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

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

ETORAX

Alumno/Alumna: Martinez, Garcia, Julen

Director/Directora (1): Villamañe, Gironés, Mikel

Curso: <2018-2019>

Fecha: Bilbao, 22 de julio de 2019

Resumen

Cuando un paciente se va a enfrentar a una cirugía torácica es muy importante tanto el proceso anterior a la operación (pre-operatorio) como el proceso tras la operación (post-operatorio). Las cirugías de tórax por lo general se tratan de cirugías bastante invasivas, por lo que es muy importante el estado físico en el que llega el paciente a la operación, así como realizar una correcta recuperación de la misma.

ETorax se trata de un entorno que, combinando una aplicación móvil y una página web, permite al equipo médico indicar a cada paciente cuales son la pautas que este debe seguir tanto antes de la cirugía como después de la misma. Permite al paciente consultar el cualquier momento cuales son las pautas indicadas por el equipo médico desde su teléfono móvil. También permite tanto al paciente como al equipo médico comprobar si los objetivos se están cumpliendo de forma correcta.

Laburpena

Ebakuntza torazikoetan garrantzizkoak dira bai ebakuntza aurreko prozesua baita ebakuntza ondokoa ere. Orokorrean, ebakuntza kirurgiko horiek nahiko bortitzak dira eta hori dela eta, ezinbestekoa da pazientearen ebakuntza aurreko egoera fisikoa egokia izatea eta ostean indarberritze prozesua ona izatea.

ETorax, telefono mugikorrerako aplikazioa eta webgunea elkarlanean arituz, medikuei aukera ematen die gaixo bakoitzari ebakuntza aurretik eta ostean jarraitu behar dituen urratsak adierazteko. Bestalde, pazienteak bere telefono mugikorretik, nahi duen momentuan, medikuek ezarritako jarraibideak kontsultatu ditzake. Gainera, medikuei nahiz pazienteari aukera ematen die ezarritako helburuak ondo betetzen ari diren frogatzeko.

Abstract

When a patient of thoracic surgery is going to be operated on, there are two important periods to consider: the pre-operative and the post-operative ones. In general, the thoracic operations are invasive and for this reason, it is very important that before the surgery the patient has to be in good shape, also that he/she has a correct recovery after it.

ETorax is a computing environment in which an application and a website work together. It provides a place to the doctors in where they can give the instructions that the patient has to follow before and after the thoracic surgery. The patient can check, at any moment, in his/her mobile phone the doctors´ recommendations. Finally, ETorax allows the doctors and the patient to verify that the objectives are achieving correctly.

Índice

Índice de ilustraciones

Índice de tablas

1. Introducción

La cirugía torácica es aquella que se dedica al tratamiento quirúrgico de las patologías que afectan al tórax como podrían ser las patologías pulmonares o las patologías cardiovasculares.

Cuando un paciente se enfrenta a una cirugía torácica es muy importante el correcto desarrollo tanto del proceso previo a la operación (preoperatorio) como del proceso posterior a la operación (postoperatorio).

A día de hoy para el desarrollo del preoperatorio y el postoperatorio el médico rellena un folio con los ejercicios y actividades que el paciente ha de llevar a cabo durante el proceso. El paciente es quien debe tener en cuenta día tras día los ejercicios que ha de realizar a partir de las pautas que en el folio aparecen. A sí mismo, el paciente ha de recordar cada día que ha de consultar el folio con las indicaciones y realizar los ejercicios que el médico le mando realizar en la última consulta.

El médico no tiene ningún tipo de información de si el paciente está realizando los ejercicios o no. Tampoco sabe si al paciente hay días que se le olvida realizar dichos ejercicios y si los está realizando de forma correcta. La única información que a día de hoy le llega al médico es la que le proporcionará el propio paciente a lo largo de la próxima consulta que se lleve a cabo. El paciente podría mentir, u olvidarse de si ha realizado los ejercicios, lo cual podría suponer un mayor riesgo a la hora de realizar la cirugía. De la misma manera si el paciente no se acordase de realizar los ejercicios recomendados por el médico tras la operación podría suponer una mala rehabilitación con consecuencias para la salud del paciente. Dichos perjuicios en la salud del paciente podrían hacer que fuese necesario volver a operarlo, con los riegos que eso supondría además del gasto económico y de los recursos que serían necesarios.

Por lo tanto, con este trabajo de fin de grado lo que se pretende es ayudar a ambas partes implicadas en los procesos mencionados anteriormente. La plataforma que se creará consistirá tanto de una página web como una app disponible para diferentes tipos de terminales móviles, independientemente de su sistema operativo.

La página web será la que permitirá al equipo médico introducir las pautas que el paciente ha de realizar, dichas pautas se introducirán de forma personalizada. De la misma manera dicha página debe permitir a los médicos ver si el paciente va consiguiendo los objetivos marcados.

Por otro lado, la app está orientada a los pacientes, en ella los pacientes podrán consultar los ejercicios propuestos por el equipo médico. Además de eso, la app será la encargada de recoger los datos para comprobar si el paciente cumple con los objetivos marcados por el médico.

1.1. Razones para la elección del TFG

Uno de los consejos que siempre se da a la hora de enfrentarse al trabajo de fin de grado es que se debe elegir una idea que te guste y te motive. Este es uno de los mayores puntos que hizo que me inclinara por este TFG. ETorax es un proyecto en el que el objetivo es desarrollar una plataforma la cual sea capaz de ayudar tanto al equipo médico como al paciente en un proceso que puede resultar complicado, en específico para el paciente. Gracias a este proyecto se podría conseguir una mejora en la calidad de vida del paciente además de que resultaría de gran ayuda en el día a día del equipo médico. El hecho de poder ayudar a otras personas me motiva y hace que no sienta que estoy realizando este TFG solo para finalizar el grado.

Otro de los motivos por los que me decante por este TFG es que las tecnologías utilizadas para este proyecto son tecnologías que actualmente están demandadas en el ámbito empresarial. Además de que dichas tecnologías no han sido impartidas a lo largo del grado. Por lo que durante la realización del TFG podría aprender tecnologías, las cuales hasta el momento desconozco y de esta manera aumentar mis conocimientos. Por lo que podría suponerme una ventaja a la hora de encontrar un empleo tras finalizar los estudios.

Otro de los motivos por el cual elegí este TFG es que se trata de un proyecto que dará solución a un problema de la vida real, y que tras la finalización del mismo será utilizado.

Juntando los dos puntos anteriores sería un proyecto real que está en uso y el cual ha sido creado con herramientas demandadas por las empresas. El hecho de poseer un proyecto en uso con nuevas tecnologías podría suponer una buena carta de presentación a la hora de solicitar un puesto de trabajo frente a una empresa.

2. Planteamiento inicial

A lo largo de este capítulo se abordarán los apartados referentes al planteamiento inicial del proyecto. Es decir, se expondrán los objetivos de eTorax y se especificará cual es el alcance del proyecto, se realizará una estimación del tiempo que llevará realizar cada uno de los diferentes aspectos del proyecto, se detallarán las herramientas utilizadas para el desarrollo del mismo, se realizará la gestión de riesgos y por últimos se hará una evaluación del coste económico que supondría el desarrollo del proyecto.

2.1. Objetivos

Los objetivos principales que se persiguen con la realización de este proyecto son los siguientes:

- La creación de una página web que permita al equipo médico de un paciente de cirugía de tórax definir las pautas que el paciente ha de seguir tanto en el preoperatorio como en el postoperatorio. Además de recibir feedback del paciente y poder comprobar que realmente está siguiendo las pautas marcadas por el equipo médico.

  La web debe permitir que el médico pueda actualizar las pautas de un paciente si tras recibir el feedback considera que este ha sido subestimado o sobreestimado.

  La página web también deberá permitir al equipo médico ver las gráficas del desarrollo del paciente. La web deberá crear graficas personalizadas para cada paciente en las que se muestre la evolución del paciente a lo largo del proceso del preoperatorio, la operación y el postoperatorio.

- La creación de una aplicación móvil que le permita al paciente de una cirugía torácica disponer en el móvil de todas las pautas que ha de realizar para el correcto desarrollo tanto de la fase de preoperatorio como la de postoperatorio. Los ejercicios que el equipo médico le asigne al paciente irán acompañados por un video explicativo de cómo se realiza el ejercicio de forma correcta.

  En la aplicación habrá un apartado de recomendaciones en el cual se mostrarán una serie de recomendaciones generales las cuales irán acompañadas por una descripción y un gif para que las recomendaciones se comprendan más fácilmente.

Además, si el paciente no ha realizado los ejercicios a una hora determinada la aplicación se encargará de recordar al paciente que todavía no ha realizado los ejercicios.

Durante la realización de algunos ejercicios como puede ser caminar, la aplicación podrá enviar avisos al paciente si está caminando muy deprisa o muy despacio. Además, le dirá cuando ha llegado a la mitad de su recorrido recomendado por el personal médico para que si quiere pueda dar la vuelta y dirigirse a casa.

Así mismo la aplicación deberá recoger información de la actividad diaria del paciente y realizarle las preguntas asignadas por el equipo médico a diario. Por ejemplo, si te trata de un paciente fumador el médico le podría asignar las siguientes preguntas: "¿Has fumado hoy?" y "¿Cuántos cigarrillos has fumado?".

- El aprendizaje personal de tecnologías que hasta el momento desconocía y que además son requeridas actualmente por las empresas.
  Adquirir conocimientos del desarrollo de páginas web y el desarrollo de aplicaciones móviles para distinto sistemas operativos.

2.2. Definiciones, acrónimos y abreviaturas

- REST: es el acrónimo de "REpresentational State Transfer". La clave principal de una arquitectura REST es que se trata de un tipo de arquitectura sin estado o "stateless" lo cual hace que entre dos llamadas diferentes no se almacenen los datos ni el estado. Esto hace que en cada llamada haya que recordar quien hace la llamada, normalmente a través de un token. (Tomàs, 2014)

- API: las siglas significan "Application Programming Interface"." *Es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes de la misma manera en que la interfaz de usuario facilita la interacción humano-software."* (Merino, 2014)

- API REST: Se trata de una API que sigue la arquitectura REST.

- Front-end: El front end de una aplicación se trata de la parte de la aplicación que se encuentra en el cliente. El front-end es la parte de la aplicación con la que interactúa el usuario de forma directa. Se trata de la parte gráfica de una aplicación, el front-end se encarga de realizar las peticiones al back-end. (Diaz, 2014)

- Back-end: Es la parte de la aplicación que se encuentra en el servidor. El back-end es la parte de la aplicación encargada de comunicarse con la base de datos y controlar el manejo de sesiones. El back-end se encarga

de recibir y gestionar las peticiones realizadas por el front-end. (Diaz, 2014)

- JSON: *"(JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript (…). JSON es un formato de texto que es completamente independiente del lenguaje (…)"* (JSON, s.f.)

- Framework: *"Un framework es una estructura conceptual con módulos de software, bibliotecas, programas en lenguaje interpretado que se utiliza para el desarrollo de software."* (Suarez, 2016)

2.3. Alcance

Uno de los primeros pasos a la hora de realizar un proyecto es definir el alcance del mismo. Para definir el alcance del proyecto de forma correcta será necesario dividirlo en diferentes tareas que se deberán realizar para poder conseguir los objetivos del proyecto.

Además, en este apartado se explicará cual será la metodología que se usará para llevar a cabo el proyecto, el proyecto de desarrollará usando ciclos de vida.

2.3.1. Ciclo de vida

La metodología elegida para el desarrollo eTorax es la de ciclos de vida incremental. La cual es una metodología que consiste en la iteración de ciclos de vida en cascada. En esta metodología de trabajo con cada iteración se obtendrá un prototipo mejor que el anterior y con más funcionalidades, estando cada vez más cerca del objetivo final.

Existen diferentes motivos por los que se ha elegido esta metodología de trabajo en lugar de alguna de las otra existentes. En esta metodología se dividirá el proyecto en pequeños prototipos a los cuales que les ira añadido funcionalidades, el hecho de dividir el proyecto en funcionalidades menores hace que la organización del mismo sea más sencilla y que se puedan detectar más fácilmente las necesidades de cada fase.

Otro de los motivos por el que se seleccionó esta metodología es por todo lo que supone tener un prototipo que funcione y al cual añadir funcionalidades. El hecho de disponer de un prototipo funcional hace que si en algún momento al intentar añadirle alguna funcionalidad deja de funcionar todo el proyecto siempre queda la opción de volver al prototipo que estaba funcionando y a partir de ahí volver a empezar. Además, el hecho de que el prototipo funcione con algunas de las funcionalidades del proyecto

final supone una motivación que da ganas de seguir adelante con más fuerza al ver lo conseguido y como va tomando forma el proyecto.

A continuación, se mostrarán los prototipos en los que se dividirá el proyecto, se irán realizando los prototipos de las pagina web y la app móvil al mismo tiempo según se vayan realizando funcionalidades:

- Prototipos de la página web:
    1. Login y Logout.
    2. Agregar paciente
    3. Asignar ejercicios y preguntas al paciente
    4. Modificar datos del paciente
    5. Visualizar los ejercicios y preguntas de un paciente
    6. Visualizar información diaria del paciente
    7. Ver gráficos del paciente

- Prototipos de la app móvil:

    1. Login del paciente
    2. Realizar preguntas iniciales
    3. Ver datos del paciente
    4. Mostrar preguntas frecuentes
    5. Responder preguntas
    6. Marcar ejercicio como realizado
    7. GPS y velocidad

2.3.2. Estructura de descomposición de trabajo

La estructura de descomposición del trabajo se trata de un diagrama en el cual se representan todas y cada una de las tareas que forman parte de la realización de un proyecto.

Normalmente se utilizan los mismos módulos en el EDT: organización, análisis, diseño, implementación, pruebas y documentación. Sin embargo, en este caso se ha decidido utilizar módulos personalizados para conseguir una mayor adecuación a las necesidades del proyecto.

Los módulos que se han decidido utilizar son los siguiente: organización y captura de requisitos, aprendizaje, desarrollo web, desarrollo móvil y por último documentación.

A continuación, en la siguiente imagen se muestra el esquema de la estructura de descomposición del trabajo.

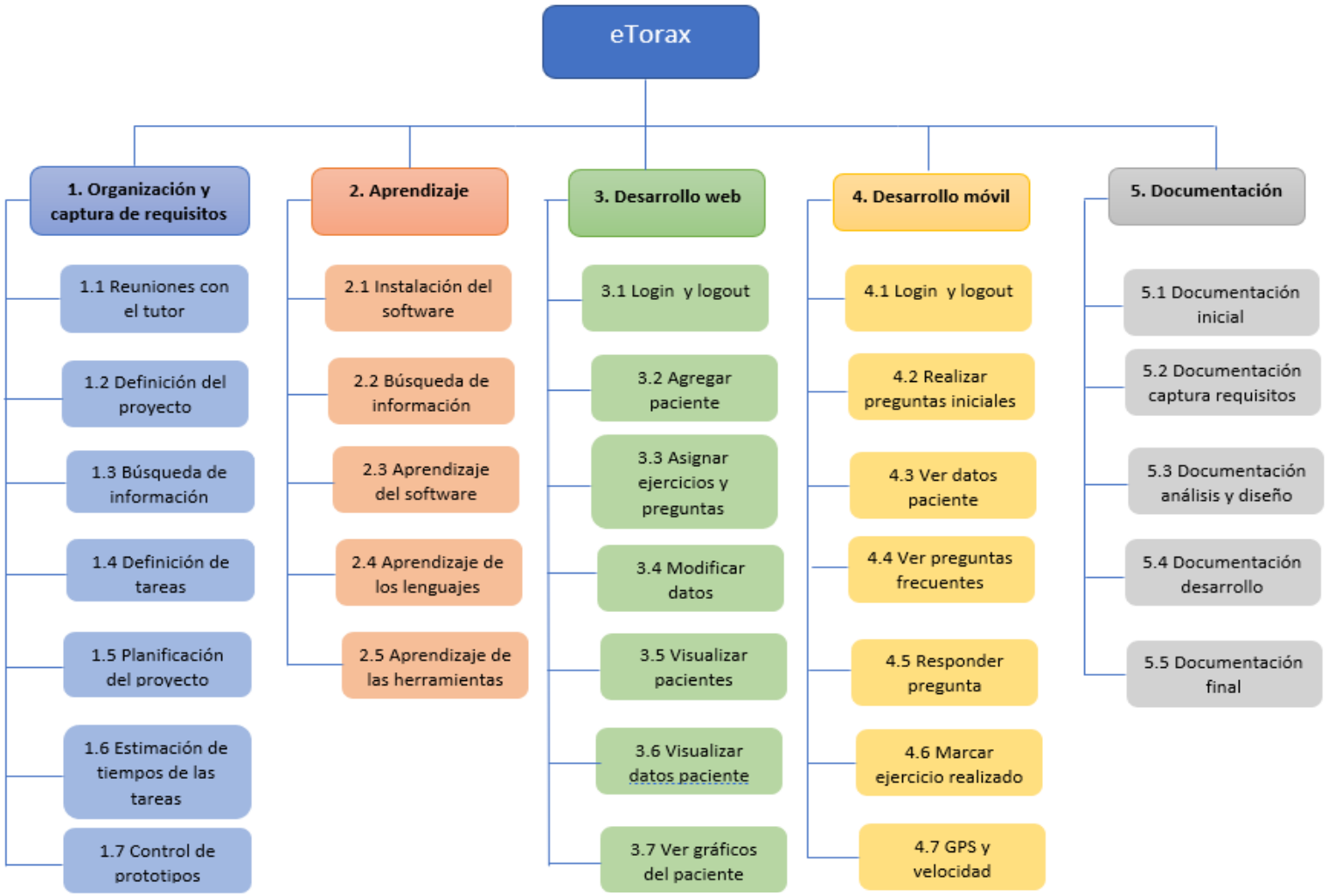


Ilustración 1 - Esquema EDT

En los siguientes puntos se realiza una breve explicación de la ilustración mostrada anteriormente y los campos que en ella aparecen.

1. Organización y captura de requisitos: Esta se trata de la primera fase que se realizará al comenzar el proyecto, sin embargo, en dicha fase también existen tareas que se deberán realizar a lo largo de todo el proyecto como podrían ser las “reuniones con el tutor”. Por lo demás se trata de una fase en la que se organizará el proyecto, se definirá y planificará el mismo.

2. Aprendizaje: En esta fase del proyecto se instalará todo el software que a priori será necesario para desarrollar el proyecto, se configurará el mismo y se aprenderá como usarlo. Al ser un proyecto en el que como se ha comentado anteriormente se van a utilizar herramientas y lenguajes desconocidos será una fase del proyecto a la que habrá que dedicarle un tiempo considerable.

3. Desarrollo web: Esta parte consiste en el desarrollo de la página web que posteriormente usará el equipo médico. Incluye el trabajo de análisis, diseño, implementación y las pruebas.

4. Desarrollo de la aplicación móvil: En esta fase se desarrollará la aplicación móvil, se desarrollará para todos los sistemas operativos al mismo tiempo. Incluye el trabajo de análisis, diseño, implementación y la realización de las pruebas.

5. Documentación: En este módulo se encuentran las tareas destinadas a crear la documentación del proyecto. al igual que el módulo de “Organización y captura de requisitos” será un módulo el cual se realizará a lo largo de todo el proyecto.

Organización y captura de requisitos

Tabla 1 - Tarea: Reuniones con el tutor

1.1 Reuniones con el tutor
Duración: 6 horas.
Descripción: Tanto la reunión inicial como las reuniones de control que sucederán a lo largo del proyecto. En estas reuniones se informará del progreso del proyecto, así como se consultarán las dudas surgidas a lo largo del desarrollo.
Entradas: Ninguna.
Salidas: Información de los avances y dudas resueltas.

Tabla 2 - Tarea: Definición del proyecto

1.2 Definición del proyecto
Duración: 12 horas.
Descripción: Se definirá como serán tanto la página web como la aplicación móvil, así como las funcionalidades que dispondrán.
Entradas: Información de las necesidades y reuniones con el cliente.
Salidas: Disponer de información precisa de cómo debe ser el proyecto.

Tabla 3 - Tarea: Búsqueda de información

1.3 Búsqueda de información
Duración: 3 horas.
Descripción: Búsqueda de información general de las herramientas con las que se realizará el proyecto, así como de la arquitectura del mismo.
Entradas: Requisitos del cliente.
Salidas: Tener definido cuáles serán las herramientas a utilizar y la arquitectura**.**

Tabla 4 - Tarea: Definición de tareas

1.4 Definición de tareas
Duración: 6 horas.
Descripción: Definir las tareas necesarias para el desarrollo del proyecto.
Entradas: Definición del proyecto.
Salidas: Tareas del proyecto definidas y documentadas.

Tabla 5 - Tarea: Panificación del proyecto

1.5 Planificación del proyecto
Duración: 4 horas.
Descripción: Planificación de las tareas y elección del orden en que se realizarán.
Entradas: Definición de tareas.
Salidas: Planificación de tareas.

Tabla 6 - Tarea: Estimación de tiempos

1.6 Estimación de tiempos
Duración: 2 horas.
Descripción: Definir el tiempo necesario para realizar cada una de las tareas definidas anteriormente**.**
Entradas: Planificación del proyecto.
Salidas: Planificación temporal de las salidas.

Tabla 7 - Tarea: Control de prototipos

1.7 Control de prototipos
Duración: 5 horas.
Descripción: Control para comprobar que los prototipos cumplen con las especificaciones**.**
Entradas: Prototipos.
Salidas: Prototipos revisados.

Aprendizaje

Tabla 8 - Tarea: Instalación de software

2.1 Instalación del software
Duración: 6 horas.
Descripción: Instalar todas las herramientas que se van a utilizar a lo largo del proyecto. Esta tarea incluye la descarga de los programas y la configuración de los mismos.
Entradas: Software y herramientas a utilizar**.**
Salidas: Herramientas instaladas y configuradas de forma correcta.

Tabla 9 - Tarea: Búsqueda de información

2.2 Búsqueda de información
Duración: 10 horas
Descripción: Búsqueda de información de las herramientas y software que se va a utilizar.
Entradas: Herramienta y software a utilizar.
Salidas: Mayor conocimiento de las herramientas y software que se utilizarán.

Tabla 10 - Tarea: Aprendizaje del software

2.3 Aprendizaje del software
Duración: 20 horas.
Descripción: Aprendizaje del nuevo software que se utilizará durante el proyecto, en esta tarea no se incluye el aprendizaje de nuevos lenguajes de programación ya que se ha realizado aprendizaje de software con lenguajes ya conocidos.
Entradas: Herramientas y software a utilizar.
Salidas: Conocimiento del software a utilizar durante el proyecto.

Tabla 11 - Tarea: Aprendizaje de los nuevos lenguajes

2.4 Aprendizaje de los lenguajes nuevos
Duración: 10 horas.
Descripción: Aprendizaje de lenguajes nunca utilizados por el desarrollador.
Entradas: Herramientas y software a utilizar.
Salidas: Conocimiento de los nuevos lenguajes

Tabla 12 - Tarea: Aprendizaje de las herramientas

2.5 Aprendizaje de las herramientas
Duración: 5 horas.
Descripción: Aprendizaje de las herramientas y entornos de desarrollo que se utilizaran para desarrollar el proyecto.
Entradas: Herramientas a utilizar.
Salidas: Conocimientos sobre las herramientas y entornos de desarrollo.

Desarrollo web

Tabla 13 - Tarea: Login y Logout

3.1 Login y Logout
Duración: 10 horas.
Descripción: Desarrollar un inicio de sesión y un cerrar sesión para la página web.
Entradas: Requisitos del sistema**.**
Salidas: Funcionalidad de Login y Logout funcionando correctamente.

Tabla 14 - Tarea: Agregar paciente

3.2 Agregar paciente
Duración: 12 horas.
Descripción: Desarrollar la funcionalidad para que el médico pueda agregar un nuevo paciente.
Entradas: Requisitos del sistema.
Salidas: Funcionalidad que permita al médico agregar un nuevo paciente.

Tabla 15 - Tarea: Mostrar paciente

3.3 Mostrar pacientes
Duración: 25 horas.
Descripción: Desarrollar la funcionalidad que permita al médico mostrar la lista de pacientes y gestionarla.
Entradas: Requisitos del sistema.
Salidas: Funcionalidad que permite al médico mostrar la lista de pacientes y gestionarla de forma correcta**.**

Tabla 16 - Tarea: Mostrar datos paciente

3.4 Mostrar datos paciente
Duración: 25 horas**.**
Descripción: Desarrollar la funcionalidad que permita al médico ver los datos de un paciente y lo ejercicios asignados.
Entradas: Requisitos del proyecto.
Salidas: Funcionalidad que muestra al médico los datos de un paciente.

Tabla 17 - Tarea: Modificar datos paciente

3.5 Modificar datos paciente
Duración: 20 horas.
Descripción: Desarrollar la funcionalidad que permite al médico asignar ejercicios y pregunta a un paciente, así como modificar los anteriormente asignados.
Entradas: Requisitos del proyecto.
Salidas: Funcionalidad que permite al médico modificar los datos de un paciente.

Tabla 18: Tarea: Visualizar datos del paciente

3.6 Visualizar datos del paciente
Duración: 20 horas.
Descripción: Desarrollar la funcionalidad que permite al médico visualizar todos los datos del paciente.
Entradas: Requisitos del proyecto.
Salidas: Funcionalidad que permite al médico modificar los datos de un paciente.

Tabla 19: Tarea: Ver gráficos del paciente

3.7 Ver gráficos paciente
Duración: 16 horas.
Descripción: Desarrollar la funcionalidad que permite al médico asignar ejercicios y pregunta a un paciente, así como modificar los anteriormente asignados.
Entradas: Requisitos del proyecto.
Salidas: Funcionalidad que permite al médico modificar los datos de un paciente.

Desarrollo App móvil

Tabla 20 - Tarea: Login y Logout

4.1 Login y Logout
Duración: 10 horas.
Descripción: Desarrollar la funcionalidad que permite al paciente iniciar sesión y cerrar sesión en la aplicación móvil.
Entradas: Requisitos del proyecto.
Salidas: Funcionalidad que permite al paciente iniciar y cerrar sesión de manera correcta.

Tabla 21 - Tarea: Ver ejercicios asignados

4.2 Ver ejercicios asignados
Duración: 15 horas.
Descripción: Desarrollar la funcionalidad que permite al paciente ver los ejercicios que le han sido asignados por su médico.
Entradas: Requisitos del proyecto.
Salidas: Funcionalidad que permite al paciente ver los ejercicios que su médico le ha asignado.

Tabla 22 - Tarea: Ver datos paciente

4.3 Ver datos paciente
Duración: 4 horas.
Descripción: Desarrollar la funcionalidad que permita al paciente ver sus propios datos y modificar su contraseña.
Entradas: Requisitos del sistema.
Salidas: Funcionalidad que permite al paciente ver sus datos.

Tabla 23 - Tarea: Ver preguntas frecuentes

4.4 Ver preguntas frecuentes
Duración: 5 horas.
Descripción: Desarrollar la funcionalidad que permita al paciente ver las preguntas frecuentes.
Entradas: Requisitos del proyecto.
Salidas: Funcionalidad que permite al paciente ver las preguntas frecuentes.

Tabla 24 - Tarea: Responder pregunta

4.5 Responder pregunta
Duración: 8 horas.
Descripción: Desarrollar la funcionalidad que permita al paciente recibir las notificaciones de ejercicios y de pregunta además de poder responder la pregunta del médico.
Entradas: Requisitos del proyecto.
Salidas: Funcionalidad que permite al paciente recibir notificaciones además de responder a la pregunta del médico.

Tabla 25 - Tarea: Marcar ejercicio como realizado

4.6 Marcar ejercicio como realizado
Duración: 10 horas
Descripción: Desarrollar la funcionalidad que permita al cliente marcar cuando ha realizado un ejercicio.
Entradas: Requisitos del proyecto.
Salidas: Funcionalidad que permite al paciente marcar un ejercicio como realizado.

Tabla 26 - Tarea: Control de ejercicio caminar

4.7 Control de ejercicio caminar
Duración: 25 horas.
Descripción: Desarrollar la funcionalidad que controla la distancia recorrida por el paciente además de la velocidad a la que la recorre.
Entradas: Requisitos del proyecto.
Salidas: Funcionalidad para controlar los metros recorridos por el paciente y a la velocidad a la que los ha recorrido.

Documentación

Tabla 27 - Tarea: Documentación inicial

5.1 Documentación inicial
Duración: 10 horas.
Descripción: Documentación referente a toda la parte inicial del proyecto como la introducción, el planteamiento inicial y los antecedentes.
Entradas: Ninguna
Salidas: Documentación inicial del proyecto.

Tabla 28 - Tarea: Documentación de captura de requisitos

5.2 Documentación de captura de requisitos
Duración: 20 horas.
Descripción: Documentación referente a los requisitos que el proyecto debe cumplir, quienes serán los usuarios finales, etc.
Entradas: Requisitos del proyecto.
Salidas: Documentación de la captura de requisitos.

Tabla 29 - Tarea: Documentación de análisis y diseño

5.3 Documentación de análisis y diseño
Duración: 17 horas.
Descripción: Documentación referente a como se plantea realizar el proyecto. Partes en las que se divide y explicación de cada una de ellas.
Entradas: Documentación de la captura de requisitos
Salidas: Documentación del análisis y el diseño.

Tabla 30 - Tarea: Documentación del desarrollo

5.4 Documentación del desarrollo
Duración: 27 horas.
Descripción: Documentación referente al desarrollo del proyecto, que se ha realizado, como se ha realizado, etc.…
Entradas: Desarrollo del proyecto.
Salidas: Documentación del desarrollo del proyecto.

Tabla 31 - Tarea: Documentación final

5.5 Documentación final
Duración: 16 horas.
Descripción: Documentación referente a la parte final del proyecto. Se documentarán tanto las pruebas realizadas como las conclusiones del proyecto.
Entradas: Pruebas realizadas.
Salidas: Documentación final.

2.4. Planificación temporal

Una vez tenemos definidas las tereas que conformarán el proyecto y una duración estimada de las mismas el siguiente paso será realizar la planificación temporal. De esta manera podremos realizar una estimación de cuando se finalizará el proyecto.

Para mostrar la planificación de realizará un diagrama Gantt distribuido de forma semanal.

Para la realización del proyecto se ha estimado una carga diaria mínima de 2 horas los días en que se tenga que compaginar la realización del proyecto con las prácticas de empleo. La carga de los días en los que no se tengan que realizar practica se estima en unas 4 horas. De esta manera se pretende llegar a las 20 horas semanales invertidas en la realización del proyecto. En caso de no ser suficientes se aumentaría la cantidad de horas semanales invertidas.

Para la realización del diagrama Gantt se ha seguido la distribución de tareas realizada en el apartado anterior (Estructura de descomposición de trabajo EDT): organización y captura de requisitos, aprendizaje, desarrollo web, desarrollo de la app de móvil y documentación. Estas tareas se diferenciarán por colores y se situarán en la parte izquierda. Por otra parte, en la parte superior del diagrama aparecerá la distribución semanal.

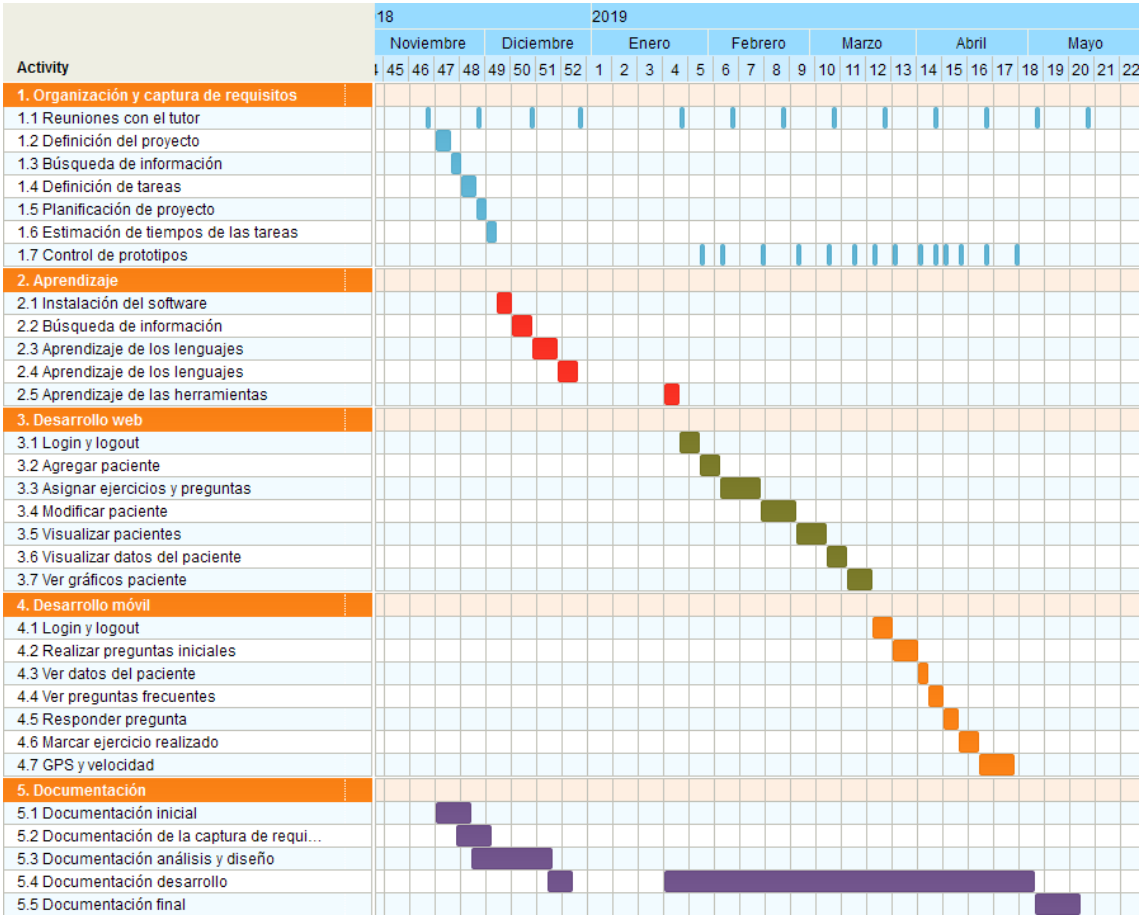


Ilustración 2 - Diagrama de Gantt

Tabla 32 - Planificación temporal

Paquete de trabajo	Tarea	Estimación de horas dedicadas	Total, estimación
Organización y captura de requisitos	Reuniones con el tutor.	6	38
	Definición del proyecto.	12	
	Búsqueda de información.	3	
	Definición de tareas.	6	
	Planificación del proyecto.	4	
	Estimación de tiempos de las tareas.	2	
	Control de prototipos.	5	
Aprendizaje	Instalación del software	6	51
	Búsqueda de información.	10	
	Aprendizaje del software.	20	
	Aprendizaje de los lenguajes.	10	
	Aprendizaje de las herramientas.	5	
Desarrollo web	Login y Logout.	10	118
	Agregar paciente.	12	
	Mostrar pacientes.	25	
	Mostrará datos paciente.	25	
	Modificar datos paciente.	20	
	Visualizar datos del paciente	20	
	Ver gráficos paciente	10	
Desarrollo app móvil	Login y Logout.	10	77
	Ver ejercicios asignados	15	
	Ver datos paciente.	4	
	Ver preguntas frecuentes	5	
	Responder pregunta	8	
	Marcar ejercicio realizado	10	
	Control de caminar	25	
Documentación	Documentación inicial	10	90
	Documentación captura de requisitos	20	
	Documentación análisis y diseño.	17	
	Documentación desarrollo	27	
	Documentación final.	16	
Total			374

2.5. Arquitectura

Para la realización de eTorax desde un primer momento de optó por una arquitectura de tipo REST (Representacional State Transfer). Esta arquitectura se trata de una tecnología muy flexible la cual comunica los datos a través del protocolo HTTP (Hypertext Transfer Protocol). HTTP es capaz de comunicar los datos en diferentes formatos como podrían ser HTML, XML o JSON, este último es el que se utilizará en este proyecto ya que JavaScript resulta muy fácil manejar este tipo de datos puesto que es interpretado de forma natural por JavaScript.

La arquitectura REST utiliza el protocolo cliente servidor sin estado, lo cual quiere decir que en las peticiones realizadas mediante HTTP se enviará toda la información necesaria para que el servidor pueda realizar las operaciones pertinentes. Las peticiones del cliente en HTTP se realizan utilizando diferentes métodos (POST, GET, DELETE, PUT) y una vez realizada la petición del cliente este esperará la respuesta del servidor, el cual responde a través de códigos de respuesta además de datos (2XX: Petición correcta, 3XX: Redireccionamiento, 4XX: Error del Cliente, 5XX: Error del servidor).

Esto hace que el cliente y el servidor se mantengan completamente separados, con las ventajas que ello supone.

De este modo la aplicación se desarrolla dividida en el back-end y el front-end. La parte de front-end del proyecto se desarrollará en el framework Angular para la parte de la página web y para la parte de aplicación se utilizará Xamarin. Para el desarrollo del back-end se utilizará NodeJS como framework junto con Express. El front-end formado tanto por la aplicación móvil como por la aplicación de teléfono realizará las peticiones al back-end el cual será el encargado de procesar dichas peticiones, responderlas y comunicarse con el servidor.

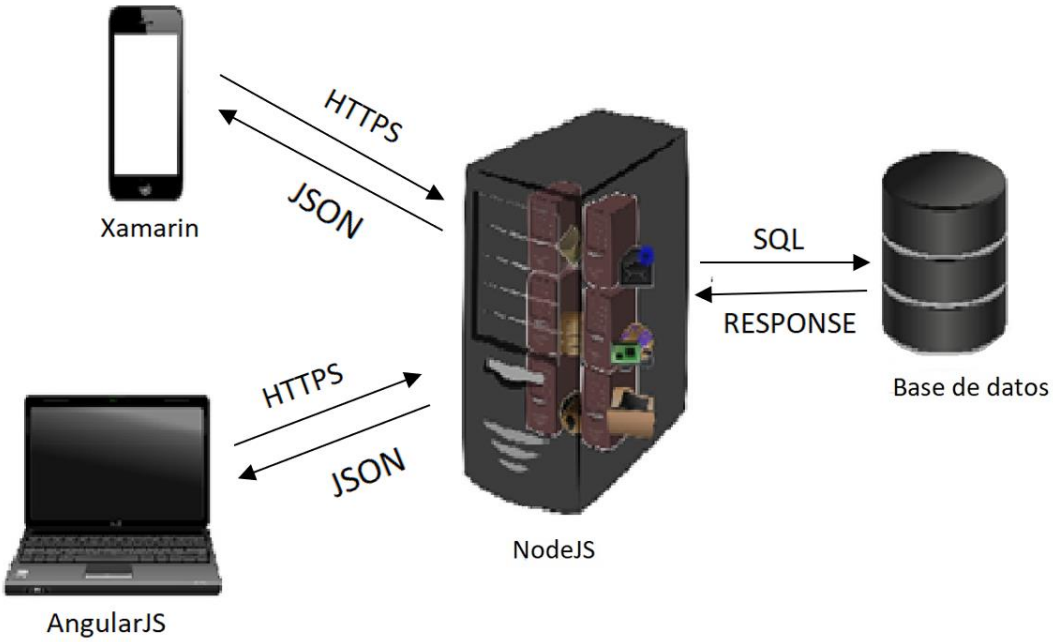


Ilustración 3 - Arquitectura

2.6. Herramientas:

En este apartado se detallarán cuáles han sido las herramientas utilizadas durante la realización de este trabajo de fin de grado. Para cada una de las herramientas se explicará para que sirve y para qué ha sido utilizada.

- **Visual Studio Code:** Se trata de un editor de código fuente el cual está desarrollado por Microsoft. Se trata de un editor de código bastante ligero pero que incluye herramientas potentes para el desarrollo como el sistema de depuración de código. En Visual Studio Code se desarrollará el front-end de la aplicación web.

- **Visual Studio:** Se trata de un entorno de desarrollo integrado en cual al igual que Visual Studio Code está desarrollado por Microsoft. En este caso Visual Studio se utilizará para desarrollar la aplicación móvil en Xamarin.

- **Justinmind:** Se trata de una herramienta la cual es capaz de crear prototipos de aplicaciones web y aplicaciones móviles de forma sencilla. Justinmind se usará para crear los prototipos de la aplicación para los dispositivos móviles y la página web, para que el equipo médico vea cual es la forma que el desarrollador tiene pensado darles a las aplicaciones y se pueda llegar a un punto común antes de comenzar el desarrollo.

- **Sublime Text:** Se trata de un editor de texto y editor de código fuente el cual está escrito en C++. Sublime Text se usará para realizar el desarrollo del back-end del proyecto.

- **NodeJS:** "Es un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node está diseñado para construir aplicaciones en red escalables" (NodeJS, s.f.). NodeJS será utilizado para llevar a cabo la lógica de la aplicación en la parte del servidor.

- **Express:** Es un framework para Node.js el cual fue lanzado como software libre de código abierto. Express es un framework para JavaScript el cual está diseñado para realizar APIs. Este framework permite realizar aplicaciones web de manera sencilla.

- **AngularJS:** Se trata de un framework de JavaScript de código abierto. Google es quien mantiene dicho framework. Angular se utiliza para crear aplicaciones web de una sola página. AngularJS se utilizará para crear el front-end de la página web de eTorax.

- **TypeScript:** Es el lenguaje utilizado en angular. Se trata de un lenguaje de código abierto el cual está orientado a la programación orientada a objetos. Los navegadores son capaces de interpretar TypeScript. Se trata de un lenguaje muy similar a JavaScript.

- **Xamarin:** Xamarin es una plataforma de desarrollo que permite desarrollar aplicaciones multiplataforma. Xamarin permite desarrollar aplicaciones que funcionen en cualquier dispositivo móvil con el mismo código, escrito en el lenguaje de programación C#. (Imaginaformacion, s.f.) Xamarin se utilizará para desarrollar la aplicación móvil en todo tipo de dispositivos móviles al mismo tiempo.

- **C#:** Se trata de un lenguaje de programación orientado a objetos el cual está desarrollado por Microsoft como parte de su plataforma .NET. C# es el lenguaje utilizado por Xamarin de modo que en eTorax se utilizará en el desarrollo de las aplicaciones móviles.

- **HTML:** HTML es el acrónimo de "HyperText Markup Language". Se trata de un lenguaje de programación el cual se utiliza para desarrollar páginas web. En específico HTML permite desarrollar la parte grafica del front-end de una página web. En eTorax HTML se usará para crear el apartado grafico de la página web.

- **CSS:** Las siglas de CSS en ingles significan "Cascading Style Sheets". Se trata de un lenguaje destinado a crear el diseño gráfico. CSS se complementa con HTML y le aporta un estilo.

- **JavaScript:** Se trata de un lenguaje de programación orientado a objetos el cual es utilizado generalmente para la creación de páginas web dinámicas.

- **SQL:** Sus siglas en ingles significan “Structured Query Language”. Es un lenguaje el cual está diseñado para administrar y recuperar información de sistemas de bases de datos relacionales.

- **Navicat:** Es una herramienta de desarrollo de base de datos. En eTorax se utilizará para crear la base de datos.

- **Heroku:** Heroku es uno de los PaaS más utilizados en la actualidad. Además, permite manejar los servidores y sus configuraciones, escalamiento y la administración. (ricardocelis, 2017). En eTorax Heroku se utilizará para subir a un servidor tanto el front-end como el back-end de la aplicación, así como la base de datos. De este modo la aplicación será accesible para todo el mundo y el equipo médico podrá probarla.

- **Google Drive:** Es el servidor de alojamiento de archivos de Google. Será utilizado para almacenar las diferentes copias de seguridad creadas a lo largo del proyecto

- **Google Chrome:** Se trata del navegador web creado por Google. El navegador se utilizará para probar la aplicación web, así como para buscar la información necesario para realizar el proyecto.

- **Microsoft Office Word:** Se trata de un programa orientado al procesamiento de textos creado por Microsoft. Microsoft Office Word se utilizará para crear la memoria del proyecto.

2.7. Gestión de riesgos

En este apartado se valorará los riesgos que pudieran surgir a lo largo del desarrollo del proyecto. Pese a que los riesgos que pudieran surgir sean de diferente gravedad es importante documentarlos y desarrollar un plan de prevención para cada uno de ellos independientemente de la gravedad que supondría el suceso de cada uno de los riesgos. De esta forma se podrán buscar maneras para minimizar la probabilidad de que los riesgos sucedan y de suceder alguno de ellos, que el impacto que este tuviese sobre el proyecto fuese el menor posible. Además, la gestión de riesgos nos permitirá saber cómo deberemos reaccionar si uno de los riesgos documentados sucede.

La probabilidad de que suceda una amenaza será clasificada según la siguiente tabla:

Tabla 33 - Probabilidad de los riesgos

Probabilidad	Porcentaje
Improbable	0%-25%
Poco probable	26%-50%
Probable	51%-75%
Muy probable	76%-100%

Además de la probabilidad de que un riesgo suceda también es necesario analizar el impacto que este tendría en el proyecto. El impacto del riesgo se ha dividido en cinco: muy leve, leve, medio, grave y muy grave dependiendo del retraso que causaría en el proyecto. Al igual que para la probabilidad de los riesgos también se ha definido una tabla para el impacto de los mismos.

Tabla 34 - Impacto de los riesgos

Impacto	Días
Muy leve	<1 (Horas)
Leve	1
Medio	2-3
Grave	3-7
Muy grave	>7

Lo primero que se debe realizar en la gestión de riesgos es identificar los posibles riegos del proyecto.

Tras identificar los riesgos se pasará a analizarlos uno a uno, para de esta manera poder asignarles la probabilidad de que suceda el riesgo y el impacto que este tendría si sucediese. Además de ello se describirá brevemente en que consiste el riesgo, se realizará un plan de prevención contra el riesgo para que de esta manera se disminuya lo máximo posible la probabilidad de que el riesgo suceda y se realizará un plan de contingencia para que si pese a él plan de prevención el riesgo sucediese se sepa cómo actuar para minimizar los efectos del mismo.

Tabla 35 - Riesgo: Enfermedad

Enfermedad.	
Descripción	Contraer alguna enfermedad durante el desarrollo de proyecto.
Plan de prevención	Extremar las precauciones para con contraer ninguna enfermedad además de intentar llevar una alimentación sana y tener bajos niveles de estrés.
Plan de contingencia	Tomar medidas para lograr una pronta recuperación y reorganizar la planificación temporal.
Riesgo	Muy probable.
Impacto	Leve.

Tabla 36 - Riesgo: Problemas familiares o sociales

Problemas familiares o sociales.	
Descripción	Problemas ajenos a mi persona.
Plan de prevención	Mantener un contacto continuo con las personas cercanas.
Plan de contingencia	Intentar solucionar el problema surgido a la mayor brevedad posible.
Riesgo	Poco probable.
Impacto	Muy leve.

Tabla 37 - Riesgo: Pérdida o robo del ordenador

Pérdida o robo del ordenador.	
Descripción	Pérdida o robo del ordenador portátil utilizado para el desarrollo del proyecto.
Plan de prevención	Extremar las precauciones e intentar sacarlo lo menos posible de casa para así la posibilidad de pérdida o robo. Además de ello realizarán copias de seguridad externas de forma periódica tanto online como offline.
Plan de contingencia	Utilizar el equipo de sobremesa intentando recuperar todos los datos posibles desde las copias de seguridad realizadas.
Riesgo	Improbable.
Impacto	Grave.

Tabla 38 - Riesgo: Rotura del ordenador

Rotura del ordenador	
Descripción	Rotura del ordenador portátil utilizado para el desarrollo del proyecto
Plan de prevención	Extremar las precauciones, no usar el ordenador con bebidas cerca e intentar sacarlo lo menos posible de casa para así evitar golpes o caídas. Además de ello realizar copias de seguridad externas de forma periódica tanto online como offline.
Plan de contingencia	Utilizar el equipo de sobremesa intentando recuperar todos los datos posibles desde las copias de seguridad realizadas.
Riesgo	Poco probable.
Impacto	Grave.

Tabla 39 - Riesgo: Rotura del móvil

Rotura del móvil	
Descripción	Rotura del dispositivo móvil utilizado para el desarrollo del proyecto.
Plan de prevención	Extremar las precauciones, utilizar fundas protectoras y protectores de pantalla.
Plan de contingencia	Utilizar otro dispositivo móvil que sirva como sustitución.
Riesgo	Poco probable.
Impacto	Muy leve.

Tabla 40 - Riesgo: Pérdida o robo del móvil

Pérdida o robo de móvil	
Descripción	Pérdida o robo del dispositivo móvil utilizado para el desarrollo del proyecto.
Plan de prevención	Extremar las precauciones, mantener el móvil siempre localizado e intentar evitar dejarlo descuidado en lugares públicos
Plan de contingencia	Utilizar otro dispositivo móvil que sirva como sustitución.
Riesgo	Poco probable.
Impacto	Muy leve.

Tabla 41 - Riesgo: Caída del servidor de prueba

Caída del servidor de prueba	
Descripción	Caída del servidor en el que se hacen las pruebas de Angular y la el API-REST.
Plan de prevención	Utilizar un servidor fiable y robusto.
Plan de contingencia	Mientras dure la caída realizar las pruebas en local.
Riesgo	Improbable.
Impacto	Muy leve.

