UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Desarrollo de una plataforma móvil para el envío de paquetes

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: David Zamora Villalobos

Tutor: Pedro José Valderas Aranda

Curso 2016-2017

Desarrollo de una plataforma móvil para el envío de paquetes

Resumen

Easytrans es una aplicación para *smartphone*, que permite a los usuarios solicitar el transporte de un paquete desde su móvil. Sin necesidad de moverse de casa, todos los trámites se realizan desde la aplicación.

Se encuentra disponible actualmente para sistemas operativos Android. Se ha desarrollado nativamente, lo cual implica un aprovechamiento máximo de las funciones del dispositivo.

Permite el registro de usuarios, tanto clientes, como repartidores. Integra una solución de notificaciones *push*. Gracias a esto se tiene a sus usuarios al tanto de las últimas novedades acerca de sus envíos, tanto si eres remitente, como si eres destinatario. Además de ofrecer la funcionalidad de chat, la cual permite a los dos interesados mantener el contacto en todo momento ante cualquier circunstancia sin ningún tipo de intermediación.

La infraestructura que esta aplicación utiliza está íntegramente basada en la plataforma Firebase de Google. Esto permite un alto grado de disponibilidad y una velocidad notable, lo cual mejora en gran medida la experiencia del usuario.

Palabras clave: smartphone, móvil, paquetería, android, push, chat, firebase.

Abstract

Easytrans is a smartphone application, which allows users to request the transport of a package within their mobile.

It is currently available for Android operating systems. It has been developed natively, which implies a maximum use of the device functions.

It allows the registration of users, both customers and workers. Integrates a push notifications solution. Thanks to this you have your users up to date of the latest news about your shipments, whether you are a sender, or a recipient. Moreover, it offers chat functionality, which allows both involved users to stay in touch in any circumstance without any intermediation.

This application infrastructure is entirely based on the Google Firebase platform. This allows a high degree of availability and a remarkable speed, which greatly improves the user experience.

Keywords: smartphone, mobile, parcels, android, push, chat, firebase.

Resum

Easytrans és una aplicació per a *smartphone*, que permet als usuaris sol·licitar el transport d'un paquet des del seu mòbil. Sense necessitat de moure's de casa, tots els tràmits es realitzen des de l'aplicació.

Es troba disponible actualment per a sistemes operatius Android. S'ha desenvolupat nativament, la qual cosa implica un aprofitament màxim de les funcions del dispositiu.

Permet el registre d'usuaris, tant clients, com a repartidors. Integra una solució de notificacions push. Gràcies a açò es té als seus usuaris al tant de les últimes novetats sobre els seus enviaments, tant si eres remitent, com si eres destinatari. A més d'oferir la funcionalitat de xat, la qual permet als dos interessats mantenir el contacte en tot moment davant qualsevol circumstància sense cap tipus d'intermediació.

La infraestructura que aquesta aplicació utilitza està íntegrament basada en la plataforma Firebase de Google. Açò permet un alt grau de disponibilitat i una velocitat notable, la qual cosa millora en gran manera l'experiència de l'usuari.

Paraules clau: smartphone, mòbil, paqueteria, android, push, chat, firebase.

Tabla de contenidos

Desarrollo de una plataforma móvil para el envío de paquetes

1. Introducción

1.1 Introducción

1.1.1 Propósito

El propósito de este trabajo es facilitar la labor de la entrega de paquetería entre usuarios sin necesidad de realizar trámites en las oficinas de la empresa de paquetería.

Se consigue mejorar la interacción entre remitente, destinatario y repartidor, gracias a las funcionalidades que ofrece entre ellos.

De este modo es posible estar al tanto instantáneamente del estado de tus encargos desde tu móvil sin necesidad de interacción externa.

1.1.2 Declaraciones de alcance y objetivos

El alcance para la primera versión de este proyecto, es la implementación para Android, sin desestimar la posibilidad de una ampliación a otros sistemas operativos en un futuro no muy lejano.

Con la cuota de mercado que posee Android en España, superior al 90%, hay unas altas expectativas de una adaptación rápida y eficaz en la población española.

1.1.3 Definiciones, siglas y abreviaturas

En este documento se van a utilizar una serie de abreviaturas y siglas, las cuales van a describirse debajo:

- **Firebase**: Plataforma de Google que ofrece funcionalidades de servidor en tiempo real.
- **Mockup**: Esquema o dibujo del diseño de una aplicación que se usa para aplicarse al diseño final de la aplicación.
- **Push**: Notificación asíncrona que se entrega al dispositivo en el momento en que esté disponible.
- **FCM**: *Firebase Cloud Messaging*. Conjunto de servicios de Google que ofrece en su plataforma Firebase, los cuales se encargan de entregar notificaciones push.
- **JSON**: *JavaScript Object Notation*. Notación de objetos Javascript, que permite representar objetos de una manera legible.
- **SQLite**: Sistema gestor de bases de datos relacionales que proporciona Android por defecto.
- **API**: *Advanced Programming Interface*. Proporciona una interfaz de abstracción de implementación sobre un servidor y sus bases de datos. Suele contener los servicios web o *web services*.
- **HTTP**: *HiperText Transfer Protocol*. Es el protocolo más famoso de transmisión de información a través de la red. Fue desarrollado por el *W3C* y la *IETF*, colaboración que culminó en 1999 con la publicación de una serie de *RFC*.
- **W3C**: *World Wide Web Consortium*. Es un consorcio internacional que genera recomendaciones y estándares que aseguran el crecimiento de la red de Internet a largo plazo.

- **IETF**: *Internet Engineering Task Force*. Es una organización internacional abierta de normalización, que tiene como objetivos el contribuir a la ingeniería de Internet, actuando en diversas áreas, como transporte, encaminamiento, seguridad. Se creó en los Estados Unidos, en 1986. Es mundialmente conocido porque se trata de la entidad que regula las propuestas y los estándares de Internet, conocidos como *RFC*.
- **RFC**: Request For Comment: Publicaciones oficiales de *IETF* que describen diversos aspectos del funcionamiento de Internet y otras redes de computadoras, como protocolos, procedimientos, etc. y comentarios e ideas sobre estos.
- **Servicio web**: Es un servicio que proporciona una *API*, diferenciado por una URL distinta para cada caso. Los más famosos son los servicios web *REST*.
- **REST**: Es un estilo de arquitectura software para sistemas distribuidos como la Internet.
- **Cloud**: Es el término que se utiliza para referirse a los servicios que se encuentran en algún servidor, del cual tienes una abstracción sobre cómo funciona, pero que ofrece la funcionalidad descrita a través de la red.
- **SDK**: *Software Development Kit*. Esto representa el conjunto de herramientas que se facilitan sobre una plataforma para permitir a los desarrolladores hacer uso de sus funcionalidades. Normalmente va a compañado de una documentación de su *API*, con fines de facilitar las tareas de programación.
- **IDE**: *Integrated Development Environment*. Es la notación que se le da al software que integra los SDK de la plataforma sobre la que estás desarrollando.
- **Funciones core**: Las funciones *core* son aquellas comunes a un conjunto de librerías, que comparten funcionalidad, y que son las mínimas que se requieren para funcionar.
- **Tupla**: Una tupla básicamente es un conjunto de dos elementos que tienen algún tipo de relación entre sí. En resumen, dos elementos.
- **Token**: Cadena de texto que identifica inequívocamente un dispositivo para que reciba una notificación desde el servidor *push* de Google.
- **DCU**: Diseño centrado en el usuario. Filosofía de diseño que tiene por objeto la creación de productos que resuelvan necesidades concretas de sus usuarios finales, consiguiendo la mayor satisfacción y mejor experiencia de uso posible con el mínimo esfuerzo de su parte.
- **SaaS**: *Software as a Service*. Se trata de un modelo de distribución de software donde el soporte lógico y los datos que maneja se alojan en servidores de una compañía TIC, a los que se accede vía Internet desde un cliente. La empresa proveedora TIC se ocupa del servicio de mantenimiento, de la operación diaria y del soporte del software usado por el cliente.
- **TIC**: Tecnologías de la Información y la Comunicación.
- **CDN**: *Content Delivery Network*. Es una red superpuesta de computadoras que contienen copias de datos, colocados en varios puntos de una red con el fin de maximizar el ancho de banda para el acceso a los datos de clientes por la red. Un cliente accede a una copia de la información cerca del cliente, en contraposición a todos los clientes que acceden al mismo servidor central, a fin de evitar embudos cerca de ese servidor.
- **Target**: Se refiere a un determinado perfil de persona, que cumple unas determinadas características que tienen varios en común.
- **Mockup***:* Modelo a escala o tamaño real de un diseño o un dispositivo, utilizado para la demostración, evaluación del diseño, promoción, y para otros fines. Cuando también incluye demostración de funcionalidad se le denomina prototipo.

- **Payload**: Es aquello que se define en cualquier tipo de comunicación que requiera un intercambio de información como la carga o el contenido del mensaje.
- **Listener**: Un escuchador es un mecanismo mediante el cual se obtienen callbacks cuando sucede el evento sobre el que se está escuchando.
- **Callback**: Son funciones a ejecutar en un futuro. Un usuario utilizaría un *callback* para recibir un resultado una vez se ha obtenido, asumiendo que obtener el resultado es una tarea no es instantánea y que requiere que esta se complete antes.
- **CRUD**: *Create Read Update Delete*. Son las 4 operaciones básicas sobre las que se basa un sistema de gestión de bases de datos. Creación, lectura, actualización y borrado de elementos.
- **Snippet**: Pequeño fragmento de código fuente, en principio reutilizable, que en este proyecto se utilizará a modo de ejemplo. Proporcionará una visión general, con un nivel de abstracción sobre el nivel de detalle técnico que el código completo conlleva.
- **Middleware**: Es aquel software intermediario capaz de conectar dos sistemas separados entre sí.

1.1.4 Referencias y aserciones

En este proyecto se tienen que utilizar una gran cantidad de recursos para que todos los servicios que ofrece sean lo más correctos y tengan la mayor disponibilidad y accesibilidad posible.

Por ello se han utilizado referencias bibliográficas, las cuáles serán visibles más adelante. Tanto referencias de sitios web, como referencias de libros de texto.

Además, también se han usado las premisas con las que se trabaja en entornos de paquetería, para ofrecer lo que el usuario necesita en el momento adecuado.

Amazon, la empresa líder en el sector del comercio electrónico, ha servido como referencia para observar cuales son los procesos habituales de entrega, las funcionalidades que ofrece, y también las que no ofrece. Todas aquellas que el usuario echa en falta.

A posterior se expondrá una encuesta realizada durante dos días a los viandantes de Valencia. Se revelarán los resultados más adelante.

1.2 Descripción general

Aquí se va a hacer una descripción formal del proyecto, junto con una descripción un poco más técnica acerca de las funcionalidades ofrecidas y de sus detalles.

1.2.1 Perspectiva del producto

El nombre de la aplicación es **Easytrans**. Que hace una mezcla entre las palabras ***easy***, que significa fácil en inglés, y **trans**, que viene de la palabra transporte.

Easytrans se presenta ante el público como una solución móvil para mantenerte al tanto del estado de sus envíos. Mantiene la conexión entre sus clientes y los encargos a través de notificaciones push. Mantiene el contacto entre los involucrados facilitándoles un servicio de mensajería instantánea, el cual les permite chatear para solucionar cualquier incidencia que pueda surgir.

Figura 1. Logo de la aplicación.

1.2.2 Funciones del producto

Las funciones del dispositivo han sido nombradas anteriormente, pero van a ser enumeradas ahora, muchas de estas funciones se proveen gracias a la implementación de Firebase y sus conjuntos de características que implica. Estas se comentarán más adelante, ahora describiremos las distintas funcionalidades:

- **Función de registro de usuarios**: En esta función se desea que los nuevos clientes de la aplicación, se registren con un rol definido desde el principio. Deben poder registrarse tanto como usuario normal como usuario trabajador. En función de dicho rol, se ofrecerán menús y ventanas distintas en la aplicación a cada tipo de usuario.
- **Función de inicio de sesión**: En el inicio de sesión los usuarios tendrán la posibilidad de acceder tanto al nombrado registro de usuarios, como identificarse frente al sistema y acceder al menú principal de la aplicación. Otra función a proveer es la posibilidad de activar la opción de iniciar sesión automáticamente, la cual te debe permitir recordar tu usuario en cada inicio de sesión.
- **Función de menú principal**: En el menú principal se ofrecerá la interfaz principal de la aplicación, la cual variará como se ha comentado dependiendo el tipo de usuario. Además de la interfaz, también debe tener una barra lateral, que muestra información actualizada en tiempo real acerca de la información del usuario que ha iniciado sesión. Esta característica se describirá en profundidad más adelante.
- **Función de cambio de perfil del usuario**: Esta función debe proporcionar la posibilidad de tener un perfil para ser identificado en el sistema por los demás usuarios. Cada usuario debe tener una imagen que seleccionará a partir de una colección de avatares predefinidos, y la cual podrá cambiar siempre que lo desee. También podrán cambiar el correo asociado a su cuenta, la cual les servirá para iniciar sesión. Por último, debe ser posible cambiar la dirección del usuario, la cual se usará para calcular los envíos que realicen.
- **Función de mapa**: La función del mapa se podrá usar tanto para la selección de la dirección del usuario en su perfil, como para la selección del destino de un envío. Este implementa dos funciones internas, la del cálculo del nombre de la dirección seleccionada, como de la búsqueda de una dirección introducida por el usuario. A nivel interno almacenará la dirección como coordenadas de latitud y longitud. Para los trabajadores aparecerá un conjunto de marcadores con colores diferenciados, los que estén asignados, y los que estén sin asignar.
- **Función de solicitud de encargo**: En este formulario se debe poder introducir la información para crear un nuevo encargo, que debería publicarse en la base de datos como un encargo sin asignación, que aparecerá en el mapa para los trabajadores. Los trabajadores podrán postularse a ellos. Esto también añade una función interna para calcular el precio que estipula la aplicación, para un encargo. El cálculo se realiza una estimación de la distancia entre la recogida y la entrega. Se definirá la función matemática utilizada.

- **Función de listado de encargos relacionados**: En este se desea que proporcione una interfaz para representar de manera rápida y concisa la información acerca de un envío. En la que se muestre la información de remitente, destinatario, dirección de envío, y una muestra del estado actual del envío con cinco posibles variantes.
    - En preparación: El producto está preparándose para poder recogerlo.
    - En tránsito: El producto se encuentra en tránsito hacia el almacén del repartidor.
    - En almacén: El producto se almacena en las instalaciones del repartidor, preparándose para el proceso de entrega.
    - En reparto: El repartidor asignado a tu producto se encuentra en proceso de reparto. Esto implica que es cualquier momento puede llamar a la puerta para entregar el producto.
    - Entregado: El producto ya se encuentra en poder del destinatario
- **Función de búsqueda de encargos cercanos**: Esta función utilizaría el mapa para mostrar los encargos que tiene un trabajador en su área cercana. Utilizaría el color para diferenciar los asignados de los no asignados.
- **Función de chat con un usuario**: Desde aquí se busca que sea posible solucionar cualquier incidencia, u obtener cualquier tipo de información que pudiese facilitar el trabajador, repartidor, o receptor. También tener la posibilidad de almacenar los mensajes recibidos localmente en una base de datos SQLite local. Se utilizaría tan solo para almacenar los mensajes recibidos, hayan sido representados o no, con fines de mostrarlos ordenados y almacenarlos en caso de no ser representados en el mismo instante. Lo mejor sería usar una tabla distinta para cada usuario con fines de optimización en las consultas. Más adelante se describirá de manera técnica los esquemas y las definiciones de la base de datos necesarios para poder realizar esto.
- **Función de notificaciones push**: Mediante esta función se deben poder comunicar las distintas aplicaciones en los distintos dispositivos. Actualmente solo se soportará un dispositivo por usuario. Esta aplicación usará el paso de mensajes para gestionar la interacción entre los usuarios y la notificación a los usuarios en los momentos que se generen los cambios.

1.2.3 Características del usuario

Los dos tipos de usuarios han sido claramente diferenciados, y por lo tanto vamos a proceder a diferenciar las distintas características de cada tipo de usuario:

Ambos usuarios:

Hay una serie de funcionalidades que deben ser comunes a los dos tipos de usuarios.

La funcionalidad de perfil es común a ambos, con sus características de cambio de avatar, dirección y demás.

La funcionalidad de chat también la comparten ambos dos.

La funcionalidad de seguimiento de los envíos.

Usuario cliente:

Un usuario cliente contará con la posibilidad de generar encargos

Usuario trabajador:

Un usuario trabajador contará con la posibilidad de ver los encargos cercanos y postularse a ellos.

También debe contar con la opción de ir marcando las actualizaciones del envío del paquete para que les aparezca a los clientes.

1.2.4 Restricciones

La elección de usar Firebase como infraestructura central, traerá consigo una serie de restricciones inherentes al propio diseño de la misma.

Firebase usa como sistema de gestión de base de datos un modelo propio mediante el cual, todas las consultas que se hagan deben hacerse siguiendo una metodología especial. Esta metodología se tratará en detalle más adelante.

Esto provoca el no tener la posibilidad de hacer las típicas consultas a la base de datos, por un lado, puede resultar una ventaja por el tiempo ahorrado implementando servicios, pero no es así, dado que es necesario implementar la lógica de control de los datos obtenidos dentro de la aplicación.

Dado que Firebase es un sistema externo, la disponibilidad y fiabilidad que este ofrece es un gran punto a favor, ya que te quita la preocupación de la administración sobre el servidor y las versiones del software que este use.

Este es un punto muy importante, porque es todo un reto solucionar un problema físico en un servidor cuando el servidor es externo o no se encuentra accesible por el administrador de sistemas. Contar con un sistema no administrado, que siempre funciona como debe, siempre está disponible, es gratis y no tiene ningún tipo de limitación es un fuerte punto a favor.

La única restricción más importante, es que debido a la carga de librerías que tiene, necesita un tiempo extra para cargar la aplicación al inicio.

Por otro lado, es inherente la necesidad de una conexión a internet en el dispositivo para poder acceder al servicio.

1.2.5 Contenido de la memoria

En los consiguientes puntos de esta memoria se van a tratar los siguientes temas:

- **Contexto tecnológico**: Aquí se explicará brevemente el conjunto de herramientas que se encuentran disponibles. Se describirá la funcionalidad ofrecida por las plataformas en las que se basa la aplicación.
- **Estado del arte**: En este apartado describiremos que se encuentra actualmente en el mercado, analizando detenidamente que funcionalidades ofrece cada solución externa, y cuales no ofrece. Este análisis ayuda a la redacción del cuestionario que se ha utilizado para obtener la opinión de los clientes finales.

- **Metodología**: Todo proyecto que desee un mínimo nivel de calidad en el proceso, debe haber seguido una metodología para su diseño. En este caso, se plantea la metodología empleada en este proyecto. Se describirá su filosofía, y sus características más importantes.
- **Arquitectura**: En este apartado se detalla cual es el resultado obtenido de aplicar la metodología escogida. Este es el resultado sobre el que se basa el diseño y la implementación de la solución.
- **Análisis de necesidades**: Aquí se puede ver un análisis acerca del proceso que se sigue, incluyendo las metodologías. Haciendo un fuerte hincapié en los cuestionarios, su diseño, su obtención, y su aplicación. Además, el modelo de Persona obtenido de aplicar dichos formularios. En este punto no se detallarán los formularios.
- **Diseño**: En este punto se incluyen los diseños obtenidos finales. Los artes que se aplicarán a la aplicación. Se confirma el uso de los *mockups* generados para aplicar al diseño de la aplicación. El objetivo de esto es ayudar a seguir una línea equivalente en todo el proyecto, unificando recursos y diseño. Dado que los *mockups* son primeras versiones del diseño, durante el proceso de implementación ha habido decisiones y otros motivos que han hecho cambiar el diseño inicial. Estos cambios se han realizado siempre para añadir calidad, nunca restarla.
- **Implementación**: Este es el punto en que se habla de los detalles de diseño a más bajo nivel. Contiene ejemplos de código significativo a través del cual se ha realizado la mayor parte de la aplicación. Se dan detalles de estos pequeños fragmentos de código explicando su funcionalidad desde una perspectiva de alto nivel sin la necesidad de conocer en detalle la completitud del proyecto. De todos modos, el código fuente del proyecto se encuentra publicado en GitHub. Con ponerse en contacto con alguna persona autorizada sería posible encontrar el repositorio y utilizarlo como fuente de información.
- **Conclusiones**: Se evalúa que punto ha alcanzado el proyecto tras todo el proceso de implementación, apoyados sobre los objetivos iniciales del proyecto. Por último, se trata una opinión subjetiva del autor acerca de las expectativas y logros alcanzados por el proyecto, junto a las impresiones del mismo.
- **Bibliografía**: Este es el punto en que se citan todas las fuentes de información utilizadas en este proyecto. Tanto para realizar esta memoria, como para realizar la aplicación.
- **Apéndice**: Este punto es el que contiene la documentación específica anexa al proyecto, que no tiene lugar dentro de ningún apartado, por extensión, o por diferencia temática.

2. Contexto tecnológico

2.1 Herramientas utilizadas

En este proyecto se hace un amplio uso de herramientas de desarrollo de Android y de SQLite.

Android Studio ha sido el *IDE* utilizado para realizar toda la aplicación, el cual ofrece todas las funcionalidades de diseño e implementación para tu aplicación. Las herramientas que integra tienen un corrector sintáctico, el cual lee conforme vas escribiendo, conocido como Lint. Lint realiza comprobaciones sintácticas y semánticas para evitar que el desarrollador comenta errores a la hora de escribir código que no es posible que suceda lógicamente.

Trae consigo integrado en la última versión del software el *SDK* de Android para poder hacer uso de toda la *API* de este sistema operativo. Desde su herramienta *SDK Manager* se puede descargar todas las versiones de Android y programar sobre ellas. Incluye también el *Android Emulator*, a través del cual puedes ejecutar virtualmente cualquier versión de Android desde el ordenador.

Además, se ha hecho uso del *Android Material Design Icon Generator*, el cual ha sido usado para generar pequeños iconos de la aplicación siguiendo los patrones establecidos por Google del *Material Design*.

Se ha intentado someter el proyecto a una serie de test de mejora de calidad del software a través del software SonarQube, pero no ha podido realizarse por dos motivos. El primero honradamente ha sido por el desconocimiento del funcionamiento de la herramienta, y el segundo porque el tiempo ha venido justo para poder terminar los cambios necesarios en el proyecto.

El software SonarQube hubiese sido un buen método para valorar la calidad del código y el nivel de fallos que tiene, dado que es una herramienta de análisis a partir de casos de prueba.

El programa que se ha usado para la base de datos local, de la cual se hablará más adelante, ha sido SQLiteManager, para las tareas de depuración.

SQLiteManager es un potente sistema de gestión de base de datos para bases de datos sqlite, que combina una interfaz fácil de usar con una velocidad fulgurante y funciones avanzadas. Además, permite gestionar de manera gráfica las bases de datos y tablas locales que albergue. De este modo se puede observar gráficamente y de manera más cómoda y representativa su contenido.

Además de todo esto, la aplicación hace uso de la *API* pública de FCM, la sucesora de la antigua plataforma GCM, *Google Cloud Messaging*. Por lo tanto, ha sido necesario hacer depuración sobre las peticiones HTTP que estos generaban. En este proyecto se ha tomado la decisión de hacerlo a través de la aplicación Postman.

La aplicación de **Postman** fue creada como un proyecto secundario y fue presentada oficialmente por primera vez en octubre de 2012. Creció rápidamente para convertirse en una de las aplicaciones más populares en la Chrome Store y expandirse a aplicaciones nativas para Mac, Windows y Linux. A día de hoy cuentan con más de 3 millones de desarrolladores que utilizan sus conjuntos de aplicaciones en todo el mundo. En 2014, comenzamos a hacer hincapié sobre la aplicación gratuita y desarrollaron características más extensas para un producto ofrecido como *SaaS*, que se ha convertido en Postman versión *Pro* y *Enterprise*.

2.2 Plataformas usadas

2.2.1 Firebase

La principal plataforma utilizada en el proyecto es Firebase. Firebase ofrece un ecosistema muy completo con una funcionalidad estupenda para desarrolladores.

Figura 3. Logo identificativo de Firebase.

Firebase se planteó como una plataforma de pago por uso. Hoy en día cuenta con:

- Un plan gratuito con límites de uso de todas sus herramientas.
- Un plan de cuota fija mensual, el cual aumenta los límites gratuitos.
- Un plan de pago por uso, pagas lo que consumas.

Cuenta con los siguientes servicios *cloud*:

2.2.1.1 Analytics

Un servicio de analíticas integrado que proporciona Firebase por defecto con solo integrar sus funciones *core*. Además, se permite la implementación de unas analíticas más complejas para unos estudios más completos sobre el uso que le dan los usuarios a la aplicación que analizas.

2.2.1.2 Authentication

Los servicios de autenticación proporcionan un mecanismo que identifique a los usuarios y proteja el sistema de los usuarios malintencionados o no identificados.

Ofrece funcionalidades de cambio de datos, tanto de la información del usuario, como del cambio de contraseña.

2.2.1.3 Realtime Database

Esta es la función más interesante de Firebase, y de la que más uso se ha hecho en este proyecto.

Es una base de datos **no** relacional, basada en una estructura JSON, que tiene una raíz, y de la que cuelgan unos hijos.

Como cualquier JSON, todos los campos son tuplas clave-valor, en la cual un valor puede ser cualquier tipo de objeto, desde un tipo básico, pasado por una colección, hasta incluso otro objeto JSON.

2.2.1.4 Storage

Las funciones de almacenamiento de Firebase las provee este servicio, el cual te permite almacenar ficheros en un servidor de datos, del cual a posterior se pueden recuperar como cualquier servicio de almacenamiento masivo. Se utiliza tanto para almacenar objetos como otros tipos de datos estáticos.

2.2.1.5 Hosting

El hosting es la característica de un servidor de hospedar una página web. Este es el módulo que te ayuda a desplegar las páginas web que necesites.

Simplifica el alojamiento web estático con herramientas creadas específicamente para las aplicaciones web modernas. Cuando subes tus recursos web, ellos los envían automáticamente a su *CDN* global y les asignan un certificado SSL gratuito, de manera que los usuarios disfruten una experiencia segura, confiable y con poca latencia sin importar su ubicación.

2.2.1.6 Cloud Functions

Las funciones *cloud* te permiten ejecutar automáticamente código en segundo plano en respuesta a eventos disparados por características configuradas en Firebase y a peticiones HTTP. Tu código se almacena en el entorno de Google y se ejecutan en un entorno controlado. No hay necesidad de gestionar y escalar tus propios servidores.

Esta es la apuesta de futuro de Google para apoyar las funcionalidades que ofrece su Cloud Platform.

2.2.1.7 Cloud Notifications

Cuando se desea enviar una notificación a usuario en el momento de suceder algún evento, por alguna programación, o por algún disparador configurado, ésta es la herramienta que se encarga de permitírtelo.

Como se ha dicho, permite configurar disparadores, programarlas, o lanzarlas manualmente.

El uso de estas funciones requiere que las notificaciones push estén correctamente configuradas en el dispositivo, sino no aparecerán las notificaciones. Aunque el motivo principal también es que no se dispondrá de *token* de notificación.

2.2.1.8 AdMob

AdMob es la plataforma de Google que se encarga de gestionar la publicidad y los anuncios que pongas en tu aplicación con fines de monetización por la utilización de la misma.

Esta plataforma es de vital importancia, ya que es la plataforma principal de ingresos de una posible nueva aplicación con condiciones de financiación muy bajas. Podría ser la manera de financiar los costes que pueda generar.

2.2.1.9 Otras

Además de todas las importantes funciones anteriormente comentadas, también cuenta con otros servicios:

- **Test Lab**, para la realización de tests automáticos sobre las aplicaciones.
- ***Performance***, un servicio que le ayuda a obtener información sobre las características de rendimiento de las aplicaciones.
- ***Crash Reporting***, para tener centralizados los errores generados por la aplicación debido a errores generados por excepciones en el código. Esta función facilita las tareas de depuración cuando la aplicación se encuentra ya publicada en producción en la tienda.

Cabe destacar que todas las funciones proporcionadas por Firebase son compatibles tanto con Android como con iOS.

Es posible utilizar una gran parte del software de Firebase en cualquier sistema operativo móvil en realidad.

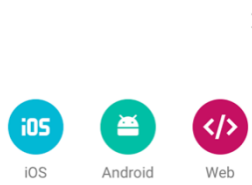


Figura 4. Integración multiplataforma.

Firebase nativamente se puede implementar tanto en Android como en iOS, pero además ofrece la posibilidad de integrarse también en una aplicación web a través de tecnología Javascript, como se representa en la Figura 4.

Como las aplicaciones híbridas están tan en auge, a través de Apache Cordova, y Firebase para Javascript sería posible usarlo en todos los S.O.

2.3 Lenguajes de programación implicados

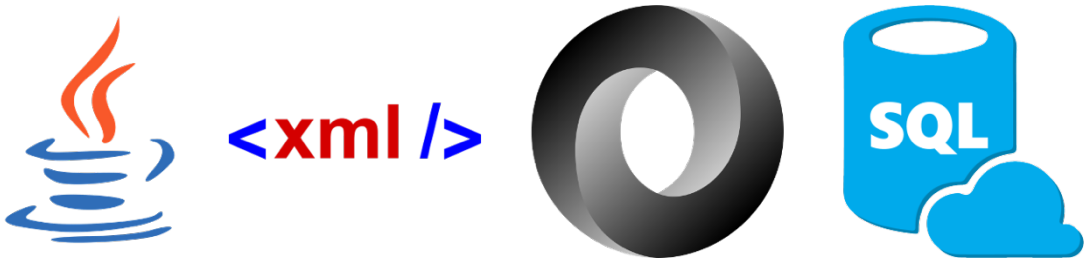


Figura 5. Logos de las tecnologías implicadas.

En este apartado se va a nombrar y describir algunos de los lenguajes de programación y de marcado que se han usado en este proyecto.

2.3.1 Java

Java es un lenguaje de programación de propósito general, concurrente, multiplataforma y orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra.

Java se utiliza como lenguaje base para el desarrollo sobre Android.

Es por eso que es totalmente necesario un amplio conocimiento sobre el uso avanzado de Java para poder realizar correctamente el uso del *API* de Android.

2.3.2 XML

Este es un meta-lenguaje que permite definir lenguajes de marcas desarrollado por el *W3C*, es ampliamente utilizado para almacenar datos en forma legible. Aunque para este propósito es más ampliamente usado *JSON*, ya que representa de manera más eficiente los datos. Permite definir la gramática de lenguajes específicos para estructurar documentos grandes.

Este es el motivo por el que se usa *XML* para la definición de las interfaces gráficas.

2.3.3 JSON

A pesar de no ser un lenguaje de programación, se ha convertido en uno de los lenguajes de más destacados de la época actual. Es un subconjunto de la notación literal de objetos de JavaScript, aunque hoy, debido a su amplia adopción como alternativa a XML, se considera un formato de lenguaje independiente. Es muy ampliamente utilizado para el intercambio ligero de datos.

2.3.4 SQL

SQL, *Structured Query Lang*uage o Lenguaje de Consulta Estructurada, es un lenguaje específico que da acceso a un sistema de gestión de bases de datos relacionales que permite especificar diversos tipos de operaciones sobre ellos. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar, de forma sencilla, información de bases de datos, así como hacer cambios en ellas. Se basan en la filosofía de trabajo básica de un sistema de bases de datos conocida como *CRUD*.

SQLite utiliza una versión estándar de *SQL*, *SQL-92*.

3. Estado del arte

En este apartado se va a introducir que hay en el mercado, como se ha hecho dicha evaluación, y una tabla comparativa

3.1 Metodología de análisis utilizada

Se ha hecho un pequeño análisis sobre el mercado actual, para detectar los valores diferenciadores que esta aplicación proporciona con respecto a las aplicaciones de la competencia.

Tras el estudio, se han identificado las características de cada una, etiquetándolas para realizar una tabla comparativa. De este modo se ofrece una vista clara de lo que ofrece cada una.

3.2 Competencia actual

Tras el pequeño análisis y unos pequeños esfuerzos de búsqueda e investigación de la competencia, se han detectado las siguientes herramientas con las funcionalidades adscritas a cada uno.

3.2.1 Packlink

Nombre	Packlink
Página web	www.packlink.es
Descripción	Es un portal de Internet, con sede en Madrid, que ofrece a sus usuarios la posibilidad de comparar y realizar envíos de paquetería y mensajería, puerta a puerta, con los mejores transportistas. Se ocupan de que tanto particulares como empresas encuentren, servicios de la máxima confianza y al mejor precio.

3.2.2 ParcelABC

Nombre	ParcelABC
Página web	www.parcelabc.com
Descripción	Es un nuevo concepto para todas las entregas, grandes y pequeñas. Su objetivo es ofrecerle el mejor servicio, al mejor precio. Unen miles de empresas de entrega locales e internacionales y llenando sus espacios vacíos con sus paquetes y entregas.

3.2.3 4Trans

Nombre	4Trans
Página web	www.mambo4.com/4trans.html
Descripción	Es la plataforma tecnológica preferida por las empresas de transporte y logística que buscan un sistema de gestión integral completo y personalizable para optimizar sus operaciones, reducir los costes y mantenerse siempre un paso por delante de la competencia.

3.2.4 Correos

Nombre	Correos/Correos Express
Página web	www.correos.es
Descripción	Nacen en 1716 como un servicio público de todos y para todos. Han conseguido ser el mejor proveedor de comunicaciones físicas, digitales y de paquetería de España trabajando con eficiencia, calidad y sostenibilidad; con los mejores profesionales, la mayor presencia territorial y los equipamientos más innovadores. Forman parte del día a día de los ciudadanos, las empresas e instituciones, haciendo su vida más fácil. CORREOS es el sitio de confianza que abandera la innovación para ayudar a los clientes. Cumplen 300 años celebrando su pasado, presente y futuro.

3.2.5 Seur

Nombre	Seur
Página web	www.seur.com
Descripción	Compañía pionera en el transporte urgente con 75 años de historia, lidera el sector en España con tres grandes ejes de negocio: internacional, comercio electrónico y negocio B2B, para empresas de todos los tamaños y sectores.

3.2.6 UPS

Nombre	UPS
Página web	www.ups.com
Descripción	Es una de las grandes empresas de paquetería del mundo. Cada día entrega más de 14 millones de paquetes a más de 200 países de todo el mundo. Recientemente se ha expandido para cubrir otras áreas relacionadas con el transporte como la logística.

3.2.7 MRW

Nombre	MRW
Página web	www.mrw.es
Descripción	De capital 100% nacional. Es la marca de Transporte urgente para envíos nacionales e internacionales con mayor implantación, más de 10.000 personas están vinculadas a la marca en más de 1.300 franquicias y 61 Plataformas Logísticas en Andorra, España, Gibraltar, Portugal y Venezuela. El Grupo realiza una media de 40 millones de envíos anuales, aportando soluciones de negocio concretas para todo tipo de empresas y particulares.

3.2.8 Tipsa

Nombre	Tipsa
Página web	http://www.tip-sa.com/
Descripción	TIPSA es una empresa especializada en servicios de Transporte Urgente de Paquetería Ligera y Documentación, en el ámbito empresarial dentro de España, Portugal y Andorra con un capital 100% español, TIPSA se ha posicionado, en un corto periodo de tiempo, entre las empresas líderes del sector, siendo además una de las compañías de Paquetería Empresarial que mayor crecimiento ha experimentado en el sector durante los últimos años.

3.3 Análisis crítico individual

Es necesario hacer previamente un análisis crítico de la competencia individualmente. De este modo se podrá tratar subjetivamente y extraer valores diferenciadores de cada uno de ellos. Este análisis será muy resumido de características muy destacables.

Los análisis entre móviles van a hacerse solo comparando Android y iOS.

- **Packlink**: Solo cuenta con versión web, no con aplicación móvil. La única funcionalidad que ofrece es seguimiento del envío. Añade un comparador para elegir el mejor proveedor para el envío. También ofrece la posibilidad de hacer envíos internacionales.

Figura 6. Interfaz de la aplicación de Packlink.

- **ParcelABC**: Estos no utilizan intermediación, gestionan el envío íntegramente ellos. Tampoco cuentan con versión móvil. Pero su administración a través de la aplicación web resulta de extrema comodidad.

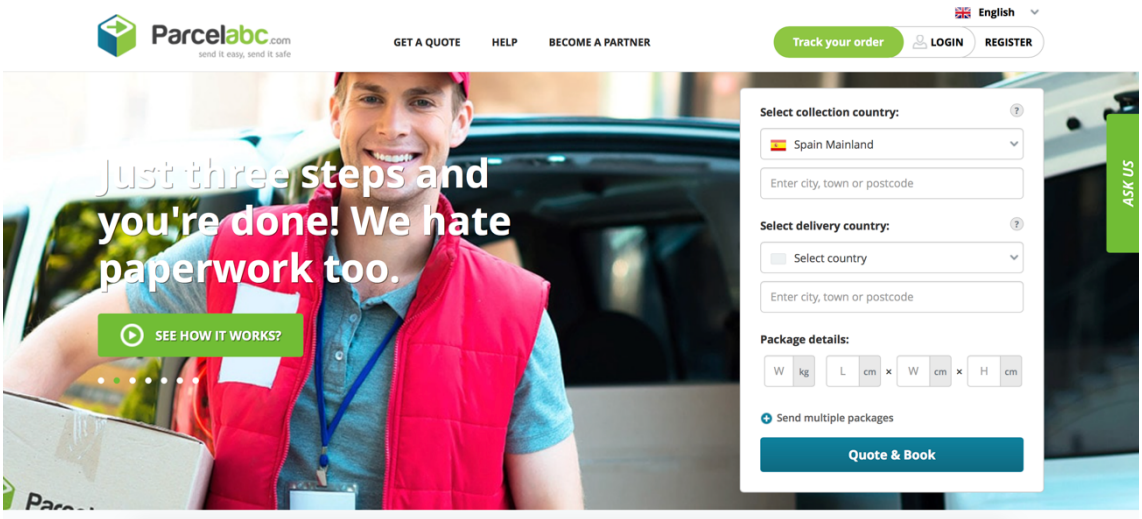


Figura 7. Interfaz de la aplicación de ParcelABC.

- **4Trans**: Afirman contar con una aplicación Android, pero no hay ninguna manera de conseguirla. Afirma tener una gran cantidad de módulos de funcionalidades disponible, pero como he dicho antes, no se pueden comprobar dado que no hay señales de la aplicación.

Figura 8. Interfaz de la aplicación de 4Trans.

- **Correos**: Cuenta con aplicación para Android y iOS, seguimiento de envíos y otras características. Pero solo al estar registrado como empresa.

Figura 9. Interfaz de la aplicación de Correos.

- **Seur**: No tiene aplicación para móvil, aunque todos los envíos pueden seguirse desde la web.

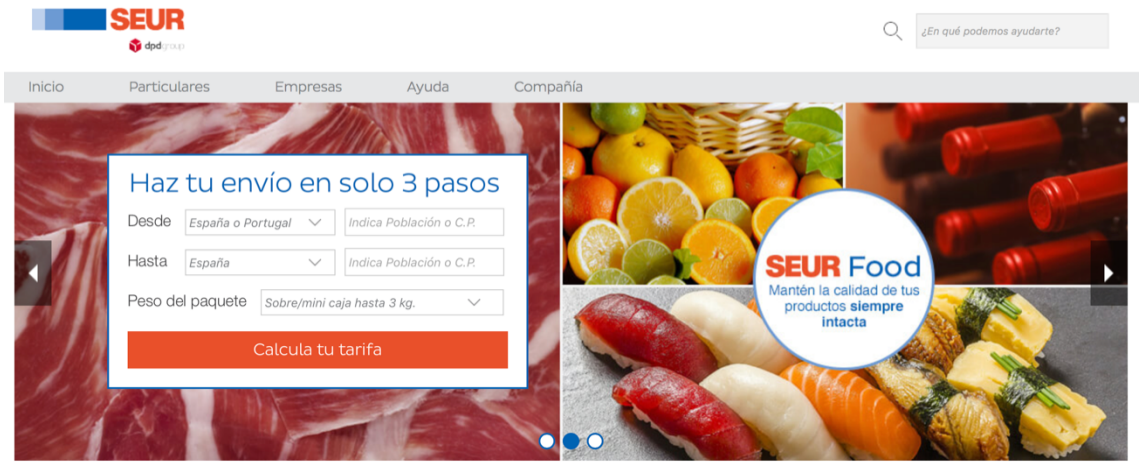


Figura 10. Interfaz de la aplicación de Seur.

- **UPS**: Estos sí que tienen aplicación para móviles y además cuenta con una API pública de integración para desarrolladores. Además, cuenta con servicios de aviso por *SMS* y correo electrónico.

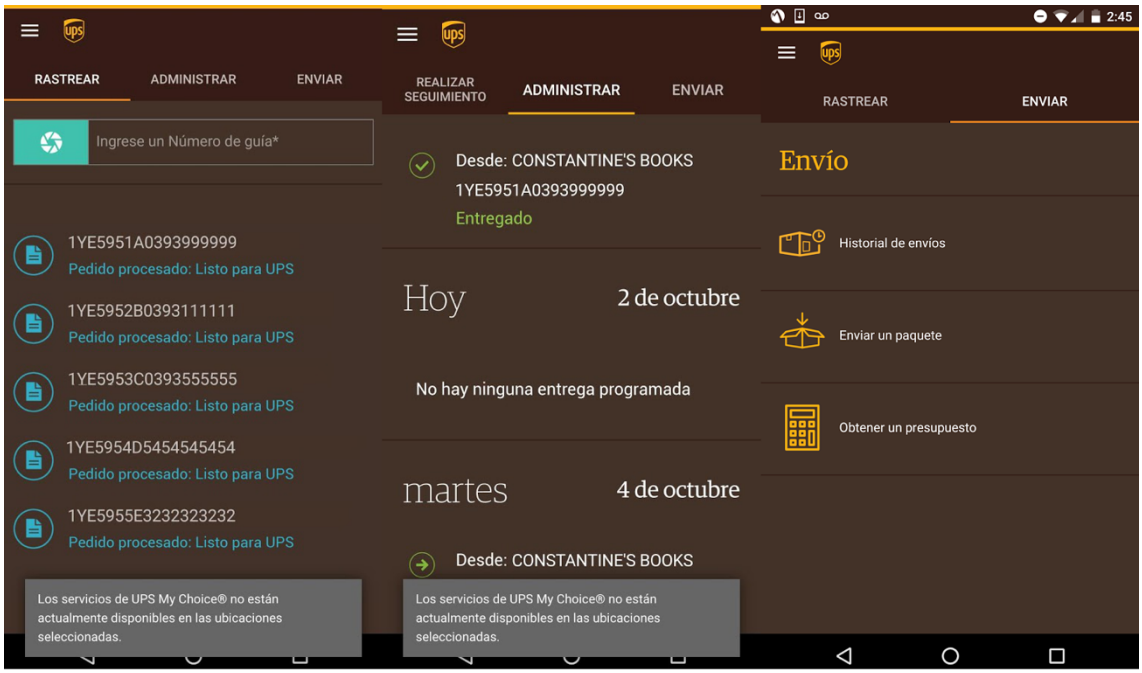


Figura 11. Interfaz de la aplicación de UPS.

- **MRW**: También cuentan con aplicaciones para móviles. Cuenta con las funciones básicas que los otros proveen.

Figura 12. Interfaz de la aplicación de MRW.

- **Tipsa**: Cuenta con aplicaciones móviles, aunque no nativas. Los demás puntos habituales los provee como sus compañeros.

Figura 13. Interfaz de la aplicación de Typsa.

3.4 Tabla comparativa de funcionalidades

Tabla de asignaciones a letras

A: Packlink B: ParcelABC C: 4Trans
D: Correos E: Seur F: UPS:
G: MRW: H: Tipsa: I: **EasyTrans**

Característica	A	B	C	D	E	F	G	H	I
Aplicación móvil				✓		✓	✓		✓
Seguimiento de envíos	✓	✓	✓	✓	✓	✓	✓	✓	✓
Sistema de avisos tradicionales					✓	✓			✓
Actualizaciones instantáneas									✓
Integración con sistemas informáticos						✓			✓
Aplicación web	✓	✓			✓	✓	✓		
Sistema de mensajería instantánea									✓
Localización de trabajos cercanos									✓
Asistente y geolocalización de envíos									✓

3.5 Interpretación de los resultados

Con los resultados obtenidos vamos a hacer una pequeña evaluación.

En esta evaluación se puede ver que todas ellas ofrecen características y funciones dispares, hay funciones interesantes que no todas integran, y desde luego no todas integran muchas de ellas.

Son soluciones separadas para cada funcionalidad especifica. No es posible tener todas las funciones de las que se tratan de manera unificada de las que podamos hacer uso.

Es por eso que se llega a la decisión de desarrollar una aplicación de paquetería, que unifique la gran mayoría de funciones, teniendo a mano todas las funcionalidades deseadas con un solo vistazo. Esa ha sido la principal motivación del proyecto.

4. Arquitectura

4.1 Diseño del modelo

El modelo ha resultado fácil de diseñar, ya que había una gran parte de trabajo hecho gracias a la metodología del DCU.

Es por esto que, tras la identificación de las necesidades, de los deseos y los demás parámetros necesarios, se ha diseñado un modelo que cumpla la gran parte de expectativas de los usuarios.

4.2 Diagrama general

El modelo implantado en la aplicación puede comprenderse en este diagrama de bloques, que facilita la compresión de cada módulo por separado y proporciona una visión de conjunto de toda la funcionalidad de que se provee de manera lo más general posible.

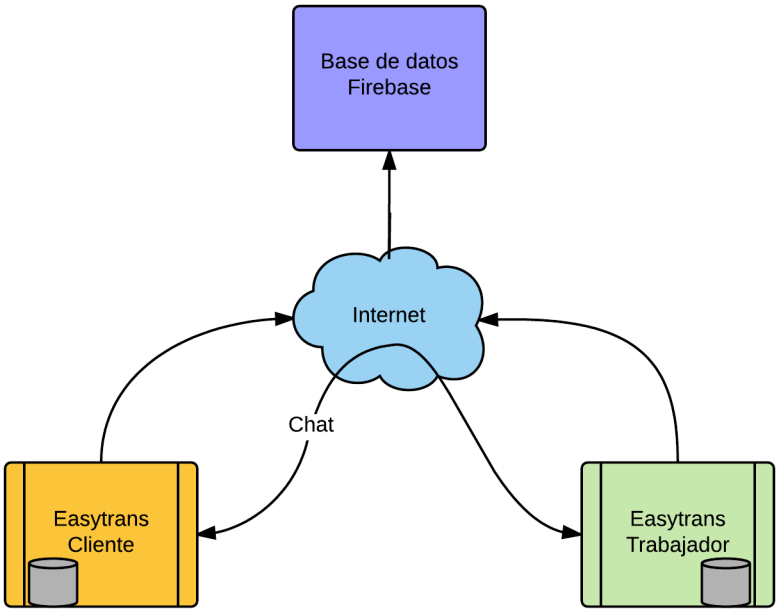


Figura 15. Diagrama de bloques del proyecto

4.3 Bloques individuales

4.3.1 Aplicación Android

La aplicación utiliza los mecanismos internos con los que cuenta el sistema operativo para todo el manejo de la lógica de aplicación y el intercambio de información. De este modo mucha lógica de intercambio de mensajes entre actividades y demás se proveen a través del sistema. Se puede consultar en detalle en el apartado de implementación.

4.3.2 Peticiones HTTP

Internamente la aplicación necesita hacer uso de peticiones HTTP para enviar los *payload* que las notificaciones push utilizarán.

De este modo ha sido necesaria la planificación del modo de envío esa petición desde el sistema operativo, ya que este permite diseñar la petición HTTP desde cero. Incluyendo todas sus cabeceras. Esta tecnología y su funcionamiento se describe en profundidad en el apartado de implementación de esta memoria.

4.3.3 Acceso a las bases de datos

La aplicación por decisión de implementación, se decidió integrar en dos bases de datos diferentes. Cada una de ellas tiene un propósito distinto, la base de datos Firebase integra toda la información personal referente a los usuarios y a la información acerca del estado de sus envíos. La base de datos local, tiene un fin complemente distinto, se encarga de almacenar los mensajes de chat enviados y recibidos. De este modo no es necesario gestionar una lógica muy compleja para el diseño del chat.

Este mecanismo tiene ventajas e inconvenientes. No hay constancia en la base de datos Firebase de ningún tipo de mensaje, cosa que suele ser frecuente, esto implica una reducción de la carga de la base de datos, y por tanto una mejora en el rendimiento de la aplicación. Por otro lado, al no haber constancia en la base de datos Firebase de estos mensajes, un cambio de dispositivo, una desinstalación u otro motivo similar provocaría la pérdida de los mensajes recibidos y enviados antes de la propia actualización del *token* destino en la base de datos Firebase.

Dado que el fuerte de la aplicación era su fuerte y robusto diseño, la decisión de hacerla local implica una ventaja de carga de trabajo en el servidor.

4.3.3.1 Acceso a la base de datos Firebase

El acceso a la base de datos Firebase viene descrito según la API de Google muy detalladamente.

El mecanismo básico para la realización de lecturas en la base de datos es mediante *listeners*. Los *listeners* generan un *callback* que es invocado en el momento en que esa lectura es efectiva. Estos escuchadores pueden ser de lectura múltiple o de lectura única.

Esto provee una interesante funcionalidad de actualización en tiempo real, ya que un escuchador de lectura múltiple generaría un *callback* cada vez que el valor se actualice o deje de escucharse. De este modo se puede proveer de una actualización instantánea con un código relativamente simple o con poco manejo de lógica.

El otro lado sería la lectura única, la que generaría un único *callback* en el momento de la lectura y no volvería a invocarse.

A través de objetos implementados en Java es posible acceder a todos los datos de la base de datos, lo cual implica la necesidad de implementar una lógica de búsqueda y tratamiento sobre los datos que recibes, muy fácilmente implementable con estructuras de datos.

Los fragmentos de código pueden observarse en la sección de implementación, la cual provee detalles mucho más concretos acerca de la implementación de este tipo de lógicas.

4.3.3.2 Acceso a la base de datos local

La base de datos local es una base de datos SQLite con un esquema muy sencillo con una filosofía que utiliza las dos primeras funciones básicas de las cuatro en cualquier base de datos. CRUD.

Solo hace inserciones y lecturas de la base de datos, no requiere la actualización ni el borrado de ningún elemento. Esto simplifica la interacción con ella.

Las tareas de obtención se simplifican gracias a una consulta unificada que devuelve el contenido directamente en el formato destino deseado.

Detalles acerca de esto se verán en el apartado de implementación.

4.3.4 Notificaciones Push

Las notificaciones push normalmente se entregan a través de un servicio web, gestionado por una *API*. Pero tras una evaluación de estimación de costes, se decidió crear una infraestructura con un coste muy reducido, la cual permite la réplica del servicio para cualquier clon que pueda surgir de la aplicación.

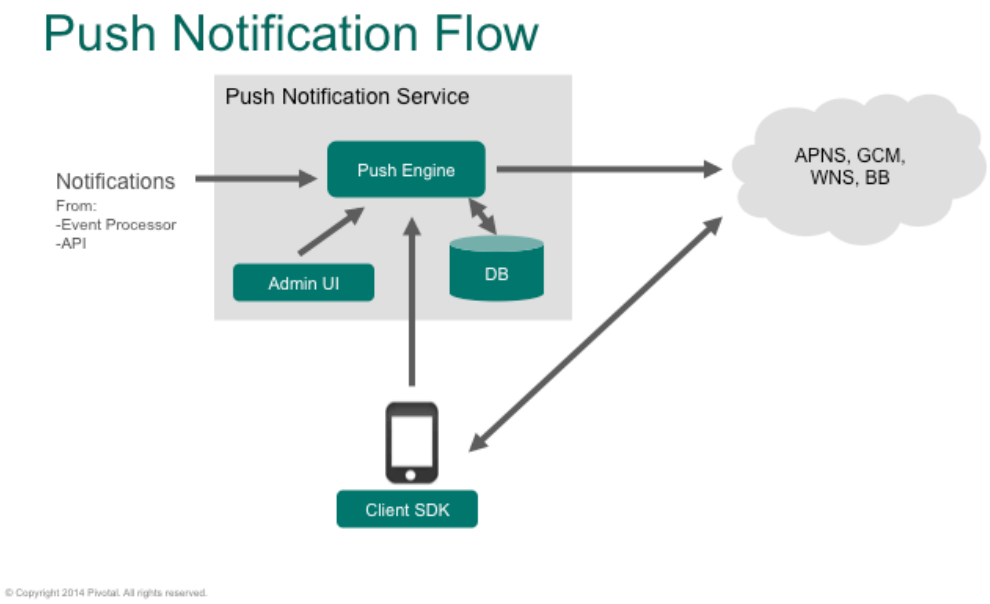


Figura 2. Flujo de una notificación Push.

Por ello se ha implementado la aplicación con una base de datos en un servidor, sin la necesidad de un *API*. Todo conexo entre si gracias a las características de Firebase.

Como puede observarse en la Figura 2, el flujo pasa por los siguientes procesos:

Primero un dispositivo genera una petición a un dispositivo que gestione las comunicaciones *push*. Un servidor web que almacene servicios web, o que contenga un API es lo más habitual.

Tras esto genera una petición al servidor push de cada destino al que se quiera enviar. APNS es el acrónimo de *Apple Push Notification Service*, siendo este el referente a dispositivos iOS, GCM equivale a *Google Cloud Messaging*, siendo este el de dispositivos Android, Windows y BlackBerry corresponderán a los restantes.

Este *middleware* será el encargado de generar una notificación que el sistema operativo destino recibirá y sabrá cómo y a quién entregársela.

Con este modelo optimizamos el tiempo de notificación reduciendo los intermediarios.

5. Metodología

5.1 Metodología empleada

Durante todo el proyecto, se ha empleado la famosa metodología de Diseño centrado en el usuario, conocida por sus siglas DCU.

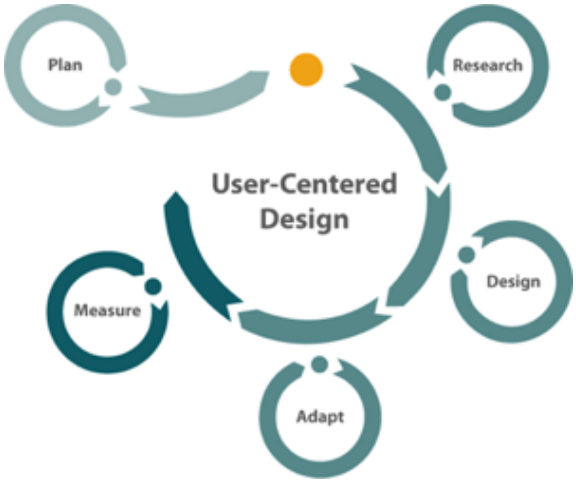


Figura 14. Diagrama de flujo del DCU.

La técnica de Diseño Centrado en el Usuario, está definida por la *Usability Professionals Association* como un enfoque del diseño, cuyo proceso está dirigido por información sobre las personas que van a hacer uso del producto.

El principal objetivo de la metodología del DCU, es el estudio de los usuarios potenciales de la aplicación y sus necesidades, con el fin de desarrollar el producto con el que van a interactuar, logrando la satisfacción de todos los usuarios potenciales, la adaptación de las tecnologías a sus expectativas y la creación de interfaces más fáciles de utilizar, que les permitan lograr fácilmente sus objetivos.

El principal proceso que se utiliza en el DCU se basa en identificar a las personas a las cuales va dirigido el producto, identificar para qué quieren utilizar el producto, identificar los objetivos del proveedor y del usuario, producir soluciones de diseño a los requisitos identificados y evaluar si en estas soluciones se cumplen todos los requisitos, o hay algún problema de usabilidad, en cuyo caso deberíamos retroceder y volver a aplicar todo el proceso de nuevo hasta que se consiga lo que se busca.

Por tanto, se puede afirmar que la técnica del DCU es un proceso cíclico donde los usuarios, que participan desde las primeras etapas del desarrollo, y los objetivos que debe cubrir el producto, son los encargados de dirigir las decisiones de diseño. Además, también será comprobada la usabilidad del producto de forma iterativa, con el fin de ir mejorándola poco a poco, hasta que se alcancen los objetivos de usabilidad deseados. Este es el principal proceso de cualquier metodología iterativa.

Por lo tanto, se puede afirmar abiertamente que esta ha sido la principal técnica que se ha utilizado en el proyecto. Se han hecho uso de las metodologías que el DCU proporciona para obtener un producto con la mayor calidad posible y siguiendo una metodología de resultados comprobados.

5.2 Herramientas utilizadas

Dentro del DCU existen una amplia variedad de herramientas para seguir una línea de diseño de producto.

Una de las herramientas que datos más útiles nos ha generado ha sido el resultado del análisis de las encuestas. Las encuestas nos han dado la impresión general que tiene los usuarios de este tipo de servicios, expectativas que se imponen de manera automática, deseos de funciones y demás.

Se han identificado, en primer lugar, las necesidades de una población que resulta representativa de los usuarios potenciales y habituales de esta aplicación. A partir de esta información, se ha utilizado la técnica del DCU de Personas y el uso de prototipos para diseñar las interfaces de usuario. De este modo obtendremos el diseño lo más parecido a lo que desea el cliente final.

5.3 Fases del proceso

Ahora se va a proceder a describir cada fase del proceso seguido, dando detalles de cada fase para conocer en detalle cómo se ha realizado.

5.3.1 Estudio de mercado

Como se lleva comentando a lo largo de toda la memoria el primer punto ha sido el estudio de mercado realizado, el cual será descrito en profundidad en un apartado posterior de esta memoria, haciendo hincapié en detalles más técnicos.

5.3.2 Diseño y aplicación del cuestionario

Este proceso es el que más tiempo requirió para poder realizarse, ya que la zona escogida es una zona muy transitada, y no todos los viandantes tiene disponibilidad para dedicarte unos minutos.

Una vez se obtuvieron los resultados, la participación esperada tras dos días de encuestas fue aceptable, suficiente como para poder avanzar en el proceso del DCU.

Los cuestionarios utilizados pueden verse en el anexo correspondiente.

5.3.3 Establecimiento de un modelo de persona

El siguiente paso es el establecimiento del modelo de persona. Este modelo nos permite identificar cual es el usuario estrella de la aplicación. Un usuario ejemplo que es el que representa con mayor fidelidad, a un usuario potencial de tu aplicación.

Esta persona será la que utilizaremos en el proceso para planificar todas las mejoras pensando en ficha persona.

La persona desarrollada puede verse en el apartado dedicado a ello más adelante en el apartado del análisis de necesidades.

5.3.4 Establecimiento de un escenario

En este caso no proporcionaba información representativa la identificación de un escenario, por lo que se ha decidido no realizar este apartado del proceso, ya que un usuario puede hacer uso de esta aplicación sin tener en cuenta el escenario en que se encuentre.

5.3.5 Organización de la información

Ahora que ya se tiene la persona, es importante tomar la decisión de que información podrá ver el usuario en cada pantalla, que espera ver, que espera no ver, para satisfacer sus expectativas y necesidades del modo más apropiado y óptimo posible.

Este proceso consiste en organizar la información de la que se dispone en la aplicación de forma que esta sea más fácil de usar y de encontrar. Dada la existencia de una infinidad de modelos de organización de la información, en este caso se ha optado por un esquema por tareas, es decir, cada tarea que se puede realizar en la aplicación le corresponderá una pantalla en nuestra aplicación.

5.3.6 Diseño de la interfaz de usuario

Siendo subjetivo, este ha sido el proceso más duro de todo el proyecto, en este punto se ha desarrollado una interfaz de usuario, partiendo siempre de los deseos de los usuarios, y aplicando las herramientas facilitadas gracias a la metodología del DCU.

Aun así, el diseño de interfaces, es una tarea que los desarrolladores tenemos pendiente de aprender a aplicar. Los diseños, a nivel empresarial, normalmente suelen ser diseñados por gente dedicada a ello, los conocidos diseñadores gráficos. La tarea se facilita en gran medida cuando esta labor la desarrollan ellos, personas especializadas en estos temas.

Este proyecto, al tener un presupuesto muy reducido, no ha podido contar con ello, por lo que los diseños han sido de elaboración propia. Pueden verse los *mockups* en el apartado 7.

Se han realizado distintos *mockups*, para el diseño de las distintas ventanas de la aplicación, con el fin de facilitar la tarea de desarrollo, teniendo especificadas todas las tareas necesarias. La fase de prototipado no ha sido necesaria, ya que era para uso propio, por lo tanto, era más que suficiente para las tareas de la implementación del diseño.

5.3.7 Implementación de lo planificado

En esta fase, con todas las tareas previas realizadas, el proceso es mucho más rápido. Las tareas de programación se ciñen a un guion muy bien definido por la metodología que permite focalizar todos los esfuerzos en realizar las tareas, bien definidas. De este modo se puede realizar una estimación objetiva del coste temporal de la tarea, el cual ha sido notablemente menor que programar “con los ojos vendados”, analogía que equivaldría a programar sin planificación, ni organización, ni definición. Esto implicaría un desastre a nivel de código, y una calidad muy baja del mismo.

Por lo que gracias al conocimiento previo que poseo sobre la API de Android, ha resultado bastante rápida esta fase, prácticamente sin problemas.

Tan solo cabe destacar que las tareas de diseño de interfaz gráfica no han resultado igual. De esto se hablará en el apartado 9 de conclusiones, en el que haré hincapié en la dificultad de tareas de diseño para personas con poca habilidad artística.

5.3.8 Evaluación de usabilidad

Una vez se tiene un producto casi finalizado, el último proceso del DCU es la evaluación de la usabilidad, para comprobar que se ha aplicado el proceso correctamente, y que la aplicación cumple en gran medida las expectativas de los usuarios tal cual se descubrió en los resultados de las encuestas.

Esta evaluación puede realizarse mediante test, como todas las anteriores y en función de los resultados, se podrá saber si se he aplicado la metodología correctamente y si has obtenido los resultados de calidad que se esperaban.

La encuesta utilizada puede observarse en el anexo correspondiente.

6. Análisis de necesidades

6.1 Estudio de mercado

El estudio de mercado que lleva comentándose en varias secciones previas de esta memoria, ha resultado de gran ayuda para darnos cuenta de una gran cantidad de necesidades que los usuarios tienen a la hora de hacer una tarea tan simple como enviar un paquete.

Este estudio ha detectado que los usuarios están muy dispuestos a la aparición de una aplicación de este tipo para cumplir sus necesidades, ya que opinan que una aplicación que englobe todos los servicios de los que se trata, es muy complicado de encontrar, y las funcionalidades, ni mucho menos complejas, resultan muy cómoda en día a día de un comprador y un vendedor que hagan uso de este tipo de logística.

Se ha preguntado a una pequeña población de gente para obtener este estudio de mercado en distintas ciudades.

El *target* del estudio ha sido bastante concreto, con un foco muy claro de compradores a través de comercio electrónico. Compradores en eBay, Amazon y demás famosas plataformas de venta online. Este es el hábito más parecido que buscamos, ya que un posible cliente de nuestra aplicación puede ser totalmente compatible con el de un cliente de estas plataformas.

Las posibilidades de la plataforma no solo se quedan ahí. Al haber hecho este estudio de mercado se ha podido observar que es totalmente compatible el uso de nuestra plataforma con el comercio electrónico, y que las plataformas actuales son potenciales clientes nuestros, dado que podría ser un servicio diferenciador para una plataforma de venta online la implantación de un servicio como este en sus servicios de logística.

6.2 Cuestionarios

El cuestionario que han rellenado los viandantes de Valencia, ha recabado información a partir de preguntas acerca de hábitos de la gente con respecto al envío de paquetería.

Estos cuestionarios se han realizado durante dos días a los viandantes de Valencia, concretamente por la zona de la calle Blasco Ibáñez en un trayecto que comprendía entre los números 40 y 70.

Fueron alrededor de unas 40 personas las que se prestaron muy amablemente a realizar las encuestas.

De este modo, haciendo una serie de preguntas con respuestas cerradas, a las que el usuario responde si usa, quiere, desea.

Al final de este cuestionario también se ha hecho un análisis de expectativas aplicando una metodología similar al método *MoSCoW*, pero aplicando una analogía para este caso, aunque no con el fin de priorizar, sino con el fin de identificar si los usuarios piensan que algunas funciones deben (***M**ust have*), deberían (***S**hould have*), podrían (***C**ould have*) o no deben estar (***W**on't have*).

Figura 16. Metodología MoSCoW

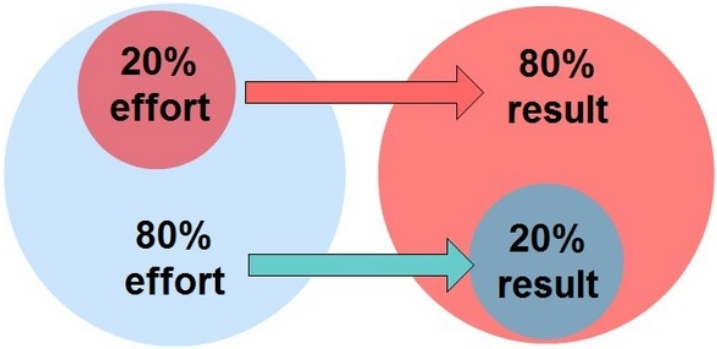


Figura 17. Relación esfuerzo resultado.

De este modo usaremos la planificación de prioridades para invertir de la manera más eficiente posible los esfuerzos de este proyecto.

En la figura 16 podemos observar el significado de las siglas de esta metodología, pero en la figura 17 observamos por qué se ha aplicado de este modo. Al invertir el esfuerzo bien planificado y organizado, con poco esfuerzo más podemos conseguir un buen resultado. Sin embargo, si usamos mucho esfuerzo, pero mal planificado, se obtendrá un resultado pésimo.

Se puede acudir al Apéndice A para ver tanto las preguntas que contiene la encuesta, como los resultados obtenidos por la misma.

6.3 Resultados recogidos por el formulario

Tras haber realizado todos los formularios pueden verse unos resultados bastante claros.

Todo el mundo afirma tener impaciencia con respecto a la fecha de llegada de sus encargos, por lo que valoran con una prioridad alta para tener dentro de las funcionalidades de nuestra aplicación.

El apartado que valoran más exitosamente es la latencia de actualizaciones del pedido. Los clientes afirman que no están en todo momento al tanto de cómo está el estado de su pedido, y que la aplicación sea la que se encargue en todo momento de recibir esa información para mantenerles informados, es algo que afirman valorar mucho.

El tema del chat ha caído dentro de la controversia. Muchos afirman que alguna vez han tenido problemas con los repartidores, desde pasar por casa tras pasar 2 minutos, de haber salido, o estar esperándolos en todo momento y que al final no aparezcan. Todos estos temas generan malestar entre los clientes. Es por eso que **Easytrans** cuenta con un sistema que permite al usuario ponerse en contacto directo con el repartidor, para tratar cualquier tema relacionado con el pedido sin más molestia que coger su móvil y hablarle. Tan simple como enviar un mensaje por Whatsapp.

A nivel profesional, dado que la aplicación se usa tanto como cliente, como trabajador, me he tomado la molestia de hablar con dos personas trabajadoras en el sector del reparto logístico. Estos han sido también entrevistados con formularios y los resultados obtenidos han sido completamente contrarios, por lo que decidí entrevistar más allá a una de las dos personas. El motivo dado es que es una idea buena, pero para el trabajador tan solo le genera más carga de trabajo que no puede sobrellevar. A nivel empresarial puede parecer una buena idea, pero el repartidor que se encuentra siempre conduciendo, resulta complicado conseguir sacar tiempo para interactuar con la aplicación.

Aun así, le pedí que hiciese una valoración como cliente. Se obtuvieron resultados similares a los de la mayoría. Por lo visto su relación laboral con el mercado logístico fue relevante para su primera valoración.

6.4 Modelo de persona

La persona ha sido definida cualitativamente a partir de los resultados obtenidos por los cuestionarios y también gracias al estudio de mercado. La persona queda del siguiente modo:

Floriano García Lluch
32 años
Técnico en instalación y mantenimiento electromecánico

Biografía:

- Vive con su pareja.
- No tiene hijos.
- Está en estado de pareja de hecho.
- Cuenta con coche propio.
- Adicto a las compras a través de Amazon.

Tecnología:

- Cuenta con acceso a Internet tanto en el móvil, como en casa, como en el trabajo.
- Vive conectado a internet y a sus servicios.
- Odia esperar para recibir actualizaciones de sus acciones.
- Le encantan las compras por Internet.

Objetivos:

- Mantenerse informado del estado de sus compras.
- Poder contar con un método rápido de comunicación con el repartidor.
- Realizar una venta de un producto sin tener que salir de casa.

7. Diseño

Este apartado contiene todos los elementos de diseño sobre los que se apoya la memoria.

Tiene una fuerte carga de trabajo, pero al mismo tiempo supone una ayuda a la hora de transformar diseños en aplicaciones. Dado que no hay que pensar en cómo realizarlo, tan solo hay que plasmar lo diseñado en código.

Es por esto que los diseños deben partir de una buena planificación, lo cual implicaría un flujo de ejecución definido y correcto. Si en esta etapa se cometiesen errores, serían propagados en cascada en todo el proceso, por eso debe ser este un punto en el que invertir tiempo en su desarrollo, su evaluación y su corrección en caso de necesidad.

7.1 Diseño de la base de datos

Los dos tipos de bases de datos utilizadas en la aplicación son diferentes en diseño y tecnología, es por eso que no es posible definirlas del mismo modo. Si bien es posible que ambas tengan objetivo común, necesitan definirse de modo distinto para su correcta implementación.

7.1.1 Base de datos Firebase

La base de datos de Firebase como se ha comentado anteriormente, es una base de datos que utiliza un esquema de almacenamiento JSON. De este modo el esquema se va a definir en modo definición de objeto JSON mostrando las posibilidades de la estructura definida.

```
{
  "orders" : {
    "02f2791eb77f4cdfb5e37fe6a650f26f" : {
      "date" : 1495722376823,
      "destination" : {
        "lat" : 39.478886,
        "lng" : -0.3577818
      },
      "sender" : "hTB9cSQS5NfHr6jD4hoY1mTxaPA2",
      "status" : 4,
      "worker" : "1Ma03uMTRPXB1l2QjAfXCudYl8Z2"
    }, ...
  }
  "users" : {
    "1Ma03uMTRPXB1l2QjAfXCudYl8Z2" : {
      "address" : {
        "lat" : 39.50454515366241,
        "lng" : -0.4476918652653694
      },
      "imagen" : 2130837634,
      "mail" : "dazavil@inf.upv.es",
      "nombre" : "David Zeta",
      "tipo" : "chofer",
      "token" : "dlX8zu2XNU4:APA91bGRVKxLYS0rdwgPqSozAtvZSjs00Mob-
6DKwE8HAMOq6zv4l-
yeZADAE5vZeT1JDbeLgejsLJTyAVnZA69k1M3_QyBh6DO6gXZinmFn0w59P3zFf7HmC6nArUkN4D0
RuwrpQQQa"
    }, ...
}
```

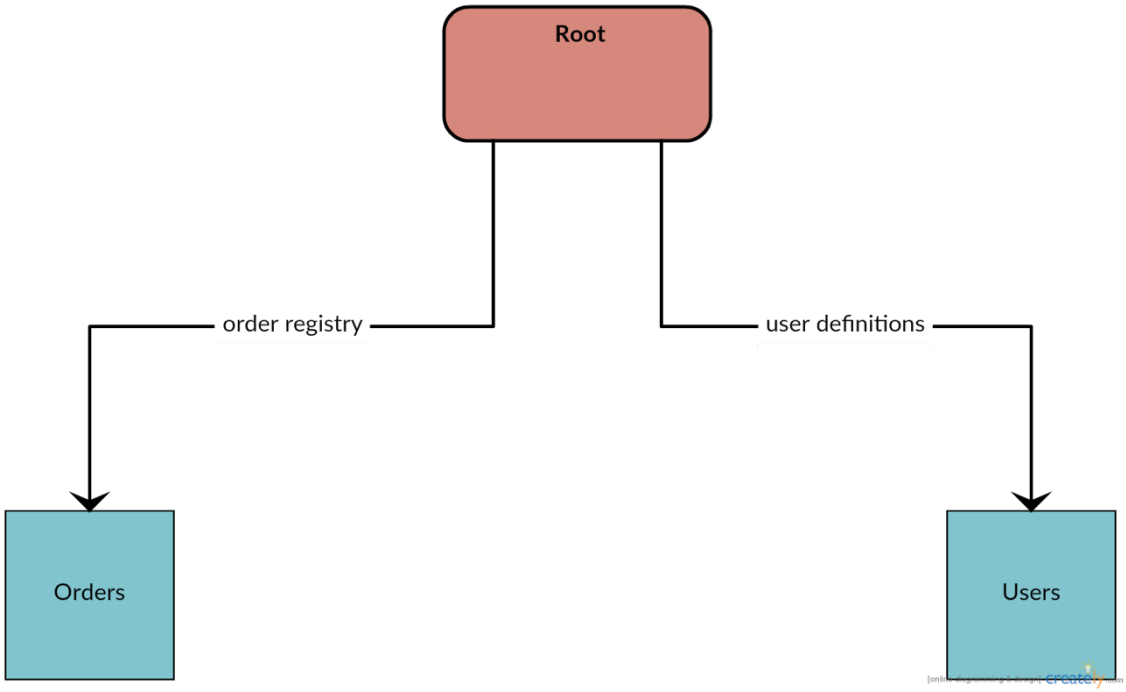


Figura 18. Esquema de bloques de la base de datos firebase.

En este esquema JSON se puede observar un ejemplo de cada tipo de entidad, tras este, hay unos puntos suspensivos. Su función es representar que existen más contenidos tras este de ejemplo. Dada su extensión, se ha decidido poner uno de ejemplo, ya que es bastante representativo y los demás son iguales cambiando algunos parámetros.

En el diagrama de bloques representado por la Figura 18, se puede observar la simplicidad del diseño de la base de datos, pero a su vez la potencia representativa que tiene.

Con estos dos esquemas se puede obtener una idea bastante cercana a la premisa con la que se planteó e implantó.

7.1.2 Base de datos local

Al tratarse de una base de datos relacional, esta puede ser fácilmente definida y comprendida mediante un diagrama de clases, o un diagrama de entidad-relación.

De este modo es posible entender por qué se ha decido implementar esta base de datos desacoplada del sistema común de Firebase. Este sistema permite una abstracción y un aislamiento del sistema, que permite que los usuarios puedan comunicarse directamente sin generar tráfico sobre el sistema central. Así solo es molestado el usuario que genere la petición, y de este modo, el servidor no cuenta con ningún tipo de acción que suponga ser un cuello de botella para el flujo de ejecución de la aplicación cuando la masa de usuarios del sistema aumente en un futuro.

Sin más este es el esquema de la base de datos SQLite usada para almacenar los mensajes:

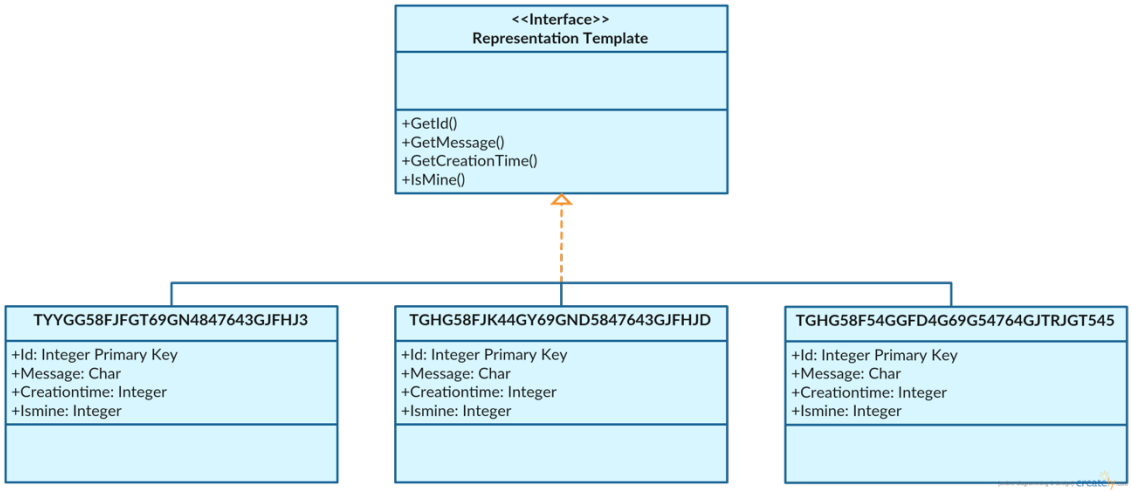


Figura 19. Diagrama de clases de la base de datos local.

Como se puede observar en este sencillo diagrama de clases ejemplificado mediante la Figura 19, todas ellas comparten la funcionalidad básica de lectura de sus atributos. Se trata de una base de datos muy básica, la cual hace que se almacenen los mensajes con el fin de delegar la responsabilidad de la obtención de la lista de mensajes ordenada de manera automática, y con ello el mostrado de los mensajes en su correcto orden.

Como se puede ver todas ellas son un calco, usando un método básico para identificarlas.

En el proyecto se usa la siguiente consulta para obtener todos los mensajes de manera ordenada:

```
select * from T"+SQL_TABLE_KEY+" order by creationtime asc;
```

Este es el modo mediante el cual es posible obtenerla ordenada a través de su fecha y hora de inserción.

Como se puede observar todas las tablas comienzan por T. SQL_TABLE_KEY se sustituye por el identificador el usuario, esto provoca la posibilidad de que haya un identificador que comience por un número. Algo prohibido en SQLite, no está permitido el comienzo de nombres de tabla por números. Por eso se usa la T inicial en el nombre.

7.2 Diseño de la aplicación

La aplicación ha seguido muchos de los patrones de diseño comentados anteriormente. Es por eso que esta fase va a comentar tan solo lo que se ha diseñado para continuar con la fase de implementación, la cual es la parte más técnica del proyecto.

7.2.1 Casos de uso y Escenarios

La aplicación se ciñe al siguiente diagrama de casos de uso, que creo que es el que mejor define la funcionalidad de la aplicación a grandes rasgos.

En este se puede ver como las funciones que pueden realizar los usuarios están muy bien definidas. Algunas funcionalidades son comunes, y otras son para un determinado perfil de usuario. Se diferencian claramente el rol del cliente, con el rol del trabajador.

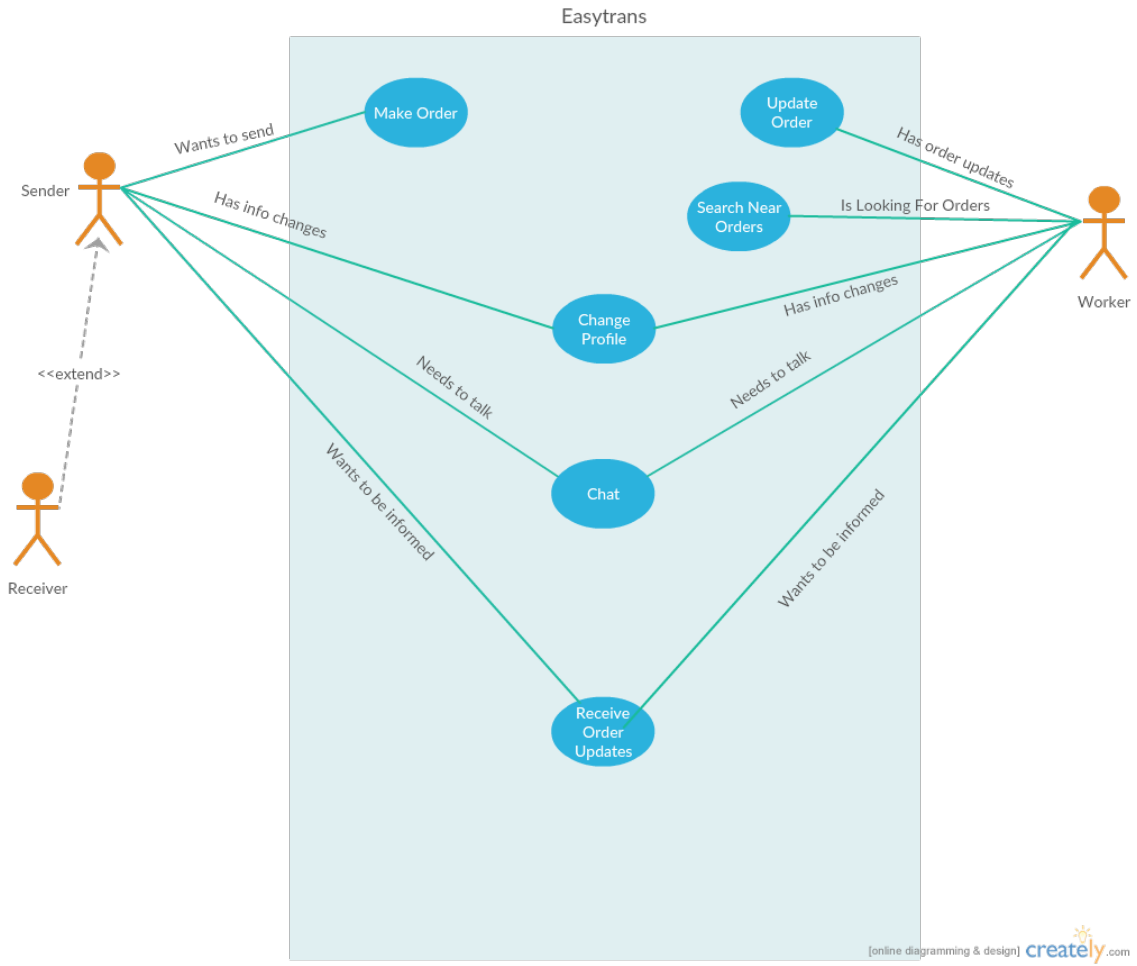


Figura 20. Diagrama de casos de uso simplificado.

Esta figura 20, representa los escenarios y casos de uso que plantea este ecosistema. Dado que el usuario tiene acceso a realizar una serie de acciones, se han definido las siguientes:

- Cliente:
  - Generar encargos: Es capaz de generar un nuevo encargo.
  - Cambiar su perfil: Puede cambiar información acerca de su perfil.
  - Chatear: Tiene la posibilidad de chatear con otros usuarios.
  - Recibir actualizaciones de sus encargos: Recibe actualizaciones.

- Trabajador:
    - Actualizar encargos: Permite actualizar información de sus encargos.
    - Mostrar encargos cercanos: Permite ver y escoger encargos cercanos a él.
    - Cambiar su perfil: Puede cambiar información acerca de su perfil.
    - Chatear: Tiene la posibilidad de chatear con otros usuarios.
    - Recibir actualizaciones de sus encargos: Recibe actualizaciones.
- Receptor:
    - En esencia es un cliente por lo que tiene las mismas funciones que un cliente normal.

7.2.2 Mockups

Tras la presentación de la parte funcional básica de la aplicación, llega el momento de ver los diseños de interfaz de usuario que se han diseñado para la app.

Los *mockups* se han realizado con una herramienta online llamada **Moqups**. Esta herramienta es de pago, pero cuenta con un plan gratuito para un número máximo de elementos. Ser adjuntará un enlace a su página web al final de este documento.

A continuación, se pueden ver los *mockups* de las distintas pantallas de la aplicación. Debe tenerse en cuenta que estos mockups cuentan con un nivel de detalle pequeño, con el objetivo de poder exprimir al máximo el diseño funcional y terminar realizando un diseño lo más práctico y útil posible.

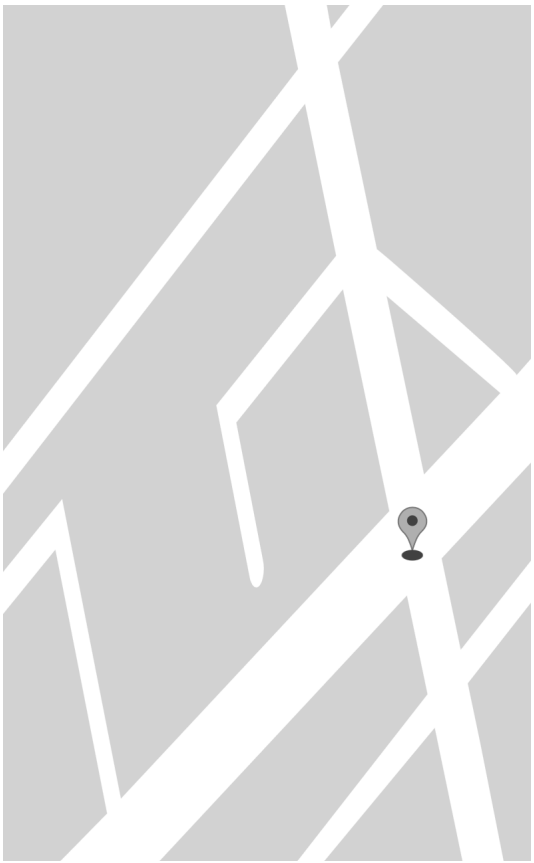


Figura 21. Pantalla de encargos cercanos.

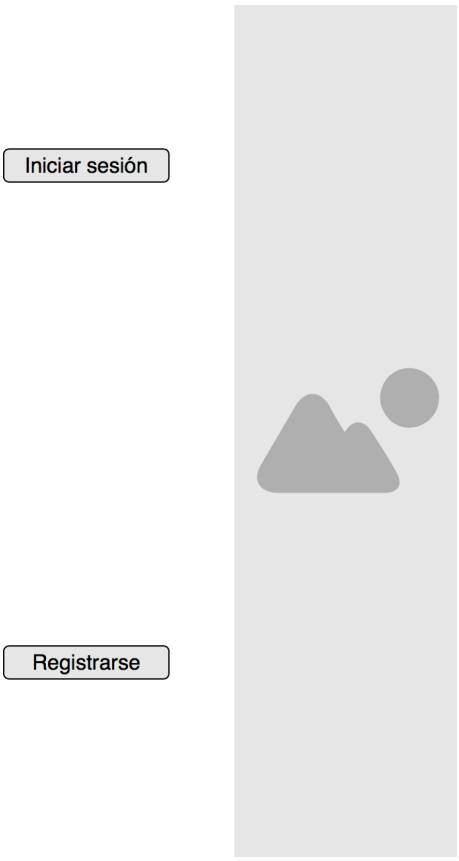


Figura 22. Pantalla de bienvenida.

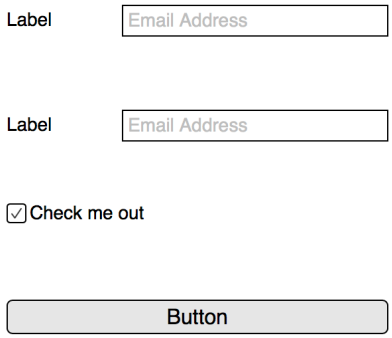


Figura 23. Pantalla de inicio de sesión.

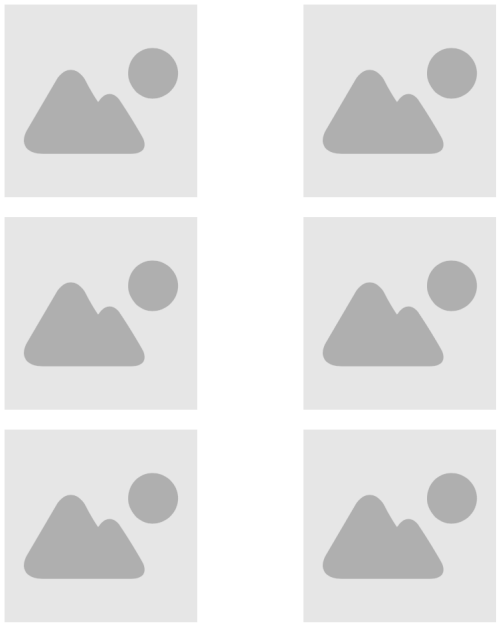


Figura 24. Pantalla de menú de funciones.

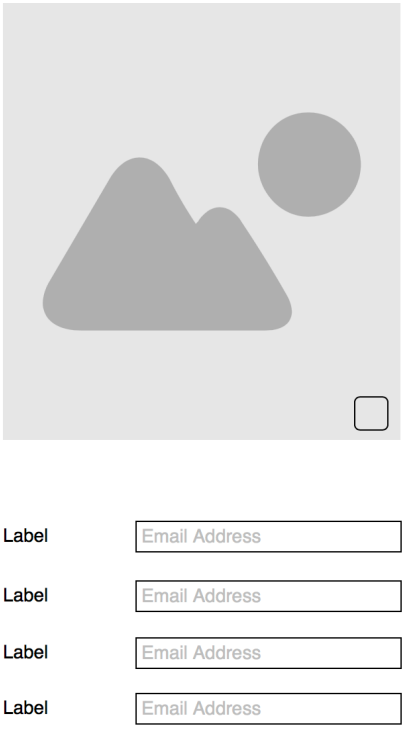


Figura 25. Pantalla de edición del perfil de usuario.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id. Sed rhoncus, tortor sed eleifend tristique, tortor mauris molestie elit, et lacinia ipsum quam nec dui. Quisque nec mauris sit amet elit iaculis pretium sit amet quis magna. Aenean velit odio, elementum in tempus ut, vehicula eu diam. Pellentesque rhoncus aliquam mattis. Ut vulputate eros sed felis sodales nec vulputate justo hendrerit. Vivamus varius pretium ligula, a aliquam odio euismod sit amet. Quisque laoreet sem sit amet orci ullamcorper at ultricies metus viverra. Pellentesque arcu mauris, malesuada quis ornare accumsan, blandit sed diam.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id. Sed rhoncus, tortor sed eleifend tristique, tortor mauris molestie elit, et lacinia ipsum quam nec dui. Quisque nec mauris sit amet elit iaculis pretium sit amet quis magna. Aenean velit odio, elementum in tempus ut, vehicula eu diam. Pellentesque rhoncus aliquam mattis. Ut vulputate eros sed felis sodales nec vulputate justo hendrerit. Vivamus varius pretium ligula, a aliquam odio euismod sit amet. Quisque laoreet sem sit amet orci ullamcorper at ultricies metus viverra. Pellentesque arcu mauris, malesuada quis ornare accumsan, blandit sed diam.

Figura 26. Pantalla de créditos.

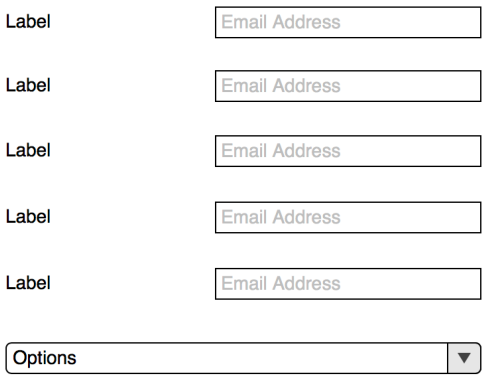


Figura 27. Pantalla de creación de encargos.

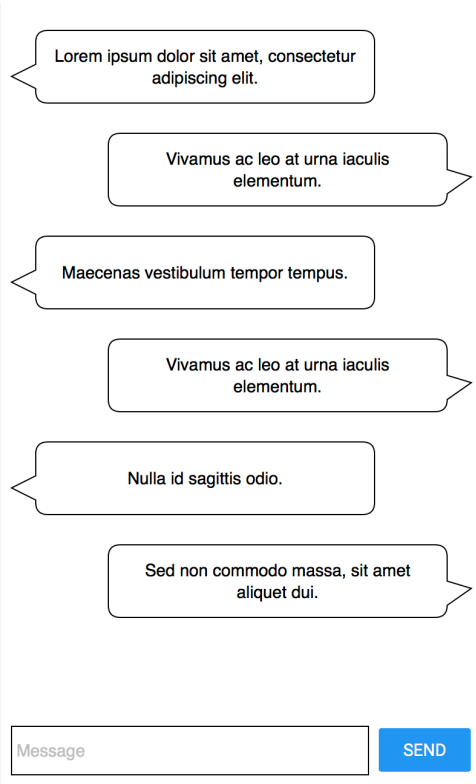


Figura 28. Pantalla de chat con otro usuario.

Es posible ver que se han intentado seguir los diseños al máximo nivel de detalle, teniendo en cuenta que son unos diseños con una especificación de detalle muy justa y que para dar el máximo nivel de detalle es necesario usar elementos nativos, los cuales hacen que varíe el diseño final en pequeña medida.

Ahora se va a describir y resaltar la funcionalidad de cada uno de los *mockups* presentados.

En la Figura 22 nos encontramos con la primera impresión de la aplicación. Esta es la ventana de bienvenida de la aplicación. Cuenta con una imagen en la parte derecha de la pantalla, y dos botones en la parte izquierda. Ambos te envían a la ventana de inicio de sesión, pero una en modo de inicio de sesión, y la otra en modo de registro de nuevos usuarios.

La ventana de inicio de sesión se puede observar en la Figura 23. Es una simple ventana con dos campos de introducción de texto, un campo de *checkbox*, y un botón.

Una vez se completado el inicio de sesión, la pantalla que salta a la acción es la representada por la Figura 24. En la que se pueden ver una serie de botones en modo imagen. Cada botón abrirá una ventana de las distintas actividades que son necesarias en la aplicación. El número de elementos no tiene por qué ser el final que se implementará. Esta ventana será la del menú principal de la aplicación.

Como cliente se pueden hacer encargos de pedidos, y chatear. Se puede observar que la pantalla de generar un encargo, representada por la Figura 27, cuanta con un formulario básico de introducción de datos, muchos campos de texto, y un selector.

Como trabajador está la funcionalidad de mostrar los trabajos disponibles en el mapa. Esta funcionalidad está representada en la Figura 21.

La pantalla de chat es común a trabajadores y usuarios, ya que tiene una función común. El mockup que la representa está etiquetado como Figura 28.

Básicamente muestra mensajes a ambos lados, y tiene un campo de envío de texto, que se acciona con un botón.

La Figura 25 es la encargada de describir la pantalla del perfil de usuario. Esta ventana nos mostrará la información de usuario con posibilidad de cambiarla en cualquier momento.

La última ventana a mostrar es la de créditos. Esta es la más sencilla. Se muestra todo el texto que referencia las utilidades y herramientas utilizadas en la aplicación. Como se puede observar es puro texto. Viene representada por la Figura 26.

8. Implementación

Durante toda esta etapa se va a contar con un pequeño nivel de detalle técnico todo lo realizado para hacer la app funcional.

En este apartado se van a proveer pequeños *snippets* de código a modo de ejemplo para ayudar a comprender la funcionalidad que se describe.

Aplicación Android

Figura 29. Icono de la aplicación sobre Android.

Aquí se van a contar detalles acerca de conceptos importantes que son usados en la aplicación. Estos conceptos son ampliamente usados por casi cualquier aplicación, por lo que el ejemplo puede dar una idea general acerca de cómo funciona, y del método correcto de uso de los mismos.

Como se puede comprobar en la Figura 29. Esa va a ser la apariencia de la aplicación instalada sobre sistemas operativos Android. El efecto rosáceo corresponde al recorte del color de fondo, que era un rosa muy llamativo.

Es importante comprender que los *snippets* irán variando de lenguaje. Para *snippets* de código Android se usará notación Java. Para definición de interfaces se usará XML.

8.1 Autenticación

Esta es una fase muy sencilla. Dado que Firebase cuenta con un sistema de autenticado fuertemente configurable desde su portal, se va a explicar cómo funciona y cómo se ha implementado este mecanismo.

El primer apartado es entrar a la consola de administración del proyecto en Firebase. Una vez dentro, en el apartado de *Authentication*, tenemos todos los parámetros de configuración que pueden observarse en la siguiente imagen separados por tres pestañas.

Es importante distinguir la clara diferencia que existe entre las distintas herramientas de la plataforma, y la solución que se ha utilizado para mitigar sus efectos. Los usuarios que obtienen una autenticación a través de esta herramienta no saben ni conocen nada de los datos de las demás plataformas. Es por eso que se utiliza el UID de un usuario, que automáticamente genera esta herramienta al obtener el usuario autenticado, para identificarlo en la base de datos en tiempo real. De este modo habrá unicidad entre ellos y podrán identificarse los métodos de inicio de sesión que utilice cada uno.

Un paso más allá puede ser la implementación de Firebase Analytics con los que todos estos datos los podemos tener monitorizados y realizar un análisis a fondo de los tipos de usuarios, sus acciones dentro de la aplicación, su método de llegada, y otra serie de datos interesantes a la hora de realizar tareas, por ejemplo, tareas de *Bussiness Intelligence*.

La primera a comentar debe ser la de “Método de inicio de sesión”. En la siguiente imagen puede verse más claro.

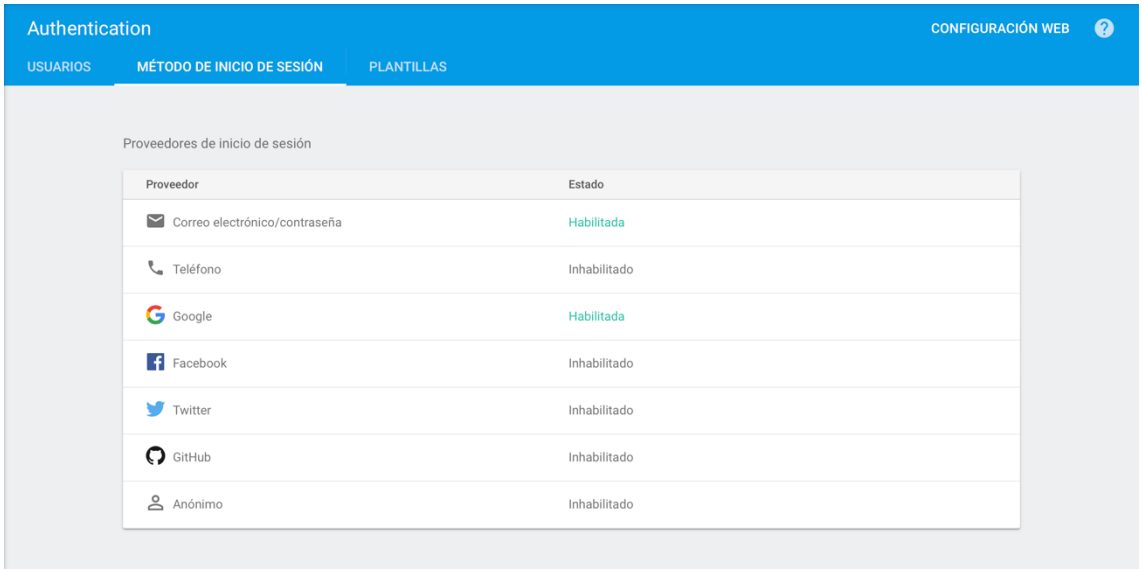


Figura 30. Consola de métodos de inicio de sesión de Firebase.

Como se observa en la Figura 30, es muy sencillo configurar las posibles fuentes de autenticación. Existe posibilidad de autenticación con correo electrónico, teléfono, Google, Facebook, Twitter, GitHub y anónimo.

En este proyecto se habilitó Google, que tiene una facilidad de implementación enorme, pero no se ha llegado a implementar, debido a que se cerró a un sistema de inicio de sesión a través de correo, con fin de gestionar internamente esos usuarios, y usarlos dentro de la base de datos para relacionar unos datos con otros.

La pestaña de plantillas permite configurar unas plantillas de texto que se enviarán a los usuarios en caso de que tengan que hacer uso del sistema integrado de confirmación del registro, y cambio de contraseña. Son simples plantillas que puedes ajustar a tu gusto.

La última pestaña y la más importante del sistema es la pestaña de “Usuarios”, desde la que se puede visualizar todos los usuarios registrados en el sistema. Se puede observar todos los usuarios generados para la app en la Figura 31.

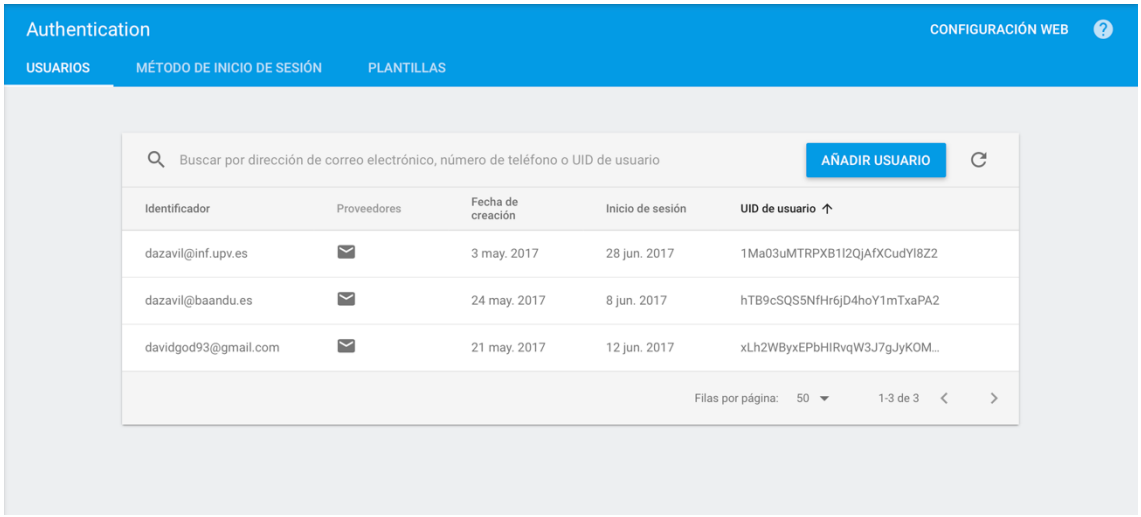


Figura 31. Consola de usuarios registrados en la aplicación.

Es importante recordar que la autenticación de Google genera un UID único para cada usuario. La aplicación es capaz de recuperar este parámetro en tiempo de ejecución, por lo que este parámetro es el que se usará en la base de datos Firebase para identificar al usuario, como se ha comentado antes.

De este modo, dentro de la aplicación, con fines de autenticación de un usuario (*mEmail*) y una contraseña (*mPassword*) se utilizaría el sistema efectivamente del siguiente modo:

```
auth.signInWithEmailAndPassword(mEmail, mPassword)
  .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull final Task<AuthResult> task) {
             //Código a ejecutar al terminar
        }
  });
```

En este caso la aplicación también provee de un sistema de registro desde la aplicación. Por ello el *snippet* de código siguiente registraría a un usuario y contraseña con los mismos parámetros del ejemplo anterior:

```
auth.createUserWithEmailAndPassword(mEmail, mPassword)
  .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull final Task<AuthResult> task) {
             //Código a ejecutar al terminar
        }
  });
```

También cuenta con fragmentos de código para funcionalidad de cambio de datos del usuario. De hecho, se han implementado en las funciones de cambio de información del usuario, por si cambia su correo electrónico, dado que este es su método de inicio de sesión principal, se actualice, y pueda iniciar sesión con este nuevo usuario.

Para dar un valor añadido al sistema de inicio de sesión, se ha implementado un mecanismo de autocompletado en el campo del usuario. Utiliza la agenda del dispositivo para proporcionar el autocompletado a partir de tus contactos.

El mecanismo de inicio de sesión además provee de la función de recordar el usuario. Esta función hace uso de las SharedPreferences, el mecanismo básico de almacenamiento en Android de pares clave-valor.

Al almacenar todos estos valores hace que el usuario entre directamente en la pantalla del menú principal sin necesidad de introducir sus datos, y saltándose la pantalla de bienvenida.

8.2 Mapas

La funcionalidad de mapas en la aplicación requiere de los servicios de Google y de la librería que importamos por Gradle, esta funcionalidad se provee fácilmente gracias al uso de la tecnología de *Fragments* de Android. Los *fragments* se pensaron para cuando era necesario tener en la misma actividad distintas interfaces de funcionalidad separada. Definiendo la interfaz de usuario, quedaría un elemento *fragment* insertado del siguiente modo entre las líneas de tu interfaz:

```
…

<fragment
  android:id="@+id/ams_map"
  android:name="com.google.android.gms.maps.SupportMapFragment"
  android:layout_width="match_parent"
  android:layout_height="match_parent"/>

…
```

Para hacer uso de los mapas de manera correcta, hay que realizar una serie de acciones en la actividad para inicializar el mapa:

```
.... implements OnMapReadyCallback {


@Override
protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_map_select);
       mapFragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.ams_map);
       mapFragment.getMapAsync(this);
}

@Override
public void onMapReady(GoogleMap googleMap) {
       googleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
       googleMap.setMyLocationEnabled(true);
}
```

En este *snippet* se puede observar que primero se obtiene una referencia al fragmento en el que se cargará el mapa. De este modo, se invoca el método pertinente para obtener de manera asíncrona el mapa. Puesto que este proceso depende de la calidad del internet y del dispositivo, debe obtenerse de manera asíncrona.

Al invocar al método getMapAsync(OnMapReadyCallback cb), el parámetro de implementar la interfaz OnMapReadyCallback. Esta interfaz contiene un método que se llama onMapReady(GoogleMap gm), el cual será invocado en el momento en que el mapa haya sido cargado con éxito dentro del fragment.

Es por eso que se le ha pasado el parámetro **`this`** a este método. Para que la propia clase implemente esa interfaz, y de ese modo se genere la llamada sobre la misma clase.

En esta clase también pueden observarse métodos de acceso a la ubicación del dispositivo para mostrar solo los trabajos más cercanos a tu ubicación. Utilizan la misma filosofía que la función anterior, se implementa una interfaz sobre la clase, que implementará un método definido por ella, y se generará un *callback* cuando la ubicación haya sido obtenida.

Con esto implementar una lógica de filtrado de encargos que no se encuentren cerca de ti es muy sencillo. Simplemente con utilizar funciones vectoriales para calcular la distancia en línea recta con el destino.

Dada la distancia entre dos puntos, dada por el módulo de un vector v con puntos s, d.

$$s = (39.504494, -0.447856)$$

$$d = (39.992695, -0.064523)$$

En este caso el sentido del vector no tiene importancia, dado que la distancia es la misma entre los dos puntos.

El módulo del vector se calcula aplicando la siguiente expresión:

$$dx = d(x) - s(x)$$

$$dy = d(y) - s(y)$$

$$v = \sqrt{dx^2 + dy^2}$$

En la que dx corresponde a la diferencia entre las coordenadas x de los puntos del vector y dy a la diferencia entre las coordenadas y.

Una vez se obtiene este valor (v), lo vamos a conocer como distancia relativa entre el trabajador y el encargo. Este será el factor determinante para decidir si un pedido le aparecerá a un usuario o no.

Este mismo valor es el que se usa para calcular el precio de un envío. Dado que un usuario cuando realizar un envío escoge el destino del paquete, y calcula la distancia relativa entre el destino y la dirección de recogida. De este modo lo usará como multiplicador de una constante para obtener el precio del trayecto.

Esta probablemente sea la vista más completa de funcionalidades a nivel tecnológico.

De este modo obtenemos una ventana que nos muestra los trabajos más cercanos al usuario de la siguiente manera, tal cual viene representado por la Figura 31a:

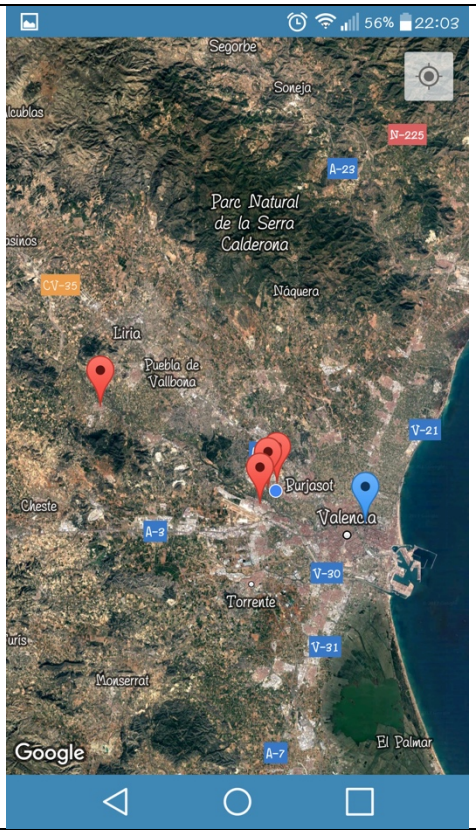


Figura 31a. Vista del mapa con los encargos cercanos.

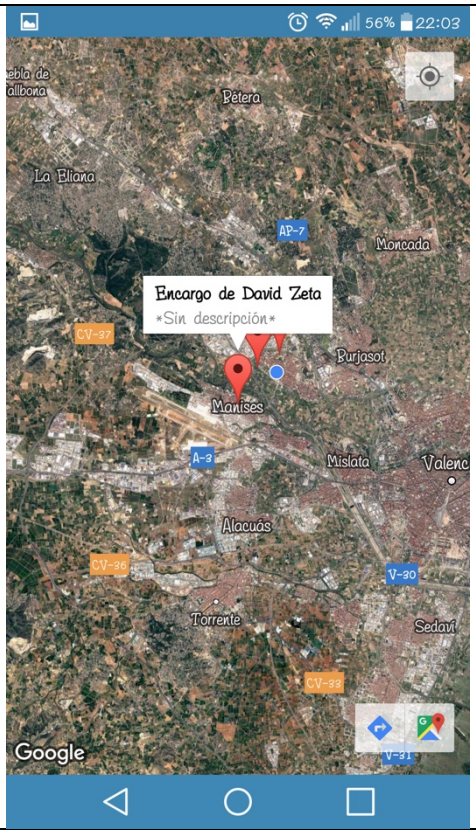


Figura 31b. Vista del mapa con un encargo seleccionado.

Como los trabajos más cercanos al trabajador, pueden estar ya asignados o no, el sistema los diferencia con dos colores claramente diferenciados. El rojo representa un trabajo sin asignar, mientras que el azul un trabajo ya asignado.

Por otro lado, como puede observarse en la Figura 31b, cuando seleccionas un encargo, este despliega una pequeña ficha con su información más representativa, Su descripción y su creador.

Como es de esperar, un encargo azul sobre el que se despliegue esta información, no tiene asociado ninguna acción, al contrario que el rojo, que te permite realizar acciones sobre este encargo.

8.3 Push

La metodología push normalmente cuenta con el apoyo de un servidor, el cual suele implementar un servidor REST que genera las peticiones al servidor push final y abstraer a la aplicación de la autenticación contra él, el manejo de usuarios y otras cuestiones.

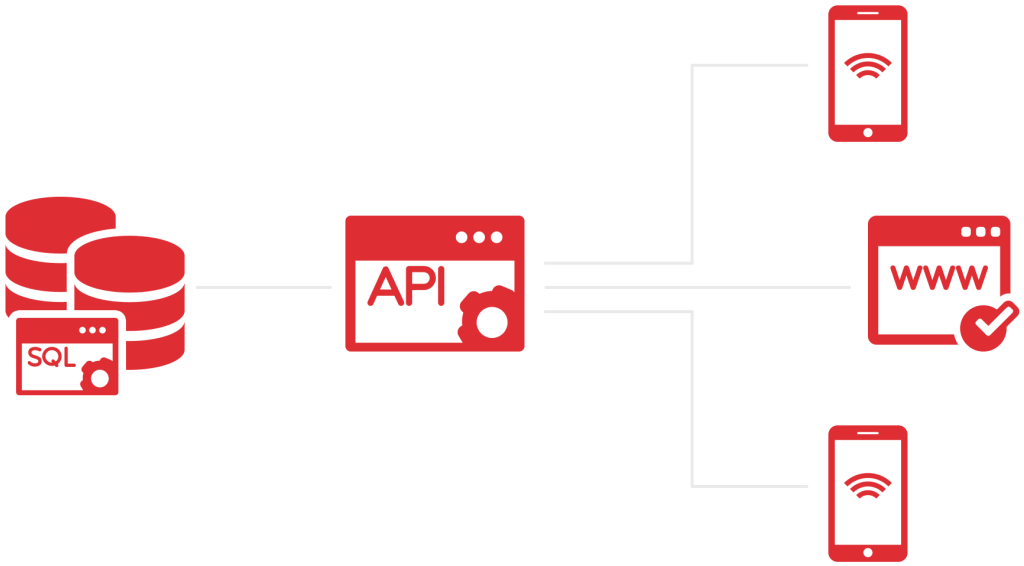


Figura 32. Esquema de una petición a una API.

En la Figura 32 se puede observar el esquema tradicional de la filosofía de una API. Tratémoslo como una especie de *middleware* que se encarga de proporcionar una funcionalidad a los dispositivos que a este se conectan, generando una implícita abstracción sobre la implementación interna que ese servicio tiene con los sistemas de almacenamiento, como la base de datos.

En este proyecto debido al reto de no haberlo realizado nunca y también debido a la restricción económica y temporal, se decidió prescindir de este. Es por eso que utiliza una metodología un tanto especial para realizar esta función.

Lo primero es tener en cuenta que es necesario un cliente para realizar peticiones HTTP. Android por defecto, al estar basado en Java permite usar sus implementaciones propias acerca de esto, tanto con las AsyncTask, los Threads, o URLConnection. Todos ellos cuentan con la peculiaridad de que hay que bajar a muy bajo nivel para poder realizar las operaciones.

Aparte de esto, existen una gran variedad de librerías que te abstraen de la implementación a bajo nivel y te proveen la funcionalidad deseada, por lo que nosotros usaremos una de ellas.

En este proyecto se ha decidido usar Volley, que es la librería que provee Google para realizar peticiones web sin necesidad de conocer los detalles de implementación de una petición básica. Una expresión muy válida para su justificación es que veo innecesaria la necesidad de reinventar la rueda.

De este modo, el siguiente *snippet* presenta el modo en el que se realizan este tipo de peticiones desde el dispositivo.

```
private static final String API_TOKEN = "AAAA4jOIAwU:.....";
public static void sendPush(Context c, String token, JSONObject data,
Response.Listener<JSONObject> responseListener, Response.ErrorListener
errorListener) {
   try {
      RequestQueue r = Volley.newRequestQueue(c);
      JSONObject req = new JSONObject();
      req.put("to", token);
      req.put("data", data);
      r.add(new JsonObjectRequest(Request.Method.POST,
"https://fcm.googleapis.com/fcm/send", req, responseListener, errorListener)
{
         @Override
         public Map<String, String> getHeaders() throws AuthFailureError {
            Map<String, String>  params = new HashMap<>();
            params.put("Content-Type", "application/json");
            params.put("Authorization", "key="+API_TOKEN);
            return params;
         }
      });
   } catch (Exception e) {
      Logger.error("Error enviado mensaje push. "+e.toString());
   }
}
```

Este fragmento de código que se acaba de mostrar tiene una complejidad importante, pero se va a abordar con detenimiento y detalle.

Volley trabaja con RequestQueue, que es una clase que provee la funcionalidad de añadir peticiones web y abstraer de cómo y de qué manera se tiene que hacer.

En la RequestQueue se ha añadido una implementación de la Request básica, que es la JsonObjectRequest. Se le ha indicado que realizará el paso de parámetros por POST, sobre la url de FCM y su parámetro será el objeto JSON req. Los dos siguientes parámetros serán las funciones/callbacks a ejecutar cuando pase la situación de éxito o error.

El parámetro contiene dos campos. El del identificador (*token*) destino de dispositivo al que notificar, y el campo de datos que este mensaje tendrá (*payload*)

Puede observarse que se ha hecho un reemplazo *inline* de un método implementado por defecto, al cual se necesitaba tener acceso para pasarle los dos parámetros que se observa a la cabecera. Ya que FCM no acepta peticiones que no vengan identificadas con un *API token*, que es una clave de acceso para restringir el acceso a enviar notificaciones a cualquier usuario.

En el responseListener básicamente se leerá la respuesta que ofrece el servidor, que es una respuesta acerca de cuáles dispositivos ha conseguido notificar y cuáles no.

La siguiente imagen propiedad de PacketZoom explica bastante bien el proceso interno que usa Volley para realizar sus peticiones a los servicios web e invocar los *callbacks* cuando reciba los resultados.

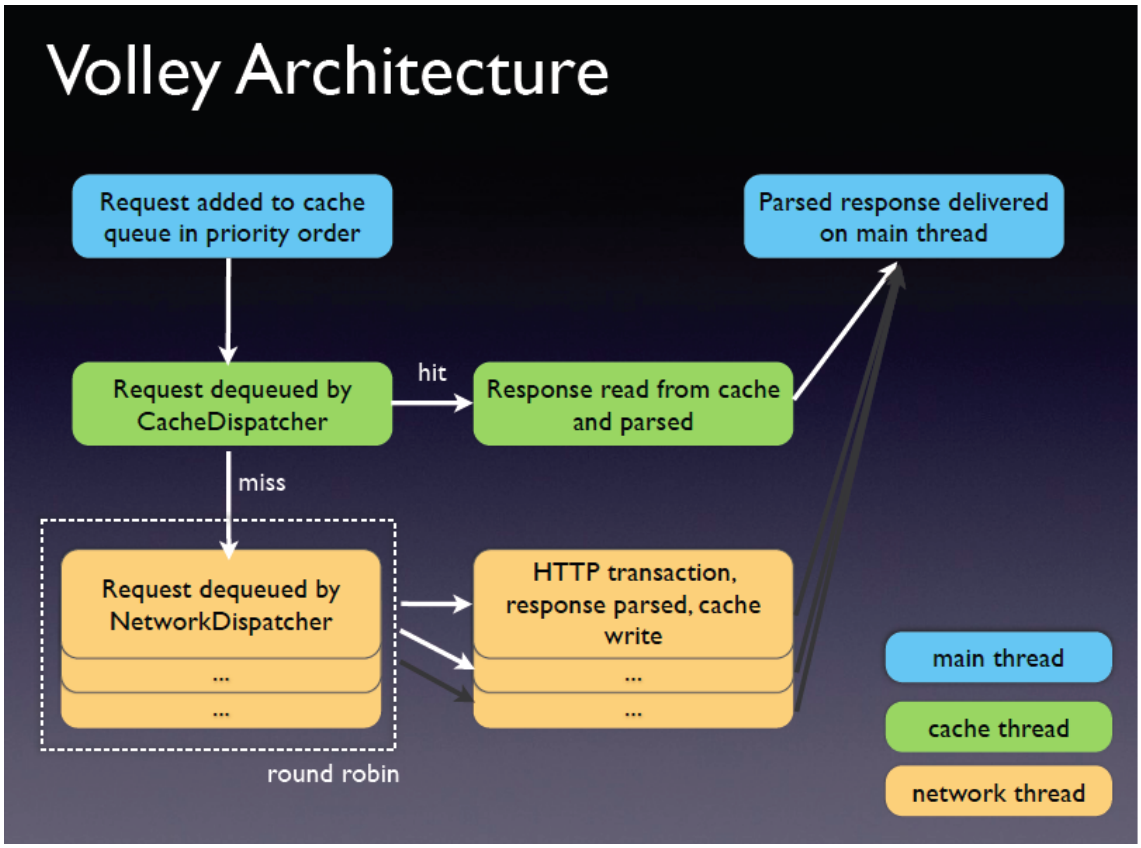


Figura 33. Funcionamiento interno de Volley.

Como describe la anterior Figura 33, las peticiones a las que tienes implícito acceso desde la librería, se ejecutan sobre el hilo principal, primera abstracción que provee al hacer que te olvides de la concurrencia, la gestiona él internamente. Tras esto, otro encargado de mantener un registro de caché, con el fin de mejorar los tiempos de respuesta y la eficiencia de consumo de datos es el siguiente hilo. En este hace una consulta para verificar que la petición no se ha realizado ya, por si acaso hubiese una copia válida en la caché. En caso de tenerla simplemente la devolverá, y se encargará el hilo principal de gestionar esa misma respuesta y devolverla al usuario a través de un *callback*. Si esta petición no hubiese estado en la caché, hubiese bajado un nivel más hasta hacer la petición al encargado de realizar el acceso a la red. El hilo encargado de acceder a la red hubiese encolado la petición en una cola interna de Volley. Esta cola se gestiona internamente aplicando un algoritmo Round Robin para la obtención de los elementos encolados. Una vez el responsable haya desencolado la petición, se realiza la transacción HTTP para obtener la respuesta enviada por el host destino.

A este esquema le faltaría un penúltimo paso, que es el del tratamiento de la respuesta. Volley por defecto, la petición más simple es la petición que devuelve un String, pero cuenta con otro tipo de valores a devolver. El String incluso también requiere un proceso de transformación del origen de la respuesta que sería un *byte-stream*, lo que en resumen es un flujo de bytes en crudo. Este proceso de transformación no ha sido tenido en cuenta en este diagrama, y cabe la pena nombrarlo, ya que realiza ese proceso de transformación sobre la respuesta dependiendo del tipo de petición realizada.

Cabe destacar también, por ejemplo, en la petición de tipo JSON, que se requiere que en la cabecera de la respuesta HTTP se especifique que el *Content-type*, o lo que es lo mismo, el contenido de la respuesta es de tipo JSON a través de su especificación de tipo MIME, *application-json*. De lo contrario Volley lo interpretará como un comportamiento inesperado.

Tras este último proceso, devuelve el objeto de respuesta demandado y le pasa la acción al hilo principal a través del *callback* facilitado a la petición.

8.4 Firebase

Este ha sido el punto más difícil de hacer funcionar correctamente, dado que Firebase utiliza un mecanismo de *callbacks* cuando consigue una lectura efectiva del resultado. De este modo soluciona la concurrencia de acceso a los valores de los datos solicitados, facilitando la lógica de la aplicación en cuanto a concurrencia se refiere, pero la complica en cuanto a control de lecturas se refiere.

Una vez hecho a la metodología esta, el control lógico de sus lecturas es sencillo. El siguiente *snippet* de código muestra un ejemplo de lectura de un elemento de la base de datos, que el mismo acaba en escritura de otro campo.

```
FirebaseDatabase.getInstance().getReference("empleados").child("david")
.addValueEventListener(new ValueEventListener() {
   @Override
   public void onDataChange(DataSnapshot dataSnapshot) {
      try {
         JSONObject j = parseDBResponse(dataSnapshot);
         int i = j.getString("salario");
         i *= 2;
         dataSnapshot.getReference("salario").setValue(i);
      } catch (Exception e) {
         onCancelled(DatabaseError.fromException(e));
      }
   }

   @Override
   public void onCancelled(DatabaseError databaseError) {
      Logger.error(databaseError.toString());
      Toast.makeText(MapSelectionActivity.this, R.string.error_near_orders,
Toast.LENGTH_SHORT).show();
      finish();
   }
});
```

Este sería el ejemplo más funcional de uso de Firebase Database. Obtener referencias a los hijos de la raíz, y desde ahí navegar hasta que hijo que busques. El método parseDBResponse es una función encargada de meter la respuesta de Firebase en un objeto JSON que es más fácil de tratar, por defecto devuelve una respuesta de tipo conjunto de clave-valor en un Map<String, Object>.

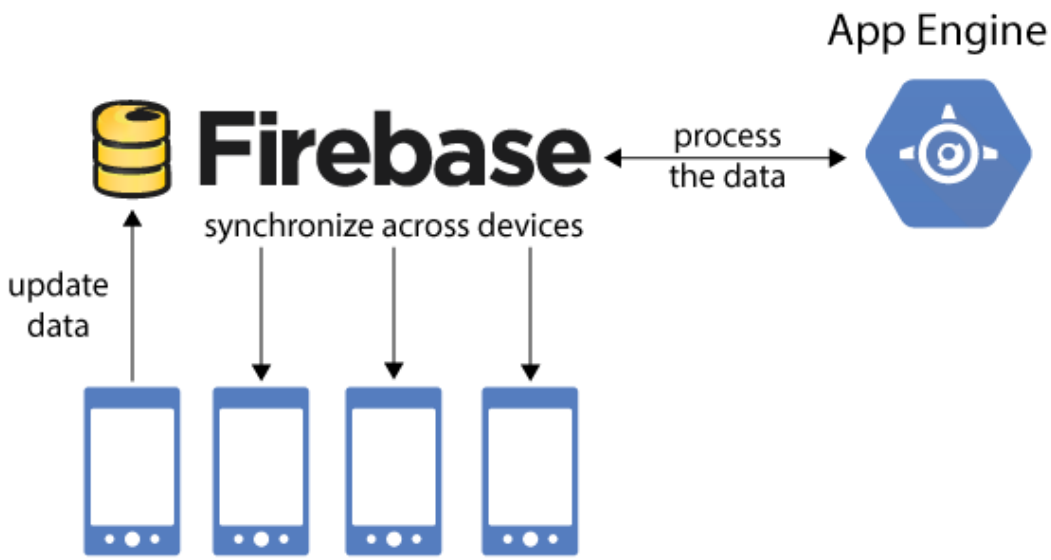


Figura 34. Esquema de funcionamiento interno de Firebase.

Firebase Database, es un sistema comercializado como *Realtime Database*, lo cual significa como se ha comentado en la introducción que las actualizaciones de su base de datos se propagan de manera automática y en tiempo real a todos los dispositivos que estén escuchando sobre el nodo en que se haga el cambio. Es por esto que hay que tener cuidado con el modo de escucha. Porque si no se elimina el *listener*, se podría generar un *callback* y la aplicación ya no tener contexto, por lo que generaría una excepción.

Como puede observarse en la Figura 34, Firebase y su mecanismo interno se puede comunicar con otras plataformas para hacer tareas de servidor, pero su característica importante destacable es la de la sincronización con los dispositivos, lo cual implica que los datos se actualicen en tiempo real en todos los dispositivos automáticamente, sin establecer lógica extra, pero como hemos comentado antes, habría que tener en cuenta los casos de pérdida de contexto.

Además, por estos motivos provee dos métodos con comportamiento distinto, pero con objetivo similar: Realizar una lectura múltiple (actualizaciones de contenido) o una lectura única (lee el valor y termina la escucha). Ambos ellos reciben su respuesta a través del *callback*, incluyendo en un parámetro de tipo DatabaseReference el resultado.

8.5 SQLite

SQLite es motor de base de datos que se ha utilizado para almacenar los mensajes de manera ordenada y que sea sencilla su inserción, obtención. De los 4 principios *CRUD*, usa 2, por lo que simplifica su uso. Gracias a los principios de una base de datos relacional, a través de una consulta con algebra relacional es posible sacar todos los datos de la base de datos ordenados.

La consulta SQL se genera simplemente a partir de las siguientes consultas definidas en constantes de texto.

La única tarea que tiene esta base de datos es almacenar los mensajes de chat con otros usuarios. Es por eso que cuenta con un diseño tan simple y fácil.

Identifica las tablas por los *tokens* de los destinatarios de los mensajes. De este modo, si existe la tabla, hay conversaciones, y si no, no las hay. La ejecución de la consulta SQL_CREATE_CMD genera los esquemas.

```
private static final String SQL_TABLE_KEY = "%dbn%",
      SQL_CREATE_CMD = "create table if not exists T"+ SQL_TABLE_KEY +" (id
integer primary key, message text, creationtime integer, ismine integer);",
      SQL_DELETE_ENTRIES = "drop table if exists T"+ SQL_TABLE_KEY +";",
      SQL_SELECT_QUERY = "select * from T"+SQL_TABLE_KEY+" order by
creationtime asc;";
```

Las otras dos consultas, al ejecutarlas obtienen los siguientes comportamientos. La consulta SQL_DELETE_ENTRIES es la encargada de eliminar la tabla. Se usa con funciones de desarrollo.

La consulta SQL_SELECT_QUERY es la consulta de recuperación de los datos de la tabla marcada. Se recuperan ordenados por su parámetro de creationtime, que representa el instante de envío del mensaje, para la obtención de los mensajes ordenados en el tiempo.

La inserción de elementos se hace gracias a los métodos que provee SQLite que te dota de un método insert(), sobre el objeto SQLiteDatabase. Pasándole el identificador de la tabla sobre la que hacer la inserción y los valores que deseas insertar sobre esta. Se realizaría del siguiente modo:

```
SQLiteDatabase db = getWritableDatabase();
db.execSQL(SQL_CREATE_CMD.replace(SQL_TABLE_KEY, hash(tName)));
db.insert("T"+hash(tName), null, getInsertContentValues(c));
db.close();
```

Como puede observarse, antes de almacenar el mensaje en la tabla, se crea si no existe la tabla, para evitar errores de almacenamiento.

También puede observarse que las consultas se reaprovechan para todos los chats, dado que se le ha insertado una palabra clave reservada, que con una alta probabilidad no aparecerá en la consulta, para hacer un reemplazo en la consulta de esa clave por el nombre de la tabla, de este modo siempre se obtiene un método limpio y sin errores de generar consultas.

Para evitar colisiones, aunque en principio no deberían suceder, se hace el cálculo de una función hash sobre el nombre de la tabla, etiquetado en el código como tName, el cual, en base, es el UID del usuario destino. Como se observa en las definiciones, se hace el truco antes comentado de insertar una “T” al principio del nombre de la tabla, ya que la función hash puede generar un nombre que comience por un número, algo que ya se ha comentado que está prohibido en SQLite.

El resultado final de la aplicación se podrá observar en el Apéndice B de esta memoria, la cual contiene un manual de usuario acerca del uso y funcionamiento de la aplicación.

9. Conclusiones

Llegados a este punto, se va a comentar de manera subjetiva todos los avances con la aplicación, hasta donde ha llegado contando el alcance inicialmente planificado. Cerrará el apartado una opinión personal acerca del proyecto y la memoria.

9.1 Evaluación de la metodología

La evaluación del resultado de la metodología DCU genera unos resultados muy positivos tras aplicarse. El proyecto se ha ceñido a la metodología, lo que ha permitido que la metodología consiga unos resultados tan evidentes.

9.2 Que se ha alcanzado

En este proyecto el resultado es claramente visible, dado que si no se hubiesen alcanzado los objetivos no habría habido una demostración funcional del producto. Por lo que yo creo que ha sido bastante positivo el resultado.

Se ha alcanzado todo lo planificado y no ha habido que hacer trabajo de más, ha sido todo aprovechable. Todo esto ha sido gracias a haber seguido una metodología, en este caso el Diseño Centrado en el Usuario.

Todos los objetivos creo que han sido cubiertos teniendo en cuenta el alcance al que pensaba que iba a llegar. La realidad es que se ha obtenido un producto totalmente funcional, con unas características interesantes para el cliente final y que tras un estudio de mercado se ha comprobado que tendría un hueco de mercado si se implantase.

9.3 Dificultad del proyecto

El proyecto como tal es uno más dentro de otros tantos, pero este ha contado con muchos contratiempos y muchas más restricciones de lo normal, lo que lo ha colocado como un proyecto duro de roer.

Se puede contar como un proyecto que se ha ceñido a la metodología provista por Firebase y que esto ha limitado en gran medida el rendimiento de la app, ya que no se trata de una simple app de lectura y representación de datos como otras muchas.

Dentro de lo que se puede esperar de un proyecto, este proyecto ha tocado una gran cantidad de funciones de Android y muy variadas en cuanto a lo que ofrecen.

Algunas de las funciones de este proyecto, como la base de datos local, la base de datos implementada completamente en Firebase, la ausencia de una API para obtener los datos, y el envío de mensajes push directamente al servidor de Google han sido algunos de los retos que yo califico de más importantes de este proyecto.

9.4 Opinión personal

Creo que este proyecto me ha ayudado a ahondar en los conocimientos que tenía sobre Android, y a realizar un proyecto completo en tiempo récord.

A pesar del poco tiempo del que he dispuesto para realizar este trabajo, por motivos laborales, la planificación previa ha ayudado a no tener que perder tiempo en cosas que no son productivas

para el avance del proyecto, y esto ha permitido el proyecto haya podido ser acabado funcionalmente al 100% y entregado dentro de tiempo.

Gracias a la ayuda de mi tutor, el cual me ha guiado a la hora de perfilar detalles en el proyecto, ha sido posible generar este producto final.

Bibliografía

- Apuntes recogidos en la asignatura DCU de cuarto curso, en la rama de especialización de Tecnologías de la Información, del Grado en Ingeniería Informática de la Universidad Politécnica de Valencia. (2016)
- Guenveur, Lauren (2017). Cuota de mercado de Android: http://es.kantar.com/tech/móvil/2017/abril-2017-cuota-de-mercado-de-smartphones-en-españa/
- Navarro Marset, Rafael (2007). Servicios web REST: http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf
- Tomás Gironés, Jesús (2016). El gran libro de Android: https://www.casadellibro.com/libro-el-gran-libro-de-android-5-ed/9788426722560/2620988
- Firebase Official Documentation (2017). Package Index: https://firebase.google.com/docs/reference/android/packages
- Firebase Official Documentation (2017). Muestras de código: https://firebase.google.com/docs/samples/#android
- Firebase Official Documentation (2017). Firebase Cloud Messaging (descendentes): https://firebase.google.com/docs/cloud-messaging/downstream
- Official SQLite Documentation (2017). SQLite Datatypes: https://sqlite.org/datatype3.html
- Official SQLite Documentation (2005). SQLite SQL Syntax: https://sqlite.org/lang.html

Enlaces de interés:

- Mockups: https://moqups.com
- Firebase: https://console.firebase.google.com/u/0/
- Diagramas de clases: https://creately.com/app/

Apéndice

En este apartado de la memoria se va a exponer de manera ligera el resto del contenido que no se ajustaba al contenido de los demás puntos, o bien por longitud, o bien por el tamaño de los mismos. De este modo puede consultarse tanto las encuestas realizadas a los usuarios, como el manual de usuario de la aplicación. Este mismo cuenta como resultado final de la implementación ya que cuenta con todas las capturas necesarias de la aplicación.

Apéndice A. Encuestas y resultados.

Preguntas de la encuesta

La encuesta está formada por las siguientes preguntas que se realizaron desde mi propio móvil apuntando las respuestas que los usuarios me facilitaban:

1. ¿Te gusta comprar por Internet? *
- ○ Sí
- ○ No

2. ¿Alguna vez has realizado una compra a través de u e-commerce? (Amazon, eBay, etc) *
- ○ Sí
- ○ No

3. ¿Ves interesante estar al tanto del estado de tus pedidos? *
- ○ Sí
- ○ No

4. ¿Envías paquetes? *
- ○ Sí
- ○ No

5. ¿Te resulta cómodo tener que ir a una tienda a entregar un paquete para enviarlo? *
- ○ Sí
- ○ No

6. ¿Preferirías que viniesen a recogerlo a tu casa? *
- ○ Sí
- ○ No

7. ¿Alguna vez has tenido una incidencia con algún pedido? *

- ◯ Si
- ◯ No

8. ¿Encontrarías útil tener un canal directo de comunicación con tu vendedor? *

- ◯ Sí
- ◯ No
- ◯ Tal vez

9. ¿Encontrarías útil tener un canal directo de comunicación con tu comprador? *

- ◯ Sí
- ◯ No
- ◯ Tal vez

10. ¿Encontrarías útil tener un canal directo de comunicación con tu repartidor? *

- ◯ Sí
- ◯ No
- ◯ Tal vez

11. ¿Verías interesante todas estas anteriores funciones juntas en una misma aplicación? *

- ◯ Sí
- ◯ No

12. ¿Crees que usarías esta aplicación si estuviese en el mercado? *

- ◯ Sí
- ◯ No
- ◯ Tal vez

13. ¿Qué edad tienes?

- ◯ 15-20
- ◯ 21-25
- ◯ 26-30
- ◯ 30 o más

14. Género

- ○ Hombre
- ○ Mujer

Como se puede observar, todas las preguntas de la encuesta son de respuesta simple, no requieren mucho tiempo, y sus respuestas consiguen una información muy valiosa para el estudio de mercado.

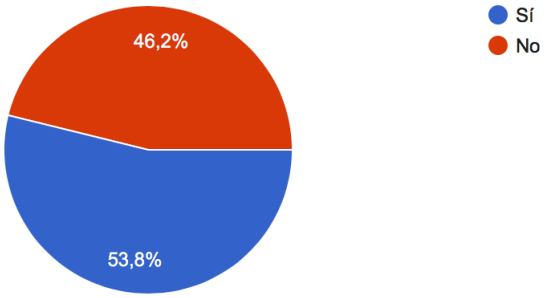
Respuestas de la encuesta

Las respuestas obtenidas se representan en gráficos de quesito para cada respuesta, por el hecho de tratarse de preguntas con no muchas opciones de respuesta múltiple. De este modo ese gráfico proporciona la información de manera gráfica muy representativa para ver a simple vista las respuestas mayoritarias y la tendencia de la gente.

Estos son los resultados para cada respuesta:

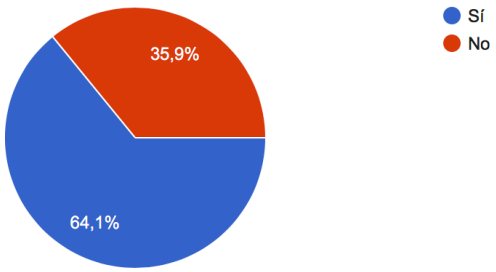
1. ¿Te gusta comprar por Internet?

39 respuestas



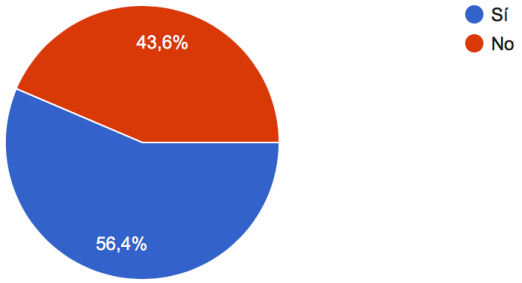
2. ¿Alguna vez has realizado una compra a través de u e-commerce? (Amazon, eBay, etc)

39 respuestas



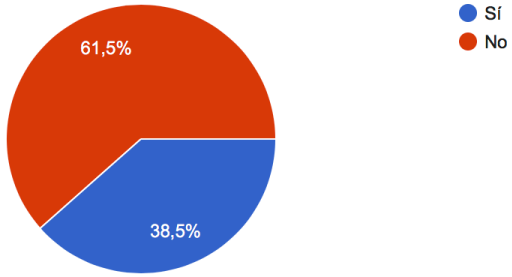
3. ¿Ves interesante estar al tanto del estado de tus pedidos?

39 respuestas



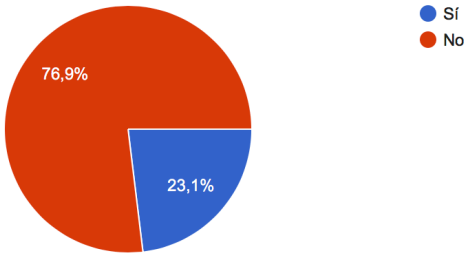
4. ¿Envías paquetes?

39 respuestas



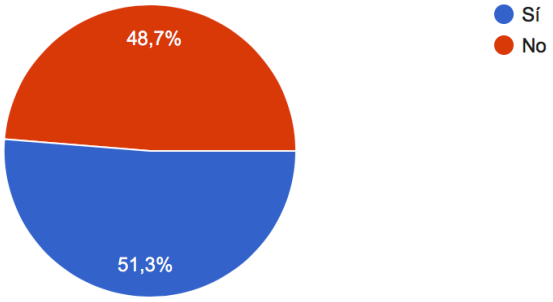
5. ¿Te resulta cómodo tener que ir a una tienda a entregar un paquete para enviarlo?

39 respuestas



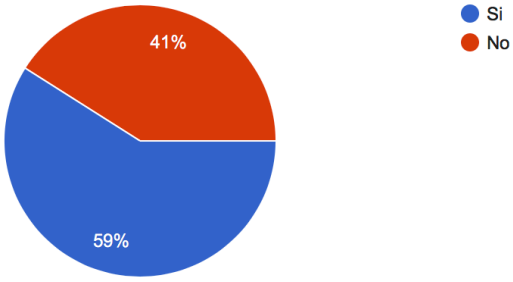
6. ¿Preferirías que viniesen a recogerlo a tu casa?

39 respuestas



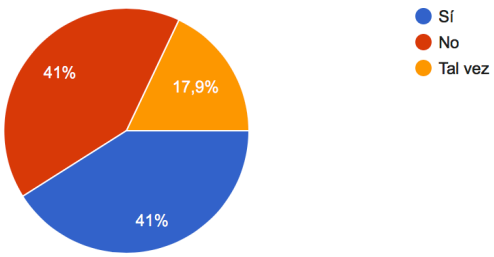
7. ¿Alguna vez has tenido una incidencia con algún pedido?

39 respuestas



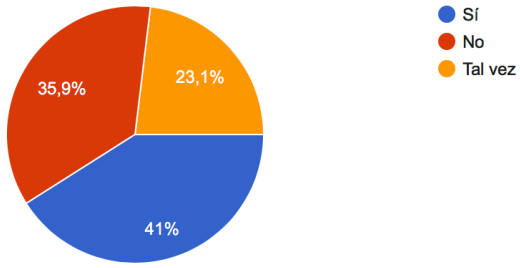
8. ¿Encontrarías útil tener un canal directo de comunicación con tu vendedor?

39 respuestas



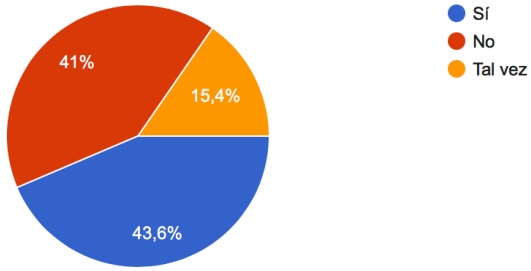
9. ¿Encontrarías útil tener un canal directo de comunicación con tu comprador?

39 respuestas



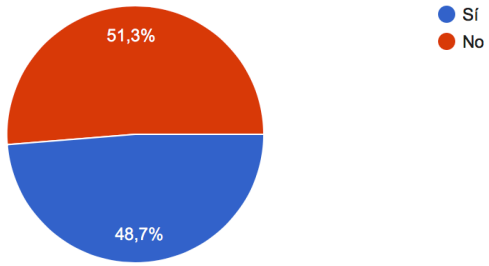
10. ¿Encontrarías útil tener un canal directo de comunicación con tu repartidor?

39 respuestas



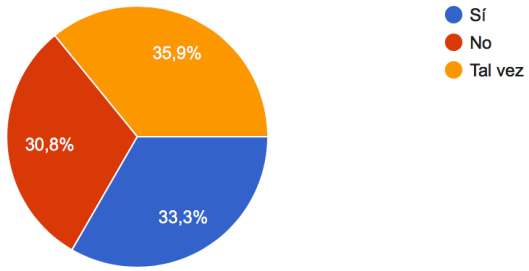
11. ¿Verías interesante todas estas anteriores funciones juntas en una misma aplicación?

39 respuestas



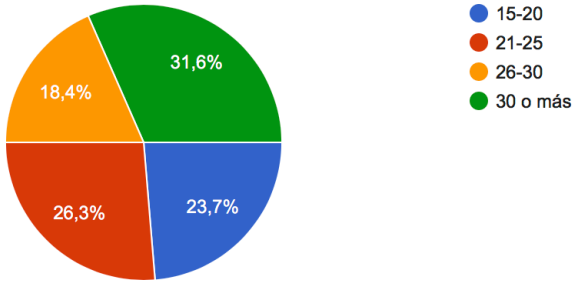
12. ¿Crees que usarías esta aplicación si estuviese en el mercado?

39 respuestas



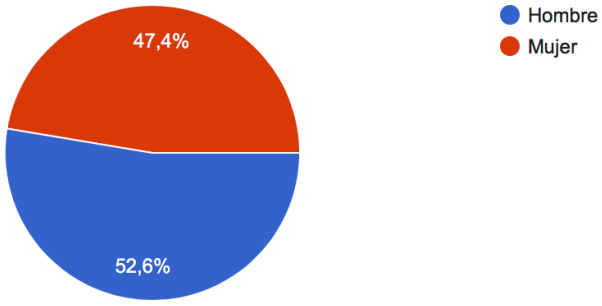
13. ¿Qué edad tienes?

38 respuestas



14. Género

38 respuestas



Los resultados que se pueden observar en estas encuestas son los mismos que se exponen anteriormente en la memoria, en el momento del análisis de resultados del DCU, es por eso que se ha preferido poner esta documentación al final, para facilitársela al lector.

Apéndice B. Manual de usuario.

Aquí se van a exponer cada una de las pantallas de la aplicación a modo de manual, para la fácil y correcta compresión de la funcionalidad que ofrece la aplicación de cara a los usuarios.

Todas las imágenes contenidas en este apéndice se van a nombrar y numerar siguiendo una nomenclatura especial, no van a corresponder con las anteriores figuras de la memoria, sino que se van a etiquetar como Figura An, donde n será el número de figura, p.ej. Figura A1, Figura A2… y así sucesivamente.

Dado que esto tiene el objetivo de servir como manual de usuarios se van a obviar muchos conceptos y se van a realizar muchas aserciones de cosas ya explicadas en el documento.

Esto solo servirá de modo funcional para describir el comportamiento de cada función de la aplicación. De este modo podrá reforzarse los conocimientos obtenidos en los apartados anteriores de esta memoria.

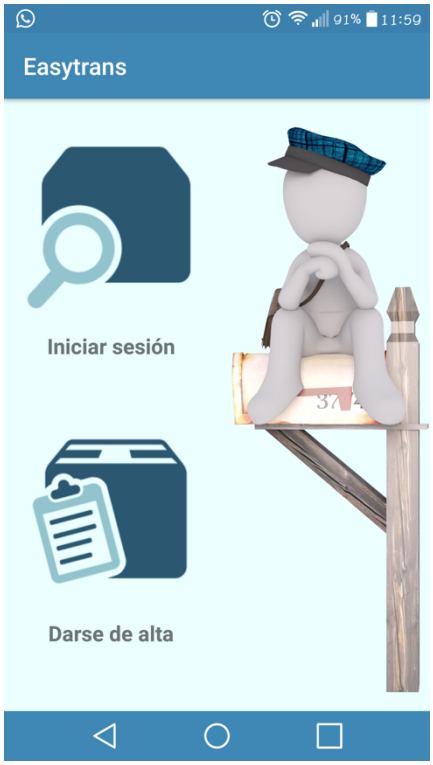
Por último, se desea hacer referencia a que el código de esta aplicación se encuentra disponible en GitHub publicado para el apoyo del conocimiento expresado en esta memoria, dado que algunos conceptos se entienden mejor con ejemplos, pero esta memoria no puede albergar todo el código.

El acceso al repositorio de GitHub se encuentra disponible bajo la siguiente URL:

https://github.com/davidgod93/HomePost

Es posible que dentro del código se encuentren referencias *HomePost*, incluyendo en propio nombre del repositorio. El motivo de esto, es que era el anterior nombre que recibió esta aplicación, pero en última instancia se ha decidido cambiar por el nombre actual por deseo propio. Simplemente me parecía más atractivo y más comercial.

Figura A1.1: Pantalla de bienvenida.

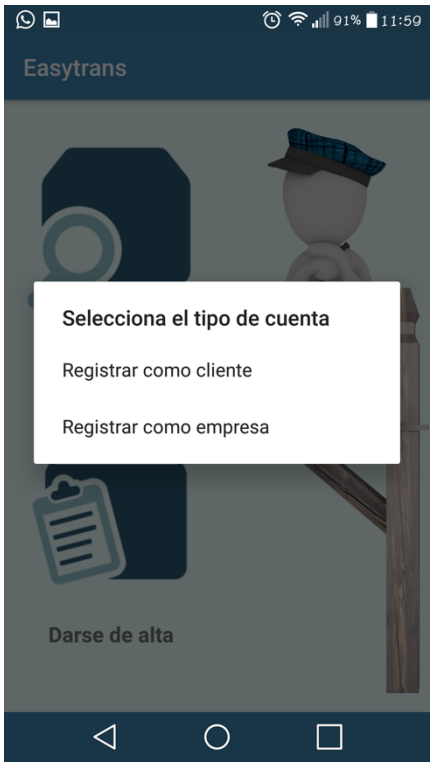


Esta es la primera pantalla que te encuentras nada más entrar a la aplicación.
Esta pantalla tiene dos funciones:

- Iniciar sesión de un usuario ya existente.
- Registrar un nuevo usuario en el sistema.

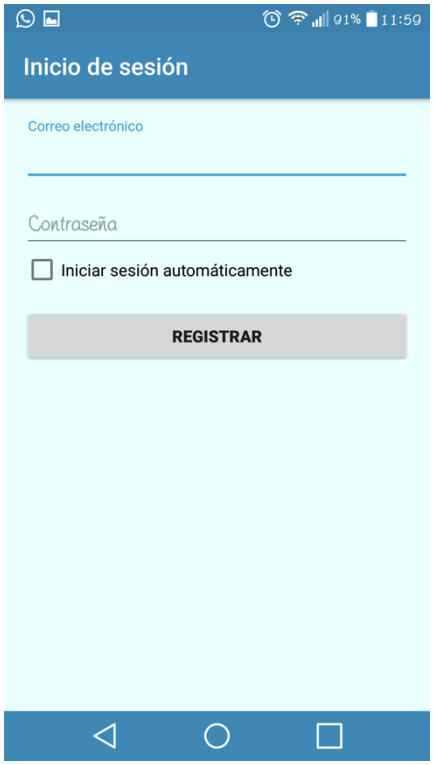
Ambas dos funciones accesibles desde las imágenes etiquetadas para ello.
La imagen de la derecha simplemente realza el diseño de la pantalla.

Figura A1.2: Pantalla de bienvenida. Tipo de registro.



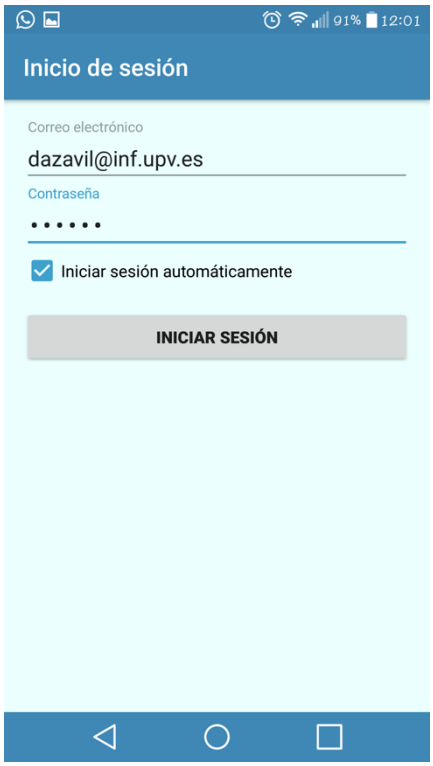
En la opción de registro es posible escoger si quieres registrarte como cliente o como trabajador.
De este modo te aparecerán distintas opciones en los menús que más adelante se verán.

Figura A2.1: Pantalla de inicio de sesión. Registro.



Esta es la pantalla de registro del usuario, una vez has escogido la opción de registro. O lo que es lo mismo, el tipo de usuario que serás.
Este tipo de usuario no es visible para el usuario en esta ventana, pero internamente tiene guardado el tipo que es, de este modo luego al crear el usuario se verá reflejado.

Figura A2.2: Pantalla de inicio de sesión. Inicio de sesión.



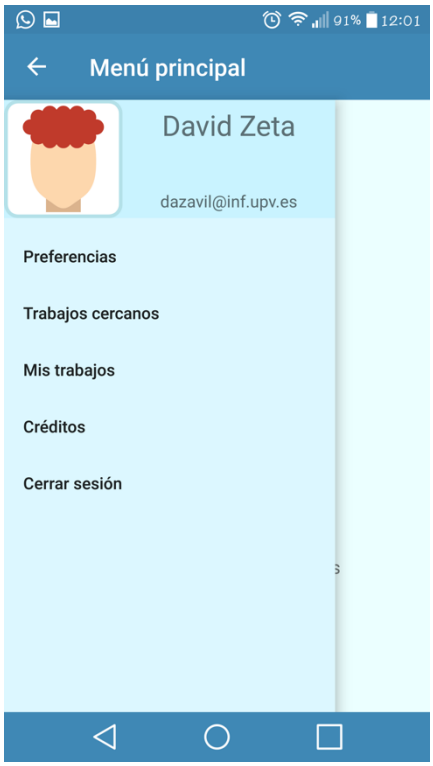
En esta ventana, como se puede apreciar, es la misma ventana a nivel funcional que la anterior. Ya hemos rellenado la información de inicio de sesión y hemos marcado el *checkbox* para almacenar en memoria y autