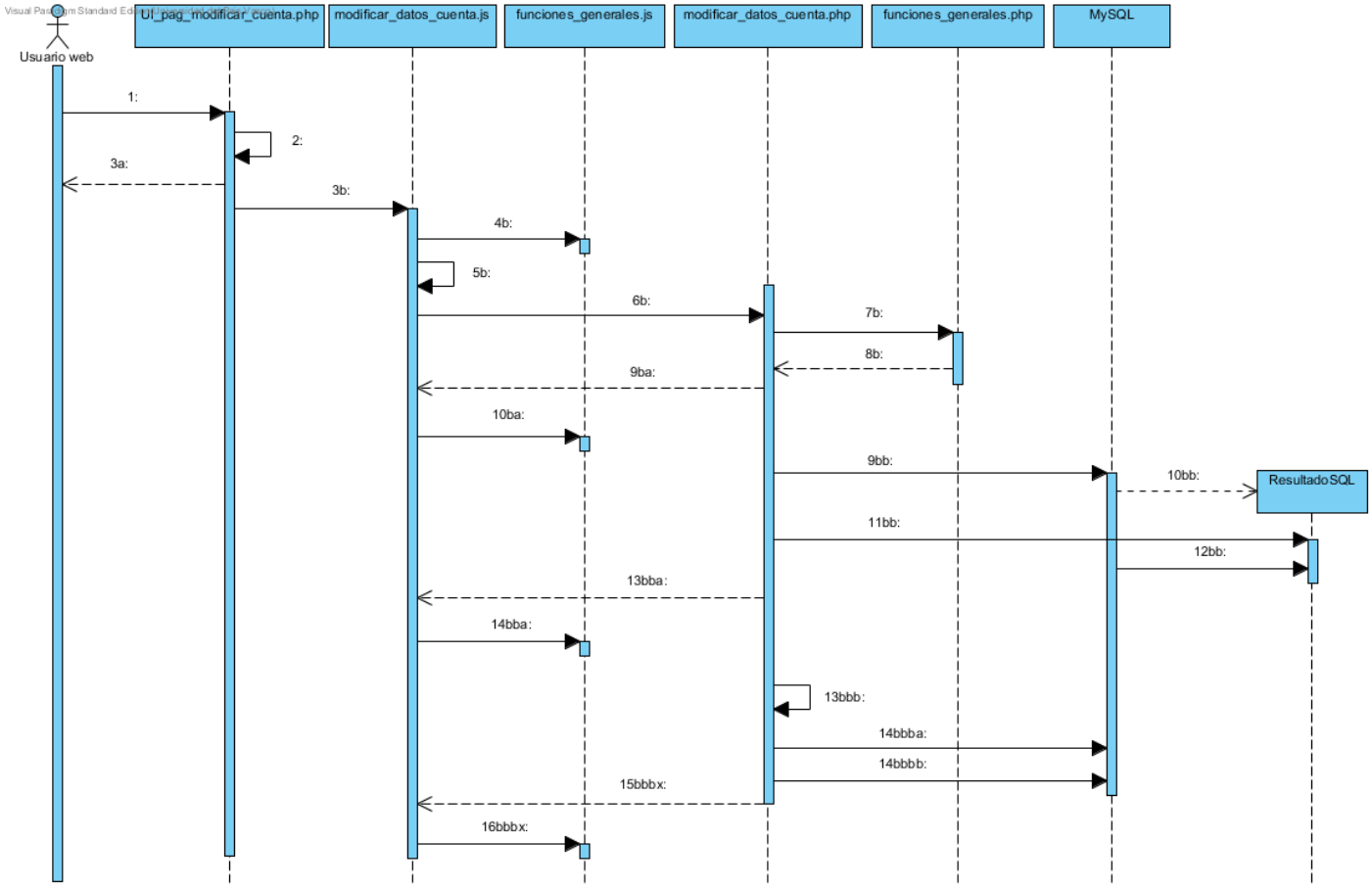


Ilustración 175: Diagrama de secuencia Web: Modificar cuenta

1- Tras modificar los datos deseados el usuario pulsará “Modificar”.
2- validarFormularioHTML(form);
[Si la validación de HTML detecta errores]
3a- html.setCustomValidation(error);
[Si la validación ha sido correcta]
3b- contraseñasSonIguales(pass, pass2);
[Si las contraseñas no son iguales]
4b- mostrarDivError(“Las contraseñas introducidas son diferentes”);
[Si las contraseñas son iguales]
5b- llamadaAJAX();
6b- validarFormulario(nombre, apellidos, email, pass, pass2);
7b- validarFormulario(nombre, apellidos, email, pass, pass2);
8b- return $validation();
[Si la validación ha resultado fallida]
9ba- return $errMsg();
10ba- mostrarDivError($errMsg);
[Si la validación ha sido satisfactoria]
9bb- execSQL(Consulta1);

```
SELECT * FROM usuarios
WHERE nickUsuario <> '$_SESSION[usuario]'
&& email = '$_POST[email]'
```

10bb- new();
11bb- mysqli_num_rows();
12bb- destroy();
[Si la consulta ha devuelto líneas (email en uso por otro usuario)]
13bba- return $errMsg = “Email introducido está en uso por otro usuario.”
14bba- mostrarDivError($errMsg);
[Si la consulta no ha devuelto líneas]
13bbb- passVacías();
[Si el usuario quiere actualizar contraseñas]
14bbba- execSQL(Consulta3)

```
UPDATE usuarios
SET nombre='$nombre', apellidos='$apellidos', email='$email'
WHERE nickUsuario='$_SESSION[usuario]'
```

[Si no quiere actualizar contraseñas]

14bbbb- execSQL(Consulta4);

```
UPDATE usuarios
SET pass=md5('$pass'), nombre='$nombre', apellidos='$apellidos', email='$email'
WHERE nickUsuario='$_SESSION[usuario]'
```

15bbbx- return " ";

16bbbx- mostrarDivExito();

Eliminar cuenta:

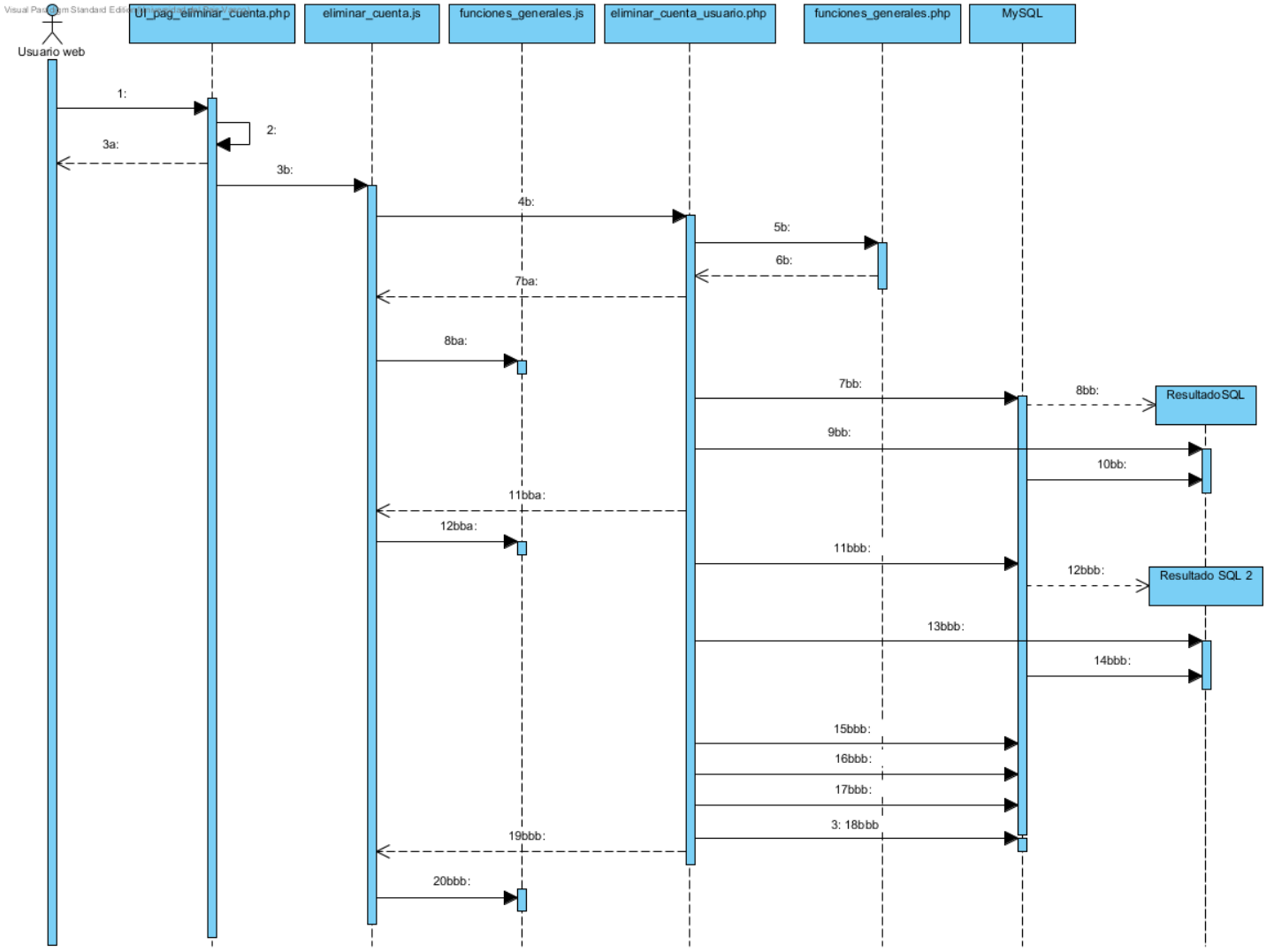


Ilustración 176: Diagrama de secuencia Web: Eliminar cuenta

1- Tras introducir su contraseña de cuenta el usuario pulsará “Eliminar”.
2- validarContraseñaHTML(form);
[Si la validación de HTML detecta errores]
3a- html.setCustomValidation(error);
[Si la validación ha sido correcta]
3b- llamadaAJAX();
4b- validarFormulario(pass);
5b- validarFormulario(pass);
6b- return $validation;
[Si la validación detecta errores]
7ba- return $errMsg;
8ba- mostrarDivError($errMsg);
[Si la validación es correcta]
7bb- execSQL(Consulta1);

```
SELECT * FROM usuarios
WHERE nickUsuario = '$_SESSION[usuario]'
&& pass=md5('$pass')
```

8bb- new();
9bb- mysqli_num_rows();
10bb- destroy();
[Si la consulta no ha devuelto líneas (contraseña incorrecta)]
11bba- return $errMsg = “La contraseña introducida es incorrecta”;
12bba- mostrarDivError($errMsg),
[Si la consulta devuelve líneas (contraseña correcta)]
11bbb- execSQL(Consulta2);

```
SELECT * FROM usuarios
WHERE  nickUsuario= '$master'
```

12bbb- new();
13bbb- get(idUsuario)
14bbb- destroy();
15bbb- execSQL(Consulta3);

```
UPDATE ejercicios
SET nickUsuario='$idMaster'
WHERE nickUsuario='$_SESSION[usuario]'
```

16bbb- execSQL(Consulta4);

```
UPDATE plantillas
SET nickUsuario='$idMaster'
WHERE nickUsuario='$_SESSION[usuario]'
```

17bbb- execSQL(Consulta5);

```
DELETE FROM favoritos
WHERE nickUsuario = '$_SESSION[usuario]'
```

18bbb- execSQL(Consulta6);

```
DELETE FROM usuarios
WHERE nickUsuario = '$_SESSION[usuario]'
&& pass=md5('$_POST[pass]')
```

19bbb- return “”;
20bbb- header(logout.php);

Cerrar sesión:

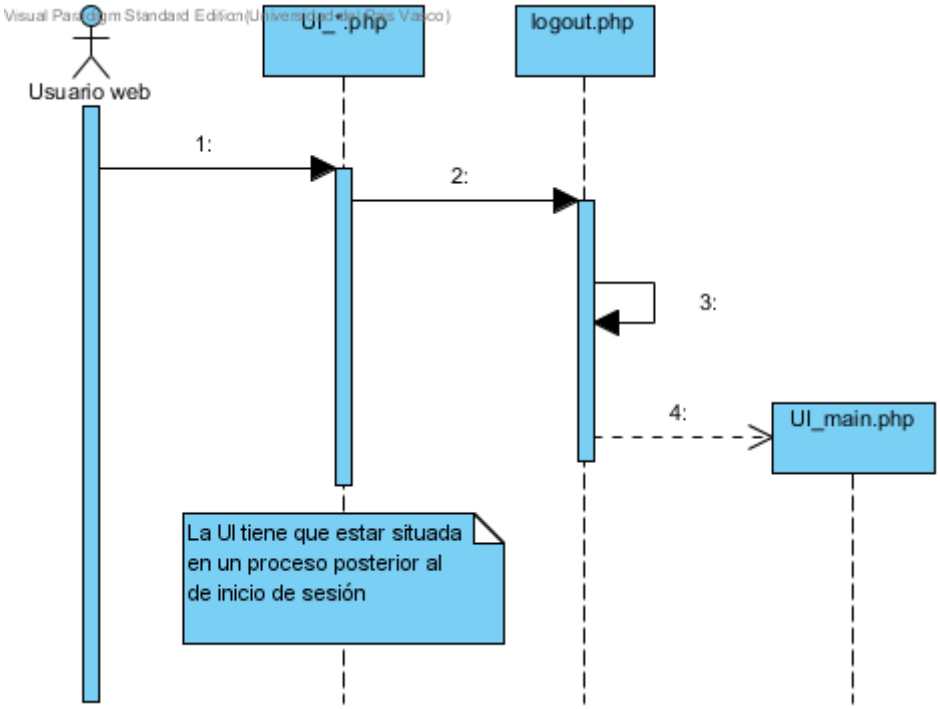


Ilustración 177: Diagrama de secuencia Web: Cerrar sesión

1- El usuario pulsa sobre el icono de cerrar sesión (el icono con forma de casa).
2- logout();
3- session_destroy();
4- header(main.php);

Aplicación Android

Nota: en estos diagramas todas las clases son .java a excepción de los ficheros .php en los cuales se indicará su extensión, MySQL o los resultados de las consultas.

Identificación App:

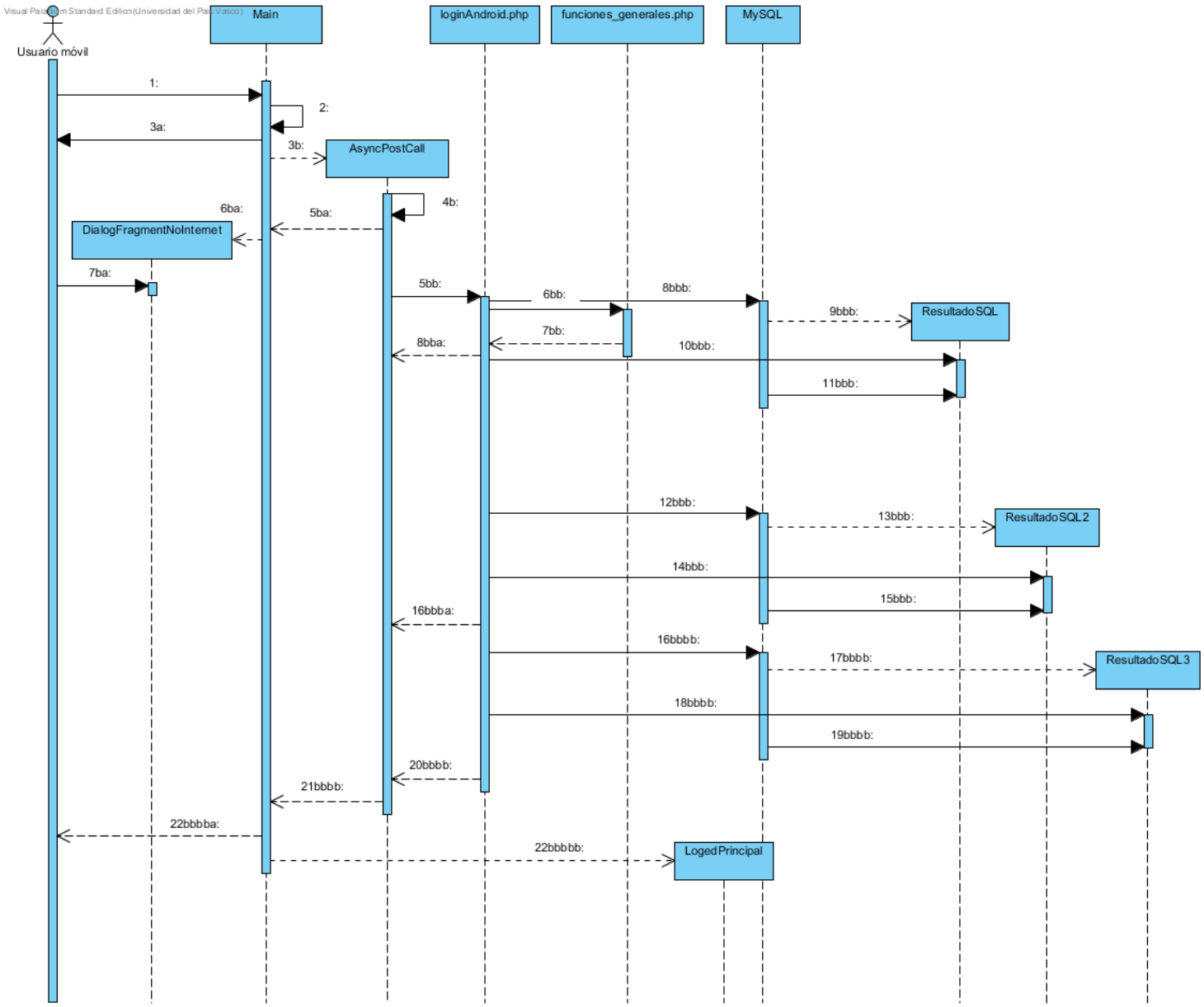


Ilustración 178: Diagrama de secuencia App: Identificación

1- Tras introducir las credenciales en los TextViews el usuario pulsará en “Iniciar Sesión”.
2- logearse();

[Si hay algún campo vacío]

3a- mostrar mensaje Toast con aviso de tal circunstancia.

[Si ha rellenado ambos campos]

3b- new AsyncPostCall(int: reqId; String: pUsuario, pPass).execute();

4b- comprobarConexion();

[Si no tiene conexión a internet]

5ba- return $response: JSON, (status=1, msg=“Cod11”);

6ba- new(DialogFragmentNoInternet.java);

7ba- El usuario pulsará aceptar.