Tabla 42 - Riesgo: Equipo infectado por malware

Equipo infectado por malware	
Descripción	Infección de malware en cualquiera de los dispositivos utilizados para el desarrollo del proyecto.
Plan de prevención	Realizar análisis completos del sistema de forma periódica y mantener el antivirus actualizado, así como el sistema operativo. No entrar en páginas Web sospechosas de contener algún tipo de malware ni realizar descargas de sitios poco seguros. Realizar copias de seguridad externas de forma periódica tanto online como offline.
Plan de contingencia	Eliminar el malware del equipo si fuese posible y restaurar a la última copia de seguridad del proyecto.
Riesgo	Improbable
Impacto	Grave.

Tabla 43 - Riesgo: Mala planificación temporal

Mala planificación temporal	
Descripción	La planificación temporal del proyecto es incorrecta.
Plan de prevención	Intentar realizar una planificación temporal realista y precisa. Revisar la planificación temporal de forma periódica.
Plan de contingencia	Reajustar la planificación temporal e intentar recuperar el tiempo de retraso.
Riesgo	Probable.
Impacto	Medio.

Tabla 44 - Riesgo: Estancamiento

Estancamiento	
Descripción	Llegar a un punto del proyecto en el que no saber cómo seguir avanzando.
Plan de prevención	Documentarse antes de comenzar a realizar cualquier tarea del proyecto.
Plan de contingencia	Realizar un pequeño descanso y buscar otras maneras para seguir avanzando.
Riesgo	Probable.
impacto	Medio.

Tabla 45 - Riesgo: Exámenes y practicas

Exámenes y practicas	
Descripción	Desarrollo de las prácticas de empresa y preparación para los exámenes finales.
Plan de prevención	Realizar una cantidad de horas de prácticas que permita no dejar de lado el proyecto. Realizar una planificación temporal teniendo la cuenta el tiempo invertido en las prácticas y exámenes.
Plan de contingencia	Reorganizar la planificación para poder recuperar el tiempo perdido.
Riesgo	Muy probable.
impacto	Medio.

Tabla 46 - Riesgo: Cambio de los requisitos del proyecto

Cambio de requisitos del proyecto	
Descripción	Los requisitos del proyecto cambian durante el desarrollo.
Plan de prevención	Mantener una comunicación lo más fluida posible tanto con el tutor como con el cliente y realizar un proyecto lo más modular posible para que los cambios se puedan realizar de una forma sencilla y no afecten a las funciones ya implementadas.
Plan de contingencia	Replanteamiento de las especificaciones modificadas.
Riesgo	Probable.
impacto	Grave.

Tabla 47 - Riesgo: Problemas de diseño

Problemas de diseño	
Descripción	Realizar un mal diseño del proyecto que ocasionaría retrasos en el mismo.
Plan de prevención	Dedicar el tiempo suficiente a la fase de diseño para no cometer errores.
Plan de contingencia	Rediseñar la parte en la que exista el problema intentando conservar lo realizado hasta el momento.
Riesgo	Poco probable.
impacto	Grave.

Tabla 48 - Riesgo: Perdida de información

Perdida de información	
Descripción	Perder información del proyecto ya sea por borrarla por descuido, sobrescribirla o por fallos del dispositivo.
Plan de prevención	Realizar copias de seguridad periódicas tanto online como offline.
Plan de contingencia	Recuperar los datos de la última copia de seguridad.
Riesgo	Probable.
Impacto	Grave.

Tras describir los riesgos, crear un plan de prevención y contingencia, definir la probabilidad y el impacto de cada uno de ellos el siguiente paso será definir la periodicidad con la que se revisará cada uno de ellos.

En la siguiente tabla se muestran cada uno de los riesgos definidos anteriormente y en función de la probabilidad del riesgo y el impacto que supondría se ha definido cada cuanto se revisaría el riesgo.

Tabla 49 - Revisión de riesgos

Riesgo	Probabilidad	Impacto	Valoración
Enfermedad	Muy probable	Leve	Leve
Problemas familiares o sociales	Poco probable	Muy leve	Leve
Ordenador roto	Poco probable	Grave	Medio
Ordenador perdido o robado	Improbable	Grave	Medio
Móvil roto	Poco probable	Muy leve	Leve
Móvil perdido o robado	Poco probable	Muy leve	Leve
Caída del servidor de pruebas	Improbable	Muy leve	Leve
Equipo infectado por malware	Improbable	Grave	Leve
Problemas de diseño	Poco probable	Grave	Medio
Mala planificación temporal	Probable	Medio	Grave
Estancamiento	Probable	Medio	leve
Exámenes o practicas	Muy probable	Medio	Medio
Cambio de requisitos del proyecto	Probable	Grave	Grave

Tabla 50 - Periodicidad de la revisión

Revisión	Valoración
Cada semana	Grave
Cada 15 días	Medio
Cada mes	Leve

2.8. Evaluación económica

En este apartado se calculará el coste total que supondría al desarrollo del proyecto si fuese lucrativo. El coste completo del proyecto se calculará mediante la suma de varios costes como podrían ser el coste de mano de obra, el coste del equipo o el coste de las licencias de software entre otros.

2.8.1. Mano de obra

El coste de la mano de obra del proyecto se calculará en base a el número de horas necesarias calculado en los apartados anteriores. Según mi propia experiencia un programador junior cobra sobre unos 20.000 euros anuales trabajando a jornada completa, sin embargo, esto no es lo que le cuesta el trabajador a la empresa. Como norma general las empresas pagan en impuestos un 30% (La información, 2019) del salario bruto del trabajador, por lo que la empresa debería pagar 26000€ anuales lo cual se traduce en un coste de 14,73 euros por hora trabajada. Teniendo en cuenta que en el apartado 2.4 se ha calculado en la planificación temporal que el tiempo necesario para realizar el proyecto será de unas 374 horas el coste de mano de obra del proyecto será de 5.509,02€.

2.8.2. Equipo

El equipo utilizado para la realización del proyecto está formado por un ordenador portátil y un smartphone. El portátil utilizado ha sido un Acer Aspire E15 E5-571-51S6 con un costo de 400€ el cual incluía el sistema operativo Windows 8.1 por lo que la licencia del sistema operativo no se tendrá en cuanta en el siguiente apartado. En dispositivo móvil utilizado ha sido un Xiaomi Redmi Note 4 con un costo de 150€.

Tabla 51 - Costo del equipo

Coste del equipo			
Dispositivo	Precio	Amortización	Coste
Acer Aspire E15 E5-571-51S6	400€	5 años	40€
Xiaomi Redmi Note	150€	3 años	25€
Total:			65€

2.8.3. Licencias de software

Para la realización del proyecto se han utilizado muchas herramientas cuyo uso es gratuito, sin embargo, algunas de las herramientas utilizadas requieren de la compra de una licencia para su utilización.

Tabla 52 - Costo de las licencias de software

Licencias de software			
Software	Precio	Meses	Coste
Microsoft office	100€/año	6	50€
Visual paradigm standar	308€/año	6	154€
Total:			204€

2.8.4. Otros costes

Además de las licencias, el hardware y la mano de obra durante el desarrollo del proyecto surgirán otros gastos como los gastos de internet o desplazamientos para las reuniones.

Los gastos de internet será 35€ al mes durante los 6 meses del proyecto, lo cual supone un total de 315€.

Los gastos por desplazamiento para las reuniones son 1.78€ por viaje, se harán reuniones cada 2 semana lo cual supone un total de 4 viajes al mes. En total el coste de los viajes será de 64€.

2.8.5. Coste total

Tras analizar los costes uno por uno ahora se puede calculas el coste total del proyecto, como se puede ver en la siguiente tabla.

Tabla 53 - Coste total del proyecto

Coste total	
Mano de obra	5.509,02€
Hardware	65€
Licencias	204€
Otros gastos	379€
Total:	6157.02€

Este proyecto se trata de un proyecto no lucrativo, el cual no está enfocado a la obtención de beneficios, sino que a ayudar a otras personas dentro de lo posible. Por lo tanto, en este caso, no aplica hablar de generar beneficios económicos con el proyecto.

3. Antecedentes

A la hora de realizar un proyecto de esta magnitud es importante, antes de comenzar, estudiar las alternativas que hay en el mercado y que es lo que ofrecen dichas alternativas. De esta manera también es importante estudiar las diferentes tecnologías que se podrían usar a la hora de realizar el proyecto.

3.1. Descripción de la situación actual

En este caso no existen aplicaciones similares en el mercado. La aplicación que se va a realizar esta hecha a medida de los requisitos exigidos por el cliente y no existe ninguna aplicación que satisfaga dichos requisitos.

Actualmente las funciones que haría eTorax se realizan mediante papel. Los médicos dan a los pacientes unas hojas con información genérica de cuáles son las pautas a seguir por el paciente. Los médicos modifican las pautas genéricas de las hojas para adaptarlas lo más posible a las necesidades del paciente.

A día de hoy el equipo médico no tiene ninguna forma de saber si el paciente está siguiendo las pautas indicadas en las hojas. La única información que obtiene es la que recibe por el propio paciente a lo largo de las consultas, la cual podría no ser cierta.

Además de ello el paciente no dispone de una forma para saber si ha recorrido la distancia recomendada por el médico. El paciente podría estar recorriendo una distancia errónea sin saberlo con los riesgos que esto supondría.

Los médicos recomiendan a los pacientes una velocidad a la que realizar la distancia recomendada, sin embargo, el paciente a día de hoy no dispone de una forma con la que comprobar si la velocidad a la que está caminando es la correcta.

3.2. Estudio de las alternativas existentes.

En este apartado se analizarán las distintas alternativas que existen para realizar las diferentes partes que conforman el proyecto y se decidirá cual se trata de la mejor alternativa posible.

3.2.1. Aplicación web

Para la realización de la aplicación web se necesitan tecnologías capaces de crear una interfaz de usuario amigable y comunicarse con el servidor.

En cuanto a las aplicaciones web se diferencian en dos tipos, aplicaciones web estáticas y aplicaciones web dinámicas. Una de las principales características de las aplicaciones web estáticas es la ausencia de funcionalidades frente a la posibilidad de que el cliente altere el contenido de las páginas web dinámicas. Por ese motivo y debido a las necesidades del proyecto se ha optado por las aplicaciones web dinámicas.

También se ha decidido que la parte web de eTorax sea una SPA (Single page app) debido a que este tipo de aplicaciones son muy rápidas y ofrecen una comunicación cliente servidor muy fluida lo que hace que la experiencia de usuario sea más satisfactoria.

Tras decidir que la web se tratase de una aplicación web dinámica y además será una single page app pasaremos a analizar las herramientas que podremos usar para crear una web con estas características.

- AngularJS: Es un framework en JavaScript el cual esta mantenido por Google. Este framework se utiliza para la creación de SPA. AngularJS está enfocado a crear aplicaciones de navegador con capacidad de modelo vista controlador para que de esta manera sea más sencillo el desarrollo y las pruebas.

- Angular 2: Este framework al igual que el anterior también esta mantenida por Google y es de código abierto, sin embargo, esta desarrollado en TypeScript. Se utiliza para la creación de SPA. Este framework es la evolución de AngularJS, aunque estos dos framework no son retro compatibles. Otra de las diferencias respecto a su antecesor es que Angular 2 está orientado a móviles mientras que AngularJS no se hizo para soportar móviles. Angular 2 está orientado a componentes lo cual hace que la reutilización de código sea más sencilla.

- React: Se trata de una biblioteca en JavaScript de código abierto. React está mantenida por Facebook y la comunidad de software libre. Está diseñada para la creación de la interfaz de usuario de una SPA, sería la vista de un Modelo-Vista-Controlador. React está pensado para construir la interfaz de una aplicación en que los datos cambian todo el tiempo. Es importante señalizar que React se trata de una librería y no un framework a diferencia de Angular.

- EmberJS: Es un framework en JavaScript de código abierto el cual está basado en el Modelo-Vista-Modelo de vista para la creación de SPA. Se trata de un framework el cual utiliza de HTMLBars para la creación de plantillas.

Una vez analizadas todas las tecnologías disponibles se ha decidido utilizar Angular 2 ya que se trata de la evolución de AngularJS y cumple con todas las necesidades para desarrollar el proyecto. Además de que se trata de un framework muy utilizado y del cual hay mucha información disponible en internet.

3.2.2. Aplicación móvil

A continuación, se mostrarán las alternativas que existen para desarrollar la aplicación móvil. Para decantarme por una de ellas se tendrá en cuanta para que sistema operativo móvil se puede desarrollar en ellas así mismo también se tendrá en cuanta cual es el público objetivo de la aplicación.

- Android Studio: Es el framework oficial para la plataforma de Android, el cual es el sistema operativo más utilizado a día de hoy, además de utilizarse en otro tipo de dispositivos como televisores. (Zuriarrain, 2017) Pese a que Android es el sistema operativo más utilizado Android Studio no permite la creación de aplicaciones para otros sistemas operativos lo cual haría que si se quiere abarcar la mayoría de sistemas operativos halla que desarrollar la aplicación de forma específica para esos sistemas operativos también.

- Xamarin: Es un framework el cual permite el desarrollo de la aplicación de forma simultánea para varios sistemas operativos. Esto hará que la aplicación desarrollada sea multiplataforma sin tener que realizar varias aplicaciones, una para cada sistema operativo. Los lenguajes que utiliza Xamarin son C++ y .NET.

- Apache Cordova: Se trata de un framework que al igual que el anterior permite desarrollar aplicaciones multiplataforma. Sin embargo, a diferencia de Xamarin los lenguajes utilizados por Apache Cordova son CSS3, HTML5 y JavaScript. Esto hace que las aplicaciones creadas en Apache Cordova sean hibridas de tal modo que los distintos sistemas operativos comparten interfaz, pero tienen código nativo para cada sistema operativo.

Una vez analizadas las opciones disponibles se ha decidido utilizar Xamarin como framework de desarrollo ya que es necesario que la aplicación esté disponible para el mayor número de pacientes posible, de esta forma gracias a Xamarin se podrá desarrollar una aplicación multiplataforma que se pueda usar en varios sistemas operativos.

3.2.3. Back-end

A continuación, se especificarán cuáles han sido las opciones valoradas a la hora de escoger la tecnología que se usará para desarrollar el back-end. Se ha valorado el framework para el cual está diseñado y la sencillez de uso.

- Express: Es un framework para Node.js el cual fue lanzado como software libre de código abierto. Express es un framework para JavaScript el cual está diseñado para realizar APIs. Este framework permite realizar aplicaciones web de manera sencilla.

- Spring: Se trata de un framework de código abierto para java el cual permite desarrollar aplicaciones web. Este framework no impone ningún modelo de programación especifico y se ha vuelto muy popular entre los desarrolladores de java y como sustituto de Enterprise JavaBeans.

Para la realización del back-end se ha decidido utilizar el framework Express, ya que su uso es muy sencillo.

Además, al ser un framework para Node.js se podrá utilizar NPM ("Node.js package manager") el administrador de paquetes para Node.js el cual permite instalar paquetes de forma muy sencilla y cómoda. Gracias a estos paquetes disponemos de muchas funcionalidades ya desarrolladas, lo cual agilizará el avance del proyecto.

4. Captura de requisitos

En este apartado se presentará la captura de requisitos del proyecto. Este es uno de los puntos más importantes del proyecto, gracias a la captura de requisitos sabremos cuales son las necesidades que el proyecto tiene que satisfacer, quienes serán los usuarios del mismo y cuáles son las metas que tiene que cumplir.

Es necesario que la captura de requisitos del proyecto se haga de forma correcta para que el durante el desarrollo del mismo no surjan problemas.

4.1. Requisitos funcionales

Aquí será donde se definirán los requisitos funcionales que eTorax debe cumplir, es decir, las funcionalidades que las aplicaciones deben de ofrecer al usuario una vez el proyecto se ha finalizado.

Se presentarán tanto los requisitos funcionales de la aplicación web como los de la aplicación móvil, sin embargo, se comenzará con los de la página web:

- Solo se permitirá el acceso a la página web a los médicos. Estos accederán mediante un usuario y contraseña que compartirán. El registro de nuevos médicos no estará permitido. Se deberá disponer del usuario y la contraseña para poder acceder al sistema.

- La página web debe permitir visualizar todos los pacientes que se encuentran en el sistema, así mismo el usuario debe de ser capaz de ordenar la lista de pacientes en función de diferentes parámetros.

- Se debe permitir que el usuario realice búsquedas de los pacientes del sistema, con esta función el médico podría buscar un paciente en el sistema a través de varios parámetros como el número de historial clínico o su nombre y apellidos.

- Permitir la creación de nuevos pacientes. El sistema debe permitir que el médico cree nuevos pacientes en el sistema a través de la página web. Al momento de crear un paciente el sistema debe de ofrecer al médico la posibilidad de asignar al paciente los ejercicios que debe hacer, así como la longitud que debe caminar a diario y las preguntas que se le realizarán. Estos usuarios creados por el médico en este momento serán los que posteriormente aparecerán en el sistema y que utilizarán los pacientes desde sus aplicaciones móviles.

- Permitir la modificación de los ejercicios asignados a un paciente de la misma forma que también debe permitir la modificación de la distancia que ha de caminar a diario, de las preguntas que se le fueron asignadas y los ejercicios

que le fueron asignados junto con el número de repeticiones asignado a cada ejercicio.

- Permitir la visualización de los datos diarios del paciente. La aplicación web debe de dar acceso al médico a los datos diarios del paciente, de esta forma el médico puede comprobar si el paciente va cumpliendo con los objetivos propuestos por el médico, así como si este responde de forma satisfactoria a las preguntas asignadas al paciente.

- La aplicación debe permitir que el médico silencie los datos de un paciente. Tras un tiempo de recuperación tras la operación es posible que los médicos deseen no recibir más datos de un paciente, gracias a esta funcionalidad los médicos podrán dejar de recibir datos de un paciente y de esta forma no llenar la base de datos. Esta función no borrará los datos del paciente ni la cuenta del paciente, simplemente hará que el sistema no reciba nuevos datos del mismo. Además, el sistema también permitirá eliminar todos los datos de un paciente, así como la cuenta del mismo.

- Otra de las opciones que debe permitir la página web es ver los gráficos de los avances que van realizando los pacientes. La web deberá crear gráficos personalizados que muestren al médico cuales son los avances del paciente y el progreso antes y después de la operación.

Una vez descritos los requisitos funcionales que la aplicación web debe cumplir hay que mencionar cuáles serán los requisitos funcionales que la aplicación móvil debe cumplir:

- Solo podrán acceder a la aplicación móvil los usuarios que ya tengan una cuenta creada. Como se ha mencionado anteriormente esta cuenta debe ser creada por los médicos desde la página web. Darán al paciente su nombre de usuario y contraseña durante la consulta tras crear el usuario en el sistema.

- Permitir al paciente ver los ejercicios que tiene asignados por el médico. El paciente podrá ver un listado con los ejercicios que tiene asignados por el médico. Podrá ver el nombre del ejercicio y una pequeña descripción de cuál es la forma correcta de realizar el ejercicio. Además, una vez realizado el ejercicio el paciente podrá marcar que ha realizado el ejercicio, de esta manera dejará constancia en el sistema de cuáles son los ejercicios que ha realizado.

- La aplicación móvil deberá ser capaz de medir la distancia que el paciente ha caminado y la velocidad a la que lo ha realizado. Una vez finalizada la caminata enviará la información al sistema para que los datos sean almacenados.

- Debe permitir al paciente responder las preguntas asignadas por el médico. Una vez respondidas las preguntas asignadas la aplicación enviará la información al back-end para que esta sea almacenada y que sea accesible desde la página web.

- Permitir ver al paciente las preguntas más frecuentes que le podrían surgir a un paciente. De esta forma si el paciente tiene alguna duda es probable que la resuelva gracias a la sección de preguntas frecuentes.

- Permitir al paciente ver sus propios datos. La aplicación debe permitir que el paciente pueda consultar sus datos, además también debe permitir que el usuario cambie su contraseña ya que esta se la ha proporcionado el médico.

4.2. Requisitos funcionales no obligatorios

Aparte de los requisitos funcionales mencionados anteriormente también existen ciertos requisitos que podría cumplir el proyecto pero que no es totalmente necesario que los cumpla.

Primero se mencionará los requisitos funcionales secundarios de la página web:

- Permitir al médico ver la ruta que el paciente ha realizado. El móvil del paciente guardaría en el sistema los puntos por los que va caminando y posteriormente se le ofrecería al médico la opción de ver cuál ha sido el recorrido realizado por el paciente.

- Permitir ver que distancia a caminado el paciente a lo largo de las horas. Esta funcionalidad debería permitir que el médico pudiese ver cuánto ha caminado un paciente cada hora, ya que no es lo mismo caminar cinco kilómetros a lo largo del día que realizar una caminata de esa distancia.

Una vez mencionados los requisitos funcionales no obligatorios se presentarán los requisitos funcionales secundarios de la aplicación móvil:

- Permitir que el paciente vea en su ciudad que cumplan con las especificaciones asignadas por el médico. El paciente podría ver en el mapa recorridos cercanos que sean de una distancia similar a la distancia que el médico le ha asignado.

- Permitir que el paciente vea los recorridos que ha realizado anteriormente junto con la distancia y la duración del recorrido.

4.3. Casos de uso

Tras realizar la captura de requisitos del proyecto el siguiente paso es detectar y analizar los casos de uso, los cuales están estrechamente relacionados con los requisitos del sistema. Los casos de uso son una representación gráfica de las funcionalidades que ofrece el sistema y de cómo los actores deberían interactuar con el sistema para ejecutar dichas funcionalidades.

4.3.1. Jerarquía de actores

En este apartado se expondrán los actores que forman parte de cada una de las aplicaciones desarrolladas.

En primer lugar, se mostrará la jerarquía de actores de la página web.

Ilustración 4 - Jerarquía de actores de la página web.

Usuario no identificado: Representa a los usuarios que llegan a la página de inicio de sesión de la página web.

Médico identificado: Representa a los médicos una vez han iniciado sesión en la página web de forma satisfactoria.

A continuación, se mostrará el diagrama de la jerarquía de actores de la aplicación móvil, la cual al igual que la anterior solo consta de dos actores.

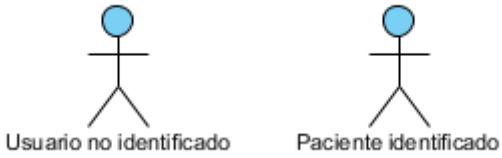


Ilustración 5 - Jerarquía de actores de la aplicación móvil.

Usuario no identificado: Representa a los usuarios que descargan la aplicación y todavía no se han identificado.

Paciente identificado: Representa a los pacientes que se han identificado de forma correcta en la aplicación móvil.

4.3.2. Página web

En primer luchar se mostrará el modelo de casos de uso de la página web, como se puede observar en la siguiente ilustración.

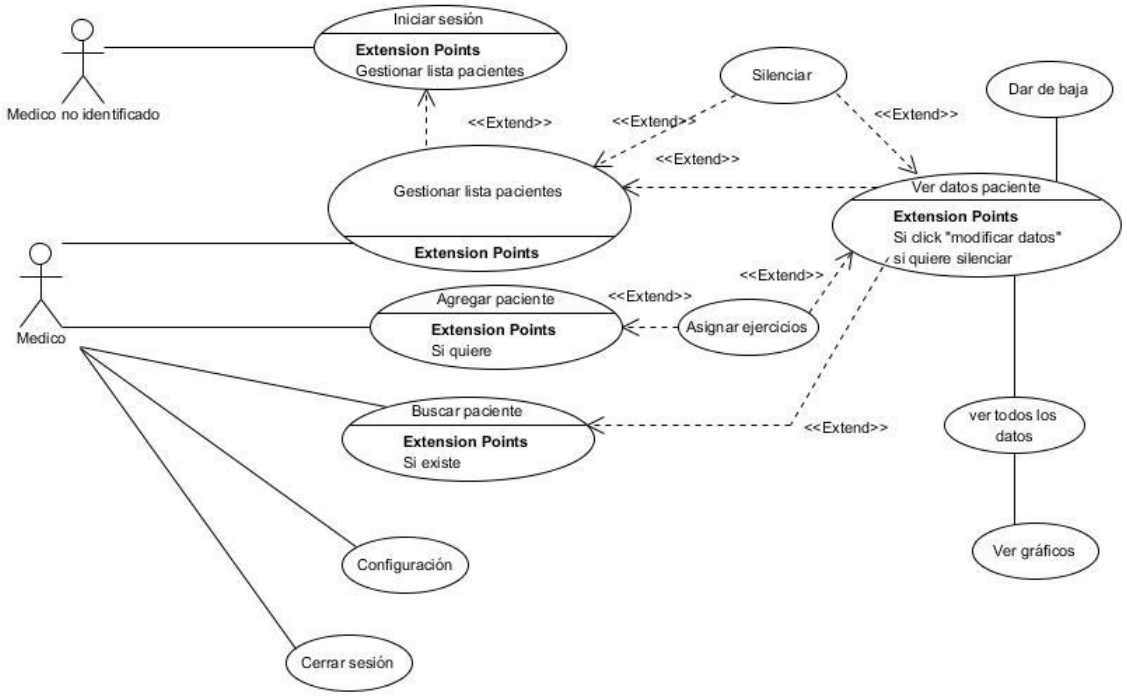


Ilustración 6 - Casos de uso página web.

- **Iniciar sesión:** Permite que el usuario introduzca las credenciales de acceso, si estas credenciales son correctas el sistema le responderá con el token de acceso. Gracias a este token el médico tendrá acceso a todas las funcionalidades del sistema.

Este es el único caso de uso que pertenece a los usuarios no identificados, por lo tanto, será lo único que se pueda realizar en el sistema sin estar identificado, a continuación, se presentarán los casos de uso para cuando el médico ya se ha identificado:

- **Agregar paciente:** Una vez el usuario ha iniciado sesión en el tendrá acceso a una serie de funciones en la página principal de la web. Una de esas funciones de las que el médico dispondrá es añadir un paciente al sistema. Podrá crear una cuenta que posteriormente dará acceso a la aplicación móvil a un paciente.

    - **Asignar ejercicios:** Mientras el médico está creando la cuenta del paciente dispondrá de la opción de asignarle los ejercicios que este debe realizar cada día, así como la distancia que ha de caminar y las preguntas que se le realizarán. El médico podrá introducir al paciente dentro de un grupo de riesgo lo cual le asignará por defecto una serie de ejercicios, preguntas y distancia objetivo de caminar. Sin embargo, podrá quitar o añadir ejercicios y preguntas, así como modificar la distancia que el paciente ha de caminar.

- **Buscar paciente:** Filtrará la lista de pacientes que aparece en la página principal con lo que introduzca en la búsqueda. La búsqueda se podrá realizar en base a

cualquiera de los campos de la tabla como por ejemplo nombre y apellidos o número de historial clínico. Además de esto también se podrá ordenar la tabla de pacientes en función a alguno de los campos (por defecto la fecha de la última modificación).

- **Gestionar pacientes:** En este caso de uso el médico podrá gestionar la lista de pacientes que aparece en la página principal. Tendrá principalmente dos acciones para cada uno de los pacientes: podrá ver sus datos si hace click sobre la línea del paciente en la tabla o podrá silenciar al paciente si hace click sobre el botón de “silenciar”.
    - **Ver datos paciente:** En este sub-caso de uso el médico obtendrá la información de la actividad del paciente, de las preguntas y sus respuestas y de la distancia que ha caminado el paciente los últimos días.
        - **Ver todos los datos:** Mostrará todos los datos guardados en el sistema del paciente de forma detallada. Además, el médico tendrá acceso a graficas para ver cómo ha sido la progresión del paciente a lo largo del tiempo.
        - **Dar de baja:** Este caso eliminará toda la información de un paciente del sistema. Además, hará que el paciente ya no tenga acceso a la aplicación móvil.
    - **Silenciar:** El médico podrá silenciar las notificaciones del paciente, este dejará de enviar datos al sistema para no saturarlo. Esta opción no eliminará los datos del paciente ni eliminará su cuenta, el paciente podrá seguir utilizando la aplicación de forma normal, pero sin enviar sus datos al sistema.
- **Configuración:** En este apartado el médico podrá crear preguntas y grupos de riesgo, cambiar el idioma y exportar los datos.
- **Cerrar sesión:** Con esta opción el usuario podrá cerrar la sesión de la aplicación web. Esto le llevará a la página de Login y eliminará los tokens del navegador.

4.3.3. Aplicación móvil

Ahora se mostrará el modelo de casos de uso de la aplicación móvil

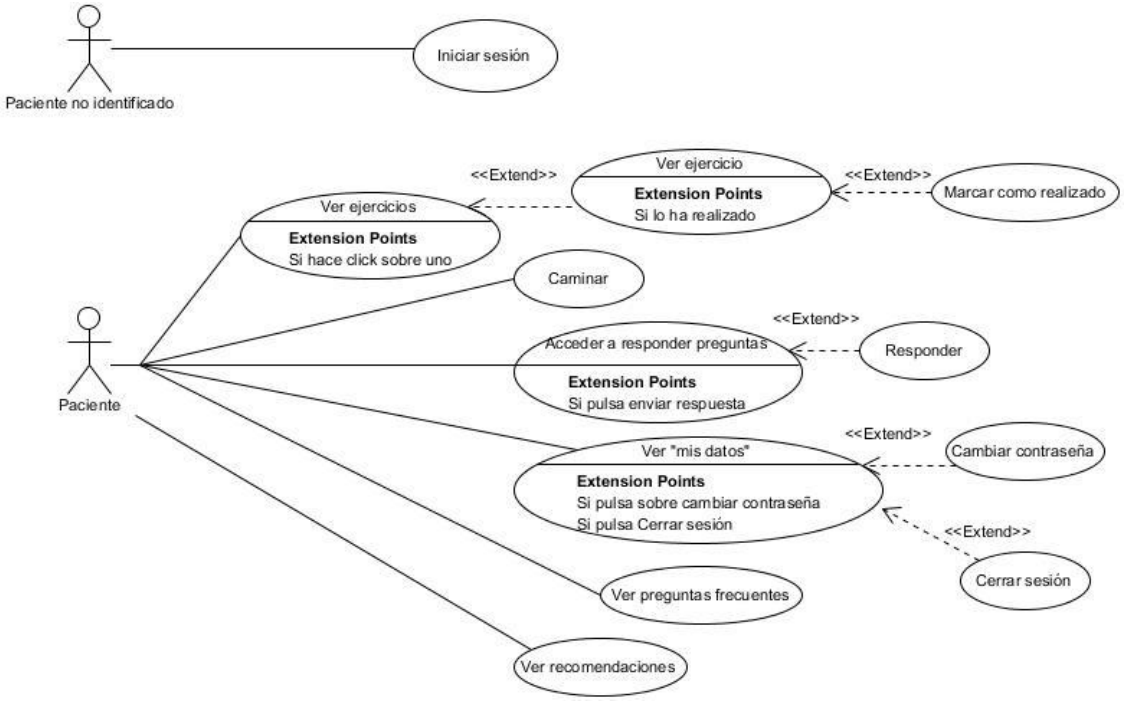


Ilustración 7 - Casos de uso aplicación móvil

- **Iniciar sesión:** Permite que el usuario que se ha descargado la aplicación introduzca las credenciales de acceso a la aplicación. Solo podrán iniciar sesión las personas a las que un médico les haya creado la cuenta anteriormente.
  El único caso del que puede hacer uso una persona no registrada en la aplicación es iniciar sesión, ya que el resto de casos requieren que el paciente este identificado:

- **Ver ejercicios:** En este caso el paciente podrá ver la lista de ejercicios que el médico le ha asignado. El usuario de la aplicación podrá hacer click sobre uno de los ejercicios lo cual le llevará al sub-caso de uso “Ver ejercicio”.

    - **Ver ejercicio:** Aquí es donde el paciente podrá ver la información detallada del ejercicio además de un video explicativo de cuál es la forma correcta de realizar el ejercicio. Aparte de esto el paciente podrá marcar el ejercicio como realizado, lo cual lo llevará al sub-caso de uso “Marcar como realizado”.

        - **Marcar como realizado:** El paciente puedo marcar los ejercicios como realizados, lo cual hará que en el sistema aparezca que el

paciente ha realizado dicho ejercicio. Esta información estará disponible para el médico cuando la quiera ver.

- **Caminar:** Permite al usuario llevar el control de la caminada que está haciendo. Este caso de uso contralara la distancia que realiza el paciente y el tiempo que trata en realizarla.

- **Opciones:** El paciente tendrá acceso a las opciones de la aplicación. Dentro de estas opciones podrá ves sus datos, cambiar la contraseña y cerras sesión.

- **Acceder a responder preguntas:** El usuario tendrá la opción de acceder a una pantalla en la que responder las preguntas que el médico le ha asignado.

    - **Responder:** Este sub-caso de uso enviará las respuestas del usuario a las preguntas asignadas por el médico al sistema. Esto hará que el médico pueda acceder a las respuestas.

- **Ver preguntas frecuentes:** Si el usuario tienes dudas sobre alguno de los aspectos de la aplicación también dispondrá de un apartado donde podrá consultar las preguntas más frecuentes que se suelen hacer los pacientes.

- **Ver recomendaciones**: El usuario podrá ver una serie de recomendaciones las cuales estarán acompañadas por un gif animado. Para que sean más fácil de entender.

4.4. Modelo de dominio

El modelo de dominio es una representación gráfica de todos los datos que se van a almacenar en el sistema y también la relación que existe entre ellos. En este apartado se mostrará y explicará el modelo de dominio utilizado para el desarrollo del proyecto.

4.4.1. Modelo de dominio de la página web

En la siguiente ilustración se muestra el modelo de dominio utilizado para la aplicación web.

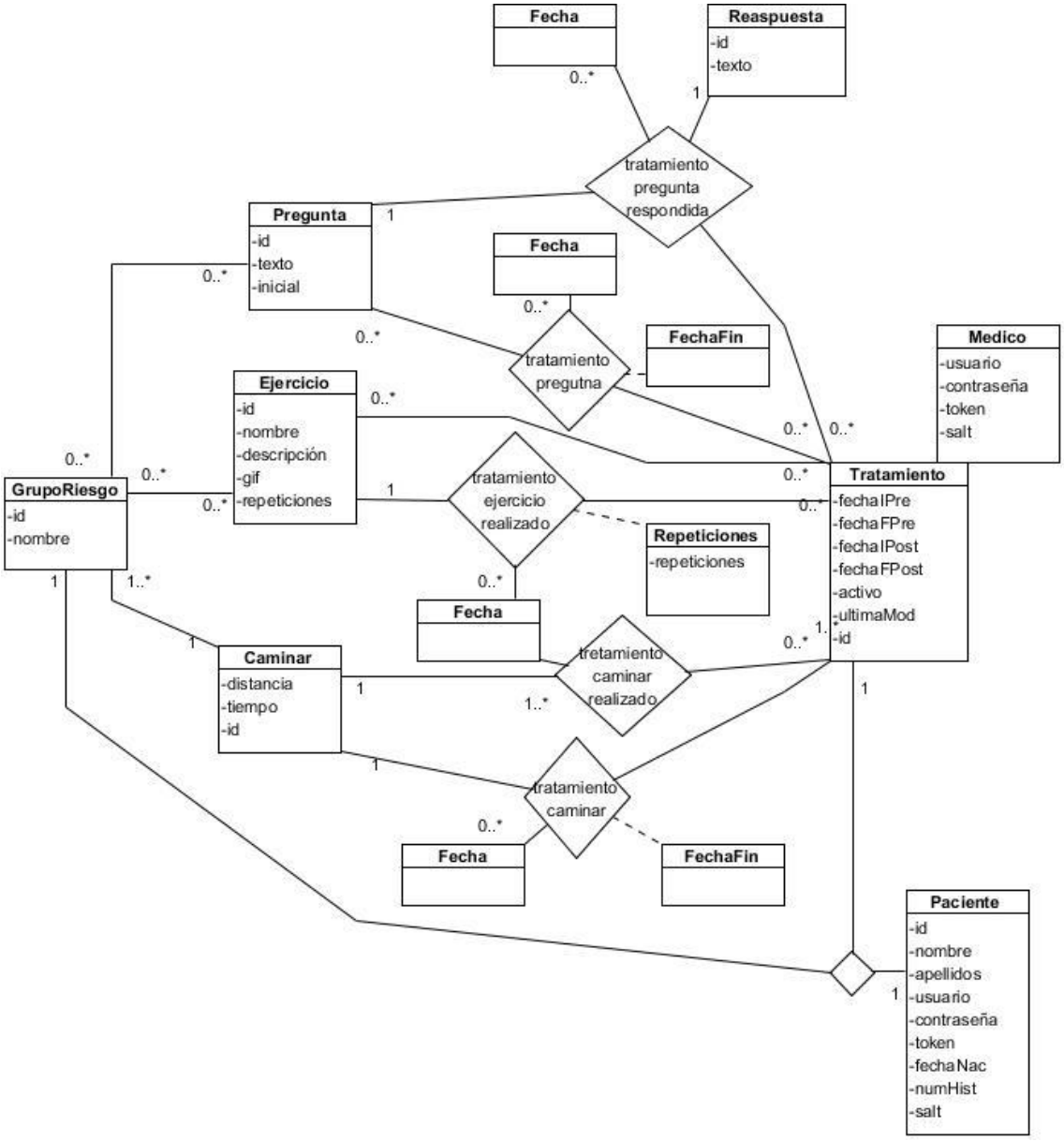


Ilustración 8 - Modelo de dominio

A continuación, se detallará el significado de cada una de las entidades y las relaciones que las unen:

- **Médico:** Esta entidad hace referencia a los médicos que tendrán acceso a la página web. En ella se almacenarán los datos de acceso necesarios para que los médicos puedan iniciar sesión en la página web.

- **Paciente:** En la entidad Paciente se guardarán los datos personales de cada uno de los pacientes del sistema, así como los datos de acceso para la aplicación móvil. Cuando el médico cree un nuevo paciente sus datos se guardarán en una entidad Paciente.

- **GrupoRiesgo:** Entidad que define el grupo de riesgo al que pertenece un paciente. Cada grupo de riesgo tendrá unos ejercicios, preguntas y distancia predeterminados para que al crear un nuevo paciente y definir el grupo de Riesgo al que pertenece este ya tenga una serie de ejercicios, preguntas y distancia asignadas, aunque estos parámetros serán personalizables para cada uno de los pacientes.

- **Tratamiento:** El paciente podrá formar parte del sistema en diferentes ocasiones, por ese motivo ha sido creada esta entidad. Será la encargada de guardar las fechas desde cuando está activo el paciente, cuando se operará/opero y si actualmente este tratamiento esta activo en el sistema.

- **Caminar:** En esta entidad se guardan las distancias y velocidades que han de recorrer los pacientes. La entidad grupoRiesgo está relacionada con esta entidad, de esta forma se le asigna una distancia y velocidad por defecto a un grupo de riesgo. La entidad caminar también tiene dos relaciones múltiples con la entidad tratamiento, en “tratamientocaminar” se guardará la distancia y velocidad que se le ha asignado a el paciente en ese tratamiento y en “tratamientocaminarrealizado” se guardará los datos diarios de cuanto camina un paciente y a que velocidad los hace.

- **Ejercicio:** La entidad ejercicio es la encargada de guardar los datos de los ejercicios. Esta entidad está relacionada con la entidad grupoRiesgo, de esta forma se consiguen guardar los ejercicios que pertenecen por defecto a un grupo de riesgo. La entidad ejercicio también está relacionada con la entidad tratamiento con una relación múltiple y una relación binaria. La entidad binaria servirá para guardar que ejercicios se han asignado a cada tratamiento mientras que la relación múltiple (tratamientoejerciciorealizado) se usará para guardar cuales son los ejercicios que ha realizado cada paciente a diario.

- **Pregunta:** Esta entidad será donde se guardarán los datos de las preguntas almacenadas en el sistema. Tiene una relación binaria con la entidad grupoRiesgo, al igual que las entidades explicadas anteriormente, esta relación sirve para almacenar cuales son las preguntas asignadas por defecto a un grupo de riesgo. Además de ello la entidad pregunta tiene dos relaciones múltiples con la entidad tratamiento. “tratamientopregunta” se utilizará para almacenar las preguntas que se le han asignado al paciente para el tratamiento mientras que “tratamientopreguntarespondida” será la encargada de guardar las respuestas

del paciente a diario gracias a la entidad respuesta que se explicará a continuación.

- **Respuesta:** La entidad Respuesta será donde se almacenarán los datos de las respuestas que el paciente dará a diario a las preguntas que tiene asignadas por el médico.

5. Análisis y diseño

Tras la realización de la captura de requisitos el siguiente paso en un proyecto de este tipo es realizar la fase de análisis y diseño del proyecto. En esta fase del proyecto lo que se realizará es profundizar en la arquitectura presentada en el planteamiento inicial y se explicará cual es la estructura utilizada para el desarrollo de eTorax.

5.1. Estructura de la API-REST

Para el desarrollo del Back-End se ha decidido separar el proyecto en diferentes capas o módulos. Se ha decidido utilizar una estructura multicapas ya que de esta forma se aumentará la seguridad y el modularidad del proyecto. De esta manera se conseguirá que si en algún momento se han de realizar cambios en el back–end estos cambios afecten a la menor cantidad de código posible además de facilitar la realización de cambios.

Las capas en las que se dividirá la API-REST son las siguientes: capa de seguridad o middleware, capa del núcleo y capa de la base de datos. En la siguiente ilustración se muestra el esquema de las capas en las que se divide el back-end y como interactúan entre ellas

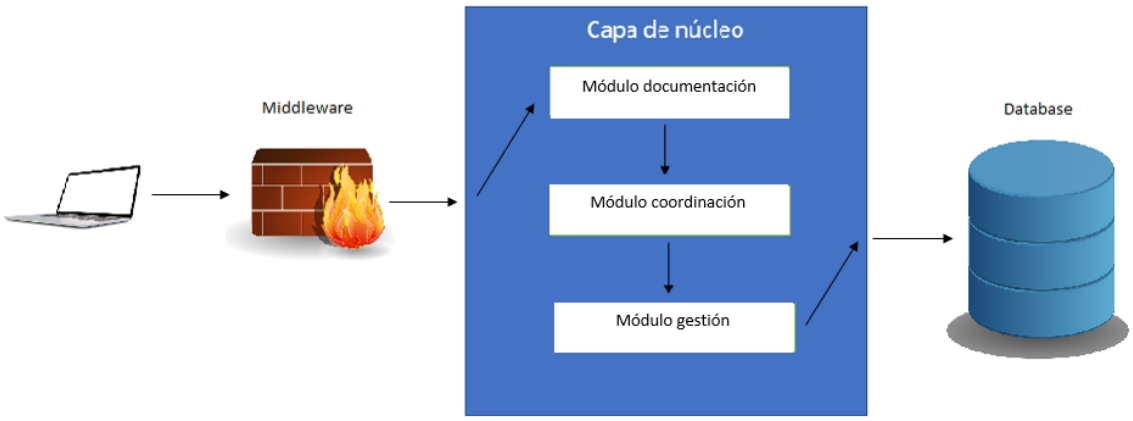


Ilustración 9 - Estructura back-end

5.1.1. Capa se seguridad o Middleware.

Esta es la primera capa a la que llegarán las peticiones del cliente. Esta capa será la encargada de comprobar las cabeceras de la petición y dar acceso al resto de módulos o rechazar el acceso.

Para aceptar o rechazar las peticiones la capa de seguridad se fijará en el token, al cual es una serie de caracteres creados a partir del nombre de usuario, la fecha de creación del token, la fecha de expiración y una palabra secreta (cuanto más compleja mejor) almacenada en el sistema. El token debe de ser único para cada usuario he irrepetible a través del tiempo. Además, para más seguridad el token estará encriptado.

Al iniciar sesión al usuario se le enviará su token de acceso. Este token se guardará en el cliente para poder enviarlo en las cabeceras de las peticiones. Al realizar una petición el cliente introducirá el token del usuario en la cabecera de la petición. La capa de seguridad será la encargada de analizar la cabecera y comprobar si cumple las condiciones necesarias.

Lo primero que hará será comprobar que la cabecera de la petición tiene un token de acceso. Si la cabecera no lleva ningún token se rechazará el acceso a la petición. Por lo contrario, si hay un token en la cabecera se comprobará si el token es válido. Si no es válido rechazará la petición y en cambio si el token fuese valido la petición pasará a la siguiente capa, la capa del núcleo.

5.1.2. Capa del núcleo.

En esta capa se alojará la mayor parte de la lógica referente a la gestión de las peticiones realizadas por el cliente y las respuestas que se le enviará.

Esta capa esta a su vez dividida en tres submódulos cada uno de ellos con diferentes funciones. Pese a ser módulos independientes trabajarán de forma conjunta para responder las peticiones del usuario. Los módulos que participan en la capa del núcleo son: módulo de comunicación, módulo de coordinación y módulo de gestión.

- Módulo de comunicación: Una vez las peticiones han superado la validación realizada por la capa de seguridad es aquí donde llegarán. Este módulo será en el encargado de extraer los parámetros de la petición y comprobar que todos ellos sean válidos. Si los parámetros de la petición del cliente no son los esperados se enviará la respuesta con el error correspondiente. En caso de que los parámetros sean correctos se llamará al siguiente módulo con los parámetros extraídos. Cuando el siguiente modulo (módulo de coordinación) termina la ejecución devolverá el resultado de la ejecución a este módulo para que este pueda enviar la respuesta al cliente.

- Módulo de coordinación: La función de este módulo es la de la gestión de datos. Además, este módulo tendrá que recopilar las respuestas de las peticiones a la base de datos y crear la respuesta para el front-end.

- Módulo de gestión: El módulo de coordinación llama a este módulo para que se comunique con la base de datos. Este será el único módulo que se comunique con la base de datos y esta será la única función que debe realizar este módulo.

5.1.3. Capa de la base de datos

En la siguiente ilustración se muestra el diagrama relacional de la base de datos para el back-end. Este diagrama se ha obtenido a partir del modelo de dominio ya presentado anteriormente.

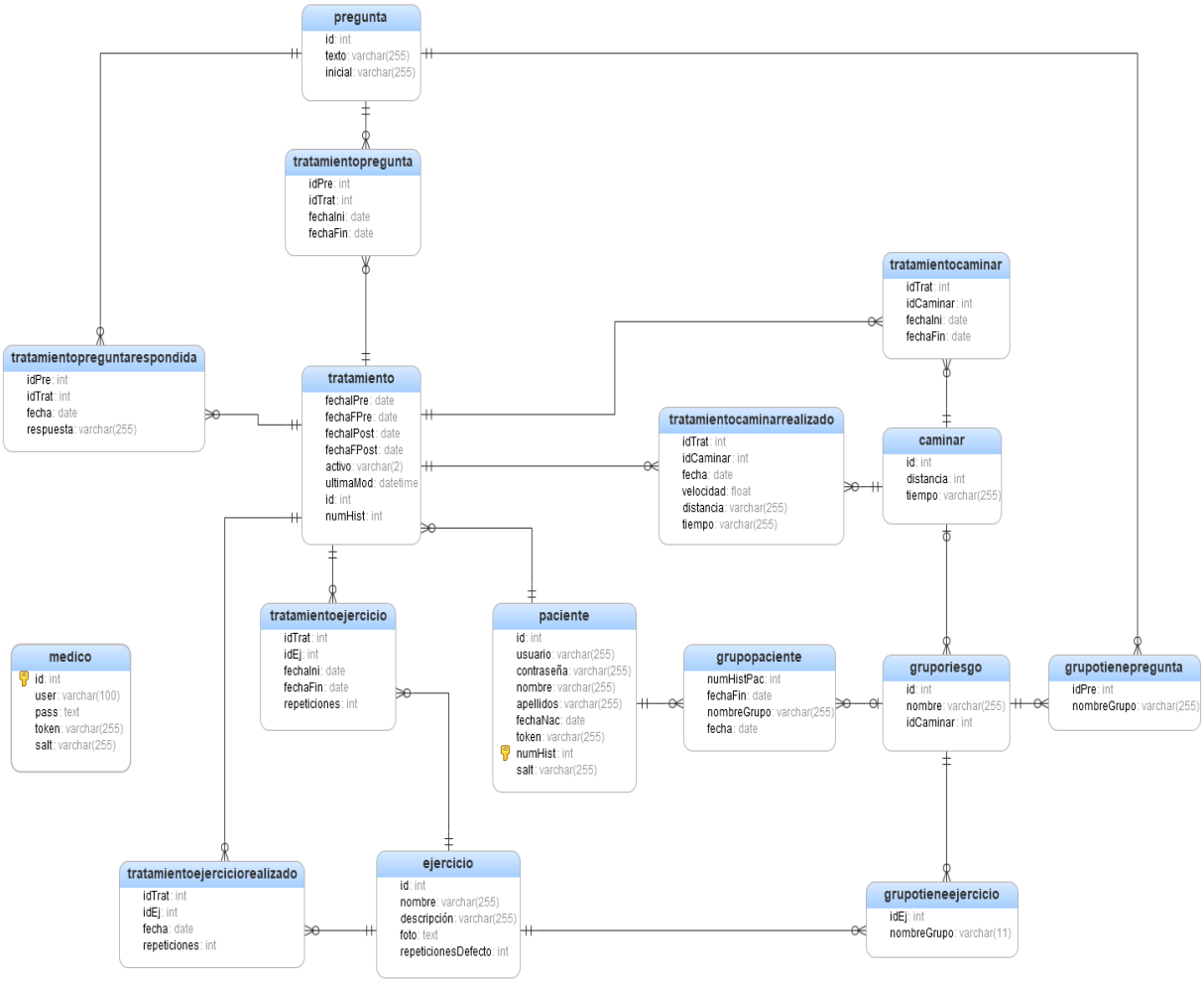


Ilustración 10 - Diagrama referencial

A continuación, se explicarán cada una de las tablas que forman parte de la base de datos.

