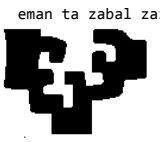


Escuela Universitaria de Ingeniería Técnica Industrial de Bilbao

Grado en Ingeniería en Informática de Gestión y Sistemas de la Información

Trabajo Fin de Grado

2014 / 2015

Gestor de Radio FM y Sistema RDS
FM Radio RDS

Memoria

Datos del Alumno	**Datos del Director**
Nombre: Imanol	Nombre: Mikel
Apellidos: Gonzalez Barcina	Apellidos: Villamañe Gironés
	Departamento: Lenguajes y Sistemas Informáticos
Fdo.:	Fdo.:
Fecha:	Fecha:

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

El proyecto a desarrollar es una aplicación (también llamado app) para un sistema operativo móvil relativamente nuevo, Firefox OS. Consiste en un gestor de radio FM avanzado que, con ayuda de varios de los sensores que contienen los dispositivos, como el de radio y el de GPS, tratará de mejorar la experiencia de uso del usuario a la hora de escuchar la radio. El nombre del proyecto será FM Radio RDS.

Con la intención de crear una herramienta que se pueda mantener y actualizar su contenido en gran parte por sí misma, se desea lograr la implicación de las personas que utilicen el sistema para mejorar y completar el funcionamiento de FM Radio RDS. La información reportada por los usuarios colaboradores de la app ayudará al desarrollador a mantener al día la información en la que se basa el sistema para su funcionamiento, por lo que se espera la aportación objetiva y de datos veraces de los usuarios. A lo largo del trabajo se describen algunos métodos que se implementan para valorar la calidad de la información aportada por estos usuarios colaboradores.

Con este trabajo de fin de grado se quiere poner al alcance del usuario un servicio innovador y sencillo, tanto para la búsqueda y la sintonización de las emisoras como en su gestión, que complemente lo que hasta ahora existe y añadiendo una serie de mejoras en cuanto al funcionamiento de la aplicación de serie, utilizando nuevas posibilidades que disponen los dispositivos hoy en día.

La razón principal para la elaboración de este proyecto ha sido la búsqueda de una solución a un problema que desde hacía tiempo se estaba viendo y echando en falta en los dispositivos inteligentes que se usan habitualmente.

Hace unos años, realizando un viaje de Bilbao a Madrid en autobús, el desarrollador de esta aplicación se dio cuenta que no había un sistema de radio potente para los smartphones que lograra no perder la señal de la emisora de radio que se estuviera escuchando. Su experiencia personal es que al salir de las grandes ciudades, suele ocurrir que se pierde la señal de la cadena que se está escuchando, la calidad de la sintonía se reduce y se tiende a cambiar de emisora realizando una búsqueda manual, intentando recuperar la emisora que se estaba escuchando. Se hizo un análisis de los sistemas que incorporan algunos vehículos como los coches y su sistema RDS[1] [1][el número de este corchete indica el elemento de la bibliografía] y se propuso simular su funcionamiento para un dispositivo móvil.

1 Radio Data System

Teniendo decidido el enfoque del trabajo de fin de grado, hay que analizar los diferentes sistemas operativos móviles existentes (Android, iOS, Windows Phone, Firefox OS, etc) y las características de cada uno para poder decidir en cual o cuales de ellos centrarse. La decisión es desarrollar el proyecto en un incipiente sistema operativo móvil como es Firefox OS debido a la posibilidad de manejar elementos del hardware del dispositivo desde la propia aplicación web (en adelante webapp), algo que resulta imposible en las otras grandes plataformas al no tener liberadas las APIs que gestionan el sensor de radio (Android) o porque directamente no incorporan este sensor dentro de las características del terminal (iOS y algunos Android).

Otra de las razones para tomar esta decisión son las ganas de lograr un producto diferenciador con la competencia. Hoy en día, la radio digital a través de Internet es una realidad, existiendo varias aplicaciones que retransmiten la señal de las cadenas con buena calidad, pero que necesitan de una conexión a Internet para su escucha. Además, las aplicaciones de radio que vienen de serie en los terminales son demasiado básicas, no han evolucionado a la par que otras de las herramientas de estos dispositivos. Con FM Radio RDS se quiere evitar el gran consumo continuo de datos móviles y aportar un gestor de radio avanzado, utilizando el sensor de radio integrado del dispositivo, además del sensor de geoposición.

A la hora de enfocar este proyecto, el principal escollo encontrado es cómo conseguir todas las frecuencias de las emisoras de radio en cada posición global. Buscando a través de varias páginas web, existen enormes listados de emisoras de radio y sus frecuencias según la población donde se emiten. Se podría utilizar esta información para rellenar la base de datos que el sistema utilizará para ir eligiendo la frecuencia óptima en cada momento y lugar. Sin embargo, de ese modo no se estaría ofreciendo un servicio de calidad, ya que no asegura de ningún modo que los datos estuvieran actualizados ni sean correctos en cada posición ni contemplaría cambios futuros. La opción de que el desarrollador realizase cambios manuales en el contenido de la base de datos no se contempla a excepción de casos puntuales. Por ello, se pensó en desarrollar una aplicación inicial independiente (RFOS[2]) que posteriormente estuviera integrada en la aplicación final FM Radio RDS.

El funcionamiento de RFOS consiste en que un usuario con un terminal con Firefox OS ejecute esta app, el sistema realice una búsqueda de todas las señales de radio que recibe en ese instante y lugar, y devuelva la información obtenida a la aplicación de forma que se pueda rellenar automáticamente la base de datos. El escaneo inicial realizado por RFOS conseguiría obtener las frecuencias de todas las

2 Radio Frequency Obteining System

cadenas de radio, en función de la posición donde se ha realizado esa búsqueda. La información del nombre de la emisora la añadirá manualmente el usuario para cada una de ellas, y así mejorar la clasificación en la base de datos.

Con el apoyo de la comunidad de beta-testers y los usuarios colaboradores, conseguidos a través de comunidades de Google+, usuarios de Twitter interesados en la aplicación y el sistema operativo móvil, foros como Mozilla Hispana y algún otro, se conseguiría ir completando la base de datos de las emisiones. El desarrollador creará un sistema para controlar la calidad de toda la información aportada por los usuarios colaboradores, verificando en la medida de lo posible la validez de estos datos y dando la opción al resto de usuarios de valorar la calidad de esta información.

2. Planteamiento inicial

En este punto se desarrollan varios apartados del planteamiento inicial del trabajo final de grado, como son los objetivos, el alcance, la planificación temporal, las herramientas utilizadas, la gestión de riesgos y la evaluación económica.

2.1. Objetivos

El objetivo principal del desarrollo de la aplicación FM Radio RDS es lograr crear un gestor de radio FM avanzado donde los usuarios puedan disponer al alcance de su mano un listado de emisoras de radio en función de la posición donde se encuentran, además de simular el funcionamiento del sistema RDS anteriormente explicado. Se quiere dotar al usuario de una herramienta que no consuma gran cantidad de datos de Internet en el acceso a la información almacenada en la base de datos final, por ello se da importancia al tipo de base de datos interna que maneja el sistema operativo para varias de las funcionalidades de FM Radio RDS.

Habrá dos tipos de usuarios a la hora de usar la app, el identificado y el no identificado. Se podrán crear perfiles de usuario e identificarse como tal. Si el usuario no está identificado en el sistema, las funcionalidades a su disposición se reducen.

Se desea que los usuarios identificados colaboren puntuando la calidad de la recepción de una emisora en una posición concreta para mejorar las recomendaciones futuras al resto de usuarios de la plataforma, así como que añadan emisoras a su listado de emisoras favoritas, información que luego le aparecerán en su perfil. Este tipo de usuarios podrán ejecutar los sistema RFOS y RDS manualmente, explicados en la introducción.

Será necesaria la conexión a Internet para el funcionamiento completo de la app, pero se utilizarán componentes tecnológicos de la web más actual para el almacenamiento de información en el dispositivo, para simular el funcionamiento en modo sin conexión de la aplicación en varios de sus apartados.

La webapp deberá tener unas interfaces atractivas, intuitivas y de fácil manejo, para que los usuarios la prefieran antes que a otras de características y funcionalidades similares. Una de sus grandes virtudes será el acceso multiplataforma al contenido de la aplicación web desarrollada, ya que esta se adapta a los navegadores más modernos y los terminales móviles de cualquier ecosistema. De este modo, se consigue abrir el nicho de mercado de los posibles

usuarios, de solamente usuarios con terminales Firefox OS a todos los smartphones de diferentes sistemas operativos.

Con todo ello, el principal propósito es aprender a desarrollar una aplicación móvil que pudiera aportar algo en este mundo donde se supone todo está desarrollado, innovar para dar solución a un sistema incompleto y mejorable.

2.2. Alcance

Este apartado se divide en varias subpartes descriptivas:

Fases del proyecto

Gestión: es el primer paso a la hora de realizar un proyecto y en este punto se definirá como llevarla a cabo. Para ello, habrá que buscar información, estar al día de las novedades y definir las tareas que van a componer.

Análisis: en este punto se definirán las funcionalidades del proyecto, así como las herramientas necesarias para su realización y el tipo de sistema de gestión de bases de datos, entre otros.

Diseño: en este paso, se definirá la estructura de la app y sus interfaces gráficas. Tendrá que cumplir con lo estipulado en los puntos de gestión y análisis anteriores.

Implementación y pruebas: se definirán los diferentes prototipos en los que se va a dividir el sistema, se realizará la implementación y las correspondientes pruebas para cada prototipo, para asegurar su correcto funcionamiento. Al acabar la implementación, se harán las pruebas finales.

En el siguiente listado aparecen los prototipos a desarrollar: registrarse en el sistema, identificarse, ver comentarios de usuarios cercanos, ver emisoras cercanas, RDS, elegir emisora al pulsar lista, elegir emisión al pulsar lista, play/stop, gestionar perfil, ver favoritos, gestionar perfil, cerrar sesión y eliminar cuenta, añadir emisoras manualmente, valorar emisora en posición concreta y frecuenta concreta, poner favorita emisora, hacer escaneo RFOS y ver información completa de la emisora.

Documentación: será el último paso a seguir, donde se reunirá toda la documentación y experiencias obtenidas a lo largo de todo el proyecto y plasmarlas en el documento, así como la preparación de la defensa.

Esquema de descomposición del proyecto

En la siguiente figura, se representa gráficamente la descomposición inicial del trabajo.

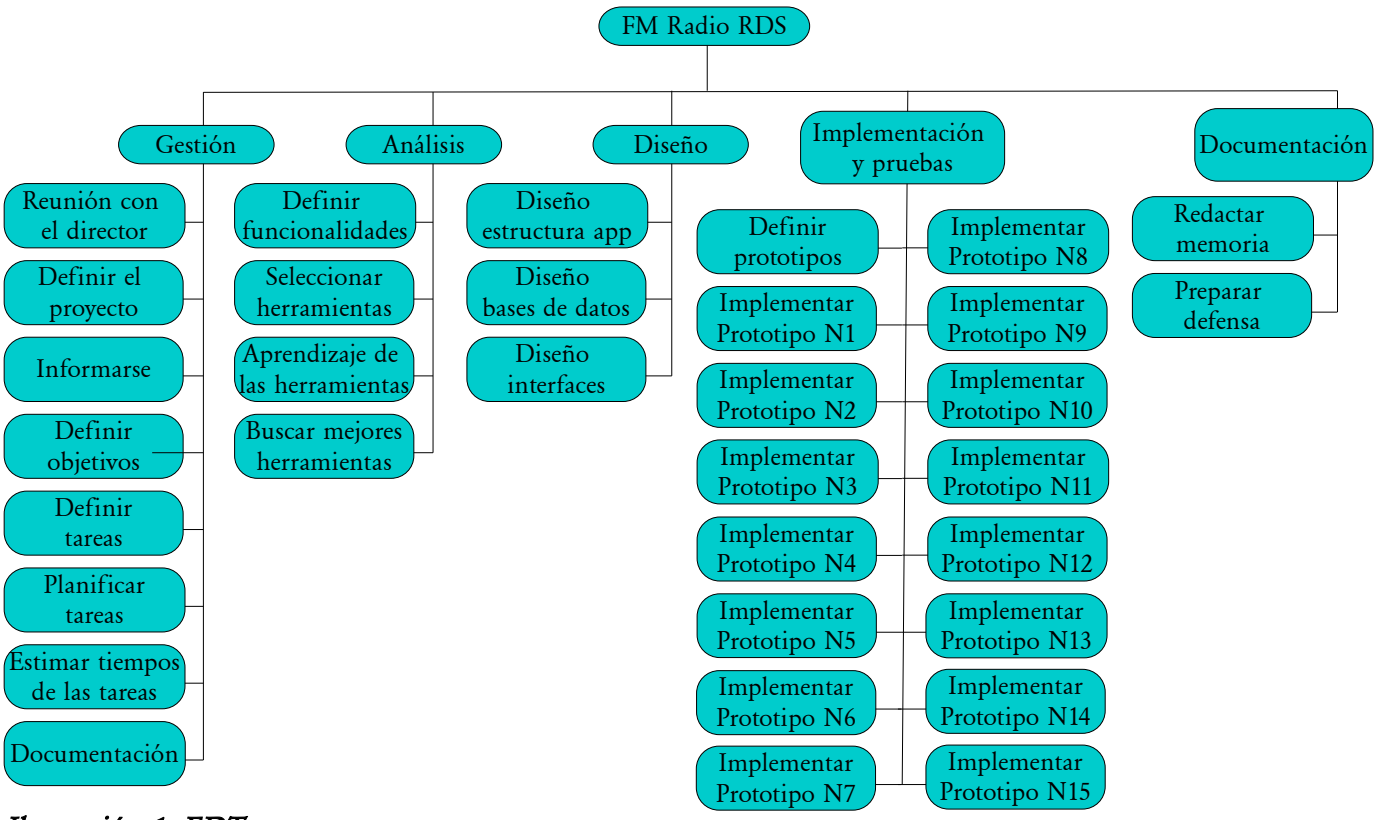


Ilustración 1: EDT

Tareas

A continuación, se van a definir todas las tareas esquematizadas en el EDT.

1 Gestión

1.1 Paquete de trabajo: Reunión con el director

Duración: 2 horas.

Descripción: Tener una reunión con tu director del proyecto donde se expone la idea a realizar, que funcionalidades se van a desarrollar en el trabajo, con una descripción concisa de todo ello.

Salidas/Entregables: Una descripción informal del proyecto.

Recursos necesarios: Tener la idea clara de la aplicación a desarrollar.

1.2 Paquete de trabajo: Informarse

Duración: 12 horas.

Descripción: Una vez escuchados los consejos del director del trabajo de fin de grado, es necesaria una profunda búsqueda de información sobre varios aspectos fundamentales como el sistema operativo sobre el que se va a desarrollar la app, los lenguajes a utilizar, etc.

Recursos necesarios: Un dispositivo con conexión a Internet y saber que buscar.

Precedencias: Paquete de trabajo 1.1.

1.3 Paquete de trabajo: Definir el proyecto.

Duración: 4 horas.

Descripción: Siguiendo los consejos del director del proyecto y la información obtenida tras la intensa búsqueda, se define el proyecto final por completo.

Salidas/Entregables: Una descripción formal del proyecto.

Recursos necesarios: Tener la idea totalmente clara de la aplicación a desarrollar.

Precedencias: Paquete de trabajo 1.1 y Paquete de trabajo 1.2.

1.4 Paquete de trabajo: Definir objetivos

Duración: 2 horas.

Descripción: Establecer los objetivos que se van a llevar a cabo en el proyecto.

Entrada: La descripción del proyecto.

Salidas/Entregables: Los objetivos del proyecto.

Recursos necesarios: Tener la idea totalmente clara de la aplicación a desarrollar.

Precedencias: Paquete de trabajo 1.3.

1.5 Paquete de trabajo: Definir tareas

Duración: 4 horas.

Descripción: Establecer las tareas que van a ser necesarias para llevar a cabo para realizar el proyecto.

Entrada: Los objetivos a cumplir del proyecto.

Salidas/Entregables: Las tareas que serán necesarias, con su descripción.

Recursos necesarios: Tener claras las fases del proyecto.

Precedencias: Paquete de trabajo 1.4.

1.6 Paquete de trabajo: Planificar tareas

Duración: 3 horas.

Descripción: Establecer un plan para el desarrollo de las tareas y ver cuales se podrán subdividir en otras.

Entrada: La descripción de las tareas que haga falta planificar.

Salidas/Entregables: Las planificación de estas tareas sin los tiempos.

Recursos necesarios: Tener claras las tareas que se van a llevar a cabo.

Precedencias: Paquete de trabajo 1.5.

1.7 Paquete de trabajo: Estimar tiempos de las tareas

Duración: 3 horas.

Descripción: Realizar una estimación aproximada del tiempo que se va a necesitar emplear para la realización de las tareas.

Entrada: La descripción de las tareas y su planificación.

Salidas/Entregables: Las planificación de estas tareas con los tiempos aproximados ya establecidos.

Recursos necesarios: Tener claras las tareas que se van a llevar a cabo.

Precedencias: Paquete de trabajo 1.6.

1.8 Paquete de trabajo: Documentación

Duración: 16 horas.

Descripción: En este punto habrá que reunir toda la documentación que se ha creado y unificarla, para conseguir avanzar con seguridad. Las tareas, sus tiempos, la descripción del proyecto, etc.

Entrada: Todo lo nombrado.

Salidas/Entregables: El documento de objetivos del proyecto.

Recursos necesarios: Todo lo necesario para llevar a cabo el documento de objetivos del proyecto.

Precedencias: Todas las tareas descritas hasta ahora en este apartado 1, la fase de Gestión. Desde el paquete de trabajo 1.1 al 1.7.

2 Análisis

2.1 Paquete de trabajo: Definir funcionalidades

Duración: 18 horas.

Descripción: Definir las funcionalidades que se van a implementar en la aplicación.

Salidas/Entregables: Los documentos que tengan algo que ver con las funcionalidades de la aplicación.

Recursos necesarios: Conocer en concreto que es lo que va a realizar el proyecto para poder definir sus funcionalidades.

Precedencias: Paquete de trabajo 1.1.

2.2 Paquete de trabajo: Seleccionar herramientas

Duración: 6 horas.

Descripción: Elegir las herramientas que se van a emplear para el desarrollo del proyecto, así como la búsqueda y visualización de tutoriales que faciliten el aprendizaje.

Recursos necesarios: Conocer en concreto que es lo que va a realizar el proyecto para poder definir sus funcionalidades.

Precedencias: Saber los lenguajes de programación en que se van a

desarrollar para elegir la herramienta más adecuada.

2.3 Paquete de trabajo: Aprendizaje de las herramientas

Duración: 10 horas.

Descripción: Una vez elegidas las herramienta a emplear, hay que aprender a usarlas con soltura y rapidez. Eso conlleva un tiempo.

Recursos necesarios: Tener en nuestros equipos las herramientas necesarias instaladas.

Precedencias: Conocer los diferentes lenguajes de programación en que se desarrolla.

2.4 Paquete de trabajo: Buscar mejores herramientas

Duración: 6 horas.

Descripción: La extensa lista de herramientas descrita en el punto 2.4 de la memoria se debe al intento de estar al día con las novedades que surgen, herramientas que desarrolladores web recomiendan, etc. La búsqueda de mejores programas para desarrollar el trabajo es una constante necesaria en el mundo de la tecnología

Recursos necesarios: Conexión a Internet y ganas de leer muchas entradas en blogs, post en redes sociales y plataformas lectoras de RSS para estar al día de lo que se publica.

3 Diseño

3.1 Paquete de trabajo: Diseño estructura de la app

Duración: 24 horas.

Descripción: Realizar un diseño de la estructura de la aplicación, analizando bien las partes de las que va a constar el proyecto, desarrollando un algoritmo que albergue todas estas funcionalidades pensadas.

Salidas/Entregables: El algoritmo de la estructura.

Recursos necesarios: Tener clara la estructura que tendrá la aplicación.

Precedencias: Paquete de trabajo 2.1.

3.2 Paquete de trabajo: Diseño de las bases de datos

Duración: 8 horas.

Descripción: En este paquete de trabajo se va a diseñar una base de datos para que sea utilizada por el software que se está desarrollando. Esto permitirá almacenar los datos necesarios para que funcione correctamente y poder tenerlos a la hora de su uso. Para hacer esta base de datos, habrá que tener en cuenta el modelo de dominio, el cual se pasará a tablas para poder insertarlo dentro de la BD en PhpMyAdmin. Esta es la herramienta que se va a utilizar para su administración. También hay que diseñar el uso de la base de datos interna del dispositivo IndexedDB.

Entradas: El modelo de dominio, servirá como esquema para la base de datos total.

Salidas/Entregables: El diseño general de las bases de datos.

Recursos necesarios: Tener claros los datos que se van a almacenar en los entornos, todos los datos que se usarán a lo largo de la app. Además, un ordenador con acceso a Internet y el programa XAMPP Control Panel para gestionar la base de datos.

Precedencias: Paquete de trabajo 2.1.

3.3 Paquete de trabajo: Diseño de las interfaces

Duración: 16 horas.

Descripción: Realizar un diseño preciso del aspecto que se le quiera dar a la aplicación, pensar en cada una de las pantallas y la colocación de los elementos que compondrán la app.

Salidas/Entregables: El diseño de las interfaces.

Recursos necesarios: Tener una idea de como será la aplicación, desde el punto de vista visual.

Precedencias: Paquete de trabajo 2.1.

4 Implementación y pruebas

4.1 Paquete de trabajo: Definir prototipos

Duración: 16 horas.

Descripción: La app se dividirá en diferentes prototipos para facilitar su implementación. En cada uno, se irán añadiendo funcionalidades hasta construir la aplicación completa.

Salidas/Entregables: Se logran los algoritmos de los prototipos.

Recursos necesarios: Tener claras las funcionalidades.

Precedencias: Los grupos de tareas 1, 2 y 3.

4.2 Paquete de trabajo: Implementar prototipo 1 "Registrarse en el sistema" y probar su funcionamiento.

Duración: 28 horas.

Descripción: Se implementará el prototipo de "Registrarse" y se realizarán las pruebas pertinentes para comprobar su funcionamiento.

Salidas/Entregables: El código implementado y probado.

Recursos necesarios: El prototipo definido.

Precedencias: Los grupos de tareas 1, 2 y 3.

4.3 Paquete de trabajo: Implementar prototipo 2 "Identificarse" y probar su funcionamiento.

Duración: 16 horas.

Descripción: Se implementará el prototipo de "Identificarse" y se realizarán las pruebas pertinentes para comprobar su funcionamiento.

Salidas/Entregables: El código implementado y probado.

Recursos necesarios: El prototipo definido.

Precedencias: Los grupos de tareas 1, 2 y 3.

4.4 Paquete de trabajo: Implementar prototipo 3 "Ver comentarios de usuarios cercanos" y probar su funcionamiento.

Duración: 16 horas.

Descripción: Se implementará el prototipo de "Ver comentarios de usuarios cercanos" y se realizarán las pruebas pertinentes para comprobar su funcionamiento.

Salidas/Entregables: El código implementado y probado.

Recursos necesarios: El prototipo definido.

Precedencias: Los grupos de tareas 1, 2 y 3.

4.5 Paquete de trabajo: Implementar prototipo 4 "Ver emisoras cercanas." y probar su funcionamiento.

Duración: 16 horas.

Descripción: Se implementará el prototipo de "Ver emisoras cercanas" y se realizarán las pruebas pertinentes para comprobar su funcionamiento.

Salidas/Entregables: El código implementado y probado.

Recursos necesarios: El prototipo definido.

Precedencias: Los grupos de tareas 1, 2 y 3.

4.6 Paquete de trabajo: Implementar prototipo 5 "RDS" y probar su funcionamiento.

Duración: 40 horas.

Descripción: Se implementará el prototipo de "RDS" y se realizarán las pruebas pertinentes para comprobar su funcionamiento.

Salidas/Entregables: El código implementado y probado.

Recursos necesarios: El prototipo definido.

Precedencias: Los grupos de tareas 1, 2 y 3.

4.7 Paquete de trabajo: Implementar prototipo 6 "Elegir emisora al pulsar lista" y probar su funcionamiento.

Duración: 20 horas.

Descripción: Se implementará el prototipo de "Elegir emisora al pulsar lista" y se realizarán las pruebas pertinentes para comprobar su funcionamiento.

Salidas/Entregables: El código implementado y probado.

Recursos necesarios: El prototipo definido.

Precedencias: Los grupos de tareas 1, 2 y 3.

4.8 Paquete de trabajo: Implementar prototipo 7 "Elegir emisión al pulsar lista" y probar su funcionamiento.

Duración: 20 horas.

Descripción: Se implementará el prototipo de "Elegir emisión al pulsar lista" y se realizarán pruebas para comprobar su funcionamiento.

Salidas/Entregables: El código implementado y probado.

Recursos necesarios: El prototipo definido.

Precedencias: Los grupos de tareas 1, 2 y 3.

4.9 Paquete de trabajo: Implementar prototipo 8 "Elegir emisora y frecuencia al pulsar lista" y probar su funcionamiento.

Duración: 32 horas.

Descripción: Se implementará el prototipo de "Elegir emisora y frecuencia al pulsar lista" y se realizarán las pruebas pertinentes para comprobar su funcionamiento.

Salidas/Entregables: El código implementado y probado.

Recursos necesarios: El prototipo definido.

Precedencias: Los grupos de tareas 1, 2 y 3.

4.10 Paquete de trabajo: Implementar prototipo 9 "Gestionar perfil - Ver favoritos" y probar su funcionamiento.

Duración: 12 horas.

Descripción: Se implementará el prototipo de "Gestionar perfil - Ver favoritos" y se realizarán las pruebas para comprobar su funcionamiento.

Salidas/Entregables: El código implementado y probado.

Recursos necesarios: El prototipo definido.

Precedencias: Los grupos de tareas 1, 2 y 3.

4.11 Paquete de trabajo: Implementar prototipo 10 "Gestionar perfil - Cerrar sesión y Eliminar cuenta" y probar su funcionamiento.

Duración: 16 horas.

Descripción: Se implementará el prototipo de "Gestionar perfil - Cerrar sesión y Eliminar cuenta" y se realizarán las pruebas pertinentes para comprobar su funcionamiento.

Salidas/Entregables: El código implementado y probado.

Recursos necesarios: El prototipo definido.

Precedencias: Los grupos de tareas 1, 2 y 3.

4.12 Paquete de trabajo: Implementar prototipo 11 "Añadir emisoras manualmente" y probar su funcionamiento.

Duración: 16 horas.

Descripción: Se implementará el prototipo de "Añadir emisoras manualmente" y se realizarán pruebas para comprobar su

funcionamiento.

Salidas/Entregables: El código implementado y probado.

Recursos necesarios: El prototipo definido.

Precedencias: Los grupos de tareas 1, 2 y 3.

4.13 Paquete de trabajo: Implementar prototipo 12 "Valorar emisora en posición y frecuenta concreta" y probar su funcionamiento.

Duración: 20 horas.

Descripción: Se implementará el prototipo de "Valorar emisora en posición y frecuencia concreta" y se realizarán las pruebas pertinentes para comprobar su funcionamiento.

Salidas/Entregables: El código implementado y probado.

Recursos necesarios: El prototipo definido.

Precedencias: Los grupos de tareas 1, 2 y 3.

4.14 Paquete de trabajo: Implementar prototipo 13 "Poner favorita emisora" y probar su funcionamiento.

Duración: 16 horas.

Descripción: Se implementará el prototipo de "Valorar emisora" y se realizarán las pruebas pertinentes para comprobar su funcionamiento.

Salidas/Entregables: El código implementado y probado.

Recursos necesarios: El prototipo definido.

Precedencias: Los grupos de tareas 1, 2 y 3.

4.15 Paquete de trabajo: Implementar prototipo 14 "Hacer escaneo RFOS" y probar su funcionamiento.

Duración: 60 horas.

Descripción: Se implementará el prototipo de "Hacer escaneo RFOS" y se realizarán las pruebas pertinentes para comprobar su funcionamiento.

Salidas/Entregables: El código implementado y probado.

Recursos necesarios: El prototipo definido.

Precedencias: Los grupos de tareas 1, 2 y 3.

4.16 Paquete de trabajo: Implementar prototipo 15 "Ver información completa de la emisora." y probar su funcionamiento.

Duración: 16 horas.

Descripción: Se implementará el prototipo de "Ver información completa de la emisora" y se realizarán las pruebas pertinentes para comprobar su funcionamiento.

Salidas/Entregables: El código implementado y probado.

Recursos necesarios: El prototipo definido.

Precedencias: Los grupos de tareas 1, 2 y 3.

5 Documentación

5.1 Paquete de trabajo: Redactar la memoria

Duración: 40 horas.

Descripción: Escribir por completo toda la memoria del trabajo de final de grado, incluyendo en ella toda la documentación reunida, las tareas a realizar, los diseños, problemas y dificultades encontradas, conclusiones, etc. Lo que viene siendo un documento con la recopilación de toda la labor realizada en el desarrollo de la aplicación.

Salidas/Entregables: La memoria final.

Recursos necesarios: El proyecto y la documentación recopilada a lo largo de todo el proyecto.

Precedencias: Todas las fases anteriores.

5.2 Paquete de trabajo: Preparar la defensa

Duración: 12 horas.

Descripción: Preparar la defensa del trabajo de fin de grado.

Salidas/Entregables: La presentación realizada para exponer el trabajo frente al tribunal.

Recursos necesarios: El proyecto finalizado y la memoria.

Precedencias:: Todas las fases anteriores.

2.3. Planificación Temporal

En la planificación temporal, se establece una estimación de cuántas horas conllevará la realización del proyecto, pudiendo calcular cuantas semanas serán necesarias para acabarlo. Además, se dispondrá de un gráfico GANTT para representar la planificación temporal a lo largo de las semanas.

Tareas	Horas dedicadas (Estimación)
1 Gestión	**46 horas**
1.1 Reunión con el director	2 horas
1.2 Informarse	12 horas
1.3 Definir el proyecto	4 horas
1.4 Definir objetivos	2 horas
1.5 Definir tareas	4 horas
1.6 Planificar tareas	3 horas
1.7 Estimar tiempos de las tareas	3 horas
1.8 Documentación	16 horas
2 Análisis	**40 horas**
2.1 Definir funcionalidades	18 horas
2.2 Seleccionar herramientas	6 horas
2.3 Aprendizaje de herramientas	10 horas
2.4 Buscar mejores herramientas	6 horas
3 Diseño	**48 horas**
3.1 Diseño estructura de la app	24 horas
3.2 Diseño bases de datos	8 horas
3.3 Diseño de interfaces	16 horas
4 Implementación y pruebas	**360 horas**
4.1 Definir prototipos	16 horas

Tabla 1: Planificación temporal de las tareas - Parte 1

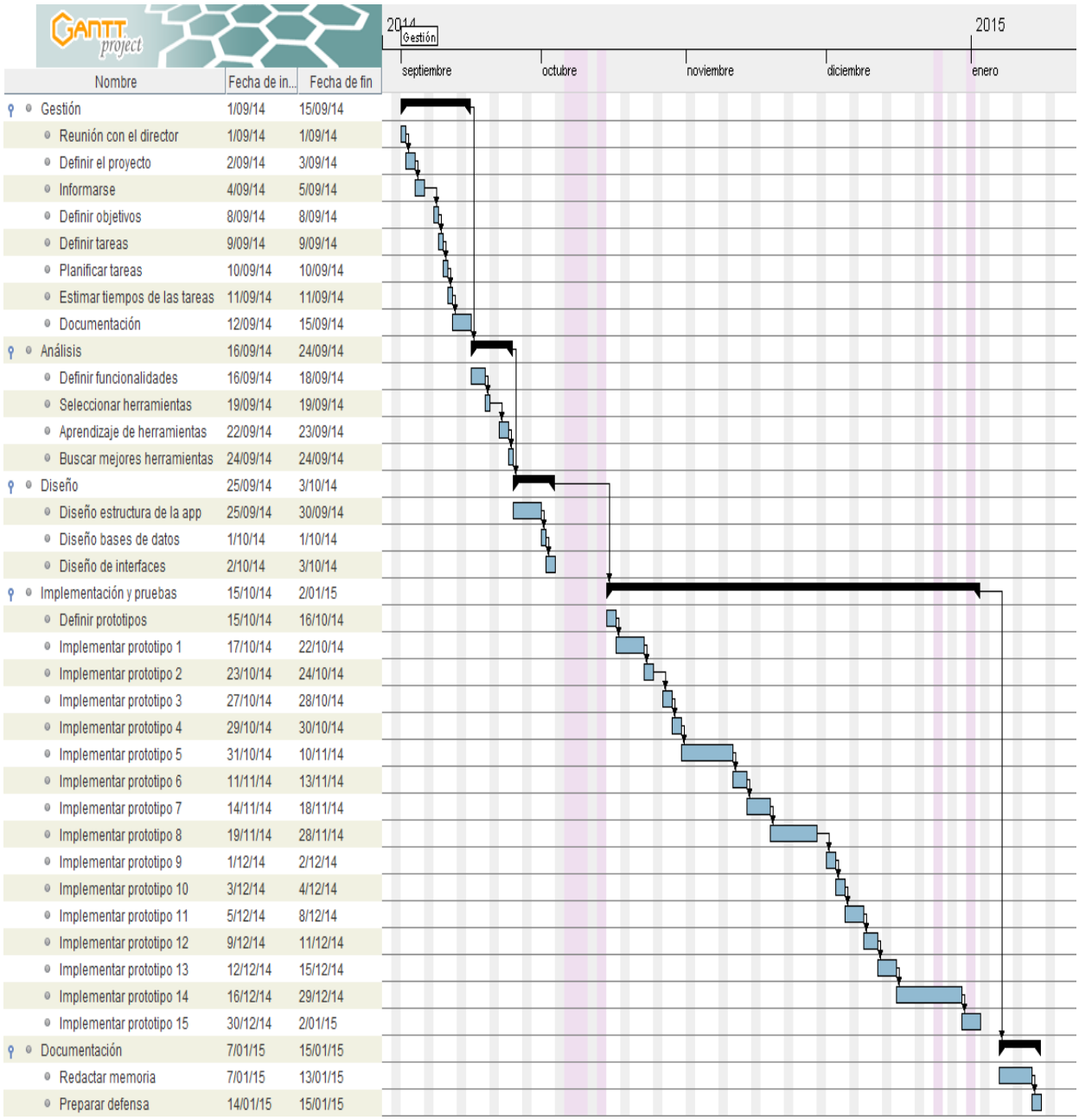
Tareas	Horas dedicadas (Estimación)
4.2 Implementar prototipo 1	28 horas
4.3 Implementar prototipo 2	16 horas
4.4 Implementar prototipo 3	16 horas
4.5 Implementar prototipo 4	16 horas
4.6 Implementar prototipo 5	40 horas
4.7 Implementar prototipo 6	20 horas
4.8 Implementar prototipo 7	20 horas
4.9 Implementar prototipo 8	32 horas
4.10 Implementar prototipo 9	12 horas
4.11 Implementar prototipo 10	16 horas
4.12 Implementar prototipo 11	16 horas
4.13 Implementar prototipo 12	20 horas
4.14 Implementar prototipo 13	16 horas
4.15 Implementar prototipo 14	60 horas
4.16 Implementar prototipo 15	16 horas
5 Documentación	**52 horas**
5.1 Redactar memoria	40 horas
5.2 Preparar defensa	12 horas
Total	**546 horas**

Tabla 2: Planificación temporal de las tareas - Parte 2

Se estima que semanalmente se dedican un total de 40 horas para el desarrollo del trabajo de fin de grado. Las horas dedicadas para la realización son 546, así que teóricamente el proyecto debería estar finalizado en unas 14 semanas.

Al ser un proyecto personal e individual, la gestión de los tiempos será muy particular, dedicándole el tiempo completo del día durante muchas jornadas, independientemente de que sea fin de semana, días festivos o días laborables comunes.

En el siguiente diagrama GANTT se intentan reflejar todos estos esfuerzos, adaptándolo únicamente a las 40 horas semanales que supuestamente se dedican.



Nombre	Fecha de in...	Fecha de fin
Gestión	1/09/14	15/09/14
Reunión con el director	1/09/14	1/09/14
Definir el proyecto	2/09/14	3/09/14
Informarse	4/09/14	5/09/14
Definir objetivos	8/09/14	8/09/14
Definir tareas	9/09/14	9/09/14
Planificar tareas	10/09/14	10/09/14
Estimar tiempos de las tareas	11/09/14	11/09/14
Documentación	12/09/14	15/09/14
Análisis	16/09/14	24/09/14
Definir funcionalidades	16/09/14	18/09/14
Seleccionar herramientas	19/09/14	19/09/14
Aprendizaje de herramientas	22/09/14	23/09/14
Buscar mejores herramientas	24/09/14	24/09/14
Diseño	25/09/14	3/10/14
Diseño estructura de la app	25/09/14	30/09/14
Diseño bases de datos	1/10/14	1/10/14
Diseño de interfaces	2/10/14	3/10/14
Implementación y pruebas	15/10/14	2/01/15
Definir prototipos	15/10/14	16/10/14
Implementar prototipo 1	17/10/14	22/10/14
Implementar prototipo 2	23/10/14	24/10/14
Implementar prototipo 3	27/10/14	28/10/14
Implementar prototipo 4	29/10/14	30/10/14
Implementar prototipo 5	31/10/14	10/11/14
Implementar prototipo 6	11/11/14	13/11/14
Implementar prototipo 7	14/11/14	18/11/14
Implementar prototipo 8	19/11/14	28/11/14
Implementar prototipo 9	1/12/14	2/12/14
Implementar prototipo 10	3/12/14	4/12/14
Implementar prototipo 11	5/12/14	8/12/14
Implementar prototipo 12	9/12/14	11/12/14
Implementar prototipo 13	12/12/14	15/12/14
Implementar prototipo 14	16/12/14	29/12/14
Implementar prototipo 15	30/12/14	2/01/15
Documentación	7/01/15	15/01/15
Redactar memoria	7/01/15	13/01/15
Preparar defensa	14/01/15	15/01/15

Ilustración 2: Diagrama Gantt

2.4. Arquitectura

La arquitectura de la aplicación es de tipo cliente/servidor. La aplicación estará alojada en los dispositivo móviles donde se haya instalado y también utilizará el almacenamiento local interno de los terminales para tareas propias a lo largo de su funcionamiento. A su vez, la base de datos con toda la información estará en un servidor, ya que los datos (emisoras, emisiones, puntuaciones, usuarios, comentarios...) que los usuarios agregan y comparten a través de ella serán accesibles por todos los que usen la herramienta.

Se accederá al contenido de la base de datos en el servidor mediante múltiples llamadas AJAX. Se adjunta una ilustración donde se representa la arquitectura explicada.

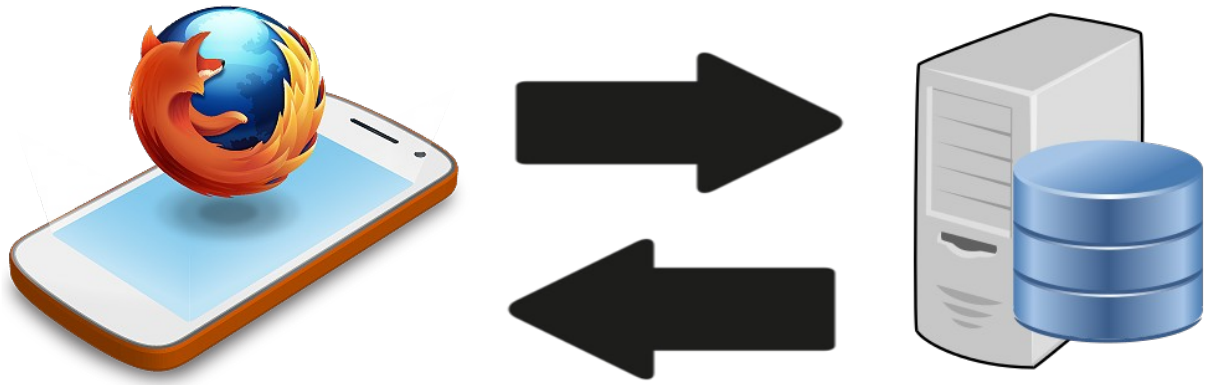


Ilustración 3: Representación de la arquitectura de la aplicación "Cliente - Servidor" [Imagen 1]

2.5. Herramientas

Desarrollar una aplicación web conlleva utilizar prácticamente las mismas herramientas y recursos que para el desarrollo de un entorno web. Para la realización de la aplicación FM Radio RDS se ha necesitado programas de edición de código, navegadores web, un terminal con el sistema operativo Firefox OS, un ordenador portátil y otro de escritorio; acceso, gestión y uso de un servidor donde alojar archivos de código y de bases de datos, así como una conexión constante a Internet.

A continuación, se van a describir en detalle estas herramientas citadas destacando las más utilizadas:

Sistema Operativo Móvil

Para el desarrollo del trabajo de fin de grado ha sido necesario conocer los fundamentos del sistema operativo móvil para el que estaba desarrollando la aplicación. Por ello, a continuación se va a realizar una completa descripción de sus características y alguna comparación con el resto de sistemas operativos móviles con la intención de destacar las bondades y las diferencias entre ellos.

El sistema operativo de la Fundación Mozilla es relativamente nuevo, siendo su lanzamiento inicial en el primer trimestre del año 2013. Firefox OS, cuyo nombre clave es *Boot to Gecko* o *B2G* y su diminutivo más usado *FxOS*, está diseñado para permitir a las aplicaciones web desarrolladas principalmente en HTML5 comunicarse directamente con el hardware del dispositivo usando JavaScript y Open Web APIs [2] [3]. Su arquitectura está dividida en tres partes importantes: *Gaia* [4], *Gecko* [5] y *Gonk* [6].

- Gaia es la interfaz gráfica del sistema operativo. Todo lo que aparece en la pantalla desde que B2G se inicia, es parte de Gaia. Es decir, las aplicaciones tales como la pantalla de bloqueo, el reproductor de radio FM y de música, la aplicación de notas, el "escritorio", la aplicación de mensajes de texto, la agenda de contactos, etc., son parte de Gaia. Esta interfaz gráfica está escrita enteramente en HTML, CSS y JavaScript.

- Gecko es el entorno de ejecución. En Gecko están implementados los estándares de HTML, CSS y JavaScript y permite que esas interfaces se ejecuten correctamente en los distintos sistemas operativos. En otra palabras, el motor Gecko puede ejecutar tareas como visualizar páginas web, manipular la interfaz de usuario o conceder permisos usando las APIs. En la práctica, consiste en una serie de pilas de gráficos, un motor de dibujado y una

máquina virtual para JavaScript, entre otras, escritas en el lenguaje C++.

- Gonk es el SO de bajo nivel de B2G, con un núcleo Linux (al igual que Android) y una capa de abstracción del hardware. El sistema hereda los drivers y componentes típicos de una distro GNU/Linux para comunicarse y poder realizar las funciones esenciales de un smartphone, como el control de batería, los sensores de luz y proximidad, las cámaras frontal y posterior, etc. Para conceder los permisos de alto nivel se ejecuta el proceso *b2g* que permite la interacción total con el sistema usando la capa de Gecko.

- Ejemplo, visualización de un video. Firefox usa un proceso (la función *MediaServer* para el empleo de códecs libres) para la ejecución del video. Este proceso consistiría en pedir permisos a Gonk para la reproducción de ese archivo. En caso de tener permisos para poder reproducirlo en ese instante, Gecko se encargaría de la decodificación de la llamada para mostrar el resultado en Gaia, en la pantalla de nuestro dispositivo.

En estos escasos dos años de vida del sistema operativo móvil FxOS, se han publicado menos de una decena de versiones estables y utilizables por el usuario final, desde la inicial 1.0-Gecko18 hasta la última 2.1-Gecko32, en Octubre del 2014. Todas ellas están disponibles para la descarga e instalación para terminales que lo soporten en la web de la comunidad de desarrollo de Mozilla (a delante MDN[3]).

Para este proyecto, el departamento de informática le proporcionó al desarrollador un terminal para hacer pruebas. Es un modelo para desarrolladores, Geeksphone Peak "Developer Preview" al que se le instaló manualmente la versión beta más actualizada a fecha de Diciembre del 2014, la Boot2Gecko 2.2.0.0-prerelease. Al ser una versión en constante mejora, está conectada con el canal de actualización "nightly-geeksphone" de la propia fundación, que recibe los cambios commiteados y añadidos en el repositorio Git correspondiente. Por ello, a diario se recibe en el dispositivo una notificación de una actualización via OTA[4] del sistema operativo móvil que obliga a reiniciar el dispositivo y analizar los cambios que se han introducido.

Esta plataforma está en fase de progreso y mejora continua, sin llegar aún a la fase de maduración deseada debido al poco tiempo de su existencia, por lo que en cada nueva versión es interesante indagar que nuevas aportaciones se han introducido para seguir conformando un ecosistema más completo y robusto. Esta labor es llevada a cabo por desarrolladores de Mozilla Foundation con el apoyo de otras empresas y una gran comunidad de voluntarios y desarrolladores de todo el

3 Mozilla Developer Network
4 *On The Air,* sin necesidad de conectar el terminal en ningún ordenador para descargar e instalar esa nueva versión

mundo que, sin ánimo de lucro, intentan aportar sus conocimientos por un bien de la comunidad.

La necesidad de encontrar la información más actualizada posible acerca del funcionamiento de la radio FM en Firefox OS, de como está implementada la WebFM API y de las futuras mejoras que se están desarrollando, le ha llevado al desarrollador a interesarse por como funcionan los grupos de trabajo como el de los desarrollados del proyecto FxOS. En las páginas citadas en la introducción de este punto, se puede encontrar la documentación sobre este sistema operativo móvil: con una gran descripción sobre cada WebAPI que se utiliza, ejemplos descriptivos de cada una que facilitan la programación y la resolución de dudas, así como de enlaces a más páginas web con la intención de ofrecer al usuario toda la documentación posible.

Todo esto no era suficiente, por lo que al desarrollador le faltaba ahondar en las nuevas mejoras que estaban por introducir, para saber si la funcionalidad que iba a desarrollar, una simulación del sistema RDS para la automatización de la búsqueda de frecuencias de radio FM en función de la posición, podía verse simplificada y mejorada gracias a algunas de las novedades introducidas, como el soporte para la señal RDS que reciben los receptores de radio. Mucha de esta información se puede encontrar en la Wiki de Mozilla [7] en el apartado de B2G [8]. En el apartado Roadmap[5], se encuentra información sobre características ya completadas, las que están atrasadas o las próximas en incluir en la nueva versión estable 2.2 de FxOS, así como datos de las características ya lanzadas y completadas en las versiones anteriores, como la 2.1. Resulta muy interesante navegar a través de este sitio web y las subpáginas que están enlazadas en él, ya que se puede ver como están repartidas las tareas y ver los registros que dejan los desarrolladores sobre los avances [9] y lo mismo con los errores que se han ido reportando individualmente [10] o en listado de errores [11], entre otras. Este es un ejemplo encontrado rastreando las páginas comentadas, en algún hilo sobre nuevas características en cuando al diseño de la app radio FM [12].

"*Decir que iOS y Android son los dos sistemas operativos móviles más utilizados a nivel mundial no debería ser una sorpresa para nadie*", según publica Javier Merchán en

Ilustración 4: Firefox OS vs Android e iOS [Imagen 2]

5 *Hoja de ruta*

la entrada en Celularis.com [13]. Tanto el SOM de Google Inc. Como el de Apple Inc. son los más utilizados, pero no los únicos para nuestros smartphones. En menor medida coexisten Windows Phone de Microsoft, Blackberry OS de Blackberry y otras alternativas como Firefox OS o Ubuntu Touch de Ubuntu Foundation. Algunos profesionales piensan que quizá la aparición de la alternativa del zorro fue tardía e innecesaria [14], pero la propia Mozilla asegura que su nicho de mercado se centra en los paises emergentes donde el acceso a un terminal de un precio reducido y con prestaciones aceptables es posible, y no a los grandes smartphones de las mayores marcas tecnológicas a nivel mundial. Concretamente, Mozilla espera consolidarse en Centroamerica (El Salvador, Panamá, Nicaragua, Guatemala, México), Sudamerica (Venezuela, Uruguay, Colombia, Costa Rica, Ecuador y Argentina), Europa (Alemania, España, Hungría, República Checa, Grecia, Polonia, Macedonia,etc) y Asia (India y Japón), e incluso en un futuro adentrarse en el continente africano, un territorio con escasos recursos donde los terminales que Mozilla quiere popularizar encajan a la perfección [15].

La Fundación Mozilla cree que la web es el futuro y por eso piensa en poder sacar el máximo provecho posible al sistema operativo Firefox OS. A pesar de que en un principio fue ideado para utilizarse en teléfonos móviles, su versatilidad da una posibilidad enorme para adaptarlo a las demás plataformas con el fin de crear un ecosistema sólido y una experiencia de usuario formidable.

Ilustración 5: Uno de los lemas de la fundación en el CES 2015 [Imagen 3]

Se estaría hablando de llevar el sistema operativo del zorro al Internet de las cosas, a las tabletas, los relojes inteligentes, los televisores, los coches, etc. Una gran oportunidad de crecimiento en número de usuarios y entornos que reportaría un gran feedback para la mejora del desarrollo de los productos en función del gusto y valoraciones de los usuarios. Como apunte interesante, a día de hoy, ya existe una alternativa a Google Chromecast [16] para el sistema operativo de Mozilla, llamado MatchStick [17] que salió adelante gracias a la colaboración de gente anónima en un proyecto de KickStarter [18]. Es un dispositivo que se conecta al puerto HDMI del televisor y con el que se podrá interactuar entre el terminal y la televisión, sin tener que adquirir una de estas teles que lleven integrada de serie el sistema operativo FxOS.

Mozilla ha creado Firefox OS con el propósito de democratizar el mercado de los smartphones y parece que esa filosofía seguirá siendo la misma a medida que el sistema operativo sea llevado a nuevas plataformas. Como es el caso de los relojes inteligentes, con el que la Fundación busca abrir posibilidades y que los usuarios no estén limitados a dispositivos con Android Wear o los futuros Apple Watch. Aparentemente, el camino no será fácil [19].

Dispositivo móvil

Para este proyecto, el departamento de Lenguajes y Sistemas Informáticos le proporciona al desarrollador un terminal donde poder hacer pruebas. Es un modelo para desarrolladores, Geeksphone Peak "Developer Preview" al que se le instala manualmente la versión beta más actualizada, Boot2Gecko 2.2.0.0-prerelease. Al ser una versión en constante mejora, está conectada con el canal de actualización "nightly-geeksphone", que recibe los cambios commiteados y añadidos en su repositorio Git. Por ello, a menudo se recibe en el dispositivo una notificación de una actualización vía OTA[6] del sistema operativo que obliga a reiniciar el dispositivo y ver los cambios que se han introducido.

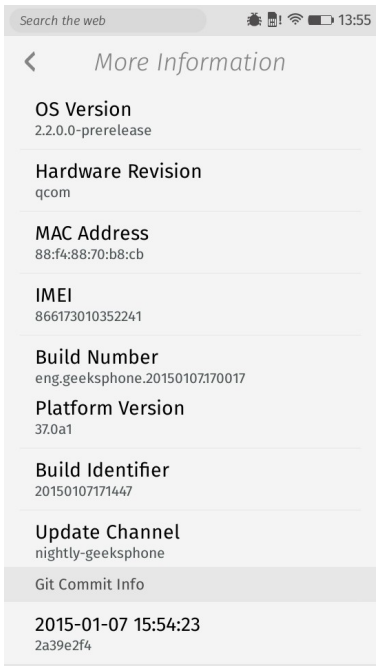


Ilustración 6: Captura de pantalla del menú Ajustes – Informacíon del Sistema - Más Información.

Sistema Operativo

El sistema operativo desde el que se trabaja durante toda la elaboración del TFG es Windows 8.1 Pro 64 bits.

6 *On The Air,* sin necesidad de conectar el terminal en ningún ordenador para descargar e instalar esa nueva versión

Navegadores de Internet

Durante los meses que se dedican al desarrollo del trabajo de fin de grado, se prueban diferentes navegadores en función de las necesidades y de las novedades y mejoras que van apareciendo. En un primer momento, se cuenta con el navegador Mozilla Firefox (versión 32) que incorpora un menú para desarrolladores (Ctrl+Shift+I) con una serie de herramientas para analizar el comportamiento de la web, los eventos JavaScript, el código HTML y los estilos CSS; prácticamente similar al funcionamiento del completo add-on Firebug for Firefox.

Se utiliza también el navegador Google Chrome (v.35) y su herramienta de desarrolladores (Ctrl+Mayus+I) para la comprobación del contenido almacenado, abriendo la pestaña *Resources*. En comparación, están un paso por delante de la competencia.

Editores de código

Acerca de las herramientas de edición de código, hay varias para mencionar ya que en un principio se intenta desarrollar con las conocidas y que ya se han trabajado. Entre ellas, citar Eclipse (v. 4), NetBeans (v. 7.4), Notepad++ (v.

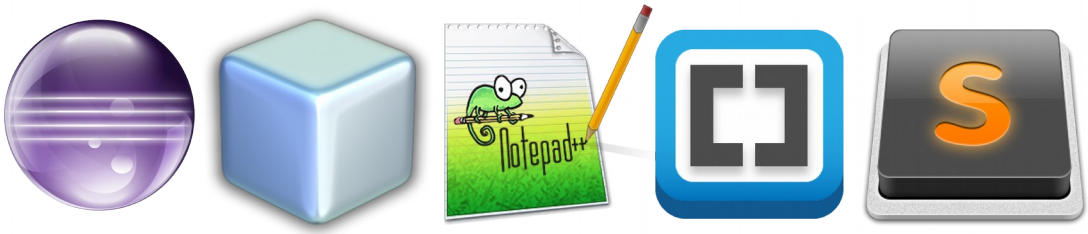


Ilustración 7. Eclipse, NetBeans, Notepad++,Brackets y Sublime Text. (Izq a Drch)

6.5.2) o Brackets (v.1.0). Las dos primeras son herramientas muy completas, un entorno de trabajo estable y fiable; la tercera muy útil para hacer cambios rápidos en código ya que reconoce todo tipo de lenguajes y las dos últimas son las más modernas y actualizadas, con refresco automático de las páginas web cuando se realiza algún cambio en el código, entre otras características.

Finalmente, el desarrollador decide utilizar la alternativa más ligera y sencilla de utilizar que recomiendan en gran parte de los lugares y cursos en los que se busca información, Sublime Text 2. Entre todos los pluggins existentes, recomendables [20] y casi necesarios, se instalan estos: Package Control, SideBarEnhancements, Alignment, BracketHighlighter, ColorPicker, Emmet, HTML5, jQuery y LESS.

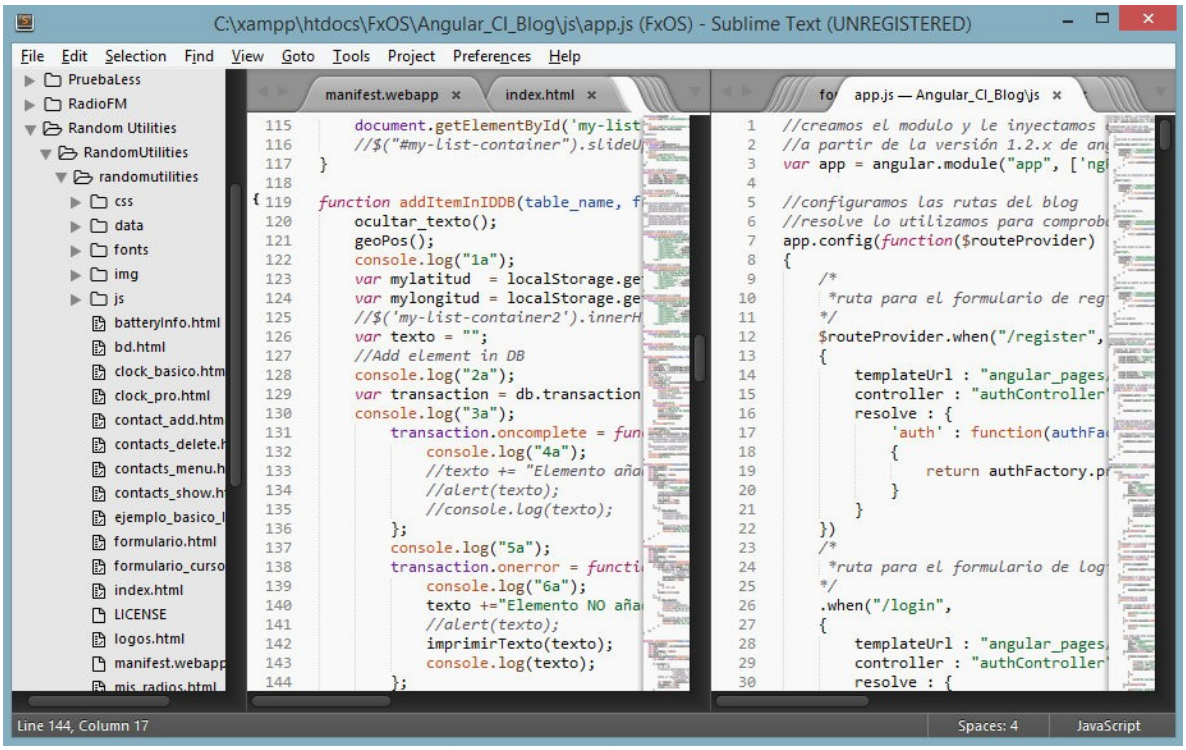


Ilustración 8. Captura de pantalla (adelante CP) de Sublime Text con la pantalla dividida en dos pestañas y barra lateral

Ofimática

El paquete ofimático que se utiliza durante la elaboración del trabajo de fin de grado es LibreOffice [21] (v.4.2.6). Es una poderosa suite de oficina de código abierto, con unas herramientas estables y completas, comparable al paquete de ofimática quizá más conocido y utilizado Microsoft Office.

De todas las herramientas que lo componen, se utilizan *Documento de Texto* y *Presentación*. El primero para realizar todos los documentos escritos que se deben presentar, tanto la propuesta inicial como el documento final. El segundo para la elaboración de la presentación final mostrada en la defensa del proyecto.

Xampp - PhpMyAdmin – Servidor apache

XAMPP es un servidor web multiplataforma formado por un servidor HTTP Apache, base de datos MySQL y los intérpretes para scripts de PHP y Perl, liberado bajo la licencia GNU. En este caso, Apache es el servidor web utilizado para este proyecto y MySQL la base de datos, un software muy bien documentado y que tiene la ventaja de ser software libre. PhpMyAdmin es una

herramienta escrita en PHP para manejar la administración de MySQL a través de páginas web creando con ella el servidor. En este caso, se usará tanto para la máquina local como para gestionar el contenido de la base de datos en el servidor. Actualmente, puede crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos y está disponible en más de 60 idiomas. Se encuentra disponible bajo la licencia GPL.

Otras herramientas utilizadas

HTML: Siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). El lenguaje HTML es un estándar reconocido en todo el mundo y cuyas normas define un organismo sin ánimo de lucro llamado World Wide Web Consortium, más conocido como W3C. Como se trata de un estándar reconocido por todas las empresas relacionadas con el mundo de Internet, una misma página HTML se visualiza de forma muy similar en cualquier navegador de cualquier sistema operativo. El propio W3C define el lenguaje HTML como "un lenguaje reconocido universalmente y que permite publicar información de forma global". Por convención, los archivos de formato HTML usan la extensión .htm o .html.

CSS: Las hojas de estilo en cascada (Cascading Style Sheets, CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML. El W3C es el encargado de formular la especificación de las hojas de estilo que servirá de estándar para los agentes de usuario o navegadores. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación. La información de estilo puede ser adjuntada tanto como un documento separado o en el mismo documento HTML. En este último podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo "style".

Las ventajas de utilizar CSS (u otro lenguaje de estilo) son: el control centralizado de la presentación de un sitio web completo, con lo que se agiliza de forma considerable la actualización del mismo; los navegadores permiten a

los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web remoto, aumentando considerablemente la accesibilidad; una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre y que un documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño, entre otras.

JavaScript: JavaScript es un lenguaje interpretado utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java. Sin embargo, al contrario que Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de herencia. Es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. Todos los navegadores interpretan el código JavaScript integrado dentro de las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM (Modelo de Objetos del Documento). JS se ejecuta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.

AJAX: acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

PHP: Es un lenguaje interpretado de propósito general ampliamente usado, diseñado especialmente para desarrollo web y que puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. Es uno de los módulos Apache más populares entre las computadoras que utilizan Apache como servidor web. PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como UNIX (y de ese tipo, como Linux o Mac OS X) y Windows [22].

Servidor EHU

Simultáneamente al servidor local, se trabaja también teniendo acceso a un servidor remoto de la propia universidad (galan.ehu.es). Para subir los archivos

desde la propia máquina al servidor a través del protocolo FTP, se utiliza FileZilla, un software libre y de código abierto distribuido bajo los términos de la licencia pública general GNU. Se utiliza para realizar transferencias de archivos entre dos máquinas, generalmente entre cliente y servidor. También está albergada la base de datos del proyecto. Se menciona también en el apartado arquitectura.

Edición Multimedia

Hay varios elementos que son necesarios crear con programas de edición multimedia. Tanto los logos y como los botones se crean con la intención de respetar las especificaciones de "Responsive View Design", haciendo que las imágenes se puedan vectorizar para lograr una mayor calidad de los elementos que se representarán en la aplicación o para que por lo menos no la pierdan, a la hora de utilizar la aplicación en varios terminales con diferente tamaño de pantalla y resolución.

Para ello, se utiliza un programa que trabaja con vectores que se llama Inkscape [23] (v. 0.48.4). Es un editor de gráficos vectoriales en formato SVG, gratuito, libre y multiplataforma, está disponible bajo la licencia pública general de GNU. Se podría comparar al Adobe Illustrator.

Se utiliza también una herramienta como GIMP2 [24] (GNU Image Manipulation Program), un programa de edición de imágenes digitales en forma de mapa de bits, tanto dibujos como fotografías. Es un programa libre y gratuito que forma parte del proyecto GNU y está disponible bajo la Licencia pública general de GNU y GNU Lesser General Public License disponible para diferentes sistemas operativos, como Linux y Windows entre otros.

Para realizar los diagramas que aparecen en el trabajo de fin de grado, se utilizan programas adaptados a lo que se requiere. En el apartado 4. Captura de requisitos se necesita realizar un modelo de casos de uso [Ver Pag.47] con su jerarquía de actores [Ver Pag.48] y un modelo de dominio [Ver Pag.53]. Para ello se utiliza el programa de software libre ArgoUML (v.0.34) [25]. En el apartado 2.3 Planificación Temporal, para realizar las capturas de pantalla que se adjuntan, se utiliza el programa de software libre Gantt Project (v. 2.7) [26] al igual que ocurre para hacer los diagramas con el programa Visual Paradigm 12.0 en la versión de prueba gratuita.

También se utilizan algunos de estos programas de edición multimedia para manipular digitalmente las ilustraciones que se añaden en este documento, para la edición de las capturas de pantalla que se necesitan incluir en varios

apartados.

Almacenamiento en la nube

Para evitar problemas y situaciones indeseadas en caso de pérdida de información relevante para el desarrollador, desde los primeros momentos se han hecho copias de seguridad en un soporte externo al ordenador portátil desde el que se trabaja. Periódicamente, se ha hecho un volcado de las carpetas que contienen todos los archivos del trabajo, tanto a *Google Drive* como *Dropbox*, para evitar perder información y ganar accesibilidad a través de Internet para la consulta de los archivos.

Otra herramienta que se han tenido en cuenta son GitHub y Bitbucket. Excelentes herramientas para el control de versiones de nuestros archivos en desarrollo, pudiendo distinguir las líneas de código que se alteran cada vez, pudiendo dejar un comentario identificativo al guardar una serie de cambios. Muy interesante la posibilidad de tener alojado en Internet (de modo público o no según si no se es usuario premium de la plataforma) el código para poder compartirlo con otros desarrolladores o poder recibir una respuesta de usuarios que te ayudan a mejorar el código

Marco de trabajo utilizado

La elección del marco de trabajo (comúnmente llamado framework por su traducción en inglés) a utilizar no es sencilla.