[Si tiene conexión a internet]

5bb- validarFormulario(String: nick, pass);

6bb- validarFormulario(String: nick, pass);

7bb- return $validacion;

[Si la validación de PHP ha encontrado errores en los campos]

8bba- return $response: JSON, (status=1, msg=“Cod00”);

[Si la validación PHP ha resultado satisfactoria]

8bbb- execSQL(Consulta1);

```
SELECT * FROM usuarios
WHERE email = '$_POST[nick]'
&& pass = md5('$_POST[pass]')
```

9bbb- new();

10bbb- mysqli_num_rows();

11bbb- destroy();

12bbb- execSQL(Consulta2);

```
SELECT * FROM usuarios
WHERE nickUsuario = '$_POST[nick]'
&& pass = md5('$_POST[pass]')
```

13bbb- new();

14bbb- mysqli_num_rows();

15bbb- destroy();

[Si no se ha encontrado ningún resultado en ninguna de las consultas]

16bbba- return $response: JSON, (status=1, msg=“Cod00”);

[Si se ha encontrado la coincidencia]

[Si inició sesión introduciendo email]

16bbbb- execSQL(Consulta3);

```
SELECT nickUsuario FROM usuarios
WHERE email= '$_POST[nick]'
```

17bbbb- new();

18bbbb- get(nombre): String;

19bbbb- destroy();

[Si inició sesión con el nombre de usuario]

20bbbb- return $result: JSON, (status= 0, msg="");

21bbbb- return JSONObject;

[Si el campo status del JSON = 1 (error)]

22bbbba- mostrar un mensaje Toast en base al código del error del objeto JSON.

[Si el campo status del JSON= 0 (correcto)]

22bbbbb- startActivity();

Registro App:

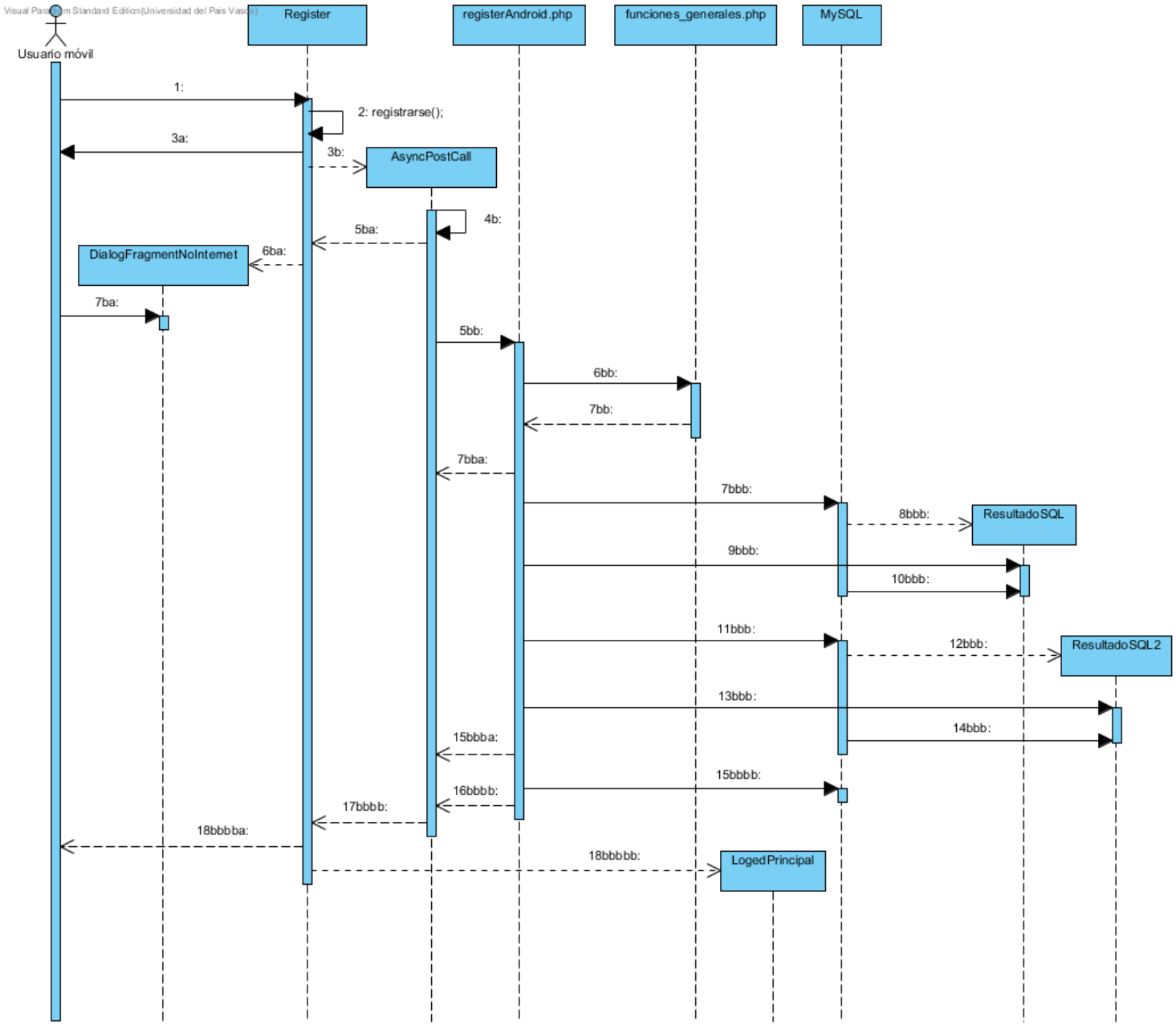


Ilustración 179: Diagrama de secuencia App: Registro

1- Completar los campos de registro y pulsar en “Registrarse”.
2- registrarse ();

[Si hay algún campo vacío o las contraseñas son diferentes]

3a- mostrar mensaje Toast informando del fallo correspondiente.

[Si está todo relleno y ambas contraseñas son iguales]

3b- new AsyncPostCall(int: reqId; String: pNick, pPass1, pPass2, pNombre, pApellido, pEmail).execute();

4b- comprobarConexion();

[Si no tiene conexión a internet]

5ba- return $response: JSONObject, (status=1, msg=“Cod11”);

6ba- new(DialogFragmentNoInternet.java);

7ba- El usuario pulsará aceptar.

[Si tiene conexión a internet]

5bb- validarFormulario(String: nick, pass1, pass2, nombre, apellido, email);

6bb- validarFormulario(String: nick, pass1, pass2, nombre, apellido, email);

7bb- return $validacion;

[Si la validación del PHP ha encontrado algún error]

7bba- return $response: JSON, (status=1, msg=“CodXX”) donde XX es el código del error {02,03,04,05,06,07,08};

[Si la validación del PHP ha resultado correcta]

7bbb- execSQL(Consulta1);

```
SELECT * FROM usuarios
WHERE nickUsuario= '$nick'
```

8bbb- new();

9bbb- mysqli_num_rows();

10bbb- destroy();

11bbb- execSQL(Consulta2);

```
SELECT * FROM usuarios
WHERE email= '$email'
```

12bbb- new();

13bbb- mysqli_num_rows();

14bbb- destroy();

[Si hay filas de una consulta u otra = usuario o email en uso]

15bbba- return $response: JSON, (status=1, msg=”CodXX”) donde XX es el código del error {09,10};

[Si el número de líneas es 0 en ambas consultas]

15bbbb- execSQL(Consulta3);

```
INSERT INTO usuarios (nickUsuario, pass, nombre, apellidos, email)
VALUES ('$nick',md5('$pass1'),'$nombre','$apellidos','$email')
```

16bbbb- return $result: JSON, (status=0, msg=””);

17bbbb- return JSONObject;

[Si el atributo status del objeto JSON = 1 (error)]

18bbbba- mostrar un mensaje Toast en base al código del error del objeto JSON.

[Si el atributo status del objeto JSON = 0 (correcto)]

18bbbbb- startActivity();

Pedir listado de ejercicios:

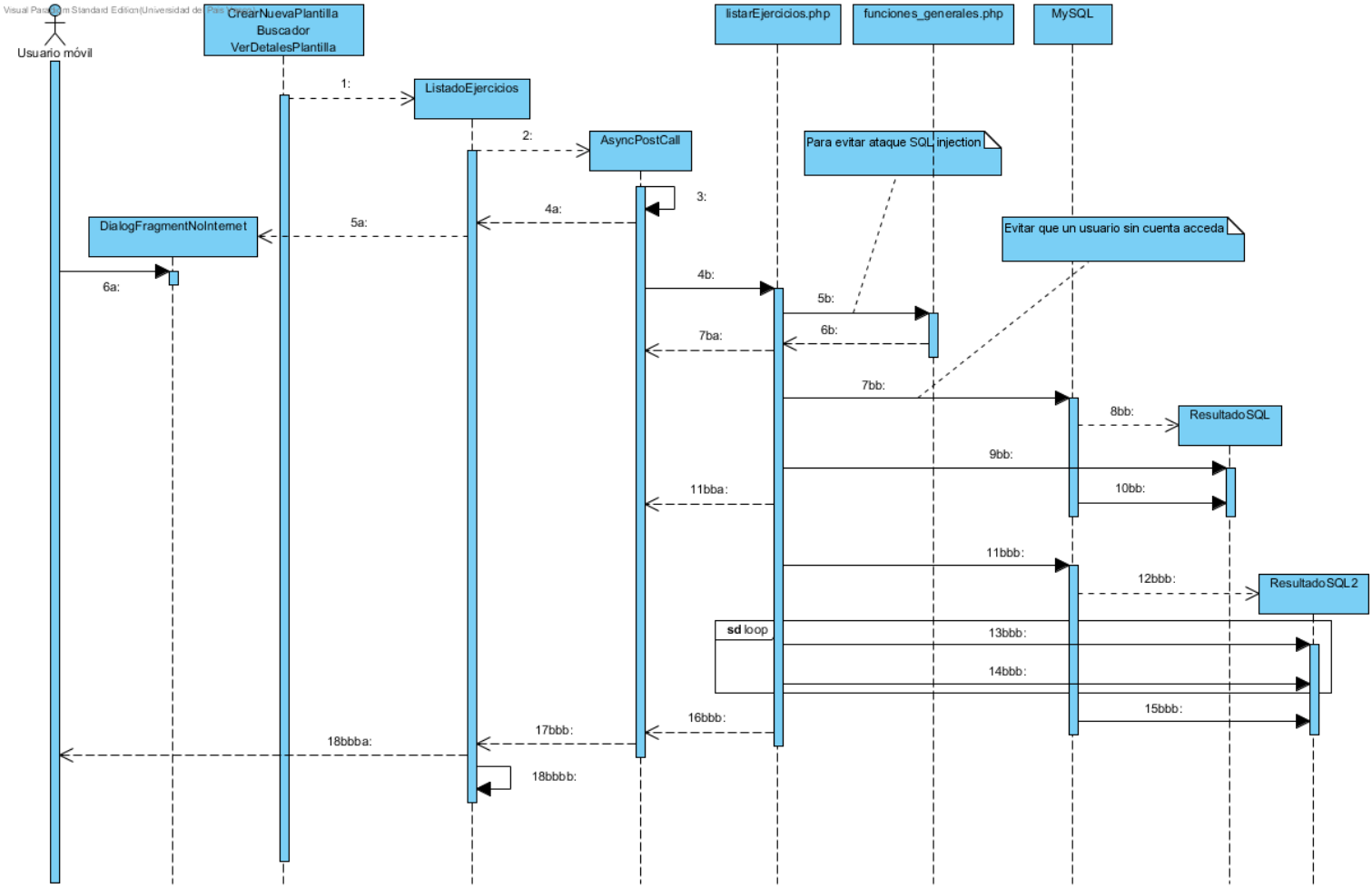


Ilustración 180: Diagrama de secuencia App: Pedir listado de ejercicios

1- Al iniciar la actividad de origen {CrearNuevaPlantilla, Buscador o VerDetallesPlantilla} y pulsar sobre el botón (añadir o buscar) se generará una instancia de ListadoEjercicios.java.
2- new AsyncPostCall(int: reqId; String: nickUsuario).execute();
3- comprobarConexion();

[Si no tiene conexión a internet]

4a- return $response: JSON, (status=1, msg="Cod11", listaEjercicios = null);
5a- new(DialogFragmentNoInternet.java);
6a- el usuario pulsará aceptar.

[Si tiene conexión a internet]

4b- validarFormulario(String: nick);
5b- validarformulario(String: nick);
6b- return $validación;

[Si la validación del PHP detecta que el nick de Usuario no sigue el patrón]

7ba- return $response: JSON, (status=1, msg="Cod13", listaEjercicios = null);

[Si la validación del PHP encuentra que es un nick válido]

7bb- execSQL(Consulta1);

```
SELECT * FROM usuarios
WHERE nickUsuario= '$nick'
```

8bb- new();
9bb- mysqli_num_rows();
10bb- destroy();

[Si no encuentra usuarios con ese nick = no tiene permiso]

11bba- return $response: JSON, (status=1, msg="Cod13", listaEjercicios = null);

[Si el usuario se encuentra en el sistema]

11bbb- execSQL(Consulta2);

```
SELECT * FROM ejercicios
ORDER BY nombreEjer ASC
```

12bbb- new();

[Por cada tupla de ejercicios que tiene la DB]

13bbb- next();
14bbb- get(nombreEjer, grupoMuscular, imagen, descripción, nickUsuario): JSON;

[Fin loop]

15bbb- destroy();
16bbb- return $response: JSON, (status=0, msg="", listaEjercicios= Ejercicio[]);

17bbb- return JSONObject;

[Si el atributo status del objeto JSON= 1(error)]

18bbba- mostrar un mensaje Toast en base al código del error del objeto.

[Si el atributo status del objeto JSON= 0 (correcto)]

18bbbb- cargarListadoEjercicios(); (realmente se llama cargarListView pero de esta manera se diferencia en los diagramas de secuencia de los otros “cargarListView”)

Cargar listado de ejercicios:

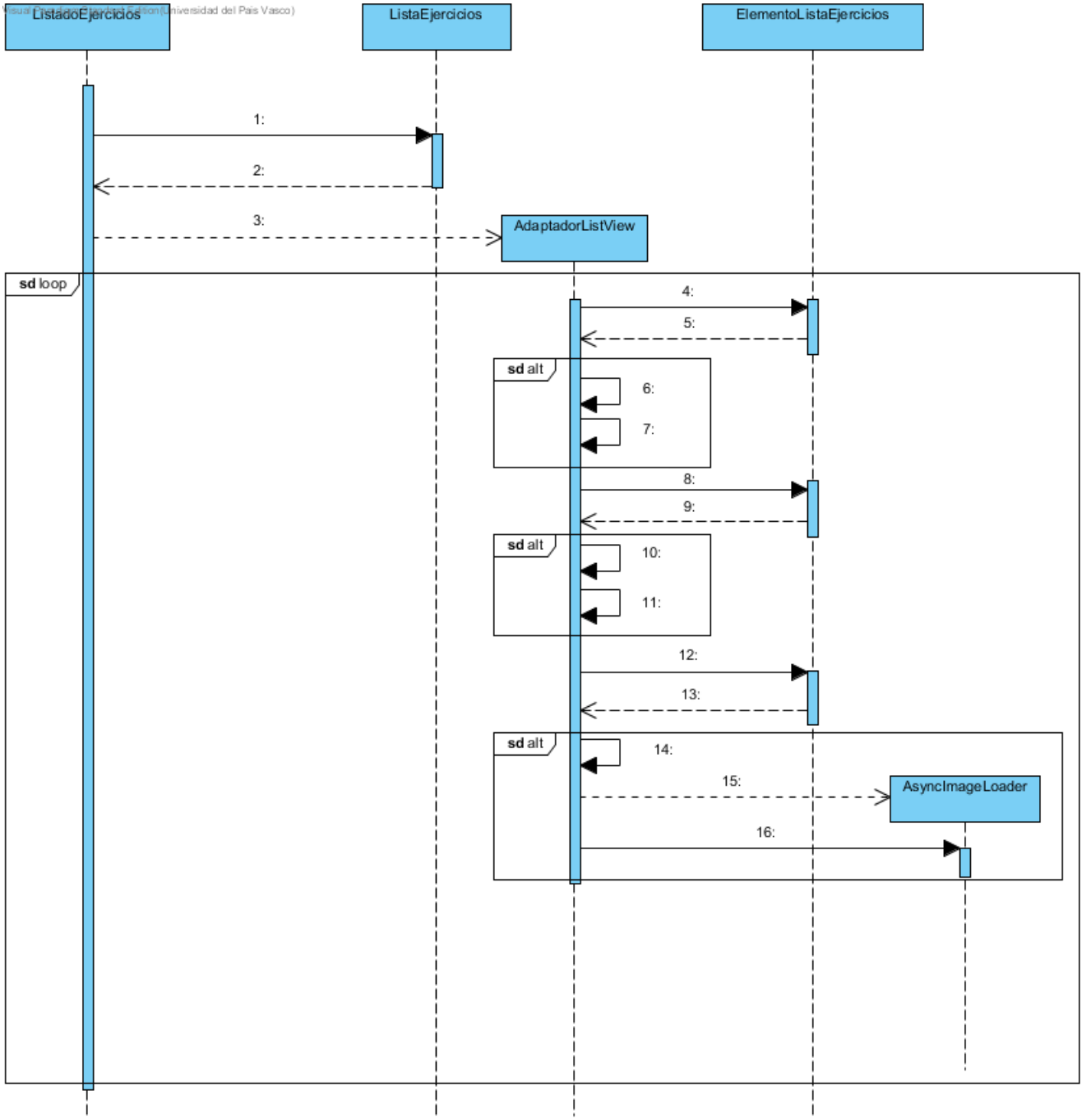


Ilustración 181: Diagrama de secuencia App: Cargar listado de ejercicios

1- transformarEnElementoListaEjercicios(ArrayList<Ejercicios>);
2- return ArrayList<ElementoListaEjercicios>;
3- new AdaptadorListView(contexto, idLayout, ArrayList<ElementoListaEjercicios>);

[Por cada elemento dentro del ArrayList pasado a la constructora]

4-findViewById(int: idTextViewNombre);
5- return tvNombre;
[Si tvNombre no es null]
6- elemento.getNombre(): String;
7- tvNombre.setText(nombreEjer);
[Fin Si]
8- findViewById(int: idTextViewGrupoMuscular);
9- return tvGrupoMuscular;
[Si tvGrupoMuscular no es null]
10- elemento.getGrupoMuscular(): String;
11- tvGrupoMuscular.setText(grupoMuscular);
[Fin Si]
12- findViewById(int: idImagen);
13- return ImageView;
[Si el ImageView no es null]
14- elemento.getImg(): String;
15- new AsyncImageLoader(UrlImagen);
16- execute();
[Fin Si]

[End loop]

Nota: Android ofrece una gran cantidad de consejos para evitar NullPointers, por ello existen diferentes estructuras “If” en el código para evitar asignar un valor a un elemento gráfico que por diferentes cuestiones (fragments visibles, ocultar botones, etc) es posible que no estén visibles, lo cual ocasionaría un error de ejecución.