Médico: Esta tabla es la encargada de almacenar los datos de los médicos para que estos puedan acceder a la página web. Esta tabla solo tendrá una única línea con tres atributos: usuario, contraseña, token y salt. El motivo por el que solo habrá una línea en la tabla es que todos los médicos utilizarán el mismo usuario para iniciar sesión en la página web. Esta es la única tabla de la base de datos que no está relacionada con ninguna otra tabla.

Paciente: en esta tabla se almacenarán los datos personales del paciente: nombre apellidos, numero de historial clínico y fecha de nacimiento, así como la información necesaria para que el paciente inicie sesión en la aplicación móvil: usuario, contraseña, token y salt. Esta tabla está relacionada con la tabla ***gruporiesgopaciente*** la cual simplemente sirve como relación para unir al paciente con su grupo de riesgo, esta tabla tendrá el nombre del grupo de riesgo, el número de historial clínico del paciente, la fecha en la que comenzó a formar parte de dicho grupo de riesgo y la fecha en la que dejo de formar parte del grupo de riesgo.

La tabla ***paciente*** también está relacionada con la tabla ***tratamiento***. Esta tabla hace referencia a los tratamientos que el usuario tiene en cada momento. En ella se almacenarán la fecha de inicio del preoperatorio, la fecha del final del preoperatorio, la fecha del inicio del postoperatorio, la fecha del final del postoperatorio, si el tratamiento esta activo y la última modificación del paciente en la base de datos.

La tabla ***gruporiesgo*** hace referencia a los distintos grupos de riesgo que hay en el sistema de los cuales pueden formar parte los pacientes. Esta tabla está relacionada con las tablas ***gruporiesgotieneejercicio***, ***caminar*** y ***gruporiesgotienepregunta***. Estas relaciones servirán para almacenar que ejercicios, preguntas y datos de caminar tiene cada grupo de riesgo por defecto. Esto hará que a la hora de crear un nuevo paciente y asignarle un grupo de riesgo se le asignen los ejercicios, preguntas y distancia de caminar por defecto.

La tabla ***caminar*** almacenará los datos que el paciente debe cumplir en su caminata diaria. Esta tabla estará relacionada con las tablas ***tratamientocaminar*** y trataientocaminarrealizado. La primera sirve para asignar a un tratamiento la distancia y el tiempo que ha de caminar y la segunda tabla almacenará los datos diarios de cada paciente, es decir la distancia y velocidad diaria a la que ha caminado el paciente.

La tabla ***ejercicio*** es la encargada de almacenar los ejercicios que se le pueden asignar a un paciente. Esta tabla está relacionada con las tablas ***tratamientoejercicio*** y ***tratamientoejerciciorealizado***. La primera de ellas hace referencia a los ejercicios que un determinado tratamiento tiene asignados y los que tubo asignados es un pasado. La tabla ***tratamientoejerciciorealizado*** será la encargada de almacenar los datos diarios sobre los ejercicios que ha realizado cada paciente. En esta tabla se almacenará si un determinado paciente ha realizado los ejercicios que le correspondan y cuantas repeticiones ha realizado de cada ejercicio.

La tabla ***pregunta*** almacena todas las preguntas del sistema que pueden ser asignables a los pacientes. Esta tabla está relacionada con las tablas ***tratamientopregunta*** y ***tratamientopreguntarespondida***. La primera de ellas será la

que almacene las preguntas asignadas a cada tratamiento. Y ***tratamientopreguntarespondida*** será la encargada de almacenar las respuestas diarias de cada paciente a cada una de las preguntas que le han sido asignadas.

5.2. Diseño de la página web.

El diseño de la página web se ha visto condicionado en todo momento por el framework seleccionado para el desarrollo de la misma, que como se ha mencionado en otros apartados es Angular. Por ello es importante saber cuál es la forma de funcionar angular, antes de nada.

Lo primero que hay que mencionar sobre angular es que se trata de una SPA ("Single Page Aplication") el cual se basa en el uso de componentes. Las paginas en angular tiene un componente principal, generalmente llamado "app.component" a partir del cual se desarrolla la aplicación.

Los componentes de angular están pensados para ser completamente reutilizables y la idea es crear la página mediante la utilización de componentes más pequeños. Por ejemplo, en una página web podríamos crear un componente para el menú de navegación, llamado navbar.component. Para utilizar dicho menú en una página bastaría con inyectar el componente del menú de navegación en el HTML del componente padre.

De esta forma se podrían ir utilizando diferentes componentes en la misma vista hasta realizar la página completa. Para que lo explicado anteriormente quede más claro se mostrará una ilustración explicativa a continuación

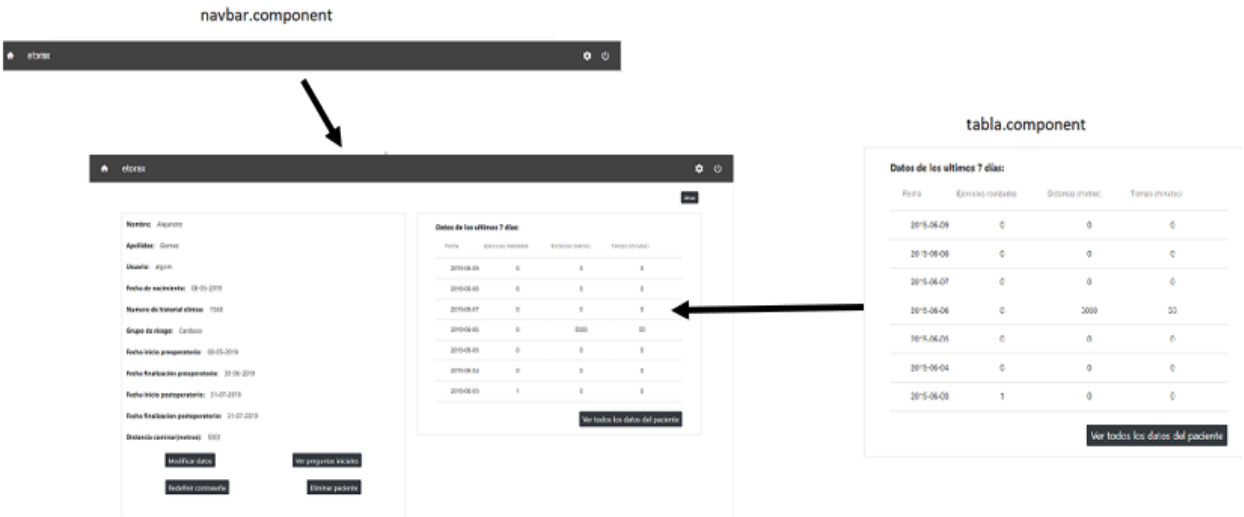


Ilustración 11 - Ejemplo del uso de componentes en Angular

Hasta ahora se ha visto como un componente puede estar formado por varios componentes, pero falta decir que cada componente está formado por tres ficheros: un fichero HTML, otro CSS y un fichero de TypeScript. El fichero de HTML se usa para

crear la vista del componente, el de CSS para dar estilo a dicha vista y por último el de TypeScript se encarga de dar la funcionalidad al componente.

Además de los componentes en angular también existen los servicios ("services"), los cuales tienen funciones a las que se puede acceder desde cualquier componente, siempre que dicho servicio haya sido definido en el componente. Estos servicios son especialmente útiles para realizar las peticiones a la API-REST.

Una vez se ha detallado cual es la forma de funcionar del framework a utilizar se pasará a explicar cuál ha sido la estructura utilizada para el desarrollo de la aplicación web.

Al igual que todas las aplicaciones desarrolladas en angular esta también tendrá un componente principal llamado "app.component" a partir del cual se desarrollará el resto de la aplicación y donde se introducirán el resto de componentes.

Además del componente "app.component" también se han creado dos carpetas: "components" y "services".

La carpeta components es donde se situarán todos los componentes creados para la aplicación web. Dentro de esta carpeta habrá una carpeta para cada uno de los componentes donde se guardarán los tres archivos que forman el componente: HTML, CSS y TypeScript.

En la carpeta "services" será donde se sitúen todos los servicios del proyecto de angular. Cada uno de los servicios creados tendrá su propia carpeta dentro de esta donde se situarán todos sus archivos.

Por último, junto a las carpetas mencionadas anteriormente también se encuentra el archivo "router", el cual será el encargado de gestionar los URL de la aplicación y la navegación que se realizará entre ellos.

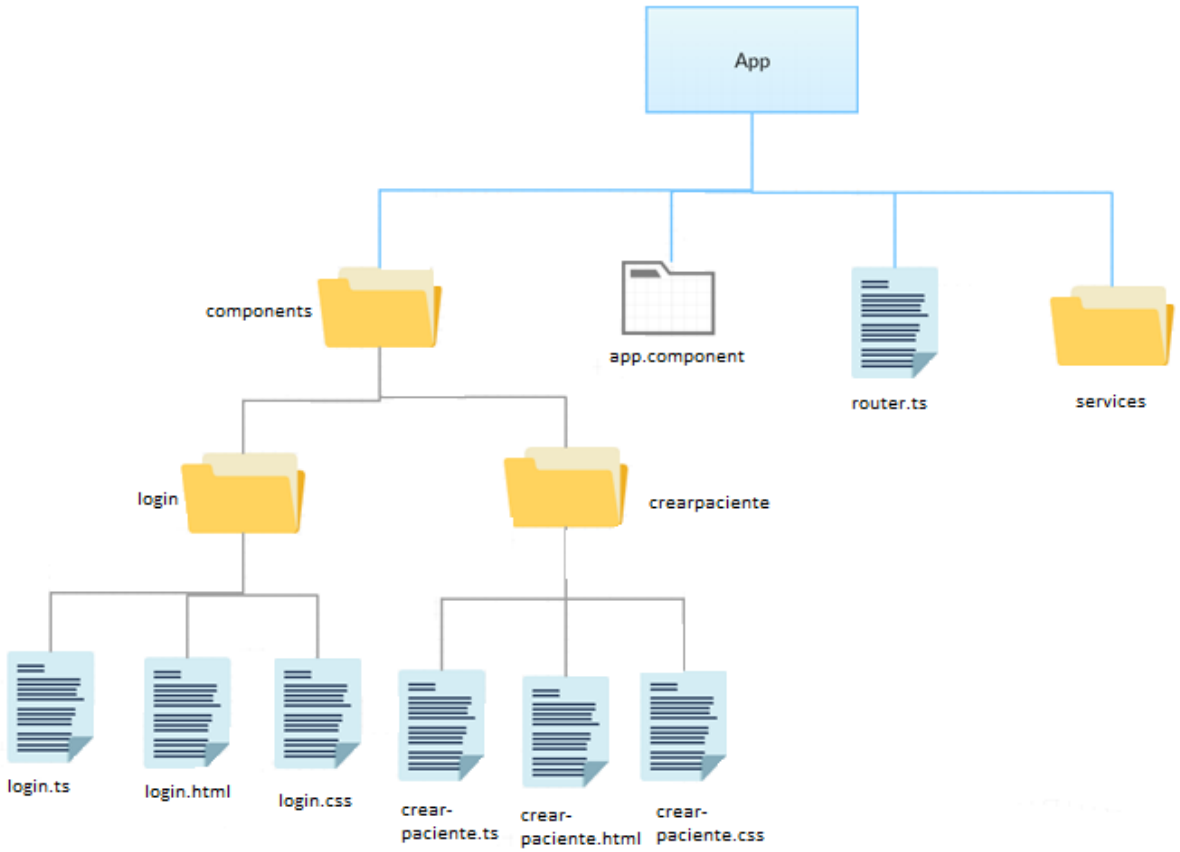


Ilustración 12 - Esquema angular

5.3. Diseño de la aplicación móvil.

Como se ha explicado en apartados anteriores el desarrollo de la aplicación de eTorax se realizará en Xamarin, ya que permite desarrollar aplicaciones para varios sistemas operativos de forma simultánea. El hecho de desarrollar la aplicación en Xamarin es algo que condiciona el diseño de la aplicación. Por lo tanto, antes de nada, es importante conocer de forma básica como se generan los proyectos en Xamarin y la estructura de los mismos.

Al crear un proyecto en Xamarin este se creará con varias carpetas en su interior las cuales variarán dependiendo de las opciones que se hayan seleccionado durante la creación el proyecto. Las carpetas que se crearán serán:

- La carpeta de código compartido. Esta carpeta tendrá el nombre del proyecto, en este caso el nombre de esta carpera será eTorax. En esta carpeta es donde tiene que ir la mayoría del código desarrollado. Sin embargo, no todo el proyecto se puede desarrollar en esta carpeta. Existen funcionalidades que han de ser desarrolladas específicamente para cada uno de los sistemas operativos. Por ello Xamarin también crea una carpeta por cada sistema operativo para el que se desea desarrollar la aplicación.

- Las carpetas específicas para cada sistema operativo: Estas carpetas tendrán el nombre formado por el nombre del proyecto, punto, el sistema operativo, por ejemplo: eTorax.Android, eTorax.iOS o eTorax.UWP (Universal Windows Platform). En estas carpetas es donde se desarrolla el código especifico de las funcionalidades que no pueden ser desarrolladas en la carpeta de código compartido. Además de ello, las imágenes del proyecto han de ser almacenadas en cada uno de los proyectos, es decir, las imágenes que utiliza la aplicación se han de introducir en la carpeta de cada sistema operativo y no en la del código compartido. Otra de las cosas que han de ser especificadas para cada proyecto son los permisos que requiere cada uno de los sistemas operativos, los cuales han de ser especificados de forma diferente para cada sistema operativo.

Una vez vista de forma básica como se estructuran los proyectos generados en Xamarin se expondrá como se ha decidido estructurar eTorax. Para realizar el diseño de eTorax se ha decidido utilizar el patrón de diseño MVVM (Model-View-ViewModel), gracias al cual se puede separar la lógica del proyecto de la vista del mismo.

Para utilizar el patrón MVVM es necesario crear tres carpetas diferentes en la carpeta de código compartido del proyecto. Estas son las tres carpetas necesarias:

- Models: en esta carpeta es donde se definen y almacenan las clases del proyecto. Por ejemplo, la clase “Ejercicio” se definirá en esta carpeta.

- Views: En esta carpeta es donde se encontrarán todas las vistas del proyecto. Las aplicaciones desarrolladas en Xamarin comparten las vistas para los diferentes sistemas operativos, por lo que en esta carpeta será el único lugar donde se desarrollarán las vistas.

- ViewModels: En esta carpeta se encontrarán los archivos encargados de implementar las propiedades de las vistas y los comandos a la que la vista puede enlazar datos. Los archivos de esta carpeta también serán los responsables de coordinar las interacciones de la vista con las clases del modelo.

Además de las carpetas necesarias para utilizar el patrón de diseño MVVM también será necesaria una carpeta donde se encontrarán los servicios de la aplicación. En esta carpeta se encontrará el archivo ApiServices, el cual es el encargado de realizar las peticiones al servidor.

6. Desarrollo

En este apartado se explicarán los pasos que se han realizado para el desarrollo de eTorax, el apartado estará dividido en tres partes al igual que el proyecto: API REST, aplicación web y aplicación móvil. Para cada una de estas partes se explicará cómo se ha realizado la instalación de los programas y librerías necesarias, así como los puntos clave del desarrollo.

6.1. Desarrollo del Back-End

En este apartado se explicará cómo se ha realizado el desarrollo de la parte referente al Back-End del proyecto. Pero antes de analizar cómo se ha realizado el desarrollo del back-end se expondrá brevemente cuales han sido los pasos necesarios para instalar las herramientas que se utilizarían durante el desarrollo del back-end.

6.1.1. Instalación y estructuración

Como ya se ha mencionado en apartados anteriores las herramientas para desarrollar el back-end son NodeJS y el framework Express. NodeJS se ha instalado mediante el instalador que proporciona la página web de NodeJS. Una vez instalado NodeJS el siguiente paso es instalar Express lo cual es muy sencillo gracias a NPM (“NodeJS Package Manager”) que es el gestor de paquetes de NodeJS.

Una vez instalado express el siguiente paso será crear el proyecto del back-end. Express dispone de comandos específicos para crear proyectos de forma automática.

El resultado de ejecutar dichos comandos es una carpeta con el nombre del proyecto la cual contiene una serie de archivos creados automáticamente.

Una vez express ha creado el esqueleto de la aplicación el siguiente paso que hay que dar es el de instalar las dependencias del proyecto, si no se hiciese este paso al intentar ejecutar el proyecto podría dar error en caso de que alguna de las dependencias no estuviese instalada.

Una vez instaladas las dependencias el proyecto ya se encuentra en un punto en el que puede ser ejecutado.

Si el proyecto se ha creado de forma correcta una vez en el navegador se puede acceder a http://localhost:3000/ , la pestaña que se mostraría seria la que aparece en la siguiente ilustración.

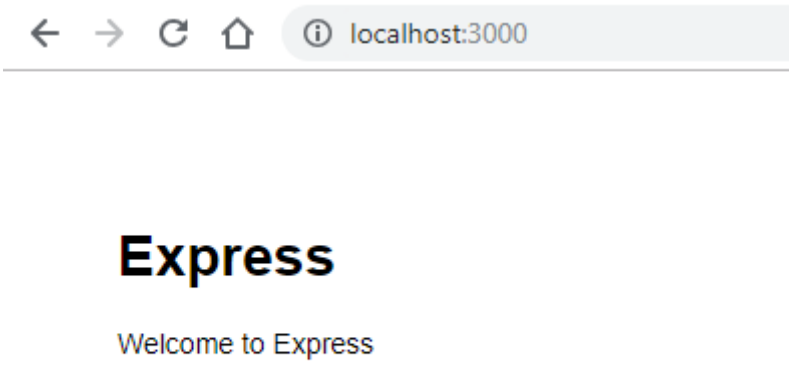


Ilustración 13 - Pagina del proyecto creado por express

Una vez se ha comprobado que el proyecto creado por express funciona de forma correcta el siguiente paso será personalizarlo para cumplir los requisitos del proyecto que vamos a crear. Por ello se han eliminado algunas de las carpetas creadas inicialmente por express y se ha modificado el contenido de otras. La primera carpeta que se ha eliminado es la carpeta "*public"*, ya que en ella es donde se almacenan las imágenes utilizadas por las vistas y las hojas de estilo del proyecto, y ya que eTorax tendrá el font-end y el back-end separados no necesitaremos esta carpeta. La carpeta "*routes*" también se puede eliminar, ya que como se ha explicado en los apartados anteriores la forma de organizar el proyecto será diferente (mediante módulos). En sustitución de la carpeta "*ruotes*" se ha creado una nueva carpeta llamada "modules" con otras tres carpetas en su interior: "mod_comunication"," mod_coordination", y "mod_object".

Al eliminarse la carpeta "*router*" que es donde se almacenan las rutas del sistema se ha creado el archivo "app.js" que es donde se definirán en eTorax. Además, este archivo también servirá para sustituir la carpeta "bin" la cual contiene la configuración para lanzar y realizar pruebas del servidor. Por lo tanto "app.js" será el fichero encargado de arrancar el servidor y configurarlo, además de ser el fichero en el que se definirán las rutas y se asociarán con las clases correspondientes.

Se han eliminado los archivos de la carpeta "*views*" y se ha creado uno nuevo llamado "index.html" en él se ha puesto un código que se mostrará cuando se entre a la URL del back-end desde un navegador. La principal utilidad de esto es saber si el back-end se encuentra funcionando. Para comprobar si el back-end esta online solo habrá que acceder al URL del mismo y si se muestra como en la siguiente imagen significa que el servidor está funcionando correctamente.

Ilustración 14 - Nueva página back-end.

También se ha creado la carpeta “utils” en la cual se situarán los distintos archivos con utilidades del proyecto. A continuación, se exponen los archivos que se han creado en la carpeta de “utils”:

- dbConection.js: Tiene las funciones necesarias para que el back-end se conecte con la base de datos.

- response.js: En este fichero es donde se crearán las respuestas a las peticiones realizadas por el front-end.

- token.js: Se encarga de todas las funciones referentes a la gestión de tokens, desde la validación hasta la creación de nuevos tokens.

- utils.js: Es donde se encuentran las funciones generales del sistema. Gracias a este fichero se consigue una mayor modularidad y se consigue que la repetición de código sea menor.

Y por último falta mencionar la creación del archivo “config.js” donde se almacenará información de la configuración del back-end, como podría ser la información de conexión a la base de datos o el secret-token.

En la siguiente ilustración se muestra cómo queda el esqueleto del proyecto una vez se han realizado todos los cambios mencionados anteriormente:

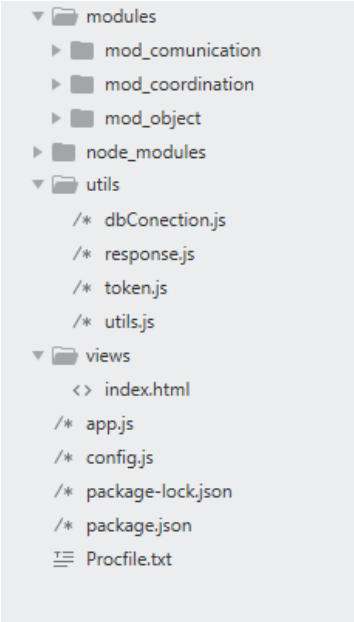


Ilustración 15 - Esqueleto del back-end final

En este punto la estructura inicial del proyecto ya ha sido creada, sin embargo, es necesario instalar todas las librerías que se usarán para desarrollar el back-end. En este caso se ha utilizado cinco librerías además de las que ya vienen por defecto en la carpeta "*node_modules*".

La primera librería es "*mysql*" ya que la base de datos utilizada para el proyecto es de este tipo. Gracias a este módulo se podrá conectar con bases de datos del tipo MYSQL. Para instalar las librerías solo será necesario introducir en la consola de comando el comando de instalación de la librería que se desea instalar

Otra de las librerías que se han utilizado es "jwt-simple". Como ya se ha comentado anteriormente para la autenticación de usuarios se usarán Tokens. El uso de estos tokens se puede realizar de varias formas, por ejemplo, desde la concatenación de variables y valores aleatorios hasta la utilización de módulos ya creados. En este caso se ha decidido utilizar una librería ya creada la cual concatenará variables y valores aleatorios y además los encriptará para mayos seguridad.

Para loguearse en el sistema será necesario el uso de nombres de usuario y contraseñas. Estas contraseñas se almacenarán en la base de datos encriptadas en SHA384. De modo que para poder encriptar las contraseñas se ha decidido utilizar la librería "sha384".

Por último, también se ha instalado una librería cuya única finalidad es facilitar el desarrollo de la aplicación. Esta librería es "nodemon" y hace que si se está ejecutando el servidor y al mismo tiempo se está realizando cambios en el código cada vez que se guardan los cambios en el código se vuelve a ejecutar el servidor aplicando los nuevos cambios realizados. Al no tener que parar el servidor y relanzarlo a mano agiliza el desarrollo.

6.1.2. Rutas y comunicación interna

Como ya se ha explicado brevemente en apartados anteriores el back-end está dividido en diferentes módulos los cuales se comunican entra ellos. De esta manera se puede gestionar las peticiones del cliente de una forma más rápida y ordenada. En este caso las rutas a las que el cliente va a realizar las peticiones están definidas en el fichero "app.js" ya que este es el fichero que arranca el servidor y lo configura. En el momento en el que llegue una petición al servidor este será el primer fichero que se ejecute. En la siguiente imagen se muestra como las rutas han sido definidas en el fichero "app.js".

```
let router_main = require('./modules/mod_comunication/router');
let router_login = require('./modules/mod_comunication/router_login');
let middleware = require('./modules/mod_comunication/middleware');
let router_pacientes = require('./modules/mod_comunication/router_pacientes');
let router_grupos=require('./modules/mod_comunication/router_grupos');
let router_ejercicios=require('./modules/mod_comunication/router_ejercicios');
let router_caminar=require('./modules/mod_comunication/router_caminar');
let router_tratamiento=require('./modules/mod_comunication/router_tratamiento');
let router_preguntas=require('./modules/mod_comunication/router_preguntas');


app.use('/', router_main);
app.use('/login', router_login);
app.use('/secured', middleware);
app.use('/secured/pacientes', router_pacientes);
app.use('/secured/grupos', router_grupos);
app.use('/secured/ejercicios', router_ejercicios);
app.use('/secured/caminar', router_caminar);
app.use('/secured/tratamiento', router_tratamiento);
app.use('/secured/preguntas', router_preguntas);
```

Ilustración 16 - Rutas definidas

Como se puede observar en la parte inferior de la imagen anterior las rutas se encuentran agrupadas, es decir, si el cliente desea realizar una petición que esté relacionada con la gestión de los ejercicios debe realizar una petición a la ruta "/secured/ejercicios". También se puede observar que después de la ruta se hace referencia a alguna de las variables definidas anteriormente. Estas variables que aparecen en la parte superior de la ilustración contienen los paths de los ficheros dentro del servidor. De modo que cuando llega una petición a una determinada ruta se sabe cuál es el fichero que se ha de ejecutar.

En la ilustración también se puede ver que todas las rutas menos dos comparten la primera parte de la ruta: "/secured" además de ello una de las rutas es simplemente "/secured". Esta última ruta hace referencia al middleware y todas las rutas que comparten esa primera parte de la ruta han de pasar primero por el middleware donde se comprobarán sus datos, y si estos son correctos podrán seguir por sus respectivas rutas. Las rutas que no comiencen por "/secured" no han de pasar por el middleware, en este caso son las rutas para iniciar sesión y la ruta raíz, la cual se usa para comprobar que el servidor se está ejecutando.

Middleware

El middleware funcionará como una capa oculta cuya funcionalidad principal será comprobar que la petición que ha realizado el cliente tiene autorización para ser ejecutada o no.

En el caso de eTorax el middleware ha sido situado en el archivo “middelware.js” dentro de la carpeta “mod_comunication”.

El middleware comprobará si en la petición del cliente hay un token, en caso afirmativo tendrá que comprobar que el token que viene en la cabecera de la petición es válido. Una vez ha comprobado que el token es válido tendrá que mirar si el token ha expirado, en caso afirmativo procederá a actualizar el token en la base de datos y cederá el control a la próxima parte de la ruta.

En caso de que el token de la petición no sea válido o no exista token en la petición (menos para las rutas mencionadas anteriormente) el middleware rechazará el acceso a la siguiente parte de la ruta y responderá la petición con el código y mensaje correspondiente.

Si todo ha ido bien y el token ha sido valido el siguiente paso es el módulo de comunicación.

Módulo de comunicación

Este es el módulo en el que se definen los *endpoints.* Se debe definir un método HTTP por cada uno de ellos. A continuación, se muestra un ejemplo de cómo se han definido dichos métodos y se explica cada una de las partes que lo forman.

```
router_pacientes.get('/', async function (req, res) {
    Functions_paciente.getPacientes()
            .then(function (data) {
                let response = Response.generateResponse(2000, data);
                res.status(response.status).send(response);
            })
            .catch(function (errorCode) {
                let response = Response.generateResponse(errorCode);
                res.status(response.status).send(response);
            })
        ;
});
```

Ilustración 17 - Ejemplo modulo comunicación

Como se puede ver en el ejemplo lo primero que aparece es el método a través del cual ha de ser llamado este *endpoint*, en este caso a través del método GET. Los métodos utilizados podrían ser GET, POST, PUT y DELETE.

A continuación, en el ejemplo también se puede observar la ruta de la petición, en este caso se trata de la ruta raíz. También se puede observar que se reciben dos parámetros “req” y ”res”. El primero de ellos es el que nos permite acceder a los datos de la petición como podría ser la cabecera o los parámetros enviados por el cliente.

El parámetro “res” sirve para crear una respuesta con la que responder la petición del cliente.

Si hubiere parámetros en la variable “req” que fueran necesarios este sería el punto en el que se extraerían de la petición y se comprobaría que los parámetros de la petición están en un formato correcto y no tiene valores nulos.

Dentro del método se crea una llamada a otro método en el archivo “functions_paciente.js” del que se espera una respuesta. Si la ejecución se ha realizado de forma correcta el servidor creará una respuesta con los datos de la ejecución provenientes de dicho método. En cambio, sí ha sucedido algún error durante la ejecución se creará una respuesta en base a el código de error proporcionado.

Módulo de coordinación

Este módulo es el encargado de realizar las acciones solicitados por la petición del cliente. Dependiendo de la solicitud puede que en este módulo se tengan que ordenar realizar varios cambios en la base de datos o que tenga que realizar varias peticiones a la base de datos. Este es el módulo encargado de gestionar las peticiones al módulo de gestión (el cual es el que realiza los cambios y peticiones en la base de datos).

A continuación, se muestra un método del módulo de coordinación en el que se puede ver como se crea un paciente y se realiza la llamada al módulo de gestión para que se comunique con la base de datos. Como se muestra en la imagen la llamada al módulo de gestión debe ir precedida por un “await” ya que se encuentra en un método asíncrono y no se conoce el tiempo que el módulo de gestión tardará en responder.

```
exports.modificarPaciente = function (pId,pNombre, pApellidos, pNombreUsuario, pFechaNac) {
    return new Promise(async function (resolve, reject) {
        try {
            let paciente={
                usuario: pNombreUsuario,
                nombre: pNombre,
                apellidos: pApellidos,
                fechaNac:pFechaNac
            }
            let rows = await Paciente.modificarPaciente(pId,paciente);
            resolve(rows);
        } catch (error) {
            console.log(error.message);
            reject(50000);
        }
    });
};
```

Ilustración 18 - Modulo de coordinación

Módulo de gestión

Se trata del módulo que realiza las llamadas a la base de datos para obtener la información necesaria o realizar los cambios en la base de datos. El módulo de coordinación solicita al módulo de gestión que realice alguna acción en la base de datos, una vez el módulo de gestión ha terminado su trabajo le devuelve los resultados al módulo de coordinación a la espera de que este realice más solicitudes.

A continuación, se muestra un ejemplo de cómo es un método en el módulo de gestión.

```
exports.aniadirGrupoPaciente = function (pGp) {
    return new Promise(async function (resolve, reject) {
        try {
            await pool.query({
                sql: 'INSERT INTO grupopaciente SET ?',
                timeout: utils.getDataBaseDefaultTimeout(),
                values: [pGp]
            }, function (error, result) {
                if (!error) {
                    resolve(result.insertId);
                } else {
                    reject(error);
                }
            });
        }
        catch(error) {
            reject(error);
        }
    });
};
```

Ilustración 19 - Ejemplo módulo de gestión

6.1.3. Conexión con la base de datos

Como ya se ha explicado anteriormente el tipo de base de datos que se utilizará para el proyecto es MYSQL y el módulo que se utilizará es *mysql* del cual ya se ha explicado cual es la forma de instalarlo.

En un principio se intentó realizar las conexiones con la base de datos de forma directa, pero esto supuso un problema, las conexiones con la base de datos de MYSQL tienen caducidad y se terminaban cerrando de forma inesperada e incontrolada.

Para solucionar este problema se pensó en crear una nueva conexión para cada petición que se realizaba a la base de datos y una vez finalizada dicha petición cerrar la conexión. Pero este sistema no es viable.

Así que finalmente para realizar la conexión con la base de datos de decidió crear un *pool* de conexiones. El pool mantiene varias conexiones abiertas al mismo tiempo y se encarga de gestionar las consultas que se hacen a la base de datos. Cuando el módulo de gestión necesita realizar una acción en la base de datos el pool marca una de las conexiones que tiene libres como ocupadas hasta que se finaliza dicha acción. Una vez finalizada el pool vuelve a marcar dicha conexión como libre. El pool no impide

que las conexiones con MYSQL se caduquen y se cierren, sin embargo, cuando detecta que una conexión se ha cerrado se encarga de reabrirla.

A continuación, se muestra la forma en la que se ha creado el pool de conexiones:

```
const pool = mysql.createPool(config.dbConfig.heroku);
pool.getConnection((err, connection) => {
    if (err) {
        if (err.code === 'PROTOCOL_CONNECTION_LOST') {
            console.error('Database connection was closed.')
        }
        if (err.code === 'ER_CON_COUNT_ERROR') {
            console.error('Database has too many connections.')
        }
        if (err.code === 'ECONNREFUSED') {
            console.error('Database connection was refused.')
        }
    }

    if (connection) connection.release()

    return
})
```

Ilustración 20 - Pool de conexiones

Por último, mencionar brevemente el hecho de que (como se puede ver en la primera línea de la imagen) los datos para la conexión de la base de datos se guardan en el archivo “config.js” para de esta manera favorecer la modularidad del código y facilitar los futuros cambios si hubiese que realizarlos

6.1.4. Seguridad

El tema de la seguridad es uno de los más importantes de hoy en día y más tratándose de una aplicación que trata con información sensible a cerca de los pacientes. Es imprescindible que los datos manejados por la aplicación estén lo más seguros posible. Por ese motivo, se han utilizado varios sistemas de seguridad en el back-end.

El primero de los sistemas de seguridad que se han utilizado es el uso de tokens. Como se ha explicado anteriormente para la gestión de los tokens se ha utilizado la librería *jwt-simple.* Los tokens están formados por el nombre de usuario, la fecha de creación del token y la fecha de caducidad del token. Además, usando la función *encode()* la cual ofrece la librería mencionada, se encriptará el token usando una contraseña secreta. La librería también ofrece la función *decode()* la cual desencripta el token.

Todas las funcionalidades relacionadas con los tokens se encuentran en el archivo “token.js” el cual se encuentra en la carpeta de “utils”. En este archivo se pueden encontrar las funciones para crear un nuevo token, comprobar la validez del mismo y actualizarlo. Estas funciones serán llamadas desde el Middleware que como se ha mencionado anteriormente es el encargado de dar o denegar el acceso de las peticiones.

Otro de los aspectos de la seguridad en los que se ha hecho hincapié es en el almacenamiento de las contraseñas de los usuarios, tanto de los pacientes como de los médicos. Estas contraseñas se utilizan para iniciar sesión en las aplicaciones y se almacenan en la base de datos encriptadas.

Sin embargo, las contraseñas no son almacenadas en la base de datos tras encriptarlas. Para más seguridad cuando se crea un nuevo usuario se le genera un código alfanumérico de forma aleatoria, este código se llama *salt*. El código *salt* se concatena a la contraseña del usuario y tras la concatenación el *string* resultante es encriptado usando SHA384. Este último string encriptado es lo que se almacena en la base de datos. La salt también es almacenada en la base de datos para poder comprobar que la contraseña introducida por el usuario durante el inicio de sesión es correcta.

El proceso que se sigue a la hora de iniciar sesión es el siguiente: El primer paso es comprobar que el usuario introducido existe en la base de datos. Si es así se obtiene la salt de este usuario, la cual es concatenada con la contraseña introducida por el usuario para iniciar sesión. Una vez se ha concatenado se encripta la cadena resultante y se compara con la contraseña almacenada en la base de datos. Si las dos contraseñas coinciden el usuario ha introducido la contraseña correcta y se le enviará su token para que pueda realizar las peticiones.

6.1.5. Respuestas del servidor

Una vez se ha procesado la petición realizada por el cliente es necesario responder a su petición, tanto si la petición ha sido procesada de forma satisfactoria como si ha sucedido algún error durante la ejecución. Sin embargo, es necesario que las códigos y mensajes de las respuestas sean diferentes en cada caso y que al mismo tiempo compartan la misma estructura.

Para gestionar las respuestas que se harán a las peticiones del cliente se ha creado el fichero “response.js” situado en la carpeta “*utils*”. El cual se encargará de crear las respuestas en base al código que le llegue desde el módulo de comunicación. En la siguiente imagen se puede ver como se solicita una respuesta desde el módulo de comunicación. Como se puede ver en la imagen en caso de que se haya procesado de forma correcta la petición se le enviará al generador de respuestas un código y los datos obtenidos. Sin embargo, si ha sucedido algún error durante la ejecución se le enviará el código del error para que se genere la respuesta en base a dicho código.

```
auth.login(username, password)
    .then(function (data) {

        let response = Response.generateResponse(20010, data);
        res.status(response.status).send(response);

    })
    .catch(function (errorCode) {
        let response = Response.generateResponse(errorCode);
        res.status(response.status).send(response);
    });
```

Ilustración 21 - Solicitud de respuesta desde el módulo de comunicación.

La respuesta que se generará estará formada por los siguientes parámetros:

- Status: El status es el código de respuesta del servidor. Se explico brevemente en el apartado 2.8 cuales eran los códigos existentes.
- Code: Este campo hace referencia al código interno del servidor. Este código es más específico y ayuda a saber dónde ha sucedido el error.
- Msg: Este es el mensaje que se enviará de respuesta al cliente, deberá explicar brevemente lo que ha sucedido.
- Body: Aquí es donde se encontrarán los datos de la respuesta.

A continuación, se muestra un ejemplo de cómo es la respuesta que recibe el cliente que realiza la petición

```
▸ {status: 200, code: 20062, msg: "Se ha  obtenido las preguntas del trata
miento.", body: Array(3)}
```

Ilustración 22 - Ejemplo respuesta en el cliente

Se ha construido un objeto JSON gracias al cual se puede crear las respuestas a partir de los códigos proporcionados por el módulo de comunicación

```
{status: 200, code: 20020, msg: '"Todos los pacientes obtenidos de forma correcta"'},
{status: 200, code: 20021, msg: '"Paciente silenciado correctamente"'},
```

Ilustración 23 - JSON para la creación de respuestas

6.1.6. Asincronía

Otro de los aspectos en los que hay que hacer hincapié es en la asincronía. La forma de funcionar de NodeJS es asíncrona, lo cual ha dificultado la gestión de los datos en algunos momentos. La asincronía hace que mientras se está haciendo alguna

consulta en la base de datos el resto del código continúa ejecutándose lo cual puede ocasionar problemas si lo la parte del condigo que se está ejecutando necesita los datos de que se han de obtener de la base de datos.

Para solucionar los problemas derivados del hecho de que la forma de trabajar de NodeJS es asíncrona se han usado las promesas o “Promises” y la función “asyn-await”.

A continuación, se muestra un ejemplo de cómo han sido implementadas las promesas en el código:

```
exports.getPacientesSilenciados = function () {

    return new Promise(async function (resolve, reject) {
        try {
            let rows = await Paciente.getPacientesSilenciados();
            resolve(rows);
        } catch (error) {
            console.log(error.message);
            reject(50000);
        }
    });
};
```

Ilustración 24 - Ejemplo promesas

Como se puede ver en la imagen anterior si el código se ejecuta de forma satisfactoria la respuesta se realizará mediante el método *resolve* mientras que si el código ha dado algún error se devolverá usando el método *reject*. En la Ilustración 24 presentada en el apartado anterior se puede ver cómo es la forma de hacer la llamada a una función que es una promesa. Como se puede ver utiliza un *then* y un catch los cuales funcionan exactamente igual que con las excepciones. En caso de que la promesa haya finalizado con el método *resolve* se ejecutará el condigo que se encuentre en el *then*, sino entrará en el *catch*.

Por otra parte, la función async se utiliza para declarar que una función es asíncrona. En una función asíncrona se utiliza la palabra await para mantener la ejecución en espera hasta que se termine de ejecutar la función. En la imagen anterior se puede ver un ejemplo del uso de “async-await”. Se trata de una función asíncrona en la que se ha utilizado un *await* en la parte del código que realiza la consulta en la base de datos, ya que los datos obtenidos de dicha consulta son los que se tienen que devolver en el *resolve*.

6.2. Aplicación web

En esta parte se expondrá como se ha realizado la parte del proyecto referente a la aplicación web. Pero antes de ello se explicará cómo se ha realizado la instalación de las herramientas necesarias y la estructuración del proyecto.

6.2.1. Instalación y estructuración

Como se ha mencionado en apartados anteriores para el desarrollo de la página web se ha utilizado el framework de TypeScript AngularJS. La instalación de Angular es muy sencilla si ha se ha instalado previamente NodeJS con su gestor de paquetes NPM. Lo único que hay que hacer es ejecutar el siguiente comando en la consola:

```
npm install -g @angular/cli
```

Una vez instalado angular y creado el proyecto, con los comandos correspondientes, se puede proceder a abrirlo. En este caso se ha utilizado el entorno de desarrollo Visual Studio Code. Al abrir el proyecto se puede ver que el esqueleto del proyecto que ha generado angular es el siguiente:

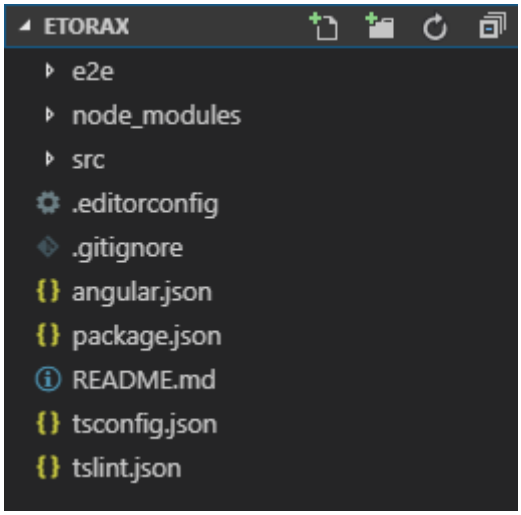


Ilustración 25 - Esqueleto proyecto angular

Todos los archivos que se ven en la imagen (que no están dentro de una de las carpetas) son archivos de configuración, menos el archivo de “readme.md”. La carpeta “node_modules” es una carpeta similar a la que existía en la parte de back-end, al igual que en la que existe en la de back-end en esta también se almacenan los módulos externos necesarios para el funcionamiento del código.

Por último, la carpeta “src” es donde se desarrolla la aplicación. A continuación, se muestra una imagen de esta carpeta antes de que esta sea modificada para adaptarla a las necesidades del proyecto.

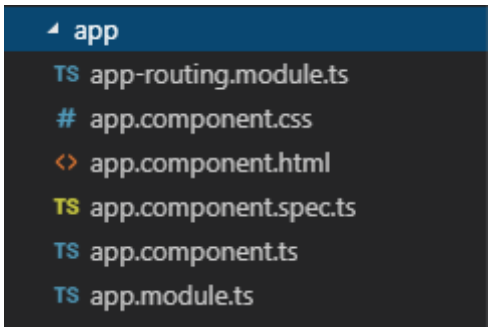


Ilustración 26 - Estructura inicial carpeta app

Como se puede ver en la imagen en esta carpeta vienen unos cuantos archivos por defecto de los cuales es importante sabes cuál es la funcionalidad de cada uno:

- App-routing.module.ts: En este fichero es donde se crearán las rutas de la aplicación web. Se relacionará el link con el componente que se ejecuta.

- App.module.ts: Es donde se han de declarar todos los componentes que formen parte de la aplicación web. El hecho de declarar los componentes en este fichero permite que los componentes puedan ser localizados por el navegador.

Sin embargo, en esta carpeta se ha realizado algunos cambios para adaptarla lo máximo posible a las necesidades del proyecto.

En primer lugar, se ha creado una carpeta llamada “componentes”, dentro de esta carpeta se introducirán todos los componentes que formen parte de la aplicación. Cada componente tendrá su carpeta dentro de esta, donde estarán los cuatro archivos que forman el componente.

También se ha creado la carpeta “guards”, en esta carpeta es donde se almacena el servicio que controlará que los usuarios no accedan a links a los que no tienen permiso para acceder.

Otra de las carpetas que se han creado es la de servicios cuyo nombre es “services”, en ella se almacenarán los servicios de los que hará uso la aplicación.

Por último, también se ha creado un fichero llamado “material.ts”. Durante el desarrollo la librería de angular material ha sido muy utilizada, por ello se ha decidido crear un nuevo fichero donde importar todos los componentes de angular material que se iban a usar. De modo que todos los componentes usados en el desarrollo pertenecientes a angular material han sido importados en este fichero y no en el fichero “app.module.ts”. Después el fichero “material.ts” ha sido importado en el fichero “app.module.ts”. De esta manera se ha favorecido el modularidad del código y la limpieza del mismo.

6.2.2. Inicio de sesión y tokens

Para poder hacer uso de la aplicación web es necesario iniciar sesión en la misma, solo lo usuarios que inicien sesión podrán navegar por la página web. Por ello se ha creado un inicio de sesión que permita a los médicos acceder al sistema.

Para iniciar sesión en el sistema los usuarios tendrán que introducir un nombre de usuario y una contraseña, los cuales se enviarán al servidor para que compruebe que los datos son correctos. En caso de que los datos introducidos por el usuario sean correctos el servidor le responderá al cliente con el token. Este token serán necesario enviarlo en cada futura petición que el cliente haga el servidor, por eso es importante almacenar el token de forma correcta.

Existen diferentes formas de almacenar el token, en este caso se ha decidido utilizar dos de ellas dependiendo si el usuario marca la opción de mantener la sesión iniciada al iniciar sesión o no.

En caso de no haber marcado dicha opción el token del usuario se almacenará en el objeto *sessionStorage.* En este caso el valor del token se eliminará de forma automática al cerrar la pestaña del navegador o de forma manual al cerrar sesión desde la aplicación.

En caso de que el usuario desee mantener la sesión iniciada el token de usuario se almacenará en el objeto *localStorage*. El objeto *localStorage* es muy parecido al *sessionStorage,* pero en este caso no se elimina al cerrar la pestaña del navegador, lo cual permite realizar las peticiones con el token de usuario sin que este haya tenido que introducir nuevamente el usuario y la contraseña. Al igual que en el caso anterior el token también se eliminará de forma manual si el usuario decide cerrar sesión.

6.2.3. Comunicación con el servidor

Las comunicaciones con el servidor se realizarán utilizando la clase HttlClient, del módulo “@angular/common/http” el cual ya viene instalado por defecto con angular. También se ha utilizado la clase HttpHeaders, la cual ha permitido crear las cabeceras de las peticiones que cliente realiza.

El primer paso que hay que realizar es importar ambas clases en el servicio que las va a utilizar, una vez importadas se añaden en el constructor para poder hacer uso de ellas:

```
import { HttpClient, HttpHeaders } from '@angular/common/http';
```

Ilustración 27 - Import HttpClient y HttpHeaders

La manera en la que hay que utilizar la clase HttpClient es muy similar a la explicada en el apartado de desarrollo del back-end de las promesas. Cuando HttpClient

realiza una petición al servidor este devuelve una suscripción al método que los llamo. En la siguiente imagen se muestra la forma en la que se ha realizado. Si la ejecución ha sido correcta se ejecutará la primera parte del código. En caso de que haya habido algún error se ejecutará la segunda parte del código. Como se puede ver en la primera parte del código hay una variable llamada "list", la cual es la respuesta que se ha obtenido del servidor.

```
this.api.getPacientesTodos().subscribe(
  list => {
    //No ha habido ningún error
  },error => {
    //Ha sucedido un error
  });
```

Ilustración 28 - Llamada a una petición HTTP

En la siguiente imagen se muestra como es el método al que se llama en la imagen anterior.

```
getPacientes(){
  let token= this.authService.getToken();
  const url_api = `http://localhost:3000/secured/pacientes`;
  return this.http.get(url_api,{headers: this.headers});
}
```

Ilustración 29 - Ejemplo petición HTTP

6.2.4. Rutas

Como se ha comentado anteriormente el fichero encargado de la gestión de las rutas es el "app-routing.module.ts". A continuación, se muestra un ejemplo de cómo han sido definidas las rutas en este fichero:

```
const routes: Routes = [
  {path: '', redirectTo: '/login',pathMatch: 'full'},
  {path: 'login', component: LoginComponent},
  {path: 'home', component: HomeComponent, canActivate: [AuthGuard]},
  {path: 'configuracion', component: ConfiguracionComponent, canActivate: [AuthGuard]},
```

Ilustración 30 - Ejemplo rutas página web

Como se puede ver en el ejemplo que se muestra en la imagen lo que se ha de hacer es relacionar el componente que se visualiza con el URL. En este fichero abría que definir todas las rutas del sistema a las que se quiere acceder.

Además, como se ve en el primer ejemplo existe la posibilidad de crear redirecciones, en este caso se ha creado una redirección de la ruta raíz a el path de *Login.* Otra de las cosas que se ven en el ejemplo presentado es que los dos primeros

path carecen de una tercera parte que dice: “canActivate: [AuthGuard]” mientras que el resto de paths si tienen esta tercera parte. La función de esta última parte es comprobar que el usuario tiene un token para entrar a la URL, en caso de que el usuario no posea un token se le negará el acceso a dicha página. Por este motivo los dos primeros path carecen de el “canActive”, ya que para acceder al Login no es necesario que el usuario tenga token.

Para controlar los casos en los que el usuario introduce un URL que no existe se ha creado una página con el error 404 que avise al usuario de que la dirección a la que está intentando acceder no existe. Esto se ha realizado con la siguiente línea de código al final de los path, es importante situarla al final ya que, si no, en caso de que el componente existiese se visualizaría este componente en lugar del correcto.

```
{path:'**', component: Page404Component}
```

Ilustración 31 - Ruta componente 404

6.2.5. Bootstrap

Bootstrap se trata de una librería externa para angular la cual facilita el desarrollo y el diseño de la aplicación web. Para el desarrollo de eTorax se han utilizado varias funcionalidades que ofrece Bootstrap.

Una de las funcionalidades utilizadas son las alertas. Se han decidido utilizar las alertas ofrecidas por Bootstrap ya que su implementación es muy sencilla y el resultado que ofrecen es muy estético. A continuación, se puede ver un ejemplo de una alerta durante el inicio de sesión.