Entre la gran cantidad de estos entornos disponibles los hay de diferentes tipos, tanto gratuitos como de pago, unos pensados para páginas web y otros que se adaptan a la plataforma donde se vayan a utilizar, que introducen el "Responsive web design"[7] como pieza fundamental de sus características. Entre las que pueden servir al desarrollador destacar Bootstrap, JqueryMobile, Ionic, Kendo UI, Sencha Touch, PhoneGap, LungoJS, TakoJS...

De todas las herramientas enumeradas, el desarrollador elige TakoJS para llevar a cabo el desarrollo de su trabajo, por la sencillez de adaptación, la sencilla documentación y la claridad de desarrollo.

TakoJS es un framework HTML5 de código abierto diseñado para hacer sencilla la creación de impresionantes aplicaciones móviles, como reza en su página web www.takojs.com.

Permite desarrollar aplicaciones y usarlas sin importar el dispositivo, al ser

7 "Diseño web responsivo", o dicho generalmente un diseño que se adapta a la pantalla del dispositivo donde se vaya a utilizar, algo que hoy en día parece ser el futuro inmediato, una necesidad en las páginas web.

aplicaciones web (webapps). Su motor es HTML5, creado aprovechando la nueva semántica y características de este lenguaje, como son las etiquetas, APIs y hojas de estilo. Aporta un alto rendimiento al haber sido optimizado para trabajar tanto en equipos antiguos como en los más nuevos.

Permite las capacidades táctiles de los dispositivos de hoy en día, soportando la pulsación simple en la pantalla, la doble pulsación, el arrastre, etc. Parece interesante destacar que para mejorar la experiencia de Tako, utiliza internamente Zepto.js, hammer.js y webdb.

Ilustración 9: Logo del framework TakoJS

Otro de sus puntos fuertes es su gran abanico de entornos donde se puede disfrutar. Entre los navegadores, funciona en Mozilla Firefox, Google Chorme, Internet Explorer y Safari. En cuanto a dispositivos móviles, se puede utilizar en las plataformas Firefox OS de Mozilla, Android de Google a partir de la versión 2.3 en adelante, iOS7 de Apple, Windows Phone y Blackberry.

2.6. Gestión de riesgos

Planificación Incorrecta

Descripción: Al ser una de las primeras planificaciones que se realizan para un proyecto de esta magnitud y con estas características, seguramente que no sea del todo correcta la planificación. Por esa razón, el tiempo necesario será bastante mayor del esperado y la fecha de entrega inicial prevista se quede corta.

Prevención: La manera de prevenir que ocurra esto será hacer una planificación con amplios plazos y dejando margen de tiempos para cada tarea.

Plan de contingencia: Si a pesar de esto, no se logra que la planificación fuera correcta y se produjera este riesgo, habría que modificar la planificación original aumentando las horas de trabajo hasta el momento de la entrega.

Probabilidad: Es muy probable.

Impacto: El impacto principal consistirá en aumentar las horas de trabajo a

medida que se va acercando la fecha esperada de finalización para cumplir el plazo previsto. La necesidad de finalizar para la fecha prevista hace que uno mismo haga un esfuerzo mayor para lograr cumplir los plazos.

Sufrir fallos en el ordenador de trabajo

Descripción: En el desarrollo de la aplicación Web es muy importante el mantenimiento del material de trabajo, que en nuestro caso sería el propio ordenador. Es probable que se produzca un fallo de corriente o se estropee el ordenador en el que se hace la mayor parte del trabajo o en uno de los discos duros externos sobre el que se esté trabajando, etc; ocasionando una importante pérdida de información que puede que no se tenga una copia.

Prevención: Hoy en día, este es un riesgo que tiene una posible solución aunque a menudo no se realice por falta de cuidado. Bastaría con tener copias de seguridad en otro ordenador, otra partición dentro del mismo disco, un disco duro externo, un NAS conectado a la red del equipo con el que se trabaja o en plataformas online como *Google Drive*, *Box*, *Dropbox, GitHub...* Estas últimas son muy útiles por el hecho de que el contenido allí albergado es accesible desde cualquier dispositivo conectado a Internet. Además se deberá proteger el ordenador con un antivirus eficaz para evitar que se produzca un fallo por un virus malicioso y realizar un mantenimiento ocasional del sistema.

Plan de contingencia: Si se produjera este riesgo, se podría seguir trabajando desde otro ordenador. De este modo, el equipo no se quedaría paralizado y sin poder seguir trabajando.

Probabilidad: Es probable.

Impacto: Sería importante el impacto, pero al haber tenido un gran cuidado a la hora de almacenar los datos y su debida copia de seguridad, se reduce su grado de peligro. El hecho de trabajar con diferentes equipos ayuda a que esos datos estén seguros.

Imposibilidad de cumplir plazos establecidos en la planificación debido a imprevistos

Descripción: Puede que surjan imprevistos o responsabilidades con las que hay cumplir. Por ejemplo, unos días de baja por enfermedad, necesidad de asistir a algún examen o curso para el que previamente se ha necesitado estudiar, realizar una reunión con el tutor no prevista pero necesaria para el desarrollo del proyecto... Esto imposibilita dedicar las horas previstas a realizar

el trabajo que se pretendía.

Prevención: La prevención de este riesgo es igual que la que se ha hablado en el punto sobre la planificación incorrecta, consistiendo en dejar márgenes de tiempo para cada tarea y de ese modo lograr recuperar el tiempo "perdido" utilizando esas horas que estaban planificadas como adicionales.

Plan de contingencia: Se debería revisar la planificación y recuperar las horas perdidas en los siguientes días posibles.

Probabilidad: Es muy probable.

Impacto: Este riesgo provocaría realizar un esfuerzo extra en los días posteriores al imprevisto hasta recuperar el tiempo "perdido".

Desaparición del sistema operativo móvil por falta de apoyos y soporte

Descripción: En algún momento puntual de todo el tiempo que se lleva teniendo el proyecto en mente y el tiempo que lleva en desarrollo, se han leído noticias y comentarios bastante pesimistas en la red acerca del futuro de este sistema operativo móvil. Estas informaciones daban muy pocas probabilidades de continuidad al proyecto de Mozilla, apostando por que la fundación dejaría a un lado Firefox OS y centrarían sus esfuerzos en su navegador Firefox. Por suerte, parecen ser un simple bulo y más cuando se presentan por parte de la compañía nuevas usos de este sistema operativo móvil, como que Panasonic planea sacar al mercado una serie de televisores con sistema operativo Firefox OS a lo largo del año 2015 [27].

Prevención: La prevención de este riesgo es simple, y basta con la necesidad de estar constantemente informado de las novedades tecnológicas que afectan a esta plataforma novedosa y aún en fase de crecimiento y mejora.

Plan de contingencia: Mantener un contacto con las personas encargadas en el proyecto, seguir los comentarios en redes sociales de los propios desarrolladores para ver sus avances.

Probabilidad: Probabilidad mínima.

Impacto: Este riesgo provocaría el replanteamiento completo del proyecto llevado a cabo al no poder concluirlo del modo inicialmente ideado.

Imposibilidad de acceso a las APIs del sistema operativo que gestionan la radio FM por privatización de las mismas.

Descripción: Relacionado con punto anterior, el futuro incierto de este novedoso sistema operativo móvil pone en alerta de modo indirecto a los

desarrolladores. Se pudiera dar el caso de que la fundación Mozilla privatizara las APIs públicas que acceden al hardware del dispositivo, como la de radio FM por ejemplo. No está estandarizada por la W3C y eso pone en duda su futuro.

Prevención: La prevención de este riesgo es simple, y basta con la necesidad de estar constantemente informado de las novedades tecnológicas que afectan a esta plataforma novedosa y aún en fase de crecimiento y mejora.

Plan de contingencia: Mantener un contacto con las personas encargadas en el proyecto, seguir los comentarios en redes sociales y grupos de desarrollo de los propios desarrollados para ver sus avances.

Probabilidad: Probabilidad mínima.

Impacto: Este riesgo provocaría el replanteamiento completo del proyecto llevado a cabo, al no poder concluirlo del modo inicialmente ideado y esperado.

Funcionamiento incorrecto del terminal cedido por el departamento, tanto en el software como en el hardware, al ser un dispositivo para desarrolladores.

Descripción: El terminal prestado al desarrollador, Geeksphone Peak Developer Preview, como su propio nombre deja entrever, es un terminal en fase de desarrollo tanto sus componentes como la versión de Firefox OS que inicialmente lleva instalada. La promesa de la compañía es una actualización periódica del software que corre en el dispositivo.

Prevención: La prevención de este riesgo conlleva la necesidad de estar constantemente informado de las novedades tecnológicas que afectan al sistema operativo móvil y a la compañía que desarrolla el terminal. Instalar una versión estable del SOM ayudará a que su rendimiento no disminuya a pesar de no contar con las innovaciones más actualizadas.

Plan de contingencia: Mantener un contacto con las personas encargadas en el proyecto y de la compañía, seguir los comentarios en redes sociales y grupos de desarrollo de los propios desarrollados para ver sus avances.

Probabilidad: Probabilidad media.

Impacto: Este riesgo provocaría la necesidad de que el desarrollador instale manualmente por su cuenta versiones del SOM, y que se encargue de sus actualizaciones.

Problemas personales

Descripción: Es el hecho de sufrir algún problema personal durante la elaboración del proyecto.

Prevención: No existe una determinada prevención ya que depende del tipo de problema que se tenga.

Plan de contingencia: Intentar dedicar más horas a realizar las tareas retrasadas.

Probabilidad: Probabilidad media.

Impacto: Este riesgo provocaría el retraso en las fechas establecidas para llevar a cabo las tareas.

2.7. Evaluación económica

En este punto, se va a realizar la evaluación económica del proyecto, pensando en la inversión realizada y como recuperarla, así como los costes totales de llevarlo a cabo. Estos son los apartados en los que se pueden dividir:

Personal:

Para la elaboración del trabajo, se ha necesitado el trabajo de un desarrollador. Teniendo en cuenta que un programador cobra 25 euros/hora y que el proyecto dura 546 horas, el coste total en cuanto al personal suma 13650 euros.

Hardware:

Ordenador portátil: valorado en 600 euros, con una vida media de 5 años, haciendo un uso prácticamente total del ordenador durante la elaboración del trabajo de fin de grado, tanto para el desarrollo de código como de la documentación. Se estima que en un porcentaje del 60%.

- Amortización anual:

  $600\ {}^{€}/\ 5\ {}_{\text{años}} = 120\ {}^{€}/_{\text{año}}$.

- Gastos del portátil:

  $((120\ {}^{€}/_{\text{año}} \cdot 13\ {}_{\text{semanas}}) \cdot 60\ \%) / 52\ {}^{\text{semanas}}/_{\text{año}} = 18\ €$.

Geeksphone Peak (smartphone): es un terminal prestado por la universidad al desarrollador de forma gratuita, pero que sino se hubiera tenido que adquirir, por lo que se gestionará como gasto en las cuentas totales. Su valoración es de 180 euros, con una vida media de 2 años y con uso casi único para el desarrollo y probaturas de las aplicaciones desarrolladas. Se estima que en un porcentaje del 33%.

- Amortización anual:

  $180\ {}^{€}/_{2\ años} = 90\ {}^{€}/_{año}$.

- Gastos del móvil:

  $((90{}^{€}/_{año} \cdot 13\ semanas) \cdot 33\%) / 52\ {}^{semanas}/_{año} = 7{,}425\ €$.

Software:

Licencias de pago:

Licencia Windows 8/8.1/10, incluida en el precio del portátil.

Licencias gratuitas:

Firefox Developer Edition Browser, LibreOffice, NetBeans, Sublime Text2, Gimp 2, Inkscape, GanttProject, ArgoUML, etc.

Otros:

Adquisición de varios auriculares con sistema de antena incorporado. Con estos cascos, al conectarlos al dispositivo, se podrá ejecutar la aplicación de radio FM: 3 unidades · 12 €/unidad = 36 euros.

Gastos	
Personal	13.650 €
Hardware	25,425 €
Software	0 €
Otros	36 €
Total: 13.711,425 €	

Tabla 3: Gastos evaluación económica

Recuperación de la inversión:

Hay varias maneras de poder intentar recuperar la inversión que se ha realizado para el proyecto. El repositorio de aplicaciones de Firefox OS se llama Firefox Marketplace y en él existen aplicaciones tanto gratuitas como de pago [28]. Para subir una app al repositorio no es necesario pagar nada, simplemente estar identificado en el sistema, subirla siguiendo unos pasos concretos [29]. Eso si, el desarrollador sólo recibe el 70% del importe obtenido. El resto, se lo reparten entre Mozilla, el operador móvil y el proveedor del pago [30].

En el caso del desarrollador de FM Radio RDS, para intentar compensar todos los esfuerzos realizados obteniendo una recompensa económica, realiza una serie de estudios:

1. Probabilísticamente hablando, una aplicación menos ventas va a tener cuanto más cara sea, por lo que se necesitará ver si la diferencia de precio compensa. Si el precio de la aplicación fuera de 1 euro, el número mínimo de compradores para obtener beneficios debería ser el siguiente:

   Beneficios: ((x · (1 - 0,3) €) - 13711,425 €) => 19588 veces descargado.

2. Si el desarrollador decide ponerlo a la venta por 5€ debido al gran número de horas que le ha llevado desarrollarla, el número mínimo de compradores para obtener beneficios es el siguiente:

   Beneficios: ((x · 5 · (1 - 0,3) €) - 13711,425 €) => 3918 veces descargado.

A pesar de la útil funcionalidad que proporciona dicha app al ecosistema de Mozilla, el escaso número de usuarios reales a día de hoy hace que el desarrollador no vea factible obtener beneficios con la compra-venta de su aplicación en la plataforma. Su mayor esperanza está puesta en que la propia fundación Mozilla integre FM Radio RDS en el sistema operativo móvil de serie en futuras versiones, ya sea con la compra de la aplicación o con el ofrecimiento de un puesto de trabajo dentro de la compañía.

3. Antecedentes

3.1. Historia y situación actual

Históricamente, la radio ha sido un medio de comunicación a nivel mundial muy importante desde su aparición.

Según varios autores, está datado que en la Nochebuena de 1906, Reginald Aubrey Fessenden transmitió desde Brant Rock Station (Massachusetts) la primera transmisión radiofónica en la historia. Buques en el mar pudieron oír una radiodifusión donde el nombrado Fessenden tocaba al violín la canción O Holy Night y leyendo un pasaje de la Biblia.

Las primeras transmisiones para entretenimiento comenzaron en 1920 en Argentina, desde la azotea del Teatro Coliseo de Buenos Aires, cuando la Sociedad Radio Argentina transmitió la ópera Parsifal de Richard Wagner, comenzando así con la programación regular de la primera emisora de radiodifusión en el mundo. Hasta entonces, se habían realizado pruebas experimentales en varios lugares del mundo con poco margen de tiempo. Los horarios de dicha programación eran breves, desde el atardecer hasta la medianoche, y muchas veces entrecortados. Para 1925 ya había una veintena de estaciones de radio por todo el país albiceleste, una muestra de la expansión mundial de este medio de comunicación para aquella época. La primera emisora de carácter regular e informativo fue la estación 8MK de Detroit (Estados Unidos) perteneciente al diario The Detroit News que comenzó a operar el 20 de agosto de 1920 en la frecuencia de 1500 kHz.

Ya en Europa, concretamente en el Reino Unido, en el año 1922, la estación de Chelmsford perteneciente a la Marconi Wireless, emitía dos programas diarios de diferente temática, música e información general. En noviembre de ese mismo año se fundó en Londres la British Broadcasting Corporation (BBC) que monopolizó las ondas inglesas.

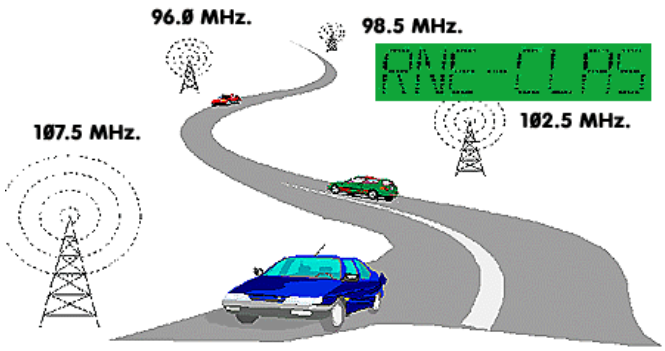


Ilustración 10: Simulación del sistema RDS en los coches [Imagen 4]

Hablando del territorio nacional, en 1906 comienza una fase de experimentación que se da por concluida en septiembre de 1923 con la inauguración de la primera emisora, Radio Ibérica, producto de la fusión de la Compañía Ibérica de

Telecomunicación y la Sociedad de Radiotelefonía Española. Emitían una programación fija, que incluían conferencias, anuncios de receptores fabricados por la empresa, música de gramófono, conciertos del Teatro Real, recitales de poesía, el sorteo de la Lotería de Navidad, etc.

En los años posteriores y gracias a las nuevas reales ordenes que iban concediéndose, aparecen las primeras emisoras de iniciativa privada como EAJ-1 Radio Barcelona posteriormente Sociedad Española de Radiodifusión (SER), Radio España de Madrid luego Onda Cero, EAJ-4 Estación Castilla, etc.; y de iniciativa pública como Radio Nacional de España (RNE). En noviembre de 1952 se aprobó una ley donde se dividen las emisoras en nacionales, comarcales/autonómicas y locales, como se conocen en la actualidad en el territorio español [31]. Es digno de destacar la estación pionera, la emisora decana del espacio radiofónico español. La nomenclatura EAJ-1 hace referencia a los códigos de los radio-aficionados, auténticos artífices del nacimiento de la radio: la letra E por España, AJ porque designa a las estaciones de Telegrafía sin Hilos y el número 1 por ser la primera [32].

En julio de 1999 se aprueba el Plan Técnico Nacional de la Radiodifusión Sonora Digital Terrenal, que sentaría las bases para la incorporación en nuestro país del Digital Audio Broadcasting (DAB). Se trata de un sistema europeo de radio digital estandarizado por el European Telecommunications Standardisation Institute (ETSI) que, entre otras cosas, supone la eliminación total de posibles interferencias, así como la recepción sin ecos de la señal. Además, la emisoras no han dudado en incorporar otra tecnología, el denominado Radio Data System (RDS). Este es un sistema que posibilita la trasmisión de una señal digital imperceptible para el oído, y que, aprovechando el ancho de banda que ofrece la Frecuencia Modulada (FM), brinda al oyente la posibilidad de visualizar en la pantalla el nombre de la emisora que está escuchando, así como algunos mensajes de texto, permite viajar oyendo el mismo programa aunque cambie la frecuencia o estar permanentemente informado de la situación del tráfico [33].

En la actualidad, hay receptores de radio en muchos dispositivos, como los transistores, automóviles, reproductores de música portables, equipos de música domésticos, televisiones a través de la señal digital terrestrelos dispositivos... Sin embargo, no todos disponen de este sistema RDS que automatiza la sintonización de la señal de una emisora concreta en función de la posición donde se este realizando esa conexión, una de las funcionalidades que quiere aportar FM Radio RDS.

No todos los teléfonos inteligentes que se disponen actualmente llevan incorporado el chip receptor de radio FM. Varía según la marca y el sistema operativo que incorporen. Los iPhone/iPod/iPad con iOS de Apple no llevan incorporado este chip, por lo que a los usuarios no les queda otra opción que usar aplicaciones que retransmitan radio digital a través de internet. Entre los dispositivos que llevan Android como sistema operativo, hay que mirar las características de cada uno para saber si incorpora o no el chip receptor de radio FM. Existen teléfonos móviles, tabletas y hasta reproductores musicales con opción a sintonizar la radio FM analógica.

"*Desgraciadamente, Android no es un sistema libre completamente, ya que la mayoría de librerías y herramientas para comunicarse con servicios internos no son libres, ni tampoco las aplicaciones, ni el firmware, ni controladores que manejan la radio, WiFi, GPS, gráficos 3D y otros*", declaraba Richard Stallman en una entrevista publicada hace unos años [34]. "*Algunos modelos de smartphones Android están diseñados para evitar que los usuarios instalen y usen software modificado*" añadía.

FxOS permite crear aplicaciones que interactuen con el hardware del dispositivo. Cualquier desarrollador interesado puede documentarse en el portal Mozilla Developer Network sobre las Web APIs [35] que se pueden utilizar para el desarrollo web utilizando JavaScript. En el caso del desarrollador del proyecto, la posibilidad de utilizar WebFM API le ha permitido llevar a cabo su idea de proyecto en esta plataforma.

Otra de las diferencias es que Firefox OS funciona con aplicaciones web y no con aplicaciones nativas. Una app nativa es una aplicación implementada en el lenguaje nativo de cada terminal (Java para Android y BlackberryOS, Objective C para iOS, C# para Windows Phone). Estas apps podrán acceder a los sensores internos del móvil para aprovecharse de funcionalidades típicas de estos dispositivos como el geoposicionamiento, brújula, cámara, etc. Una web app es una aplicación web optimizada mediante HTML5, CSS3 y JavaScript para la correcta visualización en los smartphones. Se pueden instalar estas aplicaciones en FxOS, como lo se harían en el resto de sistemas operativos móviles, consiguiendo tener una apariencia y usabilidad aparentemente nativa. Podrán acceder a los sensores internos del hardware del dispositivo para aprovecharse de sus funcionalidades, como la radio FM, la cámara, el geoposicionamiento, los sensores luminosos y de cercanía, etc. [36]

3.2. Estudio de diferentes alternativas

Las alternativas disponibles a FM Radio RDS en los smartphones hoy en día, a parte de la aplicación de serie, son inexistentes en el ámbito analógico. Hay varias aplicaciones que utilizando una conexión a Internet retransmiten la señal digital de radio que las propias emisoras cuelgan en su página web para poder escucharlo en cualquier lugar del planeta. Una vez analizado el Firefox Marketplace donde los desarrolladores exponen sus aplicaciones, se han encontrado varias apps interesantes: iVoox, Dial.Radio, World Radio Player, JamendoFM y Radio MDM, pero todas ellas utilizan la señal digital de radio, algo que FM Radio RDS quiere mejorar.

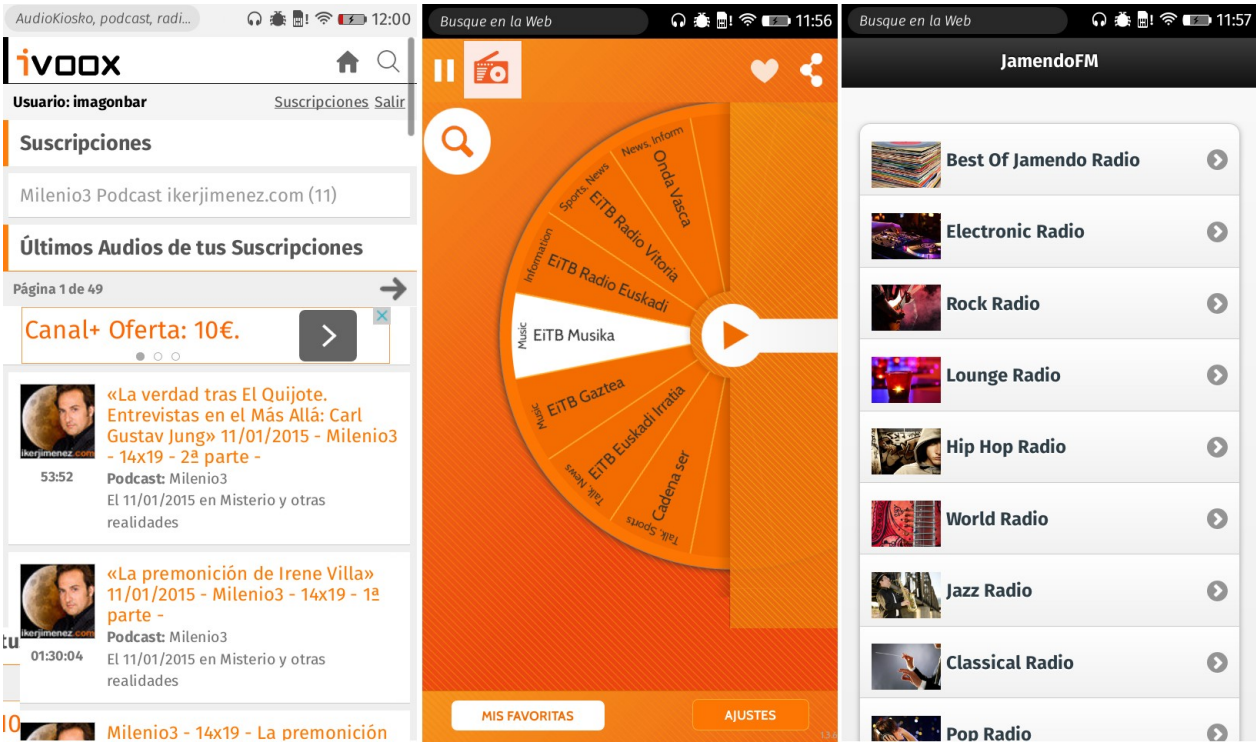


Ilustración 11: Capturas de pantalla de las apps con cierta semejanza en Firefox OS. De izq a drch: Ivoox, Dial.Radio y JamendoFM.

En otros ecosistemas como Android, se encuentran apps como iVoox, Radios de España, TuneIn Radio y Babanjo, que siguen utilizando la conexión a la red para retransmitir la señal, en vez de aprovechar la señal de radio FM que llega, que es, básicamente, lo que con FM Radio RDS se pretende destacar.

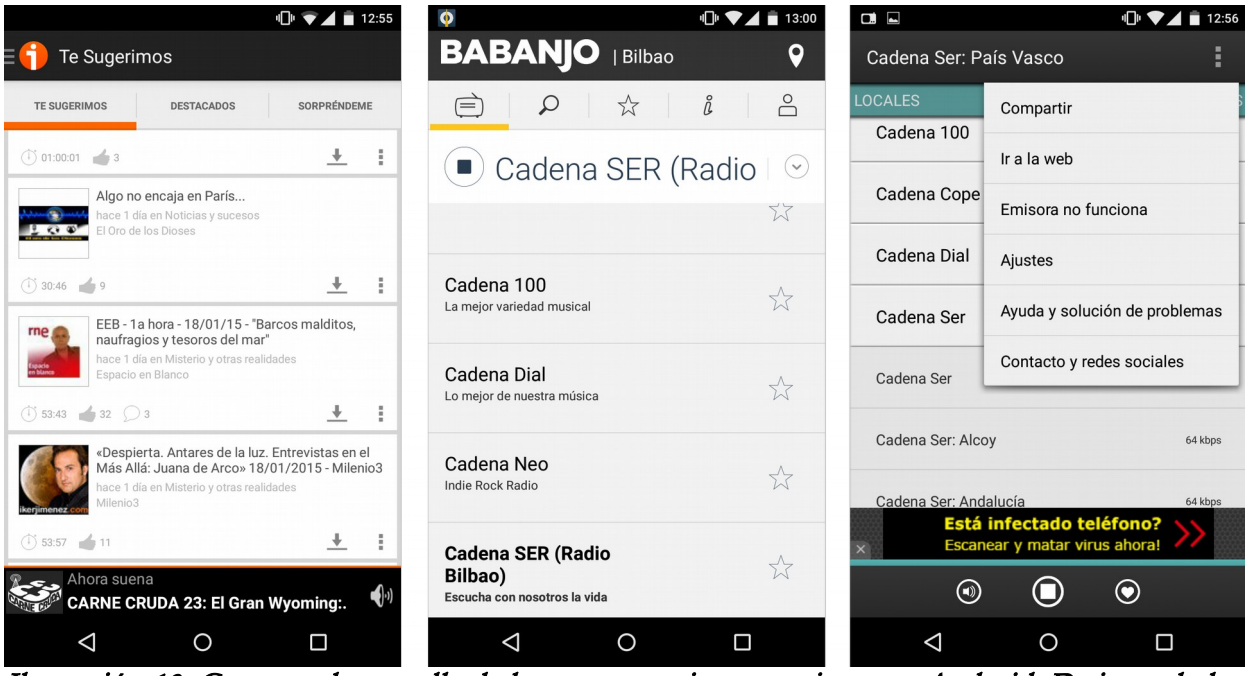


Ilustración 12: Capturas de pantalla de las apps con cierta semejanza en Android. De izq a drch: Ivoox (con la interfaz propia de Android), Babanjo y Radios de España.

4. Captura de requisitos

En este apartado, se presentan los casos de uso con la jerarquía de actores y el modelo de dominio.

4.1. Casos de uso y la jerarquía de actores.

A continuación, se muestran dos ilustraciones que representan el modelo de casos de uso y la jerarquía de actores. Posteriormente, se van a detallar ellos individualmente.

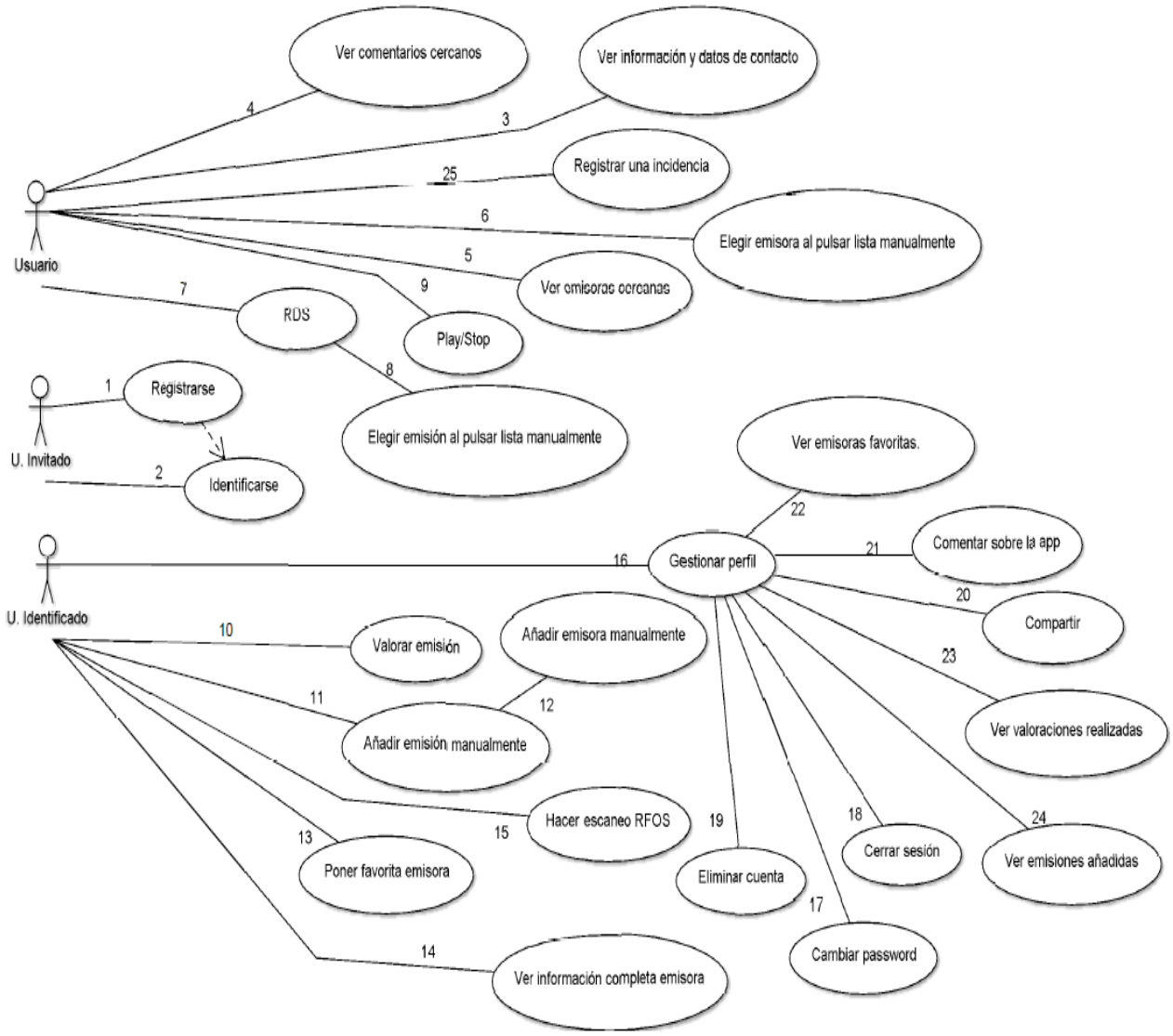


Ilustración 13: Modelo de casos de uso

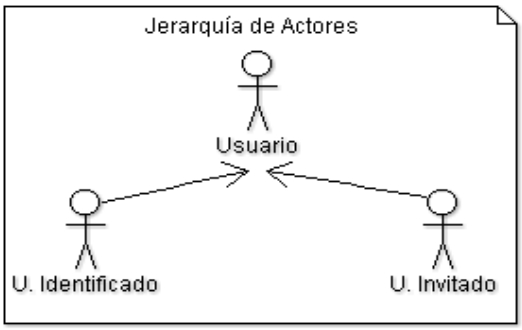


Ilustración 14: Jerarquía de actores

Jerarquía de actores

Usuario

Este actor será el que posea las funcionalidades comunes para los otros dos actores que se describen a continuación, *Usuario Identificado* y *Usuario Invitado*, que serán los actores con funcionalidades especificas según su rol.

Usuario Identificado

Este actor será el que use la aplicación habiéndose identificado. Para ello, previamente ha de haberse registrado en el sistema. Podrá acceder a todas las funcionalidades de *Usuario* más las pensadas concretamente para él, como se aprecian en el diagrama de casos de uso.

Usuario Invitado

Este actor será el que use la aplicación sin haberse identificado. Podrá acceder a todas las funcionalidades de *Usuario* más las pensadas concretamente para él, como se aprecian en el diagrama de casos de uso.

Casos de uso

1. Registrarse

Permite a *Usuario invitado* darse de alta en la aplicación, de tal modo que pueda acceder a las todas las funcionalidades al completo. Mediante la página de registro, el usuario podrá crear el usuario final con el que se identificará en la aplicación.

2. Identificarse

Permite a *Usuario invitado*, que previamente estará registrado, conectarse a la aplicación para poder acceder a las todas funcionalidades que se les permite a los

usuarios registrados.

3. Ver información y datos de contacto del desarrollador y de la app

Permite a *Usuario* ver información sobre el origen de la aplicación, datos de contacto de la misma y del desarrollador, así como enlaces a sus redes sociales (Twitter y Google Plus) y cuentas de correo. Puede acceder a estos perfiles directamente desde la aplicación.

4. Ver Comentarios de usuarios cercanos

Permite a *Usuario* ver los comentarios que los usuarios han podido escribir en sus perfiles sin necesidad de estar identificado.

5. Ver emisoras con emisiones cercanas

Permite a *Usuario* ver el listado de emisoras que tienen emisiones cercanas a su posición. Es la ventana inicial que ve *Usuario* cuando inicia la app.

6. Elegir emisora al pulsar lista manualmente

Permite a *Usuario* seleccionar cualquier elemento del listado de emisoras. La frecuencia que esté sonando se actualizará, emitiendo lo que reporte en la nueva emisora. No es del todo fiable que en esa nueva frecuencia seleccionada, la emisora que esté sintonizada sea la misma que se estaba escuchando. Depende de la calidad y veracidad de los datos que los usuarios aporten al sistema. Además, en cada posición la recepción de la señal dependerá de factores meteorológicos, como ocurre con cualquier receptor de radio FM analógica.

7. RDS

Permite a *Usuario* ver el listado de emisiones asociadas a la emisora que se está escuchando. Al pulsar el botón RDS, aparecerá en una nueva ventana un listado con las emisiones cercanas a la posición donde esté el *Usuario* de la emisora que se está escuchando.

8. Elegir emisión al pulsar lista manualmente

Permite a *Usuario* elegir un nuevo elemento que sintonizar con sólo pulsar un ítem del listado de emisiones que verá en pantalla. Este caso de uso se utilizará en varias situaciones de la aplicación. No es del todo fiable que en esa nueva frecuencia seleccionada, la emisora que esté sintonizada sea la misma que se estaba escuchando. Depende de la calidad y veracidad de los datos que los usuarios aporten al sistema. Además, en cada posición la recepción de la señal dependerá de factores meteorológicos, como ocurre con cualquier receptor de radio FM analógica.

9. Play/Stop

Permite a *Usuario* activar o desactivar la reproducción de la radio, en función de la emisión que esté seleccionada.

10. Valorar emisión

Permite a *Usuario Identificado* valorar la emisión que esté escuchando. Tendrá una escala con 5 posibles valores que indicar, desde "Muy Mala Señal" con el slider arrastrado a la parte izquierda, "Mala Señal" con el slider colocado entre la parte izquierda y el centro, "Normal Señal" con el slider colocado en el centro, "Buena Señal" con el slider colocado entre la parte derecha y el centro y "Muy Buena Señal" con el slider arrastrado a la parte derecha. El usuario, según la calidad de la recepción en el momento de puntuar, subjetivamente elegirá una de las opciones de la escala. Esta valoración servirá para las recomendaciones posteriores. Se añadirá al listado de valoraciones realizadas por cada usuario.

11. Añadir emisión manualmente

Permite a *Usuario Identificado* añadir manualmente una emisión. Si el usuario de la aplicación conoce una emisión que se sintonice en esa posición y que no está añadida, podrá incluirla pulsando la opción indicada con un signo más. Permitirá introducir la frecuencia que conoce para una emisora en la posición en la que se encuentre. El nombre de la emisora la seleccionará de un desplegable que le aparecerá en esa interfaz. Se añadirá al listado de emisiones añadidas de cada usuario.

12. Añadir emisora manualmente

Permite a *Usuario Identificado* añadir una nueva emisora que no exista en la base de datos del sistema. Se ruega que el nombre introducido se ajuste lo máximo posible al real. Se seleccionará en ese instante el alcance de la emisora y el tipo o tipos de emisora que son.

13. Poner favorita emisora

Permite a *Usuario Identificado* añadir como emisora favorita la emisora que se esté escuchando. Será un estrella que tendrá dos estados, seleccionada o no seleccionada. Se añadirá al listado de emisoras favoritas de cada usuario.

14. Ver información completa de la emisora

Permite a *Usuario Identificado* ver la información de una emisora seleccionada. Al estar identificado, la información que se le muestra está al completo, sin limitaciones.

15. Hacer escaneo RFOS

Permite a *Usuario Identificado* hacer un escaneo completo de las frecuencias encontradas en esa posición. Recorrerá en bucle el dial, mostrando uno por uno los resultados obtenidos. Se podrá poner el nombre individualmente a las emisoras encontradas. Se añadirá al listado de emisiones añadidas de cada usuario.

16. Gestionar perfil

Permite a *Usuario Identificado* ver la información de su perfil.

17. Cambiar password

Permite a *Usuario Identificado* cambiar su contraseña. Le aparecerá una ventana donde poder introducir el correo, la contraseña vieja y la nueva, para asegurarnos que el usuario es realmente el que está cambiando su clave de acceso.

18 Cerrar sesión

Permite a *Usuario Identificado* cerrar su sesión. Le redirigirá a la ventana inicial de la app.

19 Eliminar usuario

Permite a *Usuario Identificado* eliminar su cuenta. Lo que internamente haremos es poner al usuario como inactivo, con intención de no perder toda la información que ha ido generando el tiempo que ha sido usuario. Le redirigirá a la ventana inicial de la app.

20 Compartir

Permite a *Usuario Identificado* compartir un mensaje a través del correo electrónico u otro sistema que elija

21 Comentar sobre la app y ver comentarios realizados

Permite a *Usuario Identificado* hacer un comentario sobre la aplicación de modo público. Podrá publicar un comentario por día, con una restricción de 160 caracteres cada uno.

También permite a *Usuario Identificado* ver los comentarios que ha realizado.

22 Ver emisoras favoritas

Permite a *Usuario Identificado* ver las emisoras favoritas.

23 Ver valoraciones realizadas

Permite a *Usuario Identificado* ver las valoraciones que ha realizado.

24 Ver emisiones y emisoras añadidos

Permite a *Usuario Identificado* ver las emisiones y emisoras que ha añadido.

25 Registrar una incidencia

Permite a *Usuario* registrar una incidencia que ha podido haber durante el uso de la aplicación. Se facilitará el acceso a los diferentes datos de contacto.

4.2. Modelo de dominio

En este apartado, se muestra el modelo de dominio asociado a la aplicación FM Radio RDS.

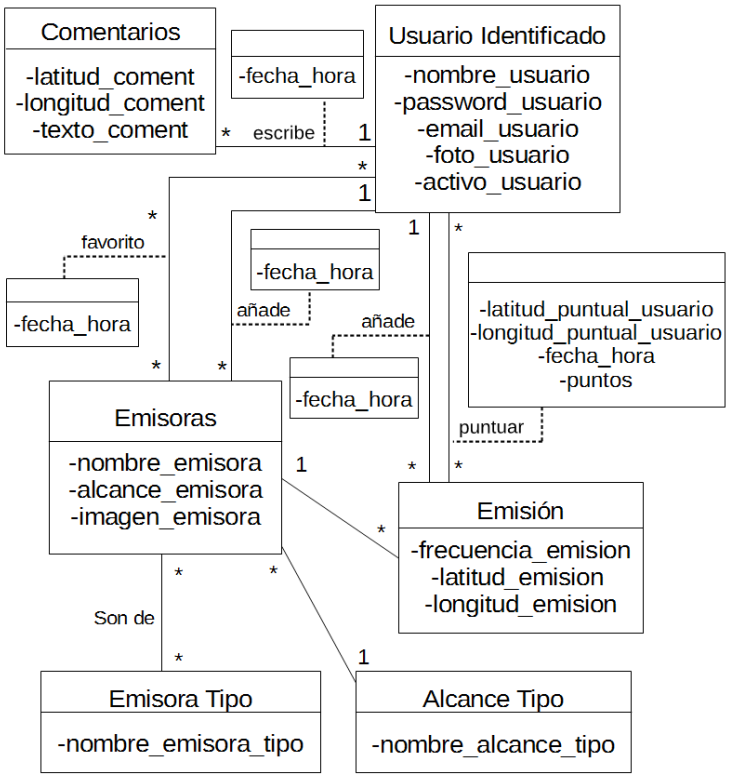


Ilustración 15: Modelo de dominio

Gracias a este diagrama, se pueden ver los tipos de datos que se almacenan en el sistema, como son las emisoras de radio y cierta información de cada una de ellas (entidad Emisora), las emisiones de una emisora en función de las posición y su frecuencia (entidad Emisión), los usuarios identificados con ciertos datos personales y propios que se van almacenando a medida que este utiliza el sistema (entidad Usuario Identificado), los comentarios que escribe según la posición y fecha (entidad Comentarios), los tipos de alcance que pueden tener las emisoras (entidad Alcance Tipo) y los tipos de emisoras que hay (entidad Emisora Tipo). De las relaciones que hay entre las entidades usuario y emisiones, aparece una asociación Puntuaciones (en el modelo de dominio aparece sin nombrar), que registrará toda la información con respecto a una puntuación.

De este modo, se establecen las relaciones entre estos tipos de elementos donde: una emisora tendrá una o varias emisiones y una emisión es propia de una emisora; un usuario puede tener como favorita a una o varias emisoras almacenando cuando se ha seleccionado así como que una emisora puede ser seleccionada como favorita por uno o varios usuarios identificados; un usuario puede añadir una o varias emisiones quedando registrada la fecha y puntuar una o varias emisiones quedando registrada además de la fecha, la posición donde se ha realizado y la valoración realizada; una emisora de radio puede ser de uno o varios tipos (musical, noticias, deportiva...), pero sólo puede tener un alcance (nacional, local, comarcal...); del mismo modo que un usuario puede escribir comentarios (con las restricciones que se establezcan) que aparecerán de modo público en el sistema.

5. Análisis y diseño

El apartado de análisis y diseño se divide en varias partes.

Estructura del trabajo. División de la solución, justificando y explicándolo

La elaboración del proyecto FM Radio RDS se puede dividir en varias apartados: la parte local, que consiste en la aplicación instalada en los dispositivos con Firefox OS; la parte web, que consiste en que la misma aplicación esté alojada en un servidor permitiendo usarla como si fuera una página web y la parte del servidor que contiene la base de datos, la clave del funcionamiento del proyecto y los ficheros PHP a los que accede la aplicación remotamente. Se pueden agrupar las dos primeras citadas por el lado de la aplicación (ya sea app o webapp) y la tercera por el lado de servidor (la BD más ficheros PHP).

Al ser un proyecto web o de creación de una webapp para Firefox OS, en este caso está desarrollado en HTML5, JavaScript y los estilos CSS, además de alojar archivos .PHP en el servidor con los que la propia aplicación interaccionará.

Como previamente se ha comentado, para el desarrollo del trabajo se ha utilizado el framework TakoJS. Desde la propia página web se accede a la descarga de los archivos necesarios, cuyo peso inicial es inferior a 2MB. Inicialmente, el marco de trabajo está formado por dos carpetas, una para los ficheros JavaScript con extensión .js y otra para los estilos, archivos con extensión .CSS, además de los archivos con el código html y el archivo manifest.webapp en el directorio padre.

En el caso que atañe a este proyecto, la estructura final de la aplicación será más completa, como muestra la imagen Ilustración 16.

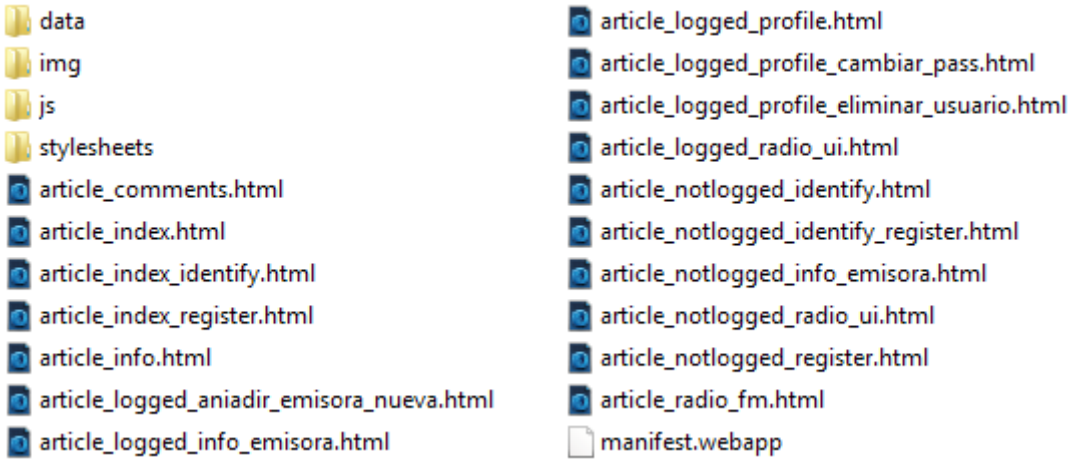


Ilustración 16: Estructura de la app

La aplicación web utiliza un gran número de archivos html a los que internamente la aplicación va llamando. En este caso, el principal será article_index.html, ya que en el archivo manifest.webapp, el atributo "launch_path" apunta a este como el archivo inicial de la aplicación. Cada uno de los otros archivos contiene el código de los diferentes "articles" en que se divide la aplicación.

En la carpeta data, encontramos los archivos necesarios para la utilización del multiidioma. Como se explica en Anexo III – Manuales sobre uso de Localization - L10N, Mozilla recomienda utilizar este sistema para realizar la gestión del multilenguaje en las aplicaciones para su sistema operativo móvil, introduciendo la etiqueta data-l10n-id en el código HTML5. Visto el esqueleto de la Ilustración 17, queda aparentemente claro que la aplicación estará traducida al inglés de los Estados Unidos, al español de España, al euskera y al francés. En el archivo locales.ini se configurarán estos idiomas, además de indicar el idioma por defecto, en caso de que un usuario de otra lengua quiera utilizar la aplicación. El desarrollador, para este proyecto, cree conveniente que sea el inglés el idioma por defecto.

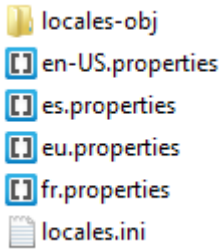


Ilustración 17: Estructura de la carpeta data

El directorio js contiene toda la funcionalidad de la aplicación creada, ya que todos los archivos JavaScript se encuentran en su interior. Al ser una aplicación creada en HTML, CSS y JavaScript, toda la parte del funcionamiento recaerá en estos ficheros .JS. Principalmente, los archivos originales que aportan funcionalidad son tako.components.js, tako.debug.js y tako.js. La funcionalidad que desarrolla el programador de la app está incluida en el archivo tako.index.js. El contenido de este archivo tako.index.js originalmente estaba incluido inicialmente en el archivo article_index.html, en un script en la parte final del mismo. Dada su larga extensión y para facilitar el uso y manejo, se prefirió externalizarlo a un archivo independiente. Otro de los archivos que se ven en la Ilustración 18 es shake.js, que se utilizará para capturar el movimiento del dispositivo en forma de agitación para registrar una incidencia.

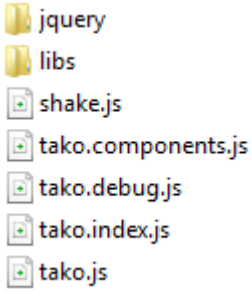


Ilustración 18: Estructura de la carpeta js

Dentro del directorio libs, se almacenan diferentes ficheros con extensión .JS que servirán para dar utilidad a la radio FM del dispositivo, algo fundamental para el trabajo llevado a cabo. En Reproducción de la radio FM, se explican las funciones que se utilizan del archivo my_fm.js, que está dentro del directorio libs.

Volviendo a la carpeta raíz del trabajo, falta por describir el contenido del directorio stylesheets, que, como su nombre en inglés describe, contiene todos los estilos que componen la aplicación web. En la Ilustración 19 se adjunta una imagen dividida en tres columnas.

La primera de ellas plasma el contenido del directorio raíz de los estilos originales del marco de trabajo TakoJS, donde se aprecian todos los archivos con extensión .CSS, uno para cada plataforma (Firefox OS, Android, iOS, WP) además de los estilos generales y los temas.

En la segunda columna de la imagen, se representa el contenido de la carpeta mine de la primera columna, donde se albergan los archivos que el desarrollador ha necesitado para moldear la aplicación a su gusto, así como las imágenes e iconos que se utilizan. En la raíz de este directorio, el archivo mine_styles.css contiene todo el código CSS con las definiciones de clases utilizadas en los html. Las carpeta favicon alberga el icono que aparece en las pestañas de los navegadores, si la aplicación web se abre desde estos tanto en pc como en teléfono móvil. El resto de carpetas, fonts, icons, icono_pua, icono_radio como logos, son bastante descriptivas en sí mismas, y se han ido utilizando para pruebas a lo largo del desarrollo. De todas ellas, se ha querido destacar el directorio icono_pua que ve en la Ilustración 19 en la tercera columna. Contiene los iconos utilizados durante la aplicación web, como el logo inicial y el logo en versión blanca y simplificada para la información de las emisoras de radio.

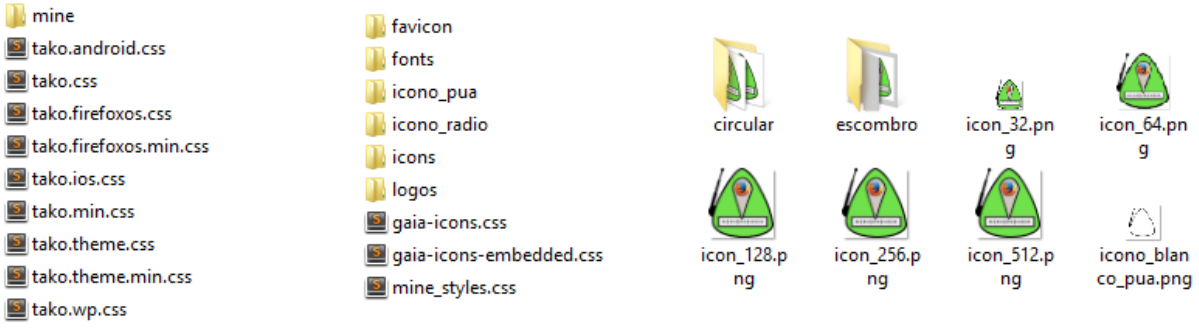


Ilustración 19: Estructura de la carpeta stylesheets, mine e icono_pua

Como dato curioso y destacable, las notificaciones que aparecen en la barra de notificaciones superior, pueden contener un icono junto al texto informativo, como se ve en las capturas de pantalla Ilustración 111, Ilustración 112 y Ilustración 113. Por las características del sistema operativo móvil, estas no pueden hacer referencia a ningún elemento de directorio interno del dispositivo, sino que deben estar alojados en algún servidor remoto. En este caso, el desarrollador optó por

almacenar en su cuenta particular de Dropbox, en un directorio de carácter público, los iconos que utilizará en las notificaciones. Esta será la dirección temporalmente:
https://dl.dropboxusercontent.com/u/5094436/fmradiordsapp/icons/icon_64.png.

Como se ha comentado en los primeros párrafos de este punto, tanto la aplicación que va instalada en el terminal como la versión web, utilizan la estructura de archivos y directorios explicados. En ambos casos, la aplicación interactúa con archivos alojados en el servidor (Ilustración 20), en una carpeta llamada php_files_in_server.

Estos archivos .PHP son bastante descriptivos con sólo ver sus nombres. La mayoría de ellos se conectan con la base de datos durante su ejecución para devolver el resultado solicitado en las llamadas de la aplicación. En el apartado Desarrollo se detalla algunos de los más importantes.

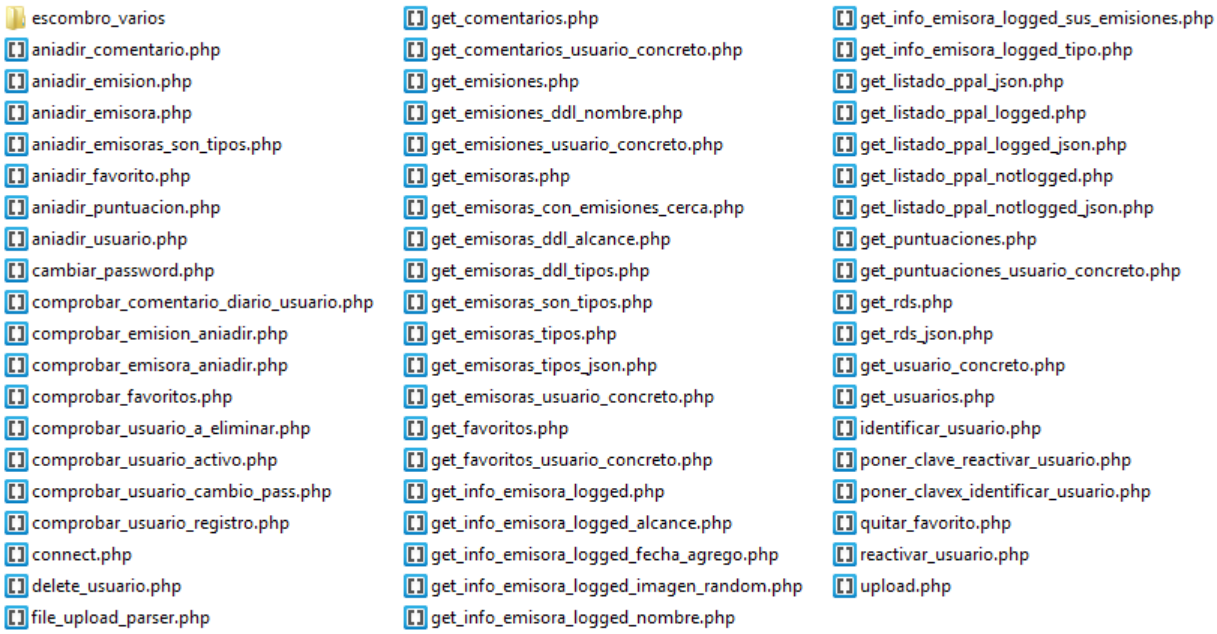


Ilustración 20: Estructura carpeta php_files_in_server

Diagrama relacional de la base de datos

Como ya se ha comentado, las conexiones a la base de datos se realizan a través del objeto XMLHttpRequest, que se conecta a la base de datos alojada en el servidor de la universidad *galan.ehu.es.* La base de datos es de tipo MySQL, la cuál es un sistema de gestión de bases de datos relacional, multihilo y multiusuario. A la hora de escoger el motor de almacenamiento, estando entre MyISAM o InnoDB, el

desarrollador opta por InnoDB ya que dota a MySQL de un motor de almacenamiento transaccional (conforme a ACID) con capacidades de commit (confirmación), rollback (cancelación) y recuperación de fallos. InnoDB realiza bloqueos a nivel de fila y también proporciona funciones de lectura consistente sin bloqueo en sentencias SELECT. Estas características incrementan el rendimiento y la capacidad de gestionar múltiples usuarios simultáneos. No se necesita un bloqueo escalado en InnoDB porque los bloqueos a nivel de fila ocupan muy poco espacio. InnoDB también soporta restricciones FOREIGN KEY. En consultas SQL, aún dentro de la misma consulta, pueden incluirse libremente tablas del tipo InnoDB con tablas de otros tipos[37].

En base al modelo de dominio explicado y profundizado en el punto 4.2 Modelo de dominio, se obtiene el siguiente diagrama relacional de la base de datos en producción:

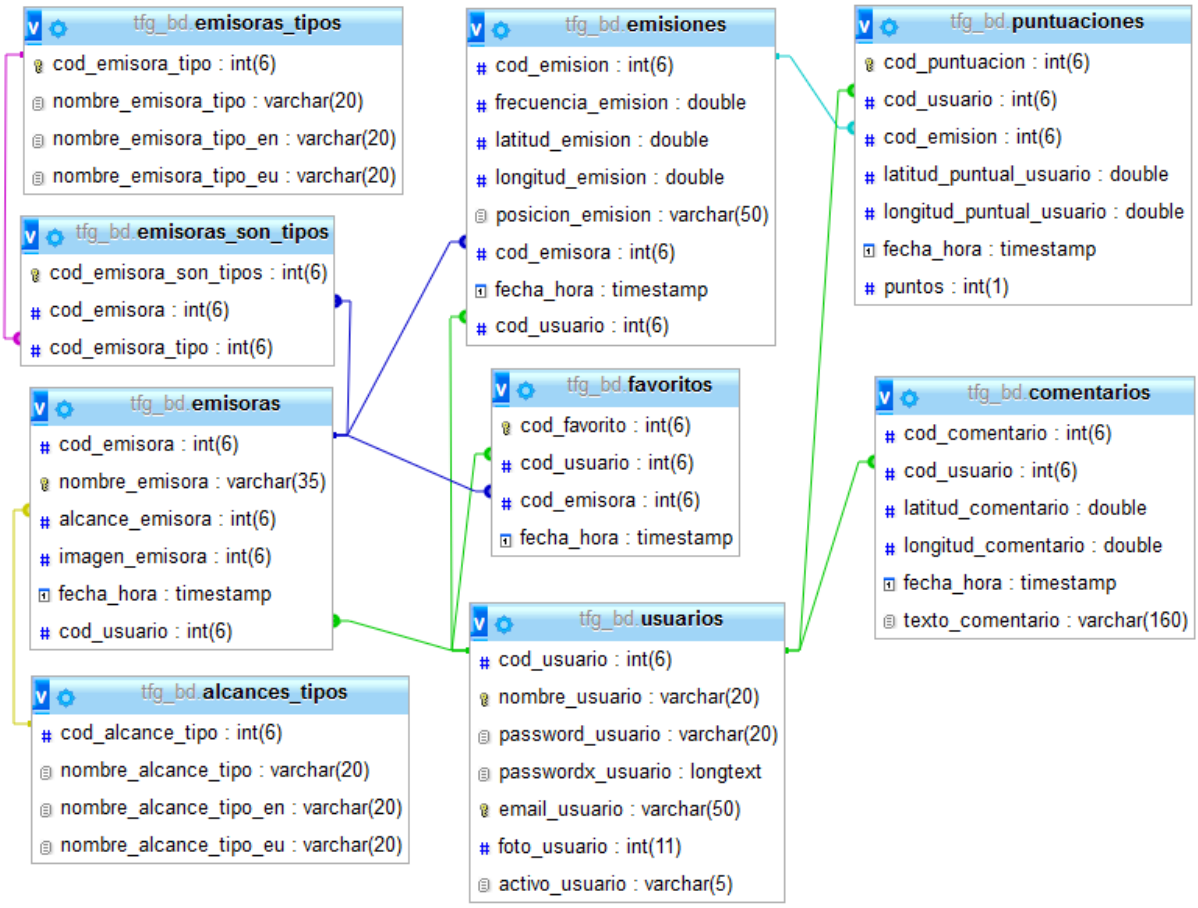


Ilustración 21: Diagrama relacional de la base de datos

En la imagen, se pueden ver las diferentes tablas con sus atributos de los que está compuesta la base de datos, así como las relaciones que hay entre dichas tablas.

La tabla usuario contiene los datos de los usuarios registrados en el sistema. Para cada registro, almacena el código, el nombre, la clave, el email y si el usuario está activo en el sistema. Tiene como clave primaria el código usuario. Esta tabla está relacionada con las tablas comentarios, puntuaciones, favoritos, emisiones y emisoras a través del código usuario, ya que cada registro que se almacena en estas tablas guardará quien ha realizado dicha acción. En concreto, quién ha realizado un comentario, quién ha añadido una puntuación, quién ha añadido una emisora como favorita, quién ha añadido una emisión y quién ha añadido una emisora.

La tabla comentarios contiene los datos de los comentarios realizados por los usuarios en el sistema. Para cada registro, almacena el código del comentario, el código del usuario que ha realizado el comentario, la latitud y la longitud de donde se ha registrado ese mensaje, la fecha y la hora del registro y fundamentalmente el texto que el usuario haya añadido. Tiene como clave primaria el código comentario. La única relación de esta tabla es con la tabla usuarios a través del atributo código usuario, siendo este atributo clave foránea.

La tabla favoritos contiene los datos de las emisoras que el usuario ha añadido como favoritas. Para cada registro, almacena el código, el código del usuario que ha añadido la emisora como favorita, el código de la emisora que el usuario ha añadido como favorita y la fecha y la hora en que se ha producido esa gestión. Tiene como clave primaria el código favorito. Esta tabla está relacionada con la tabla usuarios mediante la clave foránea código usuario y con la tabla emisoras mediante la clave foránea código emisora.

La tabla puntuaciones contiene los datos de las emisiones que el usuario ha valorado. Para cada registro, almacena el código de la puntuación, el código del usuario que ha realizado esa valoración de la emisión, el código de la emisión que el usuario ha puntuado, la latitud y la longitud en donde el usuario ha realizado esa puntuación, la fecha y la hora en que se ha producido esa valoración y principalmente, la puntuación que ha realizado. Este valor estará entre 1 y 5. Tiene como clave primaria el código puntuación. Esta tabla está relacionada con la tabla usuarios mediante la clave foránea código usuario y con la tabla emisiones mediante la clave foránea código emisión.

La tabla emisiones contiene los datos de las emisiones que los usuarios han ido añadiendo al sistema. Para cada registro, almacena el código de la emisión, la frecuencia de la emisión, la latitud y la longitud en donde el usuario ha añadido

ese registro a la base de datos, la posición en forma de ciudad y/o provincia donde se ha añadido ese registro, el código de la emisora de la emisión que el usuario ha añadido, la fecha y la hora en que se ha producido ese añadido y código del usuario que ha registrado la emisión. Tiene como clave primaria el código emisión. Esta tabla está relacionada con la tabla puntuaciones mediante la clave primaria código emisión, con la tabla usuarios mediante la clave foránea código usuarios y con la tabla emisoras mediante la clave foránea código emisora.

La tabla emisoras contiene los datos de las emisoras que los usuarios han ido añadiendo al sistema. Para cada registro, almacena el código de la emisora, el nombre de la emisora, el alcance de la emisora, la fecha y la hora en que se ha producido ese añadido y código del usuario que ha registrado la emisora. Tiene como clave primaria el código emisora. Esta tabla está relacionada con la tabla emisiones, favoritos y emisoras_son_tipos mediante la clave primaria código emisora, con la tabla usuarios mediante la clave foránea código usuarios y con la tabla alcances_tipos mediante la clave foránea alcance emisora.