Guardar plantilla:

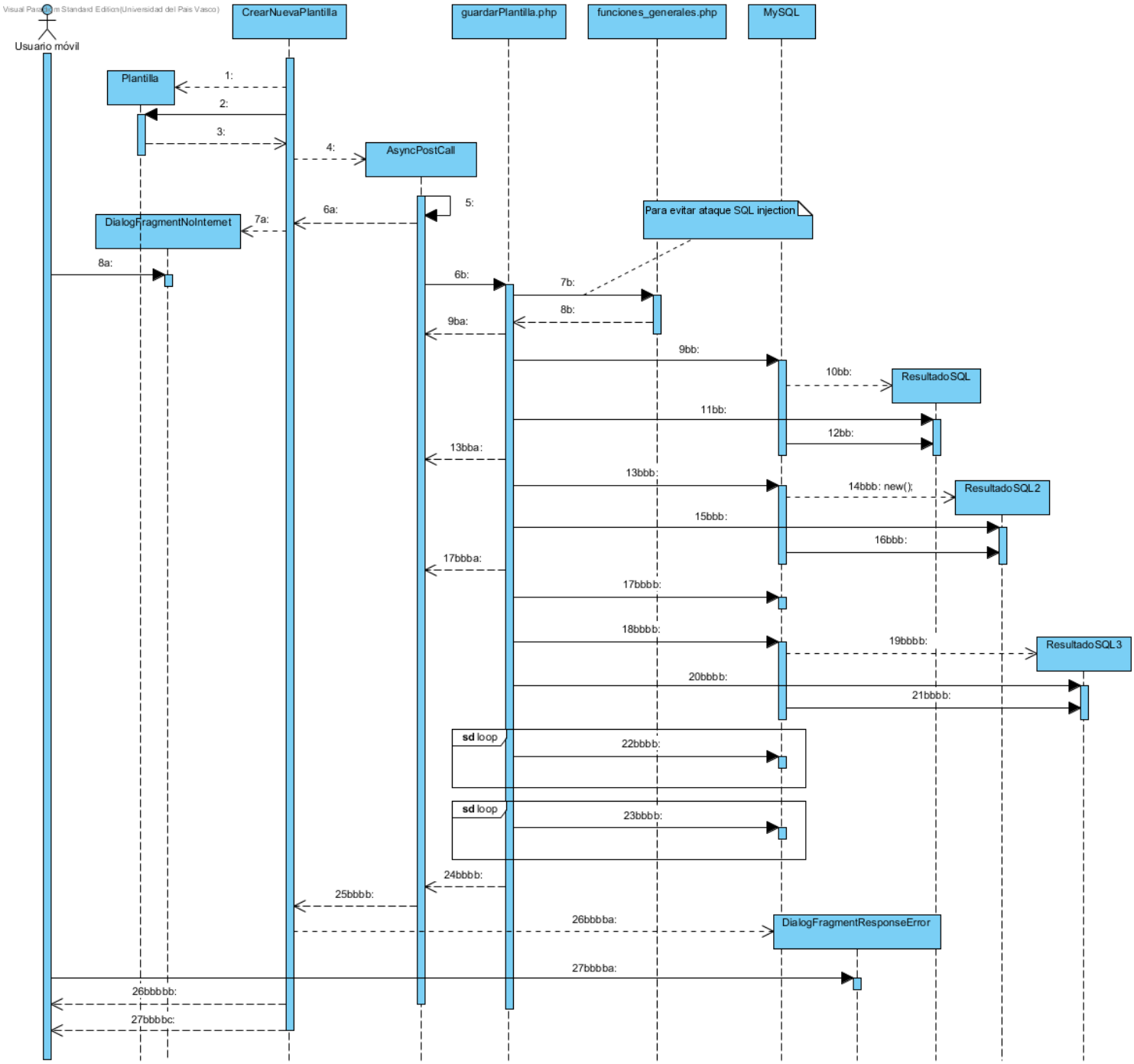


Ilustración 182: Diagrama de secuencia App: Guardar plantilla

1- new Plantilla(pNombre, listaTags, listaEjercicios).
2- aJSON();
3- return JSON;
4- new AsyncPostCall(int: reqId; String: pNick; JSON: plantilla).execute();
5- comprobarConexion();

[Si no encuentra conexión a internet]

6a- return $response: JSON, (status=1, msg="Cod11");
7a- new(DialogFragmentNoInternet.java);
8a- El usuario pulsará aceptar.

[Si tiene conexión a internet]

6b- validarFormulario(String: nick, pNombrePlantilla);
7b- validarFormulario(String: nick, pNombrePlantilla);
8b- return $validacion;

[Si la validación de PHP detecta caracteres incorrectos]

11ba- return $response: JSON, (status=1, msg="CodXX"), donde XX es el código del error {13,16,17};

[Si la validación decide que los nombres proporcionados son válidos]

9bb- execSQL(Consulta1);

```
SELECT * FROM usuarios
WHERE nickUsuario= '$nick'
```

10bb- new();
11bb- mysqli_num_rows();
12bb- destroy();

[Si el usuario no está registrado en el sistema]

13bba- return $response: JSON, (status=1, msg="Cod13");

[Si el usuario está registrado en el sistema]

13bbb- execSQL(Consulta2);

```
SELECT * FROM plantillas
WHERE nombre= '$nombrePlantilla' &&
nickUsuario= '$nick'
```

14bbb- new();
15bbb- mysqli_num_rows();
16bbb- destroy();

[Si el usuario ya tiene una plantilla con ese nombre]

17bbba- return $response: JSON, (status=1, msg="Cod14");

[Si el nombre está disponible]

17bbbb- execSQL(Consulta3);

```
INSERT INTO plantillas (nombre, nickUsuario)
VALUES ('$nombrePlantilla','$nick')
```

18bbbb- execSQL(Consulta4);

```
SELECT * FROM plantillas
WHERE nombre= '$nombrePlantilla' &&
nickUsuario= '$nick'
ORDER BY idPlantilla DESC
```

19bbbb- new();

20bbbb- get(idPlantilla: int

21bbbb- destroy();

[Por cada Tag que tenga la plantilla de entrenamiento]

22bbbb- execSQL(Consulta5);

```
INSERT INTO tagsplantilla (idPlantilla, grupoMuscular)
VALUES ('$idPlantilla','$tag')
```

[End loop]

[Por cada ejercicio que tenga la plantilla de entrenamiento]

23bbbb- execSQL(Consulta6);

```
INSERT INTO configuracionejercicio
(idPlantilla,posicion,nombreEjer,numSeries,numRepet
iciones,descanso)
VALUES ('$idPlantilla', '$orden', '$nombreEjer',
'$numSeries', '$numReps', '$descanso')
```

[End loop]

24bbbb- return $response: JSON, (status=0, msg="");

25bbbb- return JSONObject;

[Si el atributo status del JSON = 1 (error) y el atributo "msg" = "Cod13"]

26bbbba- new(DialogFragmentResponseError);

27bbbba- tras leer el motivo del error el usuario pulsará aceptar.

[Si el atributo status del JSON = 1 (error) y el atributo "msg" != "Cod13"]

26bbbbb- mostrar un mensaje Toast indicando que tipo de error que se ha producido (error leve).

[Si el atributo status del JSON = 0 (correcto)]

26bbbbc- mostrar un mensaje Toast al usuario indicando que se ha almacenado correctamente.

Crear nueva plantilla:

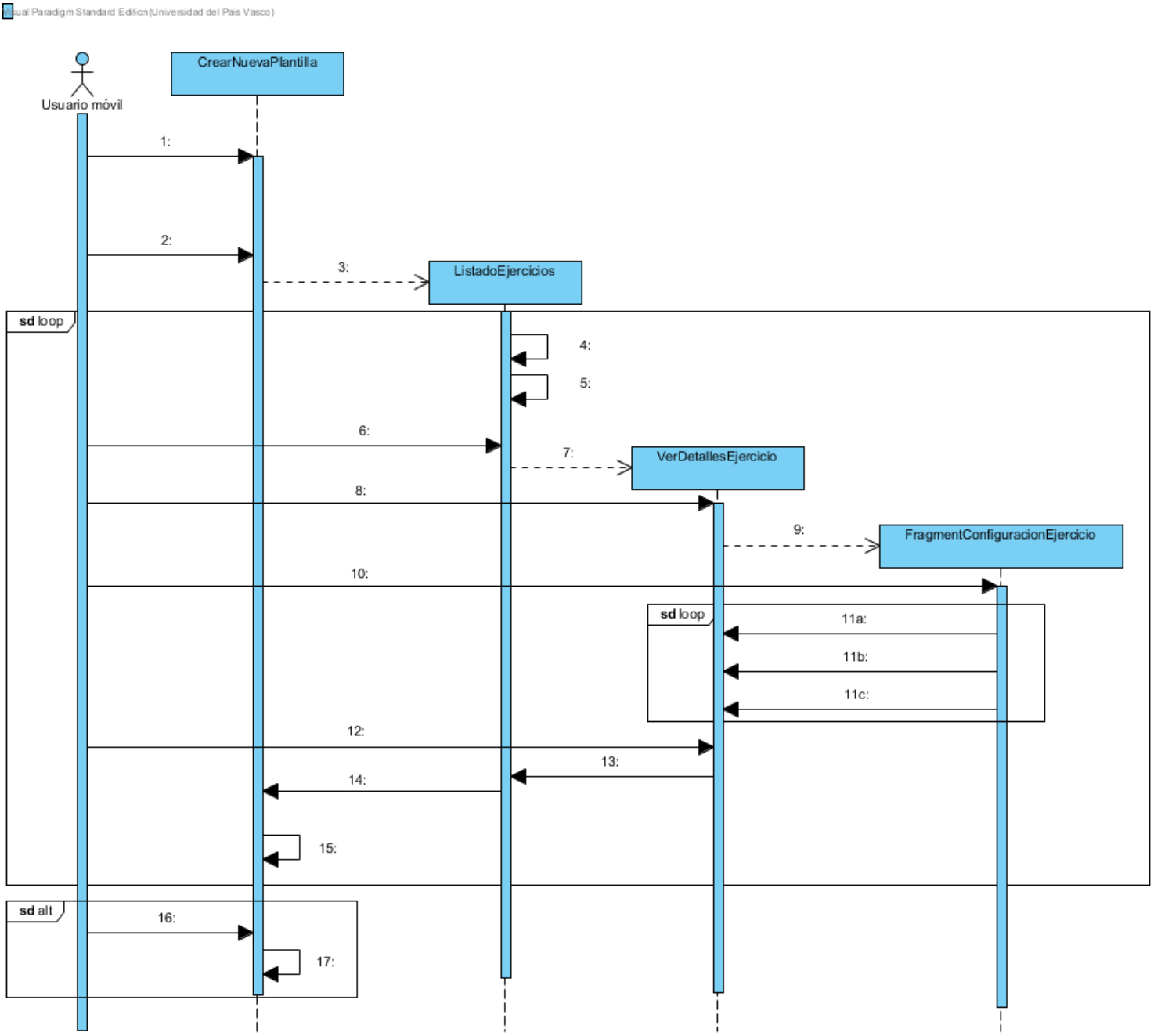


Ilustración 183: Diagrama de secuencia App: Crear nueva plantilla

1- El usuario introduce un nombre para la plantilla.

[Por cada ejercicio que el usuario quiera introducir en la plantilla de entrenamiento]

2- El usuario pulsará sobre el icono "+" para añadir ejercicios.

3- startActivityForResult(Intent: intent; int: idIntent);

4- pedirListadoEjercicios();

5- cargarListadoEjercicios();

6- el usuario pulsa sobre el ejercicio que desee añadir.

7- startActivityForResult(Intent: intent, idIntent; byte[]:ejercicioSerializado);

8- el usuario pulsará sobre el icono de la llave para habilitar la configuración.

9- new FragmentConfiguracionEjercicio();

10- el usuario modifica los valores de los NumberPickers según su gusto.

[Por cada actualización de cada NumberPicker se activa la interfaz correspondiente]

[Si actualiza el número de series]

11a- onSeriesUpdate(int: pNumSeries);

[Si actualiza el número de repeticiones]

11b- onRepsUpdate(int: pNumReps);

[Si actualiza el tiempo de descanso]

11c- onBreakUpdate(int: pBreak)

[Fin loop configuración mediante NumberPickers]

12- el usuario pulsará sobre el icono en forma de "V" de la ActionBar para guadar los cambios.

13- onActivityResult(int: requestCode; Intent: idIntent);

14- onActivityResult(int: requestCode; Intent: idIntent);

15- permitirGuardado();

[Fin loop añadir ejercicios]

[Si la función permitirGuardado ha hecho visible el botón de guardado]

16- el usuario pulsará sobre el icono en forma de "V" de la ActionBar para guardar la plantilla.

17- guardarPlantilla();

[Fin Si]

Nota: la función *permitirGuardado()* verifica que el listado contiene al menos un ejercicio y el usuario ha proporcionado nombre a la plantilla de entrenamiento. Si ambas condiciones se cumplen entonces el botón de guardado aparece.

El TextView que contiene el nombre de la plantilla tiene un listener que captura la el texto a introducir, con lo cual se procede a comprobar el estado de la plantilla cuando se proporciona un nombre y cuando se añaden ejercicios.

De la misma manera, cuando se eliminan los ejercicios de la plantilla en creación o cuando se vacía el TextView del nombre la misma función *permitirGuardado()* oculta el botón de guardado para evitar errores.

Obtener Mis Plantillas:

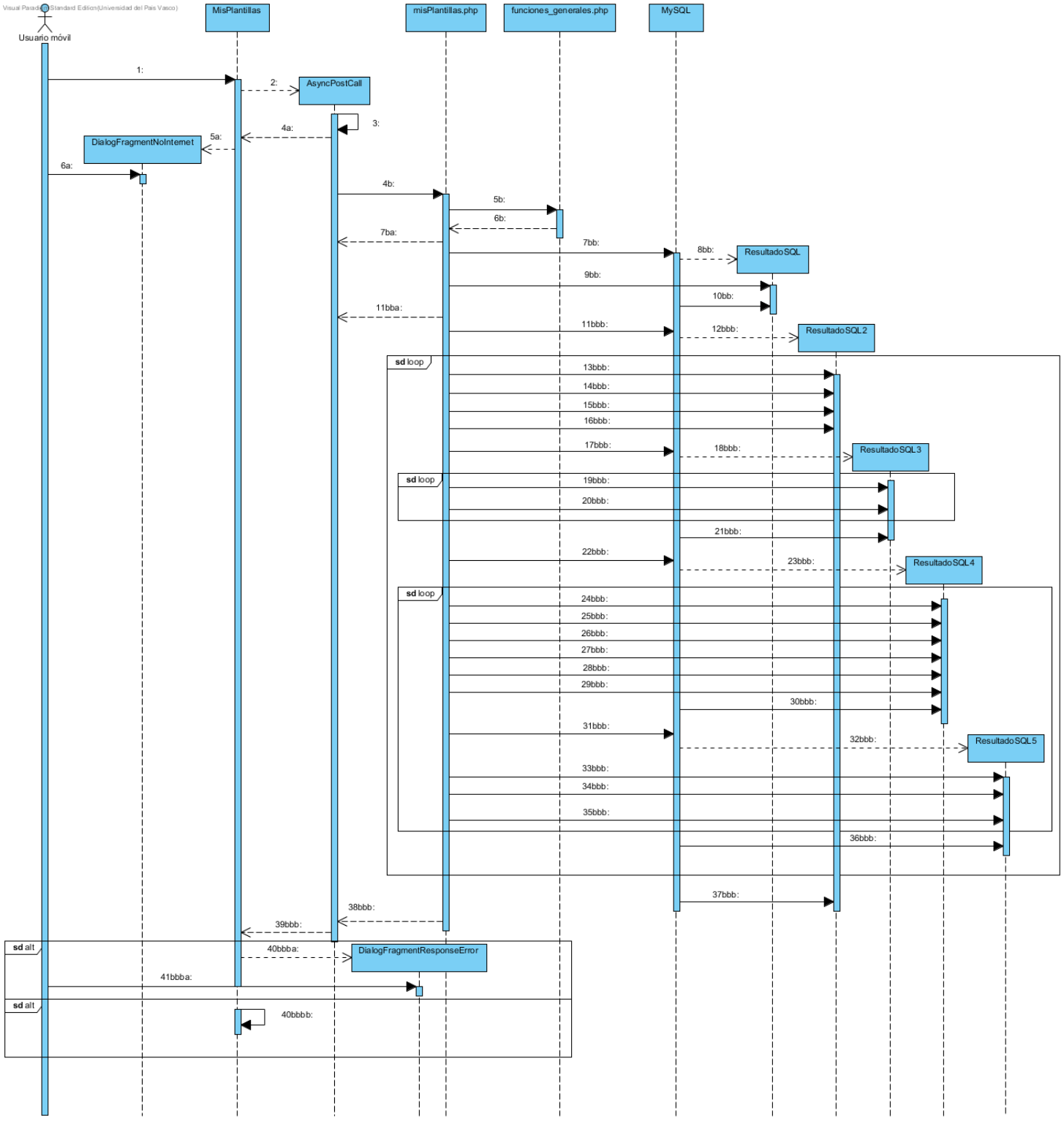


Ilustración 184: Diagrama de secuencia App: Obtener Mis Plantillas

1- Desde la actividad “LogedPrincipal” el usuario pulsa sobre Mis Plantillas y se inicia la actividad.
2- new AsyncPostCall(int: reqId, nickUsuario).execute();
3- comprobarConexion();

[Si no encuentra conexión a internet]

4a- return $response: JSON, (status=1, msg=“Cod11”, listaPlantillas=””);
5a- new(DialogFragmentNoInternet.java);
6a- El usuario pulsará aceptar.