Ilustración 32 - Ejemplo alerta Bootstrap

La aparición de estas alertas se ha realizado gracias a la propiedad de angular "*ngif" y al binding. El binding lo que permite es conectar una variable que se encuentra en el fichero en formato TypeScript del componente con el documento HTML. De esta forma el fichero HTML puede saber qué valor toma una determinada variable, y en combinación con el "*ngif", el cual muestra una etiqueta de HTML dependiendo del valor de una variable se consigue mostrar las alertas cuando es necesario.

Otro tipo de alertas de Bootstrap que también se han utilizado en eTorax son los "modals". Este tipo de alertas son diferentes a las mencionadas anteriormente, estas son como pop-ups que aparecen en pantalla. A continuación, se muestra un ejemplo de cómo se han utilizado los modals.

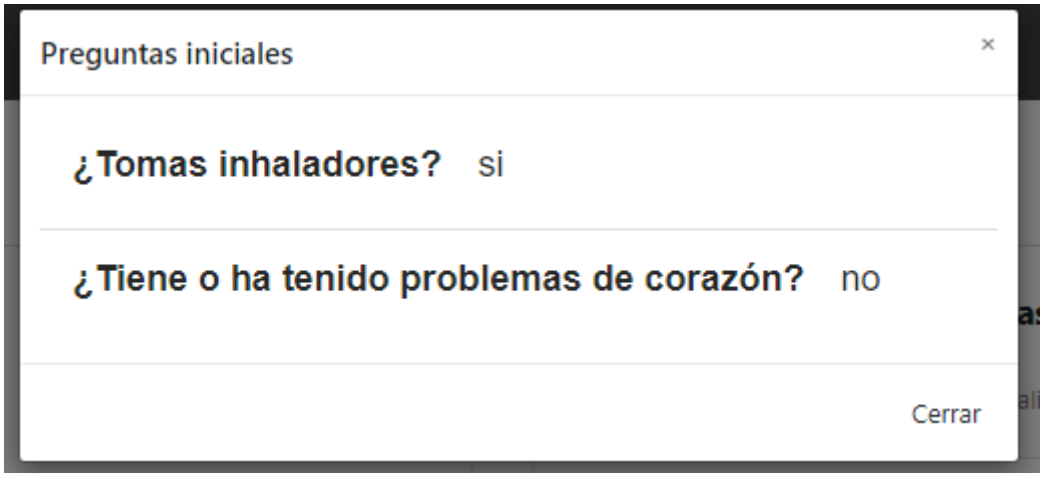


Ilustración 33 - Ejemplo alertas modal

Para crear este tipo de alertas primero hay que crear la plantilla que posteriormente se lanzará. Por lo tanto, este tipo de alertas son totalmente personalizables y se pueden usar para cualquier tipo de función. En este caso en eTorax se han utilizado además para lo que se puede ver en la imagen anterior para pedir la confirmación de los cambios o avisar de errores.

Otras de las funcionalidades de Bootstrap que se han utilizado es el "Grid". El Grid de Bootstrap divide el ancho de la vista en doce columnas imaginarias de la misma anchura y en líneas. Lo interesante de esto es que Bootstrap permite seleccionar el número de columnas que utiliza un elemente dependiendo del tamaño de la pantalla en la que se está visualizando la página web. De esta manera se consigue que la vista de la página web sea escalable a los distintos tamaños de pantalla con una sola línea de código. A continuación, se muestra el código utilizado para hacer que el inicio de sesión sea escalable a los distintos tamaños de pantalla:

```
<div class="row">
  <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3 mx-auto">
```

Ilustración 34 - Bootstrap columnas

En la imagen se puede ver como variará el tamaño del formulario para iniciar sesión según el tamaño de la pantalla. Por ejemplo, en caso de que se trate de una

pantalla “Extra small” (<576px) el formulario ocupara las 12 columnas de la pantalla y en caso de que la pantalla sea “large” o “extra large” (>992px) el formulario solo ocupará tres de las doce columnas de la pantalla. (Bootstrap, s.f.)

6.2.6. Exportación de datos

El cliente del proyecto necesitaba que los datos almacenados en la página web se pudiesen exportar de alguna forma. Para ello se decidió utilizar el formato CSV para poder exportar los datos almacenados por los usuarios en la base de datos del proyecto.

Para exportar los datos a CSV se ha decidido utilizar la librería disponible para angular llamada “export-to-csv”, tanto por su facilidad de uso como por que sin necesidad de generar más código se encarga de abrir la ventana para guardar el CSV.

La librería transforma los datos de formato JSON a formato CSV. A continuación, se muestra cual es la manera de exportar los datos a CSV con esta librería.

Lo primero que hay que hacer es configurar las opciones de configuración para la exportación. A continuación, se muestra un ejemplo de cómo se pueden configurar las opciones. Como se puede ver en estas opciones se puede definir desde cómo se separarán los datos hasta cual será el título del CVS.

```
const options = {
  fieldSeparator: ',',
  quoteStrings: '"',
  decimalSeparator: '.',
  showLabels: true,
  showTitle: true,
  title: 'CSV',
  useTextFile: false,
  useBom: true,
  useKeysAsHeaders: true,
};
```

Ilustración 35 - Opciones CSV

Una vez definidas las opciones habrá que crear un nuevo objeto ExportToCsv con las opciones definidas. Y una vez creado dicho objeto bastará con llamar al método “generateCsv” del objeto con los datos. A continuación, se muestra como se ha realizado.

```
const csvExporter = new ExportToCsv(options);
csvExporter.generateCsv(list['body']);
```

Ilustración 36 - Exportar a CSV

Como se puede ver en la imagen los datos que se pasan al llamar al método "generateCSV" son los obtenidos de la base de datos directamente, ya que estos datos se encuentran en formato JSON.

En este punto se abrirá la ventana para guardar un archivo de forma automática y el usuario podrá seleccionar donde desea que se guarde el archivo.

6.2.7. Aplicación multiidioma

Pese a que en un principio no era un requisito fundamental, dada la posibilidad de que en un futuro el proyecto creciese y se expandiese se ha decidido que el proyecto estará orientado a ser multiidioma.

En este caso se ha decidido hacer que la aplicación sea multiidioma utilizando la librería de angular "ngx-translate/core". El primer paso que se ha realizado para hacer que la aplicación sea multiidioma es instalar la librería.

Una vez se ha instalado la librería el siguiente paso es importar los módulos que se van a utilizar en el fichero "app.module.ts", en este caso "TranslateModule" y "TranslateLoader". A continuación, se muestra en la imagen como han sigo importados, ya que se realiza de una forma diferente al resto de los módulos importados.

```
TranslateModule.forRoot({
  loader: {
    provide: TranslateLoader,
    useFactory: HttpLoaderFactory,
    deps: [HttpClient]
  }
})
```

Ilustración 37 - Import de módulo de traducción.

Una vez realizados los pasos anteriores es importante definir en el fichero "app.component.ts" cuáles son los idiomas disponibles en la aplicación y cuál de ellos es el idioma por defecto de la aplicación. En la siguiente imagen se ve como se ha definido en el constructor lo anteriormente mencionado, como se ve, los idiomas disponibles de la aplicación serian el castellano y el inglés siendo el primero el idioma por defecto.

```
constructor(public translate: TranslateService){
  this.translate.addLangs(['es','en']);
  this.translate.setDefaultLang('es');
}
```

Ilustración 38 - Constructor multiidioma

Por último, se ha creado la carpeta “i18n” dentro de la de “assets”, donde se creará un fichero por cada uno de los idiomas de los que dispone la aplicación. Estos ficheros serán ficheros JSON los cuales por cada palabra aparecerá una key y el texto en el idioma correspondiente. A continuación, se muestra como es el fichero de castellano.

```
{
    "hello": "hola",
    "etorax":"etorax",
    "idioma":"Idioma:",
    "cambiarIdioma":"Cambiar idioma",
```

Ilustración 39 - JSON multiidioma castellano.

6.2.8. Angular material

Angular material se trata de una librería para angular la cual ofrece múltiples componentes. Gracias a estos componentes se puede agilizar el desarrollo de la aplicación y mejorar la estética de la misma.

Durante el desarrollo de la aplicación se han utilizado múltiples componentes que ofrece angular material como por ejemplo las tablas, los botones o las barras de búsqueda. Como se ha comentado en apartados anteriores debido al volumen de componentes utilizados de esta librería lo más adecuado era crear un fichero independiente en el que importar todos los componentes de angular material que se usarían. Posteriormente importar dicho fichero al fichero “app.module.ts”. A continuación, se muestra un ejemplo en el que todos los componentes utilizados son de angular material.

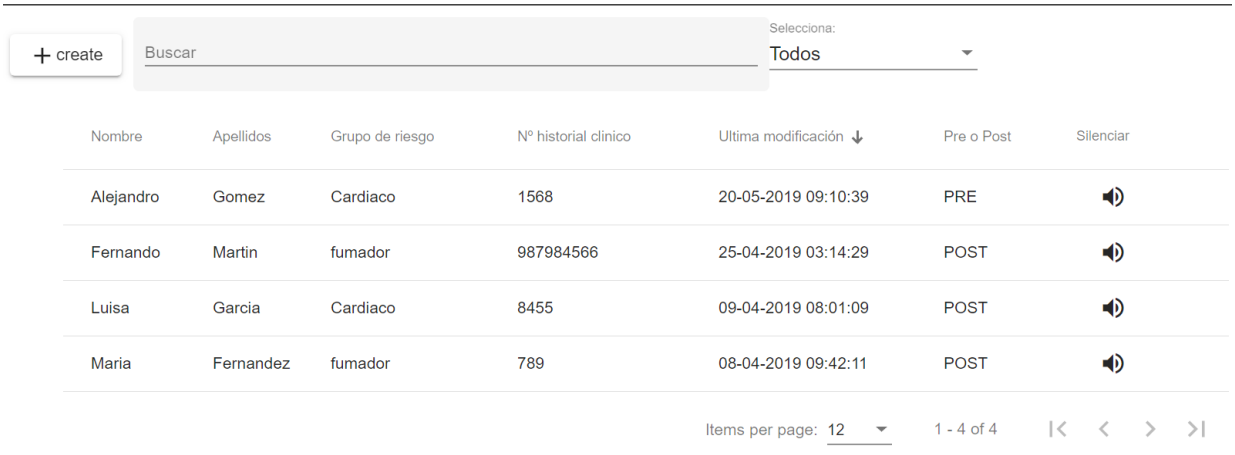


Ilustración 40 - Ejemplo componentes angular material

De la ilustración anterior tanto el botón para crear pacientes, la barra de búsqueda, el desplegable y la tabla son componentes que ofrece angular material que se han utilizado en el desarrollo de eTorax.

Uno de los componentes de angular material que más protagonismo tiene en la aplicación son las tablas, las cuales se usan en tres ocasiones distintas. El primer paso para crear una tabla como se ve en la imagen anterior es importar todos los componentes que van a participar en la creación de la misma. A continuación, se muestra el ejemplo de cuales han sido los componentes importados para crear la tabla de la imagen anterior. En este caso como se desea que la tabla sea ordenable por columnas y que tenga paginación se añaden los componentes "MatSort" y "MatPaginator".

```
import { MatTableDataSource,MatSort,MatPaginator, Sort } from '@angular/material';
```

Ilustración 41 - Import para tabla de angular material.

Una vez importados todos los componentes que se van a utilizar el siguiente paso es definirlos. Como se puede ver en la siguiente imagen también es necesario definir cuáles son las columnas que se desea que se impriman en la tabla.

```
listData: MatTableDataSource<any>;
displayedColumns:string[] = ['nombre','apellidos','grupo','numHist','ultimaMod','PreoPost',
                             'inspeccionar','silenciar'];

@ViewChild(MatSort) sort: MatSort;
@ViewChild(MatPaginator) paginator: MatPaginator;
```

Ilustración 42 - Definición componentes tabla

Otro de los puntos a tener en cuenta es que la tabla no se puede crear simplemente con los datos obtenidos desde la petición SQL. Antes de crear la tabla hay que cambiarla a formato "MatTableDataSource". En la siguiente imagen se ve como se ha realizado. En este caso como se desea que la tabla sea ordenable por columnas y que exista un paginador también es necesario añadir los objetos "MatSort" y "MatPaginator" a el "MatTableDataSource".

```
this.listData = new MatTableDataSource(array);
this.listData.sort = this.sort;
this.listData.paginator = this.paginator;
```

Ilustración 43 - Creación MatTableDataSource

También es importante crear la tabla que se va a mostrar en la vista del componente, es decir en el documento HTML. Para ello habrá que definir una "mat-table" y dentro de la tabla definir cada una de las columnas que se mostrarán. En la siguiente imagen se puede ver un ejemplo de cómo ha sido definida la tabla y la primera columna. En la etiqueta de la tabla es importante definir el campo "[dataSource]=" con la tabla en formato "MatTableDataSource" definida en el fichero de lógica del componente. Hay que tener en cuenta que las columnas definidas en el documento

HTML deben tener el mismo nombre que las columnas definidas en la variable "displayedColumns" del fichero de TypeScript del componente.

```
<table mat-table class="full-width-table" [dataSource]="listData" matSort (matSortChange)="sortData($event)"
 matSortActive="ultimaMod" matSortDirection="desc" matSortDisableClear aria-label="Elements">

  <ng-container matColumnDef="nombre">
    <th mat-header-cell *matHeaderCellDef mat-sort-header>Nombre</th>
    <td mat-cell *matCellDef="let row">{{row.nombre}}</td>
  </ng-container>

  <ng-container matColumnDef="apellidos">
    <th mat-header-cell *matHeaderCellDef mat-sort-header>Apellidos</th>
    <td mat-cell *matCellDef="let row">{{row.apellidos}}</td>
  </ng-container>
```

Ilustración 44 - Tabla HTML

Por último, en el documento HTML también es necesario definir el paginador. Este ofrece la posibilidad de personalizarlo, se puede elegir el número de opciones de paginación que se desea que existan y el valor de filas por defecto.

```
<mat-paginator [pageSizeOptions]="[12, 25, 100, 250]" [pageSize]="12" showFirstLastButtons></mat-paginator>
```

Ilustración 45 - Paginador angular material

6.2.9. Gráficos

Una de las funciones que se decidió implementar en la aplicación fue la de poder visualizar gráficos de cómo es la progresión del paciente a lo largo del tiempo. Esta funcionalidad se ha implementado utilizando la librería ChartJS.

Para crear un gráfico con la librería ChartJS es necesario crear una etiqueta "canvas" en el documento HTML, esta etiqueta tendrá que tener un identificador con el que poder referirnos desde el documento de lógica para crear el gráfico. A continuación, se muestra como debe ser definida la etiqueta en la vista:

```
<div id="divGrafico">
    <canvas id="canvas"></canvas>
</div>
```

Ilustración 46 - Etiqueta canvas HTML

Una vez creada la etiqueta en el documento HTML se obtiene desde el fichero .ts y se crea el grafico correspondiente. En la siguiente imagen se muestra un ejemplo de cómo se han creado los gráficos.

```
var canvas: any = document.getElementById('canvas');
var ctx = canvas.getContext('2d');
new Chart(ctx, {
type: "line",
data: {
  labels: this.fechasFilt,
  datasets: [{···
},
options: {responsive:true,
  maintainAspectRatio: false,
  annotation: {
    annotations: [
      {
        type: "line",
        mode: "vertical",
        scaleID: "x-axis-0",
        value: this.fechasFilt[],
        borderColor: "blue",
        label: {···
        }
      },
```

Ilustración 47 - Ejemplo creación del gráfico

Como se puede ver en la imagen lo primero que se hace es obtener la etiqueta del fichero HTML para poder trabajar sobre ella. También se puede ver que para crear el grafico existen tres atributos: type, data, option

En el atributo type en donde hay que definir qué tipo de gráfico se desea representar, por ejemplo, en este caso es un gráfico de “line” pero existen otras opciones como podrían ser los diagramas de barras o gráficos de área.

El atributo data, el cual tiene otros dos atributos en su interior, se ha de introducir los datos con los que se desea que se rellene el grafico. En el atributo label se han colocado las fechas, lo cual hará que estas estén en el eje X del gráfico. En el atributo data de datasets es donde se colocarán los datos del eje Y, en este caso la distancia recorrida. Tanto este último atributo como labels han de ser un Array de datos.

Por último, está el atributo “option” en el cual hay que definir las opciones de configuración. Algunas de las opciones que se pueden añadir en este apartado son el hecho de que el grafico se adapte su tamaño de forma automática o cual se desea que sea la escala de los ejes.

Además, como se puede ver en la imagen anterior en el atributo “option” hay otro gran atributo llamado “annotations”. Este atributo no pertenece a ChartJS por defecto, sino que hay que instalar un plug-in para poder utilizarlo. La instalación de este plug-in es muy sencilla, simplemente hay que introducir el siguiente comando en la consola.

```
npm i chartjs-plugin-annotation
```

Una vez ejecutado el comando anterior en la consola se dispondrá del plug-in. Este plug-in permite incluir anotaciones en los gráficos. Como se puede ver en la imagen anterior en el grafico se han incluido dos anotaciones, la cuales son del tipo línea y una de ellas es en vertical y la otra en horizontal. La primera de las anotaciones (la que es en vertical) se utiliza para marcar la fecha en la que el paciente se ha operado o se operará, como se puede ver la forma de marcarlo será mediante una línea vertical de color azul que dividirá en gráfico. La segunda anotación (la que es en horizontal) se utilizará para marcar el objetivo de pasos asignados por el médico.

A continuación, se muestra el ejemplo del grafico que se obtendrá como resultado de la ejecución del código anterior.

Ilustración 48 - Ejemplo grafico ChartJS

6.3. Aplicación móvil.

En este apartado se expondrá como se ha desarrollado la parte del proyecto referente a la aplicación móvil que usarán los pacientes. Pero antes de nada se expondrá como se ha realizado el proceso de instalación y estructuración del proyecto.

6.3.1. Instalación y estructuración

El primer paso a la hora de instalar las herramientas necesarias para desarrollar la aplicación en Xamarin es descargar e instalar Visual Studio Community. Tras descargar el instalador de Visual Studio Community y ejecutarlo el propio instalador pide al usuario que seleccione los complementos con los que desea instalarlo. Para el desarrollo de eTorax se han seleccionado los siguientes complementos:

- Movile delvelopement with .NET.
- Universal Windows Platform developement.
- .Net descktop developement
- ASP.NET and web developement.

Después de seleccionar los componentes necesarios comenzará la instalación automática. Es posible que tarde varias horas hasta que la instalación se haya finalizado.

Una vez la instalación se ha finalizado ya se puede comenzar con la creación del proyecto. Para ello se selecciona la opción de nuevo proyecto y se abrirá una pestaña donde se podrá crear el proyecto.

En el creador de proyectos es importante seguir una serie de pasos para que el proyecto se cree de forma correcta. Para crear el proyecto de eTorax los pasos son los siguientes:

En la pestaña para crear proyectos en la parte izquierda se ha de hacer click sobre el desplegable de “Visual C#” y en la opción de “Cross-Platform”, una vez se ha realizado click sobre “Cross-Platform” en la parte central de pestaña aparecerán las opciones disponibles, de las cuales habrá que seleccionar “Aplicación móvil (Xamarin Forms)”

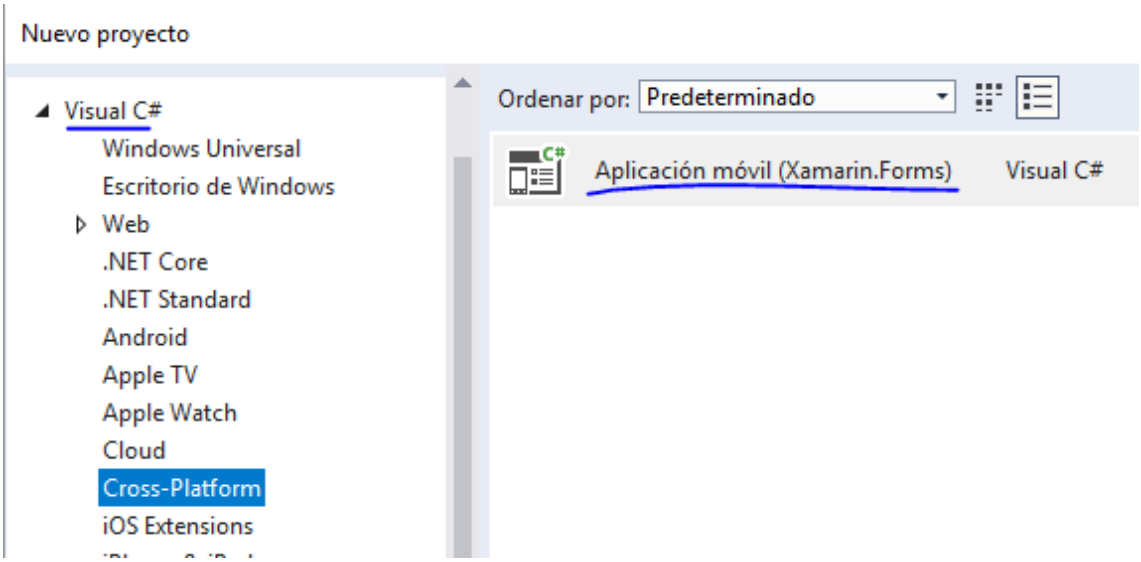


Ilustración 49 - Creación App Xamarin

Tras seleccionar el nombre de la aplicación y pulsar el botón de aceptar se pasará a la siguiente pestaña de creación de proyectos. Es esta nueva pestaña habrá que seleccionar que se desea crear un nuevo proyecto en blanco, los sistemas operativos para los cuales se va a desarrollar la aplicación y en la estrategia de uso compartido se seleccionará “.NET Standar”.

Tras hacer click sobre el botón de “OK” el proyecto ya se habrá creado, pero sin embargo todavía es necesario crear la estructura del proyecto. Como ya se ha comentado anteriormente en el apartado de Diseño, se ha decidido usar el patrón MVVM por lo que habrá que crear las carpetas necesarias: “Views”, “Models” y “ViewModels”. Dentro de la carpeta “ViewModels” se ha creado el archivo “MainViewModel”, el cual es un Singleton encargado de gestionar el resto de ViewModels.

Para utilizar el patrón de diseño MVVM se ha instalado el paquete “MvvmLigthLibs” desde el administrador de paquetes que ofrece Visual Studio, el cual permite implementar el patrón de forma sencilla.

A parte de las carpetas necesarias para realizar el patrón MVVM también se ha creado la carpeta “Services” donde estarán los servicios de los que hará uso la aplicación como el “ApiServices”, el cual realizará las peticiones al servidor.

Además, durante el desarrollo se han creado otras carpetas necearías para las diferentes funcionalidades las cuales se explicará en sus respectivos apartados.

6.3.2. Peticiones al servidor

Uno de los requisitos fundamentales de la aplicación es que esta se debe comunicar con el servidor. Esta comunicación se realiza mediante peticiones HTTP. Para que la aplicación pueda realizar dichas peticiones se han instalado los siguientes paquetes:” Xamarin.Net.Http” y “Newtonsoft.Json”.

Xamarin.Net.Http es una librería creada por Microsoft la cual permite realizar peticiones HTTP al servidor.

Aunque Newtosoft.Json no sirva para realizar peticiones HTTP es necesario para convertir los datos de la respuesta de la petición de formato JSON a objetos creado en la aplicación.

Una vez se ha explicado cuales son las librerías necesarias para comunicarse con el servidor se explicará cómo se han realizado las peticiones.

Es importante mencionar que antes de realizar cada petición al servidor la aplicación comprueba que el móvil tenga conexión a internet. A continuación, se muestra cual es la manera en la que se comprueba si existe conexión.

```
var con = await this.apiService.CheckConnection();
if (!con.IsSuccess)
{
    await Application.Current.MainPage.DisplayAlert(
        "Error",
        con.Message,
        "Aceptar");
    return;
}
```

Ilustración 50 - Comprobar conexión a internet

Como se puede ver en la imagen se realiza una llamada al método “CheckConnecction” que se encuentra en “ApiServices” el cual comprueba si la conexión con internet del móvil esta activada. Si la conexión a internet del teléfono móvil esta activada procede a realizar un “Ping” a Google. Si tras comprobar que se dispone de conexión a internet existe la conexión se procede a realizar la llamada al servidor, pero si al comprobar que existe conexión a internet realmente no se dispone de conexión

al usuario le saltará una alerta informándole para que active la conexión o informándole que no tiene conexión.

En caso de que se haya comprobado la existencia de la conexión a internet y el resultado haya sido satisfactorio se procederá a realizar la petición al back-end.

A continuación, se muestra una imagen en la que se puede ver el ejemplo de cómo se realizaría la petición al servidor para iniciar sesión en la aplicación.

```
public async Task<RespServer> Login(string username,string password)
{
    try
    {
        var formContent = new FormUrlEncodedContent(new[]...);
        var client = new HttpClient();
        var response = await client.PostAsync("https://etorax.herokuapp.com/login/paciente", formContent);
        var resultJSON = await response.Content.ReadAsStringAsync();
        var result = JsonConvert.DeserializeObject<RespServer>(resultJSON);
        return result;
    }
    catch (Exception e)
    {return null;}
}
```

Ilustración 51 - Ejemplo petición HTTP Xamarin

Como se puede ver en la imagen los métodos del servicio encargado de realizar las peticiones a el servidor se tratan de métodos asíncronos. Es necesario que estos métodos sean asíncronos ya que al realizar una petición al servidor no se sabe lo que este puede tardar en responder.

Lo primero que hay que hacer para realizar la petición al servidor es codificar los parámetros que formarán parte de la petición. En este caso al tratarse de la petición para realizar el inicio de sesión los parámetros enviados en la petición son el usuario y la contraseña. A continuación, se puede ver como se realiza la codificación de los parámetros. Como se puede ver cada uno de los parámetros debe ir acompañado de una clave. Formando así pares de clave y valor.

```
var formContent = new FormUrlEncodedContent(new[]
    {
     new KeyValuePair<string, string>("user", username),
     new KeyValuePair<string, string>("pass", password),
     });
```

Ilustración 52 - Codificación de los parámetros de la petición en Xamarin

Después de codificar los parámetros que se enviarán al servidor, para esto lo que se debe realizar es instanciar la clase “HttpClient”, la cual permitirá realizar la petición.

Una vez instanciada la clase se usará pasa llamar a uno de los métodos HTTP existentes. Los métodos de la clase HttpClient se llaman igual que los métodos HTTP, pero sucedidos por la palabra “Async”. Por ejemplo, en este caso se desea usar el método POST por lo que para ello habrá que hacer una llamada al método “PostAsync” de la clase HttpClient. Para ejecutar este método se deben pasar como parámetros la URL a la que se desea realizar la petición y los parámetros de la petición codificados.

Esta petición debe ir precedida por la palabra ”await” ya que la petición se encuentra en un método asíncrono y se desea esperar la respuesta del servidor.

Una vez se ha obtenido la respuesta de la petición se lee en formato JSON. Cuando la respuesta se encuentra en formato JSON se hará uso de la librería “Newtonsoft.Json” para pasar dicha respuesta de formato JSON a un objeto que la aplicación pueda manejar fácilmente. En este punto la respuesta de convertirá a un objeto de alguna de las clases declaradas en la carpeta “Models”.

6.3.3. Control de la distancia caminada

Una de las funcionalidades más importantes que debía cumplir la aplicación es la de llevar un control de la distancia caminada por el paciente y el almacenamiento de los datos recogidos en la base de datos.

En este apartado se explicará cómo se ha realizado el control de la distancia caminada por el paciente.

Lo primero que se debe realizar para poder geolocalizar el dispositivo es instalar la librería “Xamarin.Essentials” la cual se trata de una librería que está formada por varias librerías que antes se encontraban de forma independiente. Una de las librerías que forman Xamarin.Essentials es la librería de geolocalización. Se ha decidido utilizar esta librería ya que se trata de una librería oficial de Xamarin desarrollada por Microsoft y permite realizar la geolocalización en el código común del proyecto de manera que solo haya que desarrollar el código una vez, mientras que otras librerías solo están desarrolladas para uno de los sistemas operativos por lo que habría que desarrollar la función para cada uno de ellos.

Una vez instalada la librería se deben configurar los permisos necesarios. Esto de hace de forma diferente para cada una de las plataformas.

Para configurar los permisos en el sistema operativo Android se debe incluir el código que se muestra en la siguiente imagen en el archivo “AssemblyInfo.cs” que se encuentra en la carpeta de propiedades del proyecto de Android.

```
[assembly: UsesPermission(Android.Manifest.Permission.AccessCoarseLocation)]
[assembly: UsesPermission(Android.Manifest.Permission.AccessFineLocation)]
[assembly: UsesFeature("android.hardware.location", Required = false)]
[assembly: UsesFeature("android.hardware.location.gps", Required = false)]
[assembly: UsesFeature("android.hardware.location.network", Required = false)]
```

Ilustración 53 - Permisos de geolocalización Android

Para el Sistema operativo IOS el archivo “Info.plist” debe contener la clave “NSLocationWhenInUseUsageDescription” para que la aplicación pueda acceder a la ubicación del dispositivo. Esto se puede hacer mediante el editor de derechos de IOS o agregando el siguiente código en el archivo XML.

```
<key>NSLocationWhenInUseUsageDescription</key>
<string>This app needs access location when open.</string>
```

Ilustración 54 - Permisos de geolocalización de IOS

Para establecer el permiso de geolocalización en la plataforma universal de Windows habrá que abrir el archivo "Package.appxmanifest" y en la pestaña de "Capacidades" activar la ubicación.

De esta forma todos los sistemas operativos tendrían ya los permisos necesarios para que la aplicación geolocalice el dispositivo.

Una vez gestionados los permisos necesarios ya se puede obtener la ubicación del dispositivo mediante geolocalización. Para obtener la ubicación del dispositivo se ha utilizado el código que se ve en la siguiente imagen.

```
try
{
    var request = new GeolocationRequest(GeolocationAccuracy.Best);
    var location = await Geolocation.GetLocationAsync(request);
    if (location != null)...
}
catch (FeatureNotSupportedException fnsEx)
{
    El dispositivo no lo soporta
}
catch (FeatureNotEnabledException fneEx)
{
    No se puede obtener en este dispositivo
}
catch (PermissionException pEx)
{
    Excepcion permiso
}
catch (Exception ex)
{
    No se pede obtener
}
return null;
```

Ilustración 55 - Obtener localización

Como se puede ver en la imagen lo primero que se debe realizar es instanciar la petición de geolocalización. Para ello se crea un objeto de la clase "GeolocatioRequest" a la que habrá que pasar como parámetro la exactitud con la que se desea obtener la posición del dispositivo. En este caso se desea obtener la mejor posición posible.

Tras crear la petición ya se puede obtener la localización del dispositivo, para ello habrá que usar el método "GetLocationAsync" el cual devuelve la ubicación del dispositivo. Sin embargo, es posible que este método devuelva null porque no se ha podido obtener la localización. Si el valor recibido por el método no es null se comprobará que la localización obtenida no sea una localización ficticia. Para ello se usa el método "IsFromMockProvider" el cual devuelve un booleano.

Como se puede ver en la imagen la parte del código en la que se obtiene la localización del dispositivo está rodeada por un “Try Catch” ya que este proceso puede dar varias excepciones. En este caso se capturan distintos tipos de excepciones que podrían saltar, en cada una de las capturas de las excepciones se encuentra una alerta que explicará al usuario cual es el motivo por el que no se puede obtener la ubicación. Por ejemplo, si la aplicación no cuenta con los permisos necesarios al usuario le saltará una alerta diciéndole que debe conceder los permisos de ubicación a la aplicación.

En este punto la aplicación es capaz de obtener la ubicación en la que se encuentra el dispositivo sin embargo se necesita que la aplicación obtenga la distancia caminada a lo largo de la caminata del paciente. Para ello la aplicación ira obteniendo la ubicación del dispositivo de forma continua. Pero pese a que como se ha visto anteriormente se ha indicado que se desea obtener la ubicación con la mayor precisión posible esta no es suficiente. En Android puede existir un error de hasta 100 metros entre la posición real del dispositivo y la obtenida por la aplicación (Microsoft, 2019). Por ese motivo la aplicación calculará cual ha sido la velocidad necesaria para recorrer la distancia entre los dos últimos puntos obtenidos, en caso de que la velocidad necesaria para recorrer la distancia que separa los dos últimos puntos sea mayor que la velocidad promedio de una persona trotando se descartará el último punto. De esta manera se ha logrado aumentar considerablemente la precisión de la distancia obtenida por la aplicación. Además de esta manera si alguno de los pacientes intenta hacer trampas recorriendo la distancia asignada por el médico en coche se detectará que la velocidad es demasiado alta y la distancia recorrida no se tendrá en cuenta.

6.3.4. Gifs

Los clientes de la aplicación deseaban que existiese la posibilidad se mostrar a los usuarios de la aplicación una serie de recomendaciones. Estas recomendaciones que se mostrarían a los pacientes debían ir acompañadas por un gif el cual mostrase de forma gráfica la recomendación para que de esta forma fuese más sencillo de entender.

Para poder mostrar los gifs de las recomendaciones se han utilizado dos librerías las cuales funcionan juntas. Las librerías utilizadas son “Xamarin.FFImageLoading.Svg” y “Xamarin.FFImageLoading.Svg.Forms”. Se ha decidido utilizar estas librerías ya que permiten desarrollar el código para mostrar los gifs en el código común del proyecto y de esta forma no hay que desarrollar el código para cada una de las aplicaciones. Sin embargo, el resto de librerías están desarrolladas para uno de los sistemas operativo o como mucho para dos de ellos.

Una vez instaladas las librerías es necesario añadir una sentencia en el código especifico de cada sistema operativo para que se puedan utilizar. Algo parecido a lo explicado en el apartado anterior con los permisos.

Para IOS y UWP la sentencia que habrá que añadir es la misma, la que se muestra en la siguiente imagen, pero en cada uno de los proyectos se ha de situar en un archivo diferente.

Para IOS la sentencia habrá que añadirla en el archivo “AppDelecate.cs” dentro del método “FinishedLaunching”. Y para UWP habrá que añadir la sentencia en el archivo “MainPage.xalm.cs” dentro del constructor.

```
FFImageLoading.Forms.Platform.CachedImageRenderer.Init();
```

Ilustración 56 - Sentencia para utilizar FFImageLoading en IOS y UWP

Para Android la sentencia que se debe añadir es la que aparece en la siguiente imagen. Esta vez habrá que añadir la sentencia en el archivo “MainActivity.cs”, dentro del método “OnCreate”.

```
FFImageLoading.Forms.Platform.CachedImageRenderer.Init(enableFastRenderer: true);
```

Ilustración 57 - Sentencia para utilizar FFImageLoading en Android

Una vez incluidas estas sentencias en cada uno de los proyectos el proceso restante para visualizar los gifs en la aplicación es bastante sencillo.

En archivo de la vista en la cual se desea mostrar el gif se debe añadir la siguiente sentencia en la etiqueta “ContentPage”.

```
xmlns:ff="clr-namespace:FFImageLoading.Forms;assembly=FFImageLoading.Forms"
```

Ilustración 58 - ContentPage gif

En este punto el archivo ya está preparado para mostrar los gifs, para ello solo hay que introducir la etiqueta que mostrará el gif.

```
<ff:CachedImage x:Name="Gif" Source="{Binding Recomendacion.gif}"/>
```

Ilustración 59 - Etiqueta de gif

En este caso la fuente del gif se encuentra “bindada” al atributo gif del objeto Recomendación ya que se desea que con cada recomendación se muestre un gif diferente.

Es importante mencionar que los gifs al igual que el resto de las imágenes de la aplicación se tienen que almacenar en los proyectos de cada uno de los sistemas operativos. En Android las imágenes de la aplicación y los gifs se deben almacenar en la carpeta “drawable” dentro de la carpeta “Resources”. En IOS las imágenes y los gifs se deben almacenar directamente en la carpeta “Resource” y en UWP se deben almacenar en la carpeta raíz del proyecto.

6.3.5. Código diferente pasa cada sistema operativo

Como se ha mencionado en apartados anteriores en Xamarin la mayoría del código se implementa en el proyecto de código compartido. Sin embargo, hay determinadas funcionalidades que han de ser desarrollada específicamente para cada

uno de los sistemas operativos, ya que estas funcionalidades hacen uso de recursos específicos del sistema operativo en el que se están ejecutando.

Para poder crear este tipo de funcionalidades es necesario crear una interfaz en el proyecto de código compartido con las funciones que se van a ejecutar de forma diferente en cada sistema operativo. A continuación, se muestra la interfaz "ILocalize" que se usa para obtener la localización del dispositivo, lo cual se usará en la funcionalidad de multiidioma que se explica en el siguiente apartado.

```
public interface ILocalize
{
    CultureInfo GetCurrentCultureInfo();
    void SetLocale(CultureInfo ci);

}
```

Ilustración 60 - Interfaz ILocalize

Una vez creada la interfaz de debe desarrollar el código de las funciones para cada uno de los sistemas operativos. Todos los métodos de la interfaz deben ser desarrollados cada uno de los sistemas operativos. Es importante introducir la sentencia que se muestra en la siguiente ilustración antes de la apertura de "namespace" sustituyendo "Droid" por el sistema operativo en cuestión.

```
[assembly: Dependency(typeof(etorax.Droid.Providers.Localize))]
```

Ilustración 61 - Dependencia Android

Con los métodos ya implementados ya se puede realizar la llamada desde el código común del proyecto. para ello se usa el método Get de la clase "DependencyService". A continuación, se muestra un ejemplo en el que se pude ver cuál es la forma de utilizarlo.

```
ci = DependencyService.Get<ILocalize>().GetCurrentCultureInfo();
```

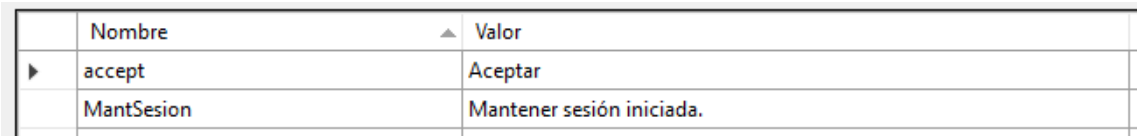
Ilustración 62 - Llamada a código especifico de cada SO

6.3.6. Multiidioma

Para facilitar la usabilidad de la aplicación a cualquier persona se ha decidido que la aplicación desarrollada sea multiidioma, es decir la aplicación cambiará de idioma dependiendo del idioma en el que el usuario tenga su dispositivo móvil.

Para ello además de las carpetas que ya existen el en proyecto se han creado otras carpetas necesarias como podrían ser la carpeta "Resources" donde se almacenarán los archivos de cada uno de los idiomas disponibles en la aplicación, la carpeta "Providers" y la carpeta "Helpers"

Para crear los archivos de cada uno de los idiomas almacenados en la carpeta “Resources” se deberá crear un tipo de archivo especifico, los cuales se pueden encontrar en la carpeta de añadir un nuevo elemento, en la pestaña de “General” con el nombre de recursos. Es importante que el archivo del idioma principal de la aplicación sea “Resource.resx” y los archivos del resto de idiomas sean “Resource.XX.resx” donde XX es el código del idioma. También es importante cambiar el “Modificador de acceso” del archivo del idioma principal y ponerlo en público. A continuación, se muestra un ejemplo de cómo es uno de estos archivos de recursos.



	Nombre	Valor
▶	accept	Aceptar
	MantSesion	Mantener sesión iniciada.

Ilustración 63 - Archivo de idioma

También se debe una clase llamada “Languages” en la carpeta “Helpers” en la cual se deben encontrar cada uno de los literales que se encuentran en los archivos “Resouerces” como se ve en la siguiente imagen.

```
public static string accept
{
    get { return Resource.accept; }
}
public static string MantSesion
{
    get { return Resource.MantSesion; }
}
```

Ilustración 64 - Archivo Languages

En este punto hay que crear la función que detecta el idioma en el que se encuentra el dispositivo y cambia el idioma de la aplicación en función del mismo. Para desarrollar esta función se ha utilizado código de terceros extraído de https://github.com/Zulu55/Lands.git. Aun así, cabe mencionar que esta parte del código se debe desarrollar para cada una de las aplicaciones, por ese motivo se ha creado la interfaz “ILocalize.cs” en la carpeta “Providers” y luego se han desarrollado las funciones en cada sistema operativo. También es importante mencionar que para IOS se debe seleccionar cual es el idioma por defecto en el archivo “Info.plist” así como cuales son los idiomas disponibles de la aplicación.

En este punto los literales que se usen desde la ViewMode ya se encuentran traducidos a los diferentes idiomas. A continuación, se muestra un ejemplo de una alerta. Como se ve en la imagen basta con llamar al método de la clase “Languages” del cual se quiere obtener el literal.

```
await Application.Current.MainPage.DisplayAlert(
    Languages.error,
    Languages.rellenarCampo,
    Languages.accept);
return;
```

Ilustración 65 - Parametrización multiidioma ViewModel

Sin embargo, las palabras que se encuentran en los archivos de vistas no se traducen con lo realizado anteriormente. Para que se pueda traducir las palabra de las vistas es necesario crear la clase “TranslateExtension” cuyo código también ha sido extraído de https://github.com/Zulu55/Lands.git esta clase será la encargada de obtener los valores de los literales de los archivos de recursos en el idioma correcto y pasárselos a las vistas.

Una vez se ha creado la clase “TransalateExtension” es necesario informar a cada una de las vistas en las que se va a incluir literales que se desea que se traduzcan donde se encuentra dicha clase. Para ello se debe añadir la siguiente línea en el “ContentPage” de la vista. Como se puede ver en la imagen en este caso la clase “TranslateExtension” se ha situado en la carpeta “Helpers” del proyecto “etorax”

```
xmlns:i18n="clr-namespace:etorax.Helpers"
```

Ilustración 66 - Traducir vistas 1

Con la sentencia anterior introducida en el “ContentPage” de la vista ya se puede traducir cada uno de los literales de la vista. A continuación, se muestra un ejemplo de cómo se traduce el contenido de una etiqueta “Label“. Para ello solo es necesario usar la sentencia {i18n:Translate XX} donde XX es la clave del valor de los archivos de recursos que se desea que se introduzca.

```
<Label Text="{i18n:Translate MantSesion}"
    HorizontalOptions="StartAndExpand"
    VerticalOptions="Center"
/>
```

Ilustración 67 - Traducción vistas

7. Verificación y evaluación

En este apartado se detallarán cuáles han sido las pruebas realizada a cada una de las diferentes partes que componen el proyecto de eTorax y los resultados que se han obtenido de las mismas.

7.1. Pruebas del back-end

Debido a que la API REST del proyecto dispone de una gran cantidad de servicios se ha decidido automatizar lo máximo posible las pruebas del back-end. Por ese motivo se ha decidido utilizar la herramienta “Postmant” para realizar las pruebas.

Postman se trata de una herramienta desarrollada por Google la cual permite realizar peticiones HTTP al back-end de manera sencilla. De esta manera se puede realizar peticiones HTTP sin tener que desarrollar ningún código y de forma independiente.

La forma de utilizar Postman es muy sencilla. Para crear una nueva petición al back-end se debe introducir la URL del back-end a la que se desea realizar la petición. Además de la URL. Postmant también permite seleccionar el método HTTP por el cual se desea realizar la petición. Una vez elegido el método HTTP y la URL a la que se desea realizar la petición Postman permite introducir las cabeceras de la petición y los parámetros del cuerpo.

Es importante recordar que para realizar todas las peticiones al back-end (menos para las peticiones de inicio de sesión) es necesario que en la cabecera exista un token valido. Por ese motivo primero será necesario hacer una petición de inicio de sesión, la cual devuelve el token para poder almacenarlo y usarlo en las siguientes peticiones.

Para realizar las pruebas del back, debido a la gran cantidad de pruebas, se ha decidido dividir las pruebas en bloques de pruebas unitarias las cuales sean fáciles de comprobar si el funcionamiento ha sido correcto o no. Tras realizar una petición mediante Postman este mostrará la respuesta del servidor en formato JSON.

Las pruebas del back-end se han dividido en bloques en función de la temática de la petición al servidor. Estos son los diferentes bloques en los que se dividen la pruebas: autenticación, caminar, ejercicios, grupos de riesgo, pacientes, preguntas y tratamientos.

7.1.1. Pruebas de autenticación

En este bloque de pruebas se comprobará el funcionamiento de los métodos de inicio de sesión. Se comprobará que las respuestas del servidor tanto cuando los parámetros sean correctos, como cuando no lo sean, así como si falta algún parámetro.

Tabla 54 - Pruebas de autenticación

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
1	Iniciar sesión en la aplicación web con un usuario correcto y una contraseña correcta.	Se recibe el mensaje de Ok y en el código del mensaje se recibe el token del usuario.	Se ha recibido el mensaje de Ok y en el cuerpo del mensaje se ha recibido el token del usuario.	**Si**	
2	Iniciar sesión en la aplicación web con un usuario que no existe en la base de datos.	Se recibe el mensaje de error informando que los parámetros introducidos no son correctos.	Se ha recibido el mensaje de error informando que los parámetros introducidos no son correctos.	**Si**	
3	Iniciar sesión en la aplicación web con un usuario que existe, pero con una contraseña incorrecta.	Se recibe el mensaje de error informando que la contraseña introducida es incorrecta.	Se ha recibido mensaje de error informando que la contraseña introducida es incorrecta.	**Si**	
4	Iniciar sesión en la aplicación web sin alguno de los parámetros o sin ambos parámetros.	Se recibe el mensaje de error informando que falta algún	Se ha recibido el mensaje de error informando que falta	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
		parámetro en la petición.	algún parámetro en la petición.		
5	Iniciar sesión en la aplicación móvil con un usuario correcto y una contraseña correcta.	Se recibe el mensaje de Ok y en el código del mensaje se recibe el token del usuario.	Se ha recibido el mensaje de Ok y en el cuerpo del mensaje se ha recibido el token del usuario.	**Si**	
6	Iniciar sesión en la aplicación móvil con unas credenciales incorrectas.	Se recibe el mensaje de error informando que los parámetros introducidos no son correctos.	Se ha recibido el mensaje de error informando que los parámetros introducidos no son correctos.	**Si**	
7	Iniciar sesión en la aplicación móvil sin alguno de los parámetros o sin ambos parámetros.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
8	Iniciar sesión en la aplicación móvil por primera vez.	Se recibe el mensaje de Ok con el nuevo token recién creado.	Se ha recibido el mensaje de Ok con el nuevo token recién creado.	**Si**	

7.1.2. Pruebas de caminar

En este apartado se mostrarán cuáles han sido las pruebas realizadas para las funcionalidades relacionadas con caminar.

Tabla 55 - Pruebas de caminar

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
1	Obtener los parámetros de caminar de un grupo de riesgo especifico.	Se obtiene la respuesta de Ok con los parámetros de distancia y tiempo.	Se ha recibido un error en el que se indica la falta de autorización.	**No**	Se ha producido un error a la hora de introducir el token en Postman
1.1			Se ha obtenido la respuesta de OK con los parámetros de distancia y tiempo.	**Si**	
2	Obtener los parámetros de caminar de un grupo de riesgo que no existe en el sistema.	Se obtiene una respuesta de error avisando que el grupo de riesgo no existe	Se ha obtenido el mensaje de error avisando de que no se pudieron obtener los parámetros.	**Si**	El mensaje de error contiene una errata
3	Obtener los parámetros de caminar de un grupo de riesgo sin enviar el grupo de riesgo	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
4	Actualizar los parámetros del objetivo a caminar al tratamiento de un paciente con todos los parámetros correctos	Se obtiene el mensaje de que se han actualizado los objetivos de caminar del paciente y en la base de datos los objetivos del paciente han sido actualizados.	Se ha obtenido el mensaje de que se han actualizado los objetivos de caminar del paciente y en la base de datos los objetivos del paciente han sido actualizados.	**Si**	
5	Actualizar los parámetros del objetivo a caminar al tratamiento de un paciente que no existe en el sistema	Se obtiene el mensaje de que se ha producido un error.	Se ha obtenido el mensaje de que se ha producido un error.	**Si**	
6	Actualizar los parámetros del objetivo a caminar al tratamiento de un paciente sin alguno de los parámetros.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
7	Comprobar distancia caminada del día actual por un paciente.	Se comprobará cual es la distancia recorrida por el paciente el día actual y se comparará con el objetivo del paciente	Se comprueba cual es la distancia recorrida por el paciente el día actual y se compara con el objetivo del paciente	**Si**	En las siguientes pruebas se realizan las pruebas según la distancia recorrida el día actual
7.1	Comprobar distancia caminada del día actual de un paciente cuando el paciente ha cumplido el objetivo.	Se compara la distancia recorrida con la distancia objetivo y se recibe un mensaje avisando de que se ha cumplido el objetivo.	Se ha comprobado la distancia recorrida con la distancia objetivo y se ha recibido un mensaje avisando de que se ha cumplido el objetivo.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
7.2	Comprobar distancia caminada del día actual de un paciente cuando el paciente no ha cumplido el objetivo.	Se compara la distancia recorrida con la distancia objetivo y se recibe un mensaje avisando de que no se ha cumplido el objetivo.	Se ha comprobado la distancia recorrida con la distancia objetivo y se ha recibido un mensaje avisando de que no se ha cumplido el objetivo.	**Si**	
8	Comprobar distancia caminada del día actual de un paciente sin alguno de los parámetros.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
9	Obtener la distancia caminada por el paciente hoy.	Se recibe el mensaje de Ok con los parámetros de distancia y tiempo caminados	Se ha recibido el mensaje de Ok con los parámetros de distancia y tiempo caminados	**Si**	
10	Obtener la distancia caminada por el paciente hoy sin alguno de los parámetros	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
11	Obtener el objetivo de caminar de un paciente	Se recibe el mensaje de Ok con los parámetros de distancia y tiempo caminados	Se ha recibido el mensaje de Ok con los parámetros de distancia y tiempo caminados	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
12	Obtener el objetivo de caminar de un paciente que no está almacenado en el sistema.	Se recibe el mensaje de error informando de que el paciente introducido no existe.	Se recibe el mensaje de Ok sin los parámetros del paciente	**NO**	
12.1			Se ha recibido el mensaje de error informando de que el paciente introducido no existe.	**Si**	
13	Obtener el objetivo de caminar de un paciente sin alguno de los parámetros	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
14	Actualizar la distancia caminada por un paciente	Se almacena los datos en la base de datos y se recibe el mensaje de Ok	Se han almacenado los datos en la base de datos y se ha recibido el mensaje de ok	**Si**	
15	Actualizar la distancia caminada por un paciente que no existe	No se almacenan los datos en la base de datos y se recibe el mensaje de error.	No se han almacenado los datos en la base de datos y se ha recibido el mensaje de ok	**No**	No se ha recibido el mensaje de error.
15.1			No se han almacenado los datos en la base de datos y se ha recibido el mensaje de error	**Si**	

7.1.3. Pruebas de los ejercicios

En este apartado se mostrarán cuáles han sido las pruebas realizadas para las funcionalidades relacionadas con los ejercicios.