La tabla alcances_tipos contiene los datos de los tipos de alcances posibles para las emisoras. Son unos valores fijos, predefinidos por el administrador/desarrollador. Para cada registro, almacena el código del tipo de alcance y el nombre del tipo de alcance. En este caso, se registran en los cuatro idiomas que el desarrollador ha introducido: castellano, euskera, inglés y francés. Tiene como clave primaria el código del tipo de alcance. Esta tabla está relacionada con la tabla emisoras mediante la clave primaria código tipo de alcance.

La tabla emisoras_tipos contiene los datos de los tipos de emisoras posibles para las emisoras. Son unos valores fijos, predefinidos por el administrador/desarrollador. Para cada registro, almacena el código del tipo de emisora y el nombre del tipo de emisora. En este caso, se registran en los cuatro idiomas que el desarrollador ha introducido: castellano, euskera, inglés y francés. Tiene como clave primaria el código del tipo de emisora. Esta tabla está relacionada con la tabla emisoras mediante la clave primaria código tipo de emisora.

Por último, la tabla emisoras_son_tipos contiene los datos de la relación entre los tipos de emisoras y las emisoras, ya que una emisora puede ser de varios tipos. Para cada registro, almacena el código del tipo de emisora que son, el código del tipo de emisora y el código de emisora. Tiene como clave primaria el código del tipo de emisora que son. Esta tabla está relacionada con la tabla emisoras tipos mediante la clave foránea código del tipo de emisora que son y mediante la clave

foránea código emisora con la tabla emisoras.

6. Desarrollo

El apartado de desarrollo se centra en describir con un lenguaje natural, en qué y cómo se ha hecho el trabajo de fin de grado, justificando las decisiones que se han tomado.

El framework Tako JS y como utilizarlo en beneficio del sistema

Una vez explicado el diseño del framework en el punto anterior, es momento de entrar en detalle sobre el contenido de los ficheros que lo componen que el desarrollador ha modificado y cómo se han utilizado para el proyecto.

El archivo tako.index.js está dividido por varias funciones. La primera de ellas es Tako.init(), que es la función inicializadora de Tako. Contendrá un objeto con las opciones de inicio, como los diferentes archivos "articles" que cargará de modo asíncrono.

Otra de las funciones es Tako.onReady(function()), que recibe la función que será cargada cuando Tako haya finalizado de cargar completamente. Dentro de esta función que retornará un callback al sistema, es donde se incluye todo el funcionamiento de la aplicación en cuanto a pulsación de botones, navegación entre sections y articles, llamadas a otras funciones, etc. El desarrollador ha dividido internamente el código en varios apartados separados por comentarios, con cada una de las funcionalidades que se pueden desarrollar en FM Radio RDS, en vez de varios archivos, todo con la intención de mantener la estructura inicial del marco de trabajo utilizado.

Estructura del código del archivo tako.index.js

El archivo está dividido en partes, separados por comentarios que indican que es cada parte del código. Utilizando las ventajas de JavaScript, se van ejecutando una serie de instrucciones y funciones según el botón que se pulse o la parte de la webapp que se va cargando. Por ejemplo, para la identificación de los usuarios, el usuario necesariamente debe pulsar en el botón de identificarse una vez haya introducido los datos que se le piden. La pulsación de ese botón provoca la ejecución de este método:

```
document.getElementById("id_article_index_identify_btn_indentify").addEventListener(
"click",function() {...});
```

En su interior, la aplicación realizará varias comprobaciones, ejecutará llamadas a notificaciones, realizará cambios en la base de datos del servidor, modificará valores de variables del almacenamiento local... todo por haber sido

ejecutado el evento addEventListener(). Este, registra un evento a un objeto especifico. El objeto especifico puede ser un simple elemento en un archivo, el mismo documento, una ventana o un XMLHttpRequest. En este caso de muestra, se registra el botón *id_article_index_identify_btn_indentify* .

En el desarrollo de FM Radio RDS también se echa mano de una de las librerías más utilizadas en el mundo del desarrollo web, jQuery. Se ha utilizado la versión **jquery-1.11.1.js**, almacenada dentro del directorio js como se aprecia en Ilustración 18: Estructura de la carpeta js. jQuery es una librería JavaScript rápida, ligera y con una gran cantidad de funciones. Permite interactuar de manera sencilla con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX, haciéndolo mucho más simple con un API fácil de usar que funciona a través de una multitud de navegadores. Con una combinación de versatilidad y capacidad de ampliación, jQuery ha facilitado la forma de escribir JavaScript de muchos desarrolladores. *$("#section_notlogged_radio_ui").on("load", function(){})* es un ejemplo de uso en el trabajo, donde es necesario lanzar una serie de funciones cuando se carga la sección en cuestión, para lo que el uso de la función *.on("load",...)* lo facilita.

En los Anexo I - Casos de uso extendidos y Anexo II - Diagramas de secuencia se desarrollan extensamente y en profundidad estos eventos .on("load",...) y addEventListener() que se han explicado en los párrafos previos y que dan gran parte de la funcionalidad de la aplicación web.

Configuración del servidor

Se configura la dirección del servidor donde están alojados los archivos PHP con los que se conecta la app.

```
var tipo_disp = "server";
if (tipo_disp == "server"){
server_path =  "https://galan.ehu.es/imagonbar88/DAS/TFG/php_files_in_server/";}
```

En la ruta que se indica en el código anterior, se encuentra el fichero connect.php. Se encarga de la conexión de la app con el servidor:

```
//SERVER
$servername = "localhost";//"GalanEHUServer";
$username = "Ximagonbar88";
$password = "*******";
$db_name = "Ximagonbar88_tfg_bd";
// Create connection
```

```
$conn = mysqli_connect($servername, $username, $password);
// Check connection
if (!$conn) {die("Connection failed: " . mysqli_connect_error());}
//echo "Connected successfully<br>"; //Innecesario
mysqli_select_db($conn,$db_name);
mysql_query("SET NAMES 'utf8'");
mysql_query("SET CHARACTER SET utf8");
mysql_query("SET COLLATION_CONNECTION,  'utf8_unicode_ci'");
//Palabra clave para las claves
$tipo = "*****";
$hitz_izkutua = "*****";
```

En las variables $servername, $username y $password se introducirán los datos que dan acceso al servidor. Para crear la conexión, se utiliza la instrucción $conn = mysqli_connect ($servername, $username, $password). Las últimas líneas servirán para la creación de un string encriptado, a la hora de almacenar en la base de datos la contraseña de cada usuario. Posteriormente, en cada fichero PHP utilizado, es necesario escribir *require 'connect.php';* para que crear la conexión.

XMLHttpRequest

Quedando explicada cómo se realiza la configuración del servidor, una parte fundamental y prioritaria es la comunicación de la aplicación con la base de datos y los archivos alojados en el servidor.

El objeto JavaScript XMLHttpRequest (también llamado XHR), proporciona una forma fácil de obtener información de una URL sin tener que recargar la página completa. Una página web puede actualizar sólo una parte de la página sin interrumpir lo que el usuario está haciendo. XMLHttpRequest es ampliamente usado en la programación AJAX. A pesar de su nombre, XMLHttpRequest puede ser usado para recibir cualquier tipo de dato, no sólo XML, y admite otros formatos además de HTTP (incluyendo file y FTP). Actualmente es un estándar de la W3C[38].

Aplicado al proyecto, un gran número de las llamadas que se realizan desde el archivo tako.index.js al servidor donde están alojados los ficheros PHP son a través del objeto XMLHttpRequest.

Por ejemplo, en el caso de uso Registrarse, se utilizan de dos modos diferentes el objeto XHR:

- El primer tipo, se usa por ejemplo en el método

comprobacion_registro_usuario_index_server(...). Se le pasan dos variables por POST, el nombre de usuario y su correo electrónico que se recogen de los campos de la interfaz, para comprobar si ese usuario ya existe previamente en el sistema. Se utiliza el método callback para obtener el valor de retorno, el que devuelve la llamada al servidor.

- La variable params pasará al servidor las variables necesarias.
- Se realiza una instancia al objeto XMLHttpRequest con la variable mozSystem:true entre sus parámetros, necesaria para Firefox OS.
- La variable url contendrá la dirección a la que hay que acceder, el archivo con el que se quiere comunicar el método.
- A través de la llamada a la función open, se realiza la conexión.
- Se deben ajustar las cabeceras para la solicitud, con las llamadas a la función setRequestHeader(“”,””);
- A través de la llamada a la función onload, se realiza la llamada a la función callback, que obtiene el resultado de la ejecución del archivo PHP remoto. En este caso, devolverá estanombre si el nombre introducido coincide con algún valor de la BD, estaemail si pasa lo mismo con el correo electrónico o estanambos si realmente coinciden las dos variables pasadas con el contenido de algún registro del sistema.
- Por último, es importante la llamada a la función send del objeto XHR ya que en este caso es la encargada de pasar las variables al PHP.

```
//Comprobación que no se registren usuarios con nombre o email igual a los que existen en la base de datos
function comprobacion_registro_usuario_index_server(callback){
 var usuario_name = $("#id_article_index_register_nombre_usuario").val();
 var usuario_email = $("#id_article_index_register_email").val();
 var params ="usuario_name="+usuario_name+"&usuario_email="
 +usuario_email;
 var xmlhttp = new XMLHttpRequest({ mozSystem: true });
 var url = server_path + "comprobar_usuario_registro.php";
 xmlhttp.onload = function(){callback(xmlhttp.responseText); };
 xmlhttp.open("POST", url, true);
 xmlhttp.setRequestHeader("Content-type","application/x-www-form-
```

```
urlencoded");
 xmlhttp.setRequestHeader("Content-length", params.length);
 xmlhttp.setRequestHeader("Connection", "close");
 xmlhttp.send(params); }
```

- El segundo tipo, se usa en el método *nuevo_aniadir_usuario_index_server()*, por ejemplo. Se le pasan dos variables por POST, el nombre de usuario y su correo electrónico que se recogen de los campos de la interfaz, para realizar el registro en el sistema. En este caso, el resultado de hacer la llamada a esa función es el añadir el texto resultante en la variable txtBD_aniadir_usuario_index, mediante el método innerHTML.
  - La variable params pasará al servidor las variables necesarias.
  - Se realiza una instancia al objeto XMLHttpRequest con la variable mozSystem:true entre sus parámetros, necesaria para Firefox OS.
  - La variable url contendrá la dirección a la que hay que acceder, el archivo con el que se quiere comunicar el método.
  - A través de la llamada a la función open, se realiza la conexión.
  - Se deben ajustar las cabeceras para la solicitud, con las llamadas a la función setRequestHeader(“”,””);
  - A través de la llamada a la función onreadystatechange, se realiza la comprobación de si se ha añadido bien o si han habido problemas. En caso de que los valores de readyState sean igual a 4 y de status a 200, se puede afirmar que el añadido del usuario se ha realizado correctamente y por ello se imprime en pantalla un mensaje indicándolo.
  - Por último, es importante la llamada a la función send del objeto XHR ya que en este caso es la encargada de pasar las variables al PHP.

```
function nuevo_aniadir_usuario_index_server(){
var usuario_nombre = $"#id_article_index_register_nombre_usuario").val();
 var usuario_pass = $("#id_article_index_register_pass").val();
 var usuario_email = $("#id_article_index_register_email").val();
 var params =
"usuario_name="+usuario_nombre+"&usuario_pass="+usuario_pass+"&usua
rio_email="+usuario_email;
 var xmlhttp = new XMLHttpRequest({ mozSystem: true });
```

```
 var url = server_path + "aniadir_usuario.php";
 xmlhttp.open("POST", url, true);
 xmlhttp.setRequestHeader("Content-type","application/x-www-form-
urlencoded");
 xmlhttp.setRequestHeader("Content-length", params.length);
 xmlhttp.setRequestHeader("Connection", "close");
 xmlhttp.onreadystatechange = function() {
 if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
        document.getElementById("txtBD_aniadir_usuario_index").
        innerHTML = xmlhttp.responseText;}
}xmlhttp.send(params);}
```

El ejemplo desarrollado, quiere demostrar cómo se han realizado varios de los apartados de la aplicación, como la obtención de los datos de usuario, emisoras favoritas, listado de nombres de emisoras, etc.

$.getJSON() - $each() - jQuery

Gracias a la librería jQuery, el desarrollador consiguió superar una barrera, salvar el mayor quebradero de cabeza encontrado a lo largo del desarrollo de este trabajo de final de grado, que se explica extensa y profundamente en este apartado (Generación de contenido dinámicamente y políticas de seguridad (CSP)). Por suerte, jQuery aporta una serie de funcionalidades que a los desarrolladores web facilita la vida considerablemente. En este caso, la función que se quiere destacar es *$getJSON()*.

Relacionada también con el acceso a los archivos alojados en el servidor, mediante esta función también se accede a estos desde el código javascript. En este caso, la función *show_content_bd_radio_ui_logged_json()* se encargará de solicitar el listado de emisiones más cercanas a la posición donde se encuentre el usuario, ordenadas por el nombre de emisora. Estos elementos serán creados uno a uno y aparecerán en forma de listado en la interfaz principal de la aplicación, tanto si se está identificado como si no. Cada uno de estos elementos necesariamente deberá ser seleccionable, ya que la pulsación en cada elemento modificará elementos de la interfaz, como el nombre de la emisora y la frecuencia que se está escuchando, así como variables internas del sistema. En la capacidad de ser pulsado un elemento es donde $getJSON proporcionó una solución realmente óptima para salvar los problemas que se tenían con XHR.

- Se acceden a dos variables del almacenamiento local con la función

localStorage.getItem() que posteriormente se pasarán como parámetros en la variable array_data..

- La variable URL contendrá la dirección del archivo PHP del servidor.
- Se realiza la llamada a la función getJSON, pasando entre paréntesis los parámetros URL, array_data y la llamada a la función.
- $each, otra función jQuery, permite iterar a través de los valores obtenidos. El parámetro i será el índice de cada iteración, donde emisoras contendrá el valor de la emisora en cada vuelta.
- Dentro de cada vuelta, se creará el nuevo elemento que posteriormente aparecerá listado, asignado a la variable nueva_linea con las etiquetas html correspondientes. Ese nuevo elemento, necesariamente se creará dinámicamente ya que depende de la posición donde el usuario se encuentre. A su vez, para cada elemento se crean dos eventos "clickables", mediante el uso de eventos addEventListener, uno para que el usuario pueda seleccionar el icono blanco del logo y ver la información de la emisora (document.getElementById("emisora_"+emisoras.cod_emisora_json).addEventListener("click", function(){); y otro para que el usuario pueda seleccionar ese elemento del listado, se actualice la emisora que se está escuchando, la información que se está mostrando de su nombre y frecuencia y para indicar que elemento del listado está pulsado en otro color (document.getElementById("emision_"+emisoras.cod_emision_json).addEventListener("click",function(){). Por todo ello, era necesario crear estos eventos pulsables.

```
function show_content_bd_radio_ui_logged_json(){
 $("#txtBD_radio_ui_logged_json").children("div").remove();
 var pos_latitud = localStorage.getItem('mylatit');
 var pos_longitud = localStorage.getItem('mylongit');
 var precision = "normal_prec";
 var url = server_path + "get_listado_ppal_logged_json.php";
 var array_data = {"posicion_lat": pos_latitud, "posicion_long":
                        pos_longitud, "prec": precision};
 $.getJSON(url, array_data, function(emisoras) {
        $.each(emisoras, function(i,emisoras){
                if ((emisora_anterior != emisoras.cod_emisora_json) ||
```

```
                (emisora_anterior == 0)){
                        emisora_anterior = emisoras.cod_emisora_json;
        nueva_linea = {...};
        $('#txtBD_radio_ui_logged_json').append(nueva_linea);
document.getElementById("emisora_"+emisoras.cod_emisora_json).addEve
ntListener("click", function(){
        seleccionar_emision_icono(emisoras.cod_emisora_json); });
{...}
document.getElementById("emision_"+emisoras.cod_emision_json).addEve
ntListener("click",function(){
                seleccionar_emision(emisoras.cod_emision_json,
                emisoras.freq_emision_json, emisoras.cod_emisora_json,
                emisoras.nombre_emisora_json);
                ver_si_es_favoritos();
        });
{...}
```

El ejemplo desarrollado, quiere demostrar como se han realizado varias de los apartados de la aplicación, como la obtención del listado RDS y el listado principal, tanto con el usuario identificado como si no lo está.

Archivos alojados en el servidor

En los puntos anteriores, se han descrito tanto la configuración del servidor como los sistemas de conexión con el servidor de la aplicación alojada en los terminales Firefox OS como de la webapp. Pero, falta detallar el contenido de alguno de los archivos .php alojados en el directorio remoto a los que accede el sistema. Como anteriormente se ha ejemplificado con el caso de uso Registrarse en XMLHttpRequest, se seguirá esa misma línea.

En este caso, se realizan dos solicitudes al servidor, una al archivo "*comprobar_usuario_registro.php*" y otra solicitud a "*aniadir_usuario.php*" alojado en el mismo directorio. Por la claridad de su código, se va a detallar el contenido del fichero aniadir_usuario.php.

Inicialmente, se declara el fichero independiente donde está albergada la información necesaria para realizar la conexión con la base de datos, *require 'connect.php'*. A continuación, hay que guardar en variables locales los valores que se reciben del JavaScript: usuario_name, usuario_pass y usuario_mail. En esta

ocasión, los elementos se pasan por POST, no quedando reflejados en la URL de la llamada. Para evitar espacios en blanco posteriores y anteriores, se realiza una llamada a la función *trim()* del valor recibido. Las variables de cadena, son necesarias tratarlas con la función *utf8_encode()* para unificar acentos y caracteres extraños.

A su vez, el campo contraseña que se va a almacenar en la base de datos se desea que esté encriptado. Para este proyecto se eligió el tipo de encriptación sha512 a través de una palabra clave. Estas dos variables son *$tipo* e *$hitz_izkutua*, que se introducen en la función *hash_hmac()* y que a su vez están albergadas en el fichero independiente connect.php al que se hace referencia.

```
<?php
  require 'connect.php';
  $nombre_usuario = trim($_POST["usuario_name"]);
  $nombre_usuario_utf = utf8_encode($nombre_usuario);
  $pass_usuario = trim($_POST["usuario_pass"]);
  $email_usuario = trim($_POST["usuario_email"]);
  $passx_usuario = hash_hmac($tipo, $pass_usuario, $hitz_izkutua);
  $sql = "INSERT INTO usuarios (nombre_usuario, passwordx_usuario, email_usuario,
foto_usuario, activo_usuario) VALUES ('$nombre_usuario_utf','$passx_usuario',
'$email_usuario','0','true')";
  $result = mysqli_query($conn,$sql);
  $conn->close(); //mysqli_close($conn);
?>
```

Una vez recogidos todos los valores por POST, se realiza la sentencia SQL de inserción de datos en la BD a través de la función *mysql._query(...)* .

En este caso, no retorna ningún valor al archivo desde el que se ha realizado la llamada. Pero se puede devolver información al JavaScript a través de la función *echo(...):* devolver los datos que se han obtenido en forma de json (*get_listado_ppal_logged_json.php*), devolver texto que luego se tratará en el archivo JavaScript (*comprobar_usuario_registro.php*) o imprimir directamente el resultado obtenido de la consulta realizada (*get_favoritos.php*). Cada una de las llamadas a estos tres archivos citados tiene un código JavaScript diferente.

Geoposicionamiento inverso

Como anteriormente se ha comentado, el funcionamiento de FM Radio RDS es completo y correcto gracias a la información que aporta el chip GPS interno del terminal. HTML5 facilita la obtención de las coordenadas en las que se encuentra

el dispositivo, mediante este código:

```
var mylat2 = pos.coords.latitude;                    var mylong2 = pos.coords.longitude;
```

Con estas instrucciones, se obtiene la información de la latitud y la longitud global del terminal. Estos datos se almacenan en la memoria local del dispositivo, para mostrarlos en el apartado de perfil del usuario o para añadirlos a la base de datos cuando el usuario introduce alguna emisión al sistema.

Ilustración 22: Captura de pantalla reducida del menú perfil de FM Radio RDS, con las coordenadas y su posición asociada

Como se puede ver en la Ilustración 22, la aplicación tiene desarrollado un método que obtiene de los datos de las coordenadas su posición asociada, en términos técnicos denominado geoposicionamiento inverso. En los primeros momentos del desarrollo, se pensó en utilizar los servicios de la gran compañía tecnológica Google y su API de Google Maps para obtener esta información. Desgraciadamente, el desarrollador se vuelve a encontrar con los problemas relacionados con CSP en las aplicaciones de Firefox OS de tipo certificadas (Generación de contenido dinámicamente y políticas de seguridad (CSP)), ya que es necesario inicializar el script que conecta la webapp (indicado al final de este párrafo) para usar los servicios de la gran compañía de Mountain View, por lo que se incumplía una de las condiciones necesarias de las aplicaciones certificadas.

```
-<script type="text/javascript" src="https://maps.googleapis.com/maps/api/js?
v=3.exp&signed_in=true"></script>-
```

En busca de soluciones para la obtención de este geoposicionamiento inverso, se probaron diferentes alternativas hasta dar con la plataforma open-data OpenStreetMap y su API Nominatim. Esta, permite con su API pública obtener fácilmente el nombre de la posición donde se encuentra el usuario gracias a su método reverse pasándole una serie de parámetros entre los que fundamentalmente están las coordenadas donde se encuentra el dispositivo.

Es un servicio open-data, con una serie de restricciones y requerimientos para

poder utilizarlo[39], entre los que está hacer un uso no excesivo de llamadas, proporcionar un agente-usuario válido para cada llamada e indicar claramente que tipo de licencia se está utilizando. En este caso, se está utilizando una herramienta con licencia ODbL(Open DataBase License) [40] y está indicado en UI_info_app.

Para obtener el nombre de la posición en donde se encuentra el usuario, se ha desarrollado el método geoPos():

```
function geoPos(){
  var geo_options = { enableHighAccuracy: true, maximumAge: 5000, timeout: 8000 };
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(getLatituteLongitude, errorFunction);
  } else {
    console.log("Errorr");
    if (localStorage.getItem('notif_gps' == false)){
      show_notify_gps_desactivated();
}}}
function errorFunction(pos) {{...}; show_notify_gps_desactivated(); } {...}
function getLatituteLongitude(pos) {
    var mylat2 = pos.coords.latitude; mylat2 = redondeo(mylat2,6);
    var mylong2 = pos.coords.longitude;mylong2 = redondeo(mylong2,6);
    {...}
    var array_data = {"format":"json","lat": my_lat2, "lon": mylong2, "zoom": 18,
"addressdetails": 1};
    var url = "https://nominatim.openstreetmap.org/reverse?";
    $.getJSON(url, array_data, function(posic) {
      var mycity = "";
      if (posic.address.county){ mycity = posic.address.county; }
      else{ if (posic.address.city){mycity = posic.address.city; } else{...}
}}
  if ((posic.address.state)&&(posic.address.country)){
    mycity = mycity + ", " + posic.address.state + ", " + posic.address.country;  {...}
  }});
```

Durante la ejecución de la aplicación web, se realiza la llamada a la función geoPos() en varias ocasiones: según se inicia la aplicación, cada vez que se va a añadir una emisión en el sistema y cuando se pulsa en el botón recargar, tanto si el usuario está identificado como si no lo está. Al inicial la app, aparecerá un mensaje de aviso preguntando si se permite que la aplicación acceda a los datos de la

geoposición que reporta el chip GPS interno y habría que aceptarlo. Si el acceso a la geoposición no es aceptada, aparecerá un mensaje de error en forma de notificación en la barra superior del sistema operativo móvil; sino, se llama a esta función:

```
navigator.geolocation.getCurrentPosition(getLatituteLongitude, errorFunction);
```

Manifest.webapp - Firefox OS app

Toda aplicación para el sistema operativo móvil Firefox OS necesita de un archivo *manifest.webapp* para poder ser instalado en el terminal y ejecutarse. "*Un Open Web App* ***manifest*** *contiene información que un navegador Web necesita para interactuar con una aplicación. Un manifest es una de las cosas claves que distinguen una Web App de un sitio web. Es un archivo JSON con un nombre y una descripción para la aplicación y que contiene además el origen de la aplicación, íconos, información del desarrollador y los permisos requeridos por la aplicación, entre otras cosas."* como se describe en la página oficial de Mozilla Developer Network [41].

Este fichero está formado por pares (clave : valor) de datos que van formando su estructura. Estos son varias de las claves imprescindibles a rellenar: el nombre de la app (name); una breve descripción de la misma que le aparece al usuario cuando la instala(description); el tipo de aplicación que es (type); el archivo inicial con el que iniciar la aplicación (launch_path); la versión de la aplicación en función de los cambios que ha ido sufriendo con el tiempo (version); la ruta de los iconos correspondientes a la app (icons), siendo obligatorios el de 128 y 512; la información del desarrollador (developer) con su nombre (name) y el enlace a su página web personal (url); el idioma por defecto de la aplicación (default_locale), en caso de que la app no esté traducida al idioma del terminal desde donde se ejecuta; los idiomas en los que está desarrollada la aplicación (locales), cada uno de ellos con su nombre y su descripción en su idioma para mostrarlo en el sistema; la orientación por defecto para la app (orientation) y los permisos requeridos. El apartado de los permisos es de los más importantes, ya que es donde el desarrollador indica los permisos que la aplicación necesita para su ejecución, como puede ser uso de la geolocalización, del almacenamiento interno, notificaciones de escritorio, acceso a fmradio, al sistema XHR para conectar con el servidor, acceso a los contactos del terminal, alarmas, etc.[42]. Aquí se presenta el manifest.webapp en versión reducida de la aplicación FM Radio RDS.

```
{
  "name": "FM Radio RDS",
```

```
  "description": "My FxOS FM Radio RDS buidt with Tako framework",
  "type": "certified",
  "launch_path": "/article_index.html",
  "version" : "1.1",
  "icons": {
    "24": "/stylesheets/mine/icono_pua/circular/icon_24.png",
    "32": "/stylesheets/mine/icono_pua/circular/icon_32.png",
    "64": "/stylesheets/mine/icono_pua/circular/icon_64.png",
    "128": "/stylesheets/mine/icono_pua/circular/icon_128.png",
    "256": "/stylesheets/mine/icono_pua/circular/icon_256.png",
    "512": "/stylesheets/mine/icono_pua/circular/icon_512.png"
  },
  "developer": {
    "name": "imagonbar", "url": "http://imagonbar.typify.io/"
  },
  "default_locale":"en-US",
  "locales": {
    "en-US":{"name":"FM Radio RDS","description":"My FxOS FM Radio RDS buidt
with Tako framework"},
    "fr":{"name":"Radio FM**","description":"Radio FM de Gaia"},
    "es":{"name":"FM Radio RDS","description":"FxOS FM Radio RDS construida con el
framework Tako"},
    "eu":{"name":"FM irratia RDS","description":"FxOS FM Radio RDS Tako framework-
ekin eraikina"}, {...}
  },
  "permissions": {
    "geolocation": { "description": "Necesario para indicar la posición actual." },
    "storage": { "description": "Necesario para almacenar rutas – IndexedDB access to
                  store library and do not rescan whole library on each start" },
    "desktop-notification": { "description": "Necesario para crear notificaciones" },
    "contacts": { "description" : "Ver los contactos", "access" : "readwrite" },
    "systemXHR": { "description": "Required to load remote content" },
    "fmradio": { "description": "Necesario para habilitar la Radio FM." },
    "speaker-control":{"description": "Necesario para habilitar el altavoz en Radio FM." }
  },
  "orientation":"portrait"
}
```

Reproducción de la radio FM a través de la aplicación desarrollada

Por descontado, la funcionalidad principal y más importante de la webapp desarrollada es la reproducción de la señal de radio FM y hasta las últimas semanas no se consiguió dar con la tecla para hacerla funcionar. Una vez se tenían todas las funcionalidades explicadas en marcha y sin aparentes fallos, como por ejemplo el registro, la identificación, la obtención de las emisoras y emisiones cercanas, los registros personales, información de las emisoras... faltaba poder escuchar las frecuencias que correspondían a los elementos que se hubieran seleccionado.

Ilusamente, la dificultad que tenía poner en marcha esto iba a ser mínima, ya que sólamente faltaba hacer sonar la radio pasándole el parámetro de la frecuencia elegida. Pero ni mucho menos fue tan sencillo.

Los desarrolladores de Mozilla tienen todo el código del S.O. subido a un repositorio público de GitHub, además del de las aplicaciones instaladas de serie. La app original de radio FM también, y gracias a ello, se han podido reutilizar métodos que se utilizan de manera satisfactoria en la aplicación FM Radio original de Firefox OS, utilizando la ingeniería inversa para interpretar el código.

Para utilizar cualquier método de la radio FM, hay que inicializar el objeto:

```
var myRadioFM = navigator.mozFM || navigator.mozFMRadio;
```

Una vez inicializado, con este método se comprueba si los auriculares están conectados al terminal, ya que estos hacen de antena para que la recepción de la señal sea mucho mejor:

```
myRadioFM.antennaAvailable();
```

Para comprobar si está activo la radio, se utiliza:

```
myRadioFM.enabled()
```

Para activar la radio FM con una frecuencia en concreto, se utiliza este código:

```
enableFMRadio(localStorage.getItem('last_frecuencia_emision'));
```

Y para desactivar o silenciar la radio, se utiliza esta función:

```
myRadioFM.disable();
```

Concretamente, se puede ver su utilización en los casos de uso extendidos “Play/Stop”, “Elegir emisión al pulsar lista manualmente” y “Elegir emisora al pulsar lista manualmente”.

Con estos métodos descritos, la funcionalidad de la aplicación se completa del modo que estaba ideado el proyecto por el desarrollador.

Problemas y las soluciones

En este apartado, se van a exponer una serie de problemas o inconvenientes que han ido surgiendo durante la elaboración del trabajo de fin de grado y se exponen las soluciones tomadas para cada caso.

Elección de base de datos y herramienta de gestión del contenido de la BD

El sistema a desarrollar necesita de una gran base de datos para su funcionamiento, ya que necesita almacenar una gran cantidad de datos de emisoras y frecuencias en función de la posición. La información estará alojada en un servidor y se harán peticiones, consultas y añadirán nuevas entradas para mantener actualizada la información; pero al mismo tiempo se desea alojar en local, en el dispositivo, una copia de la información correspondiente a la zona donde se encuentra para evitar ese consumo excesivo de datos de Internet. La aplicación consultará en todo momento la información alojada en la memoria del terminal y esta se actualizará periódicamente o manualmente con la información alojada en el servidor.

Leyendo la documentación del sitio web de referencia para el desarrollo de apps Firefox OS, Mozilla Developer Guide [43], aconsejan utilizar IndexedDB para gestionar estos datos alojados en local. Pero, por contra, al ser una base de datos almacenada en el dispositivo, no existen demasiadas herramientas que faciliten la lectura de contenido de estas base de datos.

Para el navegador Google Chrome existe una herramienta en el cajón de herramientas para desarrolladores que si permite gestionar el contenido de las bases de datos IndexedDB y funciona correctamente. Por contra, el desarrollo de la aplicación la estoy llevando a cabo con el navegador Mozilla Firefox y un complemento WebIDE que facilita la depuración instantánea de las apps, por lo que se ha intentado utilizar las herramientas que Mozilla aporta, sin suerte. Existe una pestaña en el cajón de herramientas de desarrollo llamada "Storage" que aparentemente debería gestionar la información de las bases de datos como localStorage, IndexedDB y otras, pero todavía no he conseguido hacerla funcionar quizá porque aún está en fase beta. Otra de las herramientas que se han intentado utilizar es un complemente del navegador Mozilla Firefox, IndexedDB Browser, que tampoco funciona a pesar de todos los intentos realizados.

Finalmente, se hace uso de la memoria interna del dispositivo/navegador localStorage, dejando pendiente mejorar el uso del modo offline incluyendo el uso de este sistema de bases de datos IndexedDB.

Aparición de un navegador pensado exclusivamente para desarrolladores

El tiempo que ha llevado la elaboración del trabajo de fin de grado ha sido extenso, pasando muchos meses, y eso en el mundo de la tecnología conlleva estar alerta de las novedades.

Concretamente, para la elaboración de este proyecto se estaba utilizando el navegador Mozilla Firefox y sus complementos de herramientas para desarrolladores. Simultaneamente a la versión del navegador citado, existen versiones del mismo navegador con funciones en fase de prueba, como Aurora y Nightly. Esta última versión Nightly incorporaba por finales de octubre un editor de código online llamado WebIDE, que ofrecía la posibilidad de instalar las webapps directamente en un terminal con Firefox OS, con Firefox para Android o en un simulador.

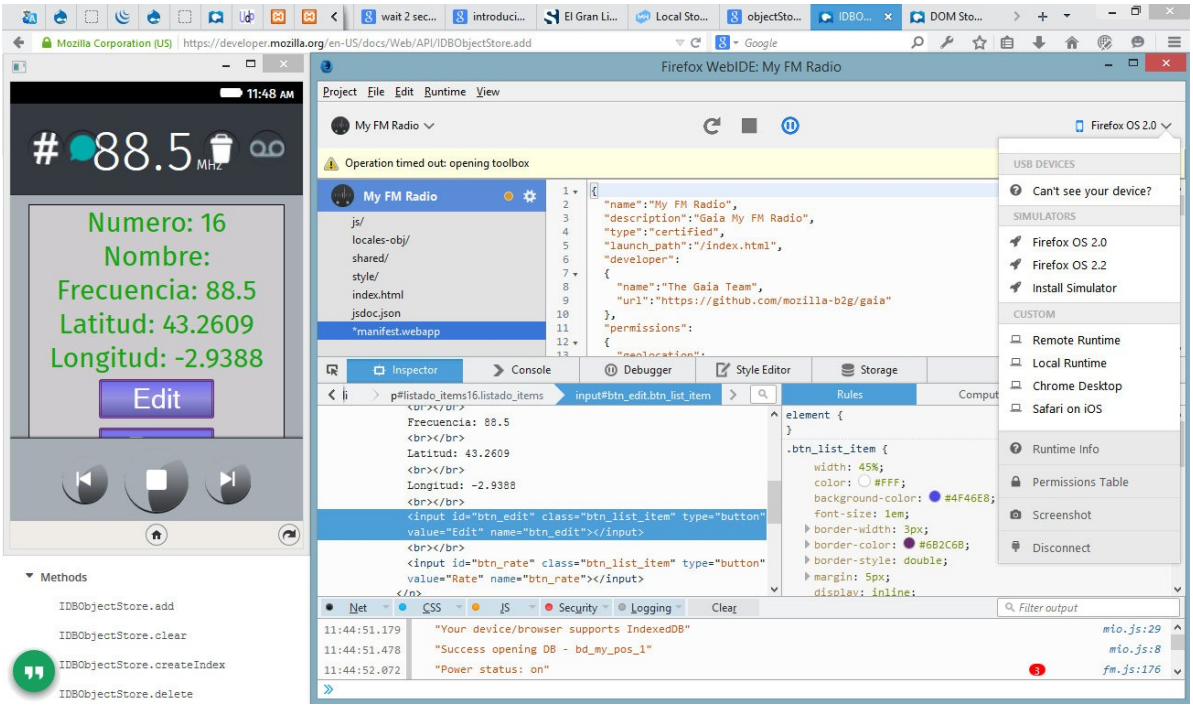


Ilustración 23. Captura de pantalla de la herramienta WebIDE, el simulador ejecutandose, el menu de desarrollador y de los dispositivos conectables desplegados.

A principios de noviembre del 2014, Mozilla celebraba su 10 aniversario presentando Firefox Developer Edition (también llamado FDE), una versión de su navegador enfocado a los desarrolladores con algunas de las funcionalidades que hasta ese momento habían estado en fase de prueba. En esta versión estable de FDE hay un acceso directo a la herramienta WebIDE, daban por fin soporte a la pestaña

de Almacenamiento-Storage pudiendo analizar en directo el contenido de los almacenes de datos, ya fueran Local Storage, Session Storage, IndexedDB y las Cookies almacenadas; y alguna otra funcionalidad más como el EyeDropper o Responsive Design Viewer, entre otras.

En definitiva, la fundación Mozilla facilitaba una herramienta que agrupaba muchas de las funciones indispensables para que cualquier desarrollador web lleve a cabo su trabajo de manera eficaz, rápida y cómoda y el responsable de este trabajo no iba a desaprovechar la oportunidad de aprender a utilizar esa nueva herramienta.

Ilustración 24. Logo Firefox Developer Edition

Con todo lo explicado, queda claro la importancia de estar constantemente informado de las novedades que van presentando las compañías de las herramientas que utilizamos, ya que estas mejoras pueden facilitar el trabajo en el futuro inminente.

Carga de contenido remoto de forma asíncrona de un servidor externo

Durante el desarrollo de la aplicación y para su funcionamiento final, se ha tenido que acceder al contenido de la base de datos y archivos alojados en el servidor. La conexión entre el dispositivo y "galan.ehu.es" no ha estado exenta de fallos o problemas de configuración y acceso a ella. A la hora de cargar contenido remoto de forma asíncrona de un servidor externo, muchos navegadores prohíben ese acceso de forma directa, alegando temas de seguridad para ello. De acuerdo con los mensajes de error indicados por los navegadores utilizados, había que solucionar el acceso remoto a esos archivos, acabar con el siguiente error:

```
Cross-Origin Request Blocked: The Same Origin Policy disallows reading the
remote resource at http://galan.ehu.es/.../php_files_in_server/get_usuarios.php.
(Reason: CORS header 'Access-Control-Allow-Origin' missing).
```

Era necesario configurar los ficheros .PHP alojados en el servidor, los que realizan peticiones a la base de datos para que pudieran remotamente acceder a esa información. Agregando unos encabezados en dichos ficheros .PHP, se aceptaban peticiones desde distintos servidores al actual, y obtener respuesta a ellas como si fuesen peticiones locales.[44][45]

```
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type,
Accept");
```

El asterisco que aparece al final es un parámetro que representa desde que hosts vamos a aceptar peticiones, en este caso se daría acceso global. Sin duda es un riesgo en términos de seguridad, por lo que se recomienda tratar de restringirlo sólo a los servidores que tenemos certeza que accederán, sustituyendo el asterisco por las direcciones URL deseadas.

En la captura de pantalla mostrada a continuación, se muestran las cabeceras que contiene la URL donde están alojados los ficheros .PHP mencionados. En este caso, lo destacable es ver como están las dos líneas de cabeceras agregadas. Para comprobar que las peticiones estén siendo aceptadas, se puede ejecutar CURL a través de consola de GIT.

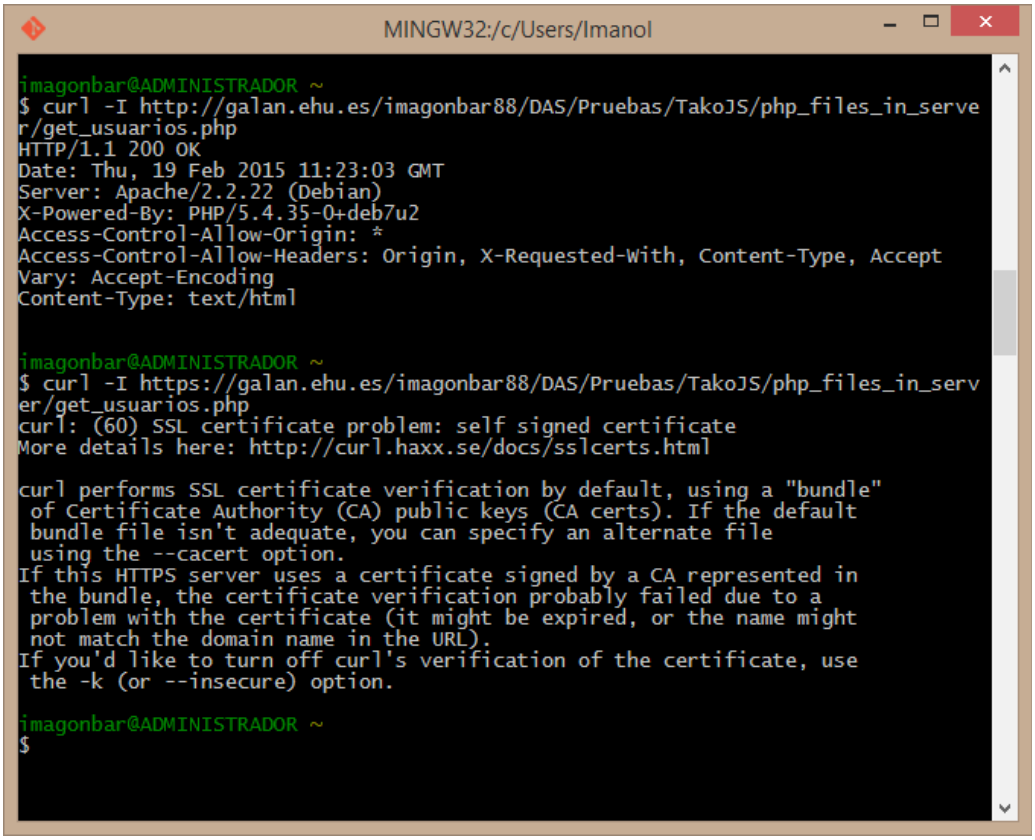


Ilustración 25: CP consola git

Utilizar un sistema que adapte el idioma de la aplicación y su contenido al idioma del terminal o navegador desde donde se esté ejecutando.

La idea del desarrollador es crear una herramienta que se pueda utilizar a nivel mundial, sin importar desde donde se ejecute, teniendo la necesidad de que los usuarios de ese país añadan contenido a las bases de datos, en este caso las

emisoras propias del territorio. Para ello, el idioma suele ser una de las grandes barreras que hagan que triunfe o fracase un sistema. Es necesario que las aplicaciones estén traducidas al mayor número de idiomas posible, sobre todo a los más utilizados a nivel global. En este caso y para cumplir unas expectativas acordes al tamaño inicial del proyecto, se han elegido el inglés, el castellano, el euskera y el francés por la situación regional.

Gracias a los avances en el desarrollo web, existen varias alternativas para llevar a cabo de manera efectiva y rápida esa posibilidad de multiidioma. En el caso de FM Radio RDS, al ser una aplicación para el sistema operativo móvil Firefox OS, se han seguido las explicaciones sobre el uso de la librería ***l10n.js*** que desde MDN aconsejan para tal fin, de modo que se ha conseguido disponer de la aplicación en los idiomas citados anteriormente, en función del idioma del sistema [46] [47]. En el Anexo III – Manuales sobre uso de Localization - L10N, se explica el funcionamiento de este sistema.

Generación de contenido dinámicamente y políticas de seguridad (CSP)

Una de las grandes virtudes que aporta la app al usuario es facilitar en cada posición unos datos adaptados. En la interfaz principal, se mostrarán el listado de emisoras que disponen de emisiones en ese lugar en concreto. De ese modo, se pretende que el usuario pueda escuchar la radio FM sin preocuparse en donde está, sin la necesidad de conocer las frecuencias de las emisoras que hay en cada lugar donde esté. El usuario pulsaría el nombre de la emisora que desea (Ilustración 52: UI_ logged_ppal) y está se empieza a reproducir a través de los auriculares conectados al terminal y se actualiza la información de la emisora y la frecuencia en la interfaz.

Todo lo comentado no es más que una repetición de los objetivos propuestos, en cuantos al funcionamiento de la aplicación. Existen varios modos de hacer las cosas, en este caso de recoger los datos que se albergan en la base de datos y representarlos en la interfaz principal.

Es en este punto cuando surge el gran problema que el desarrollador ha tenido en el desarrollo de este proyecto. Las aplicación para Firefox OS pueden ser de varios tipos, web, privilegiadas o certificadas, según el tipo de elementos hardware a los que accedan durante el uso de la app a desarrollar. Cada tipo de aplicación debe cumplir una serie de restricciones de seguridad impuestas por Mozilla [48].

- web - Una app regular. Los permisos se limitan en el listado del manifest en el campo de permissions. Si no especificas el campo type en el manifest, web

es el tipo predeterminado.

- privileged - Una Open Web App autenticada que ha sido aprobada por una tienda de aplicaciones como el Firefox OS Marketplace. Algunas APIs de seguridad o privacidad sólo están disponibles para apps tipo privileged. Su propósito es dar más seguridad a un usuario para aplicaciones que desean tener acceso a dicha API. Hay un packaged app (todos los recursos en un archivo ZIP) que tiene las siguientes características adicionales:
    - Aprobadas por una tienda de aplicaciones tras una revisión de código.
    - Tiene un app manifest firmado por la tienda de aplicaciones.
    - Usa Content Security Policy [49] e implementa otros aspectos relacionados a la seguridad.
- certified - Una Open Web App que está destinada a una función crítica de sistema como el marcado del teléfonos o el sistema de configuración de apps en un smartphone. No está destinado a aplicaciones de terceros para la tienda de aplicaciones. Los scripts remotos accedidos desde la app están prohibidos, así como la carga interna de scripts como onclick="", onload="" u onmouseover="", carga de estilos directamente en html... y una serie de restricciones más que se detallan en la siguiente página de Mozilla Foundation para Mozilla Developer Networks [50].

En el caso de FM Radio RDS, esta necesita acceder al chip radio FM utilizando las webAPI, por lo que necesariamente había que desarrollar una aplicación certificada, la más restrictiva de todas en cuanto a medidas de seguridad. Pero el uso de este tipo de aplicaciones supone una serie de inconvenientes como se han citado en el párrafo anterior.

Inicialmente, se decidió representar esos datos a través de una llamada XHR al servidor, donde este imprimía todo el listado de emisiones obtenidas para luego ser pulsadas y escuchadas. Cada elemento representado en la interfaz llevaba adscrito una serie de eventos onclick="" que daban la funcionalidad de cambio de frecuencia escuchada, renovación de la interfaz y más. Si la app fuera privilegiada no supondría ningún problema, pero no era el caso. Necesariamente la webapp desarrollada debía ser de tipo certificada para poder gestionar la radio FM, por lo que hubo que buscar una solución a este gran inconveniente, que conllevó un retraso de varias semanas en la elaboración del código.

Las Firefox OS apps aún no están extendidas en el mundo de los desarrolladores de aplicaciones para móviles. Por ello, a pesar de la buena documentación aportada por Mozilla en sus entornos web, no hay muchas

entradas abiertas en sitios como StackOverFlow, con preguntas y soluciones propuestas por otros desarrolladores, por lo que la búsqueda de soluciones a problemas tan particulares como evitar la CSP fue más costoso y largo de lo esperado.

Las frecuentes reuniones con el tutor del trabajo de fin de grado fueron fructíferas y ayudaron a solucionarlo. Visto que el contenido cargado remotamente, por mucho que contuviera eventos onclick asignados no iban a ejecutarse por temas de seguridad, se buscó una solución óptima que consiste en crear los elementos dinámicamente gracias a JavaScript. Se hace una petición al servidor acerca de los datos que hay que representar en ese lugar en ese instante, se reciben gracias a $.getJSON() - $each() - jQuery e internamente en esa llamada se crean los elementos que aparecen en la interfaz. Al mismo tiempo, se generan los eventos addEventListener de cada elemento creado, asignados a su id única, que posibilitarán responder a gestos como la pulsación en la pantalla táctil del dispositivo.

Este fragmento de código que se presenta a continuación, contiene lo que se está generando en cada iteración del contenido que devuelve del servidor en la llamada del ejemplo mostrado en el punto $.getJSON() - $each() - jQuery. Cada elemento que se genera, contiene las etiquetas html necesarias con su atributos correspondientes.

```
nueva_linea =
  "<div class=\"cada_emisora align-center sin_margin_bottom\" id="+
emisoras.cod_emisora_json + ">"+
  "<div class=\"info_vista_secund\">" +
  "<a id=\"emisora_"+ emisoras.cod_emisora_json +"\">"+
  "<img class=\"icono_ppal_alineado\"
src=\"./stylesheets/mine/icono_pua/icono_blanco_pua.png\" alt=\"\" />"+
  "</a>"+
  "<a class=\"texto_ppal_alineado\" id=\"emision_"+ emisoras.cod_emision_json +"\">"+
  emisoras.nombre_emisora_json +
  "</a>" + "</div>" + "</div>";
```

Y este nuevo fragmento, será el evento onclick que se genera dinámicamente en función del id del elemento creado. Internamente, realiza la llamada al evento seleccionar_emision(...) al que se le pasan un conjunto de parámetros y a la función ver_si_es_favoritos().

```
document.getElementById("emision_"+emisoras.cod_emision_json).addEventListener("cl
ick",function(){
  seleccionar_emision(ultima_emision_pulsada, ultima_frecuncia_pulsada,
emisoras.cod_emisora_json, emisoras.nombre_emisora_json);
  ver_si_es_favoritos();
});
```

Del mismo modo, sucede con la carga de script remotos como el de Google Maps para la obtención de la ciudad donde se encuentra el usuario según sus coordenadas. Hubo que buscar alternativas para la no ejecución de estos scripts remotos, y como se explica en Geoposicionamiento inverso, gracias a OpenStreetMap el desarrollador pudo salvar dichas adversidades.

Y lo mismo sucedió con el caso de uso Valorar emisión con el slider horizontal que aparece en la interfaz Ilustración 61: UI_logged_puntuacion. En un principio, el desarrollo se basaba en escuchar los movimientos de la barra, y cuando el usuario modificase la posición se llamaba mediante la función onchange(...) al método que escribía el resultado en pantalla.

```
var text2_2 =
 '<input type="range" min="1" max="5" step="1" id="id_slide_puntuaciones"
          onchange="valorSliderPuntuacion(value)">'+
  '<strong><output for="id_slide_puntuaciones" id="txtBD_puntuaciones_notificacion
          _valor" class="texto_destacado"></output></strong>';
```

Finalmente, se consigue el resultado esperado Ilustración 62: UI_logged _puntuacion_resultado con el siguiente código.

```
 var callback_notify = function(result){
          if(result){
 nuevo_aniadir_puntuacion(document.getElementById("id_slide_puntuaciones").value);
          }
 };
 Tako.Notification.confirm("signal",text1, text2 + text2_2, text3, text4, callback_notify);
 if (document.getElementById("id_slide_puntuaciones")){
  $("#id_slide_puntuaciones").change(function(event) {
  $("#id_slide_puntuaciones").append(valorSliderPuntuacion(document.
                    getElementById("id_slide_puntuaciones").value));    }); }
```

Problemas de inestabilidad de la aplicación, con una carga incompleta de varios apartados aparentemente bien codificados y la solución del problema

A medida que la aplicación iba aumentando por el continuo desarrollo de funcionalidades, el número de lineas de código escritos crecía paralelamente, así como los problemas de impresión de datos en pantalla y estabilidad de la interfaz desarrollada.

El framework TakoJS, por defecto, permite al desarrollador elegir entre diferentes transiciones para alternar entre articles y entre sections, y en FM Radio RDS se pensó que podía ser interesante utilizarlo. Estas elementos podían estar en un mismo archivo html o dividido en varios a los que luego se les llamaba internamente. Por ello, se divide el código en varios archivos html, quedando la estructura de archivos como la mostrada en Ilustración 16.

Pero, desgraciadamente, la estabilidad de la webapp en el terminal con Firefox OS no era ni remotamente decente, por lo que se buscaron soluciones para superar esto. Curiosamente, la versión web de la aplicación ejecutada desde un navegador corriendo en un terminal Android (v5.1.1 y Google Chrome & MozillaFirefox) y en un PC (Windows 8.1 y Google Chrome & Mozilla Firefox) funciona perfectamente, sin ese "lag" que parece sufrir en el dispositivo Geeksphone.

Entre una de las tantas pruebas realizadas, el desarrollador pensó en unificar casi todas las section y articles en un mismo archivo, article_index.html y el resultado fue el mejor posible, ya que la usabilidad mejoró exponencialmente olvidando esos saltos y cuadros negros descritos. La gran desventaja de este cambio ha sido la perdida de transiciones entre sections y articles, pero ganando estabilidad y robustez a la hora de manejar la app.

Este gran cambio que se comenta en este apartado, ha provocado que haya que modificar algunas partes del código, llamadas entre sections y articles, modificación de variables y de Estructura del trabajo., etc., y por lo tanto puede que haya partes del trabajo que ya estaban desarrolladas y plasmadas en este documento que no han sido actualizadas, por el mero hecho de dejar constancia de las diferencias encontradas. Sobre todo, puede que en los casos de uso extendidos y en los diagramas de secuencia alguna variable no coincida completamente en nombre con el indicado, pero será una similar que no afecta al entendimiento del trabajo.

Integración completa de RFOS en FM Radio RDS, sin crear la aplicación independiente prevista

En la descripción inicial del trabajo, se hacía referencia a que se iba a desarrollar una aplicación independiente para ir añadiendo las emisiones en cada posición a la aplicación. Finalmente y durante el desarrollo del mismo, se ha pensado en que la mejor opción sea utilizar unicamente la funcionalidad integrada en la webapp final. Así, no haría falta realizar un mantenimiento y desarrollo de dos productos diferentes, ahorrando en tiempo que hay que dedicar a ello. Por ello, esta funcionalidad ha quedado reservada al método hacer escaneo.

7. Verificación y evaluación

En este apartado se detallan algunas de las pruebas realizadas a lo largo de todo el proceso de desarrollo. Gracias a la constancia y exhaustiva atención a la hora de realizar las pruebas, se han podido ir identificando y subsanando los errores encontrados por cada una de las funcionalidades implementadas.

Durante el largo periodo de desarrollo del código de la aplicación y de elaboración de la documentación, se han realizado infinidad de pruebas con la finalidad de obtener un código lo más eficaz y eficiente posible.

Las pruebas que se presentan a continuación son una muestra de todas las realizadas. Se presentan las pruebas a las funcionalidades más características.

1. Registrarse

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1	Registrar un usuario teniendo errores en los campos a introducir (sin realizarse la conexión a la base de datos).	Un mensaje de aviso indicando que hay fallos.	Un mensaje de aviso indicando que hay fallos.	Correcto.
2	Registrar un usuario que no existe en el sistema.	Un mensaje de aviso indicando que se añade correctamente.	Un mensaje de aviso indicando que se añade correctamente.	Correcto.
3	Registrar un usuario que ya existe en el sistema (el email y/o el nombre).	Un mensaje de error indicando que ya existe.	Un mensaje de error indicando que ya existe.	Correcto.

Tabla 4: Pruebas de Registrarse.

2. Identificarse

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1	Identificar un usuario teniendo errores en los campos a introducir (sin realizarse la conexión a la base de datos).	Un mensaje de aviso indicando que hay fallos.	Un mensaje de aviso indicando que hay fallos.	Correcto.
2	Identificar un usuario que no existe en el sistema.	Un mensaje de aviso indicando que no existe en el sistema.	Un mensaje de aviso indicando que no existe en el sistema.	Correcto.
3.1	Identificar un usuario que existe en el sistema pero está dado de baja	Un mensaje de aviso indicando que está dado de baja y que debe reidentificarse.	Un mensaje de aviso indicando que está dado de baja y que debe reidentificarse.	Incorrecto*.
3.2	Identificar un usuario que existe en el sistema pero está dado de baja	Un mensaje de aviso indicando que está dado de baja y que debe reidentificarse.	Un mensaje de aviso indicando que está dado de baja y que debe reidentificarse.	Correcto.

Tabla 5: Pruebas de Identificarse

*En el caso 3.1, inicialmente la app devolvía un mensaje informando que todo era correcto, y se había realizado correctamente la identificación. Pero al hacer una comprobación manual del contenido de la BD, el desarrollador se dio cuenta que el usuario estaba de baja. Faltaba controlar en el código de identificación que el usuario estuviera de baja o activo.

Se mejoró el código para contemplar esa posibilidad como se demuestra en el caso 3.2.

3. Ver información y datos de contacto del desarrollador y de la aplicación

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1	Entrar en el apartado de información y pulsar en un enlace para acceder al perfil. Se accederá a las aplicaciones nativas del sistema operativo.	Se accede a la sección y se redirecciona a otras app al pulsar en los botones e iconos.	Se accede a la sección y se redirecciona a otras app al pulsar en los botones e iconos.	Correcto.

Tabla 6: Pruebas de Ver información y datos de contacto del desarrollador y de la aplicación

4. Ver comentarios de usuarios cercanos

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1.1	Ver comentarios cercanos de otros usuarios.	Un listado de comentarios cercanos, ordenados por fecha	Un listado completo de comentarios realizados.	Incorrecto*.
1.2	Ver comentarios cercanos de otros usuarios.	Un listado de comentarios cercanos, ordenados por fecha	Un listado de comentarios cercanos, ordenados por fecha	Correcto.

Tabla 7: Pruebas de Ver comentarios de usuarios cercanos

*En el caso 1.1, inicialmente la app devolvía el listado completo de comentarios, cuando sólo se quería los escritos y añadidos en posiciones cercanas. Se mejoró el código para contemplar esa condición y que devolviera únicamente los comentarios cercanos a la posición del usuario como se contempla en el caso 1.2.

5. Ver emisoras con emisiones cercanas

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1.1	Ver el listado de las emisoras, ordenadas por nombre, en la pantalla principal.	Un listado de nombres de emisoras.	Vacío.	Incorrecto.
1.2	Después de corregir parte del código, ver el listado de las emisoras, ordenadas por nombre, en la pantalla principal.	Un listado de nombres de emisoras.	Un listado de emisoras esperadas.	Correcto.
2	Ver el listado de las emisoras, ordenadas por nombre, en la pantalla principal.	Un listado de nombres de emisoras.	Vacío.	Correcto*.
3	Ver el listado de las emisoras, ordenadas por nombre, en la pantalla principal, sin tener configurado el dispositivo para compartir su ubicación.	Al iniciar la aplicación, un mensaje de aviso indicando que no dispone de la ubicación activada.	Al iniciar la aplicación, un mensaje de aviso indicando que no dispone de la ubicación activada.	Correcto.

Tabla 8: Pruebas de Ver emisoras con emisiones cercanas

*En el primer caso, hasta lograr el resultado esperado, se producen varias situaciones inestables. Como se menciona en el 1.1, el listado de emisoras es vacío a pesar de no tener aparentemente nada incorrecto. Una vez revisado el código, parece que obtenemos el resultado esperado en forma de listado de emisiones cercanas.

En el caso 2, al no haber ninguna emisora cercana al usuario y/o dispositivo que esté añadida al sistema, el listado será vacío. El funcionamiento esperado es totalmente correcto, pero se precisa mejorar la lógica del programa añadiendo un texto informativo. Se corrige ese funcionamiento recomendando al usuario que añada emisoras y/o emisiones al sistema en esa posición huérfana de ellas.

6. Elegir emisora al pulsar lista manualmente

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1.1	Seleccionar un elemento del listado.	El elemento queda marcado en el listado y se actualiza la emisión que se está escuchando.	No se pulsa el elemento.	Incorrecto*.
1.2	Seleccionar un elemento del listado.	El elemento queda marcado en el listado y se actualiza la emisión que se está escuchando.	El elemento queda marcado en el listado y se actualiza la emisión que se está escuchando.	Correcto.

Tabla 9: Pruebas de Elegir emisora al pulsar lista manualmente

*En el caso 1.1, la aplicación falla por lo explicado en el punto Generación de contenido dinámicamente y políticas de seguridad (CSP). Importante paso solucionar esto y dar lugar al caso 1.2.

7. RDS

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1.1	Acceder a las emisiones cercanas que hay de una misma emisora (que estén registradas en el sistema).	Un listado con una o varias emisiones de una misma emisora.	Un mensaje de carga constante que no se puede cerrar.	Incorrecto*.
1.2	Acceder a las emisiones cercanas que hay de una misma emisora (que estén registradas en el sistema).	Un listado con una o varias emisiones de una misma emisora.	Un listado con una o varias emisiones de una misma emisora.	Correcto.

Tabla 10: Pruebas de Elegir emisora al pulsar lista manualmente

*En el caso 1.1, la aplicación falla mostrando un mensaje que no se puede cerrar. En la búsqueda del error, se aprecia como el sistema tiene cierto retardo al escribir el valor de las variables en el almacenamiento local, por ello el sistema es inestable. Se corrige el código para intentar paliarlo, en el caso 1.2 ya funciona .

8. Elegir emisión al pulsar lista manualmente

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1.1	Seleccionar un elemento del listado.	El elemento queda marcado en el listado y se actualiza la emisión que se está escuchando.	No se pulsa el elemento.	Incorrecto*.
1.2	Seleccionar un elemento del listado.	El elemento queda marcado en el listado y se actualiza la emisión que se está escuchando.	El elemento queda marcado en el listado y se actualiza la emisión que se está escuchando.	Correcto.

Tabla 11: Pruebas de Elegir emisión al pulsar lista manualmente

*En el caso 1.1, la aplicación falla por lo explicado en el punto Generación de contenido dinámicamente y políticas de seguridad (CSP). Importante paso solucionar esto y dar lugar al caso 2.2.

9. Play / Stop

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1.1	Pulsar botón Play para escuchar una emisora de radio	Reproducción de una emisión de radio FM a través de los auriculares conectados	Nada, no se sintoniza ni se escucha nada	Incorrecto

Tabla 12: Pruebas de Play / Stop - Parte 1

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1.1	Pulsar botón Play para escuchar una emisora de radio	Reproducción de una emisión de radio FM a través de los auriculares conectados	Nada, no se sintoniza ni se escucha nada	Incorrecto
1.2	Tras hacer cambios en el codigo, pulsar botón Play para escuchar una emisora de radio	Reproducción de una emisión de radio FM a través de los auriculares conectados	Reproducción de una emisión de radio FM a través de los auriculares conectados	Correcto
2.1	Pulsar botón Pause para parar la reproducción de una emisora de radio	Parar la reproducción de una emisión de radio FM	Nada, no ocurre nada al pulsar el botón	Incorrecto
2.2	Tras hacer cambios en el codigo, pulsar botón Play para escuchar una emisora de radio	Parar la reproducción de una emisión de radio FM y cambiar los colores de los botones	Se para la reproducción de la radio pero no hay cambio de colores en los botones	Incorrecto
2.3	Tras hacer cambios de nuevo en el codigo, pulsar botón Play para escuchar una emisora de radio	Parar la reproducción de una emisión de radio FM y cambiar los colores de los botones	Parar la reproducción de una emisión de radio FM y cambiar los colores de los botones	Correcto

Tabla 13: Pruebas de Play / Stop - Parte 2

En el caso 1.1, al producirse un fallo, se intenta subsanar para dar lugar al caso 1.2. Se corrige y mejora el código controlando esa situación, ya que no se estaba consiguiendo utilizar la radio FM.

En el caso 2.1, al producirse un fallo, se intenta subsanar para dar lugar al

caso 2.2. Se corrige y mejora el código controlando que se pueda parar la radio, pero volvía a ser incompleto al no realizar el cambio de colores en los botones pulsados. En el punto 2.3, se subsana este error para obtener el funcionamiento esperado y deseado.

10. Valorar emisión

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1.1	Se pulsa el botón de "Puntuar" según se inicia la app por primera vez, sin haber seleccionado una emisora.	Mensaje de aviso que debe seleccionar la emisora que se quiere puntuar.	Ninguno.	Incorrecto*.
1.2	Se pulsa el botón de "Puntuar". Según se inicia la app por primera vez, no se produce error.	Mensaje de aviso que debe seleccionar la emisora que se quiere puntuar.	Mensaje de aviso que debe seleccionar la emisora que se quiere puntuar.	Correcto.
2	Se pulsa el botón de "Puntuar" con una emisora seleccionada.	Mensaje de aviso donde elegir la puntuación.	Mensaje de aviso donde elegir la puntuación.	Correcto.
3.1	Se mueve el slider para elegir la puntuación y se pulsa en "Puntuar"	Aparece escrita la puntuación que se le quiere dar. Al pulsar en enviar, se cierra el mensaje.	No aparece el texto en el mensaje de la puntuación que se ha elegido.	Incorrecto*.
3.2	Se mueve el slider para elegir la puntuación y se pulsa en "Puntuar"	Aparece escrita la puntuación que se le quiere dar. Al pulsar en enviar, se cierra el mensaje.	Aparece el texto en el mensaje de la puntuación que se ha elegido y se envía.	Correcto.

Tabla 14: Pruebas de Valorar emisión

*En el caso 1.1, al producirse un fallo, se intenta subsanar para dar lugar al caso 1.2. Se corrige y mejora el código controlando esa situación.

*En el caso 3.1, la aplicación falla por lo explicado en Generación de contenido dinámicamente y políticas de seguridad (CSP). Solucionado en 3.2.

11. Añadir emisión manualmente

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1	Se selecciona una emisora y se introduce una frecuencia valida.	Mensaje de aviso indicando que todo es correcto.	Mensaje de aviso indicando que todo es correcto.	Correcto
2	Se selecciona una emisora y se introduce una frecuencia ya registrada	Mensaje de aviso indicando que ya existe en esa posición esa emisión registrada.	Mensaje de aviso indicando que ya existe en esa posición esa emisión registrada.	Correcto
3	Se selecciona una emisora y se introduce una frecuencia no valida.	Mensaje de aviso indicando que algo es erroneo.	Mensaje de aviso indicando que algo es erroneo.	Correcto

Tabla 15: Pruebas de Añadir emision manualmente

12. Poner favorita emisora

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1	Se pulsa en el botón de favoritos teniendo una emisión seleccionada y se añade al listado de favoritas del usuario.	Se modifica el botón, se rellena la estrella.	Se modifica el botón, se rellena la estrella.	Correcto.

Tabla 16: Pruebas de Poner favorita emisora

13. Ver información completa de la emisora

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1	Ver información de la emisora pulsando en el icono del listado principal.	Se muestra en la interfaz datos de la emisora y sus emisiones.	Se muestra en la interfaz datos de la emisora y sus emisiones.	Correcto.

Tabla 17: Pruebas de Ver información completa de la emisora

14. Hacer escaneo RFOS

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1.1	Se pulsa en el botón Buscar siguiente y se obtiene la siguiente frecuencia.	Una nueva frecuencia.	La frecuencia se queda en null.	Incorrecto
1.2	Corrección de pulsar el botón Buscar siguiente y obtener la siguiente frecuencia.	Una nueva frecuencia.	Una nueva frecuencia.	Correcto
2	Se pulsa en el botón Buscar siguiente y se obtiene la siguiente frecuencia y la emisión que coincida con los datos pasados.	Una nueva frecuencia y si hay emisión coincidente, lo mostraría	Una nueva frecuencia y si hay emisión coincidente, lo mostraría	Correcto

Tabla 18: Pruebas de Hacer escaneo RFOS

En el caso 1.1, al producirse un fallo, se intenta subsanar para dar lugar al caso 1.2. Se corrige y mejora el código controlando esa situación. El error se subsanó controlando los segundos que se tardan en actualizar el objeto navigator.mozFM o navigator.mozFMRadio, ya que tarda 3-4 segundos en cambiar este valor.

15. Cambiar password

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1	Cambiar la contraseña teniendo errores en los campos a introducir (no se realiza la conexión a la BD).	Un mensaje de aviso indicando que hay fallos.	Un mensaje de aviso indicando que hay fallos.	Correcto.
2	Cambiar la contraseña de un usuario que no existe en el sistema.	Un mensaje de error indicando que no existe.	Un mensaje de error indicando que no existe.	Correcto.
3	Cambiar la contraseña de un usuario que si existe en el sistema.	Un mensaje de aviso indicando que todo está bien.	Un mensaje de aviso indicando que todo está bien.	Correcto.

Tabla 19: Pruebas de Cambiar password

16. Cerrar sesión

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1	Cerrar la sesión del usuario identificado.	Mensaje de aviso que pregunta que se desea hacer.	Mensaje de aviso que pregunta que se desea hacer.	Correcto.

Tabla 20: Pruebas de Cerrar sesión

17. Dar de baja usuario

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1	Dar de baja el usuario (el mismo con el que se está identificado)	Mensaje de aviso que indica que se da de baja .	Mensaje de aviso que indica que se da de baja .	Correcto.

Tabla 21: Pruebas de Dar de baja al usuario - Parte 1

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
2.1	Dar de baja el usuario (diferente usuario al que está identificado)	Mensaje de aviso que indica que se da de baja .	Mensaje de aviso que indica que se da de baja .	Incorrecto*.
2.2	Dar de baja el usuario (diferente usuario al que está identificado)	Mensaje que indica que no es el usuario al que se quiere dar baja.	Mensaje que indica que no es el usuario al que se quiere dar baja.	Correcto.

Tabla 22: Pruebas de Dar de baja al usuario - Parte 2

*En el caso 2.1, la aplicación no funciona como debiera, ya que un usuario ajeno no puede dar de baja cualquier usuario sólo con saber su email y password. En la elaboración de las pruebas el desarrollador se da cuenta de este agujero y lo subsana, contemplando esa situación como se ve en el caso 2.2.

18. Compartir

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1	Compartir un mensaje, accediendo desde la app a aplicaciones propias del sistema operativo movil.	Mensaje de aviso que pregunta que se desea hacer.	Mensaje de aviso que pregunta que se desea hacer.	Correcto.

Tabla 23: Pruebas de Compartir

19. Comentar sobre la app

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1	Escribir un comentario subjetivo sobre la app.	Mensaje de aviso que indica que se ha añadido correctamente.	Mensaje de aviso que indica que se ha añadido correctamente.	Correcto.

Tabla 24: Pruebas de Comentar sobre la app - Parte 1

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
2	Escribir un comentario subjetivo sobre la app (cantidad de palabras fuera de rango).	Mensaje de error avisando que el texto es incorrecto.	Mensaje de error avisando que el texto es incorrecto.	Correcto.
3	Escribir más de un comentario al día	Mensaje de error avisando que hay un tope diario.	Mensaje de error avisando que hay un tope diario.	Correcto.

Tabla 25: Pruebas de Comentar sobre la app. - Parte 2

20. Ver emisoras favoritas, emisoras y emisiones añadidas y valoraciones realizadas.

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1	Ver el listado correspondiente.	La información apropiada.	La información apropiada.	Correcto.

Tabla 26: Pruebas de Ver emisoras favoritas, emisoras y emisiones añadidas y las valoraciones.

21. Registrar una incidencia

Código	Descripción	Resultado esperado	Resultado obtenido	Observaciones
1	Agitar el dispositivo para registrar una incidencia.	Un mensaje que redirige a la sección de datos de contacto.	Un mensaje que redirige a la sección de datos de contacto.	Correcto.

Tabla 27: Pruebas de Registrar una incidencia.

8. Conclusiones y trabajo futuro

En este apartado se tratan las conclusiones del proyecto, tanto a nivel de gestión y de desarrollo, como las conclusiones finales desde un punto de vista crítico y personal, enfocando los resultados en primera persona.

Conclusiones en la gestión: planificación, riesgos y costes.

En cuanto a la planificación temporal realizada al comienzo del desarrollo de esta aplicación web, hay que comentar que las estimaciones eran muy halagüeñas pero el desarrollo del trabajo se ha retrasado en gran medida. Revisando la tabla de las horas estimadas una vez finalizado el proyecto, el desarrollador se da cuenta de las tareas que más tiempo han llevado completar y que tienen más desfase con lo estimado, como han sido la implementación del código de la app y la redacción de la memoria. Por lo que uno de los riesgos temidos, la planificación incorrecta, se ha cumplido.

Por una parte, la fase de aprendizaje ha sido bastante más larga de lo esperado. A lo largo de los estudios universitarios se han aprendido muchos conceptos sobre los lenguajes de programación que se han utilizado en este trabajo (JS, HTML5, CSS y PHP), pero el tamaño del proyecto que se tenía entre manos requería aprender conceptos más complejos, así como utilizar el framework elegido, TakoJS. Por otro lado, ha sido necesario aprender a acceder y a utilizar las APIs propias del sistema operativo móvil Firefox OS, para el acceso a la radio FM, al almacenamiento local de datos, al GPS y más elementos que están integrados en el dispositivo. Todo ello ha supuesto un tiempo de aprendizaje mayor de lo previsto.

La redacción de la memoria en algunos momentos ha sido una tarea larga y tediosa, sobre todo al completar detalladamente los anexos. Las reuniones con el director del trabajo han sido periódicas, así como el contacto a través del correo electrónico para solventar errores y despejar dudas sobre el formato, contenido del documento y temas de desarrollo de código. Por ello, la estimación inicial se queda corta respecto a las horas que realmente se han invertido.

En cuanto a los costes, ha sido imposible cumplir los plazos establecidos en la planificación debido a los imprevistos explicados, además de algún problema personal puntual. Esto ha ocasionado el aumento de horas necesarias para llevar a cabo el trabajo en comparación con las horas estimadas. A continuación, se muestra un gráfico y una tabla con las diferencias entre las horas estimadas y una aproximación de las horas reales invertidas en realizar el proyecto:

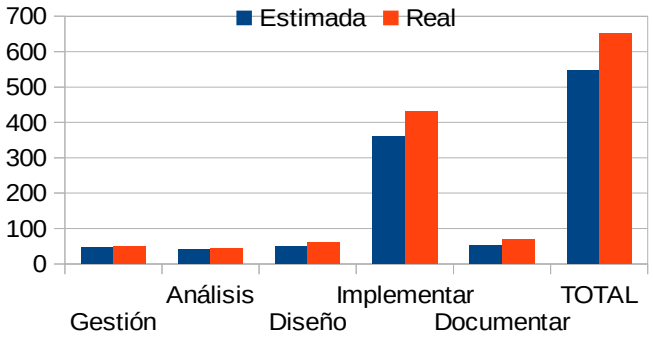


Ilustración 26: Gráfico comparativa: Horas Estimadas - Horas Reales

	Estimada	Real
Gestión	46,00	48,00
Análisis	40,00	44,00
Diseño	48,00	60,00
Implementar	360,00	430,00
Documentar	52,00	70,00
TOTAL	**546,00**	**652,00**

Ilustración 27: Tabla comparativa: Horas Estimadas - Horas Reales

Teniendo presente la Evaluación económica realizada al inicio del trabajo, hay que recalcular todos los costes con los datos de las horas reales aproximadas. Se pasan de 546 horas estimadas a 652 horas reales, 106 horas más, esto supone un aumento de 2650€ a los 13650€ previstos, dando un total de 16300€. Al aumentar las horas necesarias para llevar a cabo las tareas , hay que sumar también las nuevas amortizaciones del ordenador portátil (22,15€) y del dispositivo móvil (9,14€), repercutiendo todo ello en el coste total.

Gastos		
	Estimados	**Reales**
Personal	13.650 €	16.300€
Hardware	25,425 €	31,29
Software	0 €	0€
Otros	36 €	36€
Total	**13.711,425 €**	**16.367,29€**

Tabla 28: Tabla de gastos totales, estimados frente a reales.

De este modo, para recuperar la inversión, habría que volver a calcular los datos iniciales expuestos en la evaluación económica sobre las horas reales invertidas.

- Si el precio de la app fuera de 1 euro, el número mínimo de compradores para obtener beneficios debería ser 23381,84, casi 4000 descargas más.
- Si el desarrollador decide ponerlo a la venta por 5€, el número mínimo de compradores para obtener beneficios debería ser 4676,37, unas 700 descargas

más.

En ambos casos, los retrasos sufridos suponen la necesidad de aumentar considerablemente el número de descargas para recuperar la inversión, una tarea compleja dada la situación del mercado.

Como anécdota relacionada con la gestión de los riesgos de llevar a cabo un proyecto como este, quería destacar la importancia de tener un sistema de copias de seguridad de los datos, archivos y todo lo relacionado con el trabajo. En mi caso, periódicamente se ha realizado un BackUp en un directorio de mi ordenador personal, en un Servidor NAS dentro de la red doméstica, puntualmente en algún sistema de almacenamiento en la nube y en el servidor de la universidad al cual tengo acceso. Habitualmente, se ha estado trabajando sobre este último, albergando ahí el código de la webapp y varios archivos importantes para el desarrollador. Durante una jornada de trabajo puntual, el desarrollador quiso acceder al servidor de la universidad y se encontró con un mensaje de error, indicando que no se tenían permisos de acceso al directorio. Tras contactar con el tutor y exponer el problema, se descubrió que algún administrador del sistema había borrado la cuenta de usuario y podía haber eliminado también la carpeta. Afortunadamente, esto no fue así y se consiguió recuperar el acceso a este directorio virtual. Esta situación evidencia la importancia de disponer de un sistema de copias de seguridad efectivo y replicado en varios lugares, con la idea de evitar un desastre.

Conclusiones sobre el desarrollo: objetivos, explicación y justificación de los cambios respecto a la propuesta original.

En cuanto a los objetivos establecidos al comienzo del desarrollo de esta aplicación, se han cumplido todos. Uno de los principales era aprender a desarrollar aplicaciones para Firefox OS, así como para el resto de plataformas web y adquirir los conocimientos necesarios para poder afrontar el desarrollo de otros proyectos de dificultad similar en el futuro, y honestamente puedo decir que se ha conseguido.

Se han logrado desarrollar todas las funcionalidades que inicialmente se habían planteado en los objetivos (reproducción de radio FM a través del terminal en una aplicación independiente, disponer de las emisoras y sus emisiones cercanas a la posición del dispositivo, realizar el escaneo total de las emisiones que se sintonizan en cada posición, tablón de comentarios y panel personal, etc.), así como alguna extra (funcionamiento parcial del modo offline, rds automático, aplicación web multiplataforma).

Entre los objetivos iniciales citados, la mayoría de estas funcionalidades se han podido desarrollar bajo la idea inicial que se tenía, plasmada en los primeros bocetos de las interfaces de la aplicación.

No es el caso del escaneo de las emisoras, por ejemplo. La idea inicial era obtener todas las emisiones que se encontrasen según la posición del dispositivo y mostrarlas en un listado, dejando pendiente al usuario la elección del nombre de la emisora que le correspondería a cada frecuencia encontrada. Así, se podrían almacenar muchos registros de una vez. Pero, durante el desarrollo de esa funcionalidad, el desarrollador descubre que internamente, cuando se realiza un cambio de frecuencia, el objeto myRadioFM tarda entre 2 y 3 segundos en refrescar el valor de la variable frecuencia, el tiempo que el dispositivo necesita para sintonizar y realizar el cambio de ésta. La solución por la que se opta es hacer la búsqueda individual y lineal de las emisiones a través de todo el ancho del dial; y cada vez que se encuentre una frecuencia con sintonía, dar la posibilidad al usuario de que elija la emisora a la que esa frecuencia corresponde y añadirla al sistema. Se muestran para cada frecuencia encontrada qué emisiones hay ya almacenadas en esa posición, y así ayudar al usuario y no duplicar contenido.

Otra de las decisiones tomadas con respecto a las ideas iniciales que se tenían ha sido suprimir los botones de "Siguiente emisión" y "Emisión anterior". En toda aplicación de radio FM estándar, existe una opción de seleccionar la siguiente emisión a la que se está escuchando, sin importar qué emisora sea, simplemente por probar suerte. En este caso, se ha decidido dar importancia a las emisoras y por ello, el listado aparece ordenado de modo ascendente alfabéticamente por el nombre de cada una de ellas. La funcionalidad de búsqueda de la siguiente emisión se encuentra dentro del menú "Realizar escaneo", pero siempre con la intención de poder completar la base de datos del sistema con las emisiones nuevas que no estén ya registradas.

En cuanto a los futuros usuarios de la webapp, hace falta tiempo para ver si de verdad la aplicación aporta utilidad en el día a día y comprobar su respuesta con un elevado número de usuarios. Realmente, lo necesario es que el sistema operativo crezca en número de usuarios y se asiente en el mercado.

Problemas durante la realización del TFG y aspectos relacionados con el mantenimiento del trabajo desarrollado.

Lo más difícil para el desarrollador ha sido cumplir con la estimación temporal del proyecto. En un principio, se había puesto como fecha de

presentación la convocatoria de Marzo del 2015 o cómo tarde Mayo del 2015, pero finalmente se retrasó hasta la convocatoria de Julio del mismo año por no llegar a tiempo con los plazos estimados. En defensa del desarrollador, hay que reseñar que se ha necesitado un número de semanas hasta que se ha familiarizado con el sistema operativo móvil, se ha encontrado un entorno de trabajo que se adaptara a las necesidades de la webapp a desarrollar y hasta que uno mismo se ha centrado en la elaboración del TFG.

Todos los problemas sufridos durante el desarrollo de este trabajo se han expuesto en Problemas y las soluciones del apartado Desarrollo.

Sobre el mantenimiento del trabajo desarrollado, se han introducido elementos que hacen que los propios usuarios de la webapp sean en parte responsables de que el sistema funcione mejor. Por ejemplo, las valoraciones de las emisiones es una tarea fundamental a la hora de obtener el listado de emisiones óptimas cercanas en cada posición. Se espera una aportación objetiva y acertada de las valoraciones que se realicen. En el menú RDS, se pueden ver las puntuaciones de cada emisión. Si una emisión dispone de valoraciones negativas, se avisará al usuario con un icono de atención.

En todo momento se ha intentando controlar que no exista redundancia de datos. A la hora de añadir una emisora, se intenta evitar que se agregue una con el mismo nombre. Con las emisiones, se tiene mucho más control si cabe. Una emisión, con una frecuencia y un nombre de emisora, no puede repetirse en una distancia en un radio de 0'3º, hablando en términos de coordenadas que equivale a unos 40km, aproximadamente. Si un usuario quisiera añadir una emisión manualmente incumpliendo esta condición, la aplicación mostraría un mensaje de aviso de error. En el apartado de escaneo, el sistema le informa al usuario de la emisora que está registrada en esas coordenadas con esa misma frecuencia y si aún así decide agregarla, la app se lo impedirá.

Con los comentarios se ha tomado la decisión de que un usuario sólo pueda escribir un comentario al día. De ese modo, se evita de un modo básico que un usuario haga spam dentro de la aplicación.

En cuanto al control de acceso de usuarios, al registrarse se comprueba que no exista ni el email ni el nombre de usuario ya en el sistema, que todos los campos sean válidos y que no esté previamente registrado en el sistema. En la identificación de usuario, se comprueba que un usuario esté activo (que no se haya dado de baja) para poder identificarse, que exista su email y que su contraseña coincida con la dada al registrarse. Si un usuario estuviera inactivo, se le avisará y se le pedirá de

nuevo identificarse para poder acceder a las funcionalidades del usuario identificado.

Se puede dar el caso que un usuario esté identificado en dos dispositivos simultáneamente y en uno de ellos dé de baja su cuenta. En el otro dispositivo, en el momento que fuera a hacer algo que implique añadir registros en la base de datos, le saltará una notificación indicando que ya no es usuario y que debe volver a identificarse en el sistema.

Este tipo de aplicación conlleva un mantenimiento del contenido de las bases de datos y una mejora continua de la webapp por parte del desarrollador, a pesar de introducir los elementos de control comentados. Así que en el futuro, si la aplicación llega a un número de usuarios elevado, se pensaría en añadir sistemas de control mejorados para facilitar la labor del administrador como poder eliminar emisoras y/o emisiones por parte de los usuarios o realizar recomendaciones.

Conclusiones sobre las líneas futuras

En el apartado de líneas futuras se presentan en forma de listado algunas de las funcionalidades que se decidieron posponer para llevarlas a cabo en un futuro. Están ordenadas de menor a mayor relevancia en base al criterio del desarrollador:

- La opción de recuperar contraseña si el usuario no la recuerda. No supondría una gran labor de desarrollo y carga de trabajo. Se completaría la gestión de usuarios en su totalidad siendo muy beneficioso para los ellos .
- Cuando un usuario borra su cuenta, añadir una pregunta de por qué está abandonando FM Radio RDS, obteniendo así un feedback directo del usuario. No supondría una larga labor de desarrollo y carga de trabajo. Los usuarios no parece que se beneficiarían en gran medida; el desarrollador en cambio, obtendría información directa de los usuarios que podría servirle para cambiar y/o mejorar la aplicación.
- Dar opción a que los usuarios puedan modificar los datos personales de su perfil. Supondría una carga de trabajo baja y beneficioso para los usuarios del sistema.
- Al registrarse un usuario, poder añadir una imagen y que posteriormente pudiera modificarla en su perfil. Supondría una carga de trabajo baja y beneficioso para los usuarios del sistema. Muy enlazado con la funcionalidad de dar un aspecto más social a la webapp.
- Ver el histórico de radios escuchadas, así como estadísticas de uso. Supondría

una carga de trabajo baja y beneficioso para los usuarios del sistema para ver sus escuchas de radio a través de la aplicación.

- Añadir una guía informativa sobre la programación diaria de cada emisora. Esta es una de las líneas futuras que más trabajo puede suponer para el desarrollador, dado que el número de emisoras total es inabarcable. Se buscaría una manera de hacer partícipe al usuario para introducir información sobre la programación de cada cadena de radio o se intentaría contactar con las emisoras para que facilitasen esta información, accediendo quizá a algún recurso web donde esta información esté ya albergada. Para el usuario, tener disponible la información de la programación supondría saber en cada momento que podría escuchar, algo que me parece un gran avance.
- Darle un enfoque más social a la aplicación, con un tablón donde cada usuario pudiera publicar su actividad, sus emisoras favoritas, comentarios sobre cada emisión, etc. No es una de las prioridades para el desarrollador ya que la naturaleza de la aplicación se estaría perdiendo. El coste de realizar estos cambios implicaría reimplementar muchas partes del código y replantearse el sentido de algún apartado como el listado de comentarios. Aún así, es una posible línea futura a contemplar para ganar usuarios, dado el auge de las redes sociales.
- Añadir publicidad personalizada de las emisoras de radio que el usuario suele escuchar. El coste en cuanto a tiempo de trabajo se supone que es alto y complejo, y con nulos beneficios para los usuarios al tener que soportar molestos mensajes de publicidad durante el uso de la aplicación. Sin embargo, la inclusión de publicidad supondría una posible fuente de ingresos que el desarrollador no podría despreciar. Llegados a ese punto, quizá se replantearía la opción de crear dos versiones de la misma aplicación, una de pago libre de publicidad y otra gratuita con publicidad.
- Los cambios y mejoras en la interfaz son la última línea futura que se tiene pensada. Supondría una carga de trabajo media y constante, con beneficios para los usuarios al ver como se producen mejoras en el diseño de la app.

Conclusiones desde un punto de vista crítico y personal.

En mi caso, desde el punto de vista de usuario, aún sin ser completamente objetivo, con FM Radio RDS se ha dado solución a un vacío que existía con respecto a las aplicaciones de radio FM estándar. Se consigue una manera más moderna y práctica de escuchar la radio, al utilizar herramientas que en este

momento están al alcance de los usuarios, como la conexión a Internet y la geoposición.

Desde el punto de vista del desarrollador, la realización de este trabajo de fin de grado ha resultado ser un gran paso en muchos sentidos. Se ha promovido el auto-aprendizaje y búsqueda de soluciones continuas, se ha mejorado y aumentado la capacidad de esfuerzo, investigación y ganas de superarse uno mismo, a pesar de las dificultades encontradas durante la realización de este trabajo y que han sido señaladas con anterioridad.

Resulta gratificante haber sido capaz de desarrollar desde el inicio una idea propia y realizarla satisfactoriamente, dando así por finalizado este trabajo de fin de grado. Así mismo, se ha conseguido la motivación necesaria para que este proyecto no termine en este punto, sino pensar en varias líneas futuras ya comentadas. La realización de este proyecto sienta las bases para poder afrontar nuevos proyectos en los que poder utilizar todas las habilidades y conocimientos aprendidos.

El desarrollador se compromete a estar al tanto de las novedades que en un futuro vayan surgiendo en este mundo tan cambiante, y sobre todo en el ámbito tecnológico, en torno al acceso a las APIs de radio FM en los dispositivos no Firefox OS. En el momento que esto fuera posible, se intentará adaptar la aplicación desarrollada al resto de plataformas. Hasta ese momento, el usuario no Firefox OS deberá conformarse con la versión web a modo de guía informativa de las emisoras cercanas a su posición.

9. Bibliografía

Imágenes

1. Arquitectura aplicación Cliente Servidor. "All Firefox OS" en allfirefoxos.com. Accedido el 2015/03/30. http://allfirefoxos.com/ y "Rizalkurnia" en rizalkurnia.com. Accedido el 2015/03/30. http://rizalkurnia.com/pengertian-database-server/. La versión presentada en este trabajo es una modificación usando las dos ilustraciones que aparecen en las páginas indicadas a través de las URLs. [Imagen 1]

2. Max Castleman, "A Look at the Firefox OS" en neongoldfish.com. Creado el 2015/08/01. Accedido el 2015/01/26. http://www.neongoldfish.com/blog/general/a-look-at-the-firefox-os/.[Imagen 2]

3. Roberto Casas, "[CES 2015] Mozilla planea llevar Firefox OS a más dispositivos" en mundofirefoxos.com. Creado el 2015/01/08. Accedido el 2015/01/08. http://www.mundofirefoxos.com/2015/01/ces-2015-mozilla-planea-llevar-firefox_7.html. [Imagen 3]

4. Radio Nacional de España S.A, "RDS-Radio Data System" en rtve.es. Creado el 2005/03/11. Accedido el 2014/11/18. http://www.rtve.es/rne/emisoras/rds.htm. La versión presentada en este trabajo es una modificación de dos ilustraciones que aparecen en la página indicada a través de la URL. [Imagen 4]

Libros

Julie C. Meloni. Programación: HTML5, CSS y JavaScript. España. Editorial Anaya Multimedia-SAMS (Grupo Anaya, S.A.), 2012. ISBN:978-84-415-3193-2. Consultado por primera ves el 2014/03/10.

Referencias web

1. Radio Nacional de España S.A, "RDS-Radio Data System" en rtve.es. Creado el 2005/03/11. Accedido el 2014/11/18. http://www.rtve.es/rne/emisoras/rds.htm.

2. Mozilla Developer Network, "Firefox OS" en MDN. Creado el –. Accedido el 2014/09/20. https://developer.mozilla.org/es/Firefox_OS.

3. Mozilla Foundation, "Firefox OS" en Mozilla Firefox Web. Creado el –. Accedido el 2014/09/20. https://www.mozilla.org/en-US/firefox/os/.

4. Mozilla Developer Network and individual contributors, "Gaia" en developer.mozilla.org. Creado el –. Accedido el 2015/01/08. https://developer.mozilla.org/en-US/Firefox_OS/Platform/Gaia.

5. Mozilla Developer Network and individual contributors, "Gecko" en developer.mozilla.org. Creado el –. Accedido el 2015/01/08. https://developer.mozilla.org/en-US/docs/Mozilla/Gecko.

6. Mozilla Developer Network and individual contributors, "Gonk" en developer.mozilla.org. Creado el –. Accedido el 2015/01/08. https://developer.mozilla.org/en-US/Firefox_OS/Platform/Gonk.

7. Mozilla Foundation, "mozillawiki" en wiki.mozilla.org. Creado el 2014/12/19. Accedido el 2015/01/08. https://wiki.mozilla.org/Main_Page.

8. Mozilla Foundation, "B2G/Roadmap" en wiki.mozilla.org. Creado el 2014/12/19. Accedido el 2015/01/08. https://wiki.mozilla.org/B2G/Roadmap.

9. Mozilla Foundation, "FxOS Product Backlog" en documento compartido en docs.google.com. Creado el –. Accedido el 2015/01/08. https://docs.google.com/spreadsheets/d/1r_zkWTbold5dlA5vh8P4xugTmRlfYQo5mUAuYc2qAxQ/edit#gid=1614323545.

10. Mozilla Foundation, "Bug 1092966 - FM Radio app to follow text selection pattern" en bugzilla.mozilla.org. Creado el –. Accedido el 2015/01/08. https://bugzilla.mozilla.org/show_bug.cgi?id=1092966.

11. Mozilla Foundation, "2.2 Feature list" en mozilla.aha.io. Creado el –. Accedido el 2015/01/08. https://mozilla.aha.io/published/2a593f5468fe2b361b7e9f1292d69ff1?page=2.

12. Amy Lee, "FxOS 2.2 - FM Radio.Visual Design Specification. Version 1.0" en documento compartido en google.drive.com. Creado el 2014/11/06. Accedido el 2015/01/08. https://drive.google.com/file/d/0B1Os8CF6IOKRZ3NSTlRQdDhwUUU/view?

usp=sharing..

13. Javier Merchán, "iOS y Android: rentabilidad contra cuota de mercado, en un gráfico" en celularis.com. Creado el 2014/12/02. Accedido el 2015/01/26. http://www.celularis.com/apple/ios-y-android-rentabilidad-contra-cuota-mercado-grafico/.

14. J. Pomeyrol, "¿Es necesario Firefox OS?" en muylinux.com. Creado el 2014/04/13. Accedido el 2014/12/15. http://www.muylinux.com/2014/04/13/necesario-firefox-os.

15. Christian de la Cruz, "Firefox OS poco a poco se posiciona en mercados emergentes" en wayerless.com. Creado el 2014/10/10. Accedido el 2015/01/08. http://www.wayerless.com/2014/10/firefox-os-mercados-emergentes/.

16. Google Inc, "Chromecast" en google.es. Creado el –. Accedido el 2015/01/08. https://www.google.es/chrome/devices/chromecast/.

17. MatchStick, "Matchstick" en matchstick.tv. Creado el –. Accedido el 2015/01/08. http://www.matchstick.tv/.

18. MatchStick, "Matchstick - The Streaming Stick Built on Firefox OS" en Kickstarter. Creado el 2014/09/30. Accedido el 2014/12/02. https://www.kickstarter.com/projects/matchstick/matchstick-the-streaming-stick-built-on-firefox-os.

19. Roberto Casas, "[CES 2015] Mozilla planea llevar Firefox OS a más dispositivos" en mundofirefoxos.com. Creado el 2015/01/08. Accedido el 2015/01/08. http://www.mundofirefoxos.com/2015/01/ces-2015-mozilla-planea-llevar-firefox_7.html.

20. Demóstenes García, "Los 10 Plugins De Sublime Text 2 Favoritos En Pixmat Studios" en PixmatStudios. Creado el 2013/01/21. Accedido el 2014/11/28. http://www.pixmatstudios.com/blog/10-plugins-sublime-text-pixmat/#.VG3ZfS73g_g.

21. The Document Foundation, "LibreOffice" en libreoffice.org. Creado el –. Accedido el 2014/09/14. https://es.libreoffice.org/.

22. Compañer@ de carrera. , "Información obtenida de trabajos realizados a lo largo de mi formación académica" en –. Creado el –. Accedido el –. –.

23. Inkscape, "Inkscape - Draw freely" en inkscape.org. Creado el –. Accedido el 2014/10/18. https://inkscape.org/es/.

24. The GIMP Team, "GIMP" en gimp.org. Creado el –. Accedido el 2014/10/18. http://www.gimp.org/.

25. CollabNet, "Welcome to ArgoUML" en tigris.org. Creado el –. Accedido el 2015/01/11. http://argouml.tigris.org/.

26. GanttProject Team, "Gantt Project" en ganttproject.biz. Creado el –. Accedido el 2015/01/19. http://www.ganttproject.biz/ostrava.

27. Miguel Useche, "Firefox OS llega a los televisores" en MozillaHispano. Creado el 2015/01/09. Accedido el 2015/01/09. http://www.mozilla-hispano.org/firefox-os-llega-a-los-televisores/.

28. jorgev, "Crear una aplicación de pago para Firefox OS" en mozilla-hispano.org. Creado el 2013/05/01. Accedido el 2015/01/26. http://www.mozilla-hispano.org/creando-una-aplicacion-de-pago-para-firefox-os/.

29. Mozilla Developer Network and individual contributors, "Enviar una aplicación al Marketplace de Firefox OS" en developer.mozilla.org. Creado el –. Accedido el 2015/01/27. https://developer.mozilla.org/es/Marketplace/Submission/Submitting_an_app.

30. Mozilla Developer Network and individual contributors, "App pricing" en developer.mozilla.org. Creado el –. Accedido el 2015/01/27. https://developer.mozilla.org/en-US/Marketplace/Monetization/App_pricing.

31. teleyradio, "La Historia de la TV y la Radio" en blogspot.es. Creado el 2008/06/15. Accedido el 2015/02/28. http://teleyradio.blogspot.es/.

32. Carlos García, "Historia de la radio" en RadioEspaña.net. Creado el –. Accedido el 2015/01/07. http://www.radioes.net/historia.asp.

33. Ministerio de Educación, Gobierno de España, "Media Radio, Historia y

Evolución" en mec.es. Creado el –. Accedido el 2014/12/16. http://recursos.cnice.mec.es/media/radio/bloque1/index.html.

34. Juan Ranchal, "¿Es Android realmente libre? Richard Stallman dice no" en muycomputer.com. Creado el 2011/09/20. Accedido el 2015/01/09. http://www.muycomputer.com/2011/09/20/android-realmente-libre-richard-stallman-dice-no.

35. Mozilla Foundation, "Web API Interfaces" en MDN. Creado el 2014/12/22. Accedido el 2014/09/01. https://developer.mozilla.org/en-US/docs/Web/API.

36. Alberto Saiz, "Aplicaciones móviles: Nativas vs Web Apps" en Surática Software. Creado el 2014/06/06. Accedido el 2015/01/26. http://www.suratica.es/aplicaciones-moviles-nativas-vs-web-apps/.

37. arsys, "¿MyISAM o InnoDB? Elige tu motor de almacenamiento MySQL" en arsys.info. Creado el 2014/04/12. Accedido el 2015/04/27. http://www.arsys.info/programacion/myisam-o-innodb-elige-tu-motor-de-almacenamiento-mysql/.

38. Mozilla Developer Network, Mozilla Foundation, "XMLHttpRequest" en MDN. Creado el 2015/04/25. Accedido el 2014/11/07. https://developer.mozilla.org/es/docs/XMLHttpRequest.

39. Colaboradores OpenStreetMap, "Nominatim usage policy" en wiki.openstreetmap.org. Creado el 2015/02/04. Accedido el 2015/05/10. http://wiki.openstreetmap.org/wiki/Nominatim_usage_policy.

40. Colaboradores OpenStreetMap, "ES:Open Database License" en wiki.openstreetmap.org. Creado el 2013/04/09. Accedido el 2015/05/10. http://wiki.openstreetmap.org/wiki/ES:Open_Database_License.

41. Mozilla Developer Network, Mozilla Foundation, "App manifest" en MDN. Creado el 2015-02-17. Accedido el 2015-05-10. https://developer.mozilla.org/es/Apps/Build/Manifest.

42. Mozilla Developer Network, Mozilla Foundation, "App manifest" en MDN. Creado el 2015-05-11. Accedido el 2015-05-13. https://developer.mozilla.org/en-

US/Apps/Build/Manifest.

43. Mozilla Developer Network, Mozilla Foundation, "Web developer guide" en MDN. Creado el –. Accedido el 2014/12/16. https://developer.mozilla.org/en-US/docs/Web/Guide.

44. Paulo Andrade, "Habilitar cors en php" en Programación Azteca. Comunidad de programación web. Creado el 2015/02/19. Accedido el 2014/07/13. http://azteca.programandowebs.com/php/habilitar-cors-en-php/596/.

45. Xabadu, "Como lo hago" en Cómo habilitar CORS en Apache y/o PHP. Creado el 2015/02/19. Accedido el 2014/02/06. http://www.comolohago.cl/como-habilitar-cors-en-apache-yo-php/.

46. Mozilla Developer Network and individual contributors, "L10n" en developer.mozilla.org. Creado el –. Accedido el 2015/02/08. https://developer.mozilla.org/en-US/docs/Web/API/L10n_API.

47. Mozilla Developer Network and individual contributors, "Localization" en developer.mozilla.org/. Creado el –. Accedido el 2015/02/08. https://developer.mozilla.org/en-US/Add-ons/SDK/Tutorials/l10n#Placeholders.

48. Mozilla Developer Network, Mozilla Foundation, "App Manifest" en MDN. Creado el 2015/02/17. Accedido el 2015/05/01. https://developer.mozilla.org/en-US/Apps/Build/Manifest#type.

49. Mozilla Developer Network, Mozilla Foundation, "CSP (Content Security Policy)" en MDN. Creado el 2015/01/14. Accedido el 2015/05/01. https://developer.mozilla.org/en-US/docs/Web/Security/CSP.

50. Mozilla Developer Network, Mozilla Foundation, "Apps CSP para FxOS" en MDN. Creado el 2015/02/10. Accedido el 2015/05/01. https://developer.mozilla.org/en-US/Apps/Build/Building_apps_for_Firefox_OS/CSP.

ANEXO I
CASOS DE USO EXTENDIDOS

Anexo I - Casos de uso extendidos

En el anexo I se van a detallar los casos de uso anteriormente nombrados, explicando de forma más detallada las diferentes funcionalidades que se quieren implementar. Todas las capturas de pantalla están realizadas estando configurado el español como idioma del dispositivo. Todas estas imágenes están obtenidas con el terminal Geeksphone Developer Previuw prestado por el departamento de lenguajes y sistemas informáticos, haciendo capturas de pantalla pulsando simultáneamente la tecla de encendido y volumen menos.

A continuación, se representa la interfaz inicial (Ilustración 28) de la aplicación, cuando se inicia por primera vez la app en el terminal, la pantalla de identificación o registro cuando el usuario accede de modo invitado.

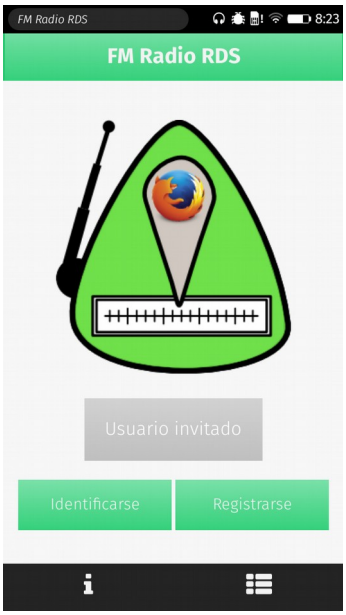


Ilustración 28: UI_menu_inicial

1 Registrarse

Nombre: Registrarse.

Descripción: Permite al usuario invitado que usa la aplicación añadirse al sistema. De este modo, poder identificarse para utilizar todas las funcionalidades que tienen disponibles los usuarios identificados.

Actores: Usuario invitado.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet activados.

Flujo de eventos:

1 El usuario rellenará los campos "Nombre usuario", "Email", "Password" y "Repetir password" y pulsará en "Realizar registro" (Ilustración 29).

1.1 Si pulsa "Realizar registro" habiendo dejado alguno de los campos sin rellenar, saltará un aviso de error (Ilustración 30).

1.2 Rellena todos los campos y pulsa el botón "Realizar registro":

1.2.1 Si el nombre de usuario ya existe y se está usando, saltará un aviso de error (Ilustración 33).

1.2.2 Si el nombre de usuario es inferior a 6 caracteres o mayor de 20, saltará un aviso de error (Ilustración 31,Ilustración 32).

1.2.3 Si el email ya existe y se está usando, saltará un aviso de error (Ilustración 34).

1.2.4 Si el email no tiene formato de correo electrónico, saltará un aviso de error (Ilustración 30).

1.2.5 Si las contraseña y/o repetir contraseña son inferior a 6 caracteres, saltará un aviso de error (Ilustración 35).

1.2.6 Si la contraseña y repetir contraseña no son iguales, saltará un aviso de error (Ilustración 36).

1.2.7 Si no hay errores, el registro se realiza correctamente y salta un mensaje de aviso correcto informando (Ilustración 37).

Postcondiciones: El usuario se habrá registrado en el sistema. Ahora podrá identificarse. La aplicación redirige automáticamente a esta interfaz.

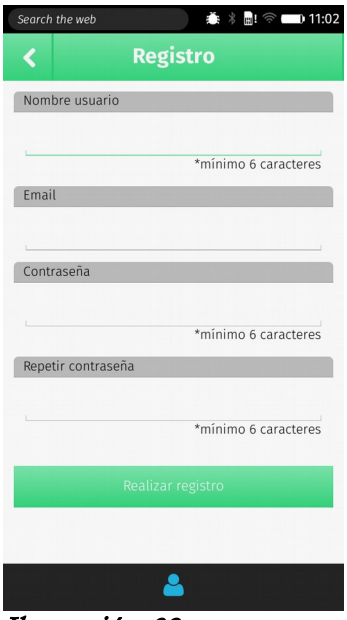


Ilustración 29: UI_registrarse

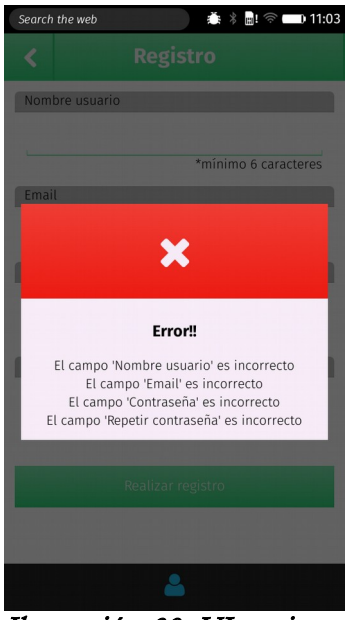


Ilustración 30: UI_registrarse_mensaje_error

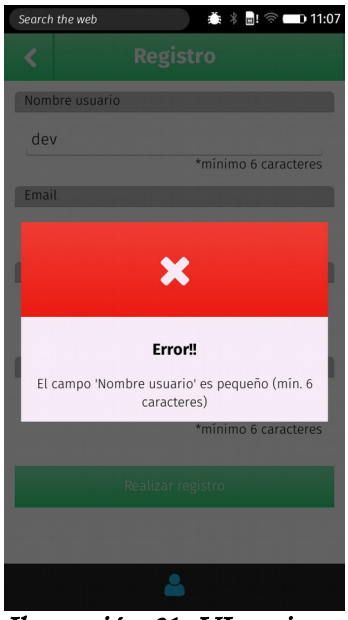


Ilustración 31: UI_registrarse_mensaje_error

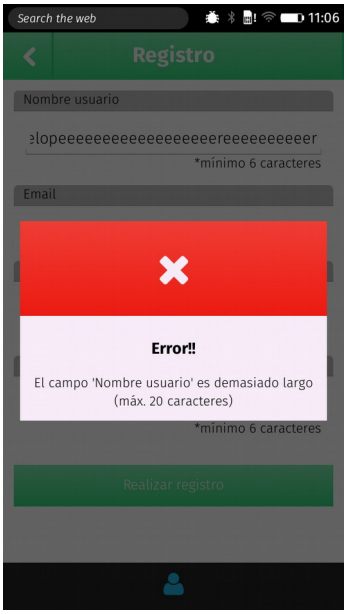


Ilustración 32: UI_registrarse_mensaje_error

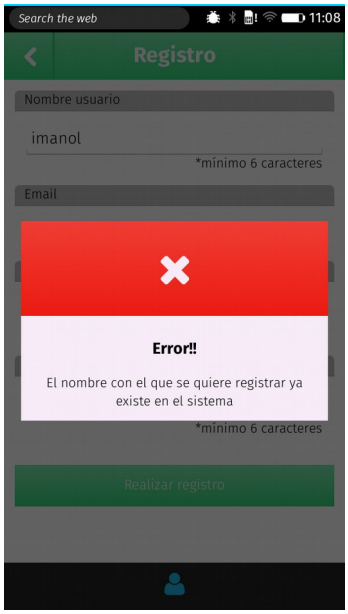


Ilustración 33: UI_registrarse_mensaje_error

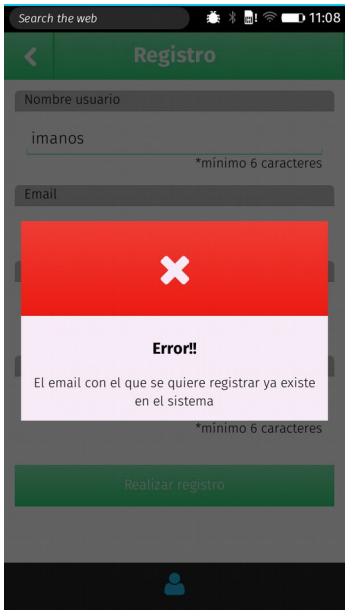


Ilustración 34: UI_registrarse_mensaje_error

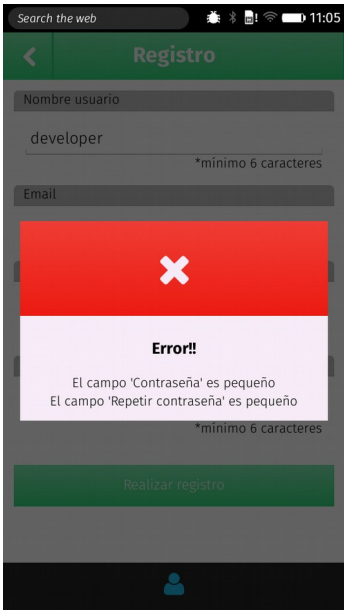


Ilustración 35: UI_registrarse_mensaje_error

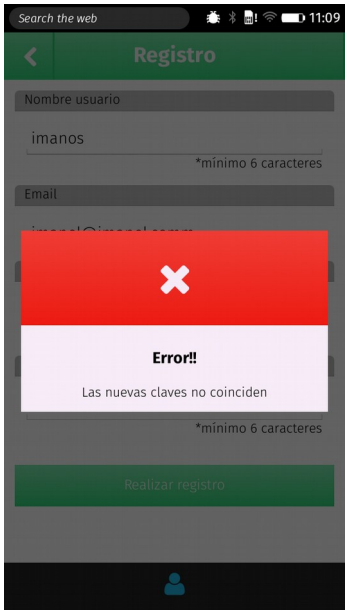


Ilustración 36: UI_registrarse_mensaje_error

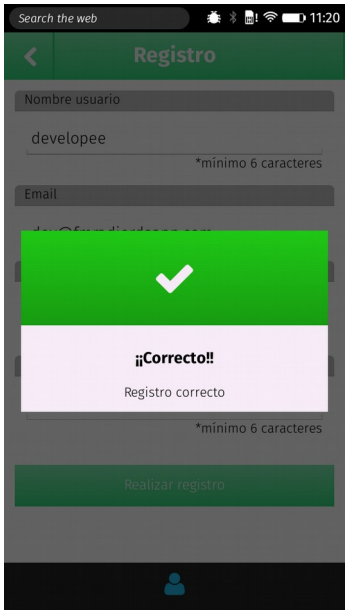


Ilustración 37: UI_registrarse_mensaje_ok

2 Identificarse

Nombre: Identificarse.

Descripción: Permite al usuario invitado que usa la aplicación identificarse en el sistema. De este modo, poder utilizar todas las funcionalidades que tienen disponibles los usuarios identificados.

Actores: Usuario invitado.

Precondiciones: Ninguno.

Requisitos funcionales: Conexión a Internet activados.

Flujo de eventos:

1 El usuario rellenará los campos "Email" y "Contraseña" y pulsará en "Entrar" (Ilustración 38).

1.1 Si pulsa "Entrar" habiendo dejado alguno de los campos sin rellenar, saltará un aviso de error (Ilustración 39).

1.2 Si rellena todos los campos y pulsa el botón "Entrar":

1.2.1 Si el email no tiene formato de correo electrónico, saltará un aviso de error (Ilustración 39).

1.2.2 Si las contraseña es inferior a 6 caracteres, saltará un aviso de error (Ilustración 39).

1.2.3 Si no falla estas dos:

1.2.3.1 Si el email no existe, saltará un aviso de error (Ilustración 40).

1.2.3.2 Si el email existe pero la contraseña no coincide, saltará un aviso de error (Ilustración 41).

1.2.4 Si no ocurre nada de lo anterior, la identificación se realiza correctamente.

1.2.4.1 Si el usuario estaba activo, salta un mensaje de aviso correcto informando que todo es correcto (Ilustración 42).

1.2.4.2 Si el usuario no está activo, salta un mensaje de aviso explicando que el usuario estaba dado de baja y ahora se ha activado. Es necesario una nueva identificación para acceder al sistema. (Ilustración 43).

Postcondiciones: El usuario se habrá identificado en el sistema. Podrá empezar a utilizar FM Radio RDS desde el modo usuario identificado. La aplicación le redirige automáticamente a la interfaz principal.

Ilustración 38: UI_identificarse

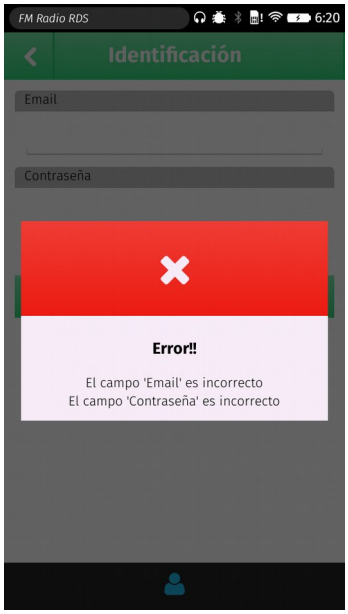


Ilustración 39: UI_identificarse_mensaje_error

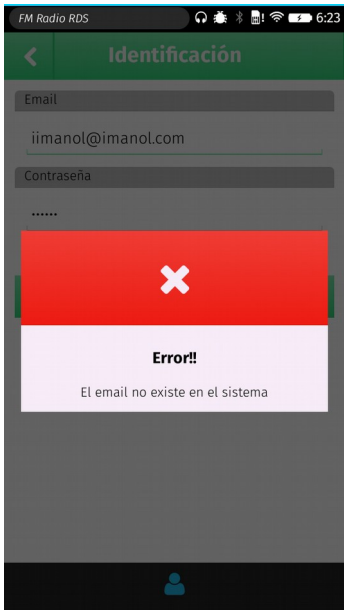


Ilustración 40: UI_identificarse_mensaje_error

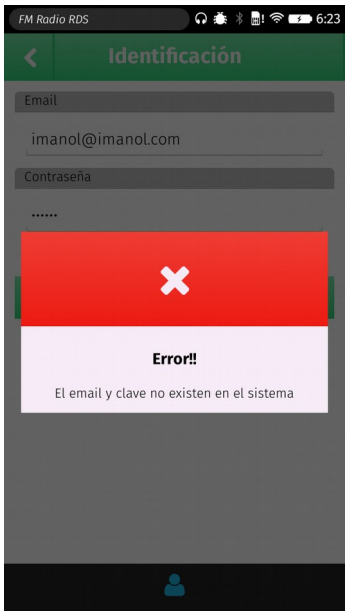


Ilustración 41: UI_identificarse_mensaje_error

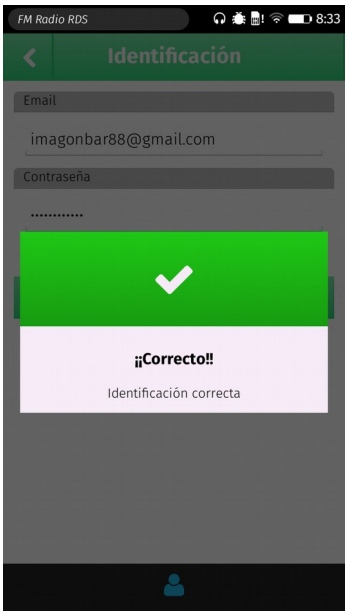


Ilustración 42: UI_identificarse_mensaje_ok

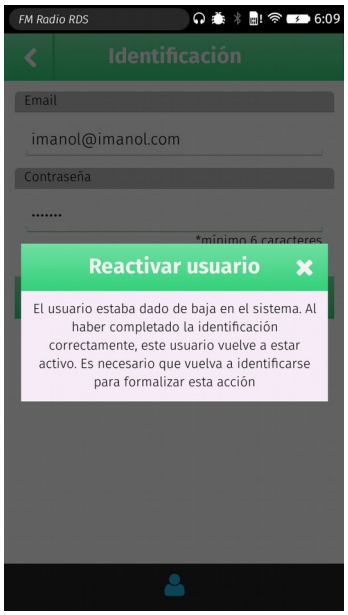


Ilustración 43: UI_identificarse_inactivo

3 Ver información y datos de contacto del desarrollador y de la app

Nombre: Ver información y datos de contacto del desarrollador y de la app.

Descripción: Permite ver información sobre el origen de la aplicación, datos de contacto de la misma y del desarrollador, así como enlaces a sus redes sociales (Twitter y Google Plus) y cuentas de correo.

Actores: Usuario invitado.

Precondiciones: Ninguno.

Requisitos funcionales: Conexión a Internet activados.

Flujo de eventos:

1 Se accederá pulsando en el botón de la esquina superior derecha del menú inicial. Podrá leer la información allí escrita (Ilustración 44).

2 En el apartado de "Datos de contacto", aparecen enlaces a los perfiles sociales y la cuenta de correo tanto del desarrollador como de la aplicación (Ilustración 45). Si se pulsa cualquiera de ellos, se accede a estos perfiles, a través de sus versiones web y apps (Ilustración 46).

Postcondiciones: El usuario se habrá informado y se podrá poner en contacto con FM Radio RDS y su desarrollador. Al volver, la aplicación le redirige automáticamente a la interfaz principal.

Ilustración 44: UI_info_app

Ilustración 45: UI_info_app

Ilustración 46: UI_info_app_rrss

4 Ver comentarios de usuarios cercanos

Nombre: Ver comentarios de usuarios cercanos.

Descripción: Permite ver los comentarios que los usuarios han podido escribir en sus perfiles sin necesidad de estar identificado.

Actores: Usuario invitado.

Precondiciones: Ninguno.

Requisitos funcionales: Conexión a Internet y geoposición activados.

Flujo de eventos:

1 Se accederá pulsando en el botón "Ver comentarios" del menú inicial. Aparecerá un mensaje de carga (Ilustración 47).

2 Se pueden ver los comentarios cercanos al usuario, según su posición (Ilustración 48). Si se hace scroll hacia abajo, se recarga el listado de comentarios (Ilustración 49).

Postcondiciones: El usuario habrá leído los comentarios que hay cercanos a su posición. Al volver, la aplicación le redirige automáticamente a la interfaz principal.

Ilustración 47: UI_show_coments_loading

Ilustración 48: UI_show_coments

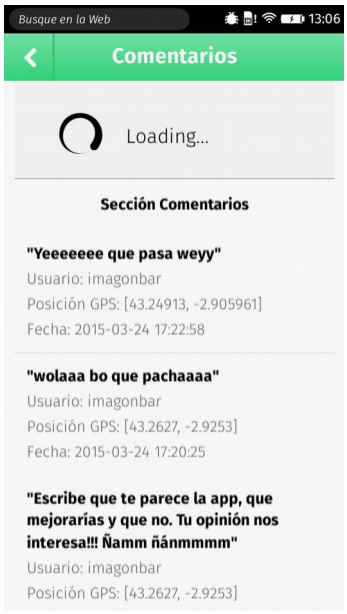


Ilustración 49: UI_show_coments_scroll

5 Ver emisoras con emisiones cercanas

Nombre: Ver emisoras con emisiones cercanas.

Descripción: Permite ver el listado de emisoras que tienen emisiones cercanas a su posición. Si el usuario está identificado, será la ventana inicial cuando inicia la app.

Actores: Usuario.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 El usuario accede a esta sección de la aplicación donde se ve un listado de emisoras ordenadas por nombre, cada una con una emisión asignada. Cada emisión contiene una frecuencia concreta que se supone se escucha en esa posición donde se encuentra el usuario.

1.1 Si el usuario no esta identificado en el sistema, accederá pulsando en el botón "Entrar como invitado" del menú inicial (Ilustración 28) y verá esta interfaz (Ilustración 50).

1.1.1 Si es la primera vez que accede a esta interfaz, verá un mensaje con una serie de consejos de uso (Ilustración 51).

1.2 Si el usuario está identificado en el sistema, accederá directamente a esta sección, que es la parte principal de la app (Ilustración 52).

1.2.1 Si es la primera vez que accede a esta interfaz, verá un mensaje con una serie de consejos de uso (Ilustración 53).

Postcondiciones: El usuario habrá accedido a la ventana principal del sistema, donde ver el listado de emisoras que hay cercanas, según la posición donde se encuentre.

Ilustración 50: UI_notlogged_ppal

Ilustración 51: UI_not logged_ppal_consejo

Ilustración 52: UI_ logged_ppal

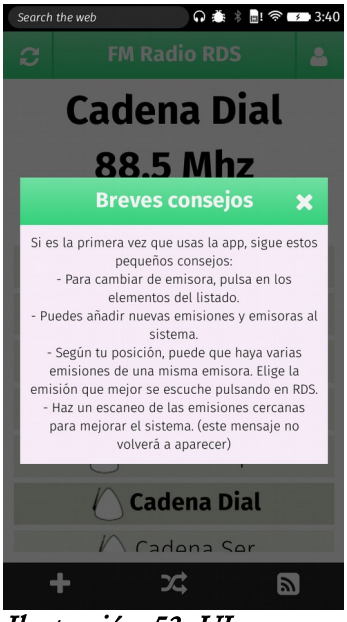


Ilustración 53: UI_ logged_ppal_consejo

6 Elegir emisora al pulsar lista manualmente

Nombre: Elegir emisora al pulsar lista manualmente.

Descripción: Permite seleccionar cualquier elemento del listado de emisoras. La frecuencia que esté sonando se actualizará, emitiendo lo que reporte en la nueva emisora. No es del todo fiable que en esa nueva frecuencia seleccionada, la emisora que esté sintonizada sea la misma que se estaba escuchando. Depende de la calidad y veracidad de los datos que los usuarios aporten al sistema. Además, en cada posición la recepción de la señal dependerá de factores meteorológicos, como ocurre con cualquier receptor de radio FM analógica.

Actores: Usuario.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 Si el usuario pulsa sobre un elemento del listado, este elemento quedará seleccionado. La frecuencia que se está reproduciendo se actualiza por la nueva que asigna el elemento pulsado y se actualiza también el nombre de la emisora (Ilustración 50 y/o Ilustración 52).

Postcondiciones: El usuario, estando en la ventana principal de la aplicación, ha podido elegir la emisora que se quiere escuchar.

7 RDS

Nombre: RDS.

Descripción: Permite ver el listado de emisiones asociadas a la emisora que se está escuchando. Al pulsar el botón RDS, aparecerá en una nueva ventana un listado con las emisiones cercanas a la posición donde esté el usuario de la emisora que se está escuchando.

Actores: Usuario.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 El usuario se encuentra en la ventana principal de la aplicación. Hay que pulsar en el botón RDS, que se encuentra en la barra inferior de la interfaz.

1.1 Si el usuario no está identificado en el sistema, en dicha barra solamente tendrá un botón al que pulsar, con el icono de dos flechas cruzándose (Ilustración 50).

1.2 Si el usuario está identificado en el sistema, en dicha barra habrá tres botones a los que pulsar. Debe seleccionar el del medio, el icono de dos flechas cruzándose (Ilustración 52).

2 Se accederá a una nueva sección de la aplicación. Cuando finalizan de cargar los elementos a listar, desaparece el aviso de carga. En esta ventana aparece un listado con las emisiones cercanas almacenadas en el sistema de la emisora que se tiene seleccionada y se está reproduciendo. Se muestra el nombre de la emisora, la frecuencia de cada emisión cercana y en donde se ha registrado esa emisión.

2.1 Si el usuario pulsa una de ellas, directamente se selecciona y se actualiza la frecuencia que se estuviera sintonizando (Ilustración 54).

2.2 Si el usuario es un usuario identificado, podrá variar la información que le aparece en esa interfaz. Podrá elegir ver las emisiones más cercanas (precisión normal) o las algo más alejadas a su posición (poca precisión) (Ilustración 56).

2.3 Si no es un usuario identificado, le aparecerán las emisiones más cercanas, las que en función de los datos recogidos entrar dentro de la precisión normal.

3 Si pulsa en el botón de información de la esquina superior derecha, le aparecerá un mensaje con un mensaje explicativo e informativo (Ilustración 55).

4 Para volver a la ventana principal, deberá pulsar el botón de la esquina superior izquierda, el icono del "*Play*".

Postcondiciones: El usuario habrá accedido a la sección RDS donde poder cambiar manualmente la emisión que se está escuchando sin cambiar de emisora.

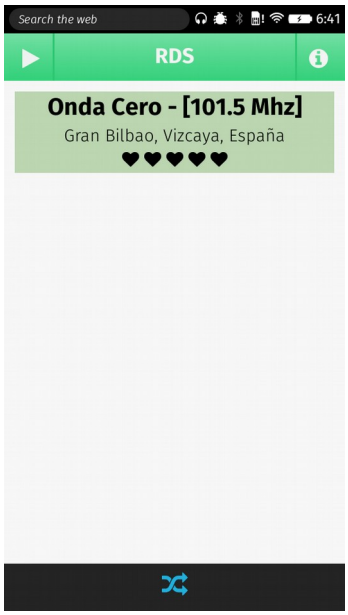


Ilustración 54: UI_notlogged_RDS

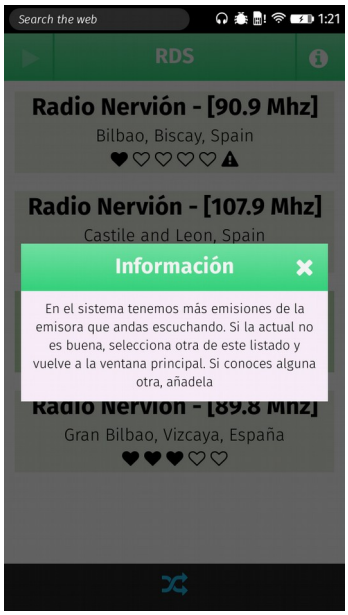


Ilustración 55: UI_notlogged_RDS_info

Ilustración 56: UI_logged_RDS

8 Elegir emisión al pulsar lista manualmente

Nombre: Elegir emisión al pulsar lista manualmente.

Descripción: Permite elegir un nuevo elemento que sintonizar con sólo pulsar un ítem del listado de emisiones que verá en pantalla. .

Actores: Usuario.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 El usuario se encuentra en la sección RDS. Aparece un listado con las emisiones cercanas almacenadas en el sistema de la emisora que se tiene seleccionada y se está reproduciendo.

1.1 Si el usuario pulsa una de ellas, directamente se selecciona y se actualiza la frecuencia que se estuviera sintonizando (Ilustración 54 y Ilustración 56).

Postcondiciones: El usuario ha seleccionado algún elemento del listado mostrado. Se habrá cambiado manualmente la emisión que se está escuchando sin cambiar de emisora.

9 Play / Stop

Nombre: Play / Stop.

Descripción: Permite iniciar o parar la reproducción de una emisión del listado principal.

Actores: Usuario

Precondiciones: Tener un elemento del listado principal seleccionado, para poder reproducir alguna emisión.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 El usuario se encuentra en la interfaz principal, tanto si está identificado como si no lo está. Teniendo pulsada una emisión del listado principal, puede pulsar tanto el botón de Play como el de Stop.

1.1 Si pulsa el botón de Play/Reproducir, si no se está reproduciendo la emisión, empezará a sonar a través de los auriculares una vez pasados unos 2-3 segundos. Si ya se estaba reproduciendo una emisión, no ocurrirá nada. Al pulsar en este botón, cambiará su color de azul a naranja e invertirá el color del botón Pause/Parar. Si ya estuviera naranja, quiere decir que ya estaba activo (Ilustración 57).

1.2 Si pulsa el botón Pause/Parar, si se estaba reproduciendo alguna emisión por los auriculares, se parara y se hará el silencio. Si no se estaba reproduciendo nada, tampoco se reproducirá nada, seguirá el silencio. Al pulsar en este botón, cambiará su color de azul a naranja e invertirá el color del botón Reproducir/Play. Si ya estuviera naranja, quiere decir que ya estaba activo (Ilustración 58).

2 La primera vez que se arranca la aplicación, los botones Play/Pause no tienen la funcionalidad deseada hasta que no se pulsa un elemento del listado principal. Sino, saltan avisos (Ilustración 59) (Ilustración 60).

Postcondiciones: El usuario ha pulsado alguno de los dos botones cuya funcionalidad se ha explicado. Se activará la reproducción o se silenciará la radio FM, además de cambiar su color de azul a naranja y viceversa.