[Si tiene conexión a internet]

4b- validarFormulario(String: nick);
5b- validarFormulario(String: nick);
6b- return $validacion;

[Si la validación de PHP detecta caracteres incorrectos en el nick]

7ba- return $response: JSON, (status=1, msg=“Cod13”, listaPlantillas=””);

[Si la validación decide que el nick es válido]

7bb- execSQL(Consulta1);

```
SELECT * FROM usuarios
WHERE nickUsuario= '$nick'
```

8bb- new();
9bb- mysqli_num_rows();
10bb- destroy();

[Si no se encuentra el usuario en el sistema]

11bba- return $response: JSON, (status=1, msg=“Cod13”, listaPlantillas=””);

[Si el usuario existe en el sistema]

11bbb- execSQL(Consulta2);

```
SELECT * FROM plantillas
WHERE nickUsuario= '$nick'
ORDER BY nombre ASC
```

12bbb- new();

[Por cada plantilla del usuario que devuelva la consulta 2]

13bbb- next();
14bbb- getIdPlantilla(): int;
15bbb- getNombrePlantilla(): String;
16bbb- getPropietario(): String;
17bbb- execSQL(Consulta3);

```
SELECT * FROM tagsplantilla
WHERE idPlantilla= '$idPlantilla'
```

18bbb- new();

[Por cada Tag que tiene la plantilla actual]

19bbb- next();

20bbb- getTag(): String;

[Fin loop obtener tags]

21bbb- destroy();

22bbb- execSQL(Consulta4);

```
SELECT * FROM configuracionejercicio
WHERE idPlantilla= '$idPlantilla'
ORDER BY posicion ASC
```

23bbb- new();

[Por cada ejercicio que tiene la plantilla]

24bbb- next();

25bbb- getNombre(): String;

26bbb- getNumSeries(): int;

27bbb- getRepeticiones(): int;

28bbb- getDescanso(): int;

29bbb- getOrden(): int;

30bbb- destroy();

31bbb- execSQL(Consulta5);

```
SELECT * FROM ejercicios
WHERE nombreEjer= '$nomEj'
```

32bbb- new();

33bbb- getGrupoMuscular(): String;

34bbb- getImagen(): String;

35bbb- getDescripcion(): String;

[Fin loop ejercicios]

36bbb- destroy();

[Fin loop plantilla]

37bbb- destroy();

38bbb- return $response: JSON; (status=0, result = JSON, listaPlantillas= Plantilla[]);

39bbb- return JSONObject();

[Si el atributo status del JSON = 1 (error)]

40bbba- new();

41bbba- el usuario pulsará aceptar;

[Si el atributo status del JSON = 0 (correcto)]

40bbbb- cargarListView();

Cargar listView (Mis Plantillas):

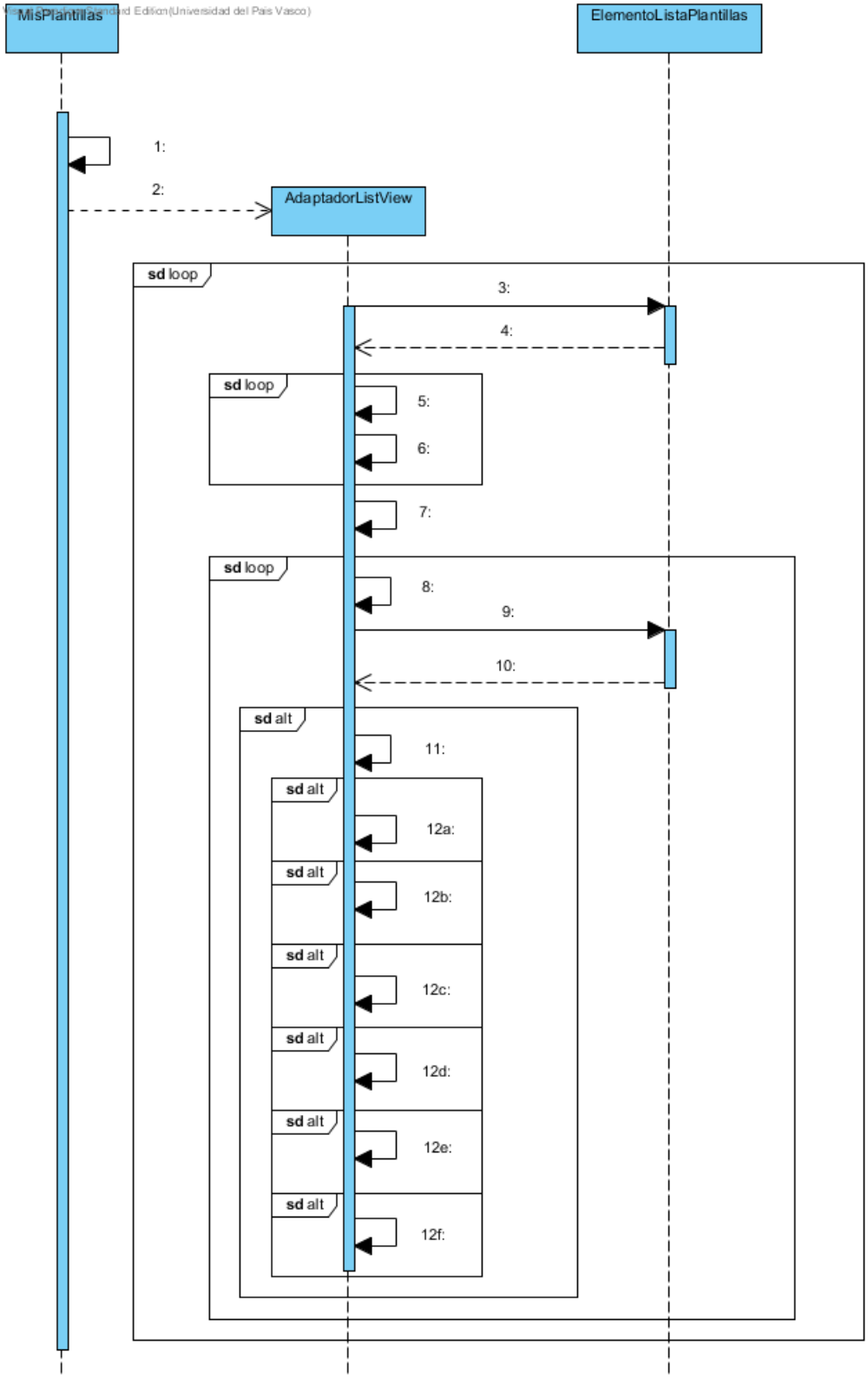


Ilustración 185: Diagrama de secuencia App: Cargar ListView (Mis Plantillas)

1- transformarEnElementoListaPlantillas(): ArrayList<ElementoListaPlantillas>;
2- new AdaptadorListView(contexto, layout, ArrayList<ElementoListaPlantillas>;
[Por cada elemento plantilla dentro del ArrayList pasado a la constructora]
3- findViewById(int: idTextViewNombre);
4- return tvNombre;
[Si el textView del nombre no es null]
5- elemento.getNombre(): String;
6- tvNombre.setText(nombrePlantilla);
[Fin si]
7- elemento.getmTags(): ArrayList<String>;
[Por cada Tag que tiene la plantilla actual]
8- obtenerIdTag(String: tagActual): int;
9- findViewById(int: idImageView);
10- return imgTag;
[Si el imageView del Tag actual no es null]
11- imgTag.setVisibility(View.Visible);
[Switch]
[Si tagActual == "Abdomen"]
12a- imgTag.setImageResource(int: idImagenAbdomen);
[Si tagActual == "Brazo"]
12b- imgTag.setImageResource(int: idImagenBrazo);
[Si tagActual == "Espalda"]
12c- imgTag.setImageResource(int: idImagenEspalda);
[Si tagActual == "Hombro"]
12d- imgTag.setImageResource(int: idImagenHombro);
[Si tagActual == "Pecho"]
12e- imgTag.setImageResource(int: idImagenPecho);
[Si tagActual == "Pierna"]
12f- imgTag.setImageResource(int: idImagenPierna);
[End switch]
[Fin Si]
[End loop Tags de plantilla]
[End loop de plantillas]

Actualizar plantilla:

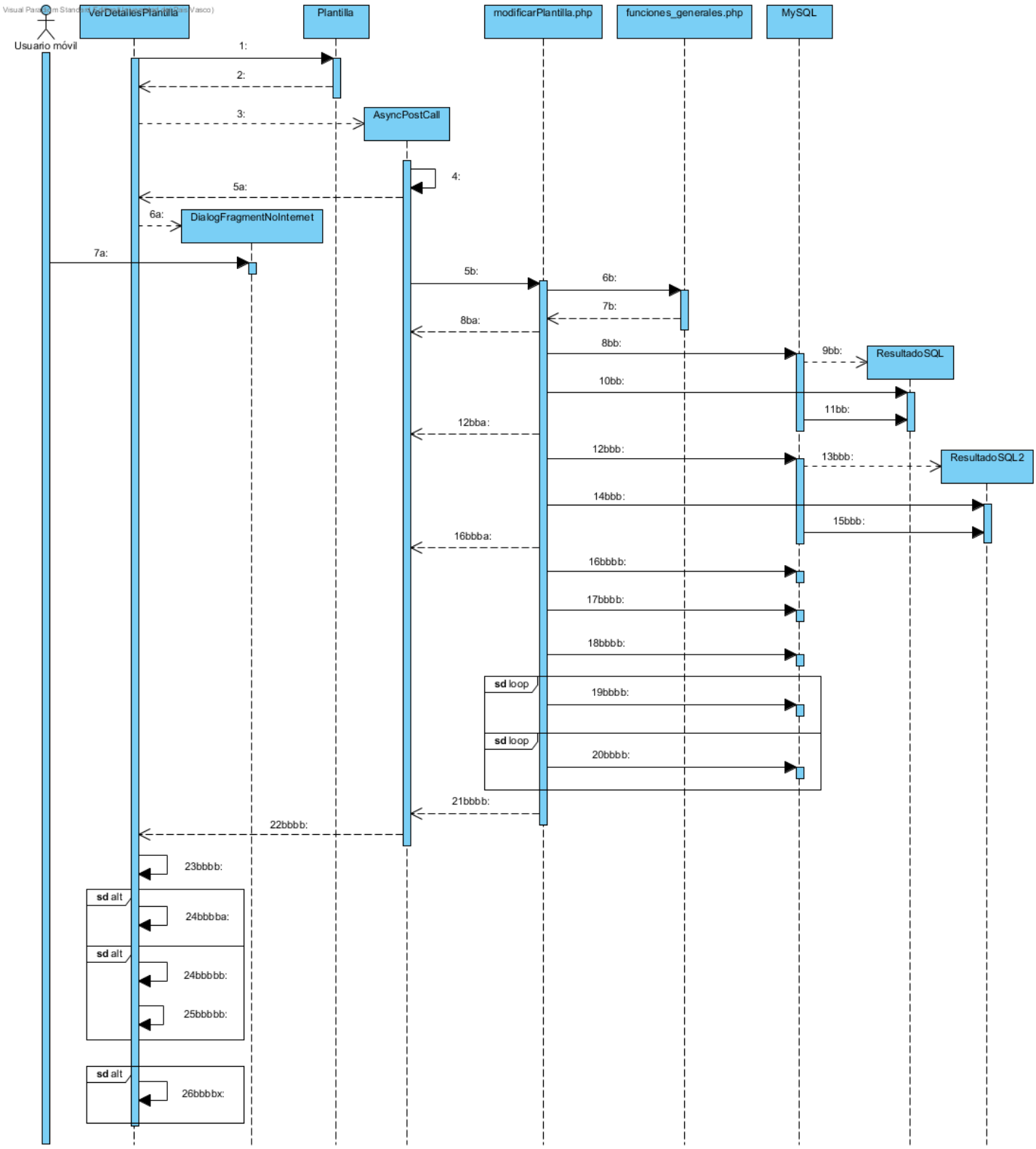


Ilustración 186: Diagrama de secuencia App: Actualizar plantilla

1- aJSON();
2- return JSON();
3- new AsyncPostCall(int: reqId; String: nick; JSON: pJSON);
4- comprobarConexion();

[Si el dispositivo no tiene conexión a internet]

5a- return $response: JSON, (status=1, msg="Cod11");
6a- new (DialogFragmentNoInternet.java);
7a- el usuario pulsará aceptar tras leer la advertencia.

[Si tiene conexión a internet]

5b- validarFormulario(String: nick, nombrePlantilla);
6b- validarFormulario(String: nick, nombrePlantilla);
7b- return $validacion;

[Si la validación del PHP detecta algún error en los nombres]

8ba- return $response: JSON, (status=1,msg="CodXX"), donde XX = código que generó el error {13,16 o 17};

[Si la validación del PHP considera correctos ambos parámetros]

8bb- execSQL(Consulta1);

```
SELECT * FROM plantillas
WHERE nickUsuario= '$nick' &&
idPlantilla='$idPlantilla'
```

9bb- new();
10bb- mysqli_num_rows();
11bb- destroy();

[Si no se corresponde el usuario con el ID de la plantilla]

12bba- return $response: JSON, (status=1, msg="Cod19");

[Si el usuario es el propietario de la plantilla]

12bbb- execSQL(Consulta2);

```
SELECT * FROM plantillas
WHERE nombre= '$nombrePlantilla' &&
nickUsuario= '$nick' && idPlantilla <> '$idPlantilla'
```

13bbb- new();
14bbb- mysqli_num_rows();
15bbb- destroy();

[Si el nombre de la plantilla está en uso por otra plantilla del usuario]

16bbba- return $response: JSON, (status=1, msg="Cod14");

[Si el nombre nuevo de la plantilla está libre]

16bbbb- execSQL(Consulta3);

```
DELETE FROM tagsplantilla
WHERE idPlantilla = '$idPlantilla'
```

17bbbb- execSQL(Consulta4);

```
DELETE FROM configuracionejercicio
WHERE idPlantilla = '$idPlantilla'
```

18bbbb- execSQL(Consulta5);

```
UPDATE plantillas
SET nombre='$nombrePlantilla'
WHERE idPlantilla='$idPlantilla'
```

[Por cada tag de la plantilla]

19bbbb- execSQL(Consulta6);

```
INSERT INTO tagsplantilla (idPlantilla, grupoMuscular)
VALUES ('$idPlantilla','$tag')
```

[End loop]

[Por cada ejercicio de la plantilla]

20bbbb- execSQL(Consulta7);

```
INSERT INTO configuracionejercicio (idPlantilla,posicion,
nombreEjer, numSeries, numRepeticiones, descanso)
VALUES ('$idPlantilla', '$orden', '$nombreEjer',
'$numSeries', '$numReps', '$descanso')
```

[End loop]

21bbbb- return $response: JSON, (status=0, msg="");

22bbbb- return JSONObject();

23bbbb- boolean recuperarPlantilla = true;

[Si el atributo status del JSON = 1 (error)]

24bbbba- mostrar mensaje Toast con el fallo en función del atributo msg.

[Si el atributo status del JSON = 0 (correcto)]

24bbbb- recuperarPlantilla=false;

25bbbb- mostrar mensaje Toast informando del éxito,

[Si la variable local recuperarPlantilla = true]

26bbbbx- mPlantilla = mPlantillaAux; (de esta forma recuperamos el estado inicial de la plantilla)

[Fin si]

Añadir ejercicio:

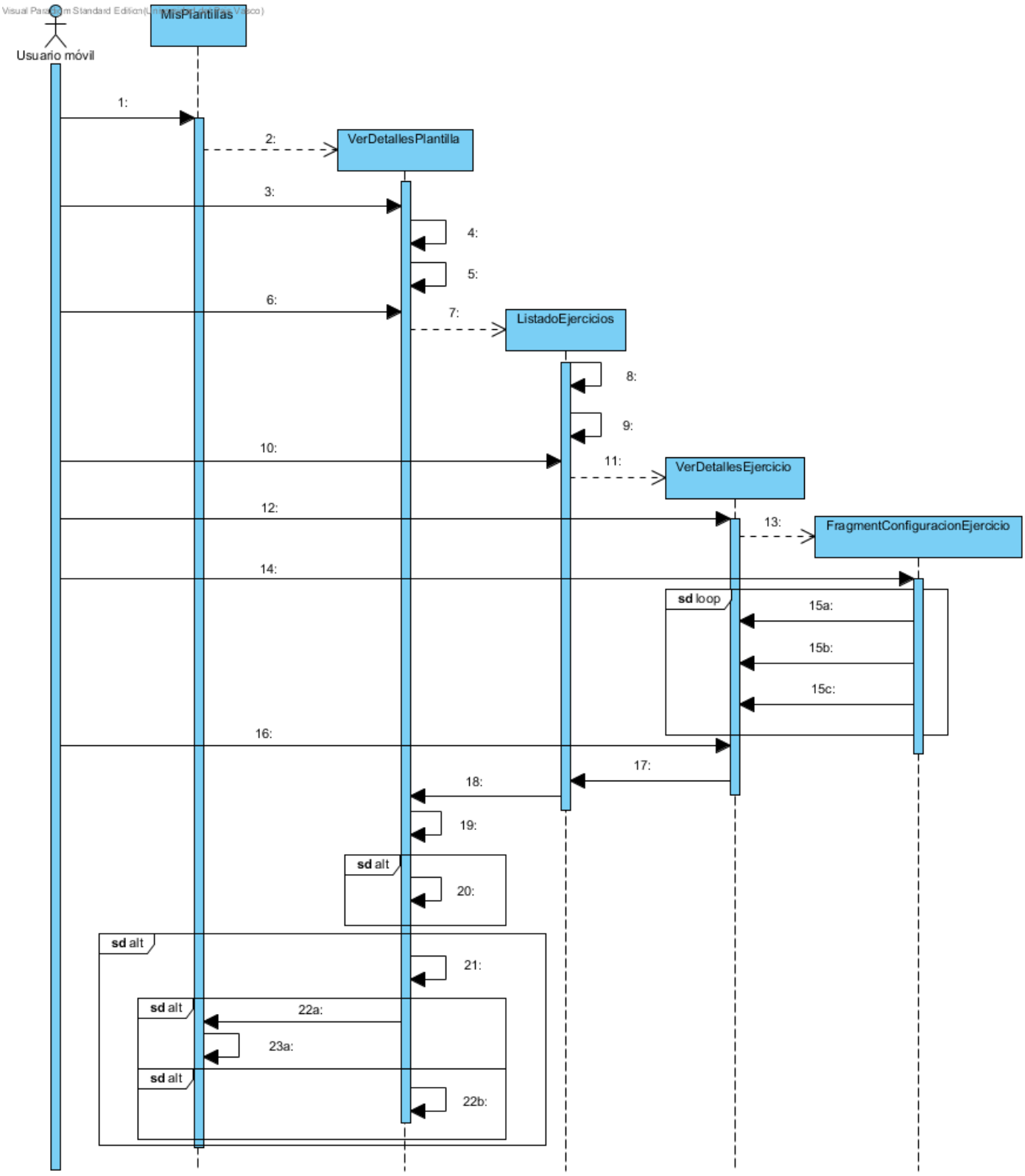


Ilustración 187: Diagrama de secuencia App: Añadir ejercicio

1- El usuario pulsa sobre una de las plantillas de “MisPlantillas”.
2- startActivityForResult(Intent: intent; int: idIntent; byte[]: plantillaSeleccionada);
3- El usuario pulsa sobre el icono de la llave para habilitar la modificación de plantillas.
4- onOptionsItemSelected(MenuItem: ítem): boolean
5- alternarConfiguracion();
6- El usuario pulsa sobre el botón “+” para añadir ejercicios.
7- startActivityForResult(Intent: intent; int: idIntent);
8- pedirListadoEjercicios();
9- cargarListadoEjercicios();
10- el usuario pulsa sobre el ejercicio que desee añadir.
11- startActivityForResult(Intent: intent; int: idIntent; byte[]: ejercicioSeleccionado);
12- el usuario pulsa sobre el icono de la llave para configurar el ejercicio a añadir.
13- new FragmentConfiguracionEjercicio();
14- El usuario modifica los valores de los numberPickers según su gusto.