Tabla 56 - Pruebas de ejercicios

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
1	Obtener todos los ejercicios almacenados en la base de datos	Se recibe el mensaje de Ok con todos los ejercicios del sistema en el cuerpo del mensaje.	Se ha recibido el mensaje de Ok con todos los ejercicios del sistema en el cuerpo del mensaje.	**Si**	
2	Obtener los ejercicios de un grupo de riesgo especifico.	Se obtiene la respuesta de Ok con los ejercicios del grupo de riesgo	Se ha obtenido la respuesta de OK con los ejercicios del grupo de riesgo.	**Si**	
3	Obtener los ejercicios de un grupo de riesgo sin enviar el grupo de riesgo	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
4	Obtener los ejercicios asignados al tratamiento de un paciente con ejercicios asignados.	Se obtiene la respuesta de Ok con los ejercicios del tratamiento del paciente.	Se ha obtenido la respuesta de OK con los ejercicios del tratamiento del paciente.	**Si**	
5	Obtener los ejercicios asignados al tratamiento de un paciente sin ejercicios asignados.	Se obtiene la respuesta de Ok sin ningún ejercicio	Se ha obtenido la respuesta de OK sin ningún ejercicio.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
6	Obtener los ejercicios asignados al tratamiento de un paciente sin introducir el parámetro.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
7	Modificar los ejercicios asignados a un paciente.	Se recibe un mensaje de OK informando que los ejercicios del paciente han sido modificados.	Se ha recibido el mensaje de OK informando que los ejercicios del paciente han sido modificados.	**Si**	
8	Modificar los ejercicios asignados a un paciente sin alguno de los parámetros	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
9	Obtener las repeticiones por defecto para los ejercicios seleccionados.	Se recibe el mensaje de OK con las repeticiones por defecto de cada ejercicio.	Se ha recibido el mensaje de OK con las repeticiones por defecto de cada ejercicio.	**Si**	
10	Actualizar las repeticiones realizadas de un ejercicio asignado a un tratamiento.	Se recibe el mensaje de OK confirmando que las repeticiones han sido actualizadas.	Se ha recibido el mensaje de OK confirmando que las repeticiones han sido actualizadas.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
11	Actualizar las repeticiones realizadas de un ejercicio asignado a un tratamiento sin uno de los parámetros.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
12	Actualizar las repeticiones realizadas de un ejercicio asignado a un tratamiento con un id de tratamiento que no existe.	Se recibe el mensaje de error informando de que algo ha fallado.	Se ha recibido el mensaje de error informando de que algo ha fallado.	**Si**	
13	Actualizar las repeticiones realizadas de un ejercicio asignado a un tratamiento con un id de ejercicio que no existe.	Se recibe el mensaje de error informando de que algo ha fallado.	Se ha recibido el mensaje de error informando de que algo ha fallado.	**Si**	

7.1.4. Pruebas de los grupos de riesgo

En este apartado se mostrarán cuáles han sido las pruebas realizadas para las funcionalidades relacionadas con los grupos de riesgo.

Tabla 57 - Pruebas de los grupos de riesgo

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
1	Obtener todos los grupos de riesgo del sistema.	Se recibe el mensaje de Ok con todos los grupos de riesgo almacenados en la base de datos.	Se recibe el mensaje de Ok con todos los grupos de riesgo almacenados en la base de datos.	**Si**	
2	Modificar el grupo de riesgo asignado a un paciente	Se recibe el mensaje de Ok informando que el grupo de riesgo del paciente ha sido modificado y se ha modificado el grupo de riesgo en la base de datos.	Se ha recibido el mensaje de Ok informando que se han obtenido los grupos de riesgo de forma correcta, Se ha modificado el grupo de riesgo en la base de datos.	**No**	El mensaje es erróneo.
2.1			Se ha recibido el mensaje de Ok informando que el grupo se riesgo se ha modificado y se ha modificado el grupo de riesgo en la base de datos.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
3	Modificar el grupo de riesgo asignado a un paciente con un grupo de riesgo que no existe.	Se recibe un mensaje de error informando de que el grupo de riesgo no existe.	Se ha recibido un mensaje de error informando de ha sucedido un error interno en el servidor y este ha tenido una excepción.	**No**	Hay que controlar el error e informar de que no existe el grupo de riesgo
3.1			Se ha recibido un mensaje de error informando de que el grupo introducido no existe.	**Si**	
4	Modificar el grupo de riesgo asignado a un paciente sin alguno de los parámetros.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
5	Crear un nuevo grupo de riesgo.	Se recibe el mensaje de Ok y el grupo se ha creado en la base de datos.	Se ha recibido el mensaje de Ok y el grupo se ha creado en la base de datos.	**Si**	
6	Crear un nuevo grupo de riesgo introduciendo un grupo de riesgo que ya existe	Se recibe un mensaje de error informando que el grupo de riesgo introducido ya existe.	Se ha recibido un mensaje de error informando que el grupo de riesgo introducido ya existe.	**si**	

7.1.5. Pruebas de los pacientes

En este apartado se mostrarán cuáles han sido las pruebas realizadas para las funcionalidades relacionadas con los pacientes.

Tabla 58 - Pruebas de los pacientes

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
1	Obtener solo los pacientes activos con tratamientos activos.	Se recibe el mensaje de OK y todos los pacientes con tratamientos activos que se encuentran almacenados.	Se ha recibido el mensaje de OK y todos los pacientes con tratamientos activos que se encuentran almacenados.	**Si**	
2	Obtener solo los pacientes con tratamientos inactivos	Se recibe el mensaje de OK y todos los pacientes con tratamientos inactivos que se encuentran almacenados.	Se ha recibido el mensaje de OK y todos los pacientes con tratamientos inactivos que se encuentran almacenados.	**Si**	
3	Obtener todos los pacientes.	Se recibe el mensaje de OK y todos los pacientes.	Se ha recibido el mensaje de OK y todos los pacientes.	**Si**	
4	Silenciar un paciente.	Se recibe el mensaje de OK informando de que el paciente ha sido silenciado.	Se ha recibido el mensaje de OK informando de que el paciente ha sido silenciado.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
5	Silenciar un paciente que no existe en el sistema.	Se recibe el mensaje de error informando de que el paciente no existe en el sistema	Se ha recibido el mensaje de error informando de que el paciente no existe en el sistema.	**Si**	
6	Silenciar un paciente que ya se encuentra silenciado.	Se recibe el mensaje de OK informando de que el paciente se encuentra silenciado.	Se ha recibido el mensaje de OK informando de que el paciente se encuentra silenciado.	**Si**	
7	Silenciar un paciente sin introducir el paciente.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
8	Desilenciar un paciente.	Se recibe el mensaje de OK informando de que el paciente ha sido desilenciado.	Se ha recibido el mensaje de OK informando de que el paciente ha sido desilenciado.	**Si**	
9	Desilenciar un paciente que no existe en el sistema.	Se recibe el mensaje de error informando de que el paciente no existe en el sistema	Se ha recibido el mensaje de error informando de que el paciente no existe en el sistema.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
10	Desilenciar un paciente que ya se encuentra silenciado.	Se recibe el mensaje de OK informando de que el paciente se encuentra desilenciado.	Se ha recibido el mensaje de OK informando de que el paciente se encuentra desilenciado.	**Si**	
11	Desilenciar un paciente sin introducir el paciente.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
12	Crear un nuevo paciente.	Se recibe el mensaje de Ok informando que el paciente ha sido introducido en el sistema.	Se ha recibido el mensaje de Ok informando que el paciente ha sido introducido en el sistema.	**Si**	
13	Crear un nuevo paciente sin alguno de los parámetros.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
14	Obtener los datos de un paciente,	Se recibe el mensaje de Ok con los datos del paciente.	Se ha recibido el mensaje de Ok con los datos del paciente.	**Si**	
15	Modificar un paciente.	Se recibe el mensaje de Ok y el paciente ha sido actualizado en la base de datos.	Se ha recibido el mensaje de Ok y el paciente ha sido actualizado en la base de datos.		

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
16	Modificar un paciente que no existe.	Se recibe el mensaje de error informando de que el paciente no existe en el sistema	Se ha recibido el mensaje de error informando de que el paciente no existe en el sistema.	**Si**	
17	Modificar un paciente sin alguno de los parámetros.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
18	Cambiar la contraseña del paciente	Se recibe el mensaje de OK informando que la contraseña ha sido actualizada.	Se ha recibido el mensaje de OK informando que la contraseña ha sido actualizada.	**Si**	
19	Cambiar la contraseña de un paciente que no existe.	Se recibe el mensaje de error informando de que el paciente no existe en el sistema	Se ha recibido el mensaje de error informando de que el paciente no existe en el sistema.	**Si**	
20	Eliminar un paciente del sistema.	Se recibe el mensaje de OK informando de que el paciente ha sido eliminado del sistema	Se ha recibido el mensaje de OK informando de que el paciente ha sido eliminado del sistema	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
21	Eliminar un paciente que no existe en el sistema.	Se recibe el mensaje de error informando de que el paciente no existe en el sistema	Se ha recibido el mensaje de error informando de que el paciente no existe en el sistema.	**Si**	
22	Eliminar un paciente del sistema sin alguno de los parámetros.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
23	Actualizar contraseña	Se recibe el mensaje de OK informando de que la contraseña ha sido actualizada	Se ha recibido el mensaje de OK informando de que la contraseña ha sido actualizada	**Si**	A diferencia de la anterior esta función es llamada por el paciente.
24	Actualizar contraseña introduciendo una contraseña antigua incorrecta.	Se recibe el mensaje de error informando de que la contraseña antigua no es correcta.	Se ha recibido el mensaje de error informando de que la contraseña antigua no es correcta.		
25	Actualizar contraseña sin alguno de los parámetros	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
26	Exportar pacientes.	Se recibe el mensaje de OK informando de que se han obtenido los pacientes y se pasa en el cuerpo del mensaje la información de los pacientes.	Se ha recibido el mensaje de OK informando de que se han obtenido los pacientes y se ha pasado en el cuerpo del mensaje la información de los pacientes.	**Si**	
27	Exportar información de un paciente.	Se recibe el mensaje de OK informando de que se han obtenido los datos del paciente y en el cuerpo del mensaje se envían los datos del paciente	Se ha recibido el mensaje de OK informando de que se han obtenido los datos del paciente y en el cuerpo del mensaje se envían los datos del paciente	**Si**	
28	Exportar información de un paciente que no existe,	Se recibe el mensaje de error informando de que el paciente no existe en el sistema	Se ha recibido el mensaje de error informando de que el paciente no existe en el sistema.	**Si**	

7.1.6. Pruebas de las preguntas

En este apartado se mostrarán cuáles han sido las pruebas realizadas para las funcionalidades relacionadas con las preguntas.

Tabla 59 - Pruebas de las preguntas

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
1	Obtener todas las preguntas,	Se recibe la respuesta de Ok y en el cuerpo de las respuestas estarán todas las preguntas del sistema.	Se ha recibido la respuesta de Ok y en el cuerpo de la respuesta esta, todas las preguntas del sistema.	**Si**	
2	Obtener las preguntas de un grupo de riesgo.	Se recibe la respuesta de Ok y en el cuerpo de la respuesta estarán todas las preguntas del grupo de riesgo	Se ha recibido la respuesta de Ok y en el cuerpo de la respuesta están todas las preguntas del grupo de riesgo	**Si**	
3	Obtener las preguntas de un grupo de riesgo sin un parámetro.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
4	Obtener las preguntas del tratamiento de un paciente.	Se recibe la respuesta de Ok y en el cuerpo de la respuesta estarán todas las preguntas del tratamiento del paciente.	Se ha recibido la respuesta de Ok y en el cuerpo de la respuesta están todas las preguntas del tratamiento del paciente.	**Si**	
5	Obtener las preguntas del tratamiento de un paciente sin un parámetro.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
6	Modificar las preguntas asignadas a un paciente.	Se recibe un mensaje de OK informando que las preguntas del paciente han sido modificadas.	Se ha recibido un mensaje de OK informando que las preguntas del paciente han sido modificadas.	**Si**	
7	Modificar las preguntas asignadas a un paciente sin alguno de los parámetros.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
8	Obtener las respuestas a las preguntas iniciales del paciente.	Se recibe un mensaje de OK informando que las preguntas del paciente se han obtenido y se pasan en el cuerpo de la respuesta.	Se ha recibido un mensaje de OK informando que las preguntas del paciente se han obtenido y se han pasado en el cuerpo de la respuesta.	**Si**	
9	Obtener las respuestas a las preguntas iniciales del paciente sin alguno de los parámetros.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
10	Crear una nueva pregunta	Se recibe el mensaje de Ok y se almacena la nueva pregunta en el sistema.	Se ha recibido el mensaje de Ok y se ha almacenado la nueva pregunta en el sistema.	**Si**	
11	Intentar crear una pregunta que ya existe	Se recibe el mensaje de error y se informa que la pregunta ya existe	Se recibe el mensaje de Ok y se ha almacenado la nueva pregunta en el sistema.	**No**	
11.1			Se recibe el mensaje de error y con la información de que la pregunta ya se encuentra en el sistema.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
12	Intentar crear una pregunta sin uno de los parámetros.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
13	Obtener preguntas no iniciales de un paciente	Se recibe el mensaje de Ok y en el cuerpo del mensaje se pasan las preguntas asignadas al paciente.	Se ha recibido el mensaje de Ok y en el cuerpo del mensaje se han pasado las preguntas asignadas al paciente.	**Si**	
14	Obtener preguntas no iniciales de un paciente sin algún parámetro.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
15	Obtener preguntas iniciales de un paciente	Se recibe el mensaje de Ok y en el cuerpo del mensaje se pasan las preguntas asignadas al paciente.	Se ha recibido el mensaje de Ok y en el cuerpo del mensaje se han pasado las preguntas asignadas al paciente.	**Si**	
16	Obtener preguntas iniciales de un paciente sin algún parámetro.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
17	Responder una pregunta.	Se recibe el mensaje de OK y se almacena la respuesta en el sistema.	Se ha recibido el mensaje de OK y se almacena la respuesta en el sistema.	**Si**	
18	Responder una pregunta que ya se ha respondida.	Se recibe el mensaje de OK y se actualiza la respuesta en el sistema.	Se ha recibido el mensaje de OK y se actualiza la respuesta en el sistema.	**Si**	
	Responder una pregunta sin alguno de los parámetros	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	

7.1.7. Pruebas de los tratamientos

En este apartado se mostrarán cuáles han sido las pruebas realizadas para las funcionalidades relacionadas con postratamientos asignados a los pacientes.

Tabla 60 - Pruebas de los tratamientos

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
1	Actualizar las fechas de un tratamiento.	Se recibe el mensaje de OK y las fechas del tratamiento se han actualizado en la base de datos.	Se ha recibido el mensaje de OK y las fechas del tratamiento se han actualizado en la base de datos.	**Si**	
2	Actualizar las fechas de un tratamiento que no existe.	Se recibe el mensaje de error y se informa al usuario de que el tratamiento no existe.	Se ha recibido un mensaje de Ok informando que las fechas se han actualizado.	**No**	
2.1			Se ha recibido el mensaje de error y se informa al usuario de que el tratamiento no existe.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
3	Actualizar las fechas de un tratamiento sin alguno de los parámetros.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
4	Obtener la tabla corta del tratamiento.	Se recibe el mensaje de Ok y en el cuerpo del mensaje se encuentra la tabla del paciente.	Se ha recibido el mensaje de Ok y en cuerpo del mensaje estala tabla.	**Si**	
5	Obtener la tabla corta de un tratamiento que no existe o no tiene datos.	Se recibe el mensaje de OK y se recibe la tabla corta sin datos.	Se ha recibido el mensaje de OK y se ha recibido la tabla corta sin datos.	**Si**	
6	Obtener la tabla corta del paciente sin alguno de los parámetros.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
7	Obtener la tabla larga.	Se recibe el mensaje de Ok y en el cuerpo del mensaje se encuentra la tabla del paciente.	Se ha recibido el mensaje de Ok y en cuerpo del mensaje estala tabla.	**Si**	
8	Obtener la tabla larga de un tratamiento que no existe	Se recibe el mensaje de error y se informa al usuario de que no se han obtenido los datos	Se recibe el mensaje de Ok con una tabla vacía.	**No**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
8.1			Error interno en el servidor, propiedad no encontrada.	**No**	Código de respuesta mal introducido
8.2			Se ha recibido el mensaje de error y se ha informado al usuario de que no se han obtenido los datos	**Si**	
9	Obtener la tabla corta del paciente sin alguno de los parámetros.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
10	Obtener los objetivos del tratamiento del paciente.	Se recibe el mensaje de OK con los objetivos del paciente en el cuerpo del mensaje.	Se ha recibido el mensaje de OK con los objetivos del paciente en el cuerpo del mensaje.	**Si**	
11	Sin alguno de los parámetros.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
12	Obtener el nombre de un paciente en función del id de tratamiento.	Se recibe el mensaje de OK con el nombre del paciente en el cuerpo del mensaje.	Se recibe el mensaje de OK con el nombre del paciente en el cuerpo del mensaje.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
13	Obtener el nombre de un paciente en función del id de tratamiento que no existe	Se recibe el mensaje de error indicando que el tratamiento introducido no existe.	Se ha recibido el mensaje de error indicando que no se han obtenido datos para el tratamiento.	**Si**	
14	Obtener el nombre de un paciente en función del id de tratamiento sin alguno de los parámetros.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	
15	Obtener la fecha de la operación de un tratamiento.	Se recibe el mensaje de OK con la fecha de la operación en el cuerpo del mensaje.	Se recibe el mensaje de OK con la fecha de la operación en el cuerpo del mensaje.	**Si**	
16	Obtener la fecha de la operación de un tratamiento que no existe.	Se recibe el mensaje de error indicando que el tratamiento introducido no existe.	Se ha recibido el mensaje de error indicando que no se han obtenido datos para el tratamiento.		
17	Obtener la fecha de la operación de un tratamiento sin alguno de los parámetros.	Se recibe el mensaje de error informando que falta algún parámetro en la petición.	Se ha recibido el mensaje de error informando que falta algún parámetro en la petición.	**Si**	

7.2. Pruebas de la página web

En este apartado se han realizado las pruebas referentes a la página web. Estas pruebas se han realizado tras realizar las pruebas del back-end por lo que las peticiones al back-end no fallarán. Al igual que en el apartado anterior las pruebas de dividirán en grupos con el fin de aumentar la organización y no dejar nada sin probar.

7.2.1. Pruebas de inicio de sesión y cierre de sesión.

En este apartado se mostrarán cuáles han sido las pruebas realizadas para la funcionalidad relacionadas el inicio de sesión y el cierre de sesión de la página web.

Tabla 61 - Pruebas de inicio de sesión y cierre de sesión

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
1	El usuario intenta iniciar sesión con unas credenciales correctas y sin marcar la opción de mantener la sesión iniciada	El back-en devuelve el token del usuario y el front-end almacena en el “Sesión storage” del navegador.	El back-en ha devuelto el token del usuario y el front-end almacena en el “Sesión storage” del navegador.	**Si**	
2	El usuario intenta iniciar sesión con unas credenciales correctas y marcando la opción de mantener la sesión iniciada	El back-en devuelve el token del usuario y el front-end lo almacena en el “Local storage” del navegador.”	El back-en ha devuelto el token del usuario y el front-end almacena en el “Local storage“ del navegador.	**Si**	
3	El usuario intenta iniciar sesión con un usuario incorrecto y una contraseña incorrecta y sin marcar la opción de mantener la sesión iniciada	Se muestra un mensaje de error avisando de que las credenciales introducidas no son correctas.	Se ha mostrado un mensaje de error avisando de que las credenciales introducidas no son correctas.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
4	El usuario intenta iniciar sesión con un usuario incorrecto y una contraseña incorrecta y marcando la opción de mantener la sesión iniciada	Se muestra un mensaje de error avisando de que las credenciales introducidas no son correctas.	Se ha mostrado un mensaje de error avisando de que las credenciales introducidas no son correctas.	**Si**	
5	El usuario intenta iniciar sesión con un usuario correcto pero una contraseña incorrecta y sin marcar la opción de mantener la sesión iniciada	Se muestra un mensaje de error avisando de que las credenciales introducidas no son correctas.	Se niega el acceso a la aplicación, pero no se muestra ningún error.	**No**	El error solo se mostraba cuando el usuario era incorrecto.
5.1			Se ha mostrado un mensaje de error avisando de que las credenciales introducidas no son correctas.	**Si**	
6	El usuario intenta iniciar sesión con un usuario correcto pero una contraseña incorrecta y marcando la opción de mantener la sesión iniciada	Se muestra un mensaje de error avisando de que las credenciales introducidas no son correctas.	Se ha mostrado un mensaje de error avisando de que las credenciales introducidas no son correctas.	**Si**	
7	El usuario intenta iniciar sesión sin la contraseña o el usuario	Se muestra un mensaje de error indicado que falta algún valor por introducir.	Se ha mostrado un mensaje de error indicado que falta algún valor por introducir.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
8	El usuario hace click sobre el botón de cerrar sesión desde cualquier parte de la aplicación.	Se le devuelve a la página de inicio de sesión y se borran los valores almacenados en el navegador.	Se le ha devuelto a la página de inicio de sesión y se han borrado los valores almacenados en el navegador.	**Si**	

7.2.2. Pruebas de configuración

En este apartado se mostrarán cuáles han sido las pruebas realizadas para las funcionalidades relacionadas la configuración de la página web.

Tabla 62 - Pruebas de configuración

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
1	El usuario selecciona un idioma y hace click sobre el botón de cambiar idioma	El idioma de la aplicación se cambia.	Se ha cambiado el idioma de la aplicación.	**Si**	
2	El usuario escribe una nueva pregunta sin marcar la opción "¿Es inicial?" y elige el tipo de respuesta.	Se almacena la nueva pregunta en la base de datos y a partir de ese momento está disponible para ser asignada a los pacientes.	Se ha almacenado la nueva pregunta en la base de datos y a partir de ese momento está disponible para ser asignada a los pacientes.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
3	El usuario escribe una nueva pregunta marcando "¿Es inicial?".	La opción de tipo de respuesta desaparece y se crea una nueva pregunta inicial en la base de datos que puede ser asignada a partir de ese momento.	La opción de tipo de respuesta ha desaparecido y se ha creado una nueva pregunta inicial en la base de datos que puede ser asignada a partir de ese momento.	**Si**	La opción de tipo de respuesta desaparece ya que las preguntas iniciales siempre son preguntas que se responde con un sí o un no.
4	El usuario intenta crear una pregunta sin escribir la pregunta.	Se le avisa al paciente de que debe escribir la pregunta.	Se ha avisado al paciente de que debe escribir la pregunta.	**Si**	
5	El usuario intenta crear un grupo de riesgo nuevo con todos los datos y eligiendo preguntas y ejercicios por defecto.	Se crea un nuevo grupo de riesgo.	Un nuevo grupo de riesgo ha sido creado.	**Si**	
6	Se intenta crear un nuevo grupo de riesgo que ya existe.	Se muestra un error informando de que el grupo que se intenta crear ya existe.	Se ha mostrado un erro informado de que el grupo que se intenta crear ya existe.	**Si**	
7	Se intenta crear un grupo de riesgo sin alguno de los parámetros.	Se muestra un error informando de que falta algún parámetro.	No se muestra ningún error.	**No**	La petición al back-end no se realiza.

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
7.1			Se ha mostrado un error informando de que falta algún parámetro.	**Si**	
8	El usuario intenta crear un grupo de riesgo nuevo con todos los datos y sin elegir preguntas ni ejercicios.	Se crea un nuevo grupo de riesgo sin preguntas ni ejercicios por defecto.	Se ha creado un nuevo grupo de riesgo sin preguntas ni ejercicios por defecto.	**Si**	
9	Se intenta exportar los pacientes seleccionados.	Se abre la ventana para guardar el archivo con la información extraída de la base de datos.	Se ha abierto la ventana para guardar el archivo con la información extraída de la base de datos.	**Si**	
10	Se intenta exportar la información de un paciente en concreto.	Se abre la ventana para guardar el archivo con la información extraída de la base de datos.	Se ha abierto la ventana para guardar el archivo con la información extraída de la base de datos.	**Si**	

7.2.3. Gestión de pacientes

En este apartado se mostrarán cuáles han sido las pruebas realizadas para las funcionalidades relacionadas gestión de pacientes de la página web.

Tabla 63 - Gestión de pacientes

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
1	El usuario pulse el botón de silenciar un paciente.	El paciente es silenciado en la base de datos y se navega a la página del paciente.	El paciente se ha silenciado en la base de datos y se ha navegado a la página del paciente.	**Si**	
2	El usuario pulsa el botón de desilenciar un paciente.	El paciente es desilenciado en la base de datos y se navega a la página del paciente.	El paciente se ha desilenciado en la base de datos y se ha navegado a la página del paciente.	**Si**	
3	Se introduce los datos en el buscador de la página principal.	Se busca los datos introducidos en el buscador en todos los campos de la tabla.	Se han buscado los datos introducidos en el buscador en todos los campos de la tabla.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
4	Se filtra los pacientes según si están silenciados o no.	En la tabla se muestran solo los pacientes que pasan el filtro.	En la tabla se ha mostrado todos los pacientes.	**No**	
4.1			En la tabla se muestran solo los pacientes que pasan el filtro.	**Si**	
5	El usuario hace click sobre el botón de crear paciente.	Se navega hasta el formulario de crear paciente.	Se ha navegado al formulario de crear paciente.cd		
6	El usuario elige el grupo de riesgo durante la creación del paciente.	Se añaden las preguntas, los ejercicios y los objetivos a caminar por defecto del grupo de riesgo.	No se añade los objetivos a caminar por defecto.	**No**	El URL de la petición estaba mal
6.1			Se ha añadido las preguntas, los ejercicios y los objetivos a caminar por defecto del grupo de riesgo.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
7	El usuario pulsa el botón continuar durante la creación del paciente.	Se navega hasta la página para elegir el número de repeticiones para cada ejercicio asignado y se muestran las repeticiones por defecto de cada ejercicio.	Se ha navegado hasta la página para elegir el número de repeticiones para cada ejercicio asignado y se han mostrado las repeticiones por defecto de cada ejercicio.	**Si**	
8	Se pulsa el botón de modificar paciente.	Se navega hasta el formulario de modificación de paciente en el que por defecto aparecen los valores asignados al paciente.	Se ha navegado hasta el formulario de modificación de paciente en el que por defecto aparecen los valores asignados al paciente.	**Si**	Para la modificación del paciente se ha realizado las mismas pruebas que para la creación del paciente.
9	El usuario pulsa sobre el botón de redefinir contraseña.	La contraseña se actualiza en la base de datos y se muestra cual es la nueva contraseña.	La contraseña se ha actualizado en la base de datos y se muestra cual es la nueva contraseña.	**Si**	
10	Se pulsa el botón de eliminar paciente.	Se pide la confirmación de que se desea eliminar el paciente.	Se ha pedido la confirmación de que se desea eliminar el paciente.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
11	Filtrar las fechas entre las que se muestran los gráficos.	Solo se mostrarán los datos entre las fechas elegidas.	Solo se han mostrado los datos entre las fechas elegidas	**Si**	

7.2.4. Datos del paciente

En este apartado se mostrarán cuáles han sido las pruebas realizadas para las funcionalidades relacionadas la visualización de los datos del paciente.

Tabla 64 - Datos del paciente

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
1	El usuario pulsa sobre un paciente en la página principal	Se navega a la página en la que demuestra la información general del paciente y una pequeña tabla de la actividad que ha realizado los ultimo 7 días.	Se Ha navegado a la página en la que demuestra la información general del paciente y una pequeña tabla de la actividad que ha realizado los ultimo 7 días, todos los datos se muestran correctamente.	**Si**	
2	En la página del paciente se pulsa sobre ver preguntas iniciales.	Se muestra las respuestas que el paciente ha dado a las preguntas iniciales.	Se han mostrado las respuestas que el paciente ha dado a las preguntas iniciales.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
3	Se pulsa sobre “Ver todos los datos del paciente”.	Se navega hasta la página donde se muestra una tabla con la actividad que el usuario ha realizado día a día	Se navega hasta la página donde se muestra una tabla con la actividad que el usuario ha realizado día a día y se muestran los datos de forma correcta.	**Si**	
4	Se pulsa sobre uno de los días de la tabla	Se navega a la página en la que se muestra todos los datos recogidos por el paciente un día determinado.	Se ha navegado a la página en la que se muestra todos los datos recogidos por el paciente un día determinado y todos los datos se han mostrado de forma correcta.	**Si**	
5	Se pulsa sobre la opción de “Ver gráficos”	Se navega a la página en la que se muestran los gráficos.	Se ha navegado a la página en la que se muestran los gráficos. Los gráficos se han mostrado de forma correcta.	**Si**	

7.3. Pruebas de la aplicación móvil

En este apartado se detallarán cuáles han sido las pruebas realizadas a la aplicación móvil. Al igual que los apartados anteriores en este también se dividirán las pruebas en diferentes grupos para de esta forma mejorar la organización y evitar que se queden pruebas sin realizar. Los grupos en los que se han dividido las pruebas son: inicio de sesión y cierre de sesión, ejercicios, preguntas, caminar, preguntas frecuentes y recomendaciones.

Cabe mencionar que las pruebas se han realizado en un móvil Android y un emulados de Windows pone. No se han podido realizar las pruebas en un iPhone ya que para compilar el código se necesita un ordenados de la misma marca.

7.3.1. Pruebas de inicio y cierre de sesión

En este apartado se mostrarán cuáles han sido las pruebas realizadas para las funcionalidades relacionadas con el inicio de sesión y el cierre de sesión de la aplicación móvil.

Tabla 65 - Pruebas de inicio y cierre de sesión

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
1	Iniciar sesión con unas credenciales correctas.	Se recibe el token del usuario y se navega a la página principal de la aplicación, el token se almacena para las futuras peticiones.	Se ha recibido el token del usuario y se ha navegado a la página principal de la aplicación, el token se ha almacenado para las futuras peticiones.	**Si**	
2	Iniciar sesión con unas credenciales incorrectas.	Salta una alerta avisando de que las credenciales introducidas no son correctas y se eliminan los valores introducidos por el usuario.	Ha saltado una alerta avisando de que las credenciales no son correctas. Pero no se han eliminado los valores introducidos.	**No**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
2.1			Ha saltado una alerta avisando de que las credenciales no son correctas y se han eliminad los valores introducidos por el usuario.	**Si**	
3	Iniciar sesión sin alguno de los parámetros.	Salta una alerta avisando de que las credenciales introducidas no son correctas.	Ha saltado una alerta avisando de que las credenciales no son correctas.	**Si**	
4	Intentar iniciar sesión sin conexión a internet.	Se avisa de que no existe conexión a internet y que se active la conexión si no está activada.	Se ha avisado de que no existe conexión a internet y que se active la conexión si no está activada.	**Si**	Esta prueba se ha realizado para todas las peticiones que se hacen al back-end
5	Se inicia sesión por primera vez	Se crea un token nuevo para el usuario y se le realizan las preguntas iniciales.	Se ha creado un token nuevo para el usuario y se han realizado las preguntas iniciales.	**Si**	
6	Se abre la aplicación cuando la última vez se inició sesión con la opción de mantener sesión iniciada.	No se piden las credenciales al usuario y se abre la página principal directamente.	No se han pedido las credenciales al usuario y se ha abierto la página principal directamente.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
7	Se pulsa sobre el botón de cerrar sesión.	Se elimina el token almacenado y se redirige al usuario a la página de Login	Se ha eliminado el token almacenado y se ha redirigido al usuario a la página de Login	**Si**	

7.3.2. Pruebas de los ejercicios

En este apartado se mostrarán cuáles han sido las pruebas realizadas para las funcionalidades relacionadas con los ejercicios de la aplicación móvil.

Tabla 66 - Pruebas de los ejercicios

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
1	En la página principal se pulsa sobre la opción de ejercicios.	Se abre la lista de los ejercicios que tiene asignados el paciente al que pertenece la sesión.	Se ha abierto la lista de los ejercicios que tiene asignados el paciente al que pertenece la sesión.	**Si**	
2	Se pulsa sobre uno de los ejercicios de la lista de ejercicios del paciente.	Se muestran los datos del ejercicio seleccionado.	Se han mostrado los datos del ejercicio seleccionado.	**Si**	
3	Se pulsa sobre la opción de ver video.	El usuario es redirigido a la aplicación de YouTube y se reproduce el video del ejercicio seleccionado.	Se ha redirigido a la aplicación de YouTube y se reproduce el video del ejercicio seleccionado.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
4	El usuario introduce el número de repeticiones del ejercicio que ha realizado y pulsa enviar.	Se envían las repeticiones al back-en y es redirigido a la lista de ejercicios.	Se han enviado las repeticiones al back-en y de ha redirigido a la lista de ejercicios.	**Si**	
5	El usuario introduce el número 0 y pulsa enviar.	Salta una alera avisando de que en número que debe introducir debe ser mayo que uno.	Ha saltado una alera avisando de que en número que debe introducir debe ser mayo que uno.	**Si**	

7.3.3. Pruebas de las preguntas

En este apartado se mostrarán cuáles han sido las pruebas realizadas para las funcionalidades relacionadas con las preguntas asignadas a un paciente de la aplicación móvil.

Tabla 67 - Pruebas de las preguntas

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
1	En la página principal se pulsa sobre la opción de preguntas	Se abre la lista de las preguntas que tiene asignadas el paciente al que pertenece la sesión.	Se ha abierto la lista de las preguntas que tiene asignadas el paciente al que pertenece la sesión.	**Si**	
2	Se pulsa sobre una de las preguntas de la lista de preguntas del paciente.	Se muestra la pregunta seleccionada. Algunas de las preguntas que se encuentran en el sistema van acompañadas de una imagen.	Se muestra la pregunta seleccionada. Si la pregunta lleva una imagen asignada se muestra esta imagen.	**Si**	
3	Se abre una pregunta cuya respuesta es de tipo si o no.	Se muestra la pregunta con dos botones uno con la opción “si” y otro con la opción “no”.	Se ha mostrado la pregunta con un área para introducir texto.	**No**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
3.1			Se ha mostrado la pregunta con dos botones uno con la opción “si” y otro con la opción “no”.	**Si**	
4	Se pulsa uno de los botones en una pregunta de tipo si o no.	Se envía la respuesta al servidos y se devuelve al usuario a la lista de preguntas	Se ha enviado la respuesta al servidor y se ha devuelto al usuario a la lista de preguntas	**Si**	
5	Se abre una pregunta cuya respuesta es de tipo numérica.	Solo se permite introducir valores numéricos.	Se pueden introducir valores de todo tipo.	**No**	
5.1			Solo se permite introducir valores numéricos.	**Si**	
6	Se pulsa enviar en una pregunta que no es te tipo sí o no con una respuesta	Se envía la respuesta al servidor y se redirige al usuario a la lista de preguntas.	Se ha enviado la respuesta al servidor y se ha redirigido al usuario a la lista de preguntas.	**Si**	
7	Se pulsa enviar en una pregunta que no es te tipo sí o no sin introducir una respuesta.	Salta una alerte avisando de que se debe introducir una respuesta.	Ha saltado una alerte avisando de que se debe introducir una respuesta.	**Si**	

7.3.4. Pruebas de caminar

En este apartado se mostrarán cuáles han sido las pruebas realizadas para las funcionalidades relacionadas con caminar de la aplicación móvil.

Tabla 68 - Pruebas de caminar

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
1	En la página principal se pulsa sobre la opción de caminar	Se abre la interfaz de caminar con los objetivos del paciente y los datos de la distancia caminada el día actual.	Se ha abierto la interfaz de caminar con los objetivos del paciente y los datos de la distancia caminada el día actual.	**Si**	
2	Se pulsa el botón iniciar.	Se inicia el cronometro y se empieza a recoger los datos de la localización del paciente y la distancia que ha realizado.	Se ha iniciado el cronometro y se empieza a recoger los datos de la localización del paciente y la distancia que ha realizado.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
3	Se pulsa el botón parar	El cronometro del paciente deja de contar el tiempo y la distancia que realiza el paciente.	El cronometro del paciente ha dejado de contar el tiempo y la distancia que realiza el paciente.	**Si**	
4	El paciente minimiza la aplicación mientras está realizando la caminata.	Le ejecución sigue en segundo plano y la aplicación seguirá controlando la distancia realizado por el paciente.	Le ejecución ha seguido en segundo plano y la aplicación sigue controlando la distancia realizada por el paciente.	**Si**	
5	El usuario bloquea el móvil durante la caminata.	Le ejecución sigue en segundo plano y la aplicación seguirá controlando la distancia realizado por el paciente.	Le ejecución ha seguido en segundo plano y la aplicación sigue controlando la distancia realizada por el paciente.	**Si**	

7.3.5. Pruebas de las preguntas frecuentes

En este apartado se mostrarán cuáles han sido las pruebas realizadas para las funcionalidades relacionadas con las preguntas frecuentes de la aplicación móvil.

Tabla 69 - Pruebas de las preguntas frecuentes

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
1	En la página principal se pulsa sobre la opción de preguntas frecuentes.	Se abre la lista de las preguntas frecuentes con todas las preguntas comprimidas.	Se ha abierto la lista de las preguntas frecuentes con todas las preguntas comprimidas.	**Si**	
2	Hacer click sobre una pregunta comprimida.	La pregunta se descomprime y el icono de la flecha desplegable apunta hacia arriba indicando que se puede volver a comprimir.	La pregunta se despliega, pero no se muestra ningún icono.	**No**	
2.1			La pregunta se descomprime y el icono de la flecha que apunta hacia arriba.	**Si**	

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
3	Hacer click sobre una pregunta desplegada.	Se vuelve a comprimir la pregunta seleccionada.	Se ha vuelto a comprimir la pregunta seleccionada.		

7.3.6. Pruebas de las recomendaciones

En este apartado se mostrarán cuáles han sido las pruebas realizadas para las funcionalidades relacionadas con las recomendaciones de la aplicación móvil.

Tabla 70 - Pruebas de las recomendaciones

Código	Descripción	Resultado esperado	Resultado obtenido	¿Correcto?	Observaciones
1	En la página principal se pulsa sobre la opción de recomendaciones.	Se abre la lista de las recomendaciones	Se ha abierto la lista de las recomendaciones	**Si**	
2	Se pulsa sobre una de las recomendaciones de la lista.	Se abre la página de la recomendación y se muestra el gif animado.	Se ha abierto la página de la recomendación y se muestra el gif animado.	**Si**	Según la documentación de la librería utilizada para los gifs en el emulador de Windows Phone no se muestran los gifs, pero en dispositivos reales no hay problema.

8. Conclusiones y trabajo futuro

Tras finalizar un proyecto de estas características es importante recapitular hasta el comienzo del proyecto y comprobar cuales han sido los pasos que se han dado y los objetivos que se han conseguido. De esta manera se podrá comprobar como de buena ha sido la planificación del proyecto y cuales han sido los objetivos que se han conseguido. De esta manera se logrará un mayor aprendizaje para futuros proyectos.

8.1. Evaluación de los objetivos

Al comienzo del proyecto, en el apartado de "Objetivos", se listaban cuáles eran los objetivos que se deseaban cumplir en el proyecto. En este punto se volverán a listar los objetivos y se analizará para cada uno de ellos si se ha cumplido de forma satisfactoria o no. Para ello se mostrarán las razones por las que el objetivo se considera cumplido o no.

- Creación de una página web que permita al equipo médico de un paciente de cirugía de tórax definir las pautas que el paciente ha de seguir tanto en el preoperatorio como en el postoperatorio.

Este objetivo se considera cumplido. Se ha creado una página web la cual permite al equipo médico Indicar pautas a un paciente determinado tanto durante el preoperatorio como durante el postoperatorio. Las pautas indicadas por el equipo médico son específicas para cada uno de los pacientes. Las pautas que el médico puede dar a cada uno de los pacientes consisten en ejercicios que el paciente debe realizar, una distancia que el paciente debe caminar a diario con un objetivo de tiempo y las preguntas diarias que el paciente debe responder.

- Recibir feedback del paciente y poder comprobar que realmente está siguiendo las pautas marcadas por el equipo médico.

Este objetivo también se considera cumplido. La aplicación del móvil enviará los datos de las pautas que va realizando al back-end y este lo almacenará en la base de datos. Estas pautas que la aplicación ha subido a la base de datos son accesibles desde la página que usa el equipo médico por lo tanto los médicos están al día de las pautas que realiza cada uno de los pacientes, como las realiza y si se han realizado de forma satisfactoria.

- La web debe permitir que el médico pueda actualizar las pautas de un paciente si tras recibir el feedback considera que este ha sido subestimado o sobreestimado

Como se ha dicho en el objetivo anterior la página web permite al médico ver los resultados de las pautas que se le han indicado al paciente. Y además el equipo médico podrá actualizar las pautas de cada uno de los pacientes cuando lo deseen. Una vez las

pautas se hayan actualizado desde la página web, a partir de ese momento, las pautas que se le indicarán al paciente en la aplicación móvil serán las nuevas pautas definidas por el equipo médico.

- La página web también deberá permitir al equipo médico ver las gráficas del desarrollo del paciente. La web deberá crear graficas personalizadas para cada paciente en las que se muestre la evolución del paciente a lo largo del proceso del preoperatorio, la operación y el postoperatorio.

En la página web se ofrece la opción de mostrar una serie de graficas personalizadas de cada uno de los pacientes. Estas graficas se crearán con los datos que el paciente ha ido almacenando en el sistema cada día con el seguimiento (o no) de las pautas indicadas por el médico. Se mostrarán diferentes gráficos en los cuales se indicará, además, cual es la fecha de la operación del paciente, por lo que las tablas aparecerán divididas en el preoperatorio y el postoperatorio para que el equipo médico pueda ver de forma gráfica el rendimiento del paciente antes y después de la operación. Por lo tanto, este objetivo también se considera cumplido.

- La creación de una aplicación móvil que le permita al paciente de una cirugía torácica disponer en el móvil de todas las pautas que ha de realizar para el correcto desarrollo tanto de la fase de preoperatorio como la de postoperatorio.

Este objetivo se considera cumplido ya que se ha creado una aplicación móvil la cual permite ver al paciente todas las pautas que el médico le ha indicado, como los ejercicios que ha de realizar a diario (con una breve explicación de los mismos y un video explicativo de como se ha de realizar el ejercicio de forma correcta), las preguntas que el paciente ha de contestar diariamente y los objetivos de caminar que el paciente ha de cumplir diariamente. Además, como el equipo médico está en disposición de cambiar las pautas en cualquier momento si lo desean pueden cambiar las pautas cuando se realice la operación para ajustarlas al postoperatorio.

- Los ejercicios que el equipo médico le asigne al paciente irán acompañados por un video explicativo de cómo se realiza el ejercicio de forma correcta.

En la aplicación al mostrar uno de los ejercicios que se le han asignado al paciente existe un botón el cual redirigirá al paciente a la aplicación de “YouTube” en la cual el paciente podrá ver el video de la explicación para realizar el ejercicio de forma correcta. Por lo tanto, este objetivo no se puede considerar totalmente cumplido ya que la idea inicial era que la reproducción del video se realizase dentro de la aplicación.

- En la aplicación habrá un apartado de recomendaciones en el cual se mostrarán una serie de recomendaciones generales las cuales irán acompañadas por una descripción y un gif para que las recomendaciones se comprendan más fácilmente.

Este objetivo se considera cumplido. En la página principal de la aplicación se encuentra un apartado de recomendaciones en el cual tras clickar sobre él se abrirá una lista de recomendaciones. El paciente podría clickar sobre cada una de las recomendaciones listadas y al hacerlo se la abrirá la página de la recomendación en la

cual se le mostrará un texto explicativo de la recomendación, el título de la recomendación y un gif animado el cual explique la recomendación de una forma gráfica. Cada recomendación dispondrá de su gif personalizado.

- Si el paciente no ha realizado los ejercicios a una hora determinada la aplicación se encargará de recordar al paciente que todavía no ha realizado los ejercicios.

Este objetivo se considera que no ha sido cumplido, al menos no de una forma completa. Esta función se trata de una función que habría que desarrollar para cada uno de los sistemas operativos de forma específica. Es cierto que para Android se ha desarrollado a funcionalidad de que la aplicación compruebe si el paciente ha cumplido el objetivo de distancia caminada y que si no la ha cumplido le avise de que todavía no ha recorrido suficientes metros. Pero para UWP (Universal Windows Platform) e IOS esta funcionalidad no ha sido desarrollada. Existen varios motivos por los que no se ha desarrollado esta funcionalidad para estos dos sistemas operativos: se trata de una funcionalidad bastante compleja de la cual no existe mucha información de cómo se puede desarrollar para estos sistemas operativos, además de que los tiempos del proyecto se han visto ajustados y se ha decidido dar prioridad a otras funcionalidades. Por último, para compilar el código de IOS es necesario contar con un ordenador de la marca Apple, el cual no dispongo, por lo que el código se habría desarrollado, pero no se podría haber testeado.

- Durante la caminata la aplicación podrá enviar avisos al paciente si está caminando muy deprisa o muy despacio. Además, le dirá cuando ha llegado a la mitad de su recorrido recomendado por el personal médico para que si quiere pueda dar la vuelta y dirigirse a casa.

En la aplicación durante la caminata se mostrará en todo momento la distancia que el paciente ha recorrido y el tiempo que ha transcurrido desde el momento en el que ha comenzado a caminar. Así mismo también se muestra el porcentaje que el paciente lleva caminado en cada momento. El paciente podría en cualquier instante el porcentaje al que esta y cuando llega al 50% dar la vuelta. Sin embargo, este objetivo no se considera cumplido.

- La aplicación deberá recoger información de la actividad diaria del paciente y realizarle las preguntas asignadas por el equipo médico a diario. Por ejemplo, si te trata de un paciente fumador el médico le podría asignar las siguientes preguntas: “¿Has fumado hoy?” y “¿Cuántos cigarrillos has fumado?”

Este objetivo se considera cumplido. La aplicación del paciente recoge la distancia que este ha caminado, las respuestas que ha dado a las preguntas y los ejercicios que ha realizado. Tras recoger los datos los actualiza en la base de datos y a partir de ese momento están disponible para que el equipo médico los consulte desde la aplicación web.

- El aprendizaje personal de tecnologías que hasta el momento desconocía y que además son requeridas actualmente por las empresas.

Adquirir conocimientos del desarrollo de páginas web y el desarrollo de aplicaciones móviles para distinto sistemas operativos.

A diferencia de los objetivos listados anteriormente este se trata de un objetivo totalmente personal y subjetivo. Hasta la realización de este proyecto nunca se habría utilizado la mayoría de las herramientas utilizadas, así como nunca se había realizado un back-end y los dos front-end en los frameworks utilizados.

El proceso de aprendizaje que el proyecto ha requerido ha sido largo y tras la finalización del desarrollo y comprobar los resultados obtenidos tanto en el back-end como en la página web y aplicación móvil se considera que el objetivo ha sido cumplido desde el punto de vista más objetivo posible, para un objetivo que se considera totalmente subjetivo.

8.2. Planificación temporal

En este apartado se analizará la planificación temporal del proyecto. Pese haber logrado cumplir la mayoría de objetivos no se ha logrado ajustar el proyecto a la planificación temporal inicial. A continuación, se muestra un gráfico y una tabla comparativa en los que se puede ver cuales has sido las desviaciones temporales que ha sufrido la planificación del proyecto.