Ilustración 57: UI_notlogged_ppal_pause

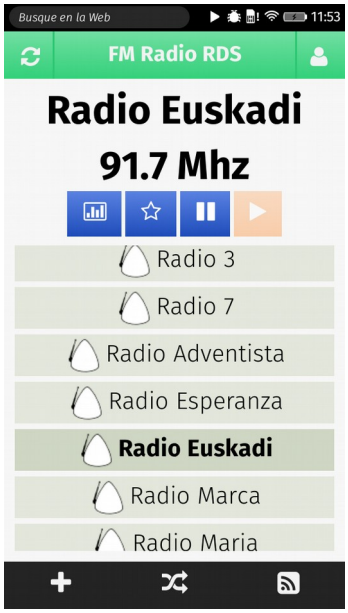


Ilustración 58: UI_logged_ppal_play

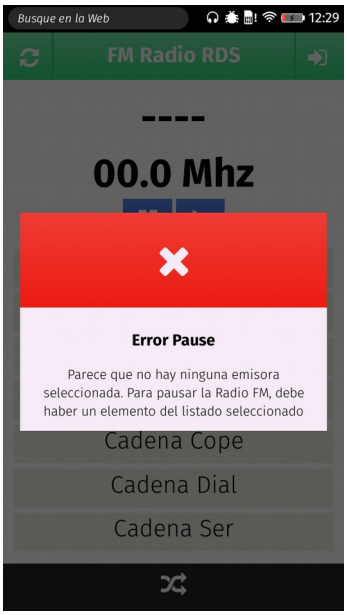


Ilustración 59: UI_notlogged_ppal_pause_error

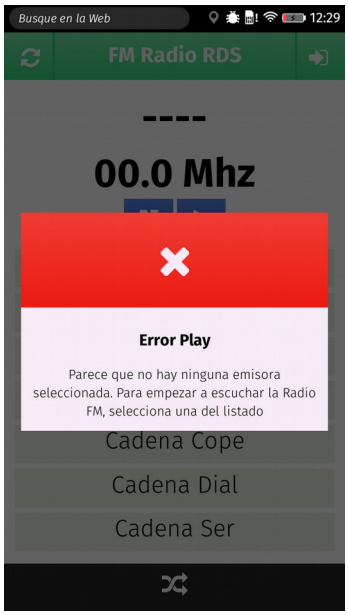


Ilustración 60: UI_notlogged_ppal_play_error

10 Valorar emisión

Nombre: Valorar emisión.

Descripción: Permite valorar la emisión que esté escuchando. Tendrá una escala con 5 posibles valores que indicar, desde "Muy Mala Señal" con el slider arrastrado a la parte izquierda, "Mala Señal" con el slider colocado entre la parte izquierda y el centro, "Normal Señal" con el slider colocado en el centro, "Buena Señal" con el slider colocado entre la parte derecha y el centro y "Muy Buena Señal" con el slider arrastrado a la parte derecha. El usuario, según la calidad de la recepción en el momento de puntuar, subjetivamente elegirá una de las opciones de la escala. Esta valoración servirá para las recomendaciones posteriores. Se añadirá al listado de valoraciones realizadas por cada usuario. .

Actores: Usuario identificado.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 El usuario se encuentra en la interfaz principal, con una emisora seleccionada (Ilustración 50 y Ilustración 52).

2 Pulsa el botón azul de las estadísticas, al lado de la estrella. Le aparece un aviso, con un mensaje informativo y un slider. Arrastrando el slider hacia la izquierda o derecha, le aparecerá un texto donde le indica la puntuación que ha marcado (Ilustración 61 y Ilustración 62).

2.1 Si pulsa en la opción "Puntuar", se registrará en el sistema esa información aportada por el usuario.

2.2 Si pulsa la opción "Ahora no", volverá a la pantalla principal.

Postcondiciones: El usuario ha puntuado una emisión y se han guardado los datos aportados.

Ilustración 61: UI_logged_puntuacion

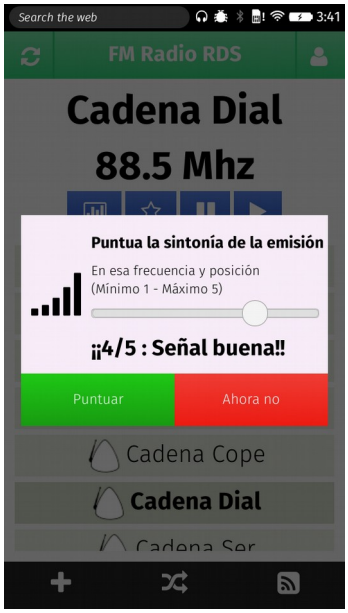


Ilustración 62: UI_logged _puntuacion_resultado

11 Añadir emisión manualmente

Nombre: Añadir emisión manualmente.

Descripción: Permite añadir manualmente una emisión. Si el usuario de la aplicación conoce una emisión que se sintonice en esa posición y que no está añadida, podrá incluirla pulsando el botón con el icono del más (+). Permitirá introducir la frecuencia que conoce para una emisora en la posición en la que se encuentre. El nombre de la emisora la seleccionará de un desplegable que le aparecerá en esa interfaz. Se añadirá al listado de emisiones añadidas de cada usuario.

Actores: Usuario identificado.

Precondiciones: Saber una emisión en esa posición que no esté añadida.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 El usuario se encuentra en la sección añadir emisión (Ilustración 63).

2 Selecciona una emisora del desplegable que le aparece en esa pantalla (Ilustración 64).

3 Introduce la frecuencia (Ilustración 63).

4 El usuario pulsa el botón "Añadir emisión".

4.1 Si la frecuencia no se ha rellenado o no tiene formato numérico, saldrá un aviso de error (Ilustración 65).

4.2 Si la frecuencia no se encuentra entre 87.0 y 108.0, saldrá un aviso de error (Ilustración 66).

4.3 Si el par frecuencia-emisora que se va a añadir ya se encuentra registrada en el sistema en una posición cercana a la que está el usuario en el momento de añadirla, saldrá un aviso de error (Ilustración 67).

4.4 Si no ha salido ningún mensaje de los indicados previamente, la emisión se ha introducido correctamente (Ilustración 68). En ese caso, la aplicación le redirige a la ventana principal.

5 Si pulsa el botón "Añadir emisora" accederá a la sección donde añadir una emisora que no exista en el sistema.

6 Si pulsa el botón de información situado en la esquina superior derecha, le aparecerá un mensaje explicativo de la sección (Ilustración 69).

7 Si pulsa en el icono de "Play" de la esquina superior izquierda, volverá a la ventana principal.

Postcondiciones: El usuario ha añadido una emisión al sistema correctamente.

Ilustración 63: UI_logged _aniadir_emision

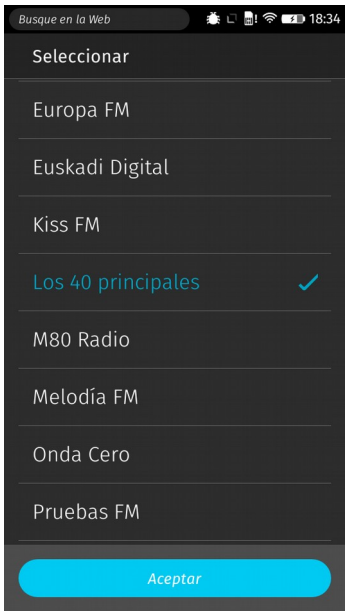


Ilustración 64: UI_logged _aniadir_emision_listado

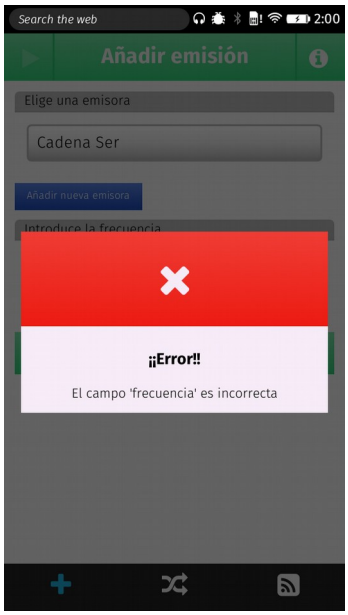


Ilustración 65: UI_logged _aniadir_emision_error

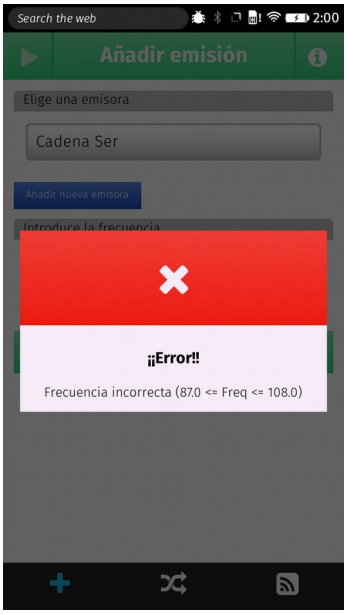


Ilustración 66: UI_logged _aniadir_emision_error

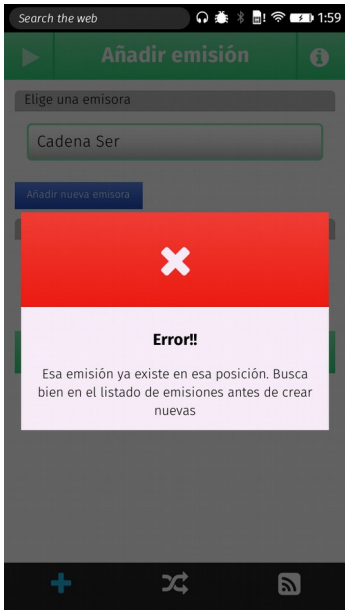


Ilustración 67: UI_logged _aniadir_emision_error

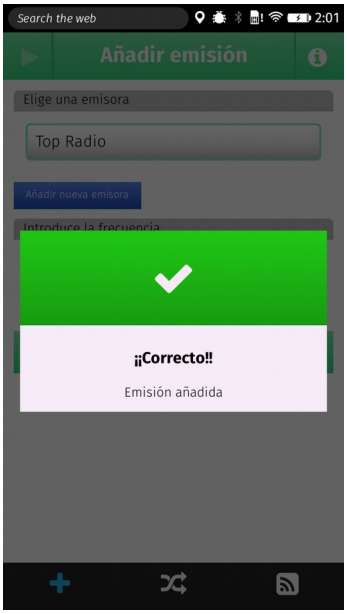


Ilustración 68: UI_logged _aniadir_emision_ok

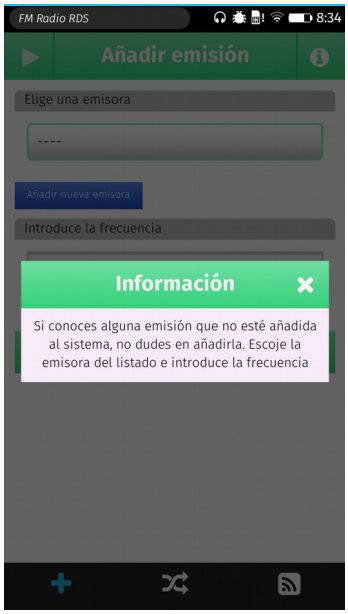


Ilustración 69: UI_logged _aniadir_emision_info

12 Añadir emisora manualmente

Nombre: Añadir emisora manualmente.

Descripción: Permite añadir una nueva emisora que no exista en la base de datos del sistema. Se ruega que el nombre introducido se ajuste lo máximo posible al real. Se seleccionará en ese instante el alcance de la emisora y el tipo o tipos de emisora que son.

Actores: Usuario identificado.

Precondiciones: Conocer una emisora que no esté añadida y preferiblemente con una emisión asociada que añadir en esa posición.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 El usuario se encuentra en la sección añadir emisora (Ilustración 70).

2 Introduce el nombre de la nueva emisora (Ilustración 71).

3 Selecciona un alcance del desplegable que le aparece en esa pantalla (Ilustración 72).

4 Selecciona uno o varios tipos de emisora del desplegable (Ilustración 73).

5 El usuario pulsa el botón "Añadir emisora".

5.1 Si el nombre de emisora tiene menos de 4 caracteres o está vació, saldrá un aviso de error (Ilustración 74).

5.2 Si el nombre de emisora ya se encuentra en el sistema, saldrá un aviso de error (Ilustración 75).

5.3 Si el usuario no selecciona ningún elemento del desplegable de tipos de emisoras, saldrá un aviso de error (Ilustración 76).

5.4 Si no ha salido ningún mensaje de los indicados previamente, la emisora se ha introducido correctamente (Ilustración 77). En ese caso, la aplicación le redirige a la pantalla de "Añadir emisión".

6 Si pulsa el botón de información situado en la esquina superior derecha, le aparecerá un mensaje explicativo de la sección (Ilustración 78).

7 Si pulsa en el icono de atrás de la esquina superior izquierda, volverá a la ventana de "Añadir emisión".

Postcondiciones: Se añade una emisora al sistema correctamente.

Ilustración 70: UI_logged _aniadir_emisora

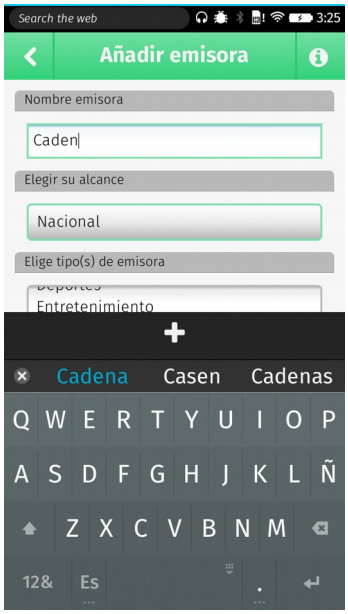


Ilustración 71: UI_logged _aniadir_emisora_teclado

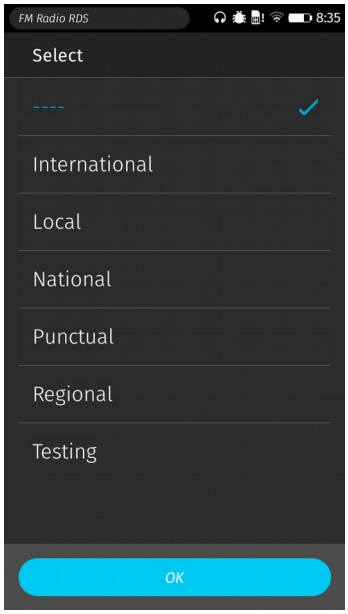


Ilustración 72: UI_logged _aniadir_emisora_listado_ alcance

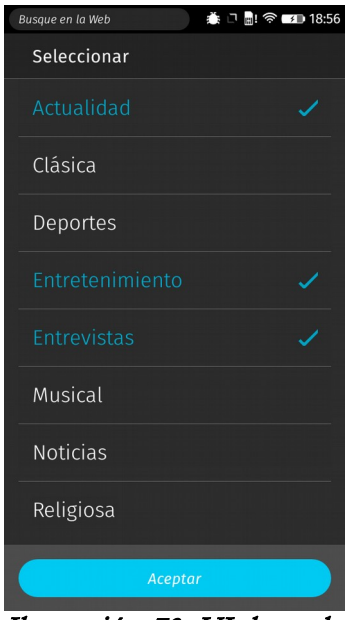


Ilustración 73: UI_logged _aniadir_emisora_listado_ tipos

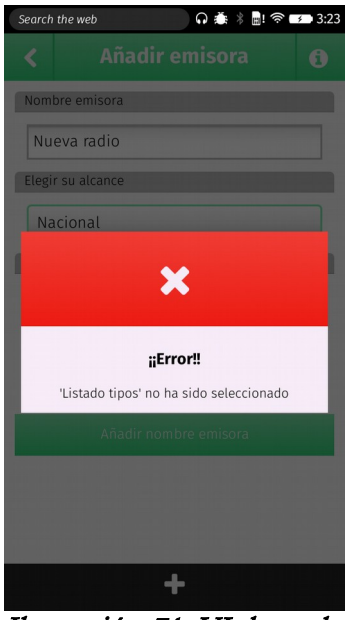


Ilustración 74: UI_logged _aniadir_emisora_mensaje _error

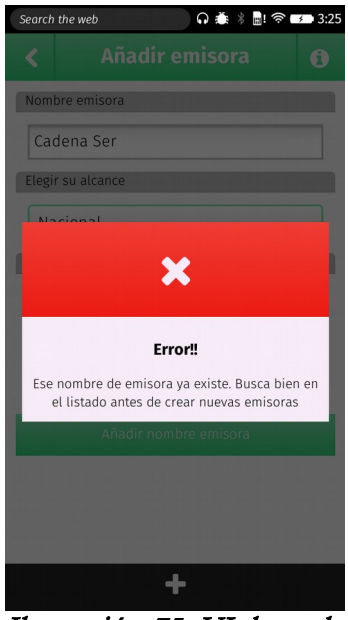


Ilustración 75: UI_logged _aniadir_emisora_mensaje _error

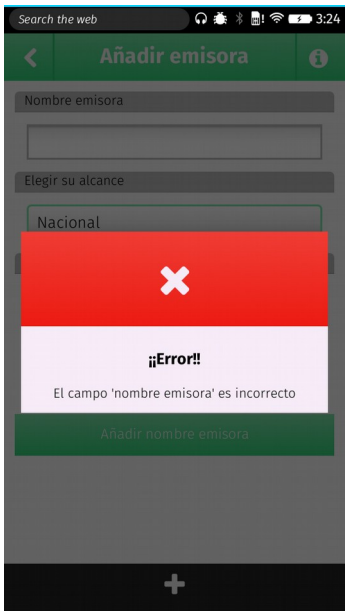


Ilustración 76: UI_logged _aniadir_emisora_mensaje _error

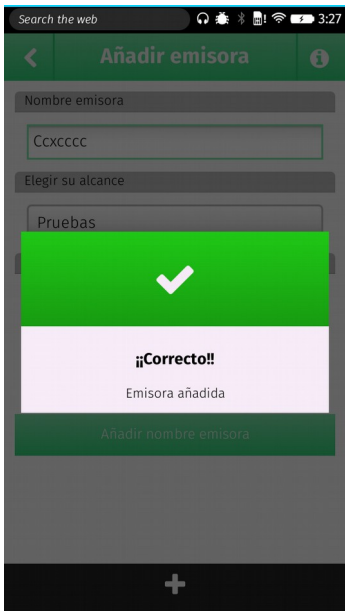


Ilustración 77: UI_logged _aniadir_emisora_mensaje _ok

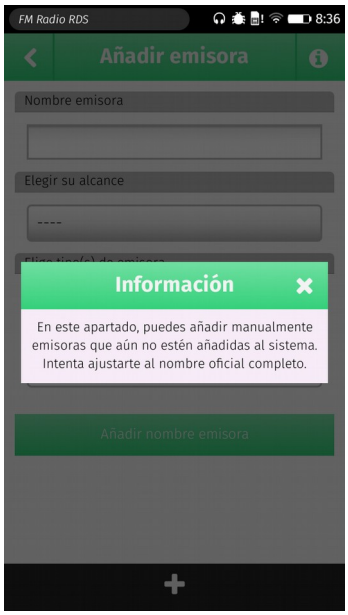


Ilustración 78: UI_logged _aniadir_emisora_mensaje _info

13 Poner favorita emisora

Nombre: Poner favorita emisora.

Descripción: Permite añadir como emisora favorita la emisora que se esté escuchando. Será una estrella que tendrá dos estados, seleccionada o no seleccionada. Se añadirá al listado de emisoras favoritas de cada usuario.

Actores: Usuario identificado.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 El usuario se encuentra en la interfaz principal, con una emisora seleccionada (Ilustración 79). Hay que fijarse en el botón azul de los favoritos, el de la estrella.

2 Cuando carga la ventana principal:

2.1 Si el usuario tiene la emisora seleccionada en su listado de favoritas, la estrella aparecerá seleccionada.

2.2 Si el usuario no tiene la emisora seleccionada en su listado de favoritas, la estrella aparecerá vacía.

3 Entonces, cuando el usuario pulsa en el botón de la estrella:

3.1 Si la estrella está seleccionada, se deseleccionará y se quitará la emisora del listado de favoritas de ese usuarios.

3.2 Si la estrella no está seleccionada, se quedará seleccionada y se añadirá la emisora al listado de favoritas de ese usuarios.

Postcondiciones: El usuario ha añadido o quitado la emisora de su listado de favoritos.

Ilustración 79: UI_logged _favorita

14 Ver información completa de la emisora

Nombre: Ver información completa de emisora.

Descripción: Permite ver la información de una emisora seleccionada.

Actores: Usuario identificado.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 El usuario se encuentra en la interfaz principal, con una emisora seleccionada (Ilustración 79).

2 Al pulsar en el icono de al lado de cada nombre de emisoras, se accede a la información completa de la emisora

3 Cuando carga esta ventana de información, se muestra el nombre de la emisora, su alcance y su tipo, la fecha de cuando se agregó y el listado de las emisiones que hay registras sobre esa emisora (Ilustración 80).

Postcondiciones: El usuario verá la información completa de la emisora.

Ilustración 80: UI_logged _ver_info_emisora

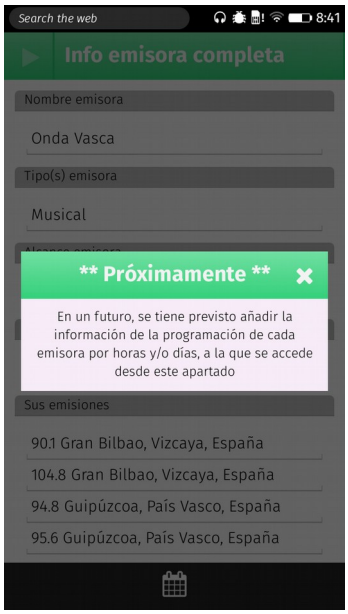


Ilustración 81: UI_logged _ver_info_emisora

15 Hacer escaneo RFOS

Nombre: Hacer escaneo RFOS.

Descripción: Permite hacer un escaneo completo de las frecuencias encontradas en esa posición. Recorrerá en bucle el dial, mostrando uno por uno los resultados obtenidos. Se podrá poner el nombre individualmente a las emisoras encontradas. Se añadirá al listado de emisiones añadidas de cada usuario.

Actores: Usuario identificado.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 Una vez accedido a esta sección, para iniciar el escaneo se debe pulsar el botón Iniciar escaneo (Ilustración 82).

2 Durante unos segundos aparece un mensaje de aviso de carga, con el mensaje de Iniciando (Ilustración 83).

3 Por cada elemento encontrado, se actualizará esta interfaz (Ilustración 85) con la frecuencia en particular de cada emisión sintonizada y el usuario tiene la posibilidad de seleccionar el nombre de la emisora correspondiente a la emisión. Aparecen también los botones para añadir esa emisión al sistema y otro botón para buscar la siguiente emisión.

4 Al pulsar el botón Añadir emisión.

4.1 Si la emisión que desea añadir no existe, se almacenará en el sistema los datos de la nueva emisión (Ilustración 87).

4.2 Si la emisión que desea añadir ya existe con esos datos en esa posición, aparecerá un mensaje de aviso (Ilustración 86).

5 Al pulsar el botón Buscar siguiente, aparecerá el aviso de Buscando (Ilustración 84) que dará lugar a la interfaz explicada del punto 3.

Postcondiciones: El usuario será capaz de buscar y añadir en el sistema todas las emisiones cercanas a su posición para un posterior uso de la aplicación FM Radio RDS. Estos registros aparecerán en el listado de emisoras y emisiones añadidos del usuario concreto.

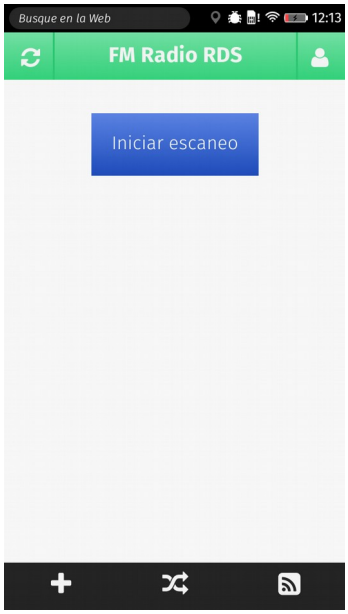


Ilustración 82: UI_logged _scan_iniciar

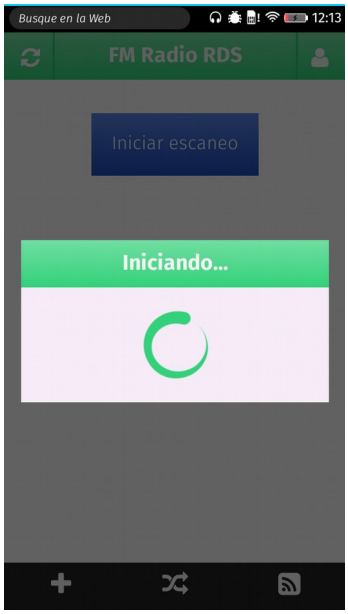


Ilustración 83: UI_logged _scan_iniciando

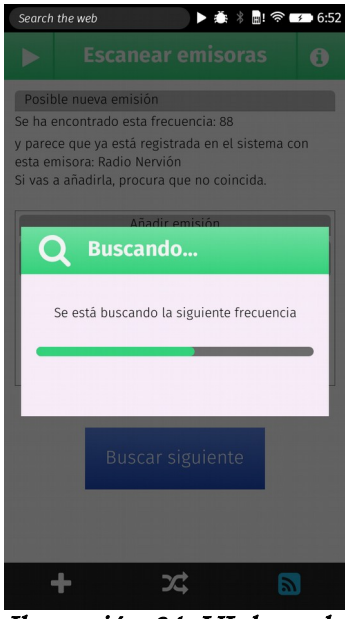


Ilustración 84: UI_logged _scan_buscando_siguiente

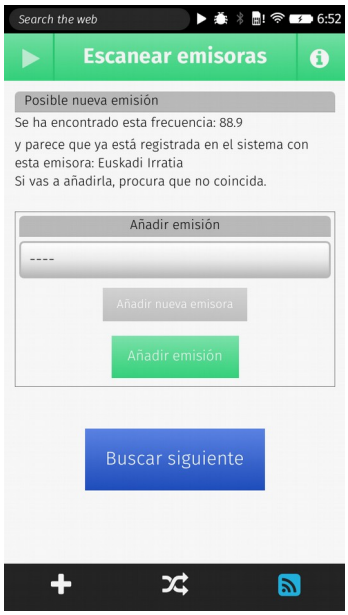


Ilustración 85: UI_logged _scan_item_encontrado

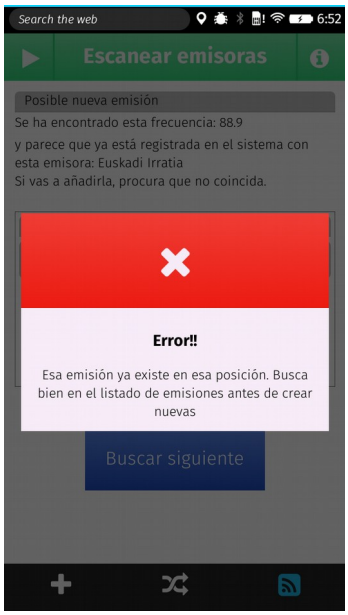


Ilustración 86: UI_logged _scan_ya_existe

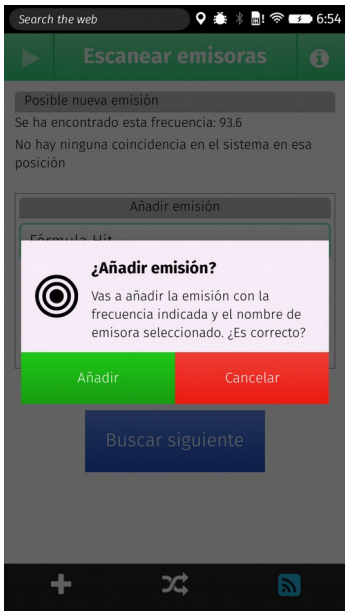


Ilustración 87: UI_logged _scan_aniadir

16 Gestionar perfil

Nombre: Gestionar perfil.

Descripción: Permite ver la información de su perfil.

Actores: Usuario identificado.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 El usuario se encuentra en la sección de su perfil (Ilustración 88). Puede ver sus datos personales que se introducen cuando el usuario se registra, sus coordenadas puntuales y datos de su geoposición.

Postcondiciones: El usuario verá la información completa de la emisora.

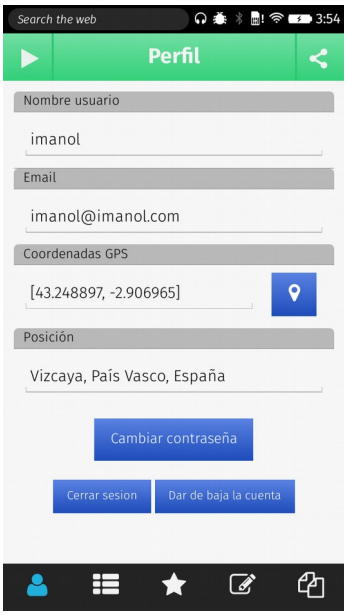


Ilustración 88: UI_logged _profile

17 Cambiar password

Nombre: Cambiar password.

Descripción: Permite cambiar la contraseña del usuario. Le aparecerá una ventana donde poder introducir el correo, la contraseña vieja y la nueva, para asegurarnos que el usuario es realmente el que está cambiando su clave

Actores: Usuario identificado.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 Habiéndose pulsado el botón "Cambiar contraseña" de la sección perfil, se accede a la ventana donde realizar el cambio de clave (Ilustración 89).

2 El usuario rellenará los campos "Email", "Contraseña actual", "Contraseña nueva" y "Repetir contraseña nueva" y pulsará en "Guardar cambios"

2.1 Si pulsa "Guardar cambios" habiendo dejado alguno de los campos sin rellenar, saltará un aviso de error (Ilustración 90).

2.2 Rellena todos los campos y pulsa el botón "Guardar cambios":

2.2.1 Si el email no existe, saltará aviso error (Ilustración 91).

2.2.2 Si el par email y contraseña actual no existen, saltará un aviso de error (Ilustración 92).

2.2.3 Si el email no tiene formato de correo electrónico, saltará un aviso de error (Ilustración 90).

2.2.4 Si las contraseña actual, nueva contraseña y/o repetir nueva contraseña son inferior a 6 caracteres, saltará un aviso de error (Ilustración 90).

2.2.5 Si la contraseña nueva y repetir nueva contraseña no son iguales, saltará un aviso de error (Ilustración 93).

2.2.6 Si la contraseña actual y nueva contraseña son iguales, saltará un aviso de error (Ilustración 93).

2.2.7 Si no ocurre nada de lo anterior, el cambio de contraseña

se realiza correctamente y salta un mensaje de aviso correcto informando (Ilustración 94).

Postcondiciones: El usuario tendrá la contraseña actualizada.

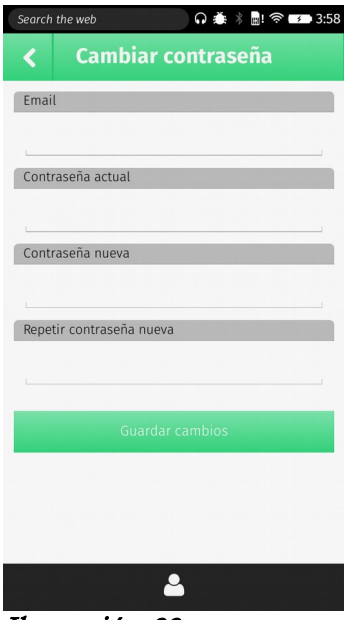


Ilustración 89: UI_logged_ cambio_pass

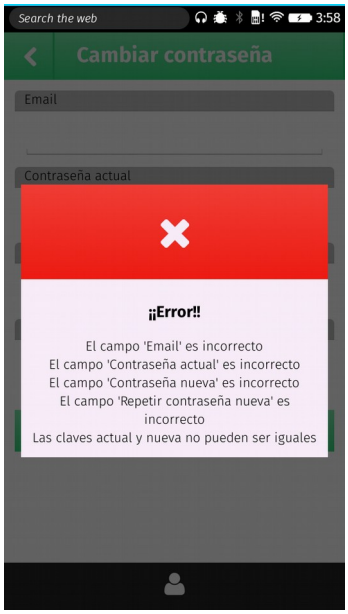


Ilustración 90: UI_logged _cambio_pass_error

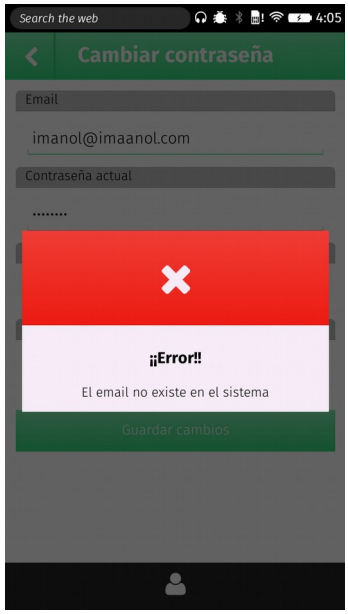


Ilustración 91: UI_logged _cambio_pass_error

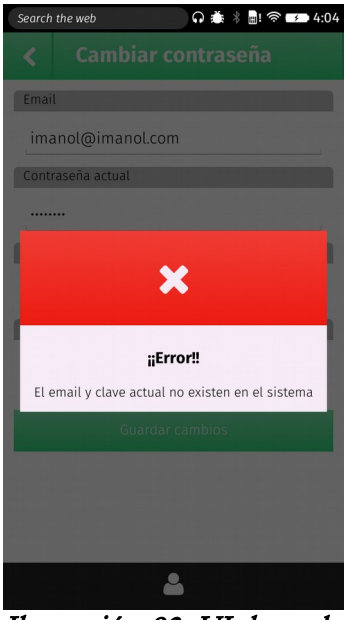


Ilustración 92: UI_logged _cambio_pass_error

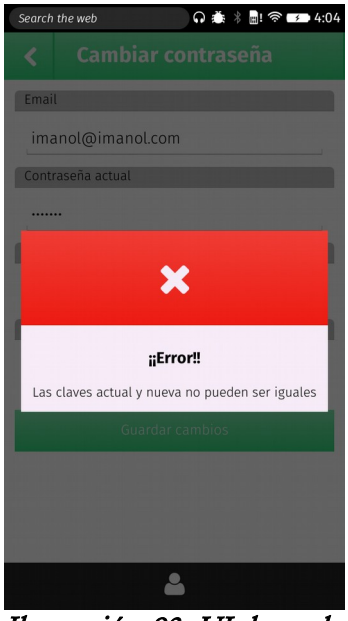


Ilustración 93: UI_logged _cambio_pass_error

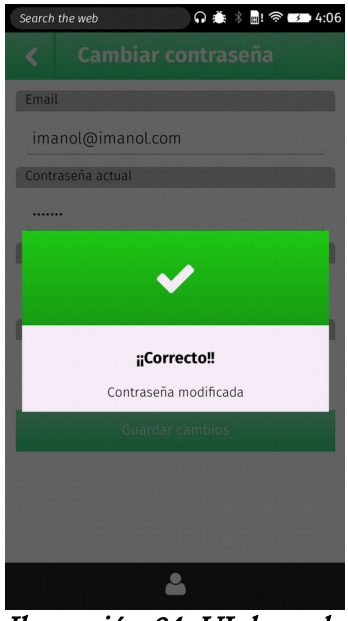


Ilustración 94: UI_logged _cambio_pass_ok

18 Cerrar sesión

Nombre: Cerrar sesión.

Descripción: Permite cerrar su sesión. Le redirigirá a la ventana principal .

Actores: Usuario identificado.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 El usuarios está en "Gestionar Perfil" y pulsa en el botón "Cerrar sesión"(Ilustración 95). Aparece un aviso, esperando confirmación:

1.1 Si pulsa "Cerrar sesión", dejará de estar identificado en el sistema

1.2 Si el usuario pulsa "Cancelar", no ocurrirá nada.

Postcondiciones: El usuario podrá cerrar sesión en la app.

Ilustración 95: UI_logged_cerrar_sesión

19 Eliminar usuario

Nombre: Eliminar usuario.

Descripción: Permite eliminar su cuenta. Lo que internamente haremos es poner al usuario como inactivo, con intención de no perder toda la información que ha ido generando el tiempo que ha sido usuario. Si el usuario vuelve a identificarse con esos datos, la cuenta se reactivará. Le redirigirá a la ventana inicial de la app.

Actores: Usuario identificado.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 El usuario está en "Gestionar Perfil" y pulsa en el botón "Dar de baja la cuenta". Se redirige a la ventana "Eliminar usuario" (Ilustración 96).

2 El usuario rellenará los campos "Email" y "Contraseña" y pulsará en "Eliminar"

2.1 Si pulsa "Eliminar" habiendo dejado alguno de los campos sin rellenar, saltará un aviso de error (Ilustración 97).

2.2 Rellena todos los campos y pulsa el botón "Eliminar":

2.2.1 Si el par email y contraseña no existen, saltará un aviso de error (Ilustración 99).

2.2.2 Si el email no tiene formato de correo electrónico y/o la contraseña es inferior a 6 caracteres, saltará un aviso de error (Ilustración 97).

2.2.3 Si no ocurre nada de lo anterior, salta un mensaje de aviso esperando respuesta del usuario (Ilustración 98).

2.2.3.1 Si realmente desea cancelar la cuenta, el usuario debe pulsar en la opción "Eliminar cuenta".

Postcondiciones: El usuario podrá cancelar su cuenta, dejándola inactiva hasta una nueva identificación.

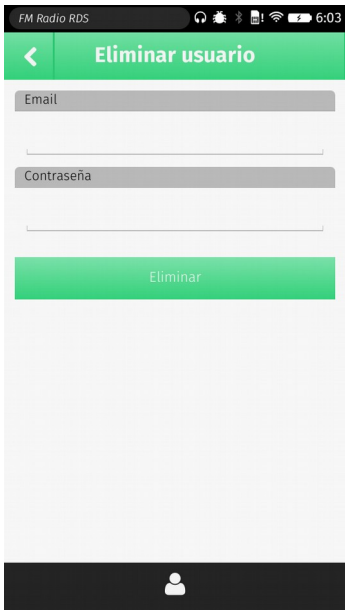


Ilustración 96: UI_logged _cancelar_cuenta

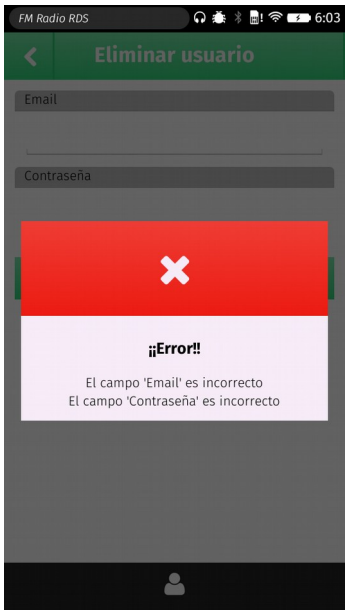


Ilustración 97: UI_logged _cancelar cuenta_error

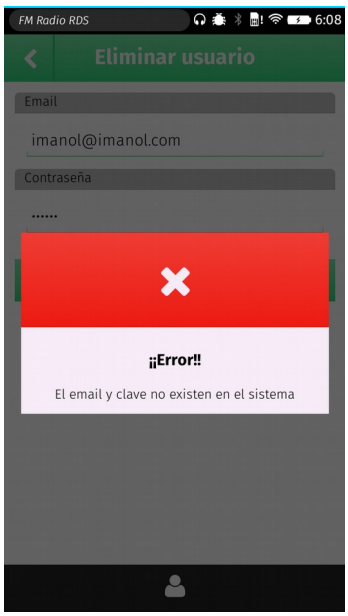


Ilustración 99: UI_logged _cancelar cuenta_error

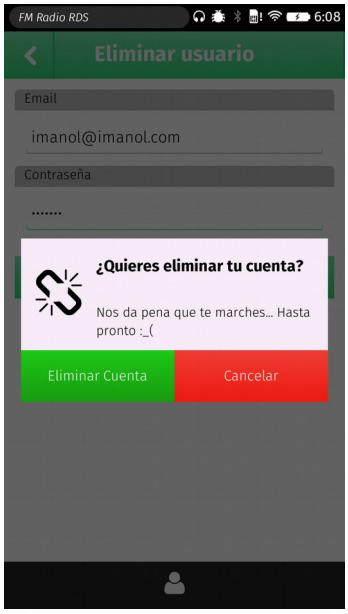


Ilustración 98: UI_logged _cancelar cuenta_ok

20 Compartir

Nombre: Compartir.

Descripción: Permite compartir un mensaje a través del correo electrónico u otro sistema que elija.

Actores: Usuario identificado.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 El usuario está en "Gestionar Perfil" y pulsa en el botón superior derecho "Compartir" (Ilustración 100), donde aparecerá un mensaje de aviso:

1.1 Si el usuario desea compartir información sobre la app a través de alguna vía del sistema operativo, pulsará "OK".

1.1.1 Se le abrirá el menú propio de Firefox OS para compartir alguna publicación o algún archivo (Ilustración 101).

Postcondiciones: El usuario podrá compartir información del sistema.

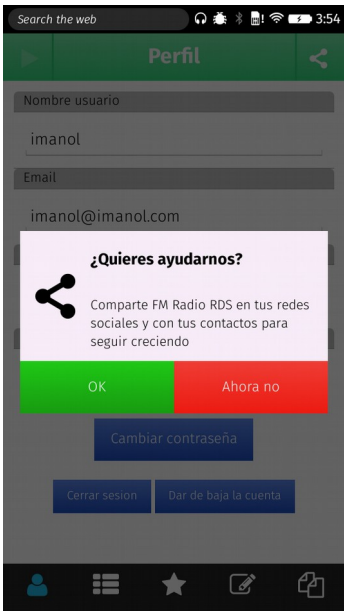


Ilustración 100: UI_logged _profile_compartir

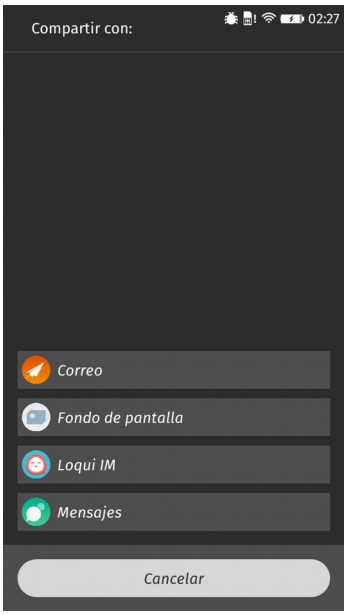


Ilustración 101: UI_logged _profile_compartir_fxos

21 Comentar sobre la app y ver comentarios realizados

Nombre: Comentar sobre la app y ver comentarios realizados.

Descripción: Permite hacer un comentario sobre la aplicación de modo público. Podrá publicar un comentario por día, con una restricción de 160 caracteres cada uno. También permite ver los comentarios que ha realizado.

Actores: Usuario identificado.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 El usuario está en "Gestionar Perfil", en la sección "Ver y añadir comentarios", segundo botón de la barra inferior(Ilustración 104).

2 Puede rellenar la caja de texto y darle al botón "Guardar comentario".

2.1 Si el comentario tiene menos de 20 caracteres o más de 160, aparecerá un mensaje de aviso (Ilustración 103).

2.2 Si el usuario ha introducido un comentario durante las últimas 24 horas, saltará un mensaje de aviso (Ilustración 105).

2.3 Si no han aparecido ninguno de los mensajes de aviso anteriores, el comentario se añadirá correctamente en el listado y el usuario lo verá en su listado de comentarios (Ilustración 106).

3 El usuario puede ver el listado de comentarios propios (Ilustración 104).

4 El usuario puede ver todos los comentarios publicados, pulsando el botón "Ver comentarios" (Ilustración 102).

Postcondiciones: El usuario podrá ver los comentarios realizados por él en el sistema, así como publicar uno máximo diario.

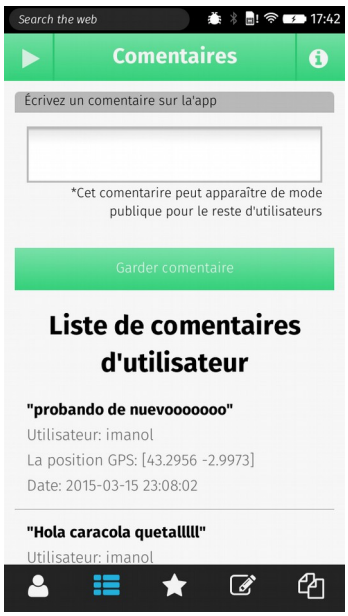


Ilustración 104: UI_logged _profile_comments

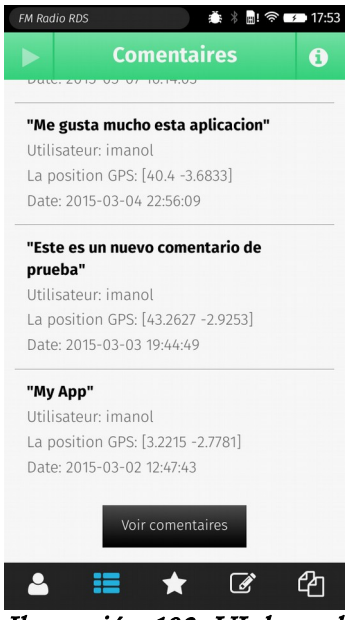


Ilustración 102: UI_logged _profile_comments_all

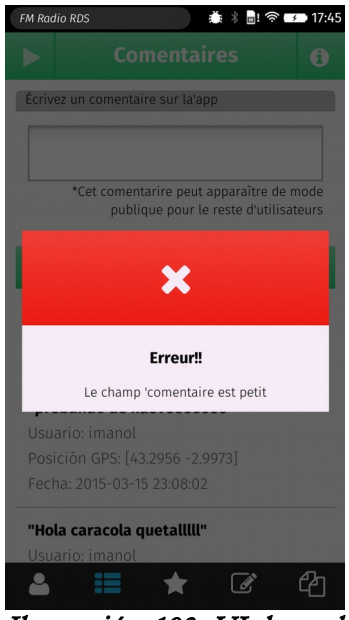


Ilustración 103: UI_logged _profile_comments_ko

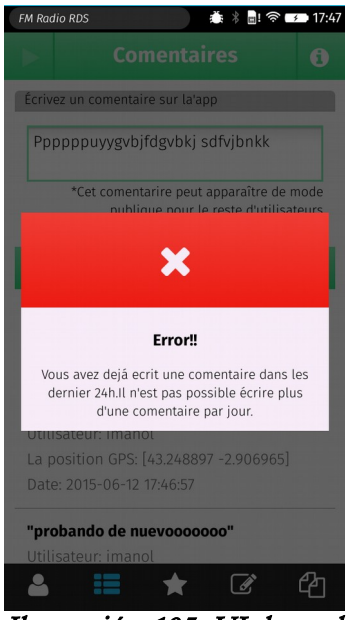


Ilustración 105: UI_logged _profile_comments_ko

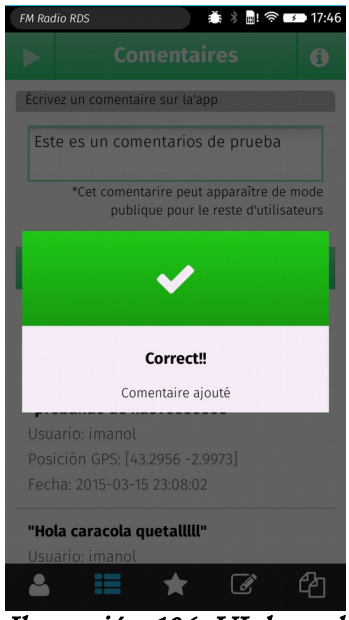


Ilustración 106: UI_logged _profile_comments_ok

22 Ver emisoras favoritas

Nombre: Ver listado de emisoras favoritas.

Descripción: Permite ver las emisoras favoritas.

Actores: Usuario identificado.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 El usuario está en "Gestionar Perfil", en la sección "Ver emisoras favoritas", tercer botón de la barra inferior. (Ilustración 107).

Postcondiciones: El usuario podrá ver el listado de emisoras que tiene seleccionadas como favoritas.

Ilustración 107: UI_logged _profile_favoritos

23 Ver valoraciones de emisiones realizadas

Nombre: Ver las valoraciones que ha realizado a una serie de emisiones.

Descripción: Permite ver las valoraciones que ha realizado sobre las emisiones.

Actores: Usuario identificado.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 El usuario está en "Gestionar Perfil", sección "Ver emisiones valoradas", cuarto botón de la barra inferior (Ilustración 108).

Postcondiciones: El usuario podrá ver el listado de emisiones a las que ha realizado una puntuación.

Ilustración 108: UI_logged_profile_puntuados

24 Ver emisiones y emisoras añadidas

Nombre: Ver las valoraciones que ha realizado a una serie de emisiones.

Descripción: Permite ver las emisiones y emisoras que ha añadido.

Actores: Usuario identificado.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 El usuario está en "Gestionar Perfil", sección "Ver emisiones valoradas", quinto botón de la barra inferior. (Ilustración 109).

Postcondiciones: El usuario podrá ver el listado de emisiones y emisoras que ha añadido en el tiempo que ha usado la aplicación.

Ilustración 109: UI_logged_profile_emisoras_y_emisiones_aniadidas

25 Registrar una incidencia

Nombre: Registrar una incidencia.

Descripción: Permite registrar una incidencia que ha podido haber durante el uso de la aplicación. Se facilitará el acceso a los diferentes datos de contacto.

Actores: Usuario.

Precondiciones: Ninguna.

Requisitos funcionales: Conexión a Internet y geoposición activados y cascos validos conectados a la entrada de auriculares.

Flujo de eventos:

1 El usuario realiza un movimiento de agitación del terminal. Al hacer el "*shake*", da igual en que apartado de la aplicación se encontrara el usuario, aparece un aviso preguntando si se desea registrar una incidencia.

1.1 Si el usuario pulsa la tecla de "Ok", se le redirige al usuario a la sección "Información App", donde se encuentran los datos de contacto tanto de los perfiles oficiales de la aplicación como del desarrollador (Ilustración 110).

Postcondiciones: El usuario ha podido ponerse en contacto vía correo electrónico, Twitter o Google Plus con los responsables y administradores para registrar la incidencia.

Ilustración 110: UI_ registrar_incidencia

Capturas de pantalla varias

Este apartado no es un caso de uso extendido al uso, sino un lugar donde presentar una serie de capturas de pantalla variadas y útiles para entender el funcionamiento completo pensado y desarrollado. Tanto la Ilustración 112: UI_notify_ gps_desactivado como la Ilustración 113: UI_notify_ cierre_sesion_inactivo, tratan sobre la situación en que un usuario que está identificado en la aplicación, puede haberse simultaneamente desde otro dispositivo dado de baja en el sistema, por lo que deberá volver a identificarse para seguir trabajando con la aplicación. En el caso de la Ilustración 111: UI_notify_ gps_desactivado, la aplicación avisará al usuario que ha sido desactivada el GPS del dispositivo, por lo que el funcionamiento de la app no será correcto.

Se han indicado en un punto aparte ya que esto sucederá en cualquier punto de la app, cuando esas situaciones ocurran, algo similar a Registrar una incidencia, pero en este caso no depende de lo que el usuario realice, sino de que el sistema le avisará y lanzará las notificaciones.

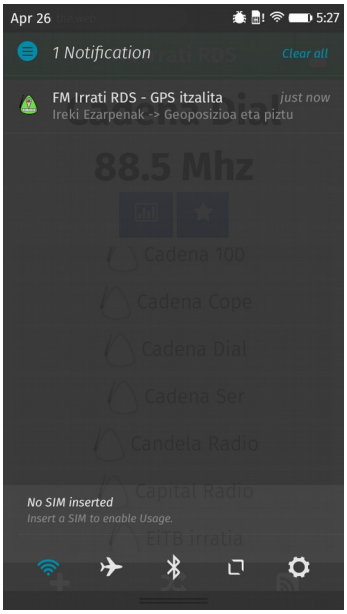


Ilustración 111: UI_notify_ gps_desactivado

Ilustración 112: UI_notify_ gps_desactivado

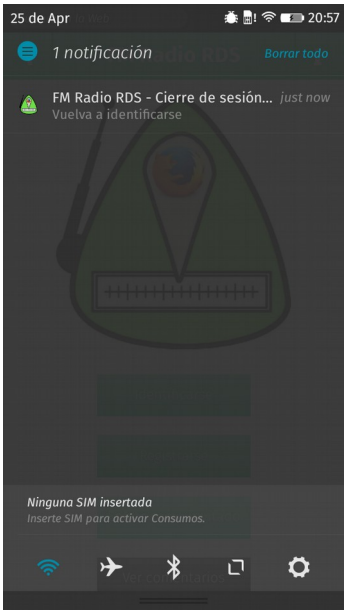


Ilustración 113: UI_notify_ cierre_sesion_inactivo

Anexo II
Diagramas de secuencia

Anexo II - Diagramas de secuencia

En el anexo II se van a desarrollar los diagramas de secuencia. Para una mejor lectura de los diagramas, algunos se presentarán en modo apaisado.

0 .- Inicio de la app

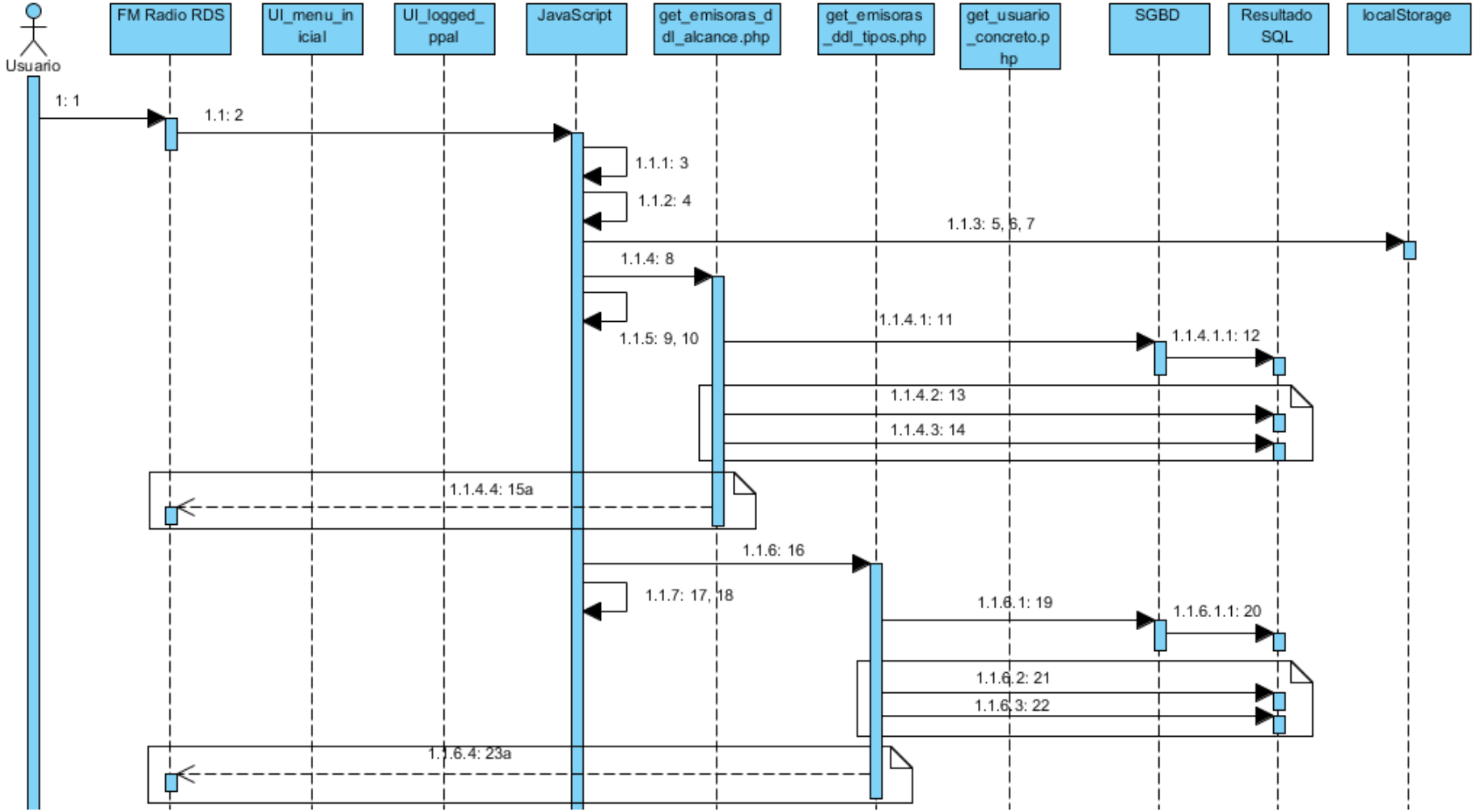


Ilustración 114: Diagrama de secuencia - Inicio de la app (primera parte)

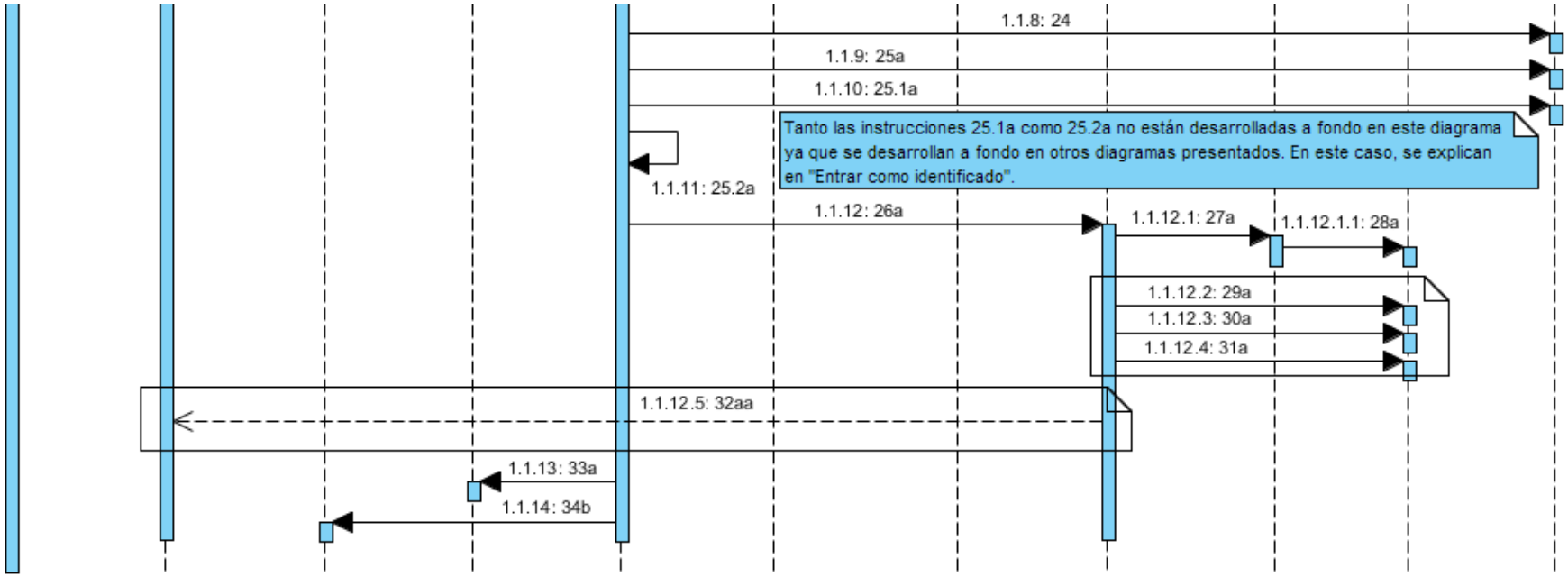


Ilustración 115: Diagrama de secuencia - Inicio de la app (segunda parte)

1.- El usuario inicia la aplicación, pulsando el icono en el escritorio.
2.- Se ejecutan una serie de funciones en segundo plano, sin la interacción del usuario con la interfaz.
3.- Obtener la posición del usuario: *geopos()*
4.- navigator.geolocation.getCurrentPosition(
getLatituteLongitude, errorFunction);
5.- Almacenar datos en almacenamiento local.
localStorage.setItem('mylatit', mylat2);
6.- Almacenar datos en almacenamiento local.
localStorage.setItem('mylongit', mylong2);
7.- Almacenar datos en almacenamiento local.
localStorage.setItem('myposicion', direccionObtenida);
//Ya se han almacenado los datos obtenidos de la posición
8.- Obtener los posibles alcances de una emisora
show_content_bd_emisoras_alcance_ddl();
9.- Obtener el idioma del navegador o dispositivo:
idioma_navegador();
10.- navigator.language
11.- mysqli_query(SQL1)
SQL1:
"SELECT alcances_tipos.cod_alcance_tipo,
alcances_tipos.nombre_alcance_tipo as nombre_alcance_tipo
FROM alcances_tipos ORDER BY nombre_alcance_tipo"
//Las consulta variará en función del valor obtenido del idioma
12.- *mysqli_result()*
[Mientras haya elementos (alcances)]
13.- $res[cod_alcance_tipo]
14.- $res[nombre_alcance_tipo]
[Fin mientras]
[Si hay elemento]
[Mientras haya elementos, recorrer]
15a.- Los datos obtenidos, escribirlos/rellenarlos en la interfaz:
echo $txt
[Fin mientras]
[Fin si]
16.- Obtener los posibles tipos de una emisora
show_content_bd_emisoras_tipos_ddl();
17.- Obtener el idioma del navegador o dispositivo:

idioma_navegador();
18.- navigator.language
19.- mysqli_query(SQL2)
SQL2: "SELECT emisoras_tipos.cod_emisora_tipo,
emisoras_tipos.nombre_emisora_tipo as nombre_emisora_tipo
FROM emisoras_tipos ORDER BY nombre_emisora_tipo"
//Las consulta variará en función del valor obtenido del idioma
20.- *mysqli_result()*
[Mientras haya elementos (tipos)]
21.- $res[cod_emisora_tipo]
22.- $res[nombre_emisora_tipo]
[Fin mientras]
[Si hay elemento]
[Mientras haya elementos, recorrer]
23a.- Los datos obtenidos, escribirlos/rellenarlos en la interfaz:
echo $txt
[Fin mientras]
[Fin si]
24.- Comprobamos si el usuario está identificado en la app
localStorage.getItem('codigo_usuario')
[Si está identificado]
25a.- Obtener datos de almacenamiento local.
localStorage.getItem('codigo_usuario');
25.1a.- *show_content_bd_radio_ui_logged();*
//Este método ya está desarrollado en profundidad en otro
//diagrama de secuencia (show_content_bd_radio_ui_logged()) y
//con intención de facilitar la lectura y comprensión del esquema,
//se incluye una referencia.
25.2a.- *poner_datos_ultima_emision_logged();*
//Este método ya está desarrollado en profundidad en otro
//diagrama de secuencia (poner_datos_ultima_emision_logged()) y
//con intención de facilitar la lectura y comprensión del esquema,
//se incluye una referencia.
26a.- Cargar datos usuario perfil
poner_datos_nuevo_usuario_perfil();
27a.- mysqli_query(SQL3)
SQL3:
"SELECT usuarios.cod_usuario, usuarios.nombre_usuario,

usuarios.email_usuario
FROM usuarios WHERE (cod_usuario = '$cod_usuario')"
28a.- *mysqli_result()*
[Mientras haya elementos (el usuario)]
29a.- $res[cod_usuario]
30a.- $res[nombre_usuario]
31a.- $res[email_usuario]
[Fin mientras]
[Si hay elemento]
[Mientras haya elementos, recorrer]
32aa.- Los datos obtenidos, escribirlos/rellenarlos en la interfaz: *echo $txt*
[Fin mientras]
33a.- Redirige a la interfaz principal de usuario identificado.
UI_logged_ppal
section_logged_radio_ui → classList.add("current")
[Si no está identificado]
34b.- Redirige a la interfaz menu inicial.**UI_menu_inicio**
section_index → classList.add("current")
[Fin si]

1 Registrarse. *Diagrama de secuencia de "Registrarse" (interfaz UI_menu_inicial).*

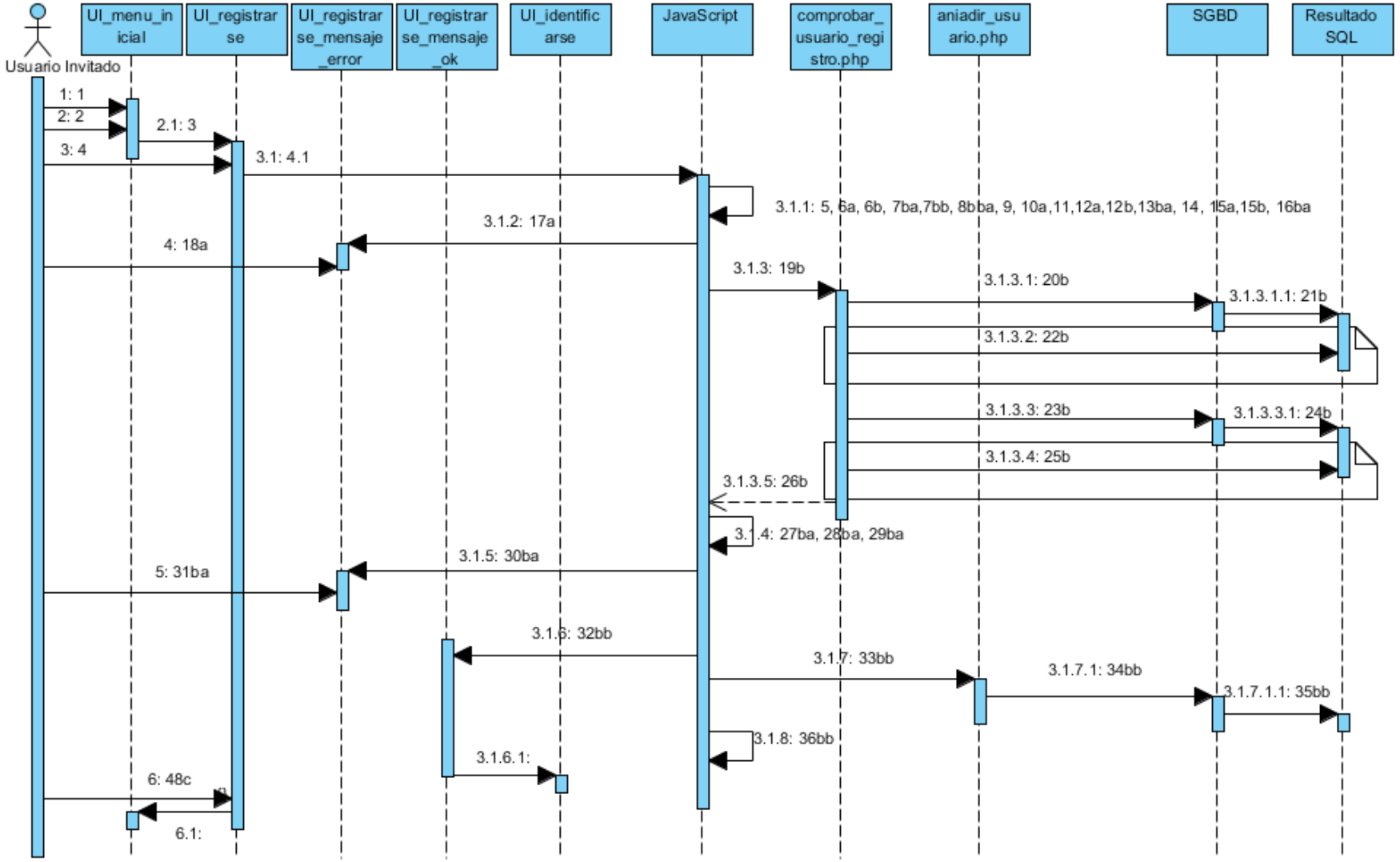


Ilustración 116: Diagrama de secuencia - Registrarse

1.- El usuario inicia la aplicación, pulsando el icono en el escritorio.
2.- En la interfaz **UI_menu_inicial**, pulsa el botón "Registrarse":
data-section="section_index_register"
3.- Se abre la nueva interfaz: **UI_registrarse**
section_index_register → classList.add("current")
4.- Rellena los campos. Pulsa en "Realizar registro":
4.1.-
document.getElementById("id_article_index_register_btn_register").addEventListener("click",function() {...});
5.- Comprueba si nombre usuario vacío: *validarNombreRegExp(nombre)*
[Si nombre es vacía]
6a.- imprimir_errores_nombre_usuario_vacia*()*
[Si no es vacía]
6b.- Comprueba si nombre tiene tamaño mín:
validarTextoTamanioMinimo(nombre)
[Si tiene tamaño muy pequeño]
7ba.- *imprimir_errores_nombre_usuario_pequenia()*
[Sino]
7bb.- Comprueba si nombre tiene tamaño grande:
validarTextoTamanioMaximo(nombre)
[Si tiene tamaño muy pequeño]
8bba.- *imprimir_errores_nombre_usuario_grande()*
[Fin si]
[Fin si]
[Fin si]
9.- Comprueba email: *validarEmailRegExp(email)*
[Si es incorrecto]
10a.- *imprimir_errores_email_valido*
[Fin si]
11.- Comprueba si pass vacía: *validarPassRegExp(pass)*
[Si pass es vacía]
12a.- *imprimir_errores_password_vacia()*
[Si no es vacía]
12b.- Comprueba si pass tamaño mín: *validarTextoTamanioMinimo(pass)*
[Si no tiene tamaño minimo]
13ba.- *imprimir_errores_password_pequenia()*
[Fin si]
[Fin si]

14.- Comprueba si pass rep vacía: *validarPassRegExp(pass_rep)*
[Si pass_rep es vacía]
15a.- *imprimir_errores_password_vacia()*
[Si no es vacía]
15b.- Comprueba si pass_rep tamaño mín:
validarTextoTamanioMinimo(pass)
    [Si no tiene tamaño minimo]
    16ba.- *imprimir_errores_password_pequenia()*
    [Fin si]
[Fin si]
[Si hay algún error]
17a.- Se abre nueva interfaz encima de la actual:
UI_registrarse_mensaje_error: *Tako.Notification.error("deny",...)*
18a.- El usuario pulsa para cerrar el mensaje.
[Si no hay error]
19b.- Comprobar usuario en BD:
comprobacion_registro_usuario_index_server(function(existe){...}) (1)
20b.- mysqli_query(SQL1)
*SQL1: "SELECT * FROM usuarios*
WHERE (usuarios.nombre_usuario = '$name_usu')"
21b.- *mysqli_result()*
[Mientras haya elementos]
22b.- $res[nombre_usuario]
[Fin mientras]
23b.- mysqli_query(SQL2)
*SQL2: "SELECT * FROM usuarios*
WHERE (usuarios.email_usuario = '$mail_usu')"
24b.- *mysqli_result()*
[Mientras haya elementos]
25b.-$res[email_usuario]
[Fin mientras]
26b.- {devuelve texto (si existe nombre, email o los dos) o vacio}
[Si devuelve texto]
    [Si devuelve estanombre]
    27ba.- *imprimir_errores_registrar_usuario_nombre_existente()*
    [Fin si]
    [Si devuelve estaemail]
    28ba.- *imprimir_errores_registrar_usuario_mail_existente()*

[Fin si]
[Si devuelve estanambos]
29ba.-
imprimir_errores_registrar_usuario_nombre_mail_existente()
[Fin si]
30ba.- Se abre nueva interfaz encima de la actual:
UI_registrarse_mensaje_error: *Tako.Notification.error("deny",...)*
31ba.- El usuario pulsa para cerrar el mensaje.

[Si no devuelve texto]
32bb.- Se abre nueva interfaz encima de la actual:
UI_identificarse_mensaje_ok *Tako.Notification.success("ok",...);*
33bb.- Añadir usuario a la BD:
nuevo_aniadir_usuario_index_server();
34bb.- mysqli_query(SQL3)
SQL3:
"INSERT INTO usuarios (nombre_usuario, password_usuario, passwordx_usuario, email_usuario, foto_usuario, activo_usuario) VALUES ('$nombre_usuario_utf', '$pass_usuario_utf', '$passx_usuario', '$email_usuario', '0', 'true')"
35bb.- *mysqli_result()*
36bb.- Poner en blanco campos:
nuevo_poner_en_blanco_aniadir_usuario_index();
37bb.- Redirige a la interfaz de identificación **UI_identificarse**.
section_index_identify → classList.add("current")

[Fin si]
36.- En **UI_registrarse**, pulsa el botón atrás:
data-article="article_index"
37.- Se abre la interfaz
UI_menu_inicial. *section_index → classList.add("current")*

2 Identificarse. Diagrama de secuencia de "Identificarse" (desde la interfaz UI_menu_inicial).

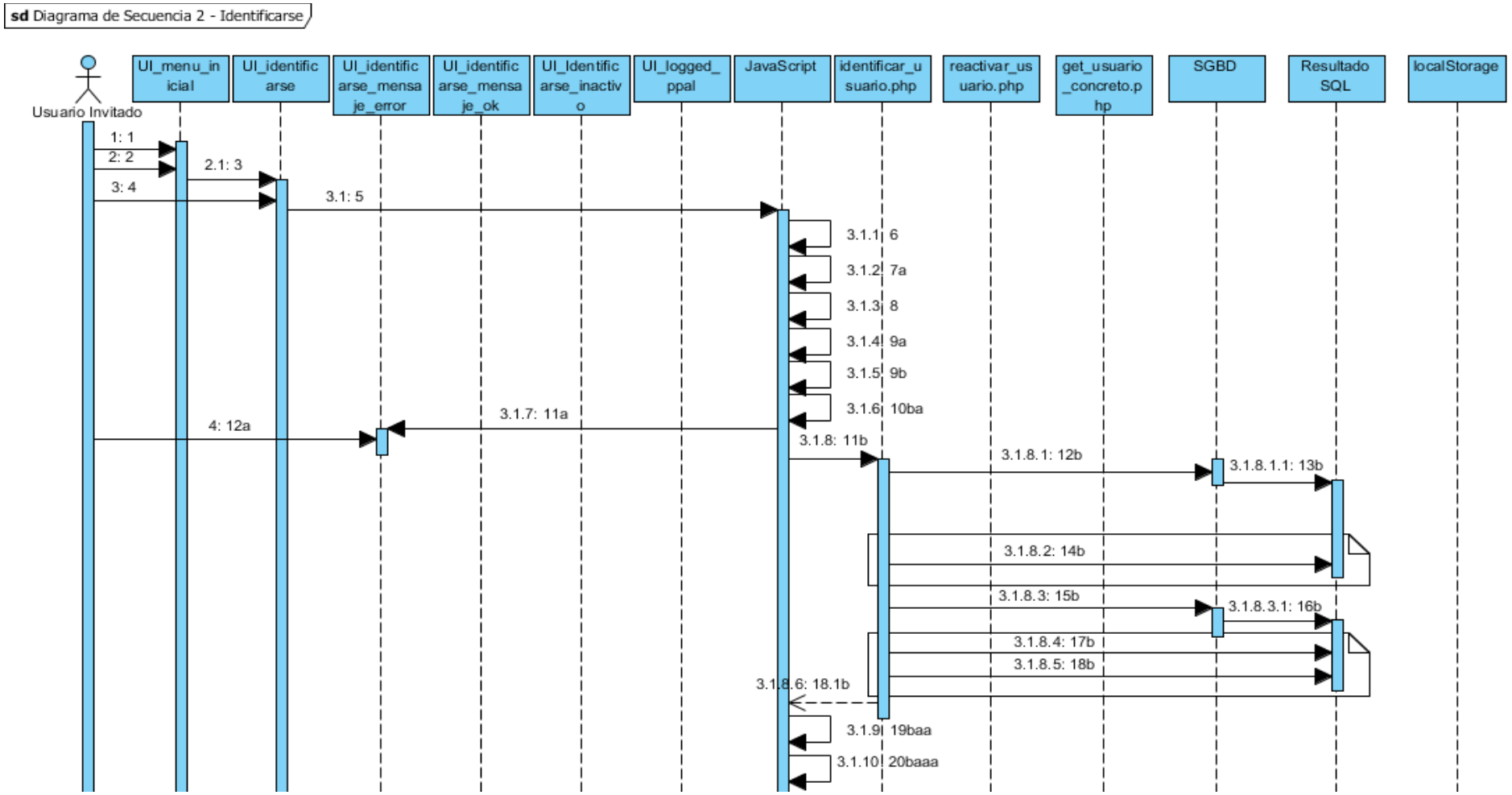


Ilustración 117: Diagrama de secuencia – Identificarse (primera parte)

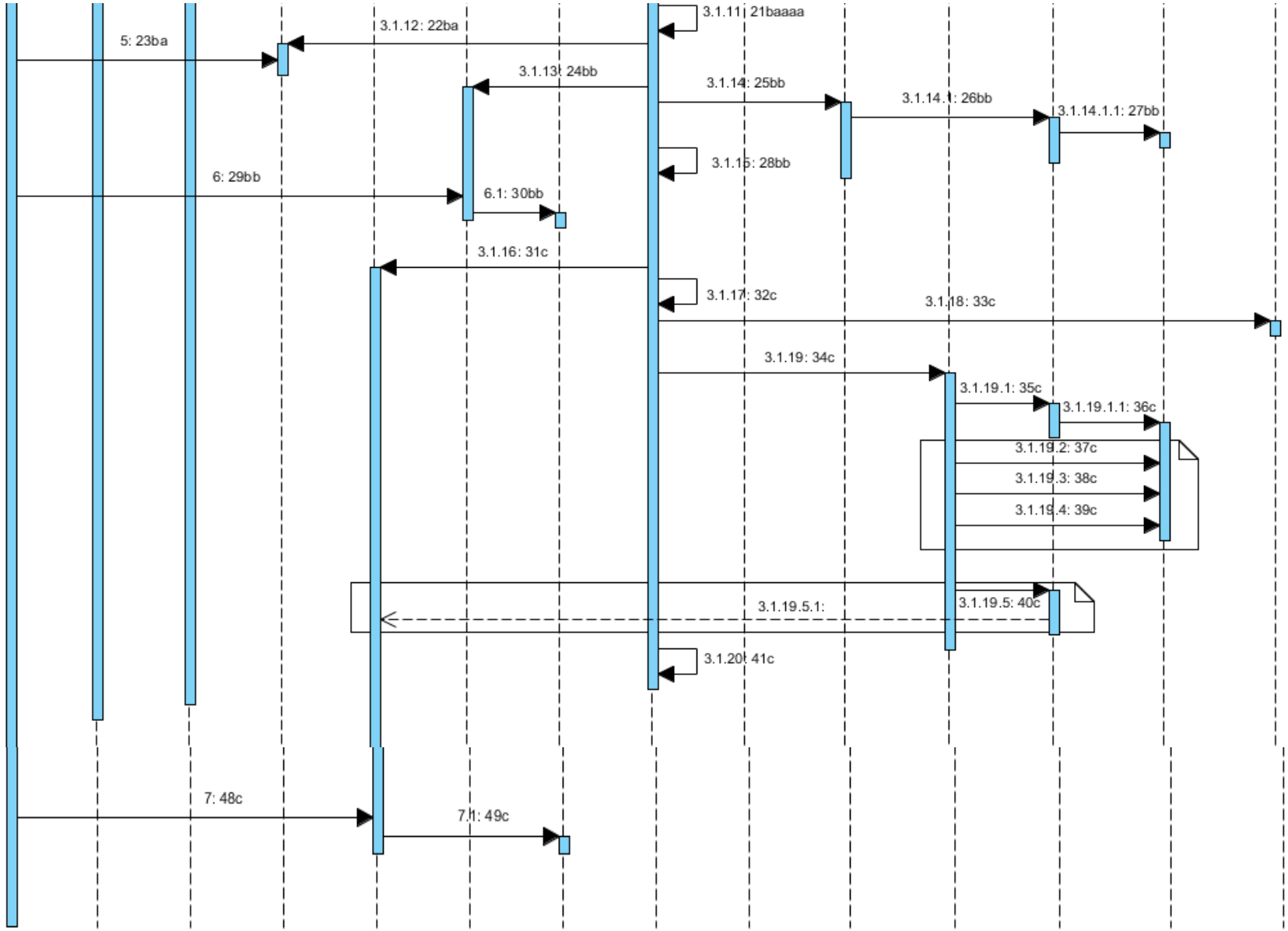


Ilustración 118: Diagrama de secuencia - Identificarse (segunda parte)

1.- El usuario inicia la aplicación, pulsando el icono en el escritorio.
2.- En la interfaz **UI_menu_inicial** pulsa botón "Identificarse":
data-section="section_index_identify".
3.- Se abre la nueva interfaz: **UI_identificarse**
section_index_identify → classList.add("current")
4.- Rellena los campos. Pulsa en "Entrar".
5.-*document.getElementById("id_article_index_identify_btn_indentify").*
addEventListener("click",function() {...});
6.- Comprueba email: *validarEmailRegExp(email)*
[Si es incorrecto]
7a.- *imprimir_errores_email_valido*
[Fin si]
8.- Comprueba si pass vacía: *validarPassRegExp(pass)*
[Si pass es vacía]
9a.- *imprimir_errores_password_vacia()*
[Si no es vacía]
9b.- Comprueba si pass tamaño mín: *validarTextoTamanioMinimo(pass)*
[Si no tiene tamaño minimo]
10ba.- *imprimir_errores_password_pequenia()*
[Fin si]
[Fin si]
[Si hay algún error]
11a.- Se abre nueva interfaz encima de la actual:
UI_identificarse_mensaje_error: *Tako.Notification.error("deny",...)*
12a.- El usuario pulsa para cerrar el mensaje.
[Si no hay error]
11b.- Comprobar usuario en BD: {devuelve texto o numero}
nuevo_identificar_usuario_index_server(function(existe){...})
12b.- mysqli_query(SQL1)
SQL1: "SELECT usuarios.cod_usuario, usuarios.nombre_usuario,
usuarios.password_usuario,usuarios.passwordx_usuario,
usuarios.email_usuario FROM usuarios
WHERE (usuarios.email_usuario = '$email_usuario')"
13b.- *mysqli_result()*
[Mientras haya elementos]
14b.- $res[cod_usuario]
[Fin mientras]
15b.- mysqli_query(SQL2)

(Le pasamos las variables $email_usuario,$pass_usuario,$passx_usuario)
SQL2: "SELECT usuarios.cod_usuario, usuarios.nombre_usuario, usuarios.password_usuario, usuarios.passwordx_usuario, usuarios.email_usuario,usuarios.activo_usuario FROM usuarios WHERE (usuarios.email_usuario = '$email_usuario') AND (usuarios.password_usuario = '$pass_usuario') AND (usuarios.passwordx_usuario = '$passx_usuario')"
16b.- *mysqli_result()*
[Mientras haya elementos]
17b.-$res2[cod_usuario]
18b.-$res2[activo_usuario]
[Fin mientras]
18.1b.- {devuelve texto o numero}
[Si devuelve texto]
[Si devuelve email]
19baa.- imprimir_errores_identificar_usuario_email_inexistente*()*
[Si devuelve emailclave]
20baaa.-
imprimir_errores_identificar_usuario_email_clave_no_coinciden()
[Si devuelve usuarionoactivo]
21baaaa.-
imprimir_errores_identificar_usuario_no_activo_info*()*
[Fin si]
[Fin si]
[Fin si]
[Si devuelve email o emailclave]
22ba.- Se abre nueva interfaz encima de la actual:
UI_identificarse_mensaje_error: *Tako.Notification.error("deny",...)*
23ba.- El usuario pulsa para cerrar el mensaje.
[Sino (devuelve usuarionoactivo)]
24bb.- Se abre nueva interfaz encima de la actual:
UI_identificarse_inactivo *Tako.Notification.custom(...)*
25bb.- Activar usuario en BD:
nuevo_reactivar_usuario_server(); (1)
nuevo_reactivar_usuario_server_notlogged(); (2)
26bb.- mysqli_query(SQL3)
(Le pasamos variab $email_usuario,$pass_usuario,$passx_usuario)
SQL3: "UPDATE usuarios SET usuarios.activo_usuario = 'true'

WHERE (usuarios.email_usuario = '$email_usuario')
AND (usuarios.password_usuario = '$pass_usuario')
AND (usuarios.passwordx_usuario = '$passx_usuario')
AND (usuarios.activo_usuario = 'false')"
27bb.- *mysqli_result()*
28bb.- Poner en blanco los campos:
nuevo_poner_en_blanco_identificar_usuario_index();(1)
nuevo_poner_en_blanco_identificar_usuario_notlogged();(2)
29bb.- El usuario pulsa para cerrar el mensaje.
30bb.- Se redirecciona a la interfaz inicial.

UI_menu_inicial *section_index → classList.add("current")*

[Fin si]
[Si no es texto, es numero]
31c.- Se abre nueva interfaz encima de la actual:
UI_identificarse_mensaje_ok *Tako.Notification.success("ok",...);*
32c.- Identificar usuario en BD: *poner_nuevo_usuario_identificar(existe);*
33c.- *localStorage.setItem(codigo_usuario)*
34c.- Coger datos usuario: *poner_datos_nuevo_usuario_perfil();*
35c.- mysqli_query(SQL4)
SQL4: "SELECT usuarios.cod_usuario, usuarios.nombre_usuario, usuarios.email_usuario FROM usuarios
WHERE (cod_usuario = '$cod_usuario')"
36c.- *mysqli_result()*
[Mientras haya elementos]
37c.- $res[cod_usuario]
38c.- $res[nombre_usuario]
39c.- $res[email_usuario]
[Fin mientras]
[Mientras haya elementos]
40c.- Los datos obtenidos, escribirlos/rellenarlos en la interfaz:
echo $txt
[Fin mientras]
41c.- Poner en blanco campos:
nuevo_poner_en_blanco_identificar_usuario_index();
42c.- Obtener emisoras cercanas.
show_content_bd_radio_ui_logged()[8]
[Fin si]

8 Desarollado extensamente en el diagrama de secuencia "Entrar como Identificado"

48c.- El usuario pulsa para cerrar el mensaje.
49c.- Se redirecciona a la interfaz principal del usuario loggeado.
UI_logged_ppal *section_logged_radio_ui → classList.add("current")*
//50d.- Si el usuario pulsa atrás: *// data-article="article_index"*

3 Ver información y datos de acceso del desarrollador de la app

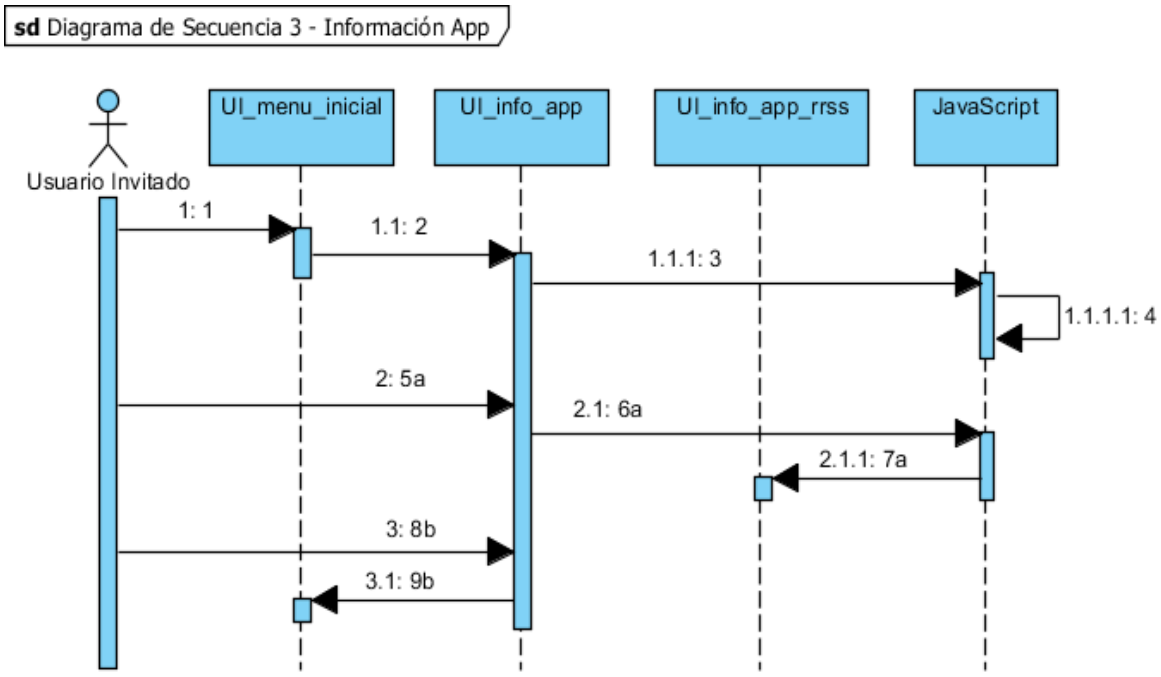


Ilustración 119: Diagrama de secuencia - Ver información y datos de acceso del desarrollador de la app

Precondición: ya está la app iniciada
1.- En la interfaz **UI_menu_inicial,** el usuario pulsa en "i" (Información): *data-article="article_info".*
2.- Se abre la nueva interfaz **UI_info_app**:
article_info➔ classList.add("current")
3.-
document.getElementById("id_article_info_btn").addEventListener("click",function(){...});
4.- Se actualiza el valor de una variable:
back_path_info_index();
[Si pulsa en un enlace (por ejemplo, perfil twitter de FM Radio RDS)]
5a.- Pulsa en enlace: *id="view_url_twitter_profile_fmradiords_firefoxos"*
6a.- *document.querySelector*
("#view_url_twitter_profile_fmradiords_firefoxos");
7a.- Se abre la ventana externa **UI_info_app_rrss:**
new MozActivity({name: "view",

data: {type: "url", url: "https://twitter.com/fmradiords"}});
[Fin si]
[Si pulsa botón atrás]
8b.- Pulsa el botón "Atrás".
data-section="section_index"
9b.- Se abre la nueva interfaz **UI_menu_inicial:**
section_index➔ classList.add("current")
[Fin si]

4 Ver comentarios

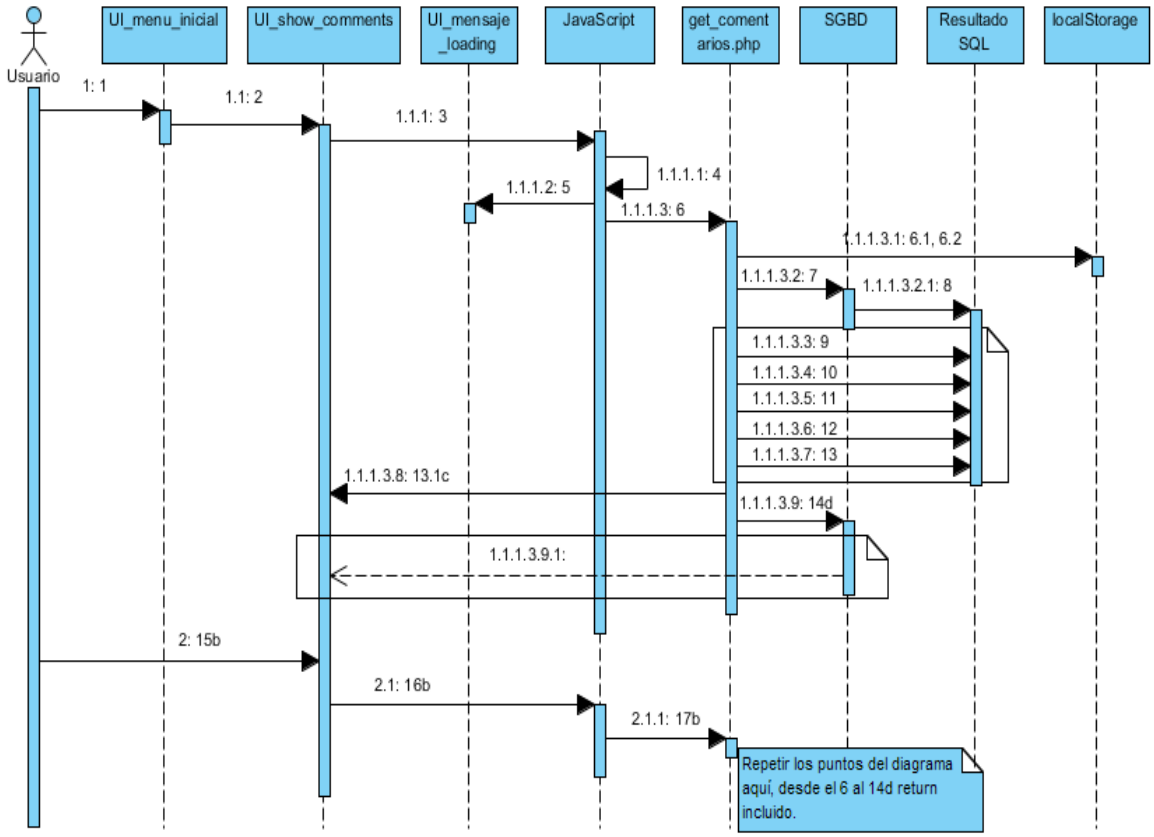


Ilustración 120: Diagrama de secuencia - Ver comentarios

Precondición: ya está al app iniciada
1.- En la interfaz **UI_menu_inicial,** el usuario pulsa en "Ver comentarios" *data-section="section_comments".*
2.- Se abre la nueva interfaz **UI_show_comments**:
section_comments➔ classList.add("current")
3.-
document.getElementById("id_btn_comments").addEventListener("click",function(){...});
4.- Se actualiza el valor de una variable:
back_path_comment_index();
[Mientras carga, esos primeros dos segundos]
5.- Se abre nueva interfaz encima de la actual:
UI_mensaje_loading *Tako.Notification.loading(...);*

6.- Obtener comentarios de la base de datos.
show_content_bd_comentarios();
6.1.- Obtención datos locales para comentarios cercanos:
localStorage.getItem('mylatit')
6.2.- Obtención datos locales para comentarios cercanos:
localStorage.getItem('mylongit')
7.- *mysqli_query(SQL1)*
SQL1:
"SELECT usuarios.nombre_usuario, comentarios.latitud_comentario, comentarios.longitud_comentario, comentarios.fecha_hora, comentarios.texto_comentario FROM comentarios
INNER JOIN usuarios
ON (usuarios.cod_usuario = comentarios.cod_usuario)
WHERE ('$resu_lat_menos' < comentarios.latitud_comentario)
AND (comentarios.longitud_comentario < '$resu_lat_mas')
AND ('$resu_long_menos' < comentarios.latitud_comentario)
AND (comentarios.longitud_comentario < '$resu_long_mas')
ORDER BY comentarios.fecha_hora DESC"
8.- *mysqli_result()*
[Mientras haya elementos]
9.- *$nombre_usuario[$k] = "$res[nombre_usuario]";*
10.-*$latitud_comentario[$k] = "$res[latitud_comentario]";*
11.- *$longitud_comentario[$k] = "$res[longitud_comentario]";*
12.- *fecha_hora[$k] = "$res[fecha_hora]";*
13.- *$texto_comentario[$k] = "$res[texto_comentario]";*
[Fin mientras]
[Si no hay elementos]
    13.1c.- Se escribe que no hay elementos.
[Si hay elementos]
    [Mientras haya elementos]
    14d.- Los datos obtenidos, escribirlos/rellenarlos en la interfaz:
    echo $txt
    [Fin mientras]
[Fin si]
[Si usuario hace scroll desde parte superior de la página]
15b.- Hace scroll hacia abajo
16b.- *window.puller = Tako.Pull_Refresh("section_comments", {onRefresh:function(){...});*

17b.- Obtener comentarios de la base de datos.
show_content_bd_comentarios();
(En el diagrama se ha indicado la repetición del código, desde el punto 7 al 14 incluidos.)
[Fin si]

5 Entrar como invitado

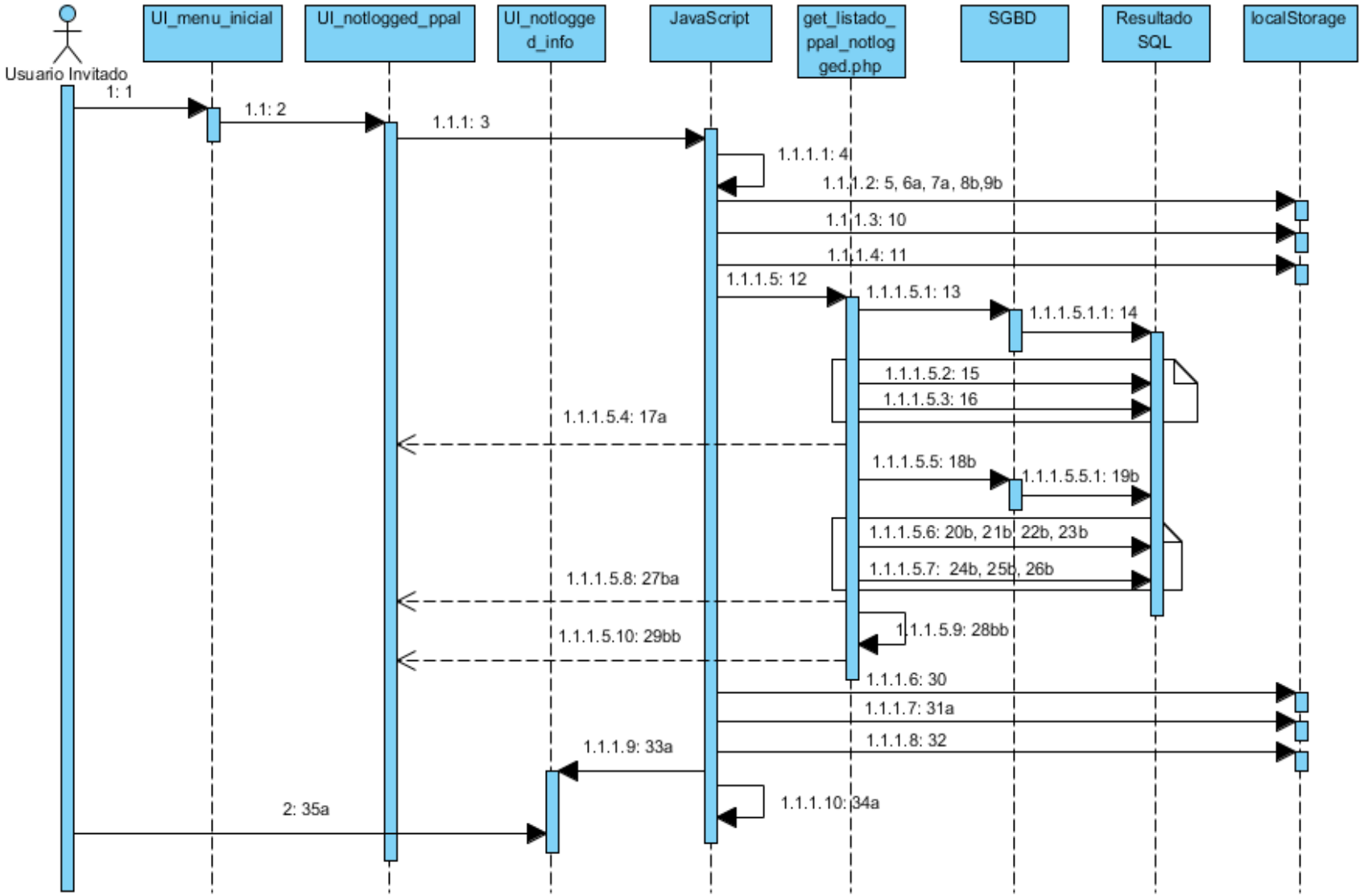


Ilustración 121: Diagrama de secuencia - Entrar como invitado

Precondición: la app está iniciada.
1.- En la interfaz **UI_menu_inicial,** el usuario pulsa en "Entrar como invitado" *data-section="section_notlogged_radio_ui".*
2.- Se abre la nueva interfaz **UI_notlogged_ppal**:
section_notlogged_radio_ui→ classList.add("current")
3.- *$("#section_notlogged_radio_ui").on("load", function(){...});*
4.- Se actualiza el valor de variables locales:
poner_datos_ultima_emision_notlogged();
5.- Obtención datos locales para última emisora:
localStorage.getItem('last_codigo_emisora')
[Si es vacía]
6a.- Obtención datos locales para frecuencia última emisora: *document.getElementById("id_article_notlogged_nombre_emisora_seleccionada").innerHTML = localStorage.getItem('last_nombre_emisora');*
7a.-
document.getElementById("id_article_notlogged_frecuencia_emisora_seleccionada").innerHTML = localStorage.getItem('last_frecuencia_emision') + " Mhz";
[Si no]
8b- Obtención datos locales para frecuencia última emisora:
document.getElementById("id_article_notlogged_nombre_emisora_seleccionada").innerHTML = "---";
9b.-
document.getElementById("id_article_notlogged_frecuencia_emisora_seleccionada").innerHTML = " 00.00 Mhz";
[Fin si]
10.- Obtención datos locales (latitud): *localStorage.getItem('mylatit');*
11.- Obtención datos locales (longitud): *localStorage.getItem('mylongit');*
12.- Obtener emisoras cercanas.
show_content_bd_radio_ui_notlogged();
13.- *mysqli_query(SQL1)*
SQL1:
"SELECT emisoras.cod_emisora, emisoras.nombre_emisora
FROM emisoras ORDER BY emisoras.nombre_emisora ASC";
14.- *mysqli_result()*
[Mientras haya elementos (emisoras)]
15.- *$res[cod_emisora]*
16.- $*res[nombre_emisora]*

[Fin mientras]
[Si no hay elementos (ni una emisora)]
17a.- Se escribe/rellena la interfaz que no hay emisoras cerca: *echo $txt*
[Si si hay elementos]
[Mientras haya elementos (emisoras), recorremos una a una]
18b.- *mysqli_query(SQL2)*
SQL2:"SELECT emisiones.cod_emision, emisiones.frecuencia_emision, emisiones.latitud_emision, emisiones.longitud_emision, emisiones.posicion_emision, emisiones.fecha_hora, emisiones.cod_emisora FROM emisiones
WHERE (emisiones.cod_emisora = '$codig_emisoraa')
AND ('$resu_lat_menos' < emisiones.latitud_emision)
AND (emisiones.latitud_emision < '$resu_lat_mas')
AND ('$resu_long_menos' < emisiones.longitud_emision)
AND (emisiones.longitud_emision < '$resu_long_mas')
ORDER BY emisiones.fecha_hora DESC"
19b.- *mysqli_result()*
[Mientras haya elementos (emisiones), obtener emisiones de la emisora]
20b.- $res[cod_emision]
21b.- $res[frecuencia_emision]
22b.- $res[latitud_emision]
23b.- $res[longitud_emision]
24b.- $res[posicion_emision]
25b.- $res[fecha_hora]
26b.- $res[cod_emisora]
[Fin mientras]
[Si hay elementos (emisiones) (se sabe de antes si hay)]
    [Si hay solo una emisión de esa emisora]
    27ba.- Se escribe/rellena la interfaz con ese par emisora&emisión: *echo $txt*
    [Si hay más de una emisión de esa emisora (elegir la mejor)]
    [Recorrer las emisiones cercanas de esa emisora para elegir la mejor]
    28bb.- Se obtiene la mejor emisión.
    [Fin recorrer emisiones de emisora]
    29bb.- Se escribe/rellena la interfaz con el par emisora&emisión mejor obtenido: *echo $txt*
    [Fin Si]

[Fin Si]
[Fin mientras]
[Fin Si]
30.- Comprobar si se ha seleccionado alguna emisora:
localStorage.getItem('seleccionado_alguna_vez') == "true"
[Si se ha seleccionado]
31a.- Marcar el elemento seleccionado en el listado.
document.getElementById(localStorage.getItem('codigo_emisora')).class List.add("cada_emisora_seleccionada");
[Fin si]
32.- Comprobar si es la primera vez que se accede a esa sección:
localStorage.getItem('info_ppal_notlogged_visto') != "true"
[Si es la primera vez]
33a.- Se abre nueva interfaz encima de la actual:
UI_notlogged_info *Tako.Notification.custom(...)*
34a.- Marcar que ya se ha accedido .
localStorage.setItem('info_ppal_notlogged_visto',"true");
35a.- El usuario pulsa el mensaje para cerrar la vista emergente
[Fin si]

6 Entrar como usuario identificado

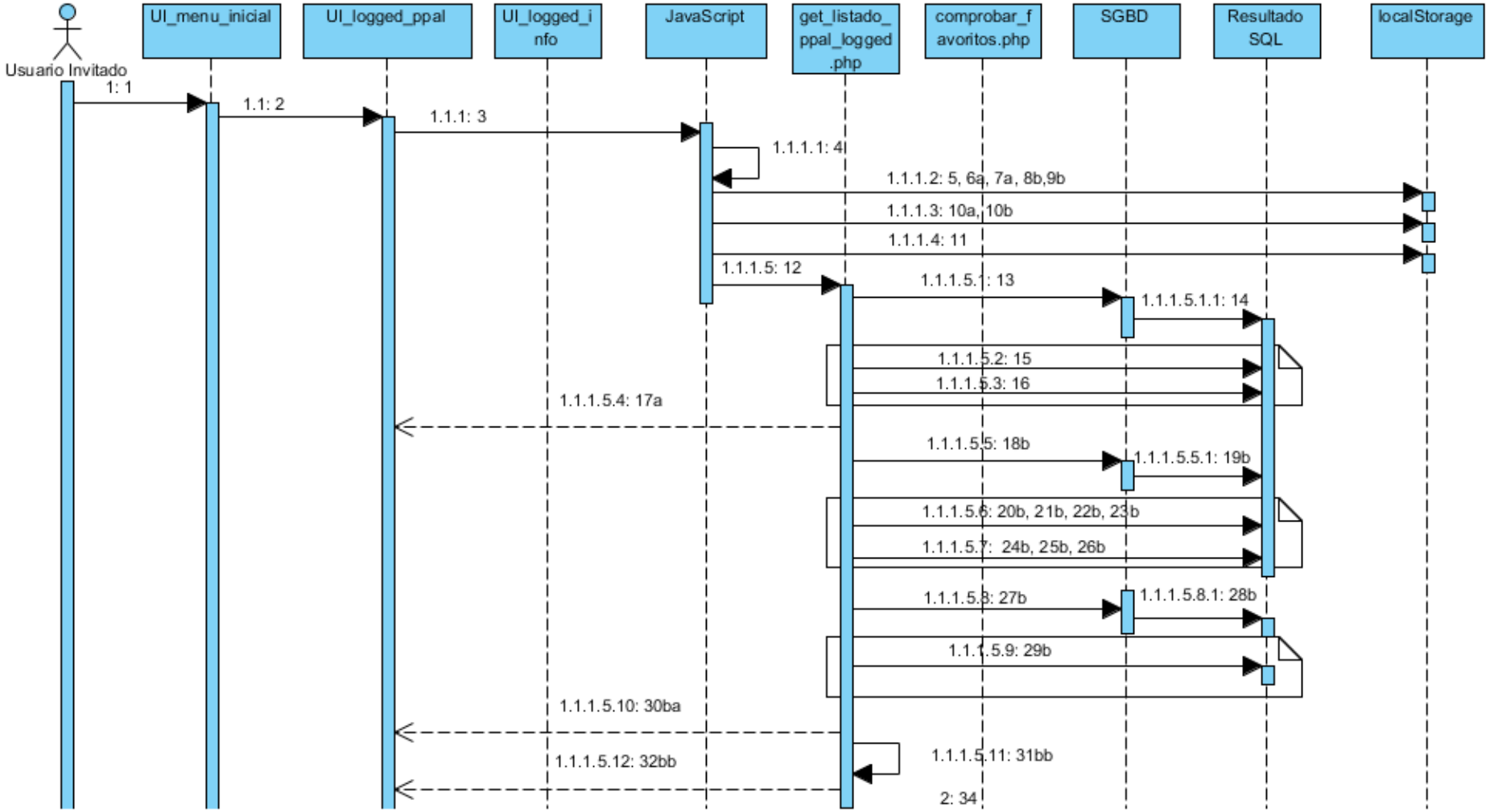


Ilustración 122: Diagrama de secuencia - Entrar como usuario identificado (primera parte)

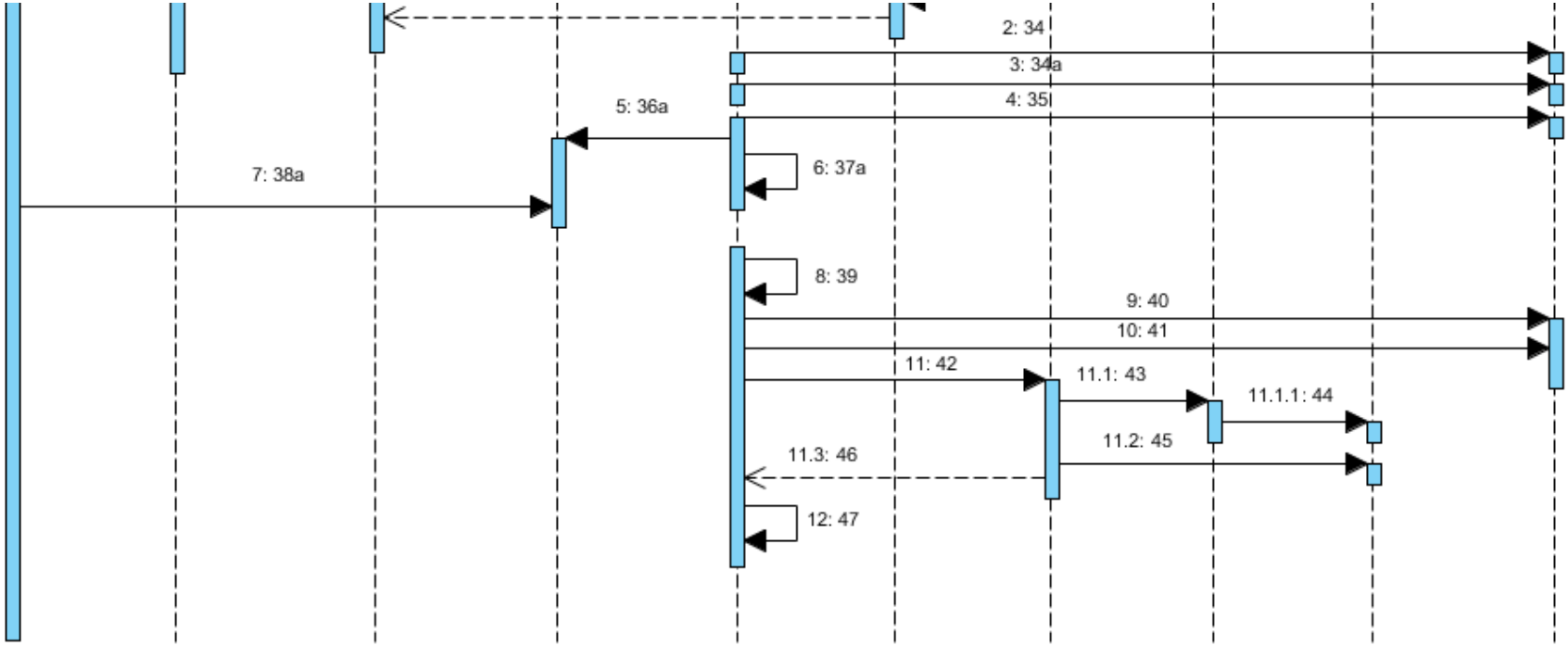


Ilustración 123: Diagrama de secuencia - Entrar como usuario identificado (segunda parte)

Precondición: la app está iniciada. Se accede a esta sección tanto si el usuario se acaba de identificar como si ha arrancado la app y previamente se había identificado y no había cerrado sesión o dado de baja su cuenta.

1.- El usuario pulsa en "Entrar" en la interfaz **UI_identificarse** o arranca la aplicación habiéndose identificado previamente y sin cerrar la sesión . *data-section="section_logged_radio_ui".*

2.- Se abre la nueva interfaz **UI_logged_ppal**:

section_logged_radio_ui → classList.add("current")

3.- *$("#section_logged_radio_ui").on("load", function(){...});*

4.- Se actualiza el valor de variables locales:

poner_datos_ultima_emision_logged();

5.- Obtención datos locales para última emisora:

localStorage.getItem('last_codigo_emisora')

[Si es vacía]

6a.- Obtención datos locales para frecuencia última emisora: *document.getElementById("id_article_logged_nombre_emisora_seleccionada").innerHTML = localStorage.getItem('last_nombre_emisora');*

7a.-
document.getElementById("id_article_logged_frecuencia_emisora_seleccionada").innerHTML = localStorage.getItem('last_frecuencia_emision') + " Mhz";

[Si no]

8b- Obtención datos locales para frecuencia última emisora: *document.getElementById("id_article_logged_nombre_emisora_seleccionada").innerHTML = "---";*

9b.-
document.getElementById("id_article_logged_frecuencia_emisora_seleccionada").innerHTML = "00.00 Mhz";

10.1.- Obtención datos locales (codigo_usuario):

localStorage.getItem('codigo_usuario');

10.2.- Obtención datos locales (latitud): *localStorage.getItem('mylatit');*

11.- Obtención datos locales (longitud): *localStorage.getItem('mylongit');*

12.- Obtener emisoras cercanas.

show_content_bd_radio_ui_logged();

13.- *mysqli_query(SQL1)*

SQL1:

"SELECT emisoras.cod_emisora, emisoras.nombre_emisora

FROM emisoras ORDER BY emisoras.nombre_emisora ASC";
14.- *mysqli_result()*
[Mientras haya elementos (emisoras)]
15.- *$res[cod_emisora]*
16.- $*res[nombre_emisora]*
[Fin mientras]
[Si no hay elementos (ni una emisora)]
17a.- Se escribe/rellena la interfaz que no hay emisoras cerca: *echo $txt*
[Si si hay elementos]
[Mientras haya elementos (emisoras), recorremos una a una]
18b.- *mysqli_query(SQL2)*
SQL2:"SELECT emisiones.cod_emision, emisiones.frecuencia_emision, emisiones.latitud_emision, emisiones.longitud_emision, emisiones.posicion_emision, emisiones.fecha_hora, emisiones.cod_emisora FROM emisiones
WHERE (emisiones.cod_emisora = '$codig_emisoraa')
AND ('$resu_lat_menos' < emisiones.latitud_emision)
AND (emisiones.latitud_emision < '$resu_lat_mas')
AND ('$resu_long_menos' < emisiones.longitud_emision)
AND (emisiones.longitud_emision < '$resu_long_mas')
ORDER BY emisiones.fecha_hora DESC"
19b.- *mysqli_result()*
[Mientras haya elementos (emisiones), obtener emisiones de la emisora]
20b.- $res[cod_emision]
21b.- $res[frecuencia_emision]
22b.- $res[latitud_emision]
23b.- $res[longitud_emision]
24b.- $res[posicion_emision]
25b.- $res[fecha_hora]
26b.- $res[cod_emisora]
[Fin mientras]
27b.- Obtenemos favoritos: *mysqli_query(SQL3)*
SQL3:" SELECT favoritos.cod_emisora FROM favoritos
WHERE (favoritos.cod_usuario = '$cod_usu')";
28b.- *mysqli_result()*
[Mientras haya elementos (favoritos)]
29b.- $res_fav[cod_emisora]
[Fin mientras]

[Si el usuario tiene favorita la emisora actual]
//Actualizar el valor de una variable, todo dentro del PHP.
[Fin si]
[Si hay elementos (emisiones) (se sabe de antes si hay)]
[Si hay solo una emisión de esa emisora]
30ba.- Se escribe/rellena la interfaz con ese par emisora&emisión: *echo $txt*
[Si hay más de una emisión de esa emisora (elegir la mejor)]
[Recorrer las emisiones cercanas de esa emisora para elegir la mejor]
31bb.- Se obtiene la mejor emisión.
[Fin recorrer emisiones de emisora]
32bb.- Se escribe/rellena la interfaz con el par emisora&emisión mejor obtenido: *echo $txt*
[Fin Si]
[Fin Si]
[Fin mientras]
[Fin Si]
33.- Comprobar si se ha seleccionado alguna emisora:
localStorage.getItem('seleccionado_alguna_vez') == "true"
[Si se ha seleccionado]
34.- Marcar el elemento seleccionado en el listado.
document.getElementById(localStorage.getItem('codigo_emisora')).classList.add("cada_emisora_seleccionada");
[Fin si]
35.- Comprobar si es la primera vez que se accede a esa sección:
localStorage.getItem('info_ppal_logged_visto') != "true"
[Si es la primera vez]
36a.- Se abre nueva interfaz encima de la actual:
UI_logged_info *Tako.Notification.custom(...)*
37a.- Marcar que ya se ha accedido .
localStorage.setItem('info_ppal_logged_visto',"true");
38a.- El usuario pulsa el mensaje para cerrar la vista emergente
[Fin si]
39.- Ver si es favorito: *ver_si_es_favoritos();*
40.- Coger código usuario: *localStorage.getItem('codigo_usuario');*
41.- Coger codigo emisora: *localStorage.getItem('codigo_emisora');*
42.- Comprobar si es favorito: *buscar_favoritos();*

43.- Obtenemos favoritos: *mysqli_query(SQL4)*
*SQL4: ""SELECT * FROM favoritos*
WHERE (favoritos.cod_usuario = '$cod_usuario')
AND (favoritos.cod_emisora = '$cod_emisora')";
44.- *mysqli_result()*
[Mientras haya elementos (emisora favorita del usuario)]
45.- $res[cod_usuario]
[Fin mientras]
46.- Devuelve el resultado obtenido
47.- Realiza cambios en variables internas de la app en función del resultado obtenido.

7 Elegir emisora al pulsar lista manualmente

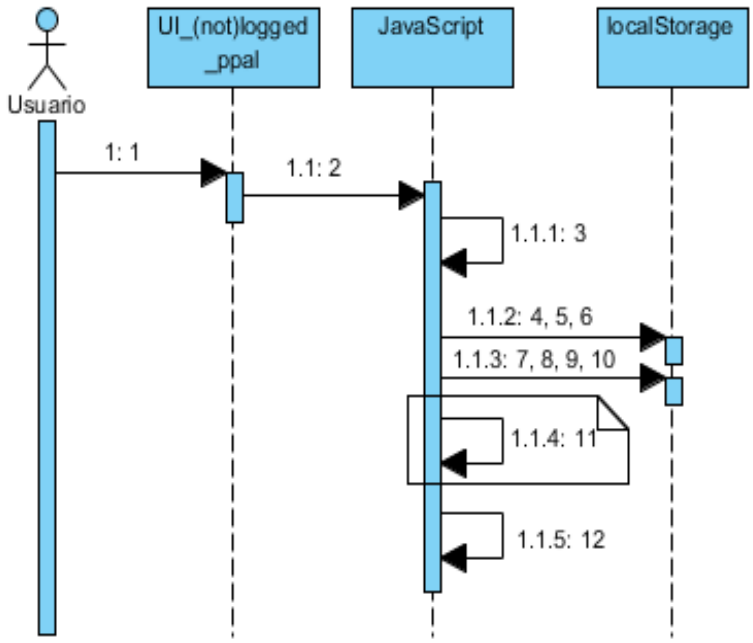


Ilustración 124: Diagrama de secuencia - Elegir emisora al pulsa lista

Precondición: está la app iniciada en la interfaz **UI_notlogged_ppal** o **UI_logged_ppal**.

1.- El usuario pulsa en un elemento del listado de emisoras cercanas.
.$nombre_emisora_utf.";
2.- Se ejecuta el evento: *onclick=""*
3.- Se ejecuta la función: *seleccionar_emision()*
4.- Se actualiza el valor de una variable local:
localStorage.setItem('seleccionado_alguna_vez',true);
5.- Se actualiza el valor de una variable local:
localStorage.setItem('codigo_emision',cod_emision);
6.- Se actualiza el valor de una variable local:
localStorage.setItem('codigo_emisora',cod_emisora);
7.- Se actualiza el valor de una variable local:
localStorage.setItem('last_codigo_emision',cod_emision);
8.- Se actualiza el valor de una variable local:

localStorage.setItem('last_frecuencia_emision',frec_emision);
9.- Se actualiza el valor de una variable local:
localStorage.setItem('last_codigo_emisora',cod_emisora);
10.- Se actualiza el valor de una variable local:
localStorage.setItem('last_nombre_emisora',nombre_emisora);
[De todos los elementos del listado de emisoras]
11.- Se elimina la clase "seleccionada", de todos tanto de logged como notlogged. c*lassList.remove("cada_emisora_seleccionada");*
[Fin de todos los elementos del listado de emisoras]
12.- Se añade la clase "c*ada_emisora_seleccionada*" a la emisora seleccionada.
document.getElementById(cod_emisora).classList.add("cada_emisora_selec cionada");

8 Play / Stop

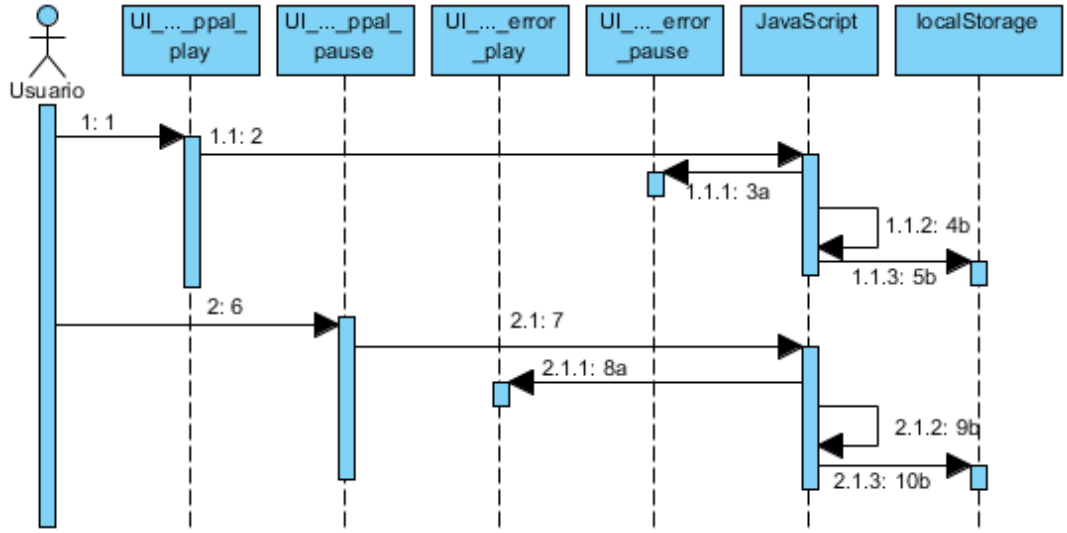


Ilustración 125: Diagrama de secuencia - Play / Pause

Precondición: está la app iniciada en la interfaz **UI_notlogged_ppal** o **UI_logged_ppal,** el funcionamiento de los botones es el mismo, ya esté el usuario registrado como si no lo está. (en los ejemplos de este caso se ejemplificará únicamente con la situación del usuario identificado).

Para el botón play

1.- El usuario pulsa en el botón btn_logged_btn_play

<button class="..." id="btn_notlogged_btn_play"></button>

2.-

document.getElementById('btn_logged_btn_play').addEventListener('click', function(){...});

[Si el elemento 'last_nombre_emisora' de localStorage es igual a null]

3a.- Se ejecuta la llamada a la notificación:

Tako.Notification.error("deny", variable_x variable_y, null, function(){ });

[Si el elemento 'last_nombre_emisora' de localStorage no es igual a null]

4b.- Se activa la recepción de radio FM con la frecuencia que hay en la variable de localStorage '*last_frecuencia_emision'*

enableFMRadio(localStorage.getItem('last_frecuencia_emision'));

5b.- Se modifican las clases correspondientes a los botones de Play y Pause, activando en naranja la del botón Play y volviendo al color azul a la de Pause:

document.getElmetById('btn_logged_btn_pause').classList.remove('orange');
document.getElementById('btn_logged_btn_pause').classList.add('blue');
document.getElementById('btn_logged_btn_play').classList.remove('blue');
document.getElementById('btn_logged_btn_play').classList.add('orange');
document.getElementById('btn_logged_btn_play').classList.add('animated');

Para el botón pause

6.- El usuario pulsa en el botón btn_logged_btn_pause
<button class="..." id="btn_notlogged_btn_pause"></button>
7.-
document.getElementById('btn_logged_btn_pause').addEventListener('click', function(){...});
[Si el elemento 'last_nombre_emisora' de localStorage es igual a null]
8a.- Se ejecuta la llamada a la notificación:
Tako.Notification.error("deny", variable_x variable_y, null, function(){ });
[Si el elemento 'last_nombre_emisora' de localStorage no es igual a null]
9b.- Se desactiva la reproducción de radio FM.
myRadioFM.disable();
10b.- Se modifican las clases correspondientes a los botones de Play y Pause, activando en naranja la del botón Pause y volviendo al color azul a la de Play:
document.getElementById('btn_logged_btn_pause').classList.remove('blue');
document.getElementById('btn_logged_btn_pause').classList.add('orange');
document.getElementById('btn_logged_btn_play').classList.remove('orange');
document.getElementById('btn_logged_btn_play').classList.add('blue');
document.getElmtById('btn_logged_btn_play').classList.remove('animated');

9 RDS (notlogged)

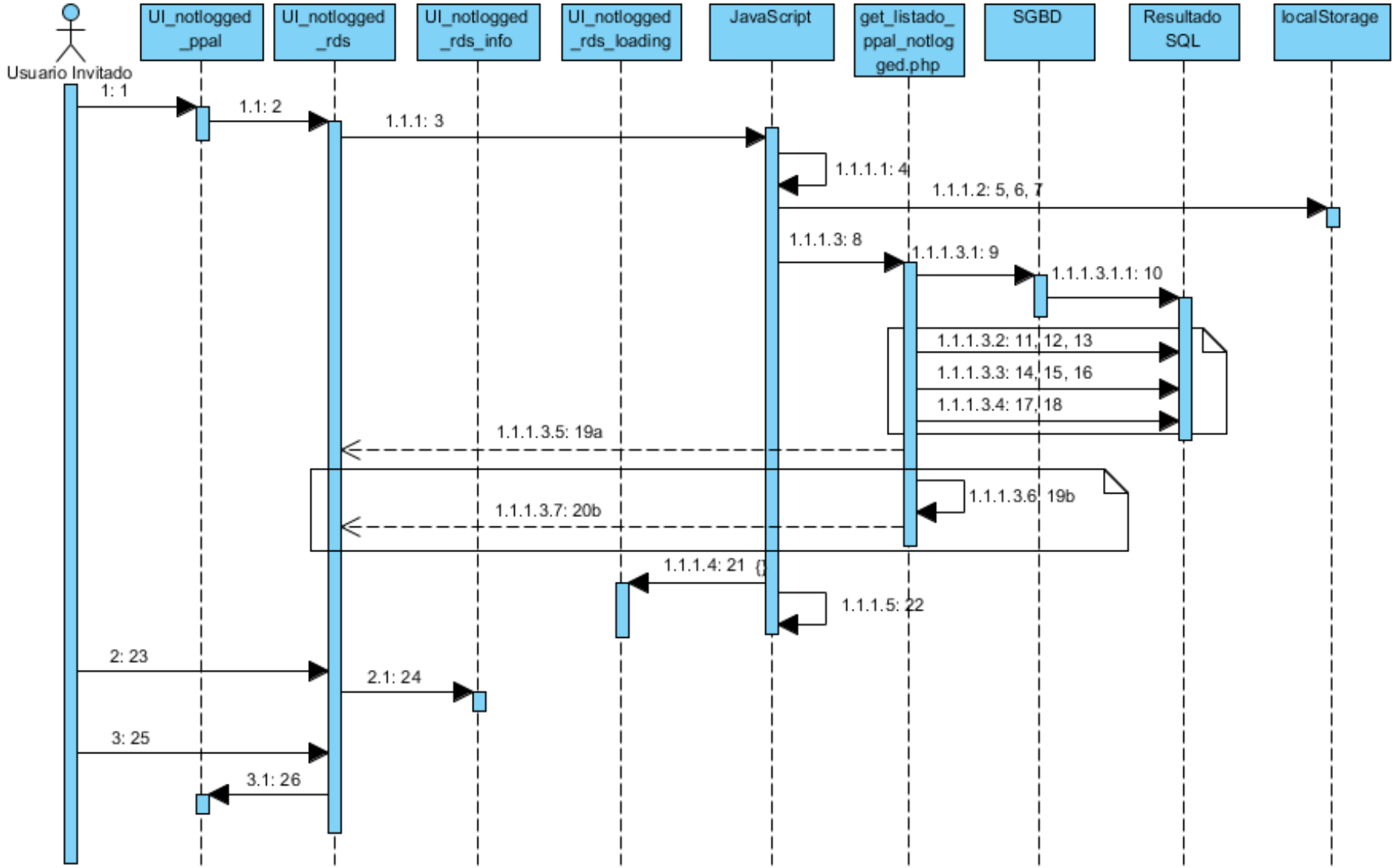


Ilustración 126: Diagrama de secuencia - RDS notlogged

Precondición: está la app iniciada en la interfaz **UI_notlogged_ppal**. El usuario tiene seleccionada alguna emisora del listado, hay una emisora y una frecuencia escritas en la parte superior de esa ventana.
1.- El usuario pulsa en el botón central de la barra de navegación inferior, el botón RDS. *data-section="section_notlogged_rds".*
2.- Se abre la nueva interfaz **UI_notlogged_rds**:
section_notlogged_rds→ classList.add("active")
3.-
document.getElementById("btn_section_notlogged_rds").addEventListener("click",function(){...}); {
4.- Se actualiza el valor de una variable:
document.getElementById("txtBD_rds_notlogged").style.visibility = "hidden";
5.- Obtención datos locales para rds:
localStorage.getItem('codigo_emisora')
6.- Obtención datos locales para rds:
localStorage.getItem('mylatit')
7.- Obtención datos locales para rds:
localStorage.getItem('mylongit')
8.- Obtener elementos (emisiones cercanas de la emisora seleccionada) de la base de datos. *show_content_bd_rds_notlogged()*
9.- *mysqli_query(SQL1)*
SQL1:
"SELECT emisiones.cod_emision, emisiones.frecuencia_emision, emisiones.latitud_emision, emisiones.longitud_emision, emisiones.posicion_emision, emisiones.fecha_hora, emisiones.cod_emisora, emisoras.nombre_emisora
FROM emisiones INNER JOIN emisoras
ON (emisiones.cod_emisora = emisoras.cod_emisora)
WHERE (emisiones.cod_emisora = '$cod_emisora')
AND ('$resu_lat_menos' < emisiones.latitud_emision)
AND (emisiones.latitud_emision < '$resu_lat_mas')
AND ('$resu_long_menos' < emisiones.longitud_emision)
AND (emisiones.longitud_emision < '$resu_long_mas')
ORDER BY emisiones.fecha_hora DESC";
10.- *mysqli_result()*
[Mientras haya elementos]
11.- $cod_emision[$k] = "$res[cod_emision]";

12.- $frecuencia_emision[$k] = "$res[frecuencia_emision]";
13.- $latitud_emision[$k] = "$res[latitud_emision]";
14.- $longitud_emision[$k] = "$res[longitud_emision]";
15.- $posicion_emision[$k] = "$res[posicion_emision]";
16.- $fecha_hora[$k] = "$res[fecha_hora]";
17.- $codi_emisora[$k] = "$res[cod_emisora]";
18.- $nombre_emisora[$k] = "$res[nombre_emisora]";
[Fin mientras]
[Si no hay elementos]
19a.- Al no haber datos, se escribe en la interfaz "No hay emisiones cerca": *echo $txt*
[Si si hay elemento]
[Mientras haya elementos]
19b Recorrer cada elemento obtenido:
20b.- Los datos obtenidos, escribirlos/rellenarlos en la interfaz: *echo $txt*
[Fin mientras]
[Fin si]
[Durante dos segundos]
21.- Se abre nueva interfaz encima de la actual:
UI_mensaje_loading *Tako.Notification.loading(...);*
22.- Cuando se cierra esta interfaz, se actualiza el valor de una variable: *document.getElementById("txtBD_rds_notlogged").style.visibility = "visible";*
[Fin de este mensaje de aviso]
[Si pulsa botón info de la esquina superior derecha]
23.- Pulsa el botón "i" (información de la sección). *document.getElementById('id_btn_notlogged_rds_info').addEventListener ('click', function(){...});*
24.- Se abre la interfaz/mensaje **UI_notlogged_rds_info:** *Tako.Notification.custom(...);*
[Fin si]
[Si pulsa botón play de la esquina superior izquierda]
25.- Pulsa el botón "Radio UI".*data-section="section_notlogged_radio_ui"*
26.- Se abre la interfaz **UI_notlogged_ppal:** *section_notlogged_radio_ui → classList.add("activo")*
[Fin si]

10 RDS (logged)

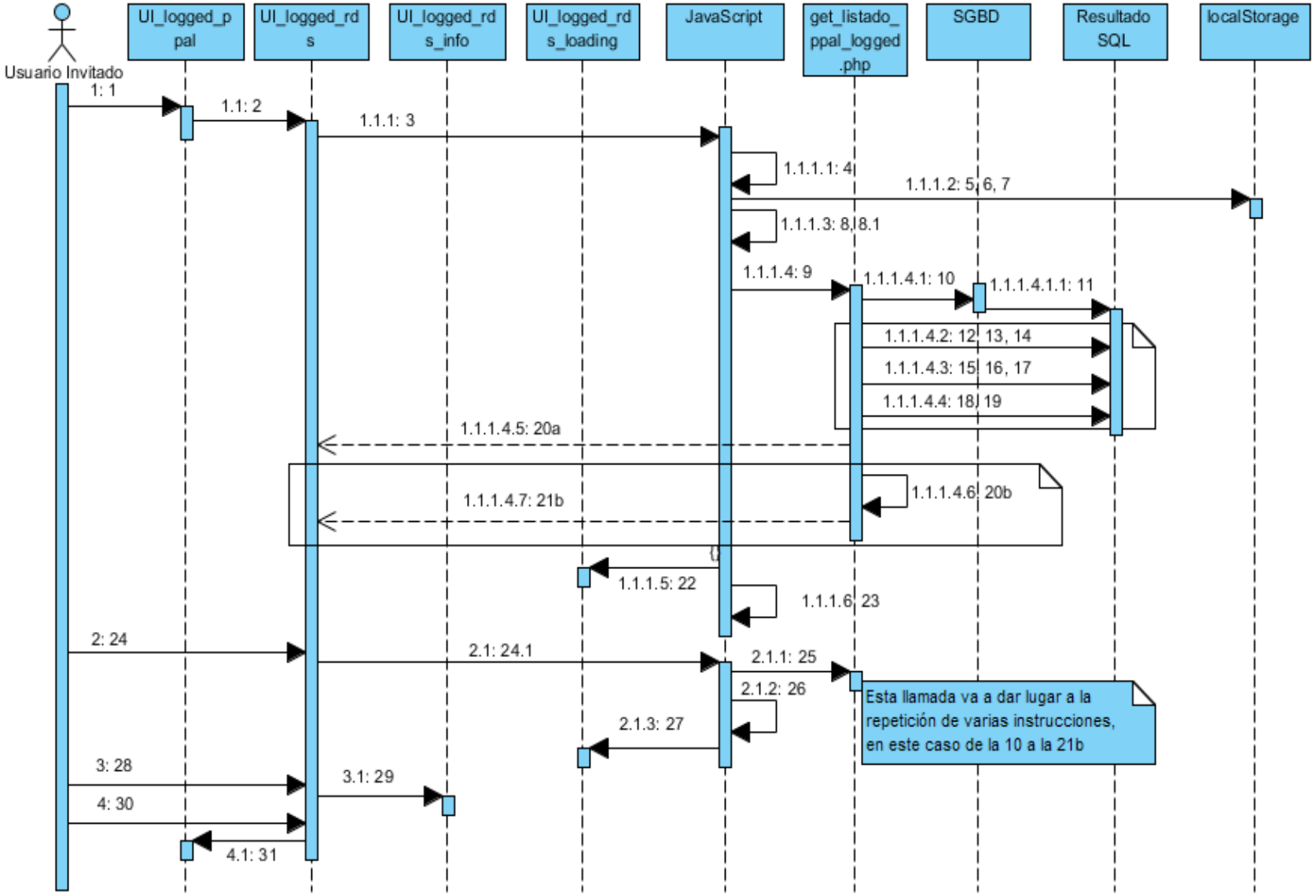


Ilustración 127: Diagrama de secuencia - RDS logged

Precondición: está la app iniciada en la interfaz **UI_logged_ppal**, por lo que el usuario está identificado. El usuario tiene seleccionada alguna emisora del listado, hay una emisora y una frecuencia escritas en la parte superior de esa ventana.

1.- El usuario pulsa en el botón central de la barra de navegación inferior, el botón RDS. *data-section="section_logged_rds".*

2.- Se abre la nueva interfaz **UI_logged_rds**:
section_logged_rds → classList.add("active")

3.-
document.getElementById("btn_section_logged_rds").addEventListener("click",function(){...}); {

4.- Se actualiza el valor de una variable:
document.getElementById("txtBD_rds_logged").style.visibility = "hidden";

5.- Obtención datos locales para rds:
localStorage.getItem('codigo_emisora')

6.- Obtención datos locales para rds:
localStorage.getItem('mylatit')

7.- Obtención datos locales para rds:
localStorage.getItem('mylongit')

8.- Obtención de la precisión que está seleccionada, por defecto "normal_prec". *que_precision();*

8.1.- En función del valor, se obtendrán unos resultados diferentes.
document.getElementById('normal_prec').checked → "normal_prec" ||
document.getElementById('poca_prec').checked → "poca_prec"

9.- Obtener elementos (emisiones cercanas de la emisora seleccionada) de la base de datos. *show_content_bd_rds_logged()*

10.- *mysqli_query(SQL1)*
SQL1:
"SELECT emisiones.cod_emision, emisiones.frecuencia_emision, emisiones.latitud_emision, emisiones.longitud_emision, emisiones.posicion_emision, emisiones.fecha_hora, emisiones.cod_emisora, emisoras.nombre_emisora
FROM emisiones INNER JOIN emisoras
ON (emisiones.cod_emisora = emisoras.cod_emisora)
WHERE (emisiones.cod_emisora = '$cod_emisora')
AND ('$resu_lat_menos' < emisiones.latitud_emision)
AND (emisiones.latitud_emision < '$resu_lat_mas')

AND ('$resu_long_menos' < emisiones.longitud_emision)
AND (emisiones.longitud_emision < '$resu_long_mas')
ORDER BY emisiones.fecha_hora DESC";
11.- *mysqli_result()*
[Mientras haya elementos]
12.- $cod_emision[$k] = "$res[cod_emision]";
13.- $frecuencia_emision[$k] = "$res[frecuencia_emision]";
14.- $latitud_emision[$k] = "$res[latitud_emision]";
15.- $longitud_emision[$k] = "$res[longitud_emision]";
16.- $posicion_emision[$k] = "$res[posicion_emision]";
17.- $fecha_hora[$k] = "$res[fecha_hora]";
18.- $codi_emisora[$k] = "$res[cod_emisora]";
19.- $nombre_emisora[$k] = "$res[nombre_emisora]";
[Fin mientras]
[Si no hay elementos]
20a.- Al no haber datos, se escribe en la interfaz "No hay emisiones cerca": *echo $txt*
[Si si hay elemento]
[Mientras haya elementos]
20b.- Recorrer cada elemento obtenido:
21b.- Los datos obtenidos, escribirlos/rellenarlos en la interfaz: *echo $txt*
[Fin mientras]
[Fin si]
[Durante dos segundos]
22.- Se abre nueva interfaz encima de la actual:
UI_logged_rds_loading *Tako.Notification.loading(...);*
23.- Cuando se cierra esta interfaz, se actualiza el valor de una variable: *document.getElementById("txtBD_rds_notlogged").style.visibility = "visible";*
[Fin de este mensaje de aviso]
[Elige cualquiera de las precisiones (los radiobuttons)]
24.- Pulsa uno de los radiobuttons .
document.getElementById('normal_prec').checked ||
document.getElementById('poca_prec').checked
24.1.-
document.getElementById("poca_prec").addEventListener("click",function() {...}); ||

document.getElementById("normal_prec").addEventListener("click",functio n() {...});

25.- Obtener elementos (emisiones cercanas de la emisora seleccionada) de la base de datos. *show_content_bd_rds_logged().* Se repiten las instrucciones mencionadas, desde la 10 a la 21b.

26.- Mostrar seleccionada la emisión que se está escuchando.

document.getElementById(localStorage.getItem('codigo_emision')).classList. add("cada_emisora_seleccionada_rds");

27.- Se abre la interfaz/mensaje **UI_logged_rds_loading:**

Tako.Notification.custom(...);

[Fin elige precision]

[Pulsa botón info de la esquina superior derecha]

28.- Pulsa el botón "i" (información de la sección).

document.getElementById('id_btn_logged_rds_info').addEventListener ('click', function(){...});

29.- Se abre la interfaz/mensaje **UI_logged_rds_info:**

Tako.Notification.custom(...);

[Fin pulsa botón info]

[Pulsa botón play de la esquina superior izquierda]

30.- Pulsa el botón "Radio UI".*data-section="section_logged_radio_ui"*

31.- Se abre la interfaz **UI_logged_ppal:**

section_logged_radio_ui→ classList.add("activo")

[Fin pulsa botón play]

11 Elegir emisión al pulsar lista manualmente

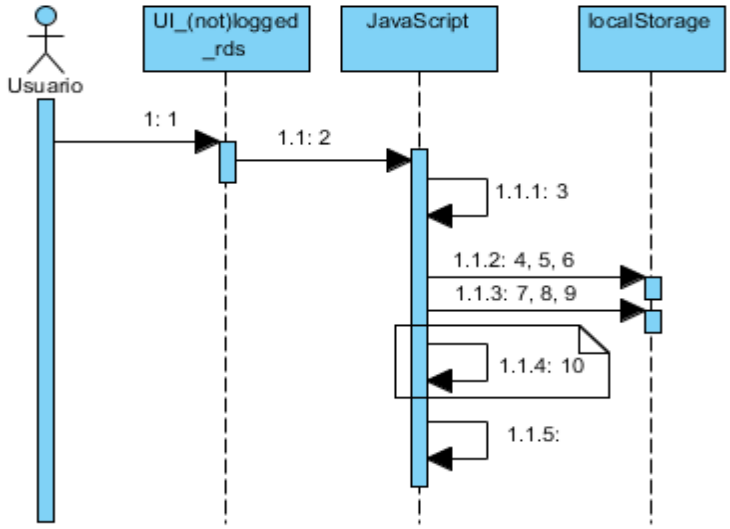


Ilustración 128: Diagrama de secuencia - Seleccionar emisión manualmente

Precondición: está la app iniciada en la interfaz ***UI_notlogged_rds*** *o* ***UI_logged_rds****. El usuario tiene una emisora seleccionada y ve un listado de emisiones, con uno o varios elementos.*

1.- El usuario pulsa en un elemento del listado de emisiones.
<a onclick="seleccionar_rds($cod_emision[$i], $frecuencia_emision[$i], $codi_emisora[$i], $nombre_emisora_utf)">
2.- Se ejecuta el evento: *onclick=””*
3.- Se ejecuta la función: *seleccionar_rds()*
4.- Se actualiza el valor de una variable local:
localStorage.setItem('codigo_emision',cod_emision);
5.- Se actualiza el valor de una variable local:
localStorage.setItem('codigo_emisora',cod_emisora);
6.- Se actualiza el valor de una variable local:
localStorage.setItem('last_codigo_emision',cod_emision);
7.- Se actualiza el valor de una variable local:
localStorage.setItem('last_frecuencia_emision',frec_emision);
8.- Se actualiza el valor de una variable local:
localStorage.setItem('last_codigo_emisora',cod_emisora);

9.- Se actualiza el valor de una variable local:
localStorage.setItem('last_nombre_emisora',nombre_emisora);
[De todos los elementos del listado de emisiones]
10.- Se elimina la clase "seleccionada", de todos tanto de logged como notlogged. c*lassList.remove("cada_emisora_seleccionada_rds");*
[Fin de todos los elementos del listado de emisiones]
11.- Se añade la clase "c*ada_emisora_seleccionada_rds*" a la emisión y emisora seleccionada.
document.getElementById(cod_emision).classList.add("cada_emisora_seleccionada_rds");
document.getElementById(cod_emisora).classList.add("cada_emisora_seleccionada");

12 Puntuar emisión

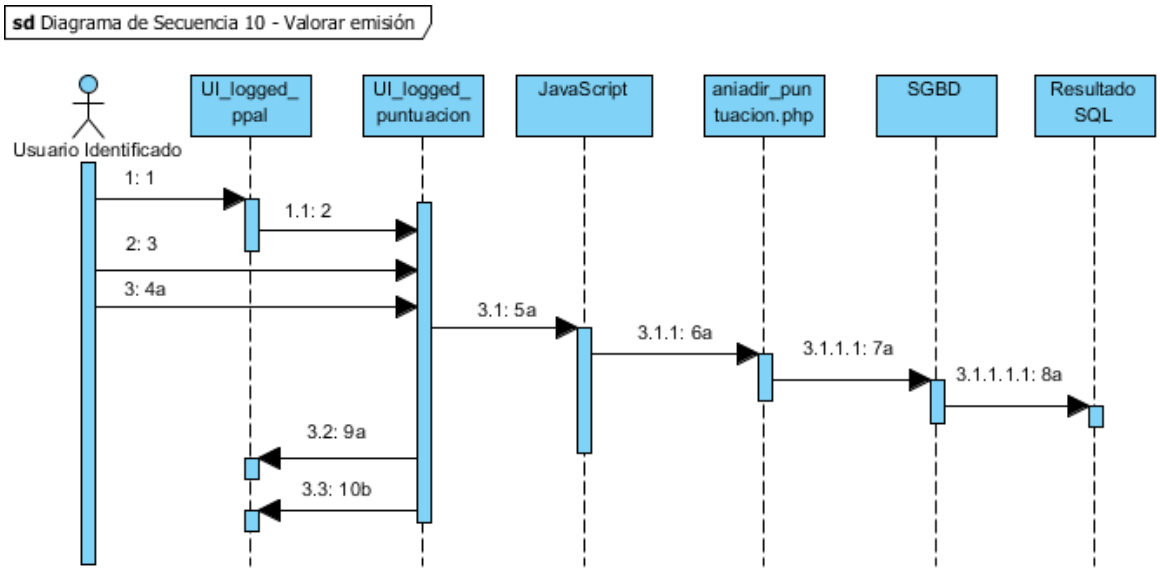


Ilustración 129: Diagrama de secuencia – Puntuar emisión

Precondición: está al app iniciada, el usuario identificado, en la interfaz UI_logged_ppal y con una emisión seleccionada, con su nombre de emisora y frecuencia apareciendo en la vista principal.
1.- El usuario pulsa el icono azul de puntuaciones.
$("#btn_logged_btn_puntuaciones").bind("tap", function(){...});
2.- Se abre la nueva interfaz **UI_logged_puntuacion**:
Tako.Notification.confirm(...);
3.- El usuario mueve el slider. Tendrá el aspecto
UI_logged_puntuacion_resultados (no se incluye en el diagrama).
[Si pulsa en "Puntuar", el icono verde]
4a.- Pulsa en "Puntuar".
5a.- Capta el valor seleccionado
document.getElementById("id_slide_puntuaciones").value
6a.- Registrar la puntuación realizada. *nuevo_aniadir_puntuacion();*
7a.- *mysqli_query(SQL1)*
SQL1:
"INSERT INTO puntuaciones (cod_usuario, cod_emision, latitud_puntual_usuario, longitud_puntual_usuario, fecha_hora,puntos) VALUES ('$cod_usuario', '$cod_emision', '$puntuacion_lat', '$puntuacion_long', '$fecha_hora', '$puntuacion')"

8a.- *mysqli_result()*
9a.- Se vuelve a la interfaz **UI_logged_ppal**.
[Si pulsa "Ahora no"]
10b.- Se vuelve a la interfaz **UI_logged_ppal**.
[Fin si]

13 Añadir emisión manualmente

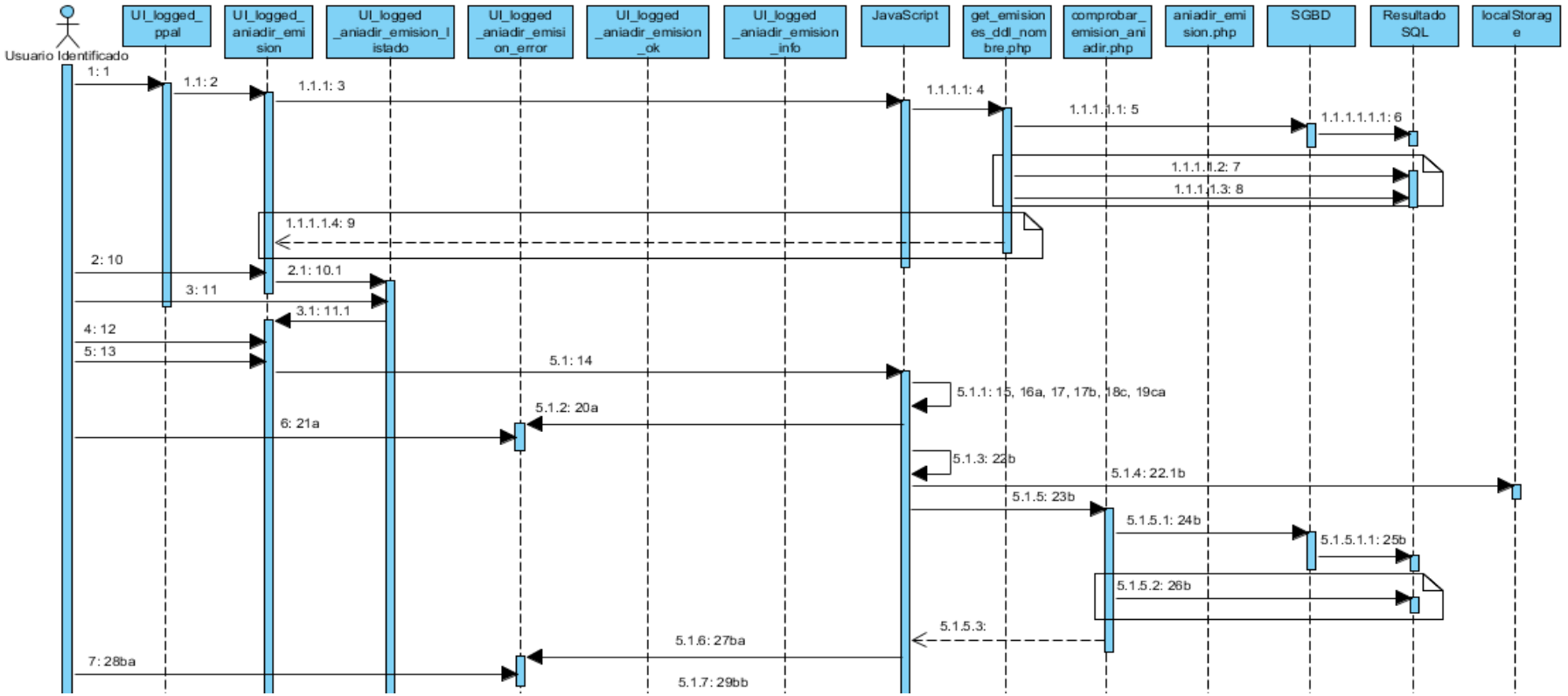


Ilustración 130: Diagrama de secuencia - Añadir emisión manualmente (primera parte)

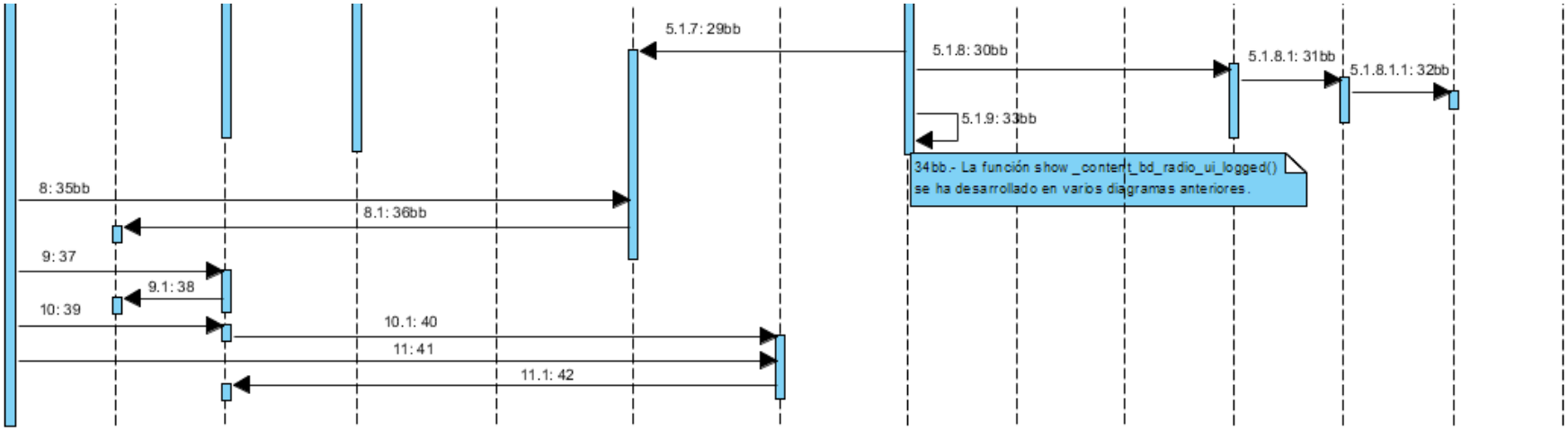


Ilustración 131: Diagrama de secuencia - Añadir emisión manualmente (segunda parte)

Precondición: está la app iniciada en la interfaz **UI_logged_ppal**. *El usuario está identificado. Este, conoce una emisión de una emisora cerca de donde está situado y quiere añadirla al sistema.*

1.- El usuario pulsa en el botón "+" de la barra inferior, situado en la posición izquierda. *data-section="section_logged_aniadir_emision"*
2.- Se abre la nueva interfaz: **UI_logged_aniadir_emision**
section_logged_aniadir_emision → classList.add("current")
3.- Al cargar esa sección:
$("#section_logged_aniadir_emision").on("load", function(){...});
4.- Obtener emisoras del sistema:
show_content_bd_emisiones_nombre_ddl();
5.- *mysqli_query(SQL1)*
SQL1:
"SELECT emisoras.cod_emisora, emisoras.nombre_emisora
FROM emisoras ORDER BY emisoras.nombre_emisora";
6.- *mysqli_result()*
[Mientras haya elementos]
7.- $res[cod_emisora];
8.- $res[nombre_emisora];
[Fin mientras]
[De cada elementos (emisoras) que hay]
9.- Los datos obtenidos, escribirlos/rellenarlos en el desplegable:
echo $txt
[Fin recorrer cada elemento]
10.- Pulsa en el desplegable.
10.1.- Se abre la nueva interfaz: **UI_logged_aniadir_emision_listado**
11.- Selecciona un elemento (la emisora) del desplegable.
document.getElementById("ddl_article_logged_aniaidir_emision_nombre").options[document.getElementById("ddl_article_logged_aniaidir_emision_nombre").selectedIndex].value
11.1.- Se cierra la interfaz: **UI_logged_aniadir_emision_listado**
12.- Rellena el campo frecuencia.
13.- Pulsa en el botón "Añadir emisión".
14.-
document.getElementById("id_article_logged_aniadir_emision_btn_aniadir_emision").addEventListener("click",function() {...});
15.- Comprobar listado nombres de emisoras seleccionado:

ningunDDLSeleccionado(emisora_seleccionada);
[Si no está seleccionado]
16a.-imprimir_errores_ddl_emisoras_no_seleccionada()
[Fin si]
17.- Comprueba si frecuencia es correcta:
validarFreqRegExp(freq)
[Si es vacio]
17b.- *imprimir_errores_frecuencia_vacia()*
[Sino]
18c.- Comprobar si está en el intervalo:
validarFrecuenciaIntervalo(freq)
[Si no está entre 87.0 y 108.0]
19ca.- *imprimir_errores_frecuencia_fuera_intervalo()*
[Fin si]
[Fin si]
[Si hay algún error]
20a.- Se abre nueva interfaz encima de la actual:
UI_logged_aniadir_emision_error *Tako.Notification.error("deny",...)*
21a.- El usuario pulsa para cerrar el mensaje.
[Si no hay error]
22b.- Obtenemos la posición actual del usuario: *geopos();*[9]
22.1b.- Se guardan en almacenamiento local datos obtenidos de la posición.
23b.- Comprobar si la emisión ya existe en una posición cercana:
comprobacion_aniadir_emision_logged_server(function(existe))
24b.- *mysqli_query(SQL2)*
SQL2:
"SELECT cod_emisora, frecuencia_emision, latitud_emision, longitud_emision FROM emisiones
WHERE (emisiones.cod_emisora = '$cod_emisora')
AND (emisiones.frecuencia_emision = '$frecuencia_emision')
AND ('$resu_lat_menos' < emisiones.latitud_emision)
AND (emisiones.latitud_emision < '$resu_lat_mas')
AND ('$resu_long_menos' < emisiones.longitud_emision)
AND (emisiones.longitud_emision < '$resu_long_mas')";
25b.- *mysqli_result()*
[Mientras haya elementos]

9 Desarrollado extensamente en el diagrama de secuencia "Inicio de la app"

26b.- $res[cod_emisora]
[Fin mientras]
{devuelve yahayunaemision o vacio}
[Si devuelve yahayunaemision]
27ba.- Se abre nueva interfaz encima de la actual:
UI_logged_aniadir_emision_error_existe
Tako.Notification.error("deny",...)
28ba.- El usuario pulsa para cerrar el mensaje.
[Sino]
29bb.- Se abre nueva interfaz encima de la actual:
UI_logged_aniadir_emision_ok
Tako.Notification.success("ok",...);
30bb.- Añadir emisión a la BD:
nuevo_aniadir_emision_server();
31bb.- *mysqli_query(SQL3)*
SQL3:
"INSERT INTO emisiones (frecuencia_emision, latitud_emision, longitud_emision, posicion_emision, cod_emisora,fecha_hora,cod_usuario) VALUES ('$frecuencia_emision', '$latitud_emision', '$longitud_emision', '$posicion_emision_utf', '$cod_emisora', '$fecha_hora', '$cod_usuario')";
32bb.- *mysqli_result()*
33bb.- Poner en blanco campos:
nuevo_poner_en_blanco_aniadir_emision();
34bb.- Obtener emisoras cercanas.:
show_content_bd_radio_ui_logged[10]
35bb.- El usuario pulsa en el mensaje.
36bb.- Redirige a la interfaz principal **UI_logged_ppal**.
section_logged_radio_ui → classList.add("current")
[Fin si]
[Fin si]
37.- En **UI_logged_aniadir_emision**, pulsa el botón atrás:
data-section="section_logged_radio_ui"
38.- Se abre la interfaz
UI_logged_radio_ui *section_logged_radio_ui → classList.add("current")*
39.- En **UI_logged_aniadir_emision**, pulsa el botón "i" info:

10 Desarrollado extensamente en el diagrama de secuencia "Entrar como Identificado"

document.getElementById('id_btn_logged_add_station_info').addEventListener('click', function(){...});

40.- Se abre la interfaz

UI_logged_aniadir_emision_info *Tako.Notification.custom(...);*

41.- El usuario pulsa para cerrar.

42.- Redirige a **UI_logged_aniadir_emision**

14 Añadir emisora manualmente

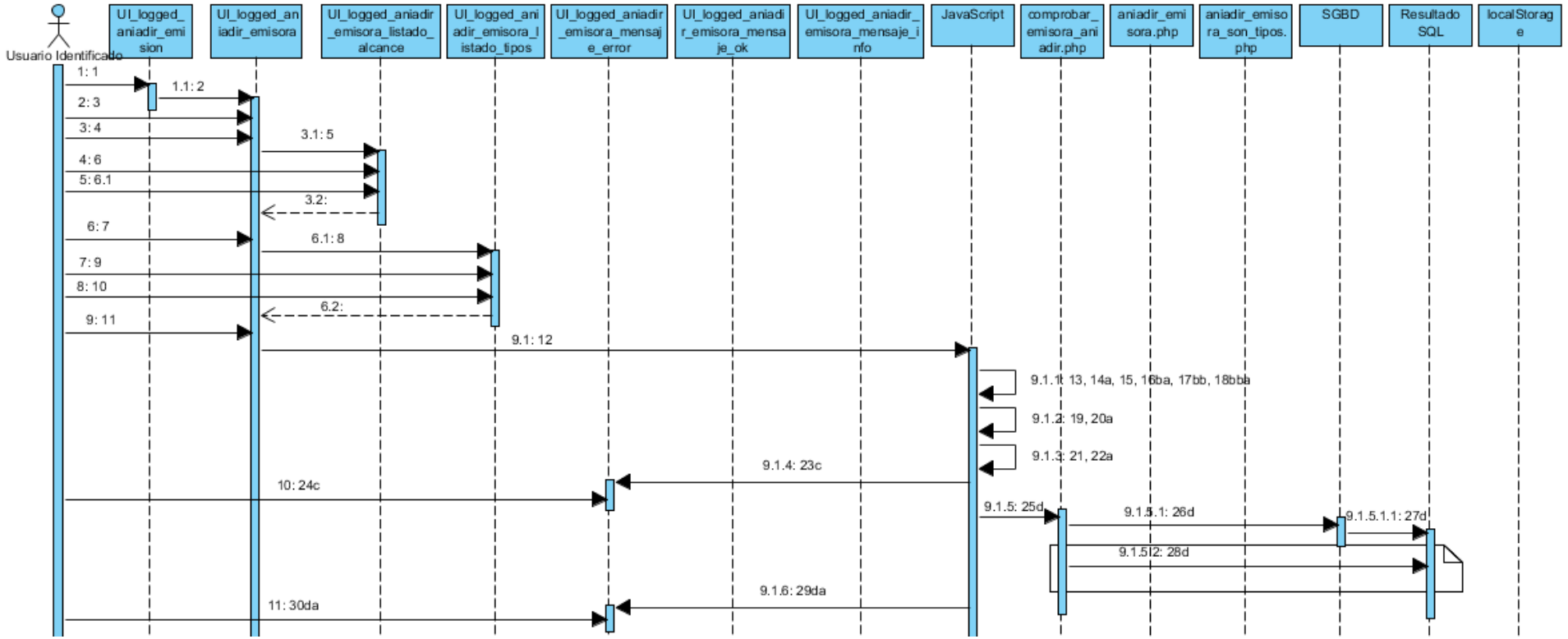


Ilustración 132: Diagrama de secuencia - Añadir emisora manualmente (primera parte)

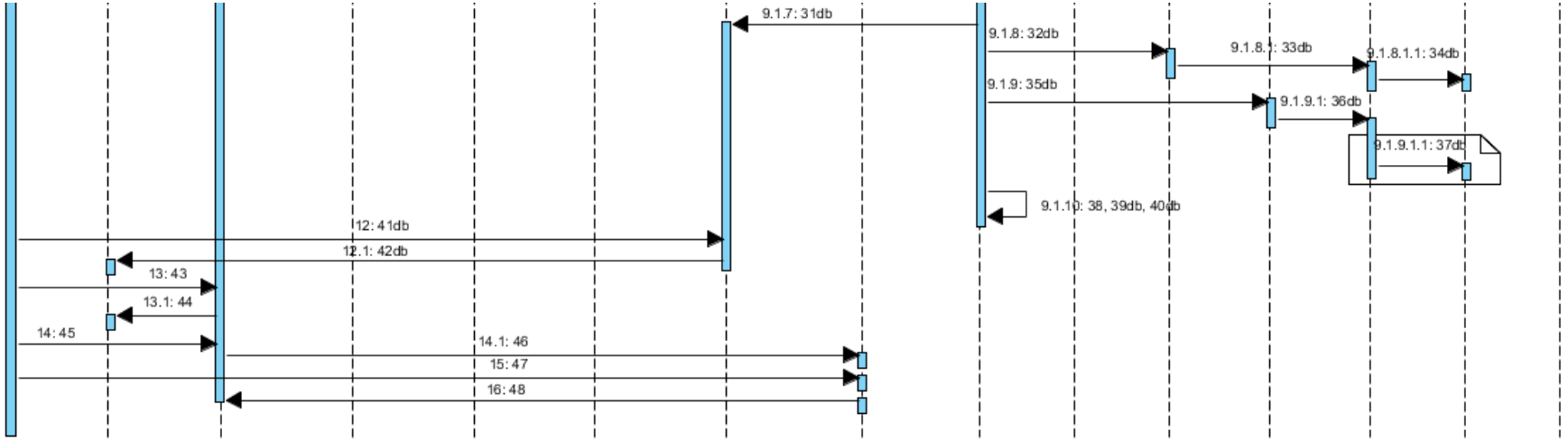


Ilustración 133: Diagrama de secuencia - Añadir emisora manualmente (segunda parte)

Precondición: está la app iniciada en la interfaz **UI_logged_aniadir_emision**. El usuario está identificado. Este, conoce una emisora que no está añadida al sistema.

1.- El usuario pulsa en el botón azul "Añadir nueva emisora", bajo el desplegable de nombres de emisoras.
data-article="article_logged_aniadir_emisora_nueva"
2.- Se abre la nueva interfaz: **UI_logged_aniadir_emisora**
section_logged_aniadir_emisora → classList.add("current")
3.- El usuario rellena el nombre de la emisora.
4.- El usuario pulsa en el desplegable de alcances.
5.- Se abre la nueva interfaz: **UI_logged_aniadir_emisora_listado_alcance**
6.- Selecciona un elemento (el alcance) del desplegable.
$("#ddl_article_logged_aniaidir_emisora_nueva_alcance option:selected")
6.1.- Pulsa en "Aceptar" y se cierra la interfaz:
UI_logged_aniadir_emisora_listado_alcance
7.- El usuario pulsa en el desplegable de tipos.
8.- Se abre la nueva interfaz: **UI_logged_aniadir_emisora_listado_tipos**
9.- Selecciona uno o varios elementos (tipos) del desplegable.
document.getElementById('ddl_article_logged_aniaidir_emisora_nueva_tipos');
10.- Pulsa en "Aceptar" y se cierra la interfaz:
UI_logged_aniadir_emisora_listado_tipos
11.- Pulsa en el botón "Añadir nombre emisora".
12.-
document.getElementById("id_article_logged_aniadir_emisora_btn").addEventListener("click",function() {...});
13.- Comprobar nombre emisora: *validarNombreRegExp(nombre)*
[Si es vacio]
14a.- imprimir_errores_nombre_emisora_vacia*()*
[Sino]
15.- Comprobar tamaño mínimo
*validarTextoTamanioMinimoEmisora(nom*bre*)*:
    [Si es pequeño]
    16ba.- *imprimir_errores_nombre_emisora_pequenia()*
    [Sino]
    17bb.- Comprobar tamaño máximo
    validarTextoTamanioMaximoEmisora(*nombre*)
        [Si es demasiado grande]

18bba.- *imprimir_errores_nombre_emisora_grande();*
[Fin si]
[Fin si]
[Fin si]
19.- Comprobar listado alcance emisoras seleccionado:
ningunDDLSeleccionado(alcance);
[Si no está seleccionado]
20a.- *imprimir_errores_ddl_alcance_no_seleccionada()*
[Fin si]
21.- Comprobar listado tipos emisoras seleccionado
document.getElementById('ddl_article_logged_aniaidir_emisora_nueva_tipos').options[i].selected*;*
[Si no hay ninguno seleccionado]
22a.- *imprimir_errores_ddl_tipos_no_seleccionada()*
[Fin si]
[Si hay algún error]
23c.- Se abre nueva interfaz encima de la actual:
UI_logged_aniadir_emisora_error *Tako.Notification.error("deny",...)*
24c.- El usuario pulsa para cerrar el mensaje.
[Si no hay error]
25d.- Comprobar si el nombre ya existe:
comprobacion_aniadir_emisora_logged_server(function(existe))
26d.- *mysqli_query(SQL1)*
SQL1:
"SELECT nombre_emisora FROM emisoras
WHERE (emisoras.nombre_emisora = '$name_emisora_utf')";
27d.- *mysqli_result()*
[Mientras haya elementos]
28d.- $res[nombre_emisora]
[Fin mientras]
[Si se ha obtenido algún elemento, es que se repite el nombre]
29da.- Se abre nueva interfaz encima de la actual:
UI_logged_aniadir_emisora_error *Tako.Notification.error("deny",...)*
30da.- El usuario pulsa para cerrar el mensaje.
[Sino existe]
31db.- Se abre nueva interfaz encima de la actual:
UI_logged_aniadir_emision_ok *Tako.Notification.success("ok",...);*
32db.- Añadir emisora a la BD: *nuevo_aniadir_emisora_server();*

33db.- *mysqli_query(SQL2)*
SQL2:
"INSERT INTO emisiones (frecuencia_emision, latitud_emision, longitud_emision, posicion_emision, cod_emisora, fecha_hora,cod_usuario) VALUES ('$frecuencia_emision', '$latitud_emision','$longitud_emision', '$posicion_emision_utf', '$cod_emisora', '$fecha_hora', '$cod_usuario')";
34db.- *mysqli_result()*
35db.- Añadir los tipos de la emisora añadida a la BD:
nuevo_aniadir_emisora_tipos_server();
36db.- *mysqli_query(SQL3)*
SQL3: "INSERT INTO emisoras_son_tipos (cod_emisora, cod_emisora_tipo) VALUES ('$cod_emisora[$k]','$tipo_emisora')";
37db.- *mysqli_result()*
38db.- Poner en blanco campos:
nuevo_poner_en_blanco_emisora_aniadir();
39db.-
deseleccionar_ddl(document.getElementById("ddl_article_logged_aniaidir_emisora_nueva_alcance"));
40db.-
deseleccionar_ddl(document.getElementById("ddl_article_logged_aniaidir_emisora_nueva_tipos"));
41db.- El usuario pulsa en el mensaje.
42db.- Redirige a la interfaz principal **UI_logged_aniadir_emision**.
section_logged_aniadir_emision → classList.add("current")

[Fin si]

43.- En **UI_logged_aniadir_emisora**, pulsa el botón atrás:
data-section="section_logged_aniadir_emision"
44.- Se abre la interfaz **UI_logged_aniadir_emision**
section_logged_aniadir_emision → classList.add("current")
45.- En **UI_logged_aniadir_emisora**, pulsa el botón "i" info:
document.getElementById('id_btn_logged_add_station_info').addEventListener('click', function(){...});
46.- Se abre la interfaz **UI_logged_aniadir_emisora_info**
Tako.Notification.custom(...);
47.- El usuario pulsa para cerrar.
48.- Redirige a **UI_logged_aniadir_emisora**

15 Poner favorita emisora

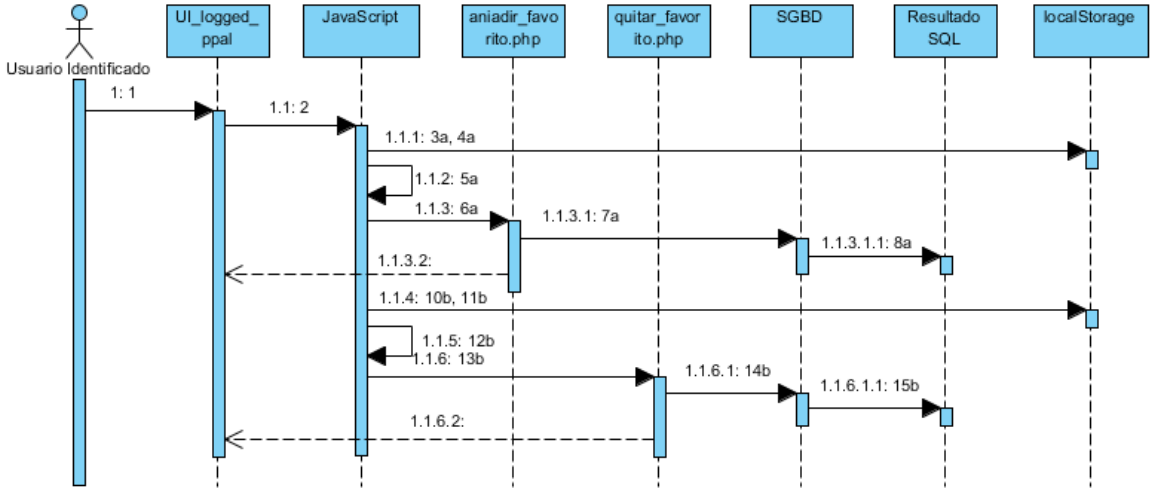


Ilustración 134: Diagrama de secuencia - Poner favorita emisora

Precondición: está al app iniciada, el usuario identificado, en la interfaz UI_logged_ppal y con una emisión seleccionada, con su nombre de emisora y frecuencia apareciendo en la vista principal.
1.- El usuario pulsa el icono azul de favoritos, icono de la estrella.
$("#btn_logged_btn_favoritos").bind("tap", function(){...});
2.- Comprobar si ya es favorito o no, para añadirlo o no.
document.getElementById('btn_logged_btn_favoritos').classList.contains('star-empty')){...}; o
document.getElementById('btn_logged_btn_favoritos').classList.contains('star')){...};
[Si no es favorito]
3a.- Consultar valor almacenamiento local.
localStorage.getItem('codigo_usuario');
4a.- Consultar valor almacenamiento local.
localStorage.getItem('codigo_emisora');
5a.- Se elimina clase "start-empty" y se añade la clase "start".
6a.- Añadir favorito *aniadir_favorito();*
7a.- *mysqli_query(SQL1)*
SQL1:
"INSERT INTO favoritos (cod_usuario,cod_emisora,fecha_hora) VALUES ('$cod_usuario','$cod_emisora','$fecha_hora')"

8a.- *mysqli_result()*
9a.- Se actualiza la interfaz **UI_logged_ppal**, la estrella estará rellena.
[Si es favorito]
10b.- Consultar valor almacenamiento local.
localStorage.getItem('codigo_usuario');
11b.- Consultar valor almacenamiento local.
localStorage.getItem('codigo_emisora');
12b.- Se elimina clase "start" y se añade la clase "start-empty".
13b.- Eliminar favorito *eliminar_favorito();*
14b.- *mysqli_query(SQL2)*
SQL2:
"DELETE FROM favoritos WHERE (cod_usuario = '$cod_usuario') AND (cod_emisora = '$cod_emisora')"
15b.- *mysqli_result()*
16b.- Se actualiza la interfaz **UI_logged_ppal**, la estrella estará vacia.
[Fin si]

16 Realizar escaneo

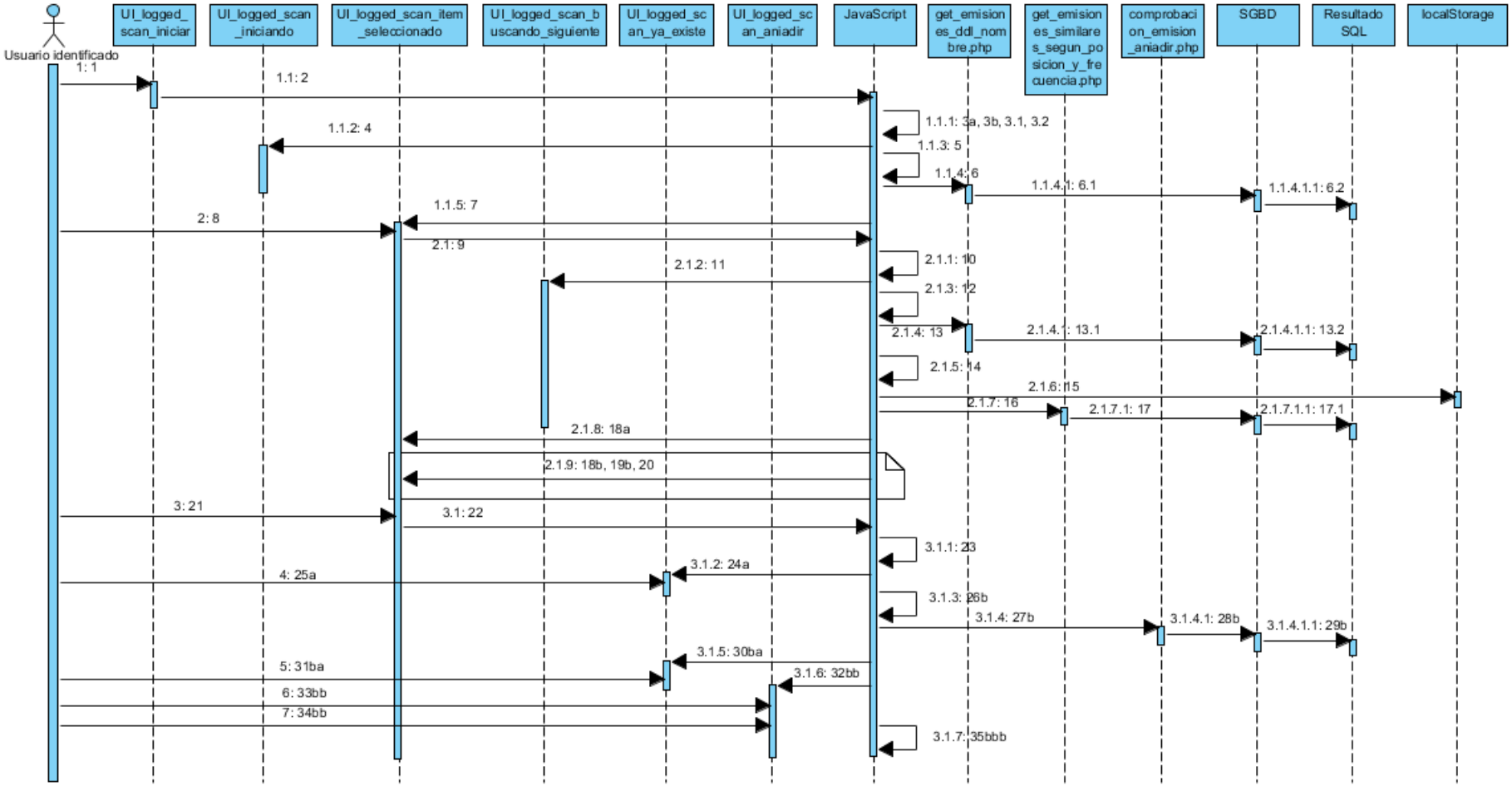


Ilustración 135: Diagrama de secuencia - Hacer escaneo

Precondición: está al app iniciada, el usuario identificado, en la interfaz UI_logged_scan.
1.- El usuario pulsa el botón azul “Iniciar escaneo”.
document.getElementById("btn_logged_btn_scan_inicializar").addEventListener("click",function(){..}
2.- Comprobar si ya hay una frecuencia sonando o está a null.
myRadioFM.frequency == null
[Si está vacía, si no hay ninguna frecuencia sonando]
3a.- Se activa la reproducción de la primera frecuencia posible.
enableFMRadio(87.5);
[Si no está vacia, si ya hubiera una frecuencia sonando, se resetea]
3b.- Se pone la primera frecuencia del dial posible.
myRadioFM.setFrequency(87.5);
3.1.- Se muestra los datos correspondientes a esa frecuencia.
mostrar_datos_radio_fm();
3.2.- Internamente, se llama a otra función para mostrar esos datos pasados 4 segundos, el tiempo que tarda el chip de radio en cambiar la frecuencia en el objeto creado.
mostrar_datos();
4.- Se inicia la nueva interfaz, el aviso de iniciando la frecuencia inicial.
Tako.Notification.loading(iniciando, 4, function(){...}
5.1.- Se producen multitud de cambios en la interfaz, mostrando y ocultando elementos.
id_scan_emision_nueva_titulo_frecuencia.style.display = 'block';
document.getElementById("txtBD_scan_nueva_frecuencia").innerHTML = imprimir_scan_frecuencia() + myRadioFM.frequency;
btn_logged_scan_save.style.display = '';
btn_logged_btn_scan_inicializar.style.display = 'none';
btn_logged_btn_scan.style.display = '';
scan_emisora_nueva_borde.style.display = 'block';
6, 6.1, 6.2.- Se obtiene el listado de emisoras cercanas para el desplegable.
show_content_bd_emisiones_nombre_ddl_scan();
(consiste en repetir lo mismo que en 5.- mysqli_query(SQL1)){...}
7.- Acabadas todas las operaciones internas de la interfaz iniciando, se redirecciona a la interfaz inicial pero con más elementos visibles, que se ha llamado UI_logged_scan_item_seleccionado.
8.- El usuario pulsa en el botón “Buscar siguiente”
9.-

document.getElementById("btn_logged_btn_scan").addEventListener("click",function(){...}
10.- Se ejecutan funciones internas:
myRadioFM.seekUp();
mostrar_datos_radio_fm();
11.- Se abre la nueva interfaz de buscando emisión
var progress = Tako.Notification.progress("search", buscando, buscando_texto, 4, function(){...}
12.- Se producen multitud de cambios en la interfaz, mostrando y ocultando elementos, como se indicaba en el punto 5.1:
id_scan_emision_nueva_titulo_frecuencia.style.display = 'block';
document.getElementById("txtBD_scan_nueva_frecuencia").innerHTML = imprimir_scan_frecuencia() + myRadioFM.frequency;
btn_logged_scan_save.style.display = '';
id_scan_emision_nueva_titulo_encontradas.style.display = 'block';
document.getElementById("txtBD_usuario_logged_scan_emisiones_parecidas").innerHTML = "";
scan_emisora_coincidente_borde.style.display = 'block';
13, 13.1, 13.2.- Se obtiene el listado de emisoras cercanas para el desplegable.
show_content_bd_emisiones_nombre_ddl_scan();
(consiste en repetir lo mismo que en 5.- mysqli_query(SQL1)){...}
14.- Se obtienen las emisiones similares a la que se acaba de encontrar, comprobando los valores de la posición y de la frecuencia.
listar_emisiones_similares_misma_posicion_misma_frecuencia();
15.- Consultar valor almacenamiento local.
localStorage.getItem('mylatit');
localStorage.getItem('mylongit');
16.- *mysqli_query(SQL1)*
SQL1:
"SELECT emisiones.cod_emision, emisiones.frecuencia_emision, emisiones.latitud_emision, emisiones.longitud_emision, emisiones.fecha_hora, emisoras.nombre_emisora, emisiones.activo_emision FROM emisiones
INNER JOIN emisoras ON
(emisiones.cod_emisora = emisoras.cod_emisora)
WHERE (emisiones.frecuencia_emision = '$freq')
AND ('$resu_lat_menos' < emisiones.latitud_emision)

AND (emisiones.latitud_emision < '$resu_lat_mas')
AND ('$resu_long_menos' < emisiones.longitud_emision)
AND (emisiones.longitud_emision < '$resu_long_mas')
AND (emisiones.activo_emision = 'true')";
17.- *mysqli_result()*
[Mientras haya elementos]
17.1.- Se obtienen los datos de las emisiones
$cod_emision[$k] = "$res[cod_emision]";
$frecuencia_emision[$k] = "$res[frecuencia_emision]";
$latitud_emision[$k] = "$res[latitud_emision]";
$longitud_emision[$k] = "$res[longitud_emision]";
$nombre_emisora[$k] = "$res[nombre_emisora]";
$fecha_hora[$k] = "$res[fecha_hora]";
[Fin mientras]
[Si no hay elementos]
18a.- Al no haber datos, se escribe en la interfaz "No hay ninguna coincidencia en el sistema en esa posición". Se presentará traducido al idioma que se esté usando.
[Si si hay elemento]
[Mientras haya elementos]
18b.- Recorrer cada elemento obtenido:
19b.- Los datos obtenidos, escribirlos/rellenarlos en la interfaz:
echo "<li>";
echo " ";
echo "Nombre emisora: $nombre_emisora_utf";
*echo "
Frecuencia: [$frecuencia_emision[$i]]";*
echo " ";
*echo "
";*
Se presentará traducido al idioma que se esté usando.
[Fin mientras]
[Fin si]
20.- Se actualiza la interfaz UI_logged_scan_buscando_siguiente.
var valor_random_1 = numero_aleatorio_entre(10,30);
var valor_random_2 = numero_aleatorio_entre(45,65);
var valor_random_3 = numero_aleatorio_entre(85,95);
var valor_random_4 = "100";
setTimeout(function(){progress.percent(valor_random_1)},500);
setTimeout(function(){progress.percent(valor_random_2)},1500);

setTimeout(function(){progress.percent(valor_random_3)},2500);
setTimeout(function(){progress.percent(valor_random_4)},3500);

21.- El usuario pulsa el botón azul "Iniciar escaneo".
22.-
document.getElementById('btn_logged_scan_save').addEventListener('click', function(){...}
23.- Se ejecutan varias funciones para realizar las comprobaciones de si la frecuencia es null o si hay algún elemento del listado seleccionado.
[Si hay algún error]
24a.- Se abre nueva interfaz encima de la actual:
Tako.Notification.error("deny",...)
25a.- El usuario pulsa para cerrar el mensaje.
[Si no hay errores]
26b.- Se actualiza la posición.
geoPos()
27b.- Se comprueba si esa emisión, con esa frecuencia y ese nombre de emisora, ya existe en el sistema.
comprobacion_aniadir_emision_logged_server_scan()
28b.- *mysqli_query(SQL2)*
SQL2:
"SELECT cod_emisora, frecuencia_emision, latitud_emision, longitud_emision FROM emisiones
WHERE (emisiones.cod_emisora = '$cod_emisora')
AND (emisiones.frecuencia_emision = '$frecuencia_emision')
AND ('$resu_lat_menos' < emisiones.latitud_emision)
AND (emisiones.latitud_emision < '$resu_lat_mas')
AND ('$resu_long_menos' < emisiones.longitud_emision)
AND (emisiones.longitud_emision < '$resu_long_mas')";
29b.- *mysqli_result() {...}*
[Si el método anterior devuelve "yahayunaemision"]
30ba.- Se abre nueva interfaz encima de la actual:
UI_logged_scan_ya_existe *Tako.Notification.error("deny",...)*
31ba.- El usuario pulsa para cerrar el mensaje.
[Si no]
32bb.- Se abre una nueva interfaz encima de la actual
UI_logged_scan_aniadir
Tako.Notification.confirm("bullseye", scan_add_text1,

scan_add_text2,scan_add_text3,scan_add_text4,function(){...});
[Si el usuario pulsa en cancelar]
33bba.- El usuario pulsa en cancelar.
//No ocurre nada
[Si el usuario pulsa en añadir]
34bba.- El usuario pulsa en cancelar.
35bbb.- Añade la nueva emisión en el sistema
nuevo_aniadir_emision_server_scan();
Añadir emisión manualmente
[Fin si]
[Fin si]
[Fin si]

17 Ver información completa de la emisora

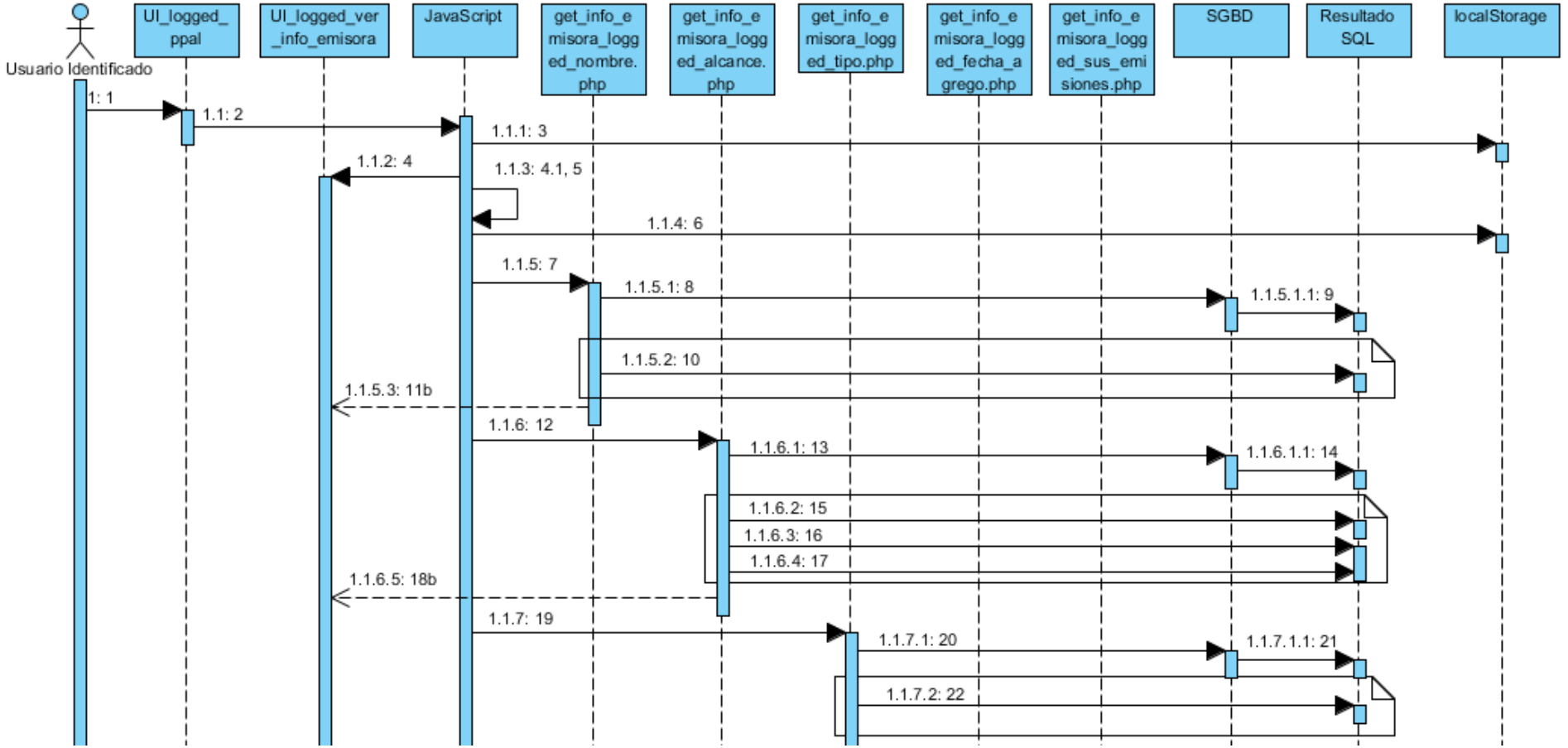


Ilustración 136: Diagrama de secuencia - Ver información completa de la emisora (primera parte)

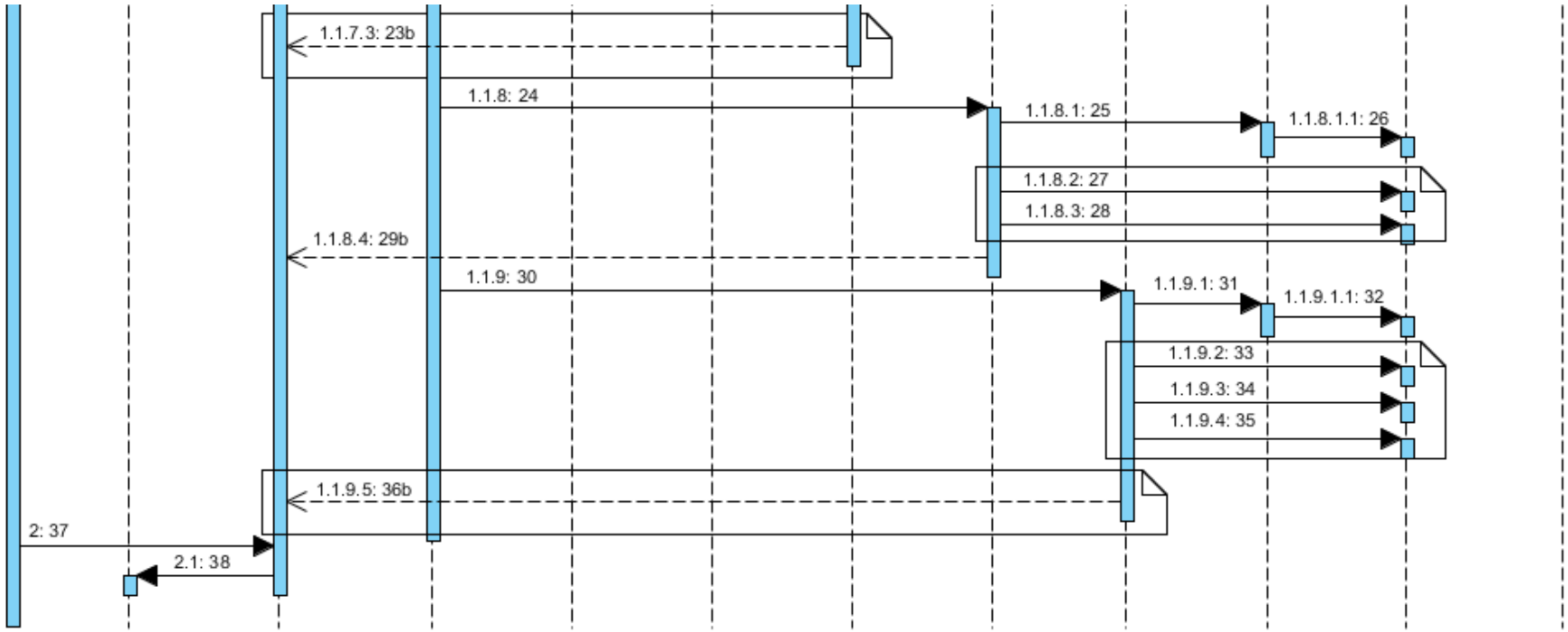


Ilustración 137: Diagrama de secuencia - Ver información completa de la emisora (segunda parte)

<u>Precondición</u>: está la app iniciada, el usuario identificado, en la interfaz UI_logged_ppal y con una emisión seleccionada, con su nombre de emisora y frecuencia apareciendo en la vista principal.

1.- El usuario pulsa en el icono con la forma del logo de la app, junto al nombre de la emisora del listado.
<a onclick="seleccionar_emision_icono($b_codi_emisora[$i2])">
2.- Se ejecuta la función: *seleccionar_emision_icono()*
3.- Asignar valor almacenamiento local
localStorage.setItem('codigo_emisora_seleccionada',cod_emisora);
4.- Se abre la nueva interfaz **UI_logged_info_emisora**:
section_logged_info_emisora → classList.add("current")
4.1.- Al cargar la sección
$("#section_logged_info_emisora").on("load", function(){...});
5.- Ver que idioma tiene el sistema *idioma_navegador();*
codigo_emisora_selec=
6.- Ver código de la emisora seleccionada
localStorage.getItem('codigo_emisora_seleccionada');
7.- Cargar nombre *cargar_info_emisora_logged_nombre();*
8.- *mysqli_query(SQL1)*
SQL1:
"SELECT emisoras.nombre_emisora FROM emisoras
WHERE (emisoras.cod_emisora = '$cod_emisora_seleccionada')";
9.- *mysqli_result()*
[Mientras haya elementos]
10.- $res[nombre_emisora]
[Fin mientras]
[Si no hay elementos (nombre emisoras)]
//11a.- Nada
[Si si hay elemento]
11b.- Los datos obtenidos, escribirlos/rellenarlos en la interfaz:
echo $txt
[Fin si]
12.- Cargar alcance *cargar_info_emisora_logged_alcance();*
13.- *mysqli_query(SQL2)*
SQL2
"SELECT emisoras.nombre_emisora,
alcances_tipos.nombre_alcance_tipo, emisoras.fecha_hora

FROM emisoras
INNER JOIN alcances_tipos
ON (emisoras.alcance_emisora alcances_tipos.cod_alcance_tipo)
WHERE (emisoras.cod_emisora '$cod_emisora_seleccionada')";
14.- *mysqli_result()*
[Mientras haya elementos]
15.- $res[nombre_emisora]
16.- $res[nombre_alcance_tipo]
17.- $res[fecha_hora]
[Fin mientras]
[Si no hay elementos (nombre emisoras)]
//18a.- Nada
[Si si hay elemento]
18b.- Los datos obtenidos, escribirlos/rellenarlos en la interfaz:
echo $txt
[Fin si]
19.- Cargar tipos de emisora *cargar_info_emisora_logged_tipo();*
20.- *mysqli_query(SQL3)*
SQL3
"SELECT emisoras_son_tipos.cod_emisora,
emisoras_tipos.nombre_emisora_tipo as nombre_emisora_tipo
FROM emisoras_son_tipos
INNER JOIN emisoras_tipos ON
(emisoras_son_tipos.cod_emisora_tipo = emisoras_tipos.cod_emisora_tipo)
WHERE
(emisoras_son_tipos.cod_emisora = '$cod_emisora_seleccionada')
ORDER BY nombre_emisora_tipo";
//La consulta se realizará en función del idioma del sistema. Variará el
//atributo nombre_emisora_tipo/ nombre_emisora_tipo_eu/
//nombre_emisora_tipo_en
21.- *mysqli_result()*
[Mientras haya elementos]
22.- $res[nombre_emisora_tipo]
[Fin mientras]
[Si no hay elementos (tipos)]
//23a.- Nada
[Si si hay elementos]
[Mientras haya elementos]

23b.- Los datos obtenidos, escribirlos/rellenarlos en la interfaz:
echo $txt
[Fin mientras]
[Fin si]
24.- Cargar fecha que se añade
cargar_info_emisora_logged_fecha_agrego();
25.- *mysqli_query(SQL4)*
SQL4
"SELECT emisoras.nombre_emisora, emisoras.fecha_hora
FROM emisoras
WHERE (emisoras.cod_emisora = '$cod_emisora_seleccionada')";
26.- *mysqli_result()*
[Mientras haya elementos]
27.- $res[nombre_emisora]
28.- $res[fecha_hora]
[Fin mientras]
[Si no hay elementos (fecha)]
//29a.- Nada
[Si si hay elemento]
29b.- Los datos obtenidos, escribirlos/rellenarlos en la interfaz:
echo $txt
[Fin si]
30.- Cargar emisiones de la emisora
cargar_info_emisora_logged_sus_emisiones();
31.- *mysqli_query(SQL5)*
SQL5
"SELECT emisiones.frecuencia_emision, emisiones.posicion_emision,
emisiones.fecha_hora FROM emisiones
WHERE (emisiones.cod_emisora = '$cod_emisora_seleccionada')";
32.- *mysqli_result()*
[Mientras haya elementos]
33.- $res[frecuencia_emision]
34.- $res[posicion_emision]
35.- $res[fecha_hora]
[Fin mientras]
[Si no hay elementos (emisiones)]
//36a.- Nada
[Si si hay elementos]

[Mientras haya elementos]
36b.- Los datos obtenidos, escribirlos/rellenarlos en la interfaz:
echo $txt
[Fin mientras]
[Fin si]
37.- En **UI_logged_ver_info_emisora**, pulsa el botón atrás:
data-section="section_logged_info_emisora"
38.- Se abre la interfaz **UI_logged_radio_ui**
section_logged_radio_ui → classList.add("current")

18 Gestionar perfil

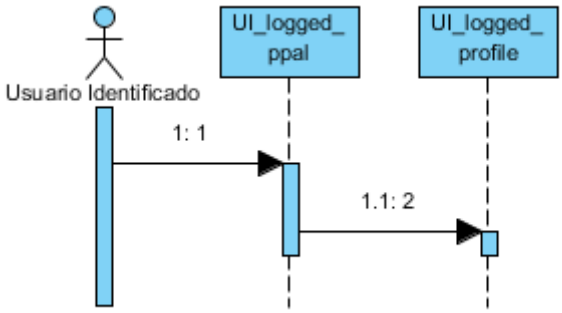


Ilustración 138: Diagrama de secuencia - Gestionar perfil

Precondición: está la app iniciada en la interfaz ***UI_logged_ppal****. El usuario está identificado.*

1.- El usuario pulsa en el icono "perfil" de la barra superior, situado en la esquina derecha. *data-section="section_logged_profile"*

2.- Se abre la nueva interfaz: **UI_logged_profile**

section_logged_profile → classList.add("current")

//La información que aquí se muestra será datos del usuario y datos de su posición, se carga al iniciar la app, explicado en el diagrama de secuencia .- Inicio de la app. Por ello no se realiza ninguna función más en este apartado.

19 Cambiar password

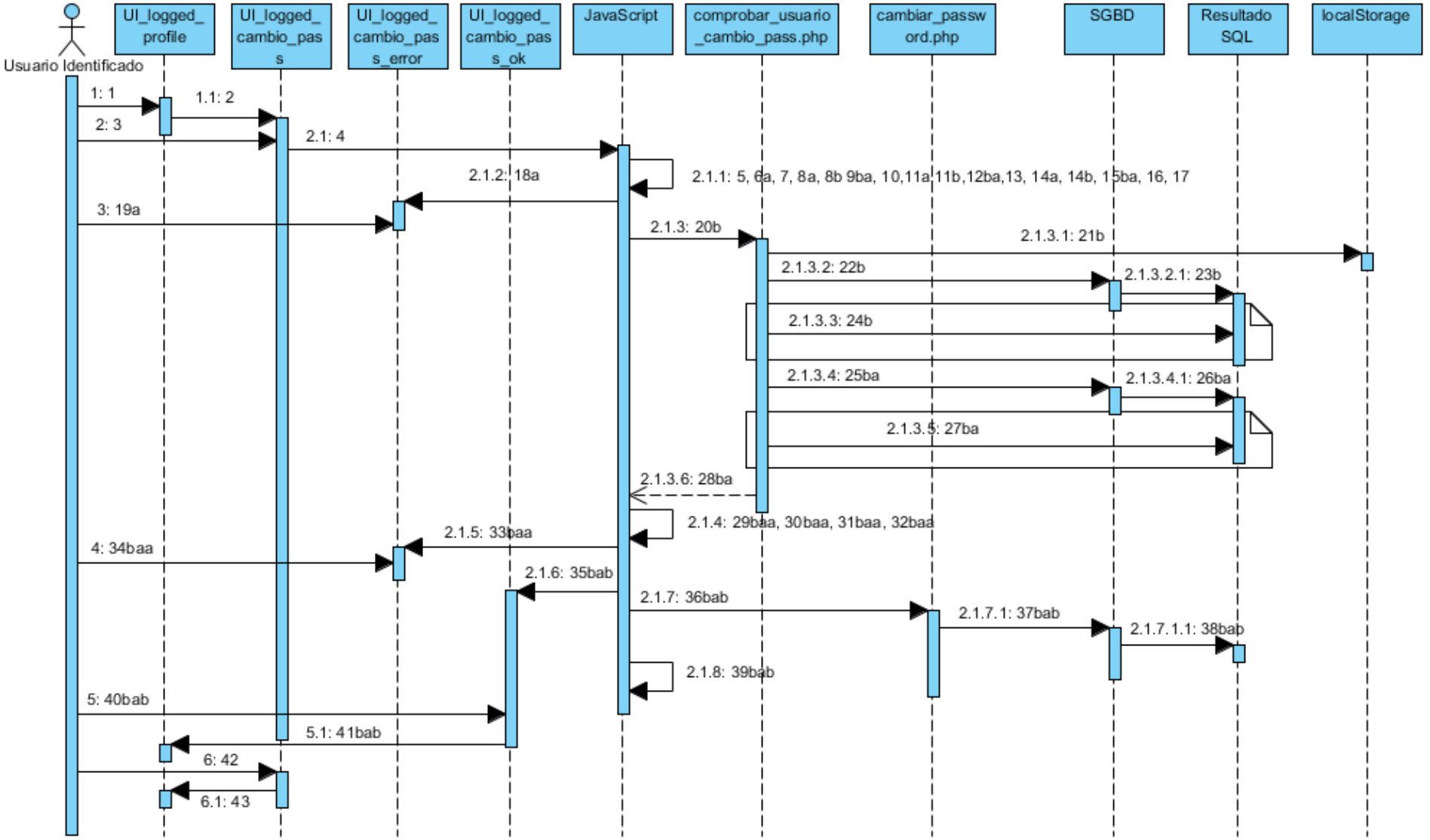


Ilustración 139: Diagrama de secuencia - Cambiar password

Precondición: está la app iniciada, el usuario identificado, en la interfaz UI_logged_profile.

1.- En la interfaz **UI_logged_profile**, pulsa "Cambiar contraseña":
data-article="article_logged_profile_cambiar_pass"

2.- Se abre la nueva interfaz: **UI_logged_cambio_pass**
article_logged_profile_cambiar_pass → classList.add("current")

3.- Rellena los campos. Pulsa en "Guardar cambios":
<button class="wide" data-l10n-id="id_guardar_cambios" id="id_article_logged_profile_cambiar_pass_btn_guardar_cambios">Guardar Cambios</button>

4.-
document.getElementById("id_article_logged_profile_cambiar_pass_btn_guardar_cambios").addEventListener("click",function() {...});

5.- Comprueba email: *validarEmailRegExp(email)*
[Si es incorrecto]
6a.- *imprimir_errores_email_valido()*
[Fin si]
7.- Comprueba si pass actual vacía: *validarPassRegExp(pass_actual)*
[Si pass actual es vacía]
8a.- *imprimir_errores_password_vacia_cambiar()*
[Si no es vacía]
8b.- Comprueba si pass actual tamaño mín:
validarTextoTamanioMinimo(pass_actual)
    [Si no tiene tamaño minimo]
    9ba.- *imprimir_errores_password_pequenia_cambiar()*
    [Fin si]
[Fin si]
10.- Comprueba si pass nueva vacía: *validarPassRegExp(pass_new)*
[Si pass nueva es vacía]
11a.- *imprimir_errores_password_vacia_cambiar_nueva()*
[Si no es vacía]
11b.- Comprueba si pass nueva tamaño mín:
validarTextoTamanioMinimo(pass_new)
    [Si no tiene tamaño minimo]
    12ba.- *imprimir_errores_password_pequenia_cambiar_nueva()*
    [Fin si]
[Fin si]
13.- Comprueba si pass nueva repetida vacía:

validarPassRegExp(pass_new_rep)
[Si pass nueva repetida es vacía]
14a.- *imprimir_errores_password_vacia_cambiar_nueva_rep()*
[Si no es vacía]
14b.- Comprueba si pass nueva repetida tamaño mín:
validarTextoTamanioMinimo(pass_new_rep)
[Si no tiene tamaño minimo]
15ba.-
imprimir_errores_password_pequenia_cambiar_nueva_rep()
[Fin si]
[Fin si]
[Si pass_nueva es diferente a pass_nueva_rep]
16.- *imprimir_errores_passwords_no_coinciden();*
[Fin si]
[Si pass_nueva es igual a pass_actual]
17.- *imprimir_errores_passwords_vieja_y_nueva_coinciden();*
[Fin si]
[Si hay algún error]
18a.- Se abre nueva interfaz encima de la actual:
UI_logged_cambio_pass_error: *Tako.Notification.error("deny",...)*
19a.- El usuario pulsa para cerrar el mensaje.
[Si no hay error]
20b.- Comprobar usuario en BD:
comprobar_usuario_logged_cambiar_pass_server(function(existe) {...});
21b.- Ver código usuario del almacenamiento local.
localStorage.getItem('codigo_usuario');
22b.- mysqli_query(SQL1)
SQL1:
"SELECT usuarios.cod_usuario, usuarios.nombre_usuario, usuarios.password_usuario, usuarios.passwordx_usuario, usuarios.email_usuario
FROM usuarios
WHERE (usuarios.email_usuario = '$email_usuario')"
23b.- *mysqli_result()*
[Mientras haya elementos]
24b.- $res[cod_usuario]
[Fin mientras]

[Si hay resultados]
25ba.- mysqli_query(SQL2)
SQL2:
"SELECT usuarios.cod_usuario, usuarios.nombre_usuario, usuarios.password_usuario, usuarios.passwordx_usuario, usuarios.email_usuario
FROM usuarios
WHERE (usuarios.email_usuario = '$email_usuario')
AND (usuarios.passwordx_usuario = '$passx_usuario')"
26ba.- *mysqli_result()*
[Mientras haya elementos]
27ba.-$res2[cod_usuario]
[Fin mientras]
28ba.- {devuelve texto (si existe email, emailclave, emailclaveusuarioidentificado, passnuevasdiferentes o cod_usuario}
[Si devuelve texto]
[Si devuelve email]
29baa.- *imprimir_errores_identificar_usuario_email_inexistente()*
[Fin si]
[Si devuelve emailclave]
30baa.- *imprimir_errores_identificar_usuario_email_clave_actual_no_coincide n();*
[Fin si]
[Si devuelve emailclaveusuarioidentificado]
31baa.- imprimir_errores_identificar_usuario_email_clave_usuario_identificad o*();*
[Fin si]
[Si devuelve passnuevasdiferentes]
32baa.- *imprimir_errores_passwords_no_coinciden()*
[Fin si]
33baa.- Se abre nueva interfaz encima de la actual:
UI_logged_cambio_pass_error:
Tako.Notification.error("deny",...)
34baa.- El usuario pulsa para cerrar el mensaje.
[Si no devuelve texto, sino un código de usuario (numérico)]

35bab.- Se abre nueva interfaz encima de la actual:
UI_logged_cambio_pass_ok
Tako.Notification.success("ok",...);
36bab.- Cambiar password en la BD:
cambiar_pass_server();
37bab.- mysqli_query(SQL3)
SQL3:
"UPDATE usuarios
SET usuarios.password_usuario = '$usuario_pass_nuevo',
usuarios.passwordx_usuario = '$passx_usuario'
WHERE (usuarios.email_usuario = '$usuario_email')"
38bab.- *mysqli_result()*

39bab.- Poner en blanco campos:
poner_en_blanco_cambiar_pass();
40bab.- El usuario pulsa en la interfaz
41bab.- Redirige a la interfaz **UI_logged_profile**.
section_logged_profile → classList.add("current")

[Fin si]

[Fin si]

42.- En **UI_logged_cambio_pass**, pulsa el botón atrás:
data-article="article_logged_profile_cambiar_pass"
43.- Se abre la interfaz **UI_logged_profile.**
article_logged_profile → classList.add("current")

20 Cerrar sesión

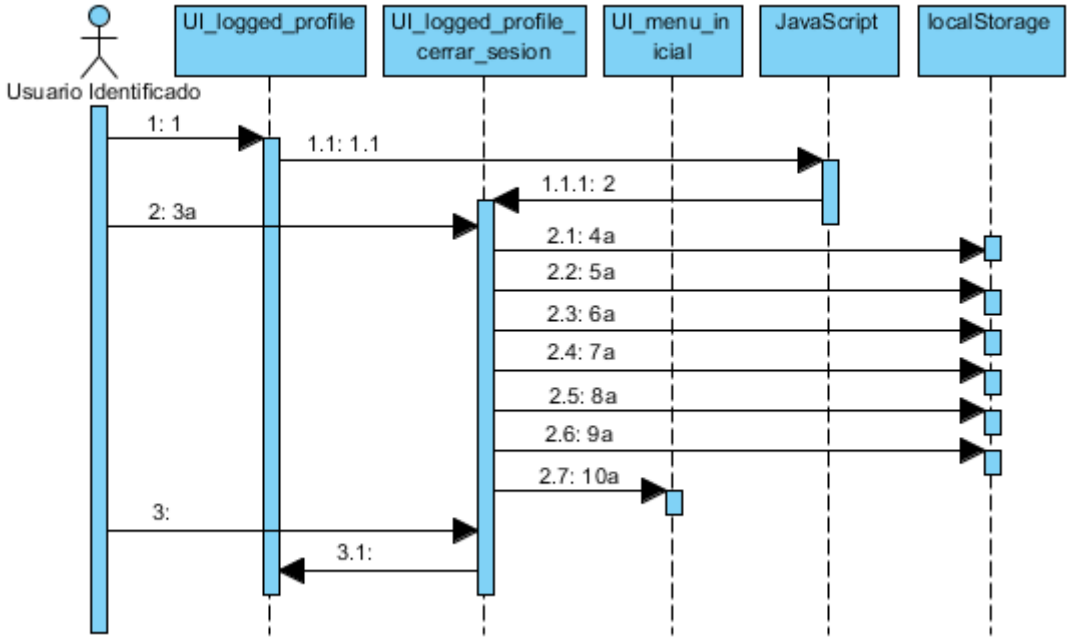


Ilustración 140: Diagrama de secuencia - Cerrar sesión

Precondición: está la app iniciada, el usuario identificado, en la interfaz UI_logged_profile.
1.- En la interfaz **UI_logged_profile**, pulsa el botón "Cerrar sesión":
1.1.- Cerrar la sesión del usuario actual
$("#btn_logged_profile_close_session").bind("tap", function(){...});
2.- Se abre la nueva interfaz: **UI_logged_cerrar_sesion**
Tako.Notification.confirm("logout,...)
[Si el usuario quiere cerrar sesión]
3a.- El usuario pulsa en "Cerrar Sesión", el botón verde.
4a.- Se cierra la sesión: *cerrar_sesion_datos_localStorage();*
5a.- Se actualiza el valor de la variable del almacenamiento local.
localStorage.removeItem('codigo_usuario');
6a.- Se actualiza el valor de la variable del almacenamiento local.
localStorage.removeItem('codigo_emision');
7a.- Se actualiza el valor de la variable del almacenamiento local.
localStorage.removeItem('codigo_emisora');
8a.- Se actualiza el valor de la variable del almacenamiento local.

localStorage.removeItem('codigo_emisora_seleccionada');
9a.- Se actualiza el valor de la variable del almacenamiento local.
localStorage.removeItem('seleccionado_alguna_vez');
10a.- Redirige a la interfaz **UI_menu_inicial**.
section_index → classList.add("current")
[Si no]
11b.- El usuario pulsa en "Cancelar", el botón rojo.
12b.- Redirige a la interfaz **UI_logged_profile**.
section_logged_profile → classList.add("current")

21 Dar baja al usuario

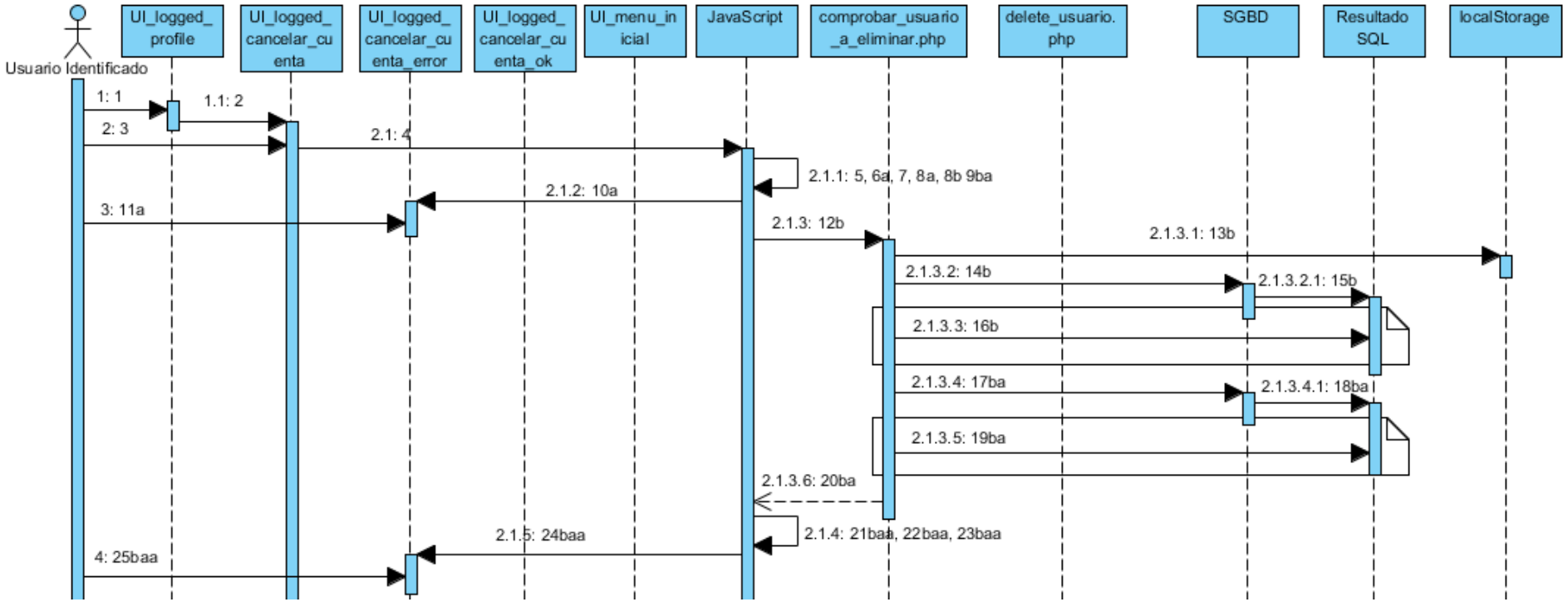


Ilustración 141: Diagrama de secuencia- Dar baja usuario (primera parte)

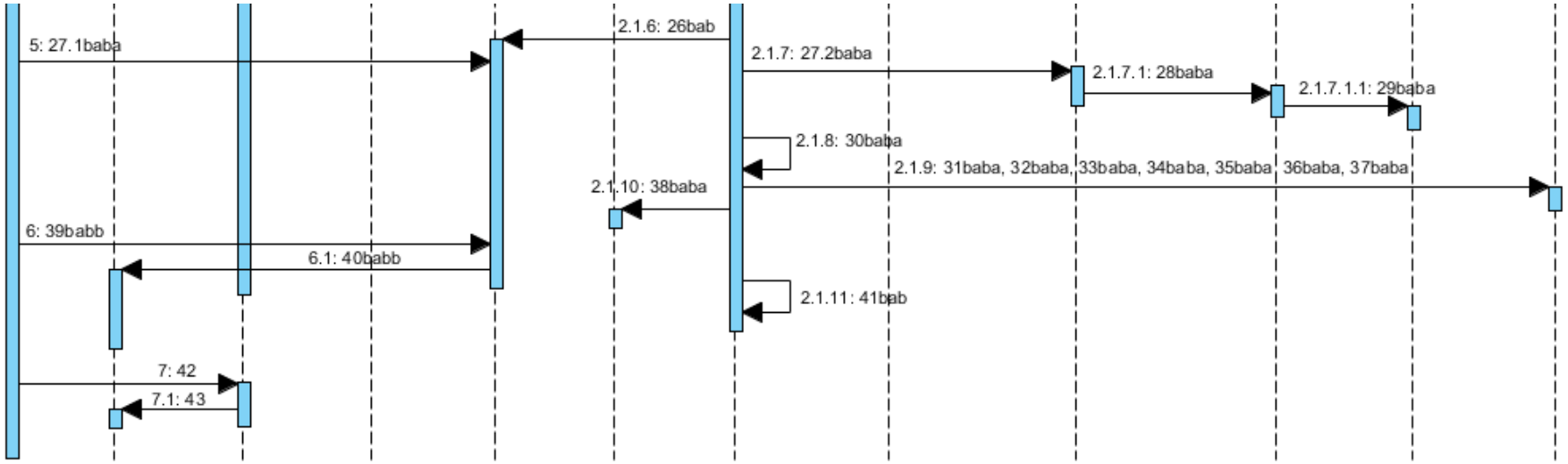


Ilustración 142: Diagrama de secuencia- Dar baja usuario (segunda parte)

Precondición: está la app iniciada, el usuario identificado, en la interfaz UI_logged_profile.
1.- En la interfaz **UI_logged_profile**, pulsa "Dar de baja la cuenta":
data-article="article_logged_profile_eliminar_usuario"
2.- Se abre la nueva interfaz: **UI_logged_cancelar_cuenta**
article_logged_profile_eliminar_usuario → classList.add("current")
3.- Rellena los campos. Pulsa en "Eliminar":
<button class="wide" data-l10n-id="id_eliminar"
id="id_article_logged_profile_eliminar_usuario_btn_eliminar">Eliminar
</button>
4.-
document.getElementById("id_article_logged_profile_eliminar_usuario_btn_eliminar").addEventListener("click",function() {...});
5.- Comprueba email: *validarEmailRegExp(email)*
[Si es incorrecto]
6a.- *imprimir_errores_email_valido()*
[Fin si]
7.- Comprueba si pass vacía: *validarPassRegExp(pass_actual)*
[Si pass actual es vacía]
8a.- *imprimir_errores_password_vacia_cambiar()*
[Si no es vacía]
8b.- Comprueba si pass tamaño mín:
validarTextoTamanioMinimo(pass)
[Si no tiene tamaño minimo]
9ba.- *imprimir_errores_password_pequenia_cambiar()*
[Fin si]
[Fin si]
[Si hay algún error]
10a.- Se abre nueva interfaz encima de la actual:
UI_logged_cancelar_cuenta_error *Tako.Notification.error("deny",..)*
11a.- El usuario pulsa para cerrar el mensaje.
[Si no hay error]
12b.- Comprobar usuario en BD:
comprobar_usuario_logged_server(function(existe){...});
13b.- Ver código usuario del almacenamiento local.
localStorage.getItem('codigo_usuario');
14b.- mysqli_query(SQL1)
SQL1:

"SELECT usuarios.cod_usuario, usuarios.nombre_usuario, usuarios.password_usuario, usuarios.passwordx_usuario, usuarios.email_usuario
FROM usuarios
WHERE (usuarios.email_usuario = '$email_usuario')"
15b.- *mysqli_result()*
[Mientras haya elementos]
16b.- $res[cod_usuario]
[Fin mientras]
[Si hay resultados]
17ba.- mysqli_query(SQL2)
SQL2:
"SELECT usuarios.cod_usuario, usuarios.nombre_usuario, usuarios.password_usuario, usuarios.passwordx_usuario, usuarios.email_usuario
FROM usuarios
WHERE (usuarios.email_usuario = '$email_usuario')
AND (usuarios.passwordx_usuario = '$passx_usuario')"
18ba.- *mysqli_result()*
[Mientras haya elementos]
19ba.-$res2[cod_usuario]
[Fin mientras]
20ba.- {devuelve texto (si existe email, emailclave, emailclaveusuarioidentificado o cod_usuario}
[Si devuelve texto]
[Si devuelve email]
21baa.-
imprimir_errores_identificar_usuario_email_inexistente()
[Fin si]
[Si devuelve emailclave]
22baa.-
imprimir_errores_identificar_usuario_email_clave_no_coinciden();
[Fin si]
[Si devuelve emailclaveusuarioidentificado]
23baa.-
imprimir_errores_identificar_usuario_email_clave_usuario_identificado*();*
[Fin si]

24baa.- Se abre nueva interfaz encima de la actual:
UI_logged_cancelar_cuenta_error:
Tako.Notification.error("deny",...)
25baa.- El usuario pulsa para cerrar el mensaje.
[Si no devuelve texto, sino un código de usuario (numérico)]
26bab.- Se abre nueva interfaz encima de la actual:
UI_logged_cancelar_cuenta_ok
Tako.Notification.confirm("unlink",...);
[Si el usuario quiere dar de baja la cuenta]
27.1baba.- Pulsa en "Eliminar cuenta", el botón verde.
27.2baba.- Eliminar usuario (poner inactivo en la BD):
eliminar_usuario_server();
28baba.- mysqli_query(SQL3)
SQL3:
"UPDATE usuarios
SET usuarios.activo_usuario = 'false'
WHERE (usuarios.email_usuario = '$email_usuario')
AND (usuarios.password_usuario = '$pass_usuario')
AND (usuarios.passwordx_usuario = '$passx_usuario')
AND (usuarios.activo_usuario = 'true')"
29baba.- *mysqli_result();*
30baba.- Eliminar usuario (eliminar datos almacenamiento local) *eliminar_usuario_datos_localStorage*
31baba.- Se actualiza variable del almacenamiento local.
localStorage.removeItem('codigo_usuario');
32baba.- Se actualiza variable del almacenamiento local.
localStorage.removeItem('codigo_emision');
33baba.- Se actualiza variable del almacenamiento local.
localStorage.removeItem('codigo_emisora');
34baba.- Se actualiza variable del almacenamiento local.
localStorage.removeItem('codigo_emisora_seleccionada');
35baba.- Se actualiza variable del almacenamiento local.
localStorage.removeItem('seleccionado_alguna_vez');
36baba.- Se actualiza variable del almacenamiento local.
localStorage.removeItem('info_ppal_notlogged_visto');
37baba.- Se actualiza variable del almacenamiento local.
localStorage.removeItem('info_ppal_logged_visto');
38baba.- Redirige a la interfaz **UI_menu_inicial**.

section_index → classList.add("current")
[Si se arrepiente y no quiere dar de baja la cuenta]
39babb.- Pulsa en "Cancelar", botón rojo.
40babb.- Redirige a la interfaz **UI_logged_profile**.
[Fin si]
41bab.- Poner en blanco campos:
poner_en_blanco_aniadir_usuario_logged();
[Fin si]
[Fin si]
42.- En **UI_logged_cancelar_cuenta**, pulsa el botón atrás:
data-article="article_logged_profile_cancelar_cuenta"
43.- Se abre la interfaz **UI_logged_profile.**
article_logged_profile → classList.add("current")

22 Compartir

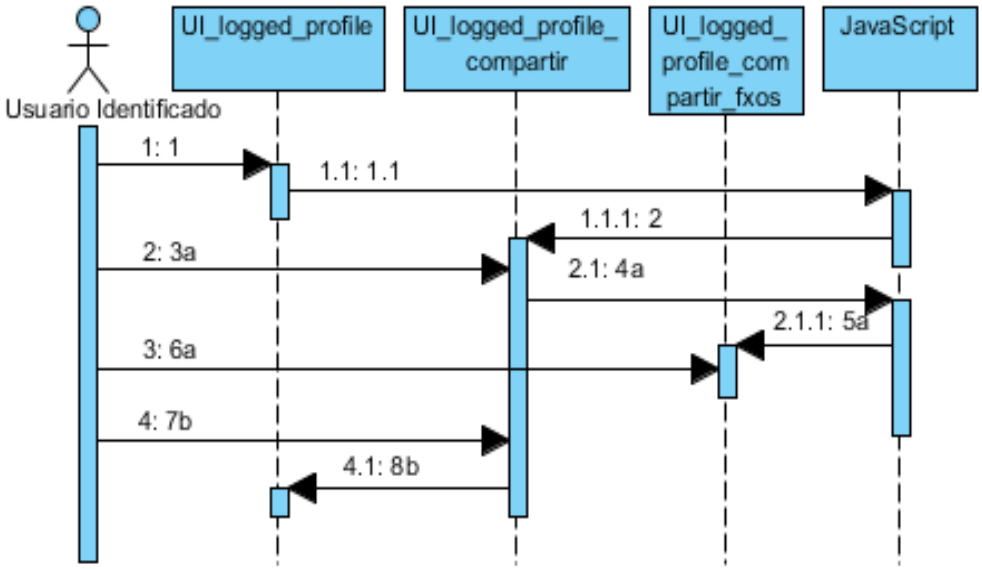


Ilustración 143: Diagrama de secuencia - Compartir

Precondición: está la app iniciada, el usuario identificado, en la interfaz UI_logged_profile.

1.- En la interfaz **UI_logged_profile**, pulsa el icono "Compartir":

1.1.- Compartir por correo el texto que se quiera.

$("#btn_logged_profile_share_notification_firefoxos").bind("tap", function(){...});

2.- Se abre la nueva interfaz: **UI_logged_compartir**

Tako.Notification.confirm("share",...)

[Si el usuario quiere compartir]

3a.- El usuario pulsa en "OK", el botón verde.

4a.- Se comparte la app. *callback(result)*

5a.- Se abre la nueva interfaz de compartir de Firefox OS: **UI_logged_compartir_fxos**

6a.- El usuario elige la opción a compartir deseada.

[Sino]

7b.- El usuario pulsa en "Ahora no".

8b.- Redirige a la interfaz **UI_logged_profile**.

section_logged_profile → classList.add("current")

[Fin si]

23 Comentar sobre la app y ver comentarios realizados

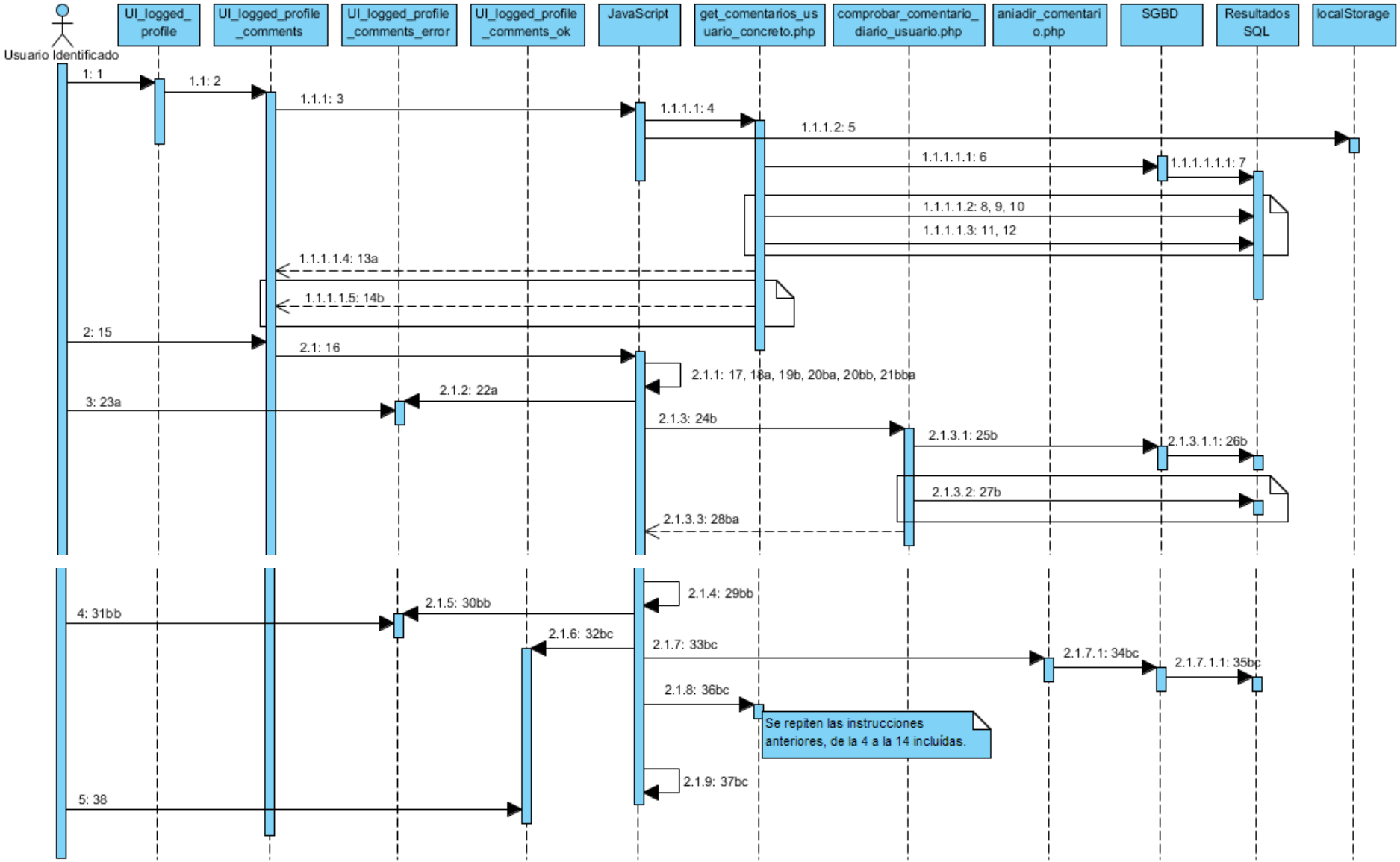


Ilustración 144: Diagrama de secuencia - Añadir comentario y listado de comentarios de usuario

Precondición: está la app iniciada en la interfaz **UI_logged_profile**. El usuario está identificado.
1.- El usuario pulsa en el botón "Comentarios", el segundo empezando por la izquierda de la barra inferior.
data-section="section_logged_profile_listado_comentarios"
2.- Se abre la nueva interfaz: **UI_logged_profile_comments**
section_logged_profile_listado_comentarios → classList.add("current")
3.-
document.getElementById("id_logged_profile_list_comentarios").addEventListener("click",function(){...});
4.- Cargar los comentarios escritos por ese usuario
listar_comentarios_propios_usuario_perfil();
5.- Obtener valor almacenamiento local
localStorage.getItem('codigo_usuario');
6.- mysqli_query(SQL1)
SQL1:
"SELECT usuarios.nombre_usuario, comentarios.latitud_comentario, comentarios.longitud_comentario, comentarios.fecha_hora, comentarios.texto_comentario
FROM comentarios
INNER JOIN usuarios
ON (usuarios.cod_usuario = comentarios.cod_usuario)
WHERE (comentarios.cod_usuario = '$cod_usuario')
ORDER BY comentarios.fecha_hora DESC"
7.- mysqli_result()
[Mientras haya elementos]
8.- $res[nombre_usuario]
9.- $res[latitud_comentario]
10.- $res[longitud_comentario]
11.- $res[fecha_hora]
12.- $res[texto_comentario]
[Fin mientras]
[Si no hay elementos (comentarios de ese usuario)]
13a.- Devuelve que no hay comentarios de ese usuario.
[Si si hay elemento]
[Mientras haya elementos (comentarios)]
14b.- Los datos obtenidos, escribirlos/rellenarlos en la interfaz:
echo $txt

[Fin si]

15.- En la interfaz **UI_logged_profile_comments**, el usuario escribe un comentario nuevo y pulsa "Guardar comentario":

<button class="wide" data-l10n-id="id_article_logged_profile_listado_comentario_btn_guardar_comentario" id="id_article_logged_profile_btn_guardar_comentario">Guardar comentario</button>

16.-

document.getElementById("id_article_logged_profile_btn_guardar_coment ario").addEventListener("click",function() {...});

17.- Comprueba si comentario vacío:

validarComentarioRegExp(comentario)

[Si comentario es vacío]

18a.- *imprimir_errores_comentario_vacia()*

[Si no es vacía]

19b.- Comprueba si comentario tamaño mínimo:

validarTextoTamanioMinimoComentario(comentario)

[Si no tiene tamaño minimo]

20ba.- *imprimir_errores_comentario_pequenia()*

[Si tiene tamaño mínimo]

20bb.- Comprobar si comentario tamaño maxomo:

validarTextoTamanioMaximoComentario(comentario)

[Si tiene demasiado tamaño]

21bba.- *imprimir_errores_comentario_grande()*

[Fin si]

[Fin si]

[Fin si]

[Si hay algún error]

22a.- Se abre nueva interfaz encima de la actual:

UI_logged_profile_comments_error:

Tako.Notification.error("deny",...)

23a.- El usuario pulsa para cerrar el mensaje.

[Si no hay error]

24b.- Comprobar comentario diario en BD:

comprobacion_logged_comentario_por_dia_server(function(existe){...});

25b.- mysqli_query(SQL2)

SQL2:

"SELECT comentarios.fecha_hora FROM comentarios

WHERE (comentarios.cod_usuario = '$cod_usuario')

ORDER BY comentarios.fecha_hora ASC"
26b.- *mysqli_result()*
[Mientras haya elementos]
27b.- $res[fecha_hora]
[Fin mientras]
[Si hay resultados(comentarios)]
// Comprueba si hay escritos en las últimas 24horas o no
28ba.- {devuelve texto (si existe yahaycomentario o vacio}
[Fin si]
[Si devuelve texto]
[Si devuelve yahaycomentario]
29bb- *imprimir_errores_add_comment_diario_ya_existe()*
[Fin si]
30bb.- Se abre nueva interfaz encima de la actual:
UI_logged_profile_comment_error:
Tako.Notification.error("deny",...)
31bb.- El usuario pulsa para cerrar el mensaje.
[Si no devuelve texto, sino vacio]
32bc.- Se abre nueva interfaz encima de la actual:
UI_logged_profile_comment_ok
Tako.Notification.success("ok",...);
33bc.- Añadir comentario a la BD:
aniadir_comentario_server);
34bc.- mysqli_query(SQL3)
SQL3:
"INSERT INTO comentarios (cod_usuario, latitud_comentario, longitud_comentario, fecha_hora, texto_comentario)
VALUES ('$cod_usuario', '$comentario_lat', '$comentario_long','$fecha_hora', '$comentario_texto_utf')"
35bc.- *mysqli_result()*
36bc.- Cargar los comentarios escritos por ese usuario
listar_comentarios_propios_usuario_perfil();
//Esta función se repite, la volvemos a llamar. Las
//instrucciones son las indicadas entre los puntos 4 y 14
//(incluidos) de este mismo diagrama.
37bc.- Poner en blanco campos:

poner_en_blanco_aniadir_comment();

38bc.- El usuario pulsa en la interfaz para cerrar el mensaje.

[Fin si]

[Fin si]

24 Ver emisoras favoritas

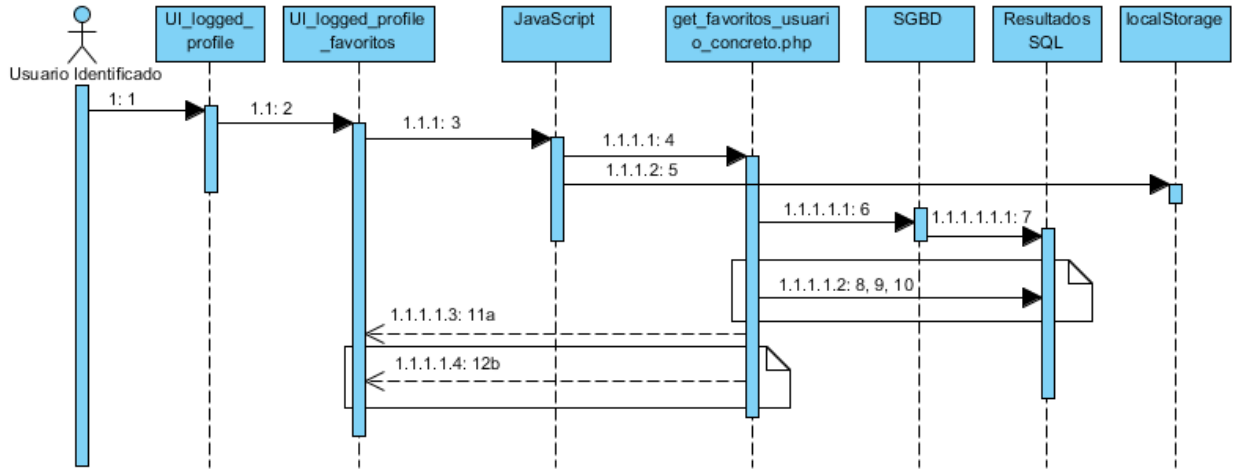


Ilustración 145: Diagrama de secuencia - Listado favoritos

Precondición: está la app iniciada en la interfaz **UI_logged_profile**. El usuario está identificado.

1.- El usuario pulsa en el botón "Favoritos", en el medio de la barra inferior.

data-section="section_logged_profile_listado_favoritos"

2.- Se abre la nueva interfaz: **UI_logged_profile_favoritos**

section_logged_profile_listado_favoritos → classList.add("current")

3.-

document.getElementById("id_logged_profile_list_favoritos").addEventListener("click",function(){...});

4.- Cargar las emisoras favoritas de ese usuario

listar_favoritos_propios_usuario_perfil();

5.- Obtener valor almacenamiento local

localStorage.getItem('codigo_usuario');

6.- mysqli_query(SQL1)

SQL1:

"SELECT emisoras.nombre_emisora, usuarios.nombre_usuario, favoritos.fecha_hora
FROM favoritos
INNER JOIN emisoras
ON (emisoras.cod_emisora = favoritos.cod_emisora)

INNER JOIN usuarios
ON (usuarios.cod_usuario = favoritos.cod_usuario)
WHERE (favoritos.cod_usuario = '$cod_usuario')
ORDER BY favoritos.fecha_hora DESC"
7.- *mysqli_result()*
[Mientras haya elementos]
8.- $res[nombre_emisora];
9.- $res[nombre_usuario]
10.- $res[fecha_hora]
[Fin mientras]
[Si no hay elementos (favoritos de ese usuario)]
    11a.- Devuelve que no hay favoritos de ese usuario.
[Si si hay elemento]
    [Mientras haya elementos (favoritos)]
    12b.- Los datos obtenidos, escribirlos/rellenarlos en la interfaz:
    echo $txt
[Fin si]

25 Ver emisoras puntuadas

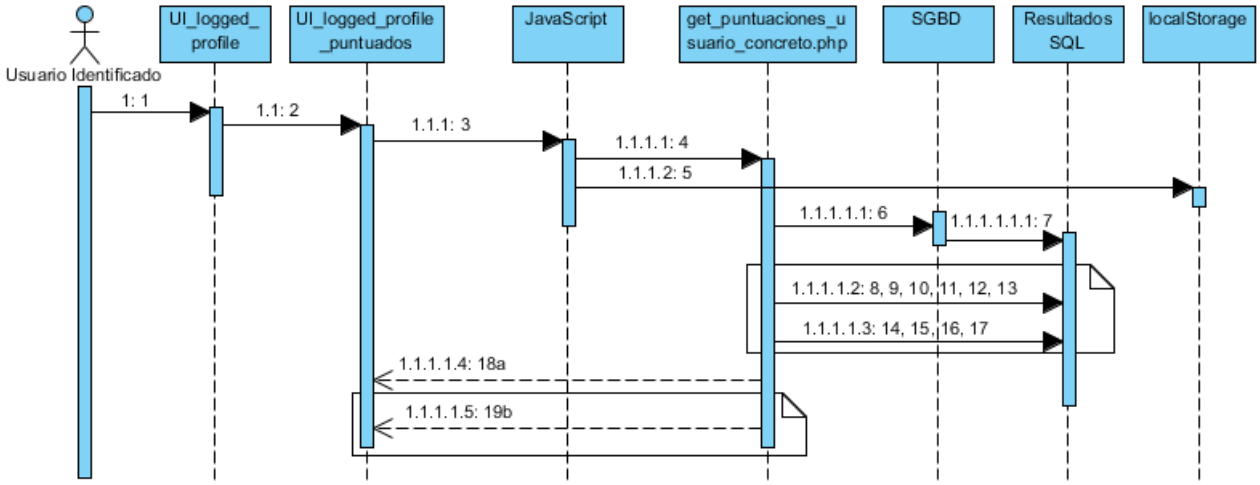


Ilustración 146: Diagrama de secuencia - Listado puntuaciones realizadas

Precondición: está la app iniciada en la interfaz **UI_logged_profile**. El usuario está identificado.

1.- El usuario pulsa en el botón "Puntuados", el segundo empezando por la derecha de la barra inferior.

data-section="section_logged_profile_listado_puntuados"

2.- Se abre la nueva interfaz: **UI_logged_profile_puntuados**

section_logged_profile_listado_puntuados → classList.add("current")

3.-

document.getElementById("id_logged_profile_list_puntuados").addEventListener("click",function(){...});

4.- Cargar las emisiones puntuadas de ese usuario

listar_puntuados_propios_usuario_perfil();

5.- Obtener valor almacenamiento local

localStorage.getItem('codigo_usuario');

6.- mysqli_query(SQL1)

SQL1:

"SELECT emisoras.nombre_emisora, emisiones.frecuencia_emision, emisiones.latitud_emision, emisiones.longitud_emision, usuarios.nombre_usuario, puntuaciones.cod_usuario, puntuaciones.cod_emision, puntuaciones.latitud_puntual_usuario, puntuaciones.longitud_puntual_usuario, puntuaciones.fecha_hora,

puntuaciones.puntos
FROM puntuaciones
INNER JOIN emisiones
ON (emisiones.cod_emision = puntuaciones.cod_emision)
INNER JOIN emisoras
ON (emisiones.cod_emisora = emisoras.cod_emisora)
INNER JOIN usuarios
ON (usuarios.cod_usuario = puntuaciones.cod_usuario)
WHERE (puntuaciones.cod_usuario = '$cod_usuario')
ORDER BY puntuaciones.fecha_hora DESC"
7.- *mysqli_result()*
[Mientras haya elementos]
8.- $res[nombre_usuario]
9.- $res[nombre_emisora]
10.- $res[cod_emision]
11.- $res[frecuencia_emision]
12.- $res[latitud_emision]
13.- $res[longitud_emision]
14.- $res[latitud_puntual_usuario]
15.- $res[longitud_puntual_usuario]
16.- $res[fecha_hora]
17.- $res[puntos]
[Fin mientras]
[Si no hay elementos (puntos)]
18a.- Devuelve que no hay puntuaciones a ninguna emisión por parte de ese usuario.
[Si si hay elemento]
[Mientras haya elementos (puntos)]
19b.- Los datos obtenidos, escribirlos/rellenarlos en la interfaz:
echo $txt
[Fin si]

26 Ver emisoras y emisiones añadidas

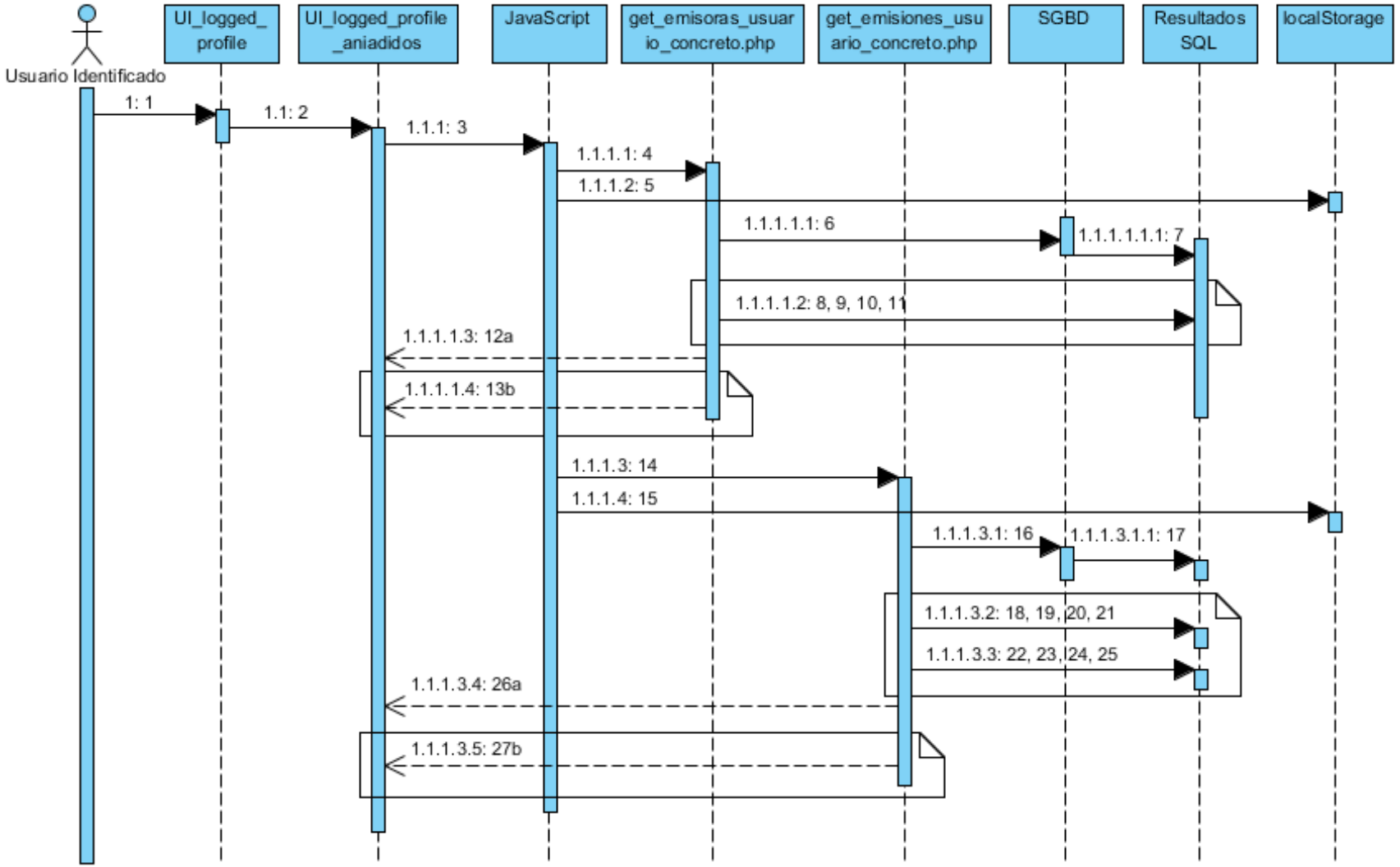


Ilustración 147: Diagrama de secuencia - Ver emisoras y emisiones añadidas

Precondición: está la app iniciada en la interfaz **UI_logged_profile**. El usuario está identificado.
1.- El usuario pulsa en el botón "Añadidos",
el primer botón de la barra inferior por la derecha.
data-section="section_logged_profile_listado_aniadidos"
2.- Se abre la nueva interfaz:
UI_logged_profile_emisoras_y_emisiones_aniadidas
section_logged_profile_listado_aniadidos → classList.add("current")
3.-
document.getElementById("id_logged_profile_list_aniadidos").addEventListener("click",function(){...});
4.- Cargar las emisoras añadidas por el usuario
listar_aniadidos_emisoras_propios_usuario_perfil();
5.- Obtener valor almacenamiento local
localStorage.getItem('codigo_usuario');
6.- mysqli_query(SQL1)
SQL1:
"SELECT emisoras.nombre_emisora,
alcances_tipos.nombre_alcance_tipo, emisoras.fecha_hora,
usuarios.nombre_usuario
FROM emisoras
INNER JOIN alcances_tipos
ON (emisoras.alcance_emisora = alcances_tipos.cod_alcance_tipo)
INNER JOIN usuarios
ON (emisoras.cod_usuario = usuarios.cod_usuario)
WHERE (emisoras.cod_usuario = '$cod_usuario')
ORDER BY emisoras.cod_emisora"
7.- mysqli_result()
[Mientras haya elementos]
8.- $res[nombre_emisora]";
9.- $res[nombre_alcance_tipo]";
10.- $res[fecha_hora]";
11.- $res[nombre_usuario]";
[Fin mientras]
[Si no hay elementos (emisoras)]
12a.- Devuelve que no ha añadido emisoras este usuario.
[Si si hay elemento]
[Mientras haya elementos (emisoras)]

13b.- Los datos obtenidos, escribirlos/rellenarlos en la interfaz:
echo $txt
[Fin si]
14.- Cargar las emisiones añadidas por el usuario
listar_aniadidos_emisiones_propios_usuario_perfil();
15.- Obtener valor almacenamiento local
localStorage.getItem('codigo_usuario');
16.- mysqli_query(SQL1)
SQL1:
"SELECT emisiones.cod_emision, emisiones.frecuencia_emision,
emisiones.latitud_emision, emisiones.longitud_emision,
emisiones.posicion_emision, emisiones.fecha_hora,
usuarios.nombre_usuario, emisoras.nombre_emisora
FROM emisiones
INNER JOIN emisoras
ON (emisiones.cod_emisora = emisoras.cod_emisora)
INNER JOIN usuarios
ON (emisiones.cod_usuario = usuarios.cod_usuario)
WHERE (emisiones.cod_usuario = '$cod_usuario')
ORDER BY emisiones.fecha_hora DESC"
17.- *mysqli_result()*
[Mientras haya elementos]
18.- $res[cod_emision]
19.- $res[frecuencia_emision]
20.- $res[latitud_emision]
21.- $res[longitud_emision]
22.- $res[posicion_emision]
23.- $res[nombre_emisora]
24.- $res[fecha_hora]
25.- $res[nombre_usuario]";
[Fin mientras]
[Si no hay elementos (emisiones)]
26a.- Devuelve que no ha añadido emisiones este usuario.
[Si si hay elemento]
[Mientras haya elementos (emisiones)]
27b.- Los datos obtenidos, escribirlos/rellenarlos en la interfaz:
echo $txt
[Fin si]

27 Registrar incidencia

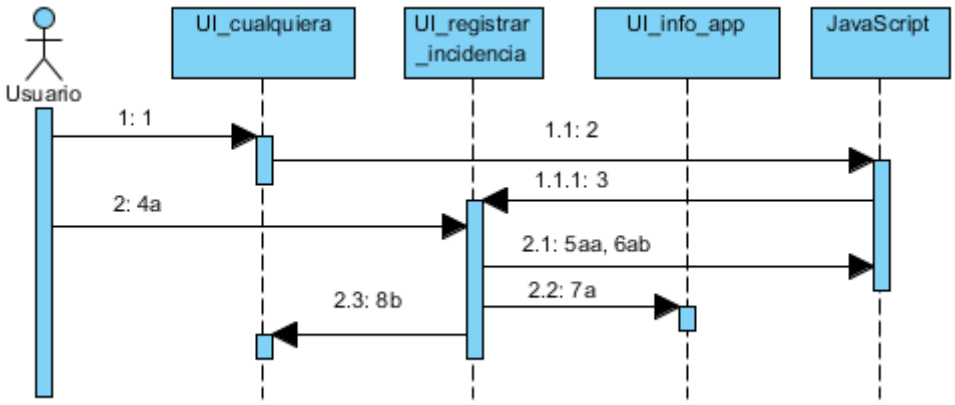


Ilustración 148: Diagrama de secuencia - Registrar incidencia

Precondición: está la app iniciada.
1.- Desde cualquier interfaz, el usuario agita el dispositivo.
window.addEventListener('shake', shakeEvent, false); //shakeEvent();
2.- Vibra el dispositivo *vibrateOneTime();*
3.- Se abre la nueva interfaz: **UI_registrar_incidencia**
Tako.Notification.confirm("hammer",...)
[Si el usuario quiere registrar una incidencia]
4a.- El usuario pulsa en "OK", el botón verde.
    [Si el usuario está identificado]
    5aa.- Cambiar valor variable interna
    back_path_info_logged_radio_ui();
    [Sino]
    6ab.- Cambiar valor variable interna
    back_path_info_index();
    [Fin si]
7a.- Se abre la interfaz **UI_info_app**
//El funcionamiento ahora, está explicado en el diagrama de secuencia
Ver información y datos de acceso del desarrollador de la app.
[Sino]
8b.- Se cierra la interfaz, volviendo a la que estaba antes de ser llamada.
[Fin si]

Anexo III
Manuales sobre uso de Localization - L10N

Anexo III - Manuales sobre uso de Localization - L10N

En este apartado, se explicará los conceptos claros del uso de la librería L10N. Para más información, se aconseja que se documenten con la información que aparece en los recursos bibliográficos indicados en el documento.

Habrá que incluir en el "html" principal la siguiente línea de código, después de cargar todos las hojas de estilo, en el "head":

```
<link rel="prefetch" type="application/l10n"
href="data/locales.ini" />
```

Después, antes de cerrar el "body" del "html", se añade el script donde carga la librería l10n.js, la que va a dar la funcionalidad deseada:

```
<script type="text/javascript" src="./js/libs/l10n.js"></script>
```

Se añade también otro script donde se llama a la función navigator.mozL10n.get para las traducciones:

```
<script type="text/javascript" src="./js/libs/app.js"></script>
```

Y que contiene:

```
window.addEventListener('DOMContentLoaded', function(){
    'use strict';
    var translate = navigator.mozL10n.get;
    navigator.mozL10n.once(start);
    function start() {
        var msg = document.getElementById('msg');
    }
});
```

En los apartados del código donde se usen las variables, habrá que identificarlos de este modo:

```
<li data-l10n-id="id_email" class="separator"
style="color:black;" >Email</li>

<li data-l10n-id="id_password" class="separator"
style="color:black;" >Password</li>

<li data-l10n-id="id_password_bis" class="separator"
style="color:black;" >Repetir Password</li>
```

Si hay algún valor que se carga desde JavaScript, hay que utilizar otra función para la carga del idioma correcto. En este caso, a cada valor que se quiera imprimir desde el fichero .js, será necesario hacer la llamada siguiente:

```
var pass = navigator.mozL10n.get("id_password");
```

Se almacena en *pass* el valor obtenido a través de la función *navigator.mozL10n.get( )*, pasándole correctamente la variable a mostrar.

En el directorio raíz, habrá una carpeta llamada data. Dentro de ella, el archivo *locales.ini* donde van indicados todos los archivos de idiomas que se quieren tener en cuenta en la app.

```
[en-US]

@import url(en-US.properties)

[es]

@import url(es.properties)

[eu]

@import url(eu.properties)

[fr]

@import url(fr.properties)
```

En cada uno de esos archivos, se encuentran todas las variables que se utilizan en el desarrollo de la app, individualizados para cada idioma. Por lo

que, todas las variables que se usan tanto en el código html como en JavaScript debe estar listado en los todos idiomas para que aparezcan representados perfectamente en la aplicación. En caso de que una variable no esté traducida, se mostrará la variable en el idioma por defecto, que en este caso será el inglés.

En este ejemplo se muestran parte del contenido de los archivos en español, inglés, euskera y francés.

es.properties	en-US.properties	eu.properties	fr.properties
id_fmradiords = FM Radio RDS	id_fmradiords = FM Radio RDS	id_fmradiords = FM Irrati RDS	id_fmradiords = FM Radio RDS
id_email = Email	id_email = Email	id_email = Posta elektronikoa	id_email = Adresse email
id_password = Contraseña	id_password = Password	id_password = Pasahitza	id_password = Mot de passe
id_password_bis = Repetir contraseña	id_password_bis = Repeat password	id_password_bis = Pasahitza errepikatu	id_password_bis = Repeter mot de passe
id_password_ recordar = No recuerdo el contraseña	id_password_ recordar = I don't remember my password	id_password_ recordar = Ez dut gogoratzen pasahitza	id_password_ recordar = Je ne connais pas mon mot de passe

La aplicación automáticamente reconoce el idioma del sistema, del dispositivo desde donde se esté ejecutando la aplicación, y lo adapta.