[Por cada actualización de cada NumberPicker se activa la interfaz correspondiente]

[Si actualiza el número de series]

15a- onSeriesUpdate(int: pNumSeries);

[Si actualiza el número de repeticiones]

15b- onRepsUpdate(int: pNumReps);

[Si actualiza el tiempo de descanso]

15c- onBreakUpdate(int: pBreak);

[Fin loop configuración mediante NumberPickers]

16- El usuario pulsará sobre el icono en forma de “V” de la ActionBar para guardar los cambios.
17- onActivityResult(int: requestCode; Intent: idIntent);
18- onActivityResult(int: requestCode; Intent: idIntent);
19- actualizarPlantilla();

[Si la actualización de datos en el servidor ha sido satisfactorio]

20- atributo “modificada” = true;

[Fin Si]

[Si el usuario pulsa el botón “back” del dispositivo móvil]

21- onBackPressed();

[Si la plantilla había sido modificada/actualizada correctamente]

22a- onActivityResult(int: requestCode; Intent: idIntent);
23a- cargarPlantillas();

[Si la plantilla no había sido actualizada]

22b- finish();

[Fin si]

Eliminar ejercicio:

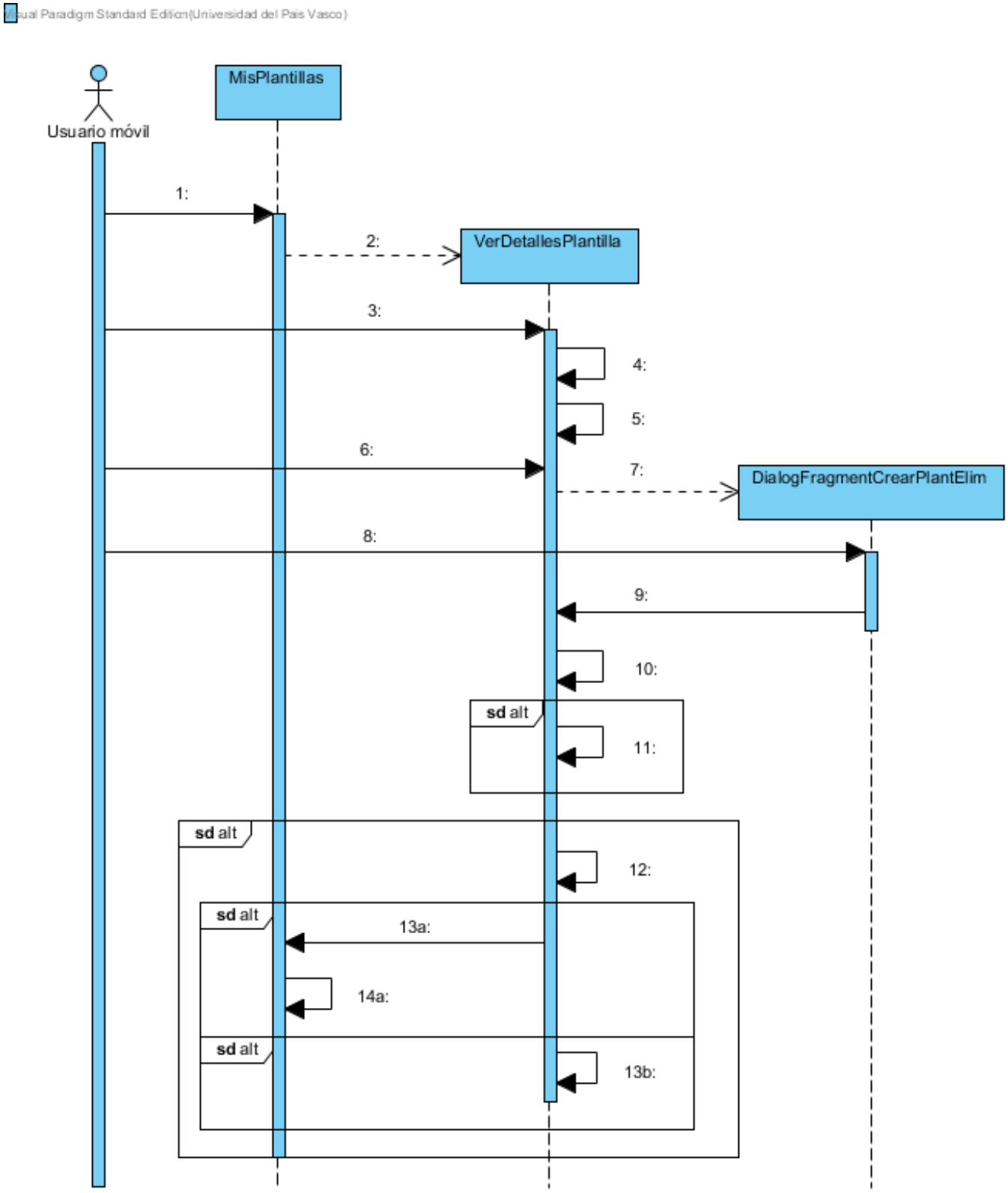


Ilustración 188: Diagrama de secuencia App: Eliminar ejercicio

1- El usuario pulsa sobre una de las plantillas de “MisPlantillas”.
2- startActivityForResult(Intent: intent; int: idIntent; byte[]: plantillaSeleccionada);
3- El usuario pulsa sobre el icono de la llave para habilitar la modificación de plantillas.
4- onOptionsItemSelected(MenuItem: ítem): boolean
5- alternarConfiguracion();
6- El usuario pulsa y mantiene pulsado sobre uno de los ejercicios de la plantilla.
7- new DialogFragmentCrearPlantElim(byte[]: ejercicioSeleccionado);
8- el usuario pulsa “Aceptar”.
9- onYesSelected(Ejercicio: seleccionado);
10- actualizarPlantilla();

[Si la actualización de datos en el servidor ha sido satisfactorio]

11- atributo “modificada” = true;

[Fin Si]

[Si el usuario pulsa el botón “back” del dispositivo móvil]

12- onBackPressed();

[Si la plantilla había sido modificada/actualizada correctamente]

13a- onActivityResult(int: requestCode; Intent: idIntent);

14a- cargarPlantillas();

[Si la plantilla no había sido actualizada]

13b- finish();

[Fin si]

Añadir/eliminar plantilla de favoritos:

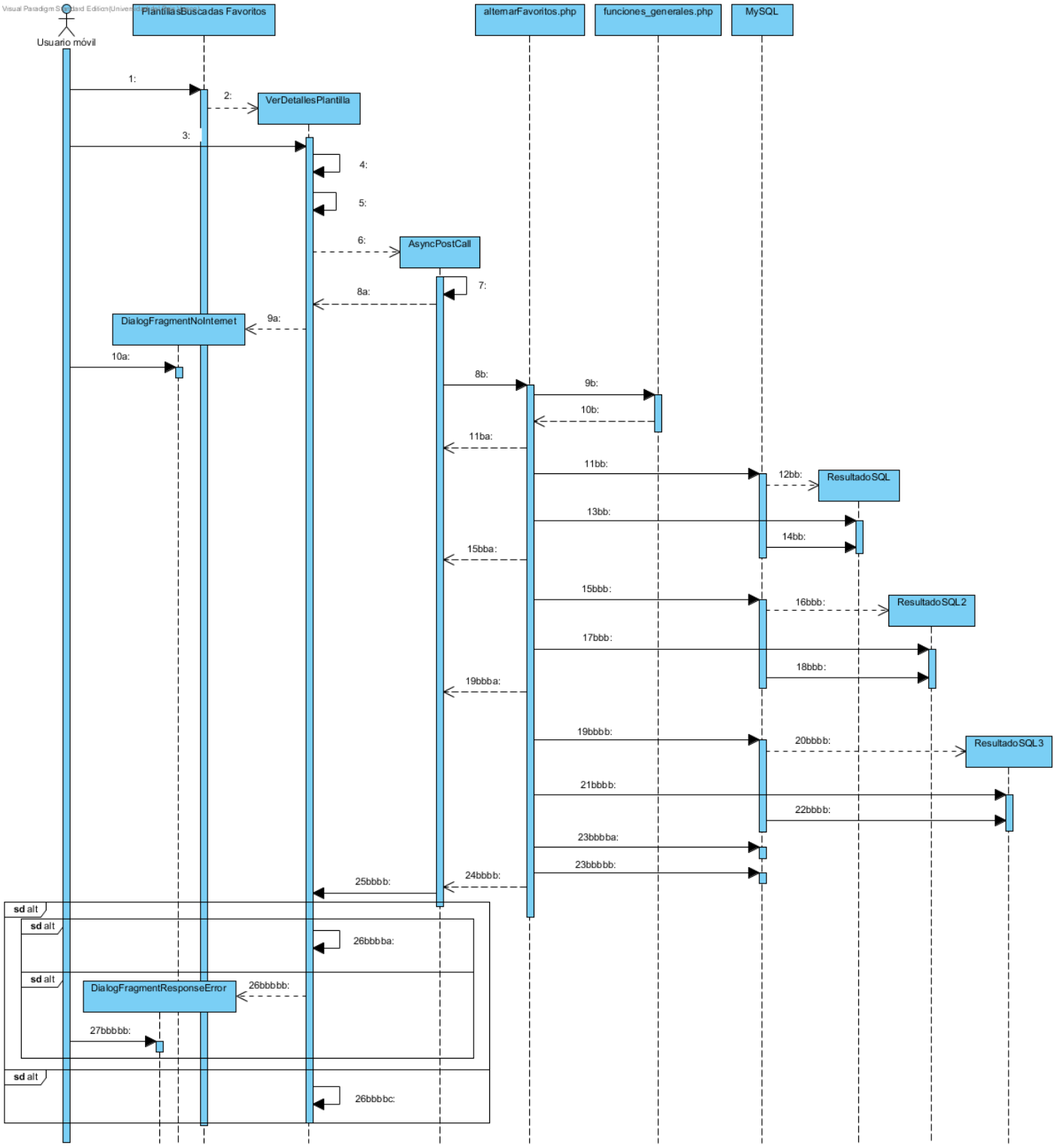


Ilustración 189: Diagrama de secuencia App: Añadir/Eliminar plantilla de favoritos

1- El usuario pulsa sobre una de las plantillas de “MisPlantillas”.
2- startActivityForResult(Intent: intent; int: idIntent; byte[]: plantillaSeleccionada);
3- El usuario pulsa sobre el icono de la estrella para añadir o eliminar de favoritos.
4- onOptionsItemSelected(MenuItem: ítem): boolean
5- alternarConfiguracion();
6- new AsyncPostCall(int: reqId, idPlantilla; String: nick);
7- comprobarConexion();

[Si el dispositivo no tiene conexión a internet]

8a- return $response: JSON, (status=1, msg=”Cod13”);
9a- new(DialogFragmentNoInternet.java);
10a- el usuario pulsa “Aceptar”.

[Si el dispositivo tiene conexión]

8b- validarFormulario(String: nick);
9b- validarFormulario(String: nick);
10b- return $validacion;

[Si la validación decide que la petición procede de un nick incorrecto]

11ba- return $response: JSON, (status=1, msg=”Cod13”);

[Si la validación es exitosa]

11bb- execSQL(Consulta1);

```
SELECT * FROM usuarios
WHERE nickUsuario= '$nick'
```

12bb- new();
13bb- mysqli_num_rows();
14bb- destroy();

[Si no existe ningún usuario en el sistema con ese nick]

15bba- return $response: JSON, (status=1, msg=”Cod13”);

[Si el usuario existe en el sistema]

15bbb- execSQL(Consulta2);

```
SELECT * FROM plantillas
WHERE idPlantilla='$idPlantilla'
```

16bbb- new();
17bbb- mysqli_num_rows();
18bbb- destroy();

[Si no existe ninguna plantilla con el ID especificado]

19bbba- return $JSON, (status=1, msg=”Cod23”);

[Si la plantilla se ha encontrado]

19bbbb- execSQL(Consulta3);

```
SELECT * FROM favoritos
WHERE idPlantilla='$idPlantilla' && nickUsuario='$nick'
```

20bbbb- new();
21bbbb- mysqli_num_rows();
22bbbb- destroy();

[Si la consulta devuelve líneas (se encuentra en favoritos)]

23bbbba- execSQL(Consulta4);

```
DELETE FROM favoritos
WHERE nickUsuario='$nick' &&
idPlantilla='$idPlantilla'
```

[Si la consulta está vacía (no está en favoritos)]

23bbbbb- execSQL(Consulta5);

```
INSERT INTO favoritos (nickUsuario,idPlantilla)
VALUES ('$nick','$idPlantilla')
```

24bbbb- return $response: JSON, (status=0, msg="");
25bbbb- return JSONObject();

[Si el atributo status del JSON = 1 (errores)]

[Si el atributo msg del JSON != 23 (error leve)]

26bbbba- mostrar mensaje Toast con el motivo del error.

[Si el atributo msg del JSON = 23 (error grave)]

26bbbbb- new(DialogFragmentResponseError);
27bbbbb- el usuario pulsará "Aceptar".

[Si el atributo status del JSON = 0 (correcto)]

26bbbbc- alternarFavoritos();

Cargar listView (Mis Favoritos):

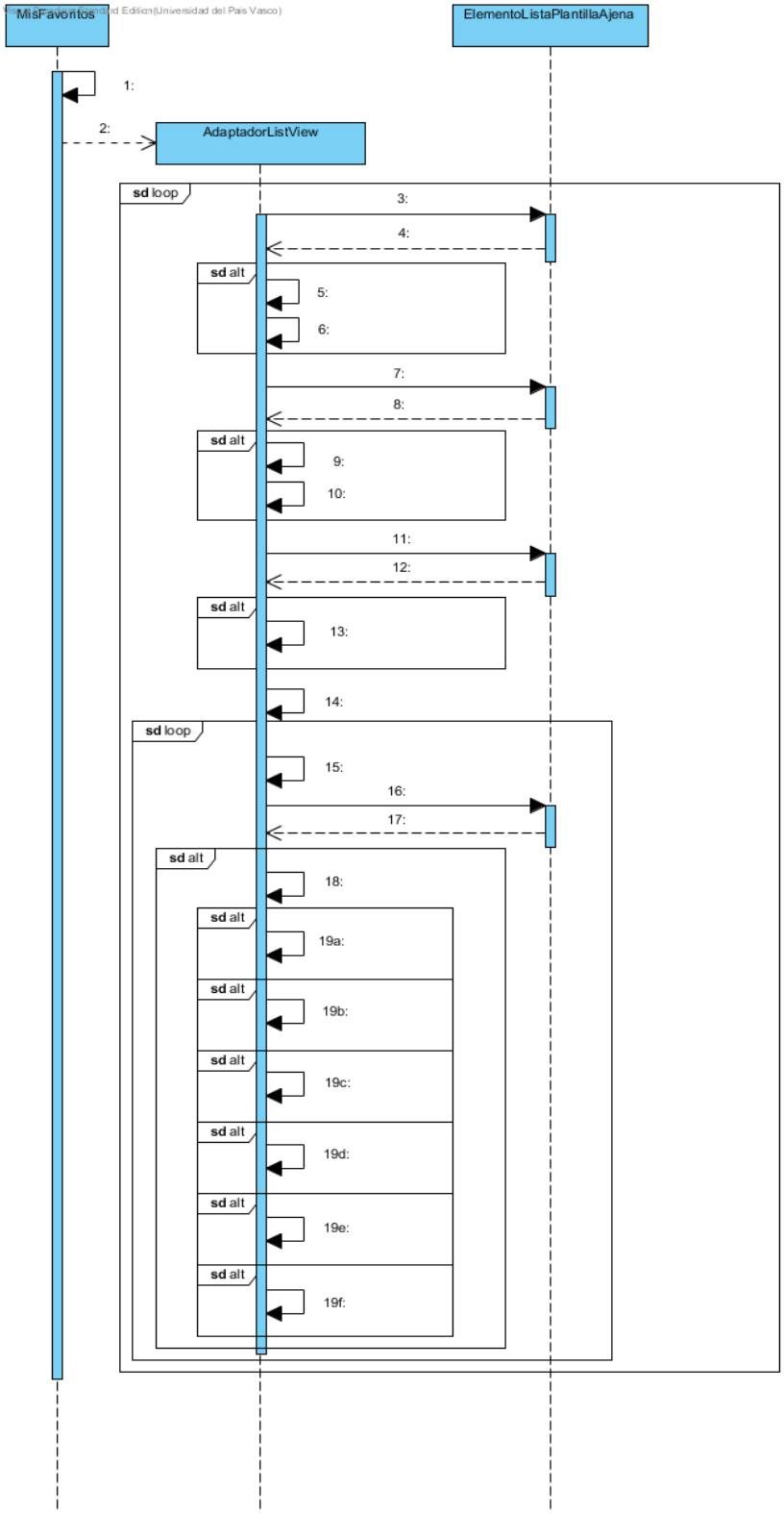


Ilustración 190: Diagrama de secuencia App: Cargar ListView (Favoritos)

1- transformarEnElementoListaPlantillas(): ArrayList<ElementoListaPlantillaAjena>;
2- new AdaptadorListView(contexto, layout, ArrayList<ElementoListaPlantillaAjena>;
[Por cada elemento plantilla dentro del ArrayList pasado a la constructora]
3- findViewById(int: idTextViewNombre);
4- return tvNombre;
[Si el textView del nombre no es null]
5- elemento.getNombre(): String;
6- tvNombre.setText(nombrePlantilla);
[Fin si]
7- findViewById(int: idTextViewPropietario);
8- return tvPropietario;
[Si el textView del propietario no es null]
9- elemento.getPropietario(): String;
10- tvPropietario.setText(nombrePropietario);
[Fin si]
11- findViewById(int: idImgEstrella);
12- return imgEstrella;
[Si el ImageView de la estrella no es null]
13- imgEstrella.setVisibility(View.Visible);
[Fin si]
14- elemento.getmTags(): ArrayList<String>;
[Por cada Tag que tiene la plantilla actual]
15- obtenerIdTag(String: tagActual): int;
16- findViewById(int: idImageView);
17- return imgTag;
[Si el imageView del Tag actual no es null]
18- imgTag.setVisibility(View.Visible);
[Switch]
[Si tagActual == "Abdomen"]
19a- imgTag.setImageResource(int: idImagenAbdomen);
[Si tagActual == "Brazo"]
19b- imgTag.setImageResource(int: idImagenBrazo);
[Si tagActual == "Espalda"]
19c- imgTag.setImageResource(int: idImagenEspalda);
[Si tagActual == "Hombro"]
19d- imgTag.setImageResource(int: idImagenHombro);
[Si tagActual == "Pecho"]
19e- imgTag.setImageResource(int: idImagenPecho);
[Si tagActual == "Pierna"]
19f- imgTag.setImageResource(int: idImagenPierna);
[End switch]
[Fin Si]
[End loop Tags de plantilla]
[End loop de plantillas]

Obtener Mis Favoritos:

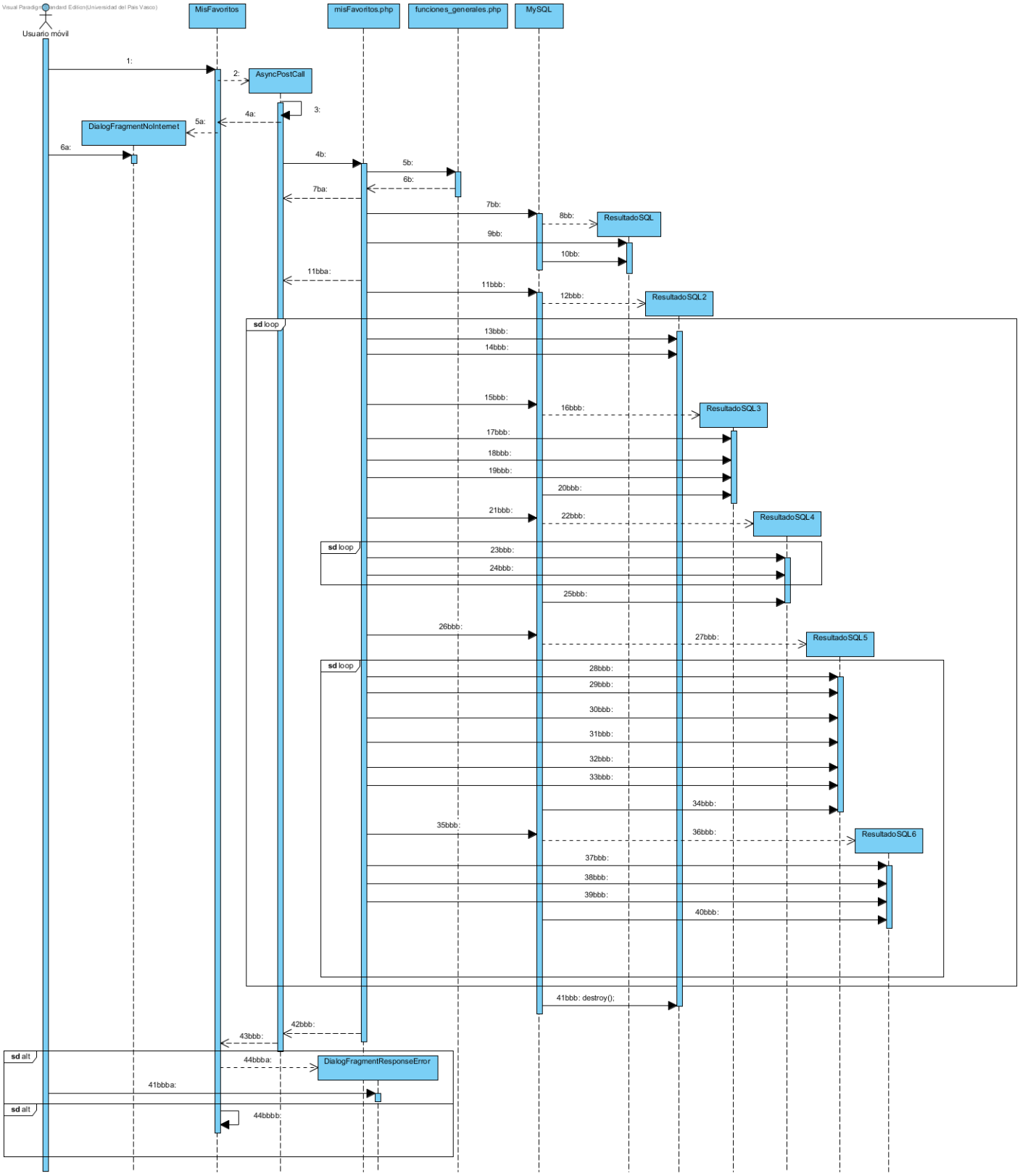


Ilustración 191: Diagrama de secuencia App: Obtener Mis Favoritos

1- Una vez iniciada sesión el usuario pulsará sobre el icono “Mis Favoritos”.
2- new AsyncPostCall(int: reqId, nickUsuario).execute();
3- comprobarConexion();

[Si no encuentra conexión a internet]

4a- return $response: JSON, (status=1, msg=“Cod11”, listaPlantillas=””);
5a- new(DialogFragmentNoInternet.java);
6a- El usuario pulsará aceptar.

[Si tiene conexión a internet]

4b- validarFormulario(String: nick);
5b- validarFormulario(String: nick);
6b- return $validacion;

[Si la validación de PHP detecta caracteres incorrectos en el nick]

7ba- return $response: JSON, (status=1, msg=“Cod13”, listaPlantillas=””);

[Si la validación decide que el nick es válido]

7bb- execSQL(Consulta1);

```
SELECT * FROM usuarios
WHERE nickUsuario= '$nick'
```

8bb- new();
9bb- mysqli_num_rows();
10bb- destroy();

[Si no se encuentra el usuario en el sistema]

11bba- return $response: JSON, (status=1, msg=“Cod13”, listaPlantillas=””);

[Si el usuario existe en el sistema]

11bbb- execSQL(Consulta2);

```
SELECT * FROM favoritos
WHERE nickUsuario= '$nick'
```

12bbb- new();

[Por cada tupla de plantillas que el usuario tenga en favoritos]

13bbb- next();
14bbb- getIdPlantilla(): int;
15bbb- execSQL(Consulta3);

```
SELECT * FROM plantillas
WHERE idPlantilla= '$idPlantilla'
```

16bbb- new();

[Por cada plantilla que devuelva la consulta]

17bbb- next();
18bbb- getNombrePlantilla(): String;
19bbb- getPropietario(): String;
20bbb- destroy();
21bbb- execSQL(Consulta4);

```
SELECT * FROM tagsplantilla
WHERE idPlantilla= '$idPlantilla'
```

22bbb- new();
[Por cada Tag que tenga la plantilla actual]
23bbb- next();
24bbb- getTag(): String;
[Fin loop obtener tags]
25bbb- destroy();
26bbb- execSQL(Consulta5);

```
SELECT * FROM configuracionejercicio
WHERE idPlantilla= '$idPlantilla'
ORDER BY posicion ASC
```

27bbb- new();
[Por cada ejercicio que tiene la plantilla]
28bbb- next();
29bbb- getNombre(): String;
30bbb- getNumSeries(): int;
31bbb- getRepeticiones(): int;
32bbb- getDescanso(): int;
33bbb- getOrden(): int;
34bbb- destroy();
35bbb- execSQL(Consulta6);

```
SELECT * FROM ejercicios
WHERE nombreEjer= '$nomEj'
```

36bbb- new();
37bbb- getGrupoMuscular(): String;
38bbb- getImagen(): String;
39bbb- getDescripcion(): String;
40bbb- destroy();
[Fin loop ejercicios]
[Fin loop plantilla]
[Fin loop ids favoritos]

41bbb- destroy();
42bbb- return $response: JSON; (status=0, result = JSON, listaPlantillas= JSON);
33bbb- return JSONObject();
[Si el atributo status del JSON = 1 (error)]
44bbba- new();
45bbba- el usuario pulsará aceptar;
[Si el atributo status del JSON = 0 (correcto)]
44bbbb- cargarListView();

Eliminar plantilla:

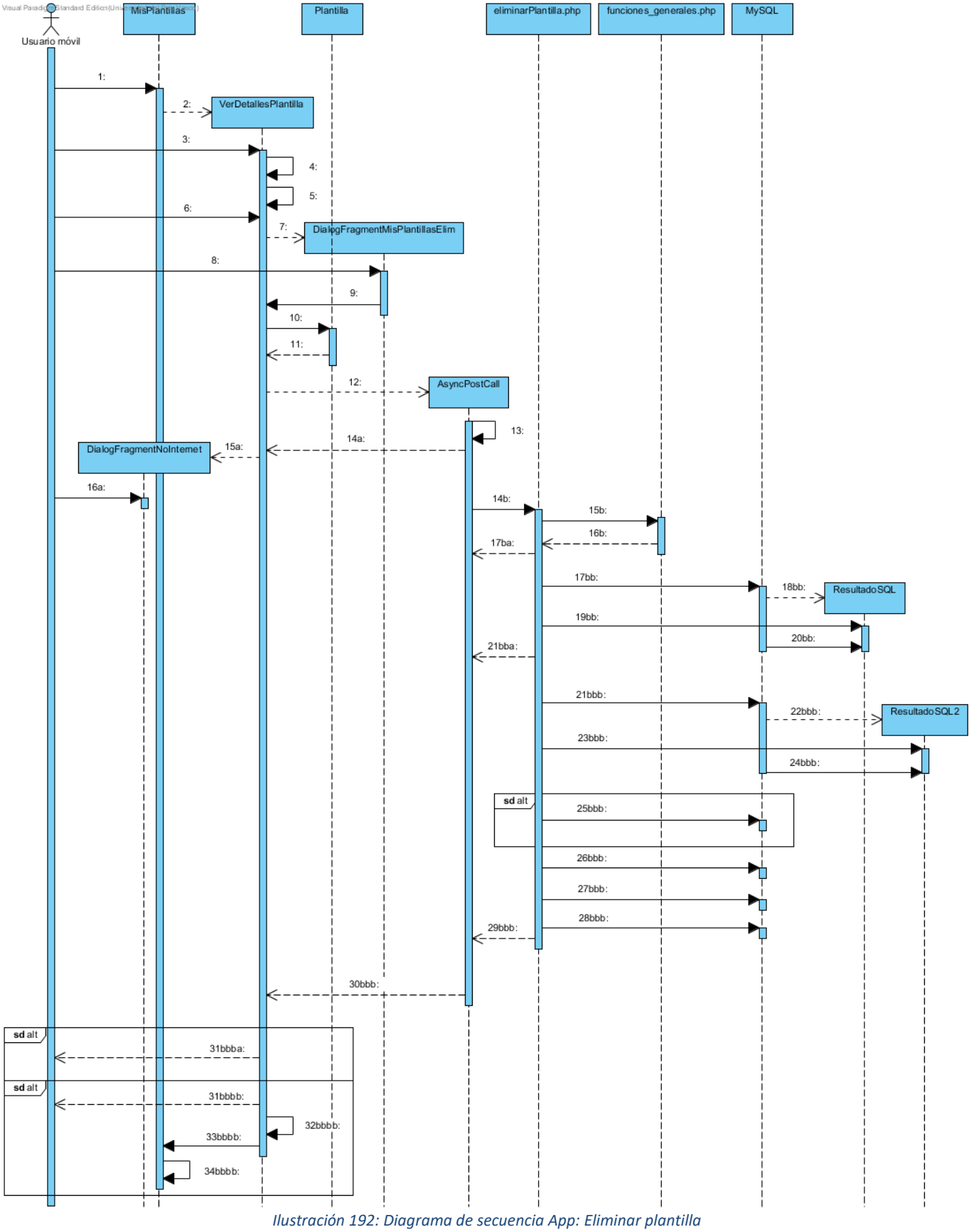


Ilustración 192: Diagrama de secuencia App: Eliminar plantilla

1- El usuario pulsa sobre una de las plantillas de “MisPlantillas”.
2- startActivityForResult(Intent: intent; int: idIntent; byte[]: plantillaSeleccionada);
3- El usuario pulsa sobre el icono de la llave para habilitar operaciones de gestión.
4- onOptionsItemSelected(MenuItem: ítem): boolean
5- alternarConfiguracion();
6- El usuario pulsa sobre el icono de papelera.
7- new(DialogFragmentMisPlantillasElim.java);
8- el usuario pulsará “Eliminar”.
9- onYesSelected(Plantilla: plantilla);
10- getId();
11- return idPlantilla: int;
12- new AsyncPostCall(int: reqId, idPlantilla; String: nick);
13- comprobarConexion();

[Si no tiene conexión a internet]

14a- return $response: JSON, (status=1, msg=“Cod11”);
15ba- new(DialogFragmentNoInternet.java);
16ba- El usuario pulsará aceptar.

[Si tiene conexión a internet]

14b- validarFormulario(String: nick);
14b- validarFormulario(String: nick);
16b- return $validacion;

[Si la validación del PHP determina que el usuario no es correcto]

17ba- return $response: JSON, (status=1, msg=”Cod13”);

[Si la validación del PHP es correcta]

17bb- execSQL(Consulta1);

```
SELECT * FROM plantillas
WHERE nickUsuario= '$nick' && idPlantilla='$idPlantilla'
```

18bb- new();
19bb- mysqli_num_rows();
20bb- destroy();

[Si el usuario no es el propietario de la plantilla]

21bba- return $response: JSON, (status=1, msg=”Cod21”);

[Si el usuario es el propietario]

21bbb- execSQL(Consulta2);

```
SELECT * FROM favoritos
WHERE idPlantilla='$idPlantilla'
```

22bbb- new();

23bbb- mysqli_num_rows();
24bbb- destroy();
[Si la plantilla está en favoritos]
25bbb- execSQL(Consulta3);

```
DELETE FROM favoritos
WHERE idPlantilla='$idPlantilla'
```

[Fin si]
26bbb- execSQL(Consulta4);

```
DELETE FROM configuracionejercicio
WHERE idPlantilla = '$idPlantilla'
```

27bbb- execSQL(Consulta5);

```
DELETE FROM tagsplantilla
WHERE idPlantilla = '$idPlantilla'
```

28bbb- execSQL(Consulta6);

```
DELETE FROM plantillas
WHERE idPlantilla = '$idPlantilla'
```

29bbb- return $response: JSON, (status=0, msg="");
30bbb- return JSONObject();
[Si el atributo status del JSON = 1 (error)]
31bbba- mostrar mensaje Toast con los motivos del fallo.
[Si el atributo status del JSON = 0 (correcto)]
31bbbb- mostrar mensaje Toast informando de la eliminación.
32bbbb- finish();
33bbbb- onActivityResult(int: requestCode; Intent: idIntent);
34bbbb- cargarPlantillas();

Buscar plantillas:

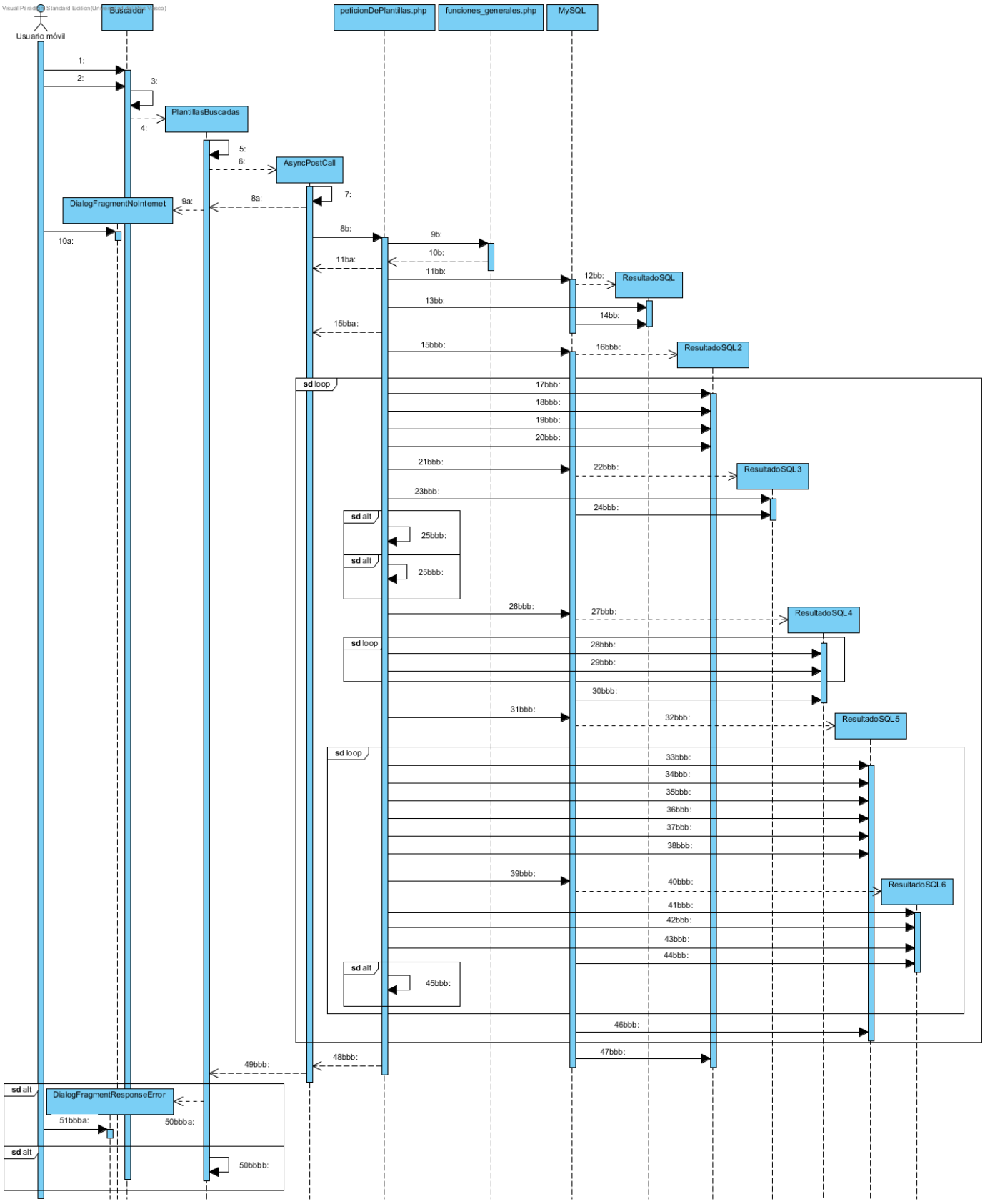


Ilustración 193: Diagrama de secuencia App: Buscar plantillas

1- El usuario marca aquellos grupos musculares que quiere que contengan las plantillas a buscar.
2- El usuario pulsará sobre el icono inferior derecho.
3- serializarSeleccion(): byte[];
4- startActivity(byte[]: criteriosBusqueda);
5- desSerializarSeleccion(byte[] seleccion): ArrayList<String>;
6- new AsyncPostCall(int: reqId; String: nick; ArrayList<String>: criterioBusqueda);
7- comprobarConexion();

[Si no tiene conexión a internet]

8a- return $response: JSON, (status=1, msg="Cod11", listaPlantillas="");
9a- new(DialogFragmentNoInternet.java);
10a- el usuario pulsará Aceptar.

[Si tiene conexión a internet]

8b- validarFormulario(String: nick);
9b- validarFormulario(String: nick);
10b- return $validacion;

[Si la validación PHP detecta que el nick no es válido]

11ba- return $response: JSON, (status=1, msg="Cod13", listaPlantillas="");

[Si la validación PHP es correcta]

11bb- execSQL(Consulta1);

```
SELECT * FROM usuarios
WHERE nickUsuario= '$nick'
```

12bb- new();
13bb- mysqli_num_rows();
14bb- destroy();

[Si el usuario no está registrado en el sistema]

15bba- return $response: JSON, (status=1, msg="Cod13", listaPlantillas="");

[Si el usuario existe]

15bbb- execSQL(Consulta2);

```
SELECT * FROM plantillas
ORDER BY nombre ASC
```

16bbb- new();

[Por cada plantilla que devuelva la tupla]

17bbb- next();
18bbb- getIdPlantilla(): int;
19bbb- getNombre(): String;
20bbb- getPropietario(): String;

21bbb- execSQL(Consulta3);

```
SELECT * FROM favoritos
WHERE idPlantilla='$idPlantilla'
&& nickUsuario='$nick'
```

22bbb- new();
23bbb- mysqli_num_rows();
24bbb- destroy();
[Si el usuario tiene la plantilla en favoritos]
    25bbba- $response[idPlantilla][esFavorita] = true;
[Si no la tiene en favoritos]
    25bbbb- $response[idPlantilla][esFavorita]= false;
26bbb- execSQL(Consulta4);

```
SELECT * FROM tagsplantilla
WHERE idPlantilla= '$idPlantilla'
```

27bbb- new();
[Por cada tag de la plantilla actual]
    28bbb- next()
    29bbb- getTag(): String;
[End loop tags]
30bbb- destroy();
31bbb- execSQL(Consulta5);

```
SELECT * FROM configuracionejercicio
WHERE idPlantilla= '$idPlantilla'
ORDER BY posicion ASC
```

32bbb- new();
[Por cada ejercicio de la plantilla]
    33bbb- next();
    34bbb- getNombreEjer(): String;
    35bbb- getNumSeries(): int;
    36bbb- getNumReps(): int;
    37bbb- getDescanso(): int;
    38bbb- getOrden(): int;
    39bbb- execSQL(Consulta6);

```
SELECT * FROM ejercicios
WHERE nombreEjer= '$nomEj'
```

40bbb- new();
41bbb- getGrupoMuscular(): String;
42bbb- getImagen(): String;
43bbb- getDescripcion(): String;
44bbb- destroy();
[Si la tiene todos los grupos musculares a buscar]
45bbb- listaPlantillas.add(plantillaActual);
[Fin si]
[Fin loop ejercicios]
46bbb- destroy();
[Fin loop plantillas]
47bbb- destroy();
48bbb- return $response: JSON, (status=0, msg="", listaPlantillas= JSON);
49bbb- return JSONObject();
[Si el atributo Status del JSON = 1 (error)]
50bbba- new(DialogFragmentResponseError.java);
51bbba- el usuario pulsará Aceptar.
[Si el atributo Status del JSON = 0 (correcto]
50bbb- cargarListView();

Pedir datos (Mi Cuenta):

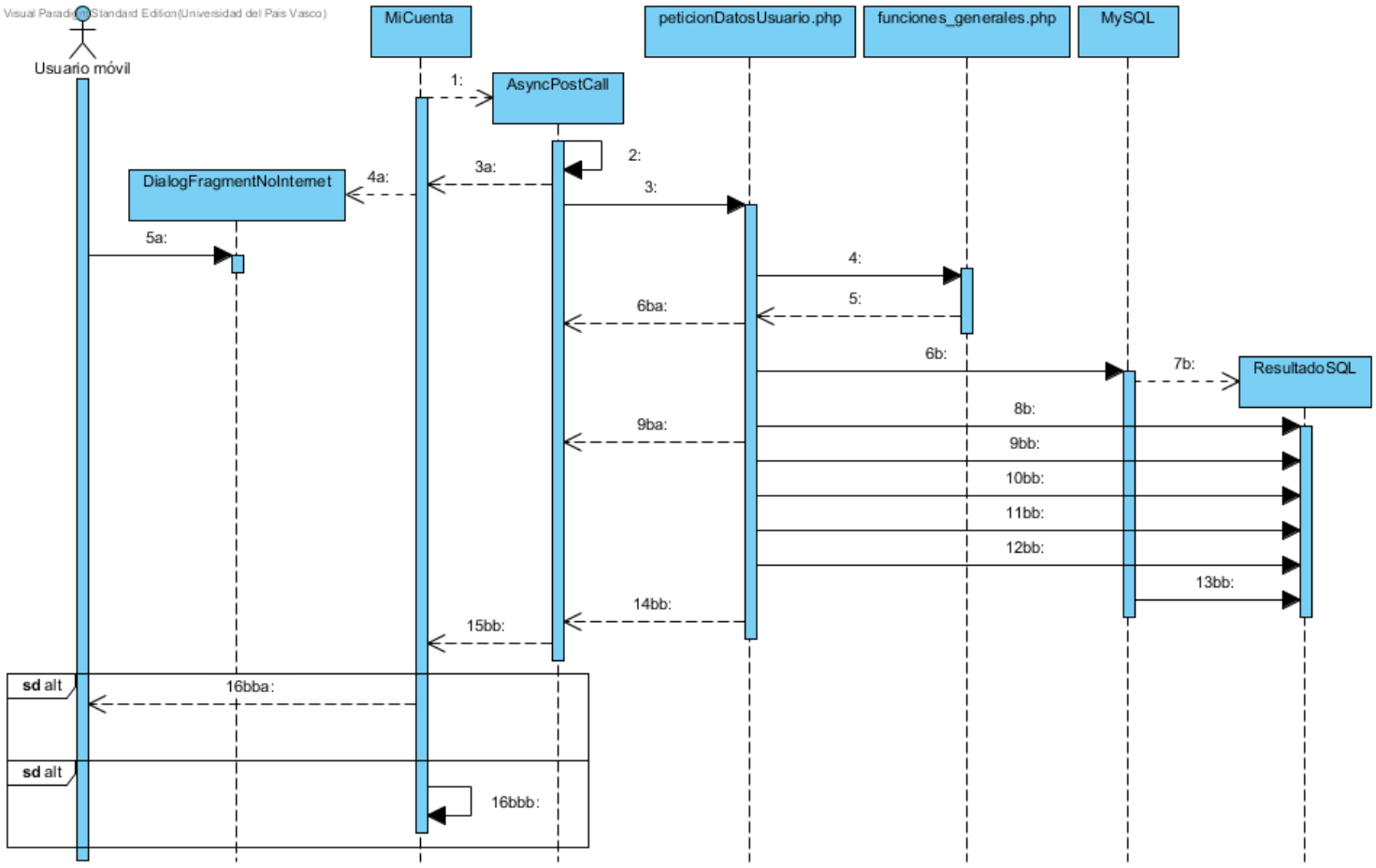


Ilustración 194: Diagrama de secuencia App: Pedir datos (Mi Cuenta)

1- new AsyncPostCall(int: reqId; String: nick);
2- comprobarConexion();

[Si el dispositivo no tiene conexión a internet]

3a- return $response: JSON, (status=1, msg="Cod11", datosUsuario="");
4a- new(DialogFragmentNoInternet.java);
5a- el usuario pulsará Aceptar.

[Si el dispositivo tiene conexión a internet]

3b- validarFormulario(String: nick);
4b- validarFormulario(String: nick);
5b- return $validacion;
6b- execSQL(Consulta1);

```
SELECT * FROM usuarios
WHERE nickUsuario ='$nick'
```

7b- new();
8b- mysqli_num_rows();

[Si el usuario no tiene cuenta en el sistema]

9ba- return $response: JSON, (status=1, msg="Cod11", datosUsuario="");

[Si el usuario tiene cuenta]

9bb- getNombre(): String;
10bb- getApellidos(): String;
11bb- getEmail(): String;
12bb- getNickUsuario(): String;
13bb- destroy();
14bb- return $response: JSON, (status=0, msg="", datosUsuario= usuario:JSON);
15bb- return JSONObject();

[Si el atributo status del JSON = 0 (error)]

16bba- new(DialogFragmentResponseError.java)
17bba- el usuario pulsará Aceptar.

[Si el atributo Status del JSON = 1 (correcto)]

16bbb- mostrarDatosUsuario(JSONObject: datosUsuario);

Modificar datos:

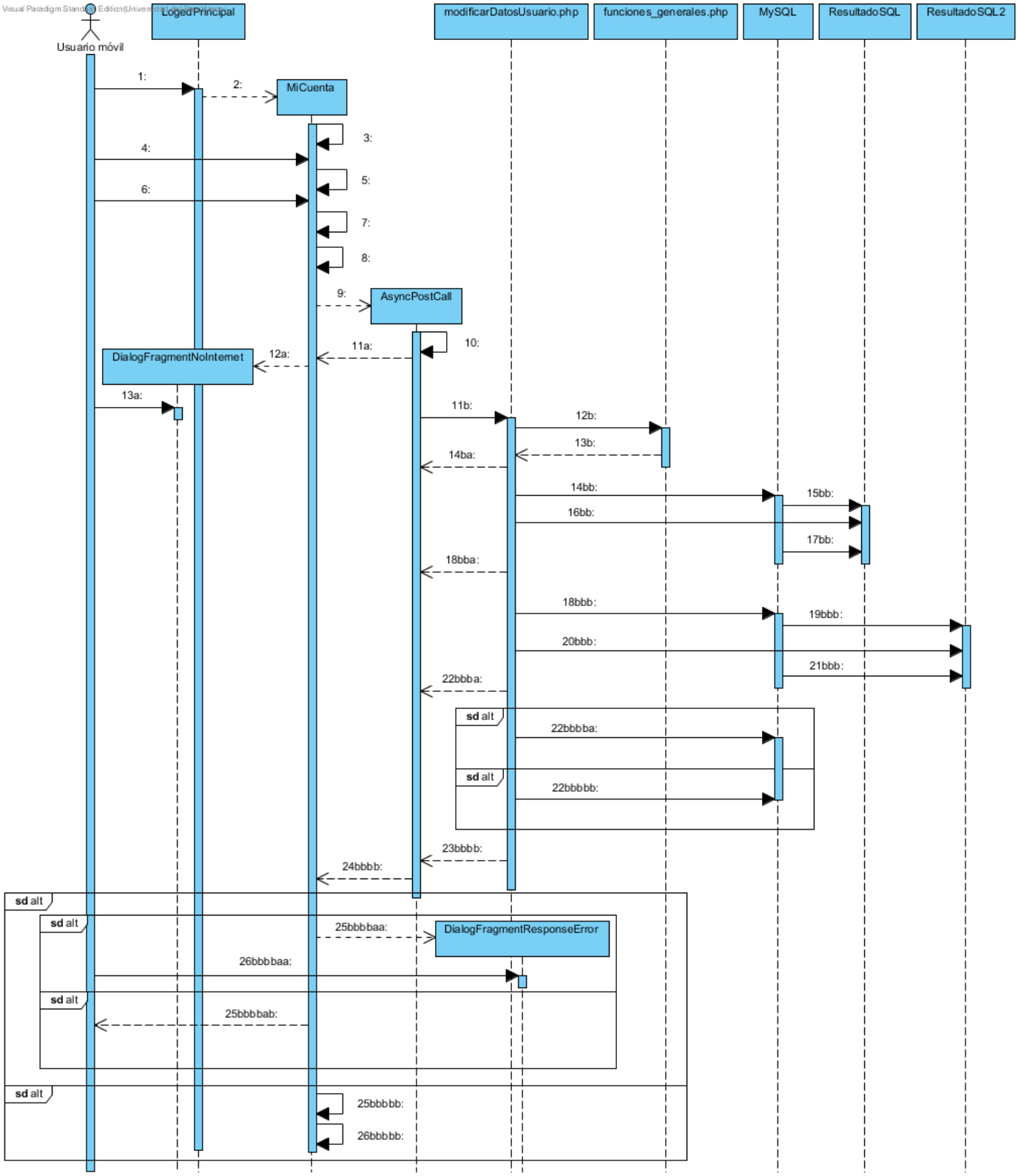


Ilustración 195: Diagrama de secuencia App: Modificar datos

1- Una vez iniciada sesión el usuario pulsará en “Mi Cuenta”.
2- startActivity();
3- pedirDatos();
4- El usuario modifica aquellos datos que desee.
5- permitirGuardado();
6- El usuario pulsará sobre el icono de la ActionBar para guardar las modificaciones.
7- onOptionsItemSelected(MenuItem: ítem): boolean;
8- comprobarDatos();
9- new AsyncPostCall(int: reqId; String: nick, nombre, apellidos, email, pass1, pass2);
10- comprobarConexion();

[Si el dispositivo no tiene conexión a internet]

11a- return $response: JSON, (status=1, msg=”Cod11”, datosUsuario=””);
12a- new(DialogFragmentNoInternet.java);
13a- El usuario pulsa Aceptar.

[Si tiene conexión a internet]

11b- validarFormulario(String: nick, pass1, pass2, nombre, apellidos, email);
12b- validarFormulario(String: nick, pass1, pass2, nombre, apellidos, email);
13b- return $validacion;

[Si la validación PHP detecta que algún campo no cumple los patrones]

14ba- return $response: JSON, (status=1, msg=”CodXX”), donde XX es el código que originó el error {02,04,05,06,07,08}.

[Si la validación PHP considera todos los parámetros como válidos]

14bb- execSQL(Consulta1);

```
SELECT * FROM usuarios
WHERE nickUsuario='$nick'
```

15bb- new();
16bb- mysqli_num_rows();
17bb- destroy();

[Si el usuario no está registrado en el sistema]

18bba- return $response: JSON, (status=1, msg=”Cod13”);

[Si el usuario tiene cuenta]

18bbb- execSQL(Consulta2);

```
SELECT * FROM usuarios
WHERE email='$email'
&& nickUsuario <> '$nick'
```

19bbb- new();
20bbb- mysqli_num_rows();
21bbb- destroy();

[Si el nuevo email ya está cogido por otro usuario]

22bbba- return $response: JSON, (status=1, msg="Cod10");

[Si el nuevo email está libre]

[Si el usuario no quiere actualizar contraseña]

22bbbba- execSQL(Consulta3);

```
UPDATE usuarios
SET nombre='$nombre', apellidos='$apellidos', email='$email'
WHERE nickUsuario='$nick'
```

[Si el usuario incluyo nueva contraseña para su modificación]

22bbbbb- execSQL(Consulta4);

```
UPDATE usuarios
SET pass=md5('$pass1'), nombre='$nombre',
apellidos='$apellidos', email='$email'
WHERE nickUsuario='$nick'
```

[Fin si]

23bbbb- return $response: JSON, (status=0, msg="");

24bbbb- return JSONObject();

[Si el atributo Status del JSON = 1 (error)]

[Si el atributo msg del JSON = "Cod13" (error grave)]

25bbbbaa- new (DialogFragmentResponseError.java);

26bbbbaa- El usuario pulsa Aceptar.

[Si el atributo msg del JSON !="Cod13" (error leve)]

25bbbbab- mostrar un mensaje Toast con los motivos del error.

[Si el atributo Status del JSON = 0 (correcto]

25bbbbb- thDatosIniciales.put(String: dato);

26bbbbb- resetearDatosIniciales();

Cerrar sesión:

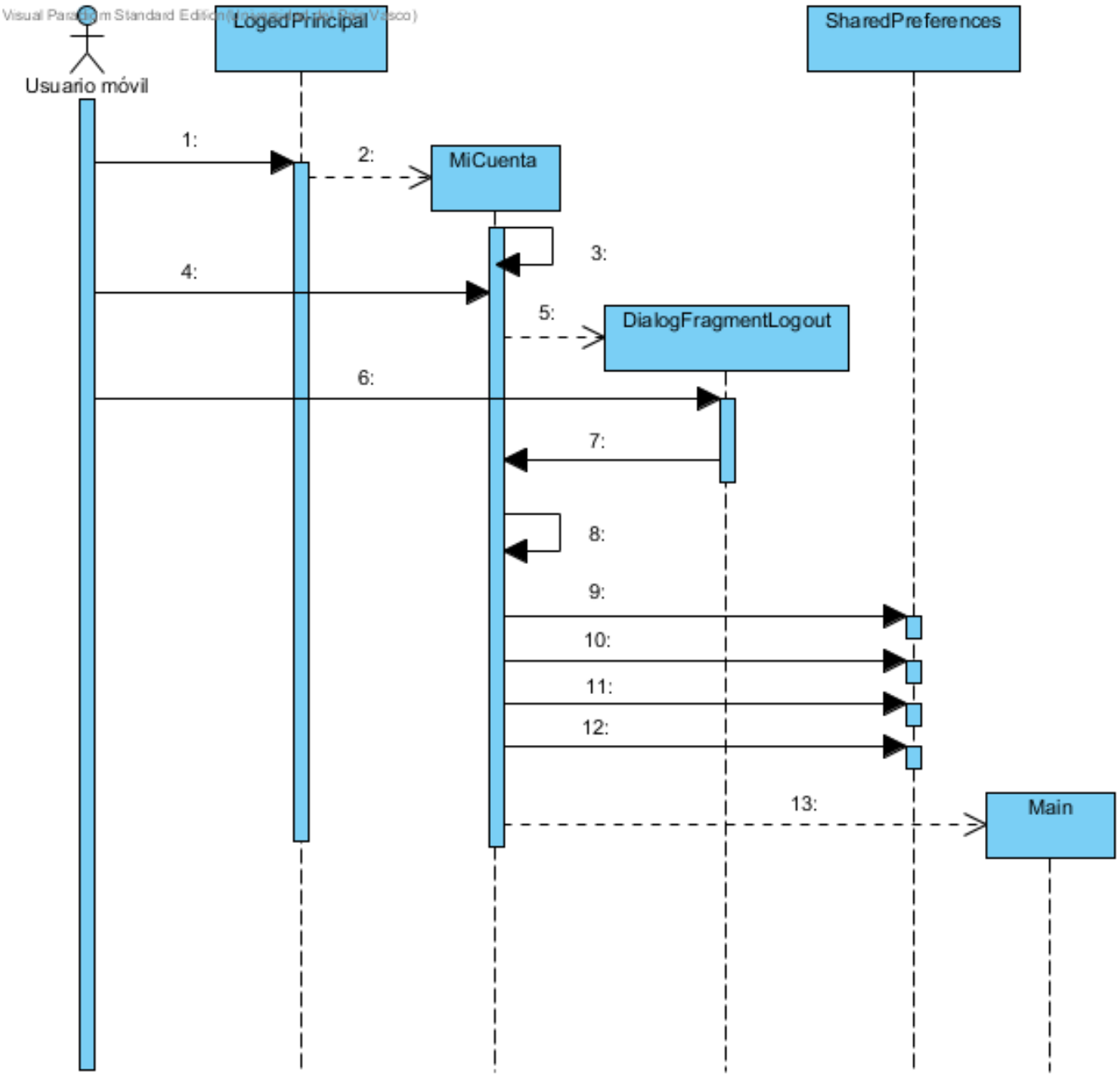


Ilustración 196: Diagrama de secuencia App: Cerrar sesión

1- El usuario pulsa para acceder al área “MiCuenta”.
2- startActivity();
3- pedirDatos();
4- El usuario pulsa sobre el icono de la puerta para cerrar sesión.
5- New (DialogFragmentLogout.java);
6- El usuario pulsa Aceptar.
7- onOk();
8- onLogoutConfirmed();
9- putBoolean(“recordarLogin”, false);
10- remove(“sesion”);
11- apply();
12- startActivity(); // Este intent es especial y tiene un Flag que ellimina todas las actividades que estuviesen en la pila.

Entrenamiento- Contar repeticiones:

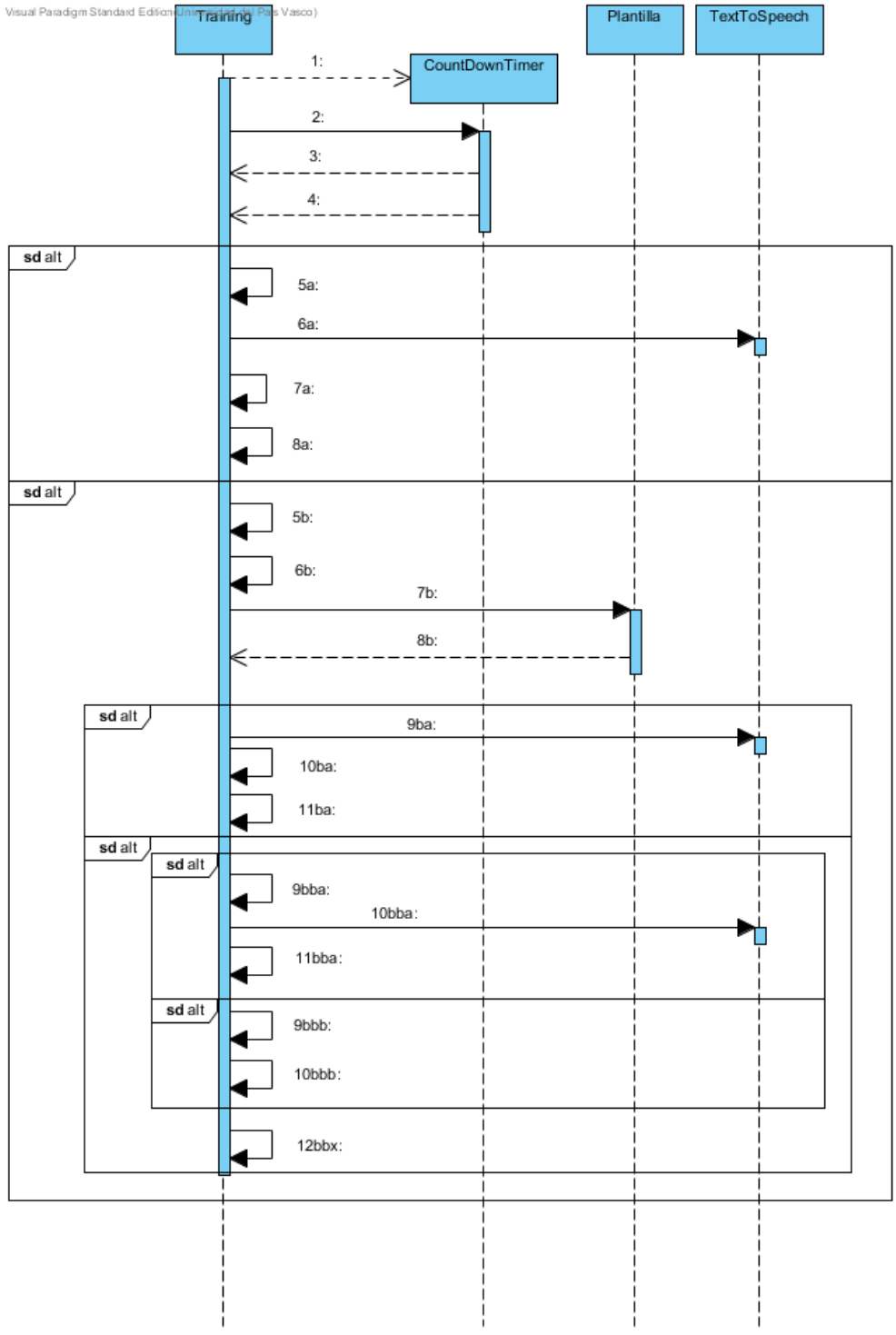


Ilustración 197: Diagrama de secuencia App: Entrenamiento- Contar repeticiones

1- new CountDownTimer(int: tMax, tPaso);
2- start();

[Cada vez que transcurre un tiempo equivalente a tPaso]

3- enMarcha = true;

[Cuando tMax llega a 0]

4- siguienteRepeticion();

[Si la repetición actual es menor al número de repeticiones del ejercicio actual]

5a- posRepActual;
6a- hablar(alternarCantidad()); // se encarga de decir “un”, “dos” traducido
7a- contarRepeticiones();
8a- actualizarDatos();

[Sino]

5b- posRepActual=1;
6b- siguienteEjercicio();
7b- mPlantilla.siguienteEjercicio();
8b- return siguienteEj: Ejercicio;

[Si siguienteEj es null (entrenamiento acabado)]

9ba- hablar(“Entrenamiento terminado”);
10ba- inicializarEntrenamiento();
11ba- cambiarIconoPlay();

[Si siguienteEj no es null]

[Si el siguiente ejercicio es diferente (cambio de serie y ejercicio)]

9bba- actualizarDatos();
10bba- hablar(“Tiempo de descanso”);
11bbb- cargarImagenEjercicio();

[Si el siguiente ejercicio es el mismo (cambio únicamente de serie)]

9bbb- posSerieActual++;
10bbb- hablar(“Tiempo de descanso”);

[Fin si]

12bbx- descansar();

Entrenamiento- Actualizar descanso:

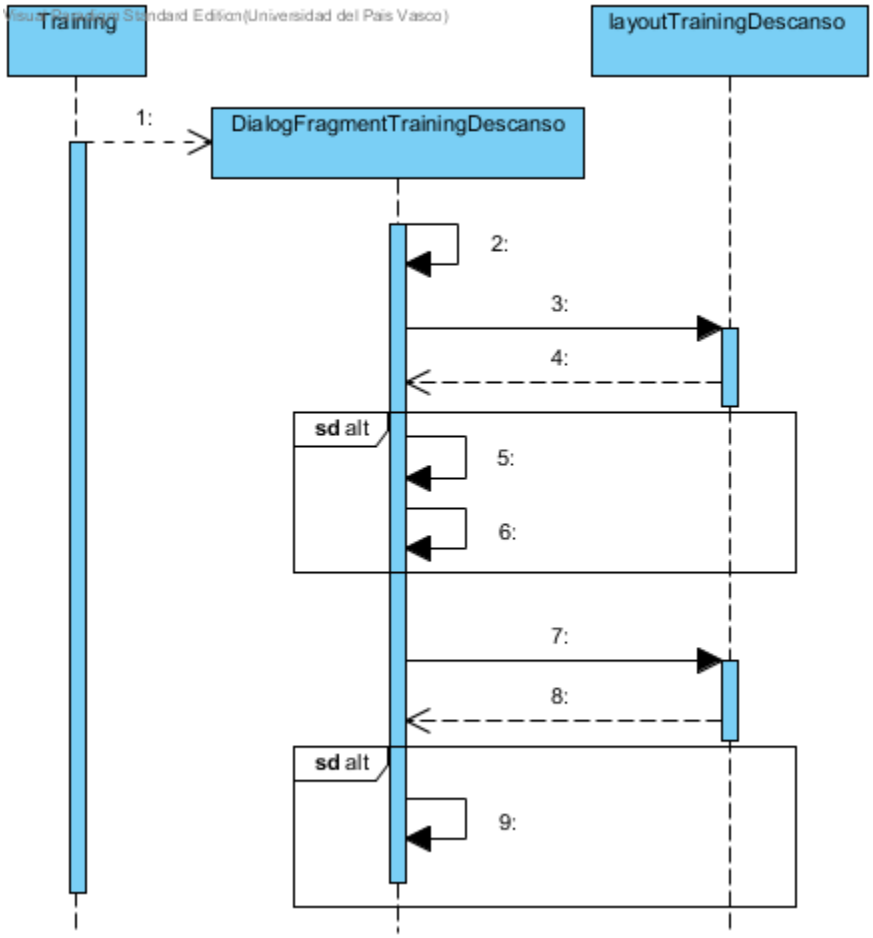


Ilustración 198: Diagrama de secuencia App: Entrenamiento- Actualizar descanso

1- new (DialogFragmentTrainingDescanso.java);
2- getArguments(): int; (devuelve maxDescanso y descansoRestante, integers que nos sirven para calcular el tiempo de descanso transcurrido para el ProgressBar).
3- findViewById(int: idTvDescanso);
4- return tvDescanso: TextView;

[Si tvDescanso no es null]

5a- tvDescanso.setTypeface(Typeface); //cambia el aspecto de las letras a una fuente personalizada.

6a- tvDescanso.setText(descansoRestante);

7- findviewById(int: idProgressBar);
8- return pBTiempo : ProgressBar;

[Si el ProgressBar no es null]

9a- pBTiempo.setProgress(maxDescanso-mDescansoRestante);

Entrenamiento- Descansar:

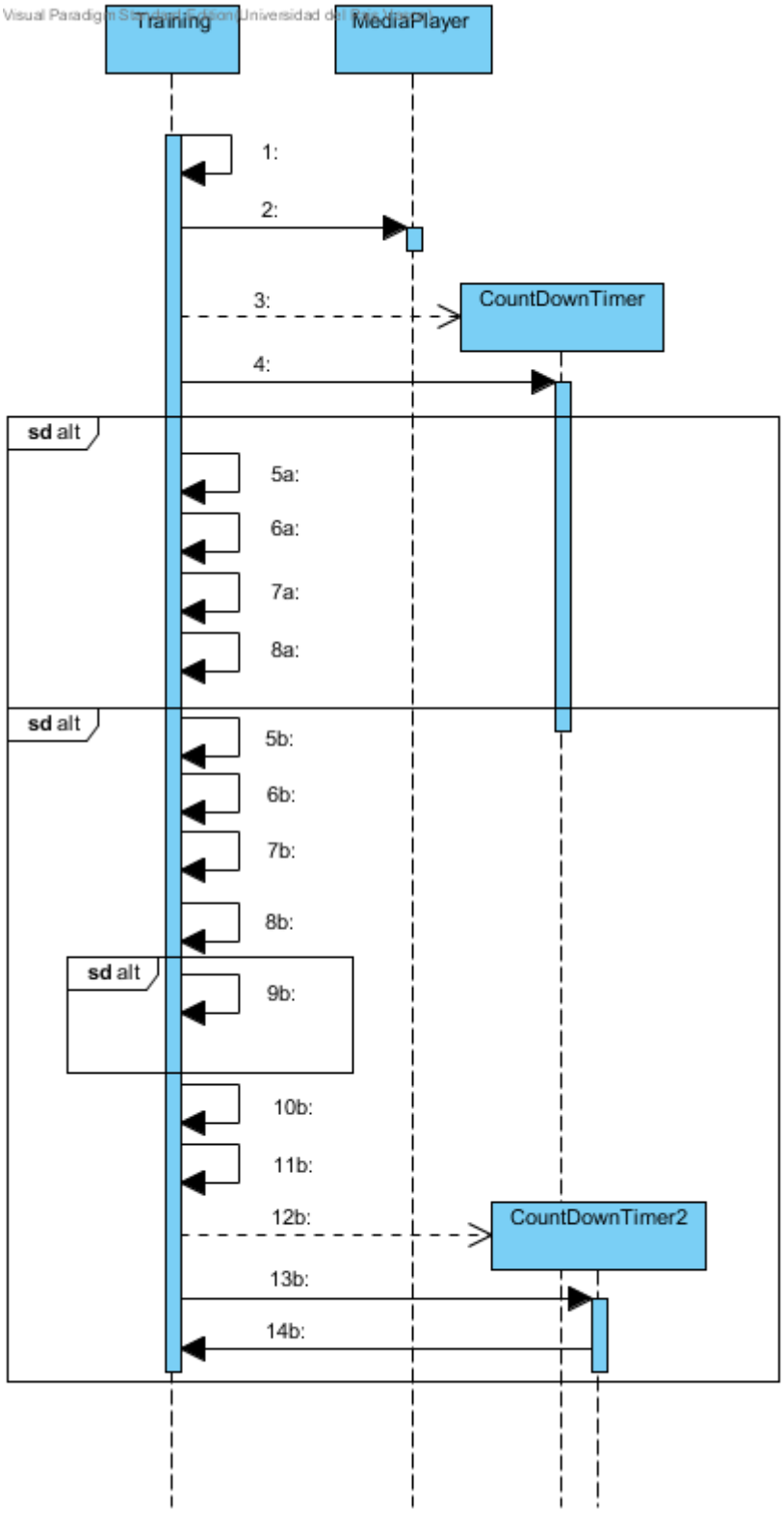


Ilustración 199: Diagrama de secuencia App: Entrenamiento- Descansar

1- actualizarDescanso();
2- mp.start();
3- new CountDownTimer(int: maxTime, tPaso);
4- start();

[Cada vez que transcurre un tiempo equivalente a tPaso]

5a- enMarcha = true;
6a- enDescanso = true;
7a- mDescanso--;
8a- actualizarDescanso();

[Cuando el tiempo maxTime llega a 0]

5b- mDescanso--;
6b- actualizarDescanso();
7b- enMarcha = false;
8b- enDescanso = false;

[Si el cronómetro está activo]

9b- mp.stop();

[Fin si]

10b- presentarEjercicio();
11b- hablar("Serie posSerieActual de maxSeries"); donde "posSerieActual" y "maxSeries" son el nº de serie a realizar y el máximo.
12b- new CountDownTimer(int: maxTime2, tPaso2);
13b- start();

[Cuando el tiempo maxTime2 llega a 0]

14b- cuentaAtras();

Entrenamiento- Play:

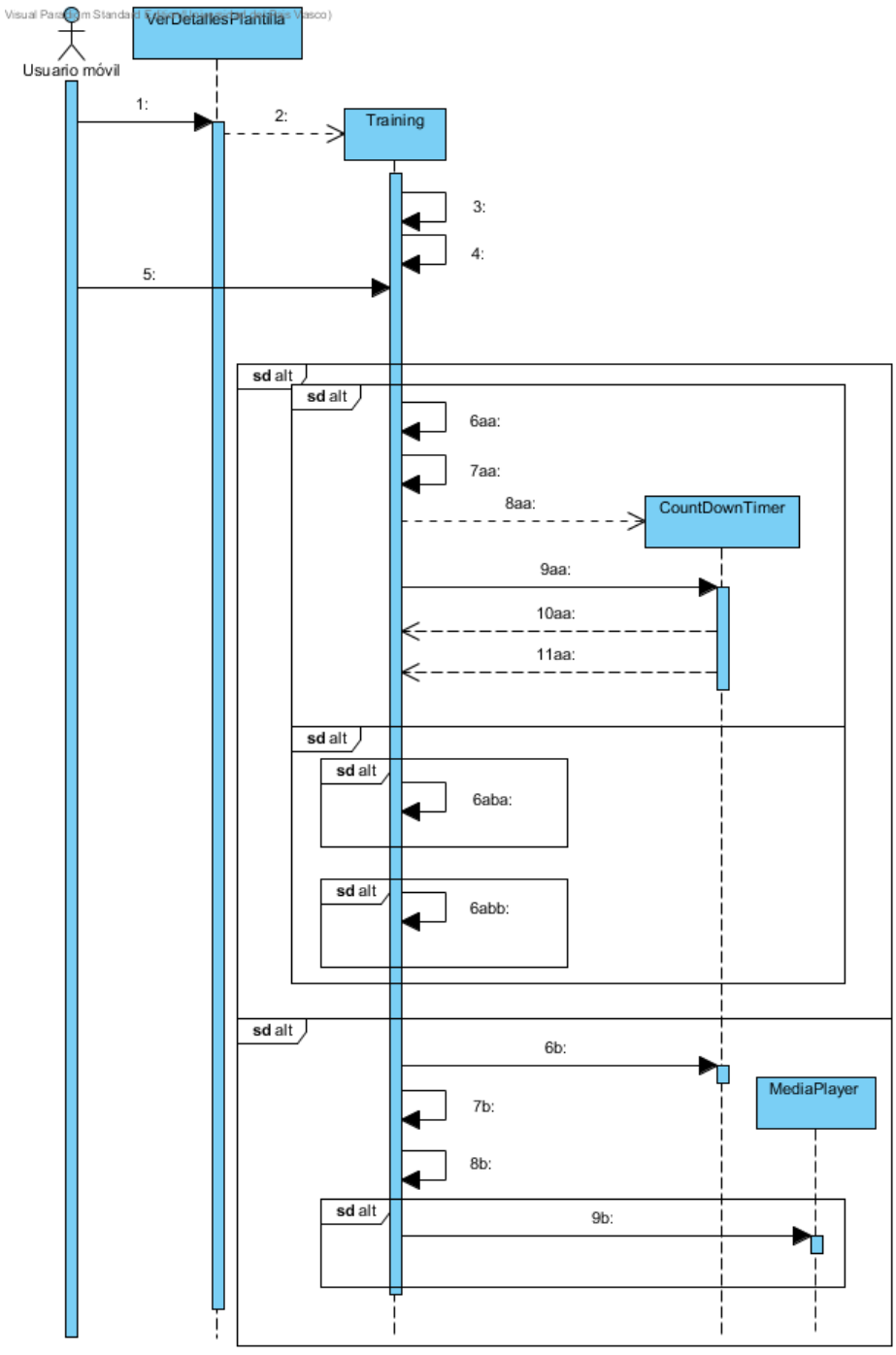


Ilustración 200: Diagrama de secuencia App: Entrenamiento- Play

1- Mientras se visualizan los detalles de una plantilla el usuario pulsará en el icono “Play”.
2- startActivity();
3- inicializarEntrenamiento();
4- inicializarAsistente();
5- pedirDatos();

[Si enMarcha = false]

[Si el ejercicio actual, la serie y la repetición actual = 1 (entrenamiento no iniciado)]

6aa- presentarEntrenamiento();
7aa- presentarEjercicio();
8aa- new CountDownTimer(int: tMax, tPaso);
9aa- start();

[Cada vez que pasa un tiempo equivalente a tPaso]

10aaa- enMarcha = true;

[Fin loop]

[Cuando tMax llega a 0]

10aab- cuentaAtras();

[Si el entrenamiento ya ha sido iniciado]

[Si enDescanso = true]

6aba- descansar();

[Si no está en tiempo de descanso]

6abb- contarRepeticiones();

[Si está activo el entrenamiento]

6b- mContador.cancel();
7b- enMarcha = false;
8b- cambiarIconoPlay();

[Si el mp (mediaPlayer) no es null y está activo (sonido del cronómetro)]

9ba- mp.stop();

[Fin si]

[Fin si]