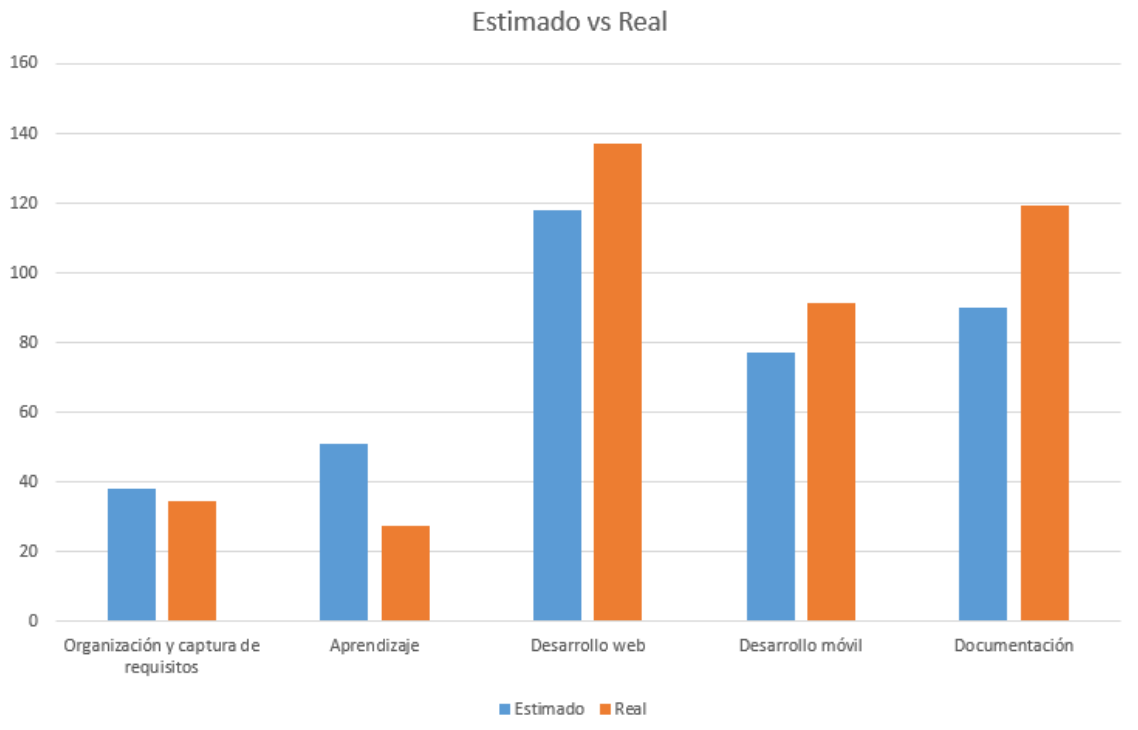


Tabla 71 - Comparación horas estimadas y horas reales

Tabla 72 - Horas estimadas vs horas reales

	Horas estimadas	Horas reales
Organización y captura de requisitos	38	34.5
Aprendizaje	51	27.5
Desarrollo web	118	127
Desarrollo móvil	77	91.5
Documentación	90	119.5
Total	374	410.5

Como se puede ver en la tabla anterior el total de horas invertidas en el proyecto ha sido de 410 frente a las 374 horas que se había estimado que supondría la realización del proyecto. Esto supone que ha habido una desviación de 36 horas, es decir casi un 10% de las horas estimadas.

Pese a que el desajuste total de la planificación no llegue al 10% internamente en los paquetes se han producido desajustes que se han compensado un poco. Como se puede ver en la tabla a la fase de aprendizaje se le dedicaron menos horas de las que se había estimado en un principio y sin embargo a las fases de desarrollo se les ha dedicado más tiempo del estimado. Estos desajustes en parte se deben a que se acorto la fase de aprendizaje antes de tiempo para poder juntar esta fase con la de desarrollo y de esta manera aprender mientras se desarrollaba.

El paquete de documentación del proyecto también ha sufrido bastante desviación. Esto se debe a que al realizar la estimación de horas se subestimo la extensión que tendría la documentación del proyecto, pensando que la memoria sería mucho más corta de lo que finalmente ha terminado siendo.

8.3. Evaluación económica

Como se ha comentado en el apartado anterior el proyecto ha sufrido un pequeño aumento en las horas invertidas respeto a las horas que se estimaron que se dedicarían. Por ese motivo es necesario rehacer la evaluación económica, para ajustarla al tiempo realmente dedicado al proyecto.

Tabla 73 - Evaluación económica tiempo real

Coste total	
Mano de obra	6.039,3€
Hardware	65€
Licencias	204€
Otros gastos	379€
Total:	6.687,3€

Como se puede ver en la tabla el costo del proyecto sería de 6.687,3€ lo cual ha supuesto un aumento de 530,1€ respecto al costo estimado al comienzo del proyecto. del proyecto.

8.4. Riesgos que han ocurrido

Durante la realización del proyecto varios de los riesgos que se analizaron han sucedido.

Uno de los riesgos que ha sucedido es el que se llamó “Exámenes y prácticas”. Es cierto que en la planificación temporal ya se tuvieron en cuenta los exámenes de enero y el tiempo que se invertiría en estudiar para dichos exámenes. Pero pese a ello, el resto del año la carga de trabajo ha sido más alta de lo esperada, lo cual ha supuesto que el número de horas que se le podía dedicar cada día al proyecto fuese menor que el esperado en un principio. Esto ha supuesto que la defensa del trabajo se tuviese que retrasar hasta septiembre cuando en un principio la intención fue defender el trabajo de fin de grado en julio.

Otro de los riesgos que ha sucedido ha sido el de los cambios de requisitos por parte del cliente. En un principio se pensó que en la parte de ejercicios de la aplicación el usuario tendría que marcar el ejercicio como realizado o no realizado. Sin embargo, cuando el cliente envió unos ejemplos de cómo serían los ejercicios que tendrían que realizar los pacientes se comprobó que sería necesario que el paciente introdujese la cantidad de repeticiones que había realizado. Por suerte esto no supuso demasiados cambios en el proyecto, además de que sucedió cuando todavía no se había terminado de desarrollar el proyecto.

Por último, apareció otro riesgo más, el riesgo de pérdida de información. La memoria del proyecto se ha realizado utilizado Microsoft Word. Al abrir Word un día para seguir realizando la memoria del proyecto me encontré que lo que había realizado el día anterior no se había guardado. Por suerte Word ofrece una especie de control de versiones en el que va realizando autoguardados. De esta forma se pudo recuperar el trabajo realizado el día anterior.

8.5. Líneas futuras

En eTorax como en la gran mayoría de proyectos una vez el proyecto se considera finalizado se pueden seguir añadiendo funcionalidades y mejoras al proyecto. En este apartado se analizarán cuáles serían las futuras mejoras que se le podrían añadir al proyecto.

Como se ha comentado en el apartado 8.1 el objetivo de añadir notificaciones que avisen al usuario cuando no ha llegado al objetivo a una hora determinada no se ha considerado cumplido ya que esta función solo esta implementada para el sistema operativo Android. Pese a que en la actualidad el sistema operativo de Android ocupe

casi el 75% del mercado mundial (Casas, 2019) esta se considera una de las mejoras a las que habría que darle mayor importancia.

Otra de las mejoras que se podría realizar seria la integración del control de la caminata del usuario con el servicio de Google Fit. Se cree que se podría usar es servicio de Google Fit para controlar la distancia que ha caminado un paciente de una forma más exacta que la desarrollada en el proyecto.

Otra de las funcionalidades que se podría implementar en un futuro sería la de que el propio médico pudiese enviarle un mensaje a un usuario determinado si ve que no está cumpliendo los objetivos. El médico podría escribirle un mensaje como si se tratase de un correo electrónico en él podría escribir lo que quisiese y que le llegaría al paciente en forma de notificación.

Por otra parte, al tratarse d un proyecto con un cliente real seguramente durante el uso de la aplicación o la página web los usuarios encuentran posibles mejoras que les serian útiles.

8.6. Reflexión personal

En lo personal este proyecto me ha servido para darme cuenta de muchas cosas y aprender otras tantas.

Antes de comenzar con el proyecto desconocía la mayoría de herramientas y lenguajes que he usado para desarrollarlo, y tenía dudas de si sería capaz de adaptarme a los nuevos lenguajes y herramientas. Sin embargo, me he dado cuenta de mi capacidad para adaptarme a los nuevos lenguajes y herramientas.

Otra de las cosas que más temía a la hora de realizar el proyecto era la de enfrentarme a los problemas que pueden surgir a lo largo del desarrollo del mismo, en específico a los problemas del desarrollo del código del proyecto. Durante el grado la gran mayoría de veces en las que se desarrolla código es de forma grupal y de esta forma cuando surge un problema hay compañeros que te pueden enseñar y ayudar a solventar dicho problema. Sin embargo, el desarrollo del proyecto se ha realizado de forma individual por lo que los problemas que han surgido a lo largo del desarrollo también se han tenido que enfrentar de forma individual. Esto me ha servido para darme cuenta de mi capacidad de superación.

Por lo demás estoy feliz con el trabajo que he realizado en este proyecto y con la cantidad de conocimiento que he adquirido en la realización del mismo. Además de por haber desarrollado un proyecto que un futuro será utilizado tanto por los pacientes como por los médicos. Que hará que el trabajo de los médicos sea más sencillo y que los pacientes que se enfrenten a una cirugía de tórax tengan la aplicación para ayudarles lo máximo posible tanto en el preoperatorio como el postoperatorio.

Bibliografía

Bootstrap. (s.f.). *https://getbootstrap.com*. Obtenido de https://getbootstrap.com/docs/4.0/layout/grid/

Casas, A. (25 de Febrero de 2019). *https://www.pcworld.es*. Obtenido de https://www.pcworld.es/articulos/smartphones/iphone-vs-android-cuota-de-mercado-3692825/

Diaz, I. A. (12 de abril de 2014). *serprogramador.es*. Obtenido de https://serprogramador.es/que-es-frontend-y-backend-en-la-programacion-web/

Imaginaformacion. (s.f.). *www.imaginaformacion.com*. Obtenido de https://www.imaginaformacion.com/que-es-xamarin-como-crear-un-proyecto/

JSON. (s.f.). *www.json.org*. Obtenido de https://www.json.org/json-es.html

La información. (1 de Febrero de 2019). *Practicopedia*. Obtenido de https://www.lainformacion.com/practicopedia/cual-es-el-coste-real-de-un-trabajador-para-una-empresa/6491464/

Merino, M. (12 de julio de 2014). *www.ticbeat.com*. Obtenido de https://www.ticbeat.com/tecnologias/que-es-una-api-para-que-sirve/

Microsoft. (13 de Febrero de 2019). *https://docs.microsoft.com*. Obtenido de https://docs.microsoft.com/es-es/xamarin/essentials/geolocation?context=xamarin%2Fxamarin-forms&tabs=uwp

NodeJS. (s.f.). *https://nodejs.org*. Obtenido de https://nodejs.org/es/about/

ricardocelis. (2017). *https://platzi.com*. Obtenido de https://platzi.com/blog/que-es-heroku-y-para-que-me-sirve/

Suarez, M. (26 de octubre de 2016). *https://es.quora.com*. Obtenido de https://es.quora.com/Qu%C3%A9-es-un-framework

Tomàs, E. (25 de abril de 2014). *desarrolloweb.com*. Obtenido de https://desarrolloweb.com/articulos/que-es-rest-caracteristicas-sistemas.html

Zuriarrain, J. M. (4 de abril de 2017). *https://elpais.com*. Obtenido de https://elpais.com/tecnologia/2017/04/04/actualidad/1491296467_396232.html

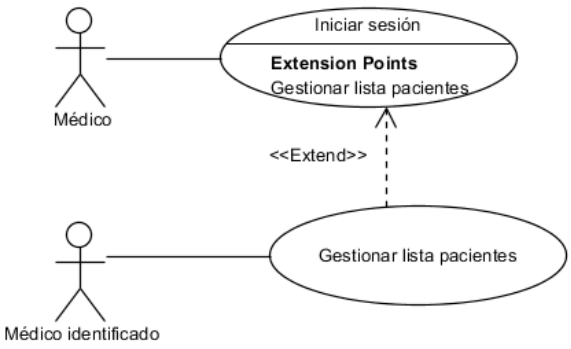
Anexo I: Casos de uso extendidos

En este apartado se mostrarán los casos de uso extendidos de los casos de uso ya presentados en el apartado 4.3. Se procederá a realizar un análisis detallado de cada caso de uso. Este anexo se dividirá en dos secciones, una para la página web y la otra para la aplicación móvil.

Página web

En esta sección del anexo se presentarán los casos de uso extendidos de la página web del proyecto.

Iniciar sesión



Nombre: Iniciar sesión.

Descripción: Permite a un usuario no identificado iniciar sesión en el sistema mediante unas credenciales. Si se inicia sesión con las credenciales correctas se mostrará la pantalla principal con la lista de pacientes.

Actores: Médico.

Precondiciones: La sesión no se encuentra iniciada.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. El médico entra en la página para iniciar sesión. (Ilustración 68)

   [Si no introducen credenciales de acceso]

   2A. Se muestra el error avisando de que no ha introducido las credenciales. (Ilustración 69)

   [Si las credenciales de acceso son incorrectas]

   2B. Se muestra el error avisando de que las credenciales introducidas son incorrectas. (Ilustración 70)

   [Si las credenciales introducidas son correctas]

2C. Se redirigirá al médico a la pantalla principal de la página web. (Ilustración 71)
Postcondiciones: En caso de que las credenciales sean correctas el médico habrá iniciado sesión en el sistema.
Interfaz gráfica:

Ilustración 68 - Pagina d inicio de sesión de la página web

Ilustración 69 - Iniciar sesión sin datos

Ilustración 70 - Credenciales introducidas incorrectas

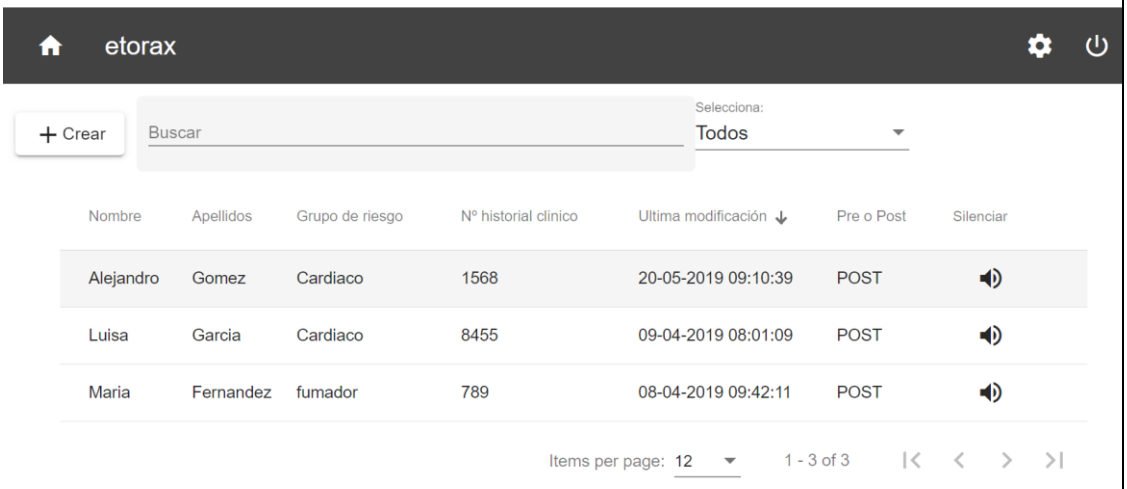
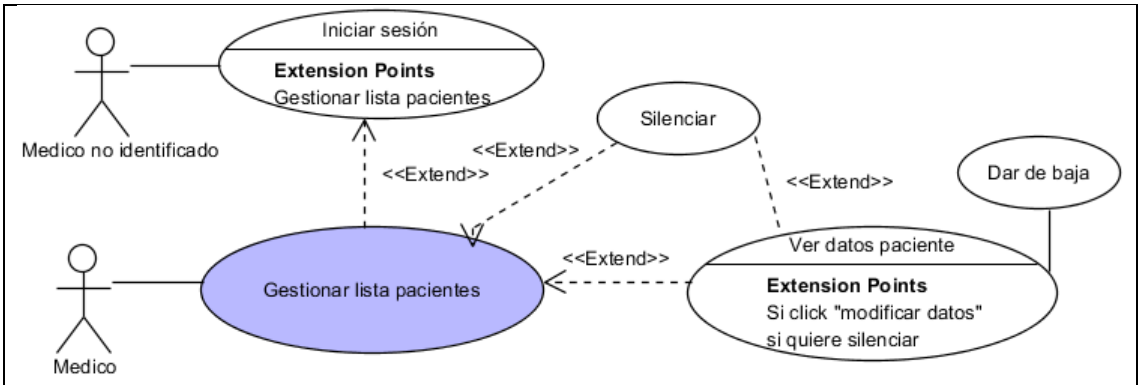


Ilustración 71 - Página principal

Gestionar pacientes



Nombre: Gestionar lista pacientes.

Descripción: Permite al médico visualizar y gestionar la lista de pacientes.

Actores: Médico.

Precondiciones: Ninguna

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. El médico visualiza la lista de los pacientes. (Ilustración 72)

   [Si pulsa sobre la línea de un paciente]

   2A. El médico es redirigido a la página del paciente. (Ilustración 73)

   [Si pulsa sobre el botón de silenciar paciente.]

   2B. El paciente es silenciado y se redirige al médico a la página del paciente.

Postcondiciones: Ninguna

Interfaz gráfica:

Nombre	Apellidos	Grupo de riesgo	Nº historial clinico	Ultima modificación	Pre o Post	Silenciar
Alejandro	Gomez	Cardiaco	1568	20-05-2019 09:10:39	POST	
Luisa	Garcia	Cardiaco	8455	09-04-2019 08:01:09	POST	
Maria	Fernandez	fumador	789	08-04-2019 09:42:11	POST	

Items per page: 12 1 - 3 of 3

Ilustración 72 - Lista gestionar pacientes

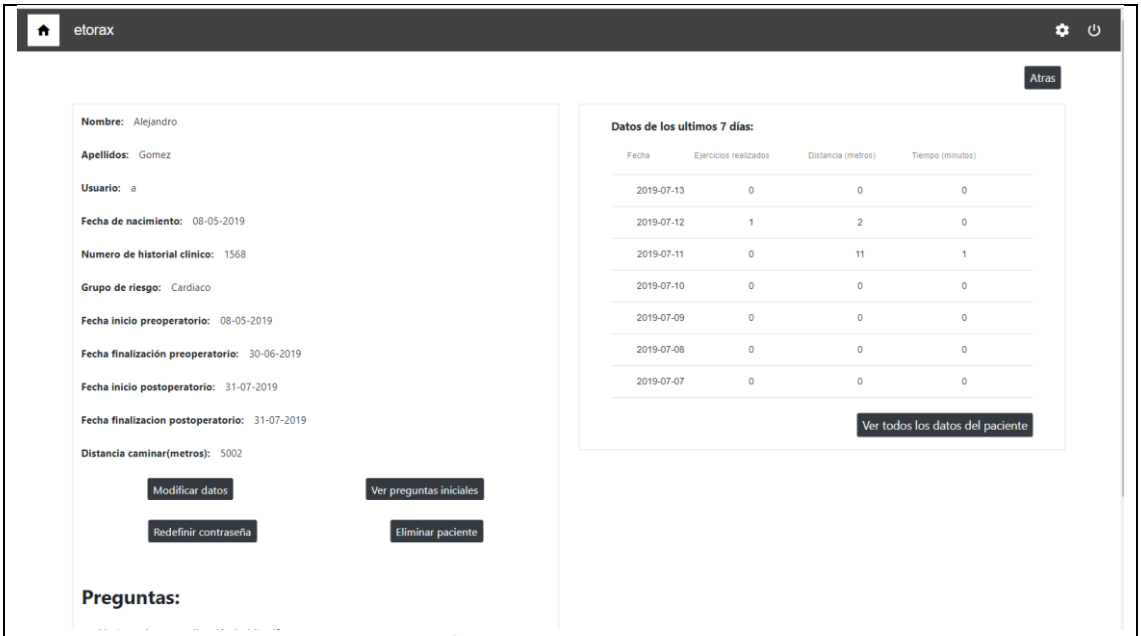
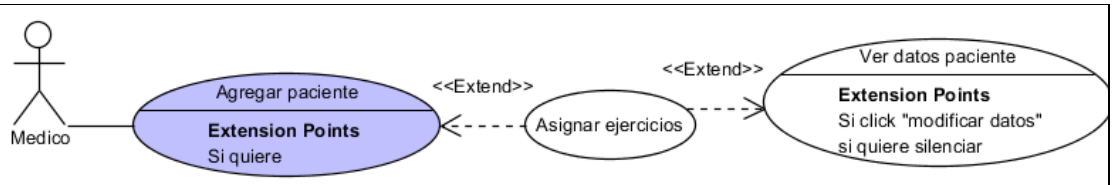


Ilustración 73 - Pagina del paciente

Agregar paciente



Nombre: Agregar paciente.
Descripción: El usuario puede crear un nuevo paciente en el sistema que luego tendrá acceso a la aplicación web.
Actores: Médico
Precondiciones: El médico tiene que tener la sesión iniciada.
Requisitos no funcionales: Ninguno.
Flujo de eventos: [El médico ha hecho click sobre la opción de crear paciente] 1. Se muestra el formulario de creación de pacientes. (Ilustración 74Ilustración 72) [El usuario pulsa el botón de continuar sin introducir todos los datos.] 2A. Se marcan en color rojo las opciones que faltan por introducir y son obligatorias. (Ilustración 75) [El usuario elige un grupo de riesgo

2B1. Se escogen de forma automática las pautas para el grupo de riesgo seleccionado, y se permite al usuario modificarlas. (Ilustración 76)

[El usuario pulsa continuar]

2B2. Se redirige el usuario a la ventana para que elija las repeticiones que desea asignarle al usuario en cada ejercicio, por defecto aparecen en cada ejercicio el número de repeticiones por defecto para ese ejercicio. (Ilustración 77)

[El usuario hace click sobre crear paciente]

2B3. Se redirige al usuario a la página del usuario recién creado. (Ilustración 78)

Postcondiciones: El usuario ha sido creado–

Interfaz gráfica:

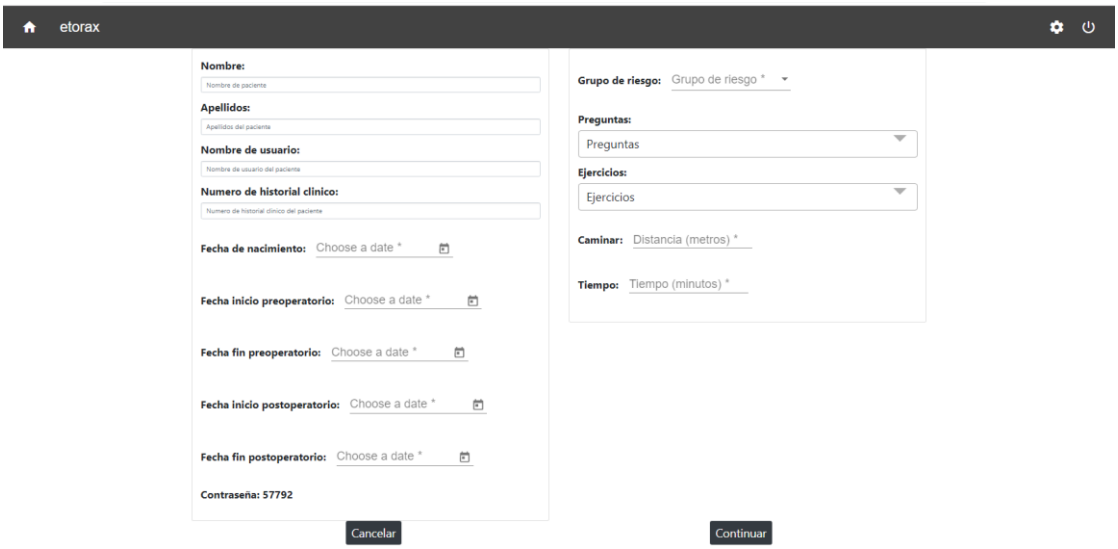


Ilustración 74 - Formulario de creación de pacientes.

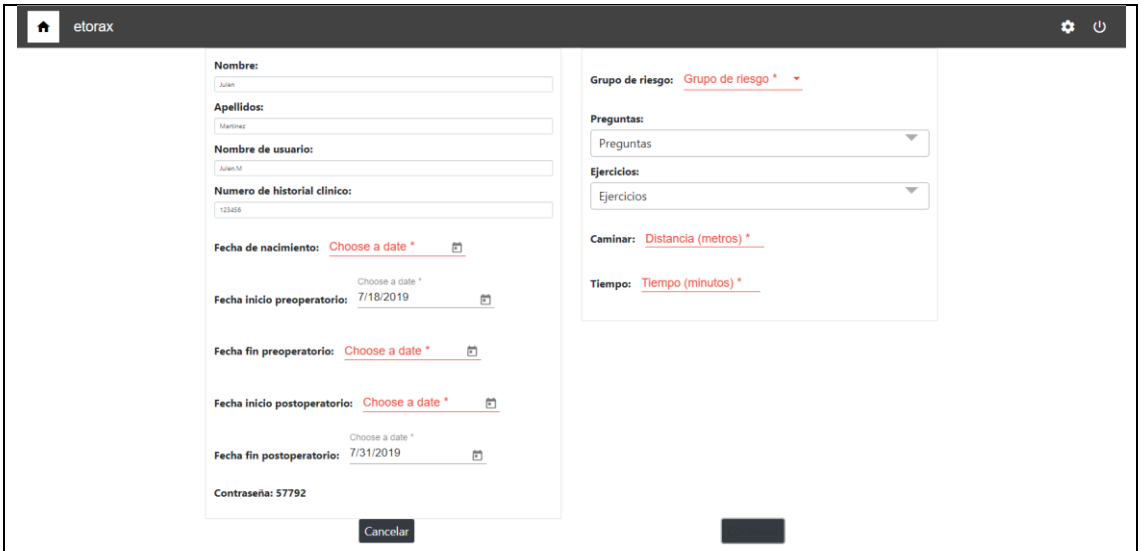


Ilustración 75 - Error creación del paciente

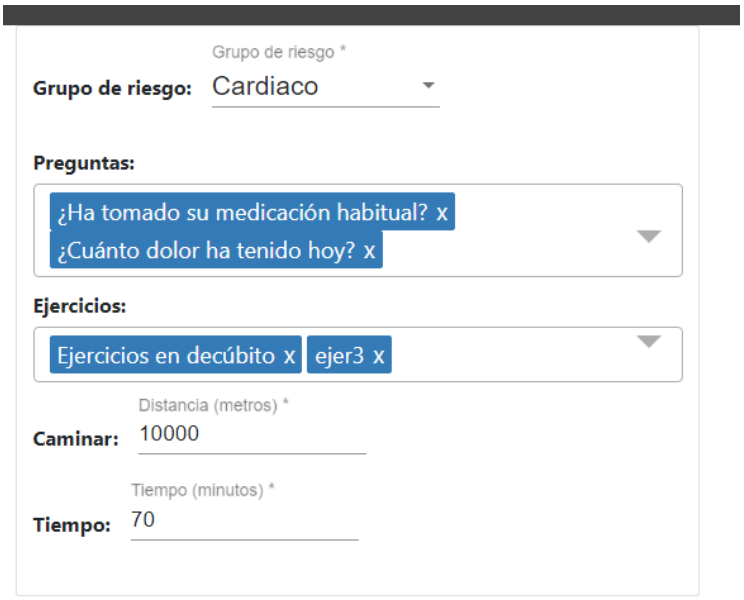


Ilustración 76 - Elegir grupo de riesgo

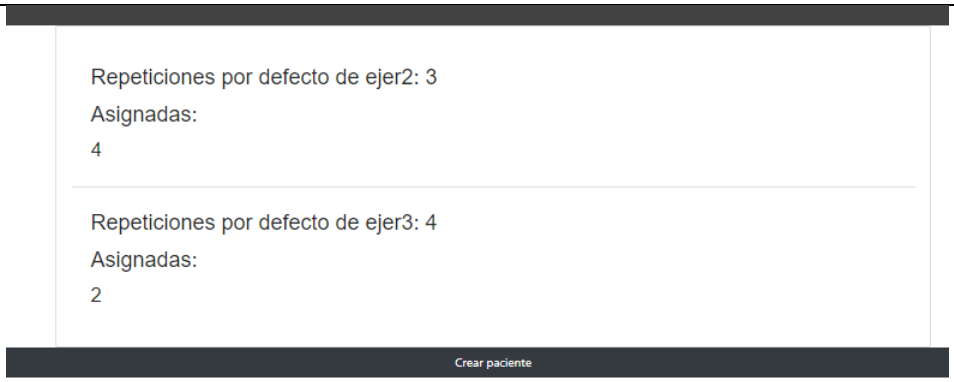


Ilustración 77 - Elegir repeticiones de los ejercicios

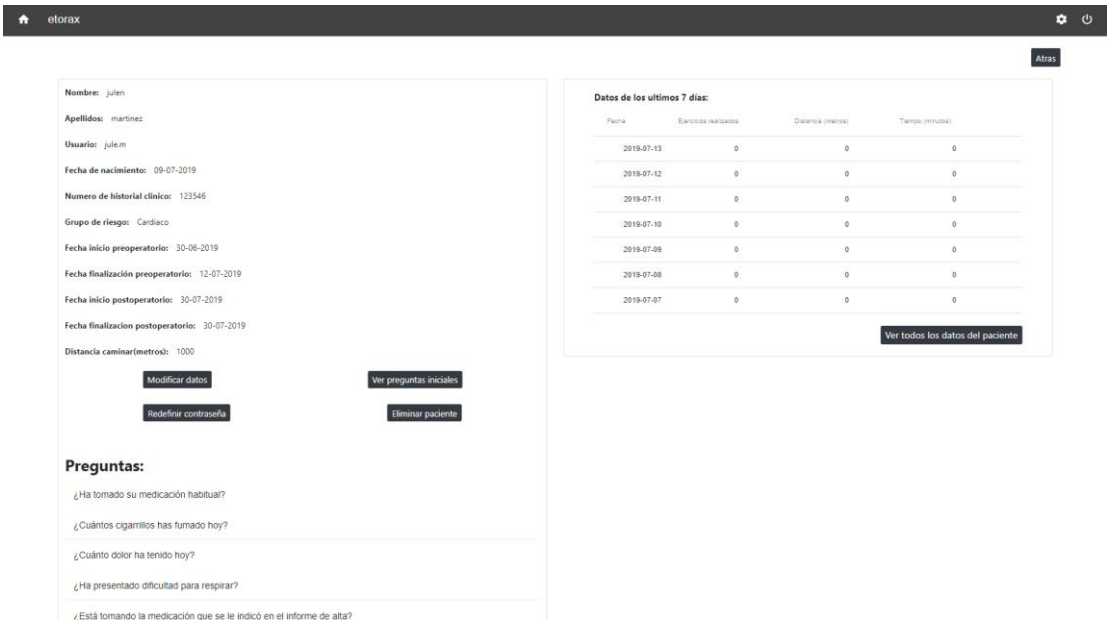
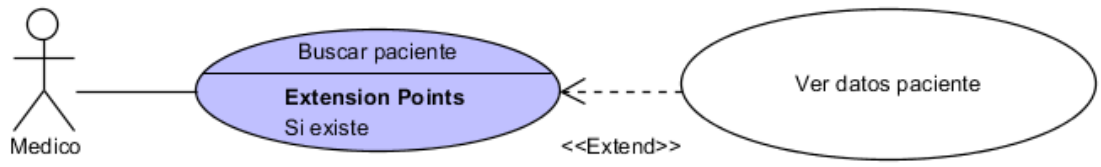


Ilustración 78 - Pagina del usuario recién creado

Buscar paciente



Nombre: Buscar paciente
Descripción: En la página principal de la aplicación el usuario podrá filtrar la tabla para encontrar el paciente deseado
Actores: Médico.
Precondiciones: El médico debe tener la sesión iniciada.
Requisitos no funcionales: Ninguno.

Flujo de eventos: [El médico introduce en la barra de búsqueda los valores a través de los cuales desea buscar] [No existen coincidencias] 1A. Se muestra la tabla vacía. (Ilustración 79) [Se encuentran resultados.] 1B. Se muestra la tabla con los resultados obtenidos. La búsqueda se puede realizar por cualquier columna. (Ilustración 80)
Postcondiciones: Ninguna.
Interfaz gráfica:

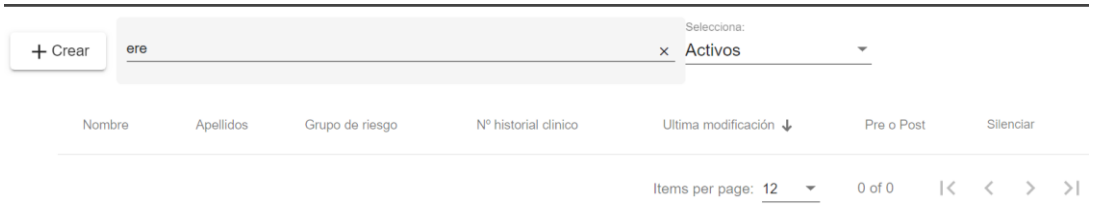


Ilustración 79 - No se han encontrado pacientes

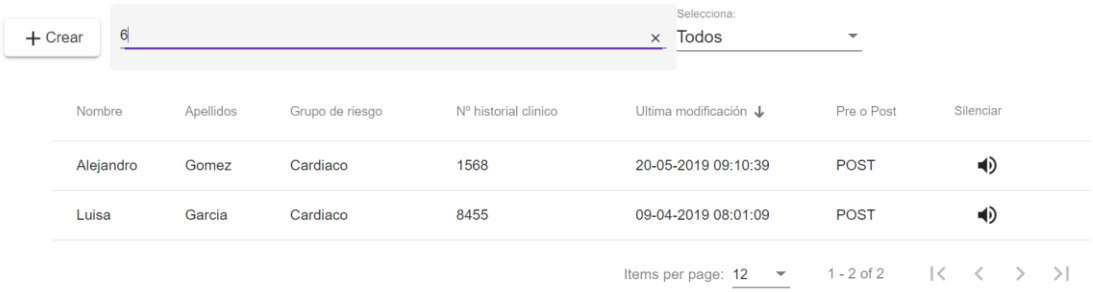
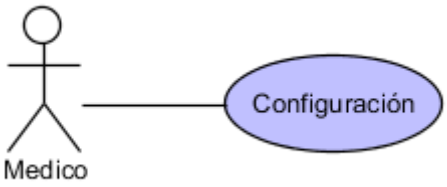


Ilustración 80 - Tabla con resultados

Configuración



Nombre: Configuración.
Descripción: Configuración de la aplicación.
Actores: Médico
Precondiciones: Tener la sesión inicia.
Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. Se muestran todas las opciones de configuración disponibles. (Ilustración 81)

   [El usuario pulsa el botón de crear pregunta sin introducir la pregunta.]

   2A. Se avisa al usuario de que no ha introducido la pregunta. (Ilustración 82)

   [El usuario introduce una pregunta que ya existe]

   2B. Se avisa al usuario de que la pregunta ya existe. (Ilustración 83)

   [El usuario introduce una pregunta correcta]

   2C. Se avisa al usuario de que la pregunta se ha introducido de forma correcta (Ilustración 84).

   [El usuario intenta crear un grupo de riesgo sin alguno de los parámetros]

   2D. Se avisa al usuario de que o ha introducido uno de los parámetros. (Ilustración 85)

   [El usuario intenta crear un grupo de riesgo que ya existe.]

   2E. Se avisa al usuario de ya existe el grupo de riesgo que intenta crear. (Ilustración 86)

   [El usuario crea un grupo de riesgo de forma correcta]

   2E. Se avisa al usuario de que el grupo de riesgo ha sido creado. (Ilustración 87)

   [El usuario cambia el idioma.]

   2F. Se cambia el idioma de la aplicación. (Ilustración 88)

   [El usuario intenta exportar los datos de un paciente que no existe]

   2G. Se avisa al usuario que el paciente no existe. (Ilustración 89)

Postcondiciones: Ninguna.

Interfaz gráfica:

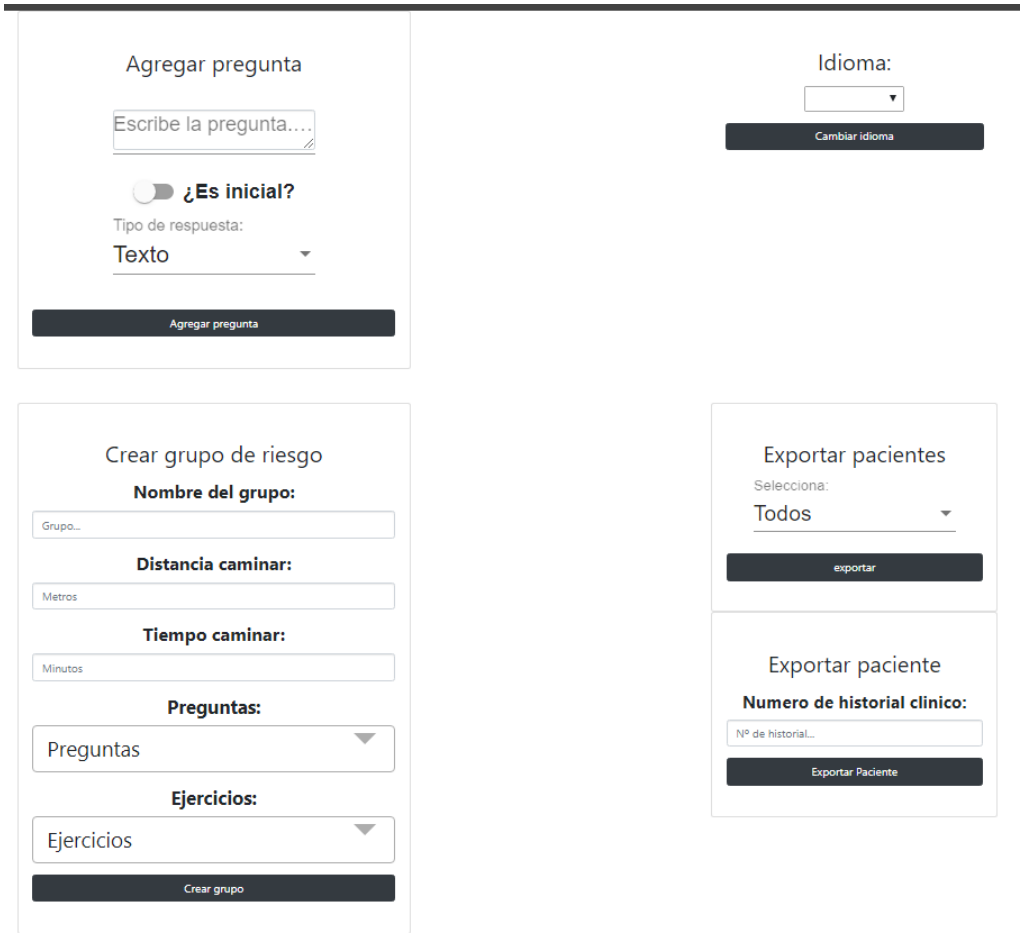


Ilustración 81 - Opciones de configuracion

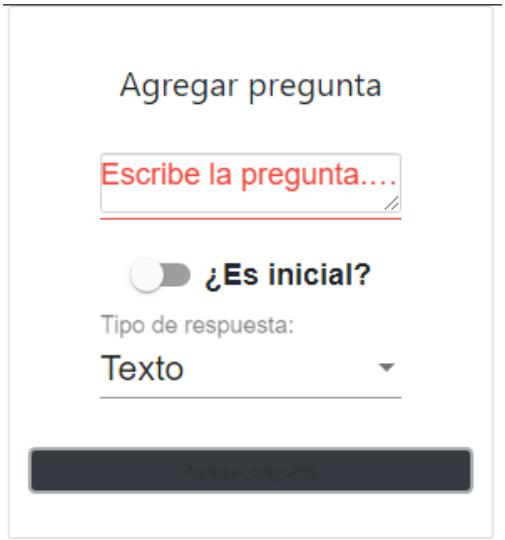


Ilustración 82 - No ha introducido la pregunta

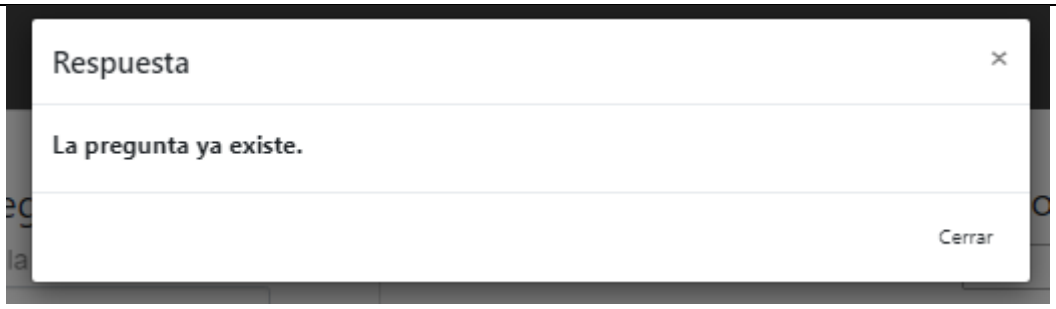


Ilustración 83 - La pregunta ya existe

Ilustración 84 - Pregunta creada

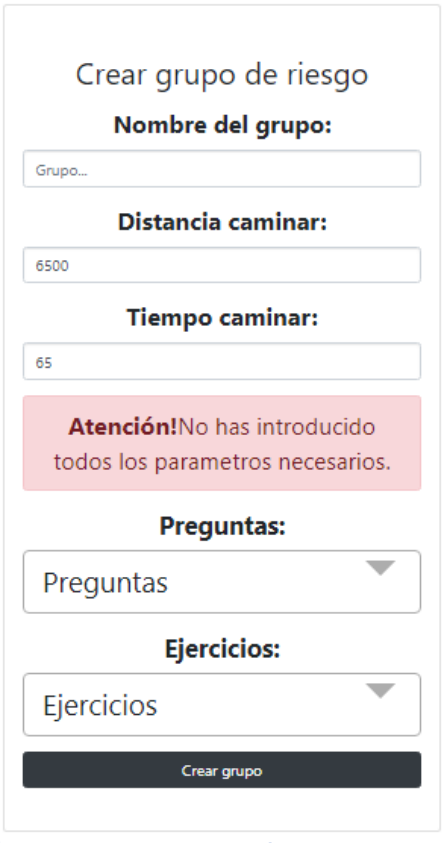


Ilustración 85 - Crear grupo sin un parametro

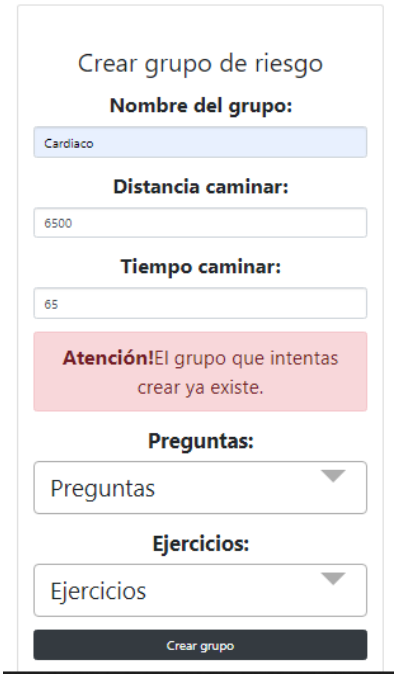


Ilustración 86 - El grupo de reisgo ya existe

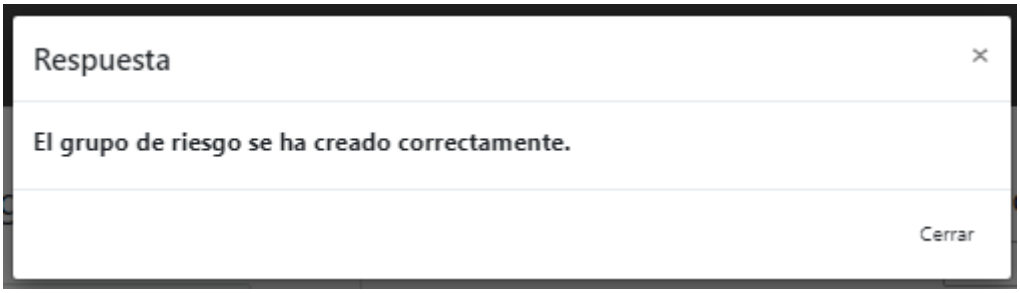


Ilustración 87 - Grupo de riego creado correctamente

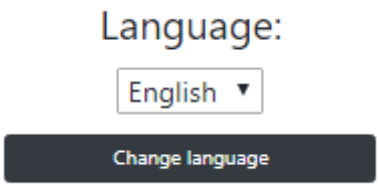


Ilustración 88 - Cambiar idioma

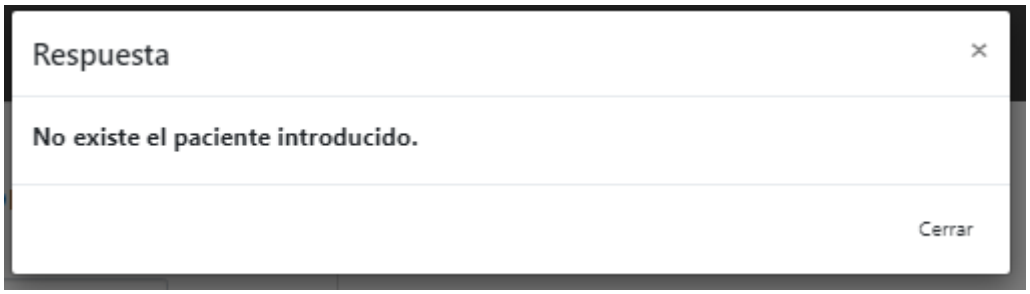
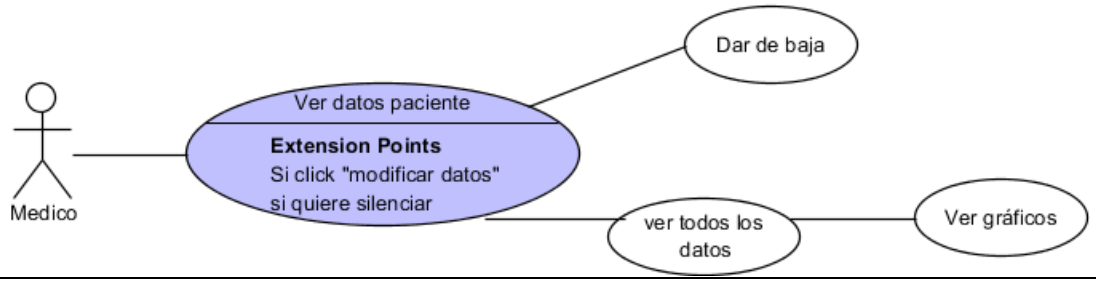


Ilustración 89 - Exportar paciente que no existe

Ver datos paciente



Nombre: Ver datos paciente

Descripción: El médico podrá visualizar todos los datos del paciente y acceder a otras opciones.

Actores: Médico

Precondiciones: El médico debe tener la sesión iniciada.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

[El usuario accede a la página del paciente]

1. Se muestra la información del paciente. (Ilustración 90)

[El médico hace click sobre el botón “Atrás”]

2A. Se redirige al usuario a la página principal. (Ilustración 91)

[El médico hace click sobre “Modificar paciente”]

2B. Se redirige al usuario a la pantalla con el formulario para modificar al paciente. (Ilustración 92)

[El médico hace click sobre el botón de “Redefinir contraseña”]

2C. Se muestra un mensaje de confirmación. (Ilustración 93)

[El médico pulsa sobre “Modificar contraseña” en el mensaje de confirmación.]

2C1. Se muestra el mensaje con la nueva contraseña. (Ilustración 94)

[El médico pulsa sobre “Ver preguntas iniciales”]
[El paciente ha respondido a las preguntas iniciales.]

2D1. Se muestran las preguntas iniciales del paciente con las respuestas que ha dado. (Ilustración 95)

[El paciente no ha respondido las preguntas iniciales todavía]

2D2. Se muestra la alerta sin las respuestas. (Ilustración 96)

[El usuario pulsa sobre “Eliminar paciente”]

2E. Salta la alerta de confirmación para eliminar al paciente del sistema. (Ilustración 97)

[El usuario pulsa sobre la opción de “Ver todos los datos del paciente”]

2F. Se redirige al usuario a la página con todos los datos de las pautas realizadas por el paciente. (Ilustración 98)

Postcondiciones: Ninguna.

Interfaz gráfica:

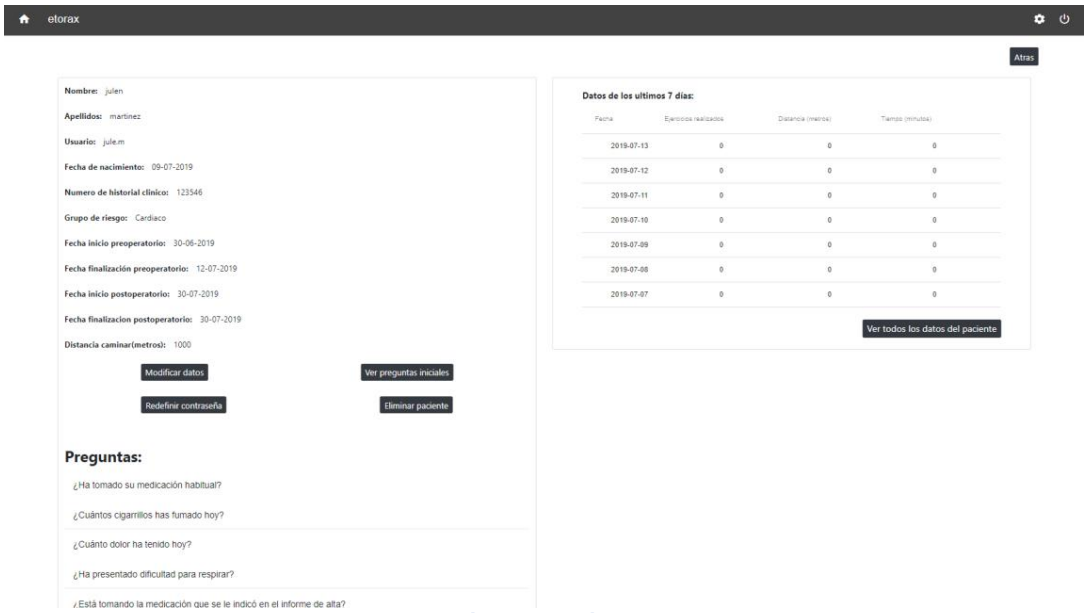


Ilustración 90 - Página del paciente.

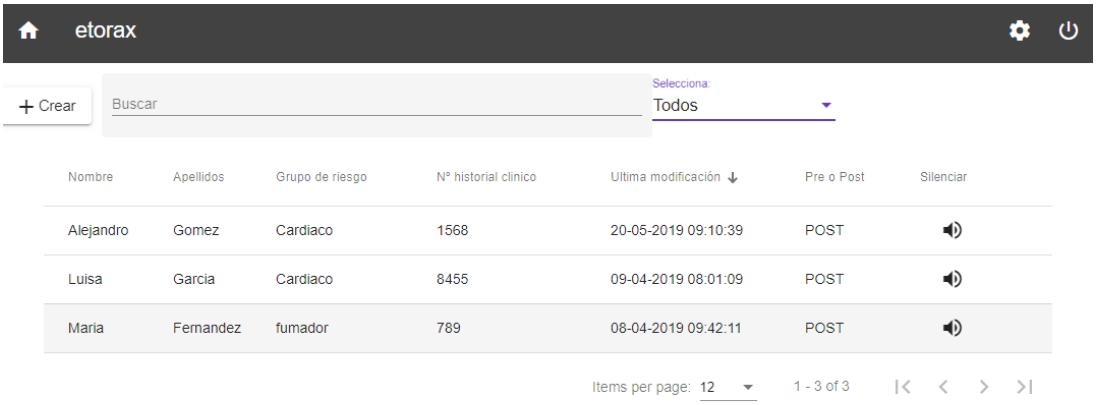


Ilustración 91 - Pagina principal

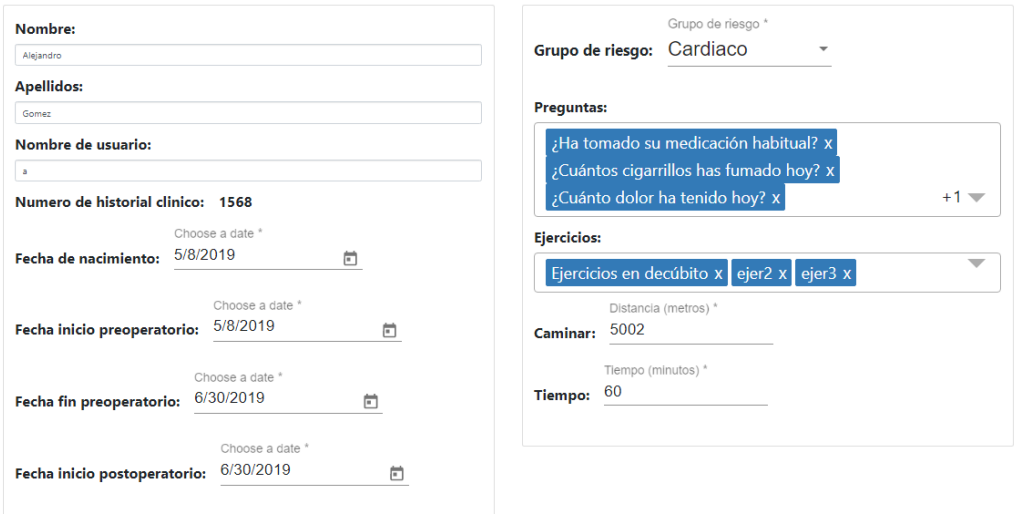


Ilustración 92 - Modificar paciente

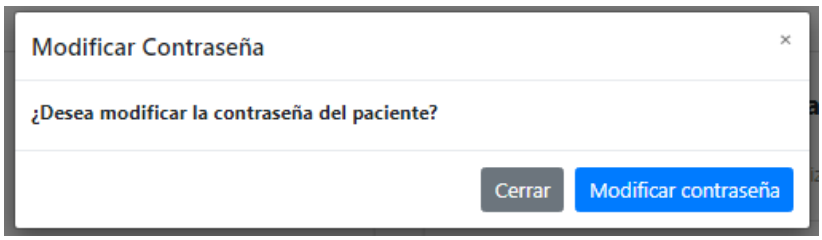


Ilustración 93 - Confirmacion modificar contraseña

Ilustración 94 - Contraseña modificada

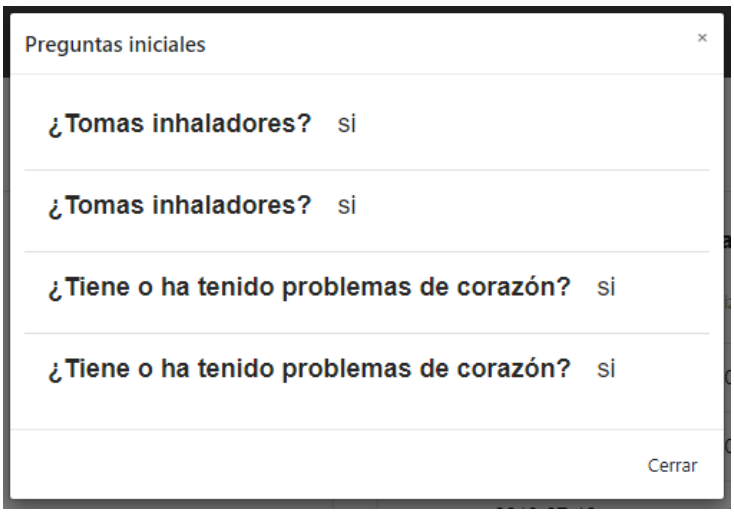


Ilustración 95 - Preguntas iniciales

Ilustración 96 - Preguntas iniciales no respondidas

Ilustración 97 - Confirmacion eliminar paciente

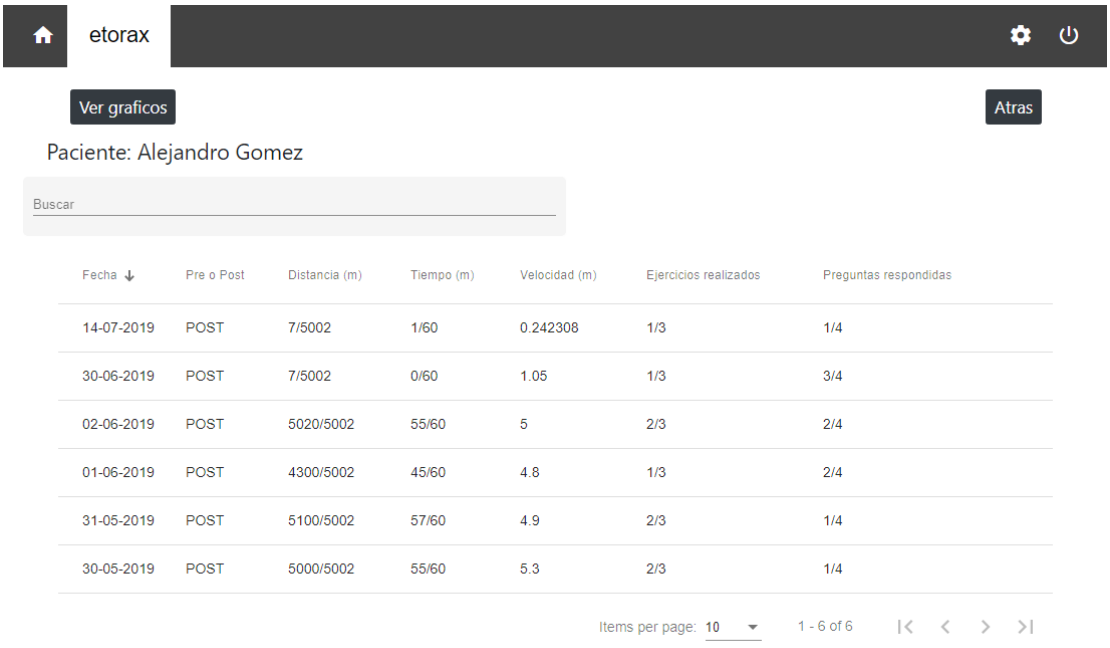
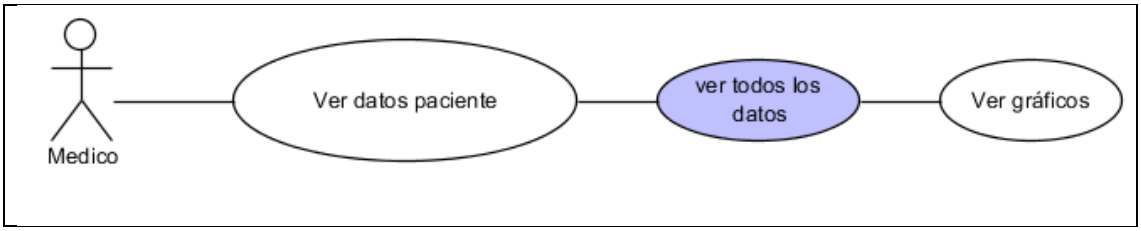


Ilustración 98 - Todos los datos del paciente

Ver todos los datos



Nombre: Ver todos los datos

Descripción: Ver todos los datos recogidos por la aplicación del paciente.

Actores: Médico

Precondiciones: El médico debe tener la sesión iniciada.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. Se muestran los datos del paciente. (Ilustración 99)

[El médico hace click sobre el botón “Atrás”]

2A. Se redirige al usuario a la página del paciente. (Ilustración 100)

[El usuario realiza una búsqueda]

2B. Se filtra la tabla.

[El médico hace click sobre una de las líneas de la tabla.]

2C. Se muestra toda la información recogida ese día. (Ilustración 101)

[El médico hace click sobre “Ver gráficos”]

2D. Se redirige a la página con los gráficos personalizados del paciente. (Ilustración 102)

[El usuario filtra las fechas de los gráficos]

2D1. Se muestran los gráficos entre las fechas indicadas. (Ilustración 103)

Postcondiciones: Ninguna.

Interfaz gráfica:

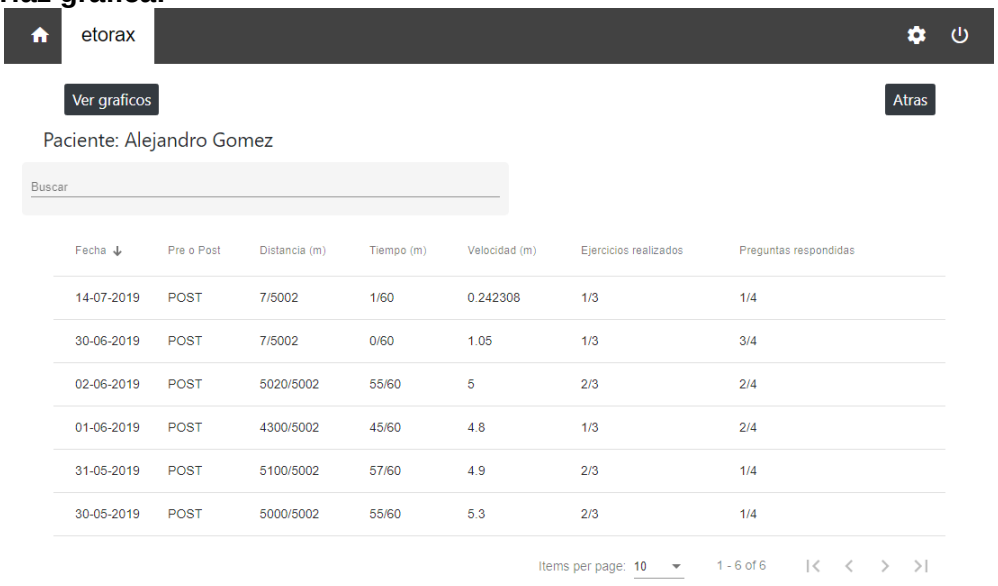


Ilustración 99 - Ver todos los datos

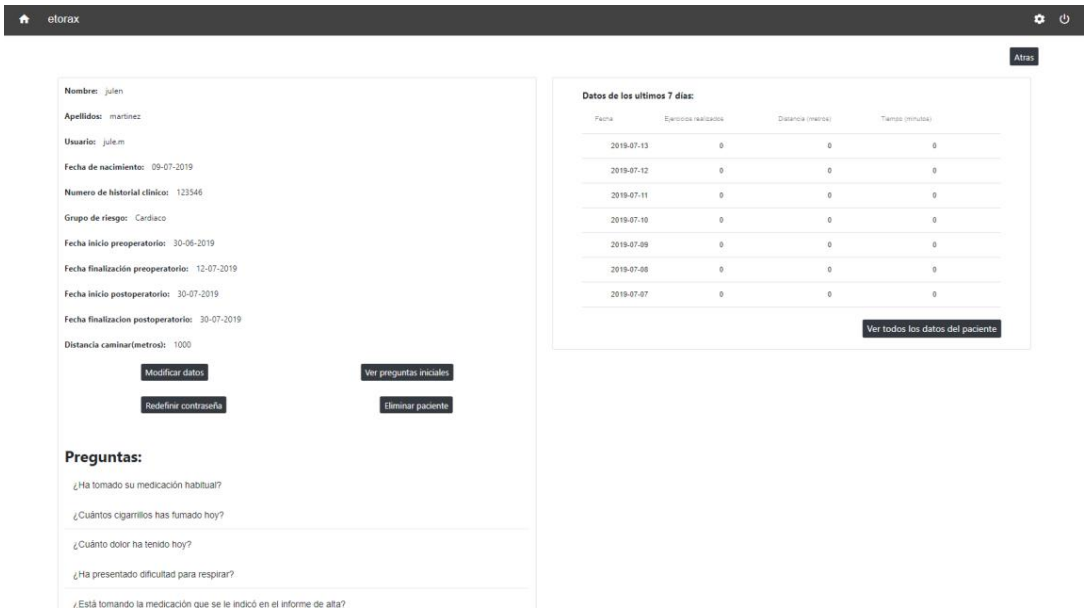


Ilustración 100 - Pagina paciente

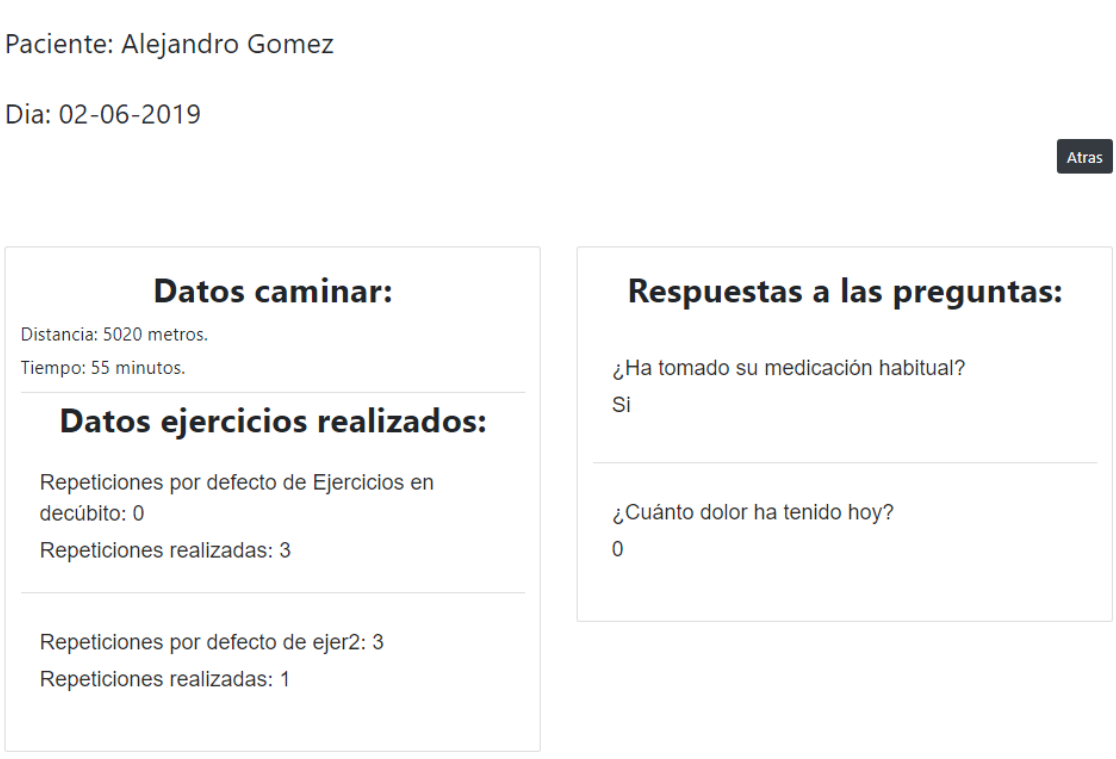


Ilustración 101 - Informacion de un dia

Ilustración 102 - Graficos del paciente

Ilustración 103 - Graficos filtrados

Aplicación móvil

En esta sección del anexo se presentarán los casos de uso extendidos de la aplicación móvil del proyecto.

Iniciar sesión

Nombre: Iniciar sesión

Descripción: El paciente inicia sesión en la aplicación.

Actores: Paciente no identificado

Precondiciones: El paciente no tiene la sesión iniciada

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. Se muestra la página para iniciar sesión. (Ilustración 104)

[El paciente introduce unas credenciales incorrectas]

2A. Se avisa que las credenciales introducidas no son correctas. (Ilustración 105)

[El paciente no tiene conexión a internet]

2B. Se avisa que debe activar la conexión a internet (esto pasará siempre que se intente hacer una petición al back-en y no exista conexión a internet). (Ilustración 106)

[El paciente inicia sesión de forma correcta]

2C. Se muestra la página principal de la aplicación. (Ilustración 107)

Postcondiciones: Si las credenciales son correctas el paciente inicia la sesión.

Interfaz gráfica:

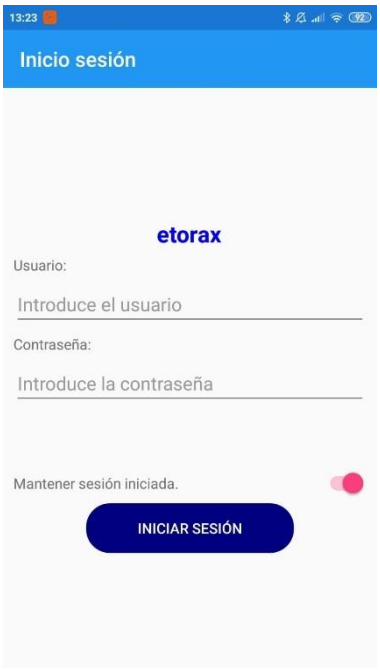


Ilustración 104 - Inicio de sesion APP

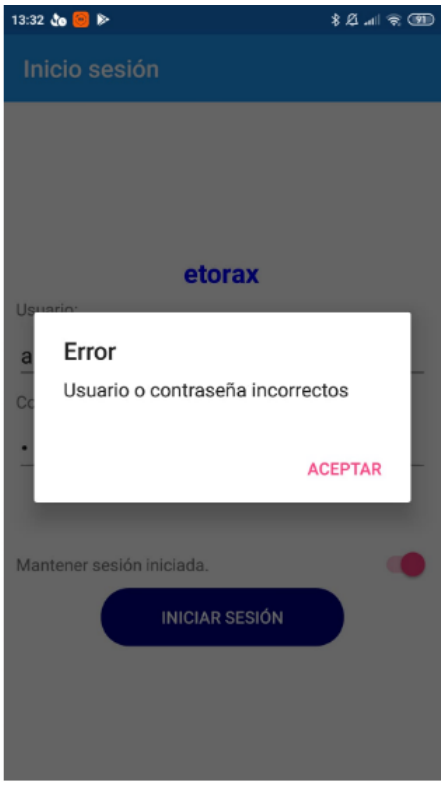


Ilustración 105 - Credenciales incorrectas APP

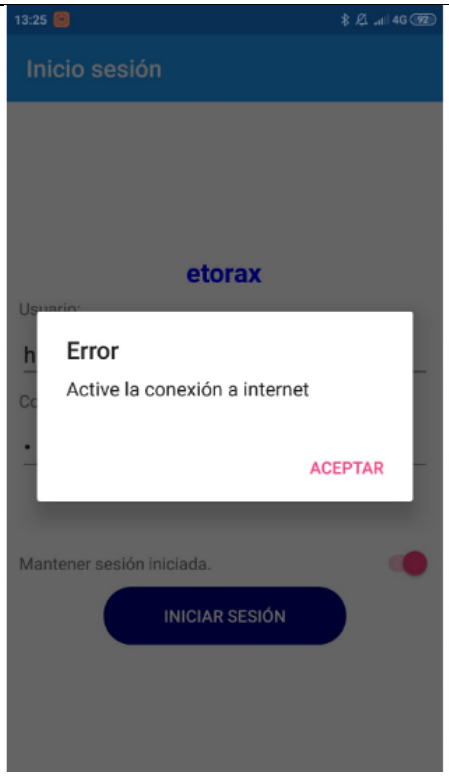
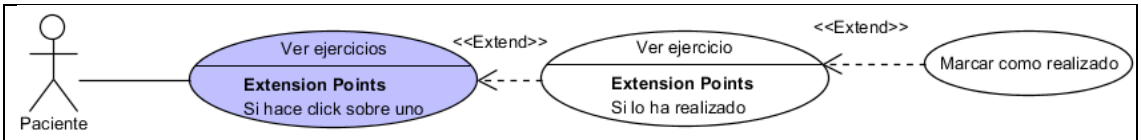


Ilustración 106 - Sin conexion APP

Ilustración 107 - Pagina principal APP

Ver ejercicios



Nombre: Ver ejercicios

Descripción: El usuario podrá ver los ejercicios que tiene asignados.

Actores: Paciente

Precondiciones: El paciente debe haber iniciado sesión.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

[El paciente ha hecho click sobre la opción de ejercicios en la página principal]

1. Se muestra la lista de ejercicios asignados al paciente. (Ilustración 108)

[El paciente selecciona uno de los ejercicios]

2. Se muestra la página del ejercicio seleccionado (Ilustración 109)

[El paciente hace click sobre la opción de ver video]

2A. Se redirige al usuario a la aplicación de YouTube para ver el video.

[El paciente intenta marcar que ha realizado un numero de repeticiones menor que 0]

2B. Se le avisa de que el número de repeticiones debe ser mayor que cero. (Ilustración 110)

[El paciente introduce y envía el número de repeticiones que ha realizado]

2C. El paciente es redirigido a la lista de ejercicios. (Ilustración 108)

Postcondiciones: Ninguna.

Interfaz gráfica:

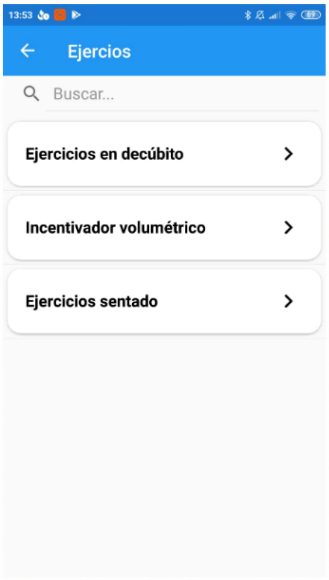


Ilustración 108 - Lista de ejercicios

Ilustración 109 - Ejercicio APP

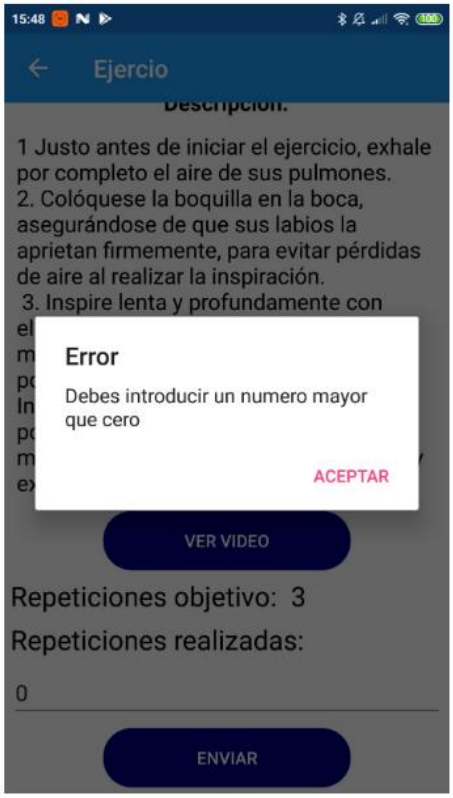
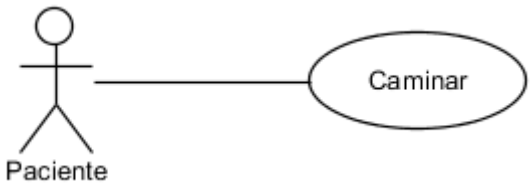


Ilustración 110 - Error ejercicios mayor que cero

Caminar



Nombre: Caminar
Descripción: Control de la distancia y el tiempo que camina el usuario.
Actores: Paciente
Precondiciones: El paciente debe haber iniciado sesión.
Requisitos no funcionales: Ninguno.
Flujo de eventos: 1. Se muestra la página del control de caminar. (Ilustración 111) [El paciente intenta iniciar la caminata sin los permisos necesarios] 2A. Se avisa de que la aplicación no tiene los permisos necesarios. (Ilustración 112) [El paciente inicia la caminata con todos los permisos]

2B. Se empieza a contar la distancia y el tiempo.
Postcondiciones: Ninguna.
Interfaz gráfica:

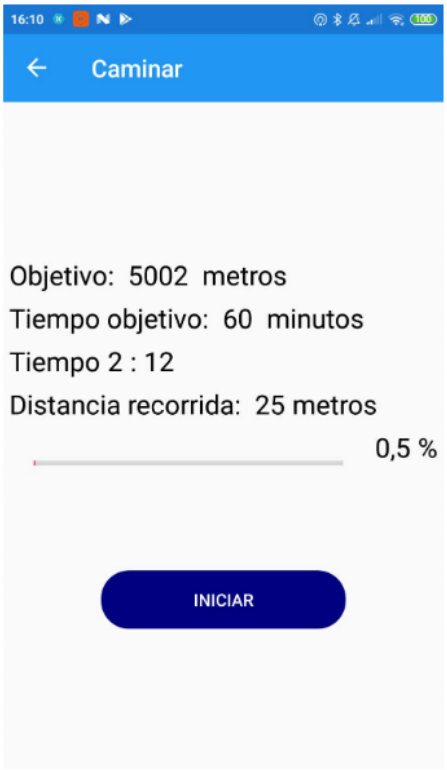
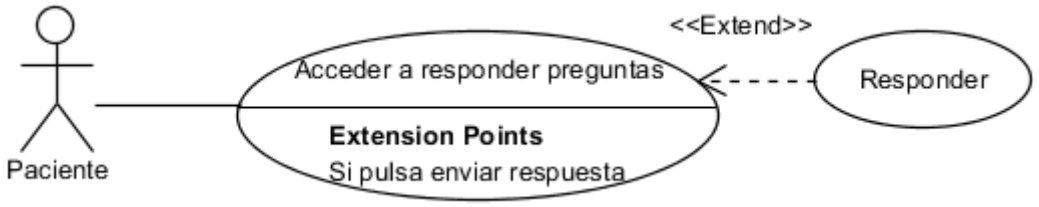


Ilustración 111 - Pagina caminar

Ilustración 112 - Error permisos ubicación

Acceder responder preguntas



Nombre: Responder preguntas

Descripción: El paciente puede responder las preguntas asignadas por el equipo médico.

Actores: Paciente.

Precondiciones: El paciente debe haber iniciado sesión.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. Se muestra la lista de las preguntas asignadas por el equipo médico. (Ilustración 113)

[El paciente selecciona una de las preguntas asignadas]
[Si se trata de una pregunta con respuesta de tipo Si o no]

2A. Se muestra la página de la pregunta con dos botones, uno para responder “Si” y otro para responder “No”. (Ilustración 114)

[Se trata de una respuesta de texto o numérico]

2B. Se muestra la página de la pregunta con un recuadro para introducir la respuesta. (Ilustración 115)

Postcondiciones: Ninguna.

Interfaz gráfica:

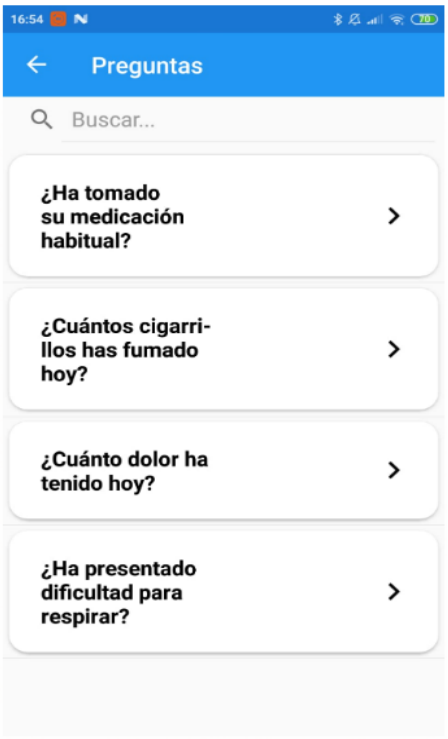


Ilustración 113 - Lista de las preguntas asignadas APP

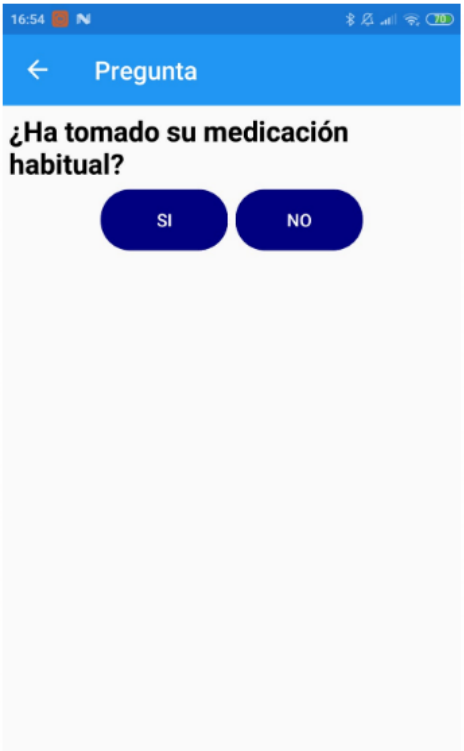


Ilustración 114 - Pregunta de respuesta Si o No

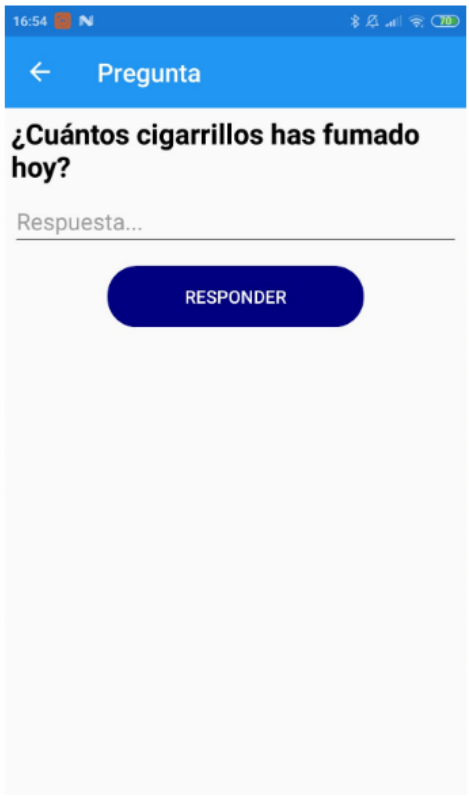
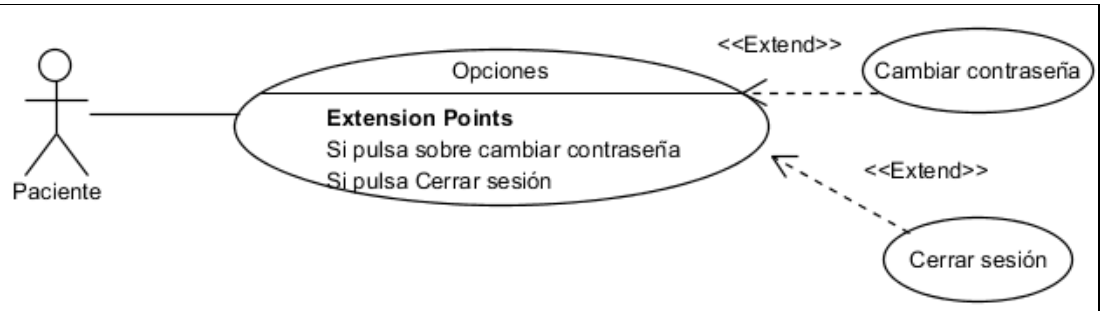


Ilustración 115 - Pregunta numerica

Opciones



Nombre: Opciones
Descripción: El paciente tendrá acceso a sus datos, a cambiar la contraseña y a cerrar sesión.
Actores: Paciente.
Precondiciones: El paciente debe haber iniciado sesión.
Requisitos no funcionales: Ninguno.
Flujo de eventos: 1. Se muestra las opciones del paciente. (Ilustración 116) [El paciente selecciona “Mis datos”]

2A. Se muestran los datos del paciente. (Ilustración 117)

[El paciente pulsa sobre “Cambiar contraseña”]

2A1.El paciente es redirigido a la pantalla de cambio de contraseña. (Ilustración 118)

[El usuario introduce contraseñas que no coinciden]

2A1A. Se muestra un aviso diciendo que las contraseñas no coinciden. (Ilustración 119)

[En el campo contraseña antigua el usuario introduce una contraseña incorrecta]

2A1B. Se muestra un aviso diciendo que la contraseña no es correcta. (Ilustración 120)

[El paciente cambia la contraseña correctamente]

2A1C. Se muestra un aviso diciendo que la contraseña se ha actualizado y se le redirige a la pantalla con sus datos. (Ilustración 121)

[El paciente pulsa sobre “Cerrar sesión”]

2B. Se redirige al paciente a la ventana para iniciar sesión. (Ilustración 104)

Postcondiciones: Ninguna.

Interfaz gráfica:

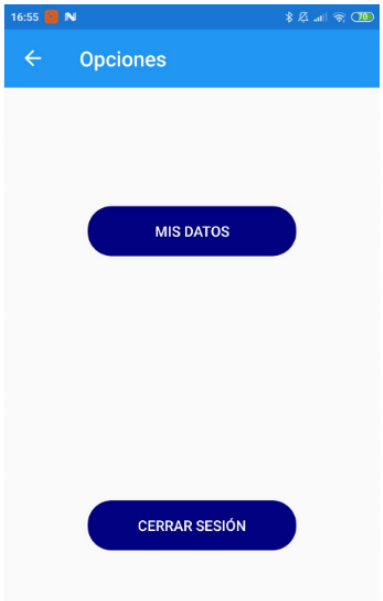


Ilustración 116 - Opciones APP

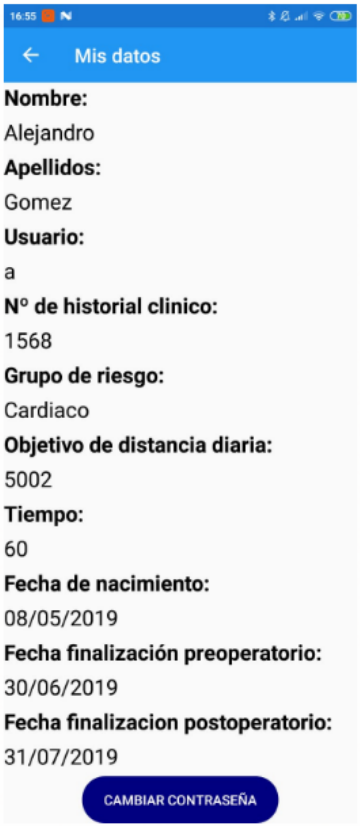


Ilustración 117 - Mis datos

Ilustración 118 - Pantalal de cambio de contraseña

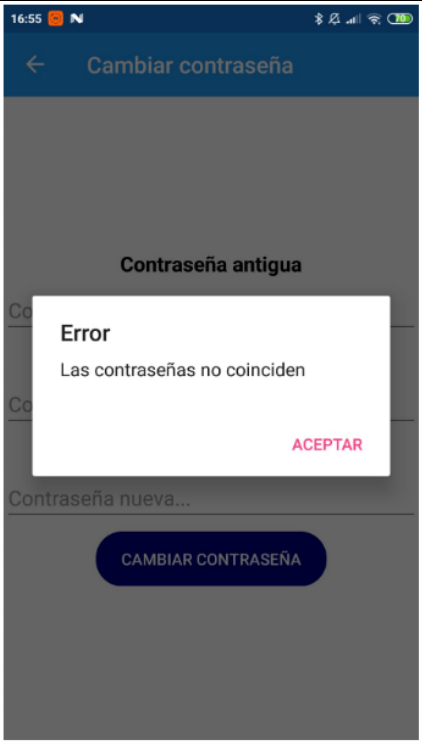


Ilustración 119 - No coinciden

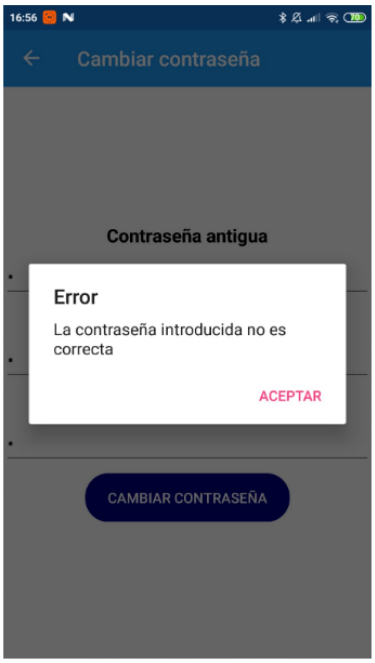


Ilustración 120 - Contraseña incorrecta

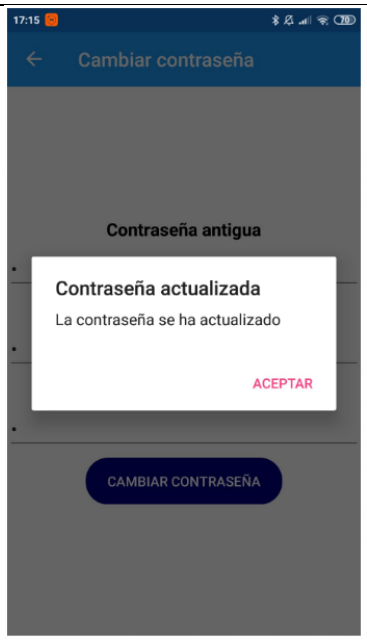


Ilustración 121 - Contraseña actualizada

Ver preguntas frecuentes

Nombre: Ver preguntas frecuentes
Descripción: El paciente podrá ver una serie de preguntas frecuentes que podrán solventar sus dudas.
Actores: Paciente.
Precondiciones: El paciente debe haber iniciado sesión.
Requisitos no funcionales: Ninguno.
Flujo de eventos: 1. Se muestran las preguntas frecuentes. (Ilustración 122) [El paciente selecciona una pregunta 2A. La pregunta se despliega. (Ilustración 123)
Postcondiciones: Ninguna.

Interfaz gráfica:

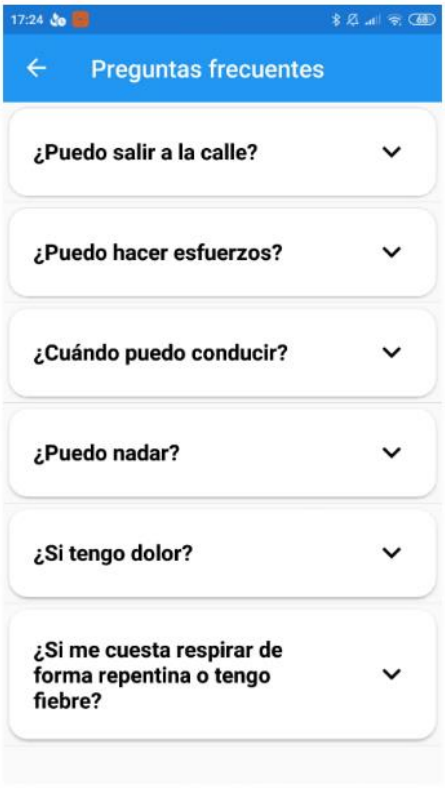


Ilustración 122 - Preguntas frecuentes

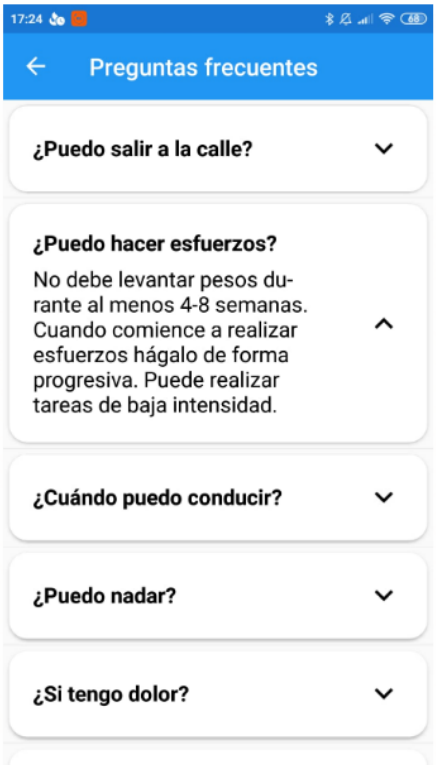
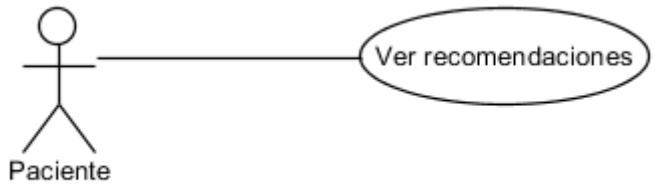


Ilustración 123 - Preguntas frecuentes deplegada

Ver recomendaciones



Nombre: Ver recomendaciones

Descripción: El paciente podrá ver lar recomendaciones del equipo médico.

Actores: Paciente.

Precondiciones: El paciente debe haber iniciado sesión.

Requisitos no funcionales: Ninguno.

Flujo de eventos:

1. Se muestran las recomendaciones. (Ilustración 124)

[El paciente selecciona una recomendación

2. Es redirigido a la página de la recomendación. (Ilustración 125)

Postcondiciones: Ninguna

Interfaz gráfica:

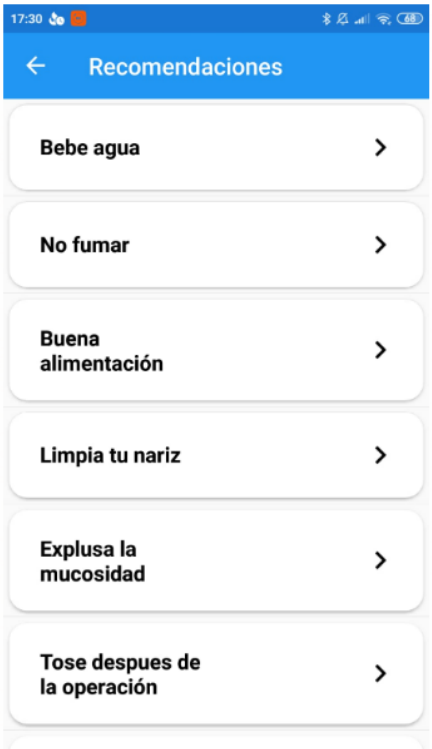


Ilustración 124 - Recomendaciones

Ilustración 125 - Recoemntdacion

Anexo II: Diagramas de secuencia

En este anexo se detallarán cada uno de los diagramas de secuencia de los casos de uso explicados en el apartado anterior (Anexo I: Casos de uso extendidos). Cabe mencionar que como el proyecto se encuentra dividido en dos partes: back-end y front-end el diagrama de secuencia también estará dividido en dos partes. En los diagramas de secuencia se podrán diferenciar ambas partes por el color. La parte del front-end aparecerá en azul y la parte del back-end en verde.

Es importante mencionar que en cada una de las peticiones que se realizan al back-end este comprobará que le hayan llegado todos los parámetros. Con el fin de que los diagramas de secuencia sean lo más simples posibles, para que así sean los más fácil de entender posibles, esta parte se ha omitido de los diagramas. A continuación, se muestra un ejemplo de cómo sería la respuesta que se obtendría en caso de faltar un parámetro. Como se puede ver se recibe un error 400 en formato JSON.


```
{
    "status": 400,
    "code": 40000,
    "msg": "Falta un parametro en la petición o la peticion no es correcta",
    "body": ""
}
```

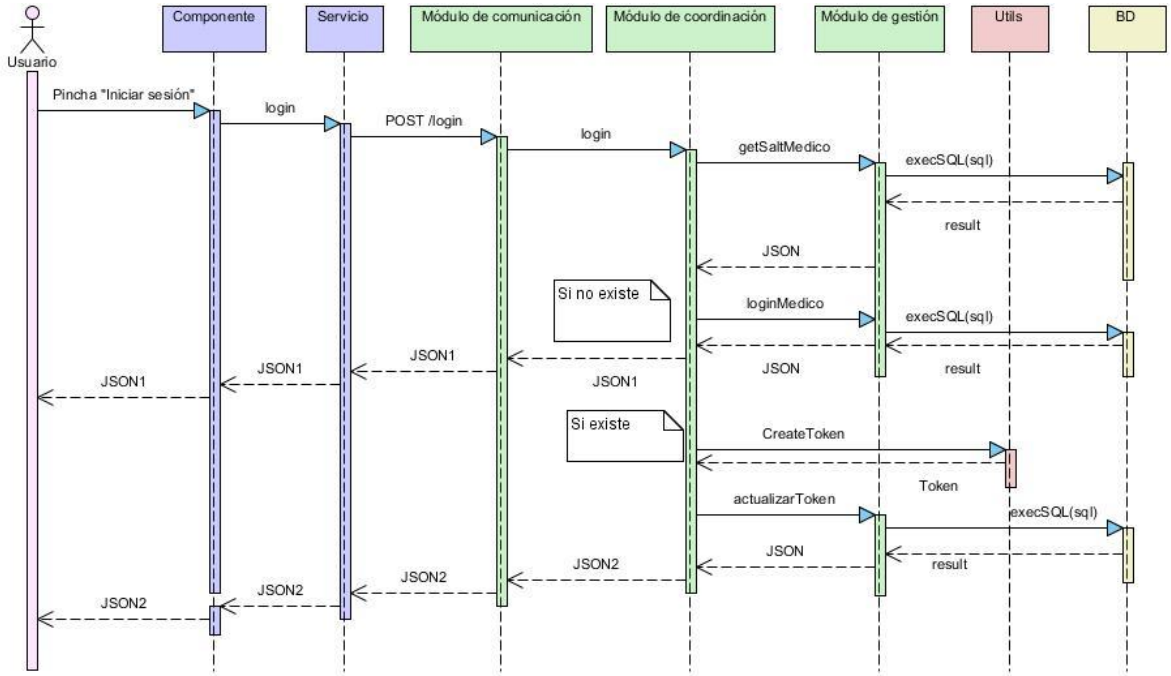

Ilustración 126 - Error falta un parametro

Página web

En este apartado se mostrarán los diagramas de secuencia para los casos de uso de la aplicación web.

Iniciar sesión

Este diagrama de secuencia corresponde con el caso de uso extendido de iniciar sesión.



Primero se hace una petición SQL a la base de datos para obtener el SALT del usuario introducido por el médico. Con este salt se formará la contraseña de la base de datos. La petición SQL es la siguiente:

```
SELECT salt FROM medico WHERE user = ?
```

Una vez se ha formado la contraseña con el SALT se comprueba que los datos introducidos por el usuario sean correctos. Para ello se usa la siguiente SQL:

```
SELECT token FROM medico WHERE user = ? and pass = ?
```

Por último, la SQL que se utiliza para actualizar el token es la siguiente:

```
UPDATE medico SET token= ? WHERE user = ?
```

Cuando el usuario ha introducido unas credenciales que no son correctas el cliente recibe la respuesta JSON1 que aparece en el diagrama. En la siguiente ilustración se puede ver el formato de la respuesta.

```
{
    "status": 201,
    "code": 20110,
    "msg": "Datos de inicio de sesión incorrectos",
    "body": ""
}
```

Ilustración 127 - Respuesta inicio de sesion incorrecto

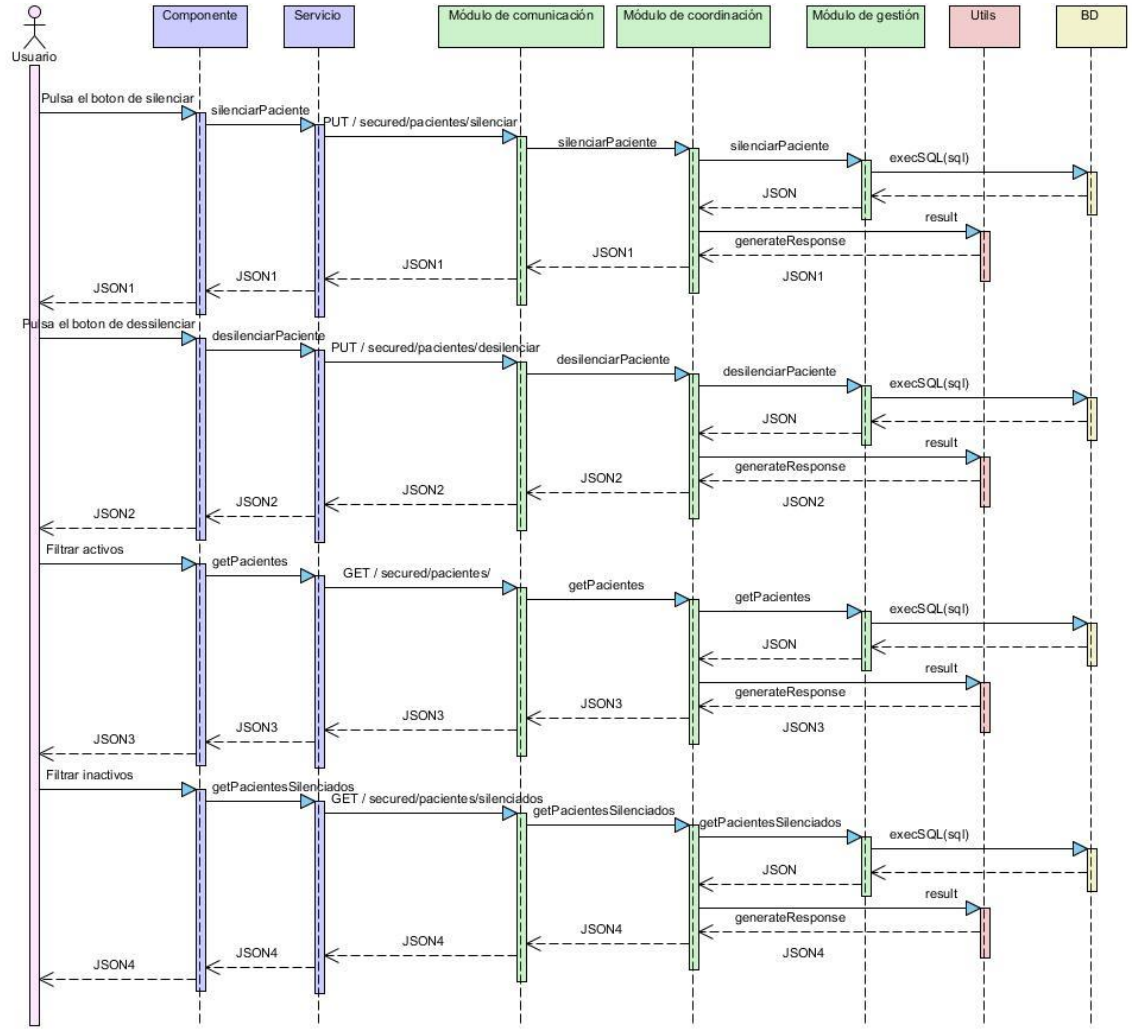
En cambio, si el usuario ha iniciado la sesión con las credenciales correctas se recibe el siguiente mensaje en el que se encuentra el token del usuario.


```
{
    "status": 200,
    "code": 20010,
    "msg": "Sesión iniciada correctamente.",
    "body":
        "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbiIsImlhdCI6MTU2MjA4Njk3OSwiZXhwIjoiMjAxOS0wNy0xNyJ9.XV-3d
        OGFHb4tlekJZe1g4Ba63L1w2lwHMeociUdoKx8"
}
```


Ilustración 128 - Respuesta inicio sesion correcto

Gestionar pacientes

Este diagrama de secuencia corresponde con el caso de uso extendido de agregar gestionar pacientes.



La sentencia SQL para “silenciar” es la siguiente:

UPDATE tratamiento SET activo =? 'WHERE id=?

La sentencia SQL para “desilenciar” es la siguiente:

```
UPDATE tratamiento SET activo =? WHERE id=?
```

La sentencia SQL para “getPacientes” es la siguiente:

```
SELECT t.id,
p.nombre, p.apellidos, p.numHist,
t.ultimaMod, t.fechaFPre, g.nombre AS grupo
FROM grupoPaciente gp
JOIN paciente p ON gp.numHistPac = p.numHist
JOIN tratamiento t ON p.numHist = t.numHist
JOIN grupoRiesgo g ON gp.nombreGrupo = g.nombre
where t.activo=? AND  (gp.fechaFin is null)
```

La sentencia SQL para “getPacientesSilenciados” es la siguiente:

```
SELECT t.id,
p.nombre, p.apellidos, p.numHist,
t.ultimaMod,t.fechaFPre, g.nombre AS grupo
FROM grupoPaciente gp
JOIN paciente p ON gp.numHistPac = p.numHist
JOIN tratamiento t ON p.numHist = t.numHist
JOIN grupoRiesgo g ON gp.nombreGrupo = g.nombre
where t.activo=? AND  (gp.fechaFin is null)
```

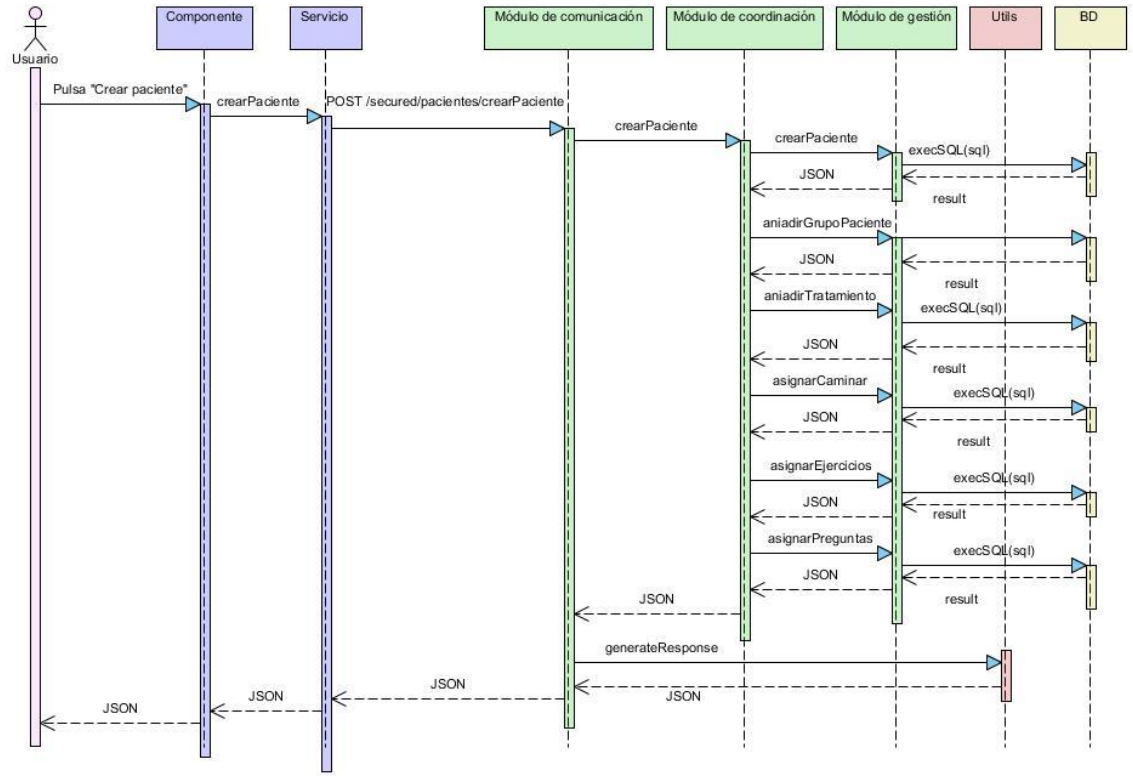
Las respuestas JSON3 y JSON4 son muy parecidas, a continuación, se muestra un ejemplo de la respuesta JSON3.


```
{
    "status": 200,
    "code": 20020,
    "msg": "Todos los pacientes no silenciados obtenidos de forma correcta",
    "body": [
        {
            "id": 441,
            "nombre": "Alejandro",
            "apellidos": "Gomez",
            "numHist": 1568,
            "ultimaMod": "2019-05-20T07:10:39.000Z",
            "fechaFPre": "2019-06-30T00:00:00.000Z",
            "grupo": "Cardiaco"
        }
    ]
}
```


Ilustración 129 - Respuesta get pacientes no silenciados

Agregar paciente

Este diagrama de secuencia corresponde con el caso de uso extendido de agregar un paciente a la aplicación.



La sentencia SQL para “crearPaciente” es la siguiente:

```
INSERT INTO paciente SET ?
```

La sentencia SQL para “aniadirGrupoPaciente” es la siguiente:

```
INSERT INTO grupopaciente SET ?
```

La sentencia SQL para “aniadirTratamiento” es la siguiente:

```
INSERT INTO tratamiento SET ?
```

La sentencia SQL para “asignarCaminar” es la siguiente:

```
INSERT INTO tratamientocaminar SET ?
```

La sentencia SQL para “asignarEjercicios” es la siguiente:

INSERT INTO tratamientoejercicio (idTrat, idEj, fechaIni,repeticiones) VALUES ?

La sentencia SQL para “asignarPreguntas” es la siguiente:

INSERT INTO tratamientopregunta (idTrat, idPre, fechaIni) VALUES ?

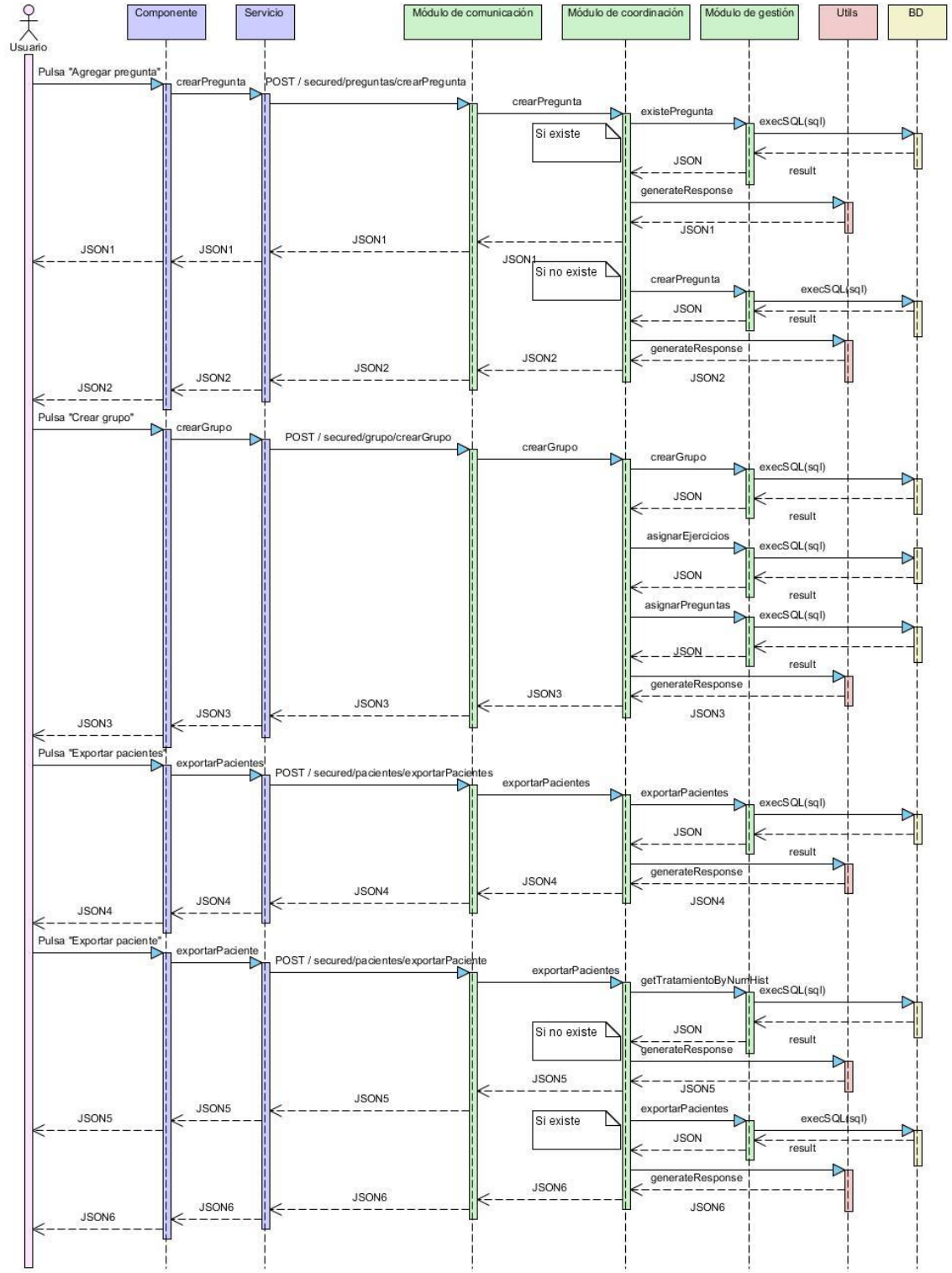
La respuesta JSON que se recibe del back-end es la siguiente:

{status: 200, code: 20025, msg: "El paciente ha sido creado de forma correcta.", body: 461}

Ilustración 130 - Respuesta agregar paciente

Configuración

Este diagrama de secuencia corresponde con el caso de uso extendido de "configuración".



La sentencia SQL para “existePregunta” es la siguiente:

```
Select id from pregunta where texto=?
```

La sentencia SQL para “crearPregunta” es la siguiente:

```
INSERT INTO pregunta SET ?
```

La sentencia SQL para “crearGrupo” es la siguiente:

```
INSERT INTO gruporiesgo SET ?
```

La sentencia SQL para “asignarEjercicios” es la siguiente:

```
INSERT INTO grupotieneejercicio (idEj, nombreGrupo) VALUES ?
```

La sentencia SQL para “asignarPreguntas” es la siguiente:

```
INSERT INTO grupotienepregunta (idPre, nombreGrupo) VALUES ?
```

La sentencia SQL para “exportarPacientes” es la siguiente:

```
SELECT t.id,
p.nombre, p.apellidos, p.numHist,
t.ultimaMod,t.fechaFPre, g.nombre AS grupo
FROM grupoPaciente gp
JOIN paciente p ON gp.numHistPac = p.numHist  '
JOIN tratamiento t ON p.numHist = t.numHist
JOIN grupoRiesgo g ON gp.nombreGrupo = g.nombre
where t.activo=? AND  (gp.fechaFin is null)
```

La sentencia SQL para “getTratamientoByNumHist” es la siguiente:

```
SELECT id from  tratamiento WHERE numHist=?
```

La sentencia SQL para “exportarPacietne” es la siguiente:

```
SELECT distancia,velocidad,fecha,ejercicios,respuestas,tiempo  FROM
(SELECT trc.velocidad,trc.distancia,trej.fecha as fec,trc.tiempo, COUNT(*) AS
ejercicios
FROM tratamientoejerciciorealizado trej
JOIN tratamientocaminarrealizado trc on trej.fecha=trc.fecha
WHERE trc.idTrat=trej.idTrat and trej.repeticiones!=0  GROUP BY trej.fecha)
t1
INNER JOIN
(SELECT fecha, COUNT(*) AS respuestas
FROM tratamientopreguntarespondida
WHERE idTrat=? GROUP BY fecha) t2
ON t1.fec= t2.fecha
```

En cuanto a las respuestas JSON la respuesta JSON1 y la respuesta JSON5 son respuestas de error con el código del error y el mensaje. Las respuestas JSON2 y JSON3 son respuestas de OK con el código de Ok y el mensaje.

A continuación, se puede ver un ejemplo de la respuesta JSON4:


```
{
    "status": 200,
    "code": 20021,
    "msg": "Todos los pacientes silenciados obtenidos de forma correcta",
    "body": [
        {
            "IDTratamiento": 301,
            "nombre": "Maria",
            "apellidos": "Fernandez",
            "numHist": 789,
            "usuario": "m",
            "fechaNac": "2019-04-09T00:00:00.000Z",
            "ultimaMod": "2019-04-08T07:42:11.000Z",
            "fechaFPre": "2019-04-11T00:00:00.000Z",
            "activo": "no",
            "fechaIPre": "2019-04-10T00:00:00.000Z",
            "fechaFPost": "2019-04-13T00:00:00.000Z",
            "fechaIPost": "2019-04-12T00:00:00.000Z",
            "grupo": "Cardiaco",
            "fechaIniGR": "2019-04-08T00:00:00.000Z",
            "fechaFinGR": "2019-06-03T00:00:00.000Z"
        },
```


Ilustración 131 - Respuesta exportar pacientes

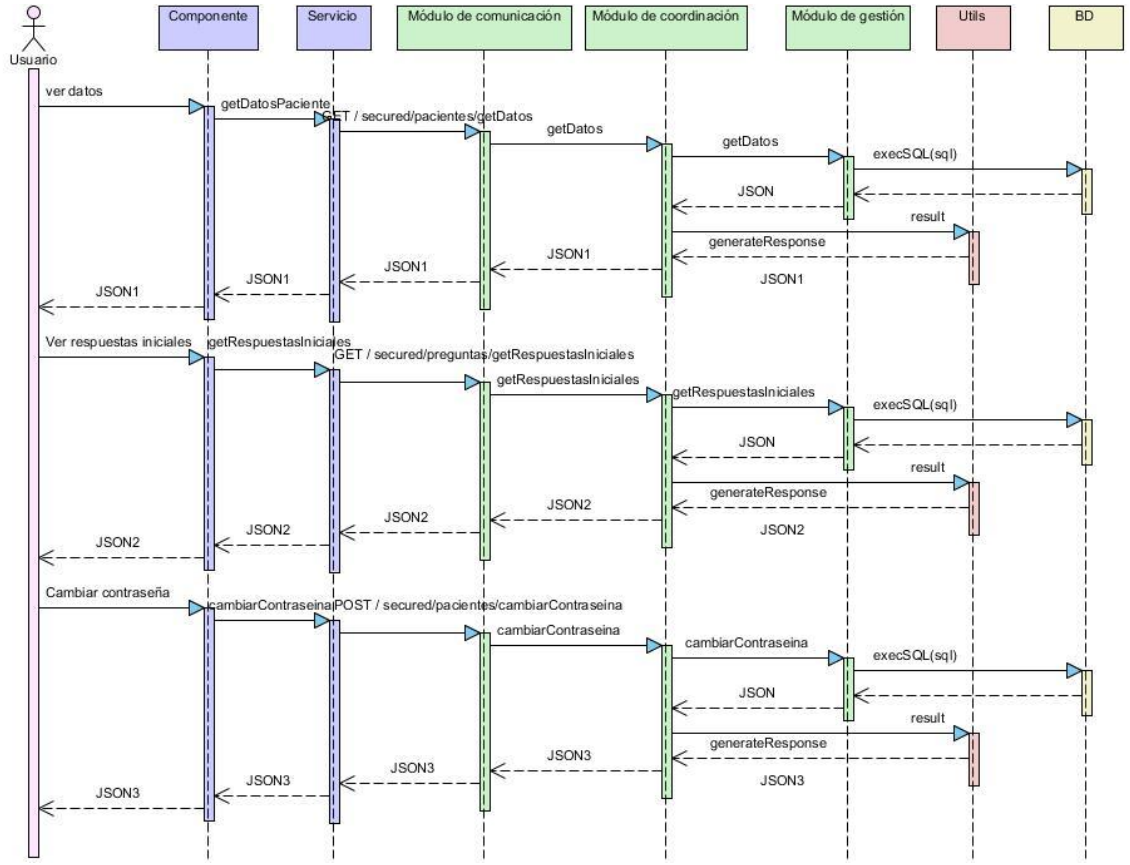
A continuación, se puede ver un ejemplo de la respuesta JSON6:

```
{
    "status": 200,
    "code": 200216,
    "msg": "Paciente exportado de forma correcta",
    "body": [
        {
            "distancia": "5000",
            "velocidad": 5.3,
            "fecha": "2019-05-30T00:00:00.000Z",
            "ejercicios": 2,
            "respuestas": 1,
            "tiempo": "55"
        },
        {
            "distancia": "5100",
            "velocidad": 4.9,
            "fecha": "2019-05-31T00:00:00.000Z",
            "ejercicios": 2,
            "respuestas": 1,
            "tiempo": "57"
        },
```


Ilustración 132 - Respuesta exportar paciente

Ver datos paciente

Este diagrama de secuencia corresponde con el caso de uso extendido de “Ver datos paciente”.



La sentencia SQL para “getDatos” es la siguiente:

```
SELECT t.fechaIPre, t.fechaFPre,t.fechaIPost,t.fechaFPost,
p.nombre, p.apellidos, p.numHist, p.usuario,p.fechaNac,cam.distancia,
g.nombre AS grupo
FROM grupoPaciente gp
JOIN paciente p ON gp.numHistPac = p.numHist
JOIN tratamiento t ON p.numHist = t.numHist
JOIN grupoRiesgo g ON gp.nombreGrupo = g.nombre
JOIN tratamientocaminar trc ON t.id = trc.idTrat
JOIN caminar cam ON cam.id = trc.idCaminar
where t.id=? AND (trc.fechaFin is null)
```

La sentencia SQL para “getRespuestasIniciales” es la siguiente:

```
SELECT pre.texto, trpreres.respuesta from pregunta pre
JOIN tratamientopreguntarespondida trpreres ON pre.id = trpreres.idPre '+
where trpreres.idTrat=? AND pre.inicial=?
```

La sentencia SQL para “cambiarContraseña” es la siguiente:

```
UPDATE paciente SET contraseña =? WHERE numHist=?
```

A continuación, se puede ver un ejemplo de la respuesta JSON1:


```
{
    "status": 200,
    "code": 20026,
    "msg": "Se han obtenido los datos del paciente.",
    "body": {
        "fechaIPre": "2019-05-08T00:00:00.000Z",
        "fechaFPre": "2019-06-30T00:00:00.000Z",
        "fechaIPost": "2019-06-30T00:00:00.000Z",
        "fechaFPost": "2019-07-31T00:00:00.000Z",
        "nombre": "Alejandro",
        "apellidos": "Gomez",
        "numHist": 1568,
        "usuario": "a",
        "fechaNac": "2019-05-08T00:00:00.000Z",
        "distancia": 5002,
        "grupo": "Cardiaco"
    }
}
```


Ilustración 133 - Respuesta getDatos

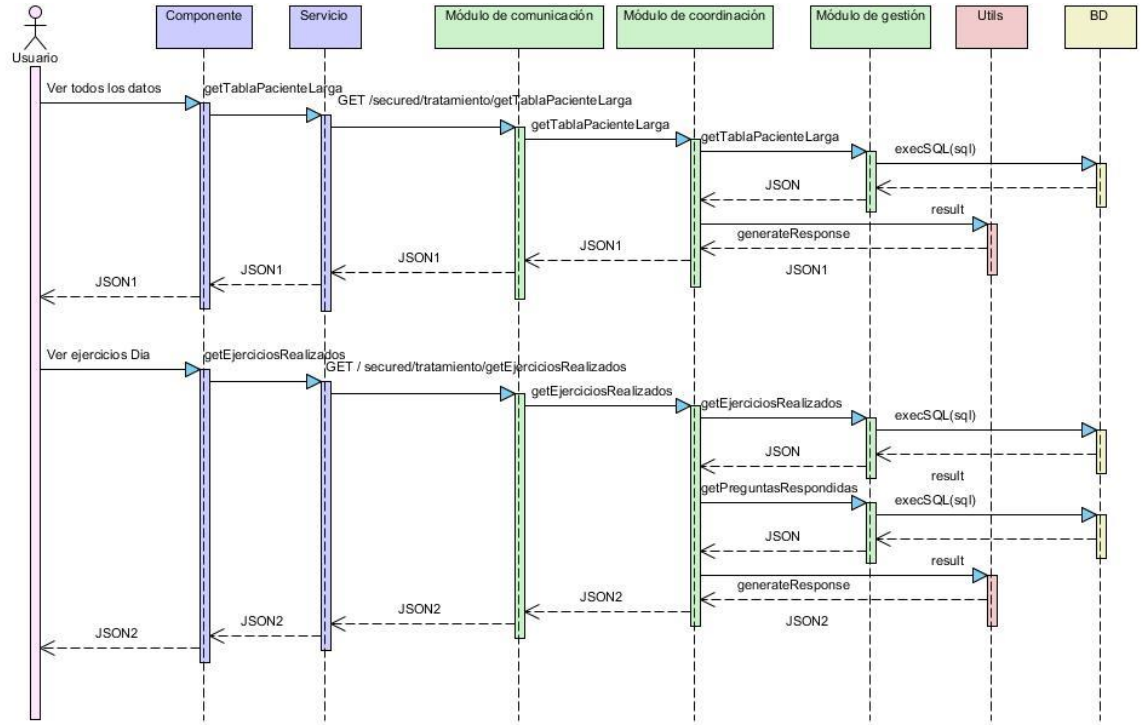
A continuación, se puede ver un ejemplo de la respuesta JSON2:

```
{
    "status": 200,
    "code": 20064,
    "msg": "Se han obtenido las respuestas iniciales.",
    "body": [
        {
            "texto": "¿Tomas inhaladores?",
            "respuesta": "si"
        },
        {
            "texto": "¿Tomas inhaladores?",
            "respuesta": "si"
        },
```


Ilustración 134 - Respuestas iniciales

Ver todos los datos

Este diagrama de secuencia corresponde con el caso de uso extendido de “Ver todos los datos”.



La sentencia SQL para “getTablaPacienteLarga” es la siguiente:

```
SELECT distancia,velocidad,fecha,ejercicios,respuestas,tiempo  FROM
(SELECT trc.velocidad,trc.distancia,trej.fecha as fec,trc.tiempo, COUNT(*) AS ejercicios
FROM tratamientoejerciciorealizado trej
JOIN tratamientocaminarrealizado trc on trej.fecha=trc.fecha
WHERE trc.idTrat=trej.idTrat and trej.repeticiones!=0  GROUP BY trej.fecha) t1
INNER JOIN
(SELECT fecha, COUNT(*) AS respuestas
FROM tratamientopreguntarespondida
WHERE idTrat=? GROUP BY fecha) t2
ON t1.fec= t2.fecha
```

La sentencia SQL para “getEjerciciosRealizados” es la siguiente:

```
select trejr.repeticiones as repRealizadas, ej.nombre,trej.repeticiones as repAsignadas
from tratamientoejerciciorealizado trejr
join  ejercicio ej on trejr.idEj=ej.id
join  tratamientoejercicio trej on trej.idEj=trejr.idEj
where trejr.fecha=? and trejr.idTrat=? and trej.idTrat=? and (trej.fechaFin is null)
```

La sentencia SQL para “getPreguntasRespondidas” es la siguiente:

```
select p.texto, trpr.respuesta from tratamientopreguntarespondida trpr
join pregunta p on trpr.idPre=p.id
where trpr.idTrat=? and trpr.fecha=?
```

A continuación, se puede ver un ejemplo de la respuesta JSON1:


```
{
    "status": 200,
    "code": 20072,
    "msg": "Tabla paciente larga obtenida.",
    "body": [
        {
            "distancia": "5000",
            "velocidad": 5.3,
            "fecha": "2019-05-30T00:00:00.000Z",
            "ejercicios": 2,
            "respuestas": 1,
            "tiempo": "55"
        },
```


Ilustración 135 - Respuesta get tabla larga paciente

A continuación, se puede ver un ejemplo de la respuesta JSON2:

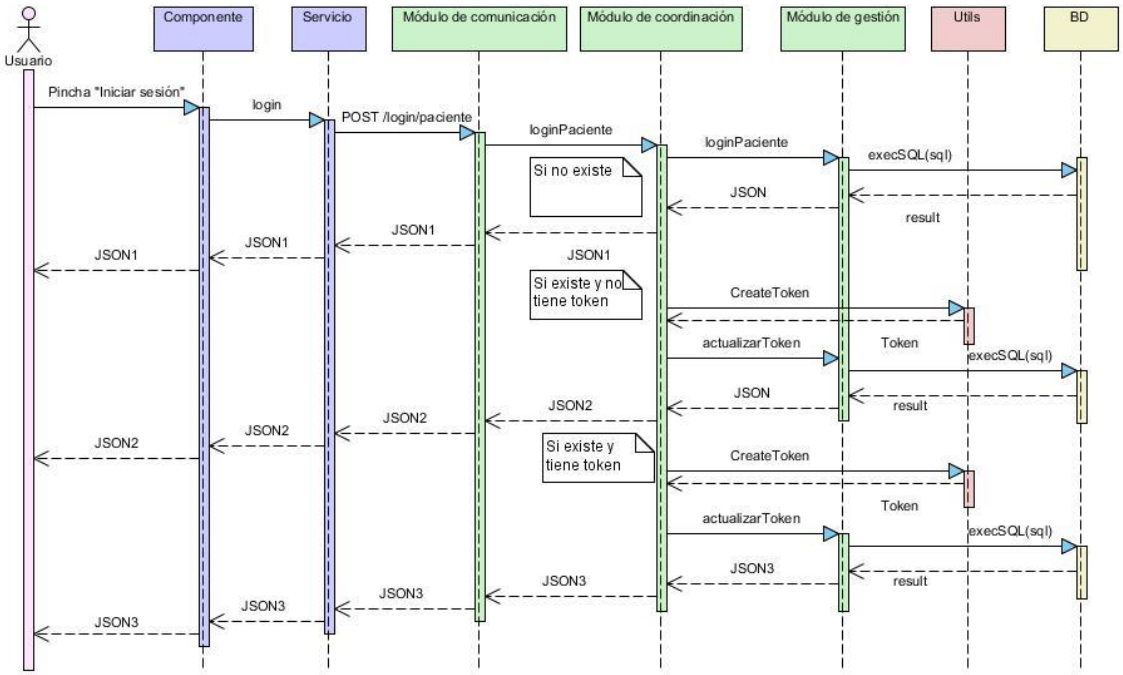
```
{
    "status": 200,
    "code": 20074,
    "msg": "Nombre del tratamiento obtenido.",
    "body": "[[],[]]"
}
```


Ilustración 136 - Respuesta get ejercicios dia

Aplicación móvil

Iniciar sesión

Este diagrama de secuencia corresponde con el caso de uso extendido de "Iniciar sesión" de la aplicación móvil.



La sentencia SQL para "LoginPaciente" es la siguiente:

```
SELECT token FROM paciente WHERE usuario = ? and contraseña = ?
```

La sentencia SQL para "actualizarToken" es la siguiente:

```
UPDATE paciente SET token= ? WHERE usuario = ?
```

A continuación, se puede ver un ejemplo de la respuesta JSON2:

```
{
    "status": 200,
    "code": 20012,
    "msg": "Sesión iniciada y token creado.",
    "body":
        "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJlIiwiaWF0IjoxNTYzNDQ4NzQwLCJleHAiOjI0Mjc0NDg3NDB9.MJfcjqwmQrYEF
        mL2YSXnwFuSeqzZWTEP7cIGl5L5KQE"
}
```


Ilustración 137 - Respuesta sesion iniciada por primera vez

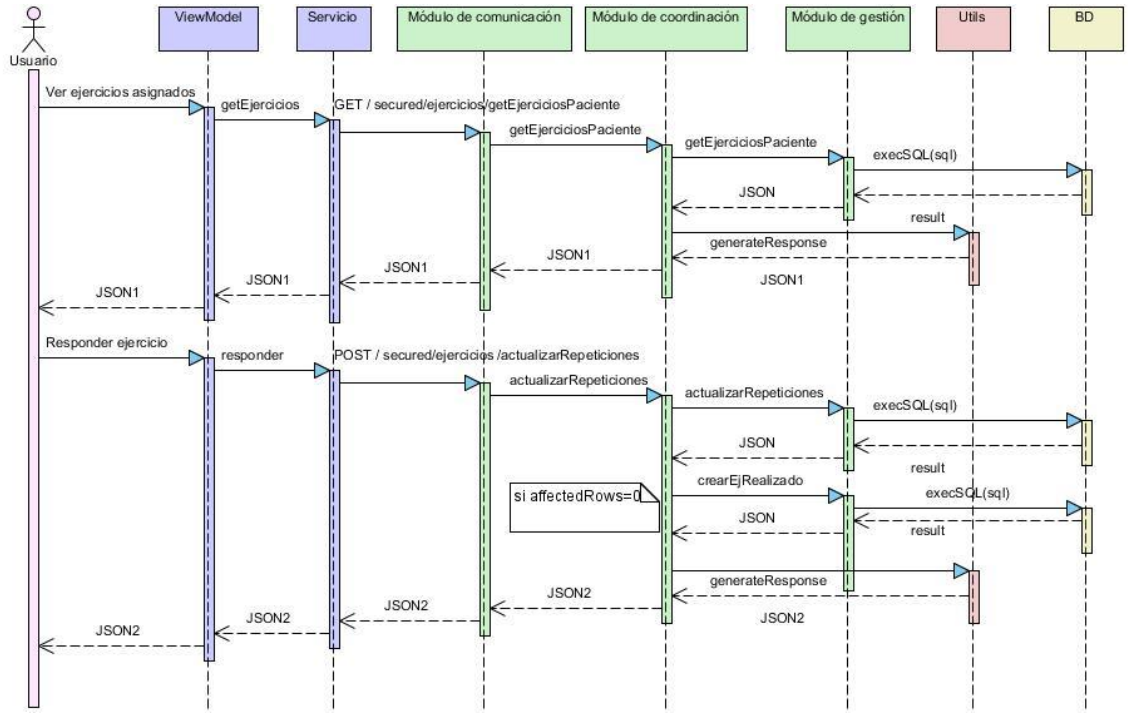
A continuación, se puede ver un ejemplo de la respuesta JSON3:


```
{
    "status": 200,
    "code": 20010,
    "msg": "Sesión iniciada correctamente.",
    "body":
        "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJlIiwiaWF0IjoxNTYzNDQ4NzQwLCJleHAiOjI0Mjc0NDg3NDB9.MJfcjqwmQrYEF
        mL2YSXnwFuSeqzZWTEP7cIGl5L5KQE"
}
```


Ilustración 138 - Respuesta sesion iniciada

Ver ejercicios

Este diagrama de secuencia corresponde con el caso de uso extendido de “ver ejercicios” de la aplicación móvil.



La sentencia SQL para “getEjerciciosPaciente” es la siguiente:

```
SELECT ej.nombre, ej.id, ej.descripcion, ej.link,trej.repeticiones from
ejercicio ej
JOIN tratamientoejercicio trej ON ej.id = trej.idEj
where trej.idTrat=? AND (trej.fechaFin is null)
```

La sentencia SQL para “actualizarRepeticiones” es la siguiente:

```
UPDATE tratamientoejerciciorealizado SET repeticiones=?
where idTrat=? and fecha=? and idEj=?
```

La sentencia SQL para “crearEjRealizado” es la siguiente:

```
INSERT INTO tratamientoejerciciorealizado (idTrat, idEj,fecha,repeticiones)
VALUES (?,?,?,?)
```

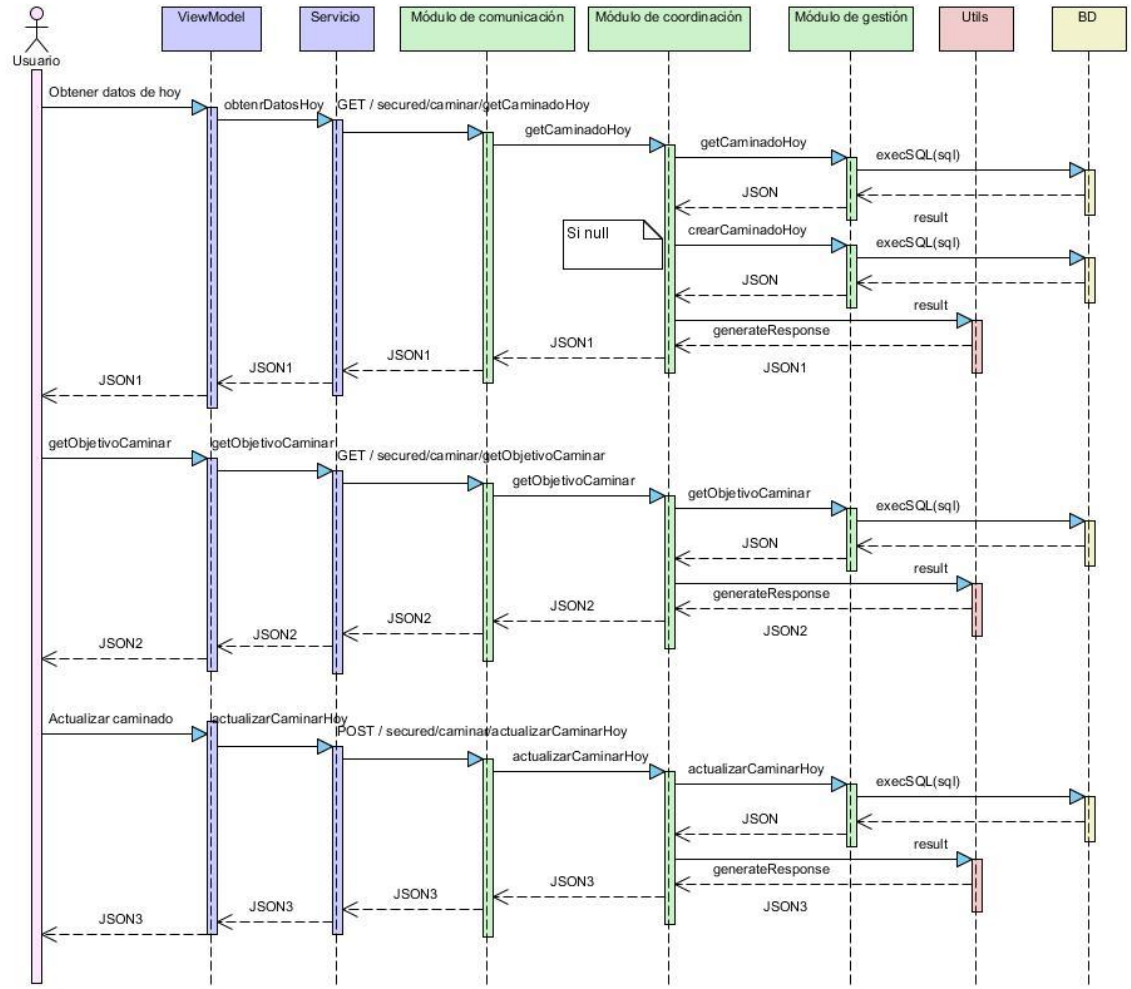
A continuación, se puede ver un ejemplo de la respuesta JSON1:


```
{
    "status": 200,
    "code": 20045,
    "msg": "Se han obtenido los ejercicios del tratamiento.",
    "body": [
        {
            "nombre": "Ejercicios en decúbito",
            "id": 1,
            "descripcion": "1 Separamos la pierna tomando aire por la nariz lentamente, posteriormente la juntamos
                echando el aire por la boca con los labios fruncidos. Alternamos las piernas. \\n2 Flexionamos la
                pierna tomando aire por la nariz lentamente, posteriormente la estiramos echando el aire por la boca
                con los labios fruncidos. Alternamos las piernas. \\n3 Subimos y bajamos los pies alternativamente.\\n4
                Levante el brazo por encima de la cabeza, tomando aire por la nariz y lo barjamos soltando el aire por
                la boca. Alternamos los brazos.\\n5 Separamos el brazo sin despegarlo de la cama, hacia la cabez sin
                doblar el codo, tomando aire por la nariz. Regresamos a la posición inicial soltando el aire por la
                boca. Alternamos los brazos.",
            "link": "www.youtube.com/watch?v=Bey4XXJAqS8",
            "repeticiones": 0
        },
```


Ilustración 139 - Respuesta a getEjerciciosPaciente

Caminar

Este diagrama de secuencia corresponde con el caso de uso extendido de “caminar ” de la aplicación móvil.



La sentencia SQL para “getCaminadoHoy” es la siguiente:

```
SELECT distancia, tiempo from tratamientocaminarrealizado where idTrat=? and fecha=?
```

La sentencia SQL para “crearCaminadoHoy” es la siguiente:

```
INSERT INTO tratamientocaminarrealizado (idTrat, idCaminar,fecha,velocidad,distancia,tiempo) VALUES (?,?,?,?,?,?)
```

La sentencia SQL para “getObjetivoCaminar” es la siguiente:

```
SELECT caminar.distancia, caminar.tiempo from caminar
JOIN tratamientocaminar trca ON caminar.id = trca.idCaminar
where trca.idTrat=?  And (fechaFin is null)
```

La sentencia SQL para “actualizarCaminarHoy” es la siguiente:

```
UPDATE tratamientocaminarrealizado SET velocidad=? ,distancia=? , tiempo=?
where idTrat=? and fecha=?
```

A continuación, se puede ver un ejemplo de la respuesta JSON1:


```
{
    "status": 201,
    "code": 20123,
    "msg": "Se ha obtenido la distancia caminada hoy",
    "body": {
        "distancia": "0",
        "tiempo": "0"
    }
}
```


Ilustración 140 - Respuesta caminado hoy

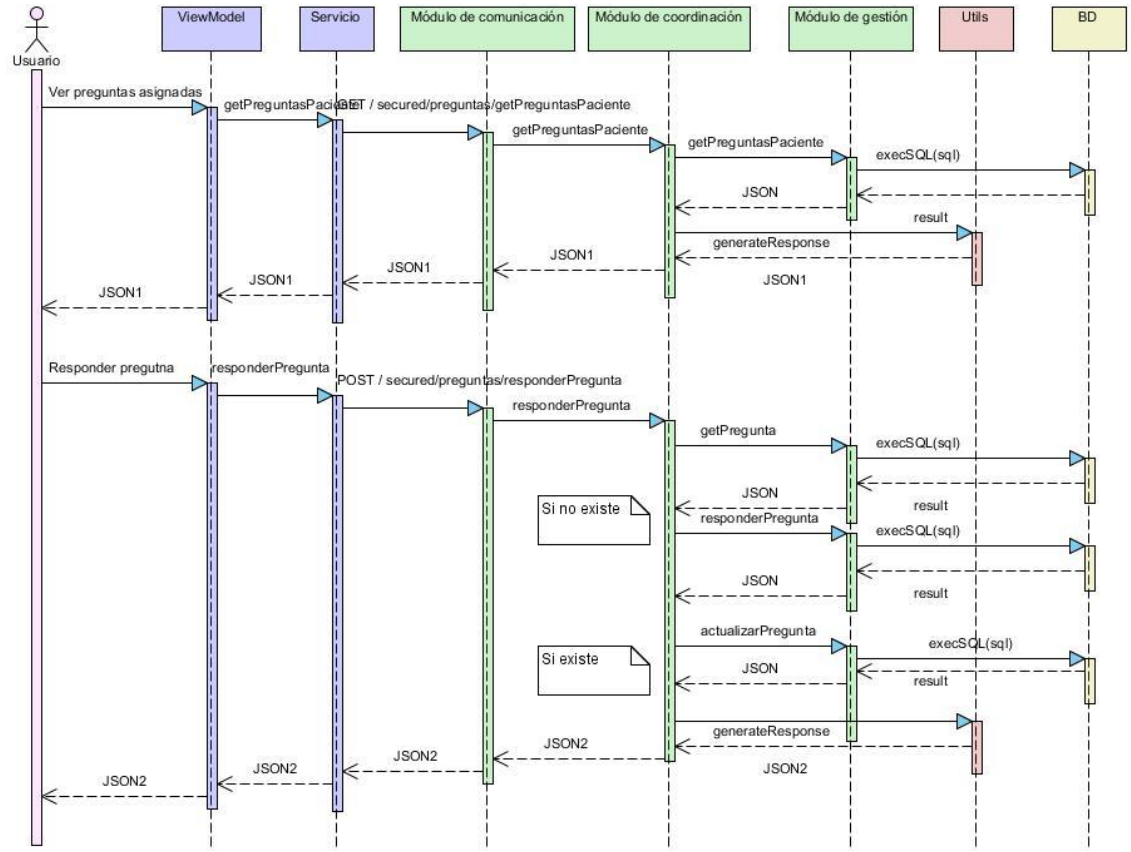
A continuación, se puede ver un ejemplo de la respuesta JSON2:


```
{
    "status": 201,
    "code": 20124,
    "msg": "Se ha obtenido el objetivo del paciente",
    "body": {
        "distancia": 5002,
        "tiempo": "60"
    }
}
```


Ilustración 141 - Respuesta get objetivo

Responder preguntas

Este diagrama de secuencia corresponde con el caso de uso extendido de "Responder preguntas" de la aplicación móvil.



La sentencia SQL para "getPreguntasPaciente" es la siguiente:

```
SELECT pre.texto, pre.id, pre.inicial, pre.keyboard from pregunta pre
JOIN tratamientopregunta trpre ON pre.id = trpre.idPre
where trpre.idTrat=? AND (trpre.fechaFin is null) AND pre.inicial=?
```

La sentencia SQL para "getPregunta" es la siguiente:

```
Select respuesta from tratamientopreguntarespondida where idPre=? and
idTrat=? and fecha=?
```

La sentencia SQL para "responderPregunta" es la siguiente:

```
INSERT INTO tratamientopreguntarespondida (idPre,idTrat, fecha, respuesta)
VALUES (?,?,?,?)
```

La sentencia SQL para "actualizarPregunta" es la siguiente:

```
UPDATE tratamientopreguntarespondida
SET respuesta=?
WHERE idPre=? and idTrat=? and fecha=?
```

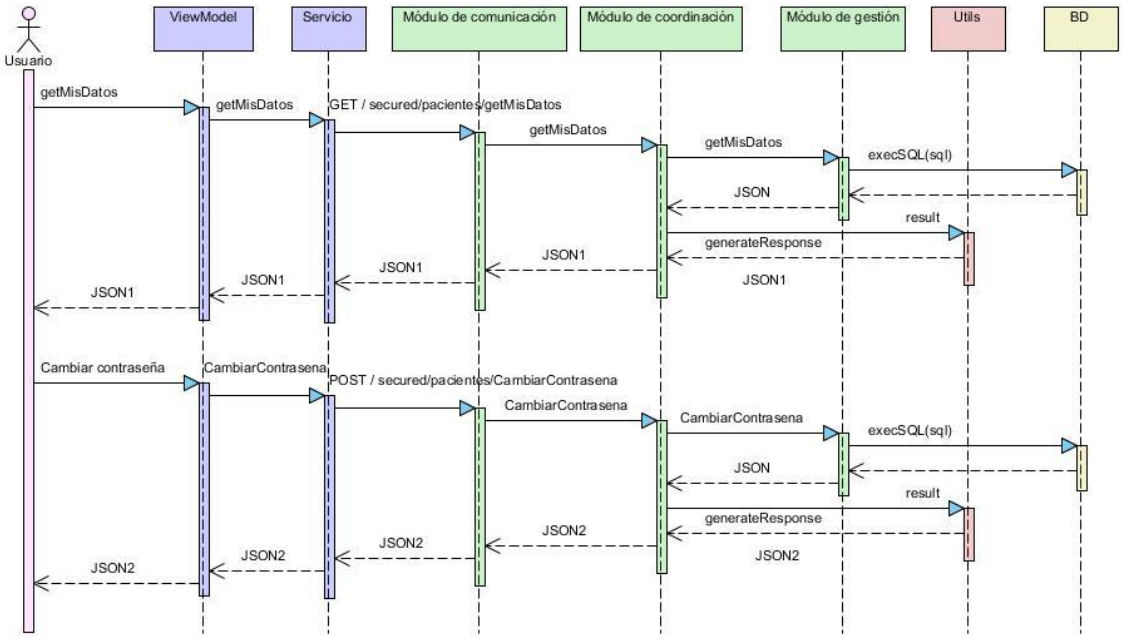
A continuación, se puede ver un ejemplo de la respuesta JSON1:


```
{
    "status": 200,
    "code": 20066,
    "msg": "La preguntas obtenidas de forma correcta.",
    "body": [
        {
            "texto": "¿Ha tomado su medicación habitual?",
            "id": 1,
            "inicial": "no",
            "keyboard": "yesno"
        },
```


Ilustración 142 - Respuesta get preguntas del paciente

Opciones

Este diagrama de secuencia corresponde con el caso de uso extendido de "Opciones" de la aplicación móvil.



La sentencia SQL para "getMisDatos" es la siguiente:

```
SELECT t.fechaFPre, t.fechaFPost, '+
p.nombre, p.apellidos, p.numHist, p.usuario,p.fechaNac,cam.distancia,
cam.tiempo, '+
g.nombre AS grupo  '+
FROM grupoPaciente gp  '+
JOIN paciente p ON gp.numHistPac = p.numHist  '+
JOIN tratamiento t ON p.numHist = t.numHist  '+
JOIN grupoRiesgo g ON gp.nombreGrupo = g.nombre  '+
JOIN tratamientocaminar trc ON t.id = trc.idTrat  '+
JOIN caminar cam ON cam.id = trc.idCaminar  '+
where t.id=? AND (trc.fechaFin is null)
```

La sentencia SQL para “CambiarContrasena” es la siguiente:

```
UPDATE paciente SET contraseña=? WHERE contraseña=? and token=?
```

A continuación, se puede ver un ejemplo de la respuesta JSON1:


```
{
    "status": 200,
    "code": 200212,
    "msg": "Datos del paciente obtenidos de forma correcta.",
    "body": {
        "fechaFPre": "2019-06-30T00:00:00.000Z",
        "fechaFPost": "2019-07-31T00:00:00.000Z",
        "nombre": "Alejandro",
        "apellidos": "Gomez",
        "numHist": 1568,
        "usuario": "a",
        "fechaNac": "2019-05-08T00:00:00.000Z",
        "distancia": 5002,
        "tiempo": "60",
        "grupo": "Cardiaco"
    }
}
```


Ilustración 143 - Respuesta get mis datos