

DAS - Segunda Entrega Individual

Irune Palacios Orueta

22 de abril de 2025

Abstract

Esta segunda entrega del trabajo profundiza en el desarrollo de aplicaciones móviles en Android Studio, gestionando tareas en primer y segundo plano, ofreciendo diferentes servicios con conexión a Internet, y cuidando la experiencia del usuario mediante nuevos elementos como widgets.

Se pide que la aplicación cumpla al menos una serie de requisitos mínimos. También se especifican otros elementos que se pueden utilizar para implementar otras funcionalidades, siempre que estas tengan un mínimo de sentido en la aplicación.

Contents

1	Enlace al repositorio de código Git	3
2	Elementos de valoración utilizados y sus funcionalidades	3
2.1	Requisitos mínimos	3
2.1.1	Gestión de usuarios en la base de datos remota	3
2.1.2	Open Street Maps y Geolocalización	3
2.1.3	Gestión de imágenes en el servidor	4
2.2	Adicionales	4
2.2.1	<i>ContentProvider</i>	4
2.2.2	Servicio en primer plano y mensajes broadcast	5
2.2.3	<i>Widget</i>	5
2.2.4	Tarea programada mediante alarma	5
3	Descripción de las clases y la base de datos	5
4	Manual de uso	11
5	Dificultades encontradas a la hora de realizar el trabajo	19
6	Bibliografía	20

1 Enlace al repositorio de código Git

Todo el código se encuentra en el siguiente repositorio de GitHub:

<https://github.com/Irune27/DAS-Entrega1>

2 Elementos de valoración utilizados y sus funcionalidades

A partir del trabajo realizado para la primera entrega, he continuado desarrollando mi aplicación Recipe Manager. Ahora, además de permitir a los usuarios almacenar sus propias recetas, implementa nuevas funcionalidades como la gestión de usuarios, el uso de geolocalización o servicios en primer plano, entre otros.

2.1 Requisitos mínimos

Se han añadido los siguientes elementos:

- Gestión de usuarios en la base de datos remota ✓
- Open Street Maps y Geolocalización ✓
- Gestión de imágenes en el servidor ✓

2.1.1 Gestión de usuarios en la base de datos remota

Al abrir la aplicación, lo primero que verá el usuario ahora es una pantalla de inicio de sesión. Si ya tiene una cuenta, podrá introducir su nombre de usuario y contraseña; si todavía no la tiene, podrá pasar a la pantalla de registro mediante el botón indicado. Además, la aplicación cuenta con un "modo offline": este permite acceder a la aplicación sin necesidad de iniciar sesión. En este caso, las recetas se guardarán localmente, igual que en la versión desarrollada para la primera entrega.

2.1.2 Open Street Maps y Geolocalización

Los servicios de Open Street Maps y Geolocalización están integrados en la actividad MapActivity, a la que se puede acceder a través del menú principal. Aquí, se determinará la ubicación del usuario y se mostrarán todos los restaurantes, cafeterías y supermercados cercanos (en un radio de 1km), utilizando Overpass API.

Mediante los tres botones correspondientes, se puede controlar qué puntos de interés se marcan. Al principio, los tres estarán activados (restaurantes, cafeterías y supermercados). Basta con pulsarlos para desactivarlos y volver a activarlos.

Si el usuario mantiene presionado sobre cualquier punto del mapa, se borrarán los marcadores anteriores y se marcarán restaurantes, cafeterías y supermercadoss cercanos a ese nuevo punto. Estas instrucciones están disponibles para el usuario en el botón de ayuda, situado en la parte inferior izquierda del mapa.

El mapa también gestiona la conexión del dispositivo. Si se accede al mapa sin conexión, se le notificará al usuario que los puntos de interés no se han podido cargar; también se limitará más el alcance del mapa (se establecerá un Bounding Box más pequeño). Sin embargo, en el momento en el que el usuario recupere la conexión, se mostrarán los puntos de interés y se ampliará el área visible del mapa sin necesidad de reiniciar la actividad.

2.1.3 Gestión de imágenes en el servidor

En el modo online de Recipe Manager (entrando con un nombre de usuario y una contraseña), hay dos tipos de imágenes que gestionar: las que están relacionadas con recetas y las de los perfiles de usuario. Aun así, en los dos casos el método para guardarlas y mostrarlas es el mismo.

Cuando se saca una foto o se elige una de la galería, primero se guardará esta imagen en el almacenamiento local. Despues, se subirá a un directorio en el servidor web y se borrará la imagen local. Por último, se guardará esta nueva ruta en el campo correspondiente de la base de datos MySQL.

Las imágenes de recetas están guardadas en el directorio *recipe_images* del servidor remoto, y las de los perfiles en *user_images*. Si el usuario edita su foto de perfil o la de alguna receta y asigna una nueva imagen, la antigua se eliminará del servidor.

Además, cada uno de estos directorios cuenta con un archivo de imagen "predeterminado": *recipe_images/default_image.jpg* y *user_images/default_user.png*. Estos archivos no se borrarán en ningún momento, y habrá una única copia de ellos.

2.2 Adicionales

Además, se han implementado los siguientes elementos opcionales:

- *ContentProvider* ✓
- Servicio en primer plano y mensajes broadcast ✓
- *Widget* ✓
- Tarea programada mediante alarma ✓

2.2.1 *ContentProvider*

Un *ContentProvider* es una clase que abstrae el acceso a datos. Permite que esos datos sean accesibles a través de URIs y que se acceda a ellos de manera uniforme

(insertar, consultar, actualizar, eliminar) mediante un *ContentResolver*. En el caso de Recipe Manager, se implementa en la clase RecipeProvider, y facilita la interacción con la base de datos (MyDB) en el modo offline. También se utilizar en RecipeWidget.

2.2.2 Servicio en primer plano y mensajes broadcast

El servicio en primer plano implementado es la sincronización de datos del servidor con el local. En la pantalla con la lista de recetas (MainActivity), hay un nuevo botón en la parte inferior: "Descargar recetas al local". Este botón solo estará disponible cuando haya un usuario con la sesión iniciada (en modo online). Al pulsarlo, las recetas de ese usuario en el servidor comenzarán a descargarse a la base de datos local.

Durante el servicio, aparecerá una notificación indicando que la sincronización está en proceso, y cuando finalice, un mensaje *Toast*, gestionado mediante mensajería broadcast. Las recetas descargadas seguirán estando disponibles online (no se borran), y también se filtra por el nombre de las recetas para evitar descargar las mismas más de una vez.

2.2.3 Widget

La aplicación cuenta con un *Widget* que se actualiza cada 24 horas y muestra la "receta del día". Esta será elegida aleatoriamente entre todas las recetas almacenadas localmente, y se mostrarán su nombre y su imagen, como en la lista de recetas. Si se pulsa sobre cualquier parte del *Widget*, se llevará al usuario a MainActivity, en modo offline.

2.2.4 Tarea programada mediante alarma

La tarea programada mediante alarma es la notificación diaria. Todos los días, a las 19:00, se mostrará una notificación recordándole al usuario que tiene a su disposición la lista de recetas, y que podría echar un vistazo y elegir alguna para cocinar el siguiente día. Para esto, se utiliza un *AlarmManager* para gestionar el tiempo y una clase *NotificationReceiver* que se encarga de lanzar la notificación en el momento adecuado.

3 Descripción de las clases y la base de datos

El diagrama de clases entero se encuentra en el repositorio de GitHub del proyecto (el archivo *classDiagram.puml*). Como es demasiado grande para verlo bien, en esta sección aparece dividido en varios subdiagramas.

Las Figuras 1, 2 y 3 muestran las clases relacionadas con la gestión de las recetas. Estas clases no presentan grandes cambios respecto a la primera versión de la aplicación, excepto los métodos necesarios para gestionar el modo online.

Las clases nuevas más relevantes son las siguientes:

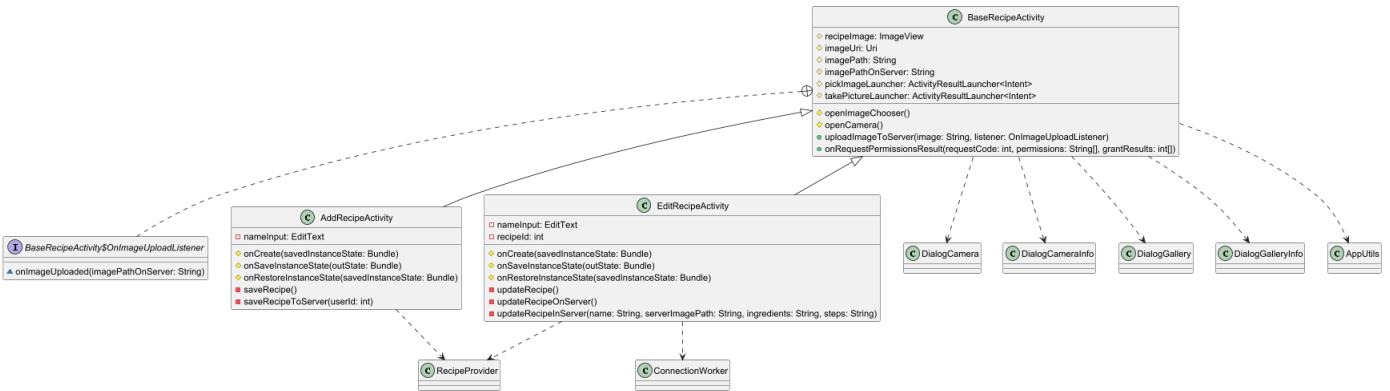


Figure 1: Base para añadir y editar recetas.

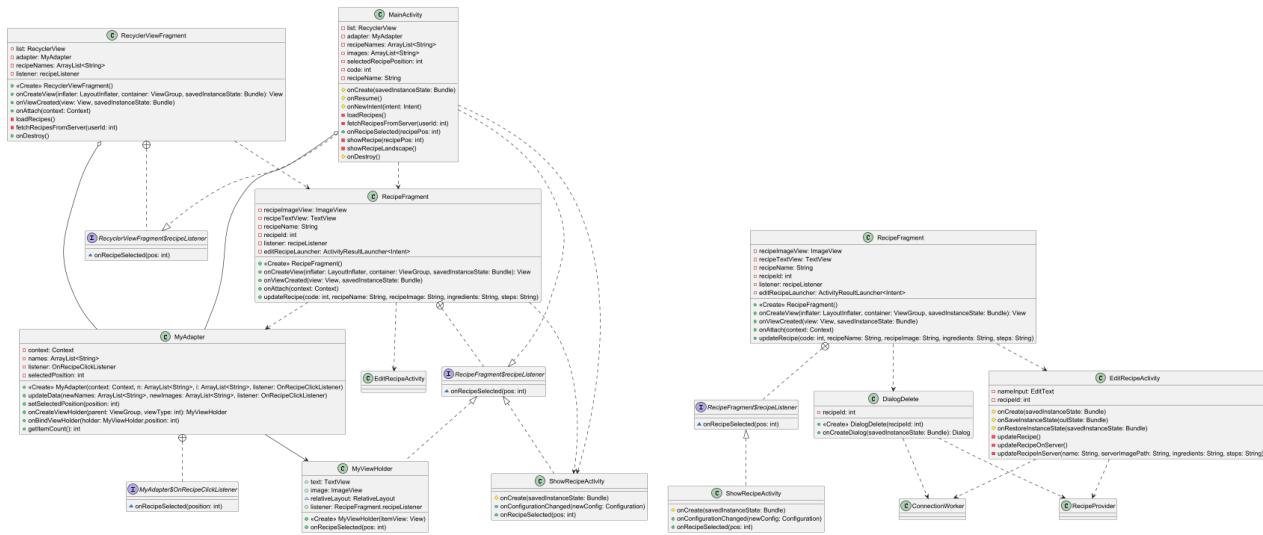


Figure 2: Lista de recetas.

Figure 3: Detalle de receta.

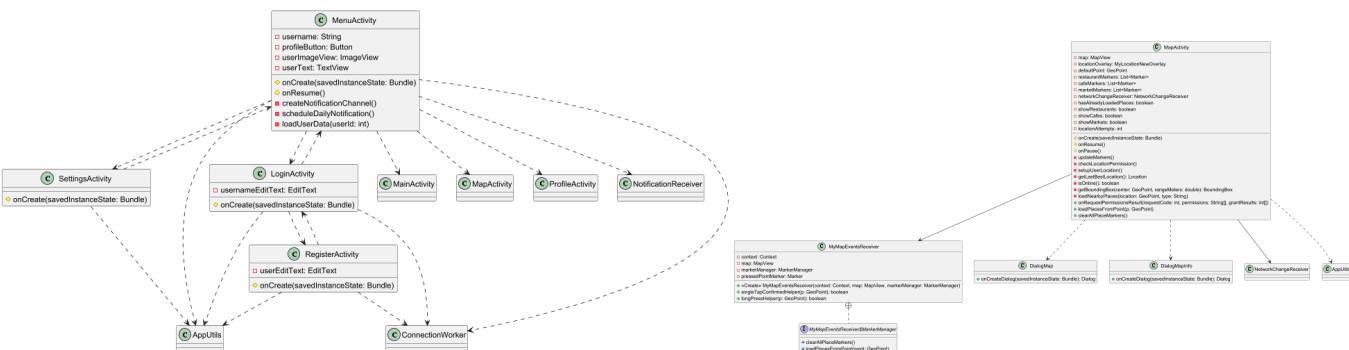


Figure 4: Menú de navegación.

Figure 5: Mapa.

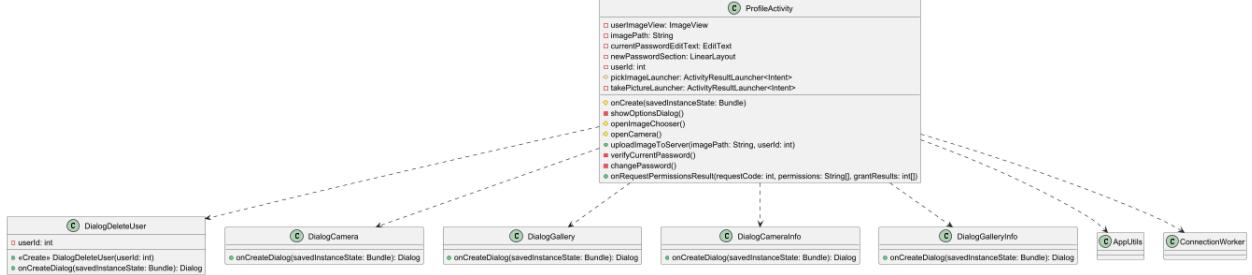


Figure 6: Perfil de usuario.

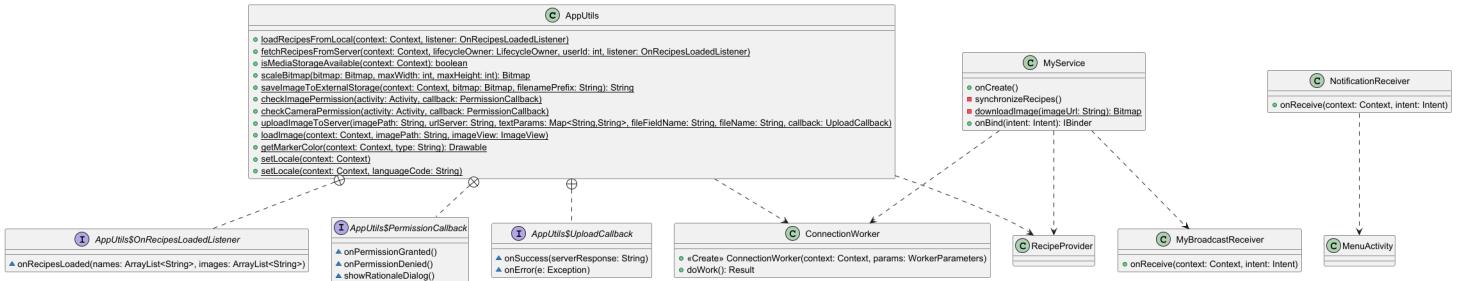


Figure 7: Utilidades.

- **MenuActivity.java** (Figura 4): muestra el menú principal de la aplicación. Permite acceder a todo el resto de funcionalidades. Además, ahora se programa aquí la notificación diaria (antes se hacía en MainActivity).
- **MapActivity.java** (Figura 5): se encarga del mapa. Gestiona la conexión mediante NetworkChangeReceiver, que avisa al usuario si hay cambios en la red (si había conexión y ya no, o al revés). Implementa MyMapEventsReceiver para controlar los pulsados largos del usuario en el mapa.
- **ProfileActivity.java** (Figura 6): contiene la información relativa al usuario, y permite cambiarla.
- **AppUtils.java** (Figura 7): clase utilitaria. Recoge funciones que se utilizan en diferentes actividades, fragmentos, diálogos...
- **ConnectionWorker.java**: gestiona las peticiones a la API y recoge las respuestas.
- **MyService.java** (Figura 7): pone en marcha el servicio en primer plano para la descarga de recetas del servidor a la base de datos local. Crea una notificación que desaparecerá en cuanto termine el servicio, y entonces notificará a la clase MyBroadcastReceiver.
- **MyBroadcastReceiver.java** (Figura 7): recoge el mensaje broadcast emi-

tido por MyService, que significa que el servicio ha terminado, y muestra un mensaje *Toast* al usuario.

- RecipeProvider.java (Figura 7): ContentProvider que hace que la base de datos local sea accesible a través de URIs, y evita que el resto de clases se relacione directamente con MyDB.

La base de datos local no ha cambiado desde la primera entrega, ya que no hay necesidad de gestionar los usuarios ahí. La base de datos remota, sin embargo, sí tiene que hacer esto, así que está formada por dos tablas. El diagrama se ve en la Figura 8.

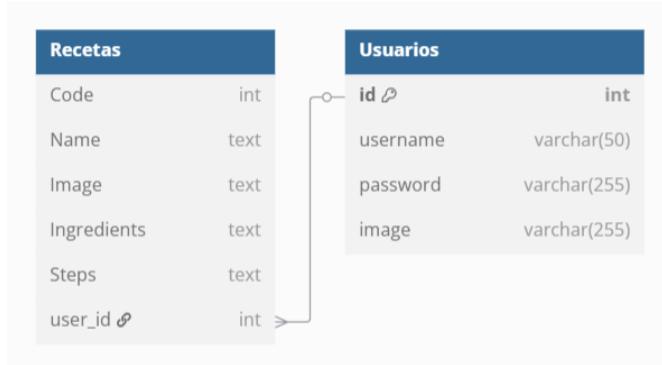


Figure 8: Diagrama de la base de datos remota.

Los diagramas de secuencia para añadir (Figura 9) y editar (Figura 10) recetas difieren un poco de los presentados en la primera versión de la aplicación, porque esta vez es necesario tener en cuenta los modos online y offline, y además, ya no se trabaja directamente con la base de datos local, sino que se usa un ContentProvider.

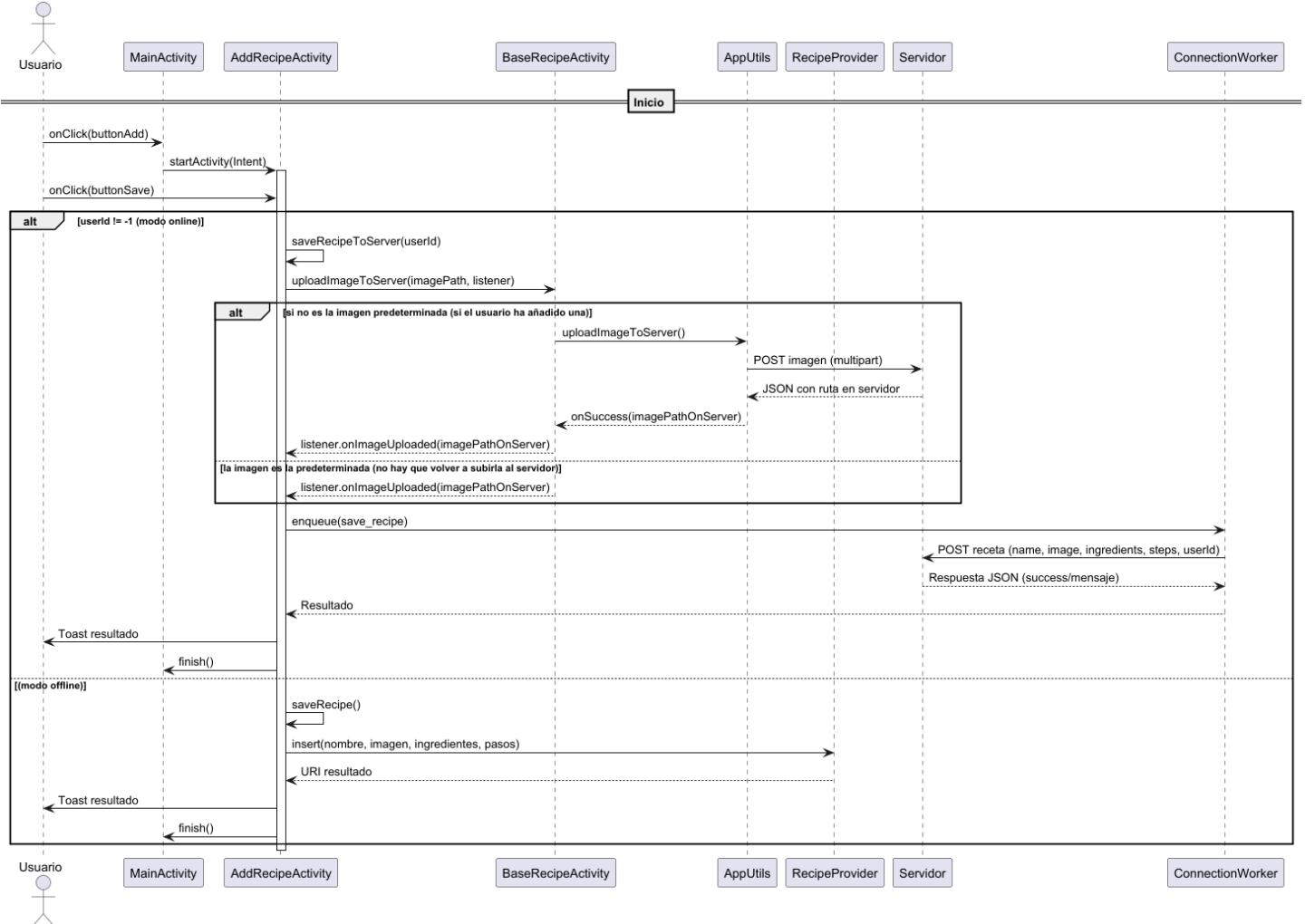


Figure 9: Diagrama de secuencia de añadir una receta.

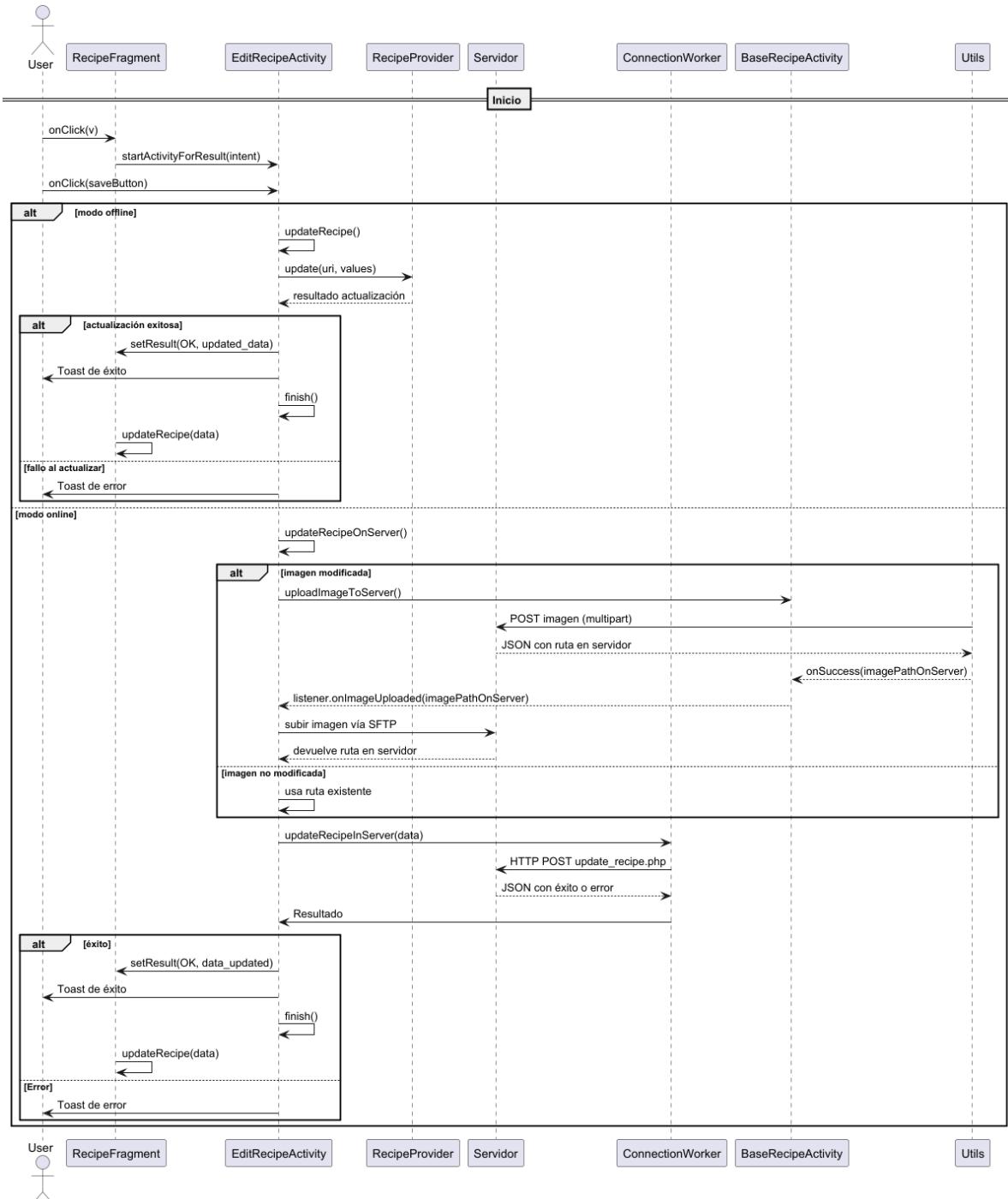


Figure 10: Diagrama de secuencia de editar una receta.

4 Manual de uso

Al iniciar la aplicación, el usuario verá una pantalla de inicio de sesión (Figura 11) donde tiene tres opciones: iniciar sesión (si ya tiene una cuenta), registrarse (si no la tiene) o entrar sin identificarse. Las dos primeras opciones constituyen el modo "online" de la aplicación, donde los datos se guardarán en la base de datos del servidor remoto, y el tercero es el modo "offline" en el que se gestiona todo en la base de datos local. Este modo offline era el que estaba disponible en la primera entrega de la aplicación.

Iniciar sesión o entrar en modo offline conducen directamente a una pantalla de menú, pero elegir la opción de registrarse conlleva pasar por otra pantalla antes de eso (Figura 11). Ahí, el usuario tendrá que especificar un nombre de usuario (único) y una contraseña. Se le pedirá que introduzca la contraseña dos veces para mayor seguridad, y una vez comprobado que se han cumplimentado todos los campos y las dos contraseñas coinciden, se creará la cuenta y se pasará al menú. En este punto, se le asignará la misma foto de perfil (predeterminada) a todos los usuarios, pero se podrá modificar más tarde.

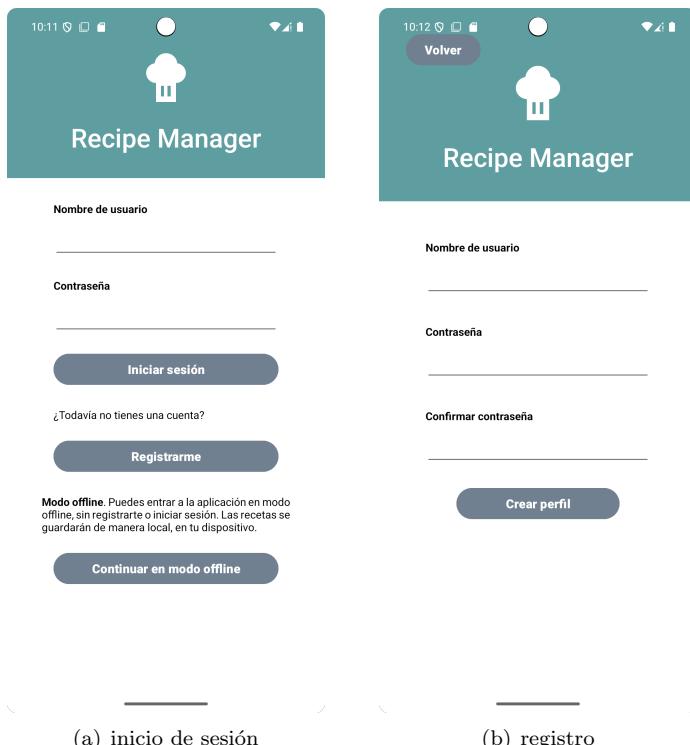


Figure 11: Pantallas iniciales.

En este menú habrá cuatro botones disponibles en el modo online, y tres en el offline (Figura 12). Los botones que conducen a las recetas, el mapa y los ajustes están habilitados en los dos modos, pero el del perfil solo será accesible en modo online. Además, en caso de estar el usuario en modo online, aparecerán su nombre de usuario y su foto de perfil en la parte central de la pantalla. Por último, el botón de "Cerrar sesión" lleva al usuario de vuelta a la pantalla de inicio de sesión en los dos casos.

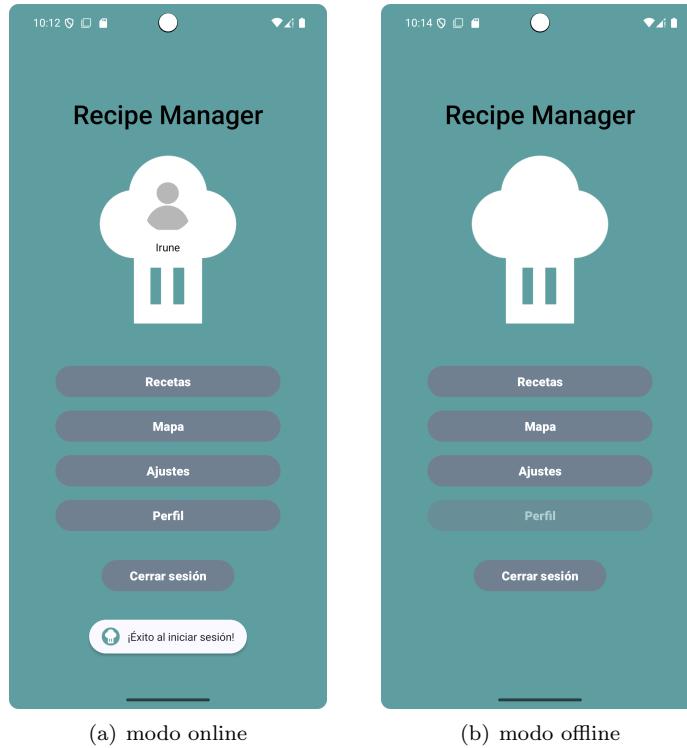


Figure 12: Pantallas de menú.

Si se pulsa sobre el botón de "Recetas", se pasará a la pantalla con la lista de recetas (Figura 13). En modo online, las recetas que se muestren serán las que estén guardadas en la base de datos remota y relacionadas con el usuario en cuestión, y en el offline, las almacenadas en la base de datos local. Si todavía no hay ninguna receta guardada, no aparecerá ninguna.

Además del botón para añadir recetas, situado en el encabezado de la pantalla, hay un nuevo botón en la parte inferior que únicamente estará habilitado en el modo online. Sirve para descargar las recetas que el usuario tenga almacenadas remotamente a la base de datos local. Al pulsarlo, aparecerá una notificación indicando que el servicio está en proceso, y cuando esté finalice, se hará saber

mediante un mensaje *Toast*. Así, si el usuario cierra sesión y entra en modo offline, verá que todas las recetas que tiene online aparecen también ahí. Durante la sincronización se verificará, además, que no se descarguen más de una vez las mismas recetas.

La pantalla para añadir nuevas recetas a la que se llega mediante el botón "+" contiene los mismo elementos que presentaba en la primera versión del trabajo y funciona prácticamente igual. Sigue siendo necesario llenar todos los campos, menos el de imagen. Simplemente, esta vez se tendrá en cuenta si el usuario está en modo online u offline, y la receta se guardará en la base de datos correspondiente.

Lo mismo pasa si el usuario decide pulsar sobre alguna receta almacenada. La pantalla que muestra los detalles se mantiene igual, así como la de editar una receta y el diálogo para borrar una, aunque claro está que los cambios se harán en la base de datos correspondiente, dependiendo de los modos online u offline.

Además, a la hora de actualizar y eliminar recetas, se gestionan los archivos de imagen para evitar que imágenes que ya no se muestran en la aplicación sigan almacenadas, tanto en el servidor web como en el almacenamiento local del dispositivo, optimizando así el espacio.

Cabe mencionar también que se ha mejorado la experiencia de usuario en modo *landscape*, y ahora estas pantallas, así como todas las demás que conforman la aplicación, pueden girarse y seguir funcionando sin ningún problema, estando todos los botones accesibles y guardando la información correctamente (Figura 14). Por ejemplo, si el usuario se encuentra en mitad del proceso de añadir una receta en modo *portrait*, ha subido ya una imagen, y rota el dispositivo a modo *landscape*, la imagen no se perderá.

Es más, en el modo *landscape* de la lista de recetas, en vez de mostrar los ingredientes y los pasos al mismo tiempo, se han añadido dos botones que permiten elegir si se muestran los ingredientes o los pasos. Esto resulta muy cómodo y visualmente más adecuado, porque se aprovecha mejor el espacio sin sobrecargar la pantalla. En la lista de recetas, se oscurece el fondo de la receta que se esté mostrando para indicar cuál de ellas está seleccionada.

Volviendo al menú principal mediante el botón "Volver", se puede acceder al mapa (Figura 15). Al entrar por primera vez, se le pedirá al usuario permiso para utilizar la ubicación. Si no acepta, solo se mostrará un marcador en la Escuela de Ingeniería de Bilbao, que es el punto por defecto. Si lo hace, y mientras se determina la ubicación del usuario, aparecerá una pantalla de carga.

Cuando desaparezca la pantalla de carga, el usuario verá marcados los restaurantes (rojo), las cafeterías (azul) y los supermercados (negro) en un radio de 1km. Si pulsa sobre uno, se mostrará el nombre del establecimiento. Al principio, se muestran los tres tipos de puntos de interés, pero se puede elegir cuáles mostrar y ocultar utilizando los tres botones correspondientes: basta con pulsar cada uno para hacer aparecer o desaparecer ese tipo de lugares.

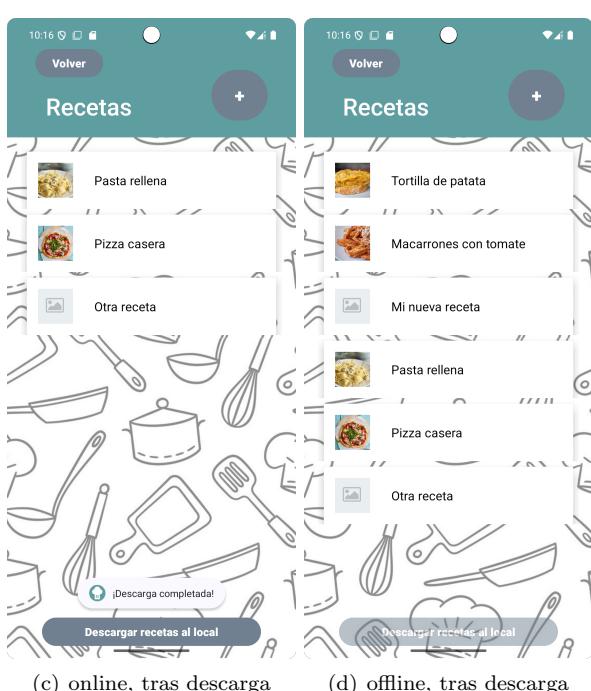
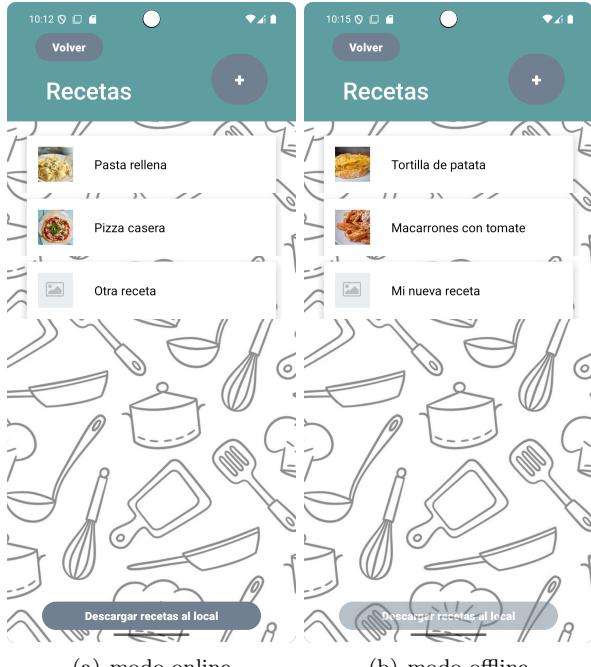


Figure 13: Las listas de recetas, antes y después de sincronizar.

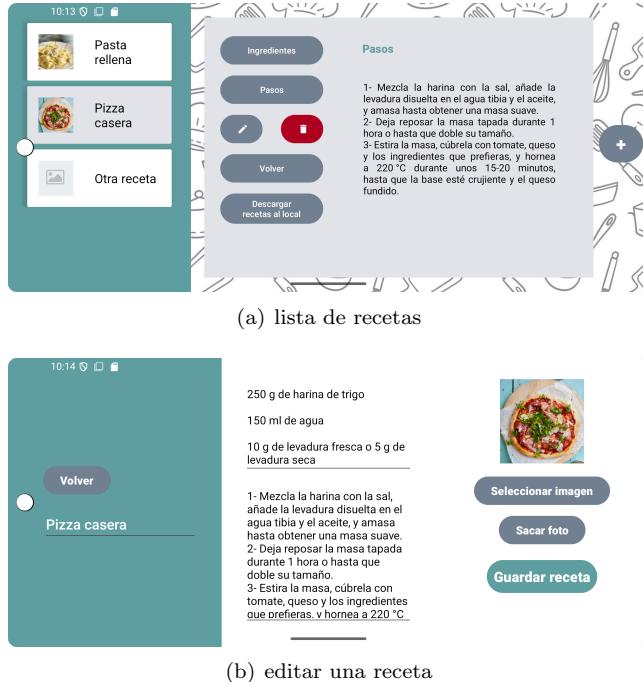


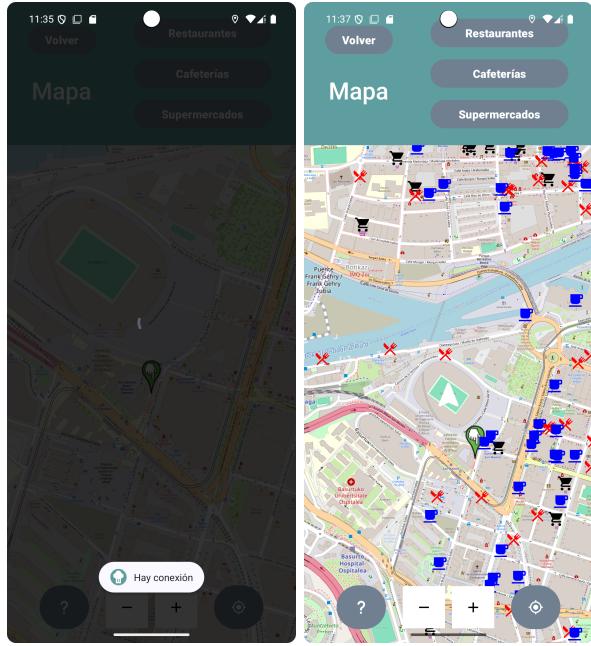
Figure 14: Pantallas en orientación horizontal.

Esos puntos de interés solo podrán marcarse si el dispositivo tiene conexión. La actividad del mapa cuenta con un método para identificar si un dispositivo que estaba sin conexión la recupera, y en ese caso actualizará inmediatamente la vista (Figura 15).

Los cuatro botones en la parte inferior del mapa permiten acceder a las instrucciones, al zoom y centrar el mapa en la ubicación actual del usuario. Las instrucciones informan al usuario de que puede mantener pulsado sobre cualquier punto del mapa, y se marcarán los puntos de interés a un radio de 1km desde ese punto. En este caso, se borrarán los marcadores iniciales (Figura 16).

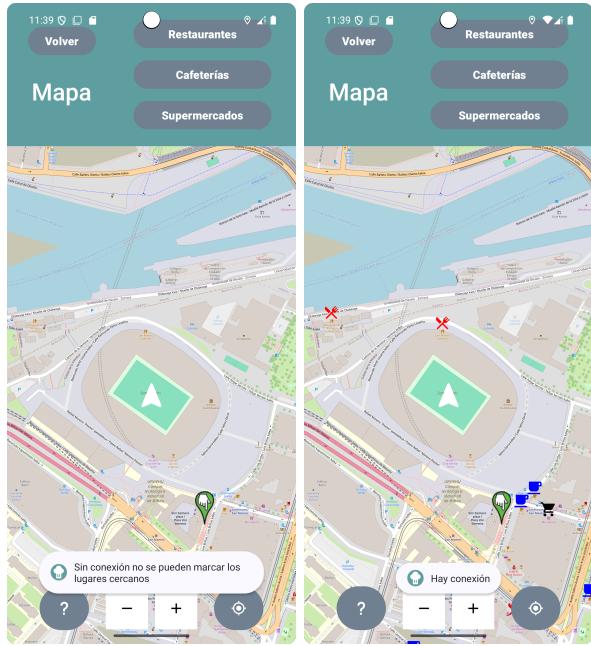
Se han implementado los botones de zoom personalizados en vez de los proporcionados automáticamente por Open Street Maps. Esto se debe a que al pulsar (un toque) sobre los botones de zoom de Open Street Maps, el sistema lo interpretaba como un pulsado largo y cargaba los puntos de interés cercanos a esos puntos, en lugar de hacer zoom.

De vuelta en el menú principal, la siguiente opción para el usuario es entrar en Ajustes. Esta pantalla no ha cambiado desde la primera entrega del trabajo. Se ha decidido enlazar las preferencias de idioma y tema solamente con el dispositivo local, en los dos modos (online y offline); es decir, las preferencias de idioma y tema no están relacionadas con los usuarios, solo con los dispositivos.



(a) pantalla de carga

(b) conexión disponible



(c) sin conexión

(d) tras recuperar la conexión

Figure 15: Gestión de la conexión en el mapa.

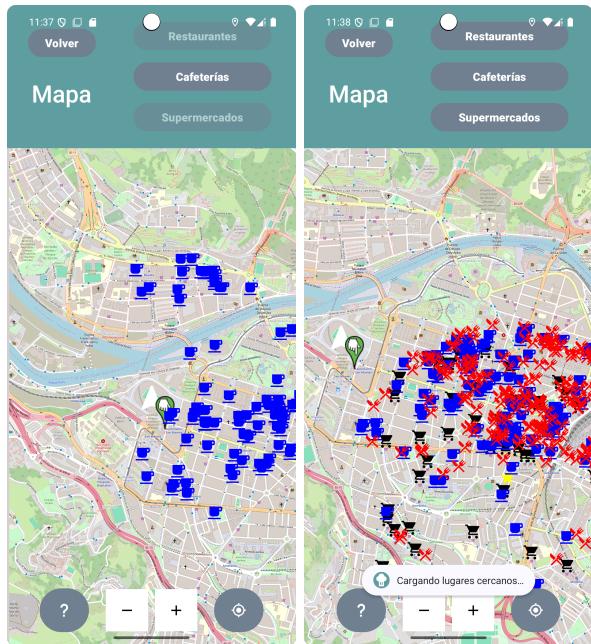


Figure 16: Mostrar diferentes elementos en el mapa.

Finalmente, el botón "Perfil" solo está habilitado en el modo online. Este llevará al usuario a una pantalla donde podrá añadir una foto de perfil o editar la que tenga actualmente, cambiar la contraseña (para lo que tendrá que verificar primero su contraseña actual) o borrar su cuenta (Figura 17).

El bloque relativo al cambio de contraseña no aparece inicialmente, sino una vez que la contraseña actual ha sido introducida: aparecerán dos campos de texto para la nueva contraseña, como en la pantalla de registro. El botón de "Borrar cuenta" mostrará un diálogo de confirmación, explicando al usuario los riesgos de hacerlo.

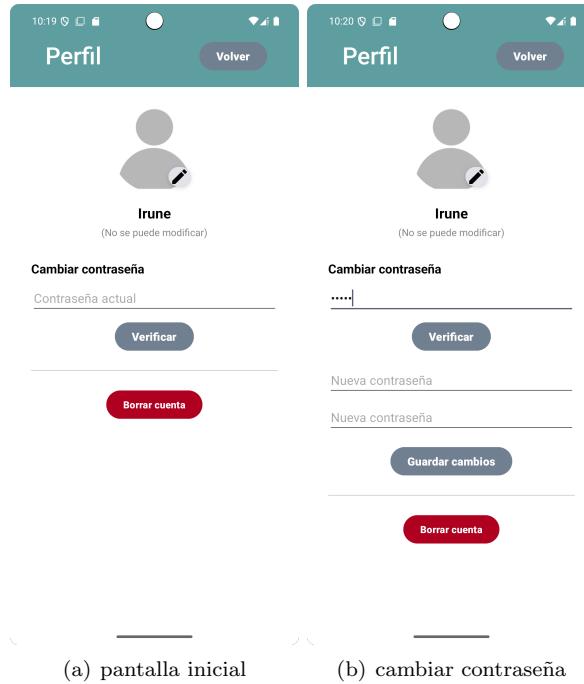


Figure 17: Pantallas del perfil.

Fuera de la aplicación, se puede mostrar el *Widget* que contiene el nombre y la imagen de la receta diaria. Esta se selecciona entre las recetas almacenadas localmente, y si no las hay, también lo indicará. Además, la aplicación enviará una notificación al usuario todos los días sobre las 19:00 (Figura 18).

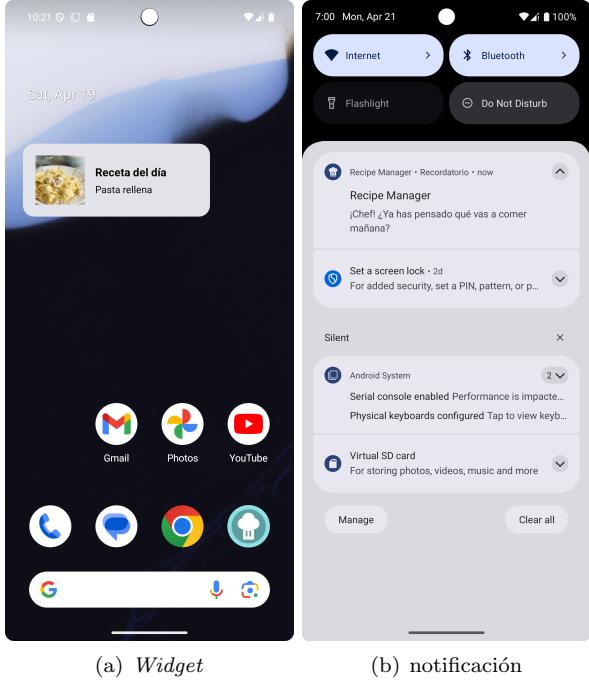


Figure 18: Elementos fuera de la aplicación.

5 Dificultades encontradas a la hora de realizar el trabajo

Para este proyecto, me llevó bastante tiempo organizar mis ideas respecto al diseño y pensar cómo integrar los nuevos requisitos mínimos de una manera que tuviera sentido. Lo primero que decidí fue añadir un menú que posibilitara acceder a más funcionalidades, y a partir de ahí me fue más fácil organizar todo. Además, tener los modos online y offline (y no solo el online, como estuve pensando al comienzo) me ha permitido implementar elementos como el ContentProvider o el servicio en primer plano (para la sincronización) de una manera coherente.

Por otro lado, los principales inconvenientes técnicos que me han surgido han tenido que ver con los servidores SFTP y MySQL.

Uno de los problemas que más complicaciones me causó fue la subida de archivos de imágenes al servidor web. Al principio no entendía por qué no se hacía correctamente, pero luego me di cuenta de que era un problema de permisos. Me costó identificar el error y entender por qué sucedía, aunque por suerte, solucionarlo no fue tan difícil.

Otra dificultad que tuve también tuvo que ver con este proceso de subir archivos al servidor. En las actividades AddRecipeActivity y EditRecipeActivity, cuando el usuario añadía una imagen, esta se subía correctamente al servidor, pero la ruta que se guardaba en el campo *Image* de la base de datos remota no era la adecuada. Finalmente, entendí que la ruta que se estaba guardando era el valor inicial de esa variable, porque cuando se insertaba la nueva fila en la base de datos, la imagen todavía no había terminado de subirse al servidor web, por lo que no daba tiempo de conseguir la nueva ruta aquí. Por lo tanto, tuve que implementar un *listener* para que avisara cuando la nueva imagen se hubiera subido y se hubiera conseguido la ruta correcta, para que se insertaran entonces los datos en la base de datos.

A la hora de gestionar la respuesta del servidor al inicio de sesión, también tuve un pequeño problema, porque si el inicio de sesión no era correcto, la respuesta (el *JSONObject*) no devolvía el campo *user_id* que yo recogía en mi clase ConnectionWorker. Lo arreglé añadiendo un bloque *try-catch*, y asignando el valor -1 al *user_id* en ese caso. Después de eso me di cuenta de que podía gestionar más situaciones parecidas con *JSONException*.

Implementar el mapa también me llevó tiempo, pero añadir el resto de elementos adicionales no fue tan complicado.

6 Bibliografía

Apuntes de la asignatura Desarrollo Avanzado de Software 2025.

<https://egela.ehu.eus/login/index.php>

ChatGPT-4o, ChatGPT-4o mini.

<https://chatgpt.com/>

Stack Overflow.

<https://stackoverflow.com/>

Overleaf LaTeX Documentation.

<https://www.overleaf.com/learn>

Material Design Icons - Icon Library - Pictogrammers.

<https://pictogrammers.com/library mdi/>

Android API Reference — Android Developers.

<https://developer.android.com/reference>

PHP: password_hash.

<https://www.php.net/manual/es/function.password-hash.php>

PHP: password_verify.

<https://www.php.net/manual/es/function.password-verify.php>

Overpass API/Overpass QL.

https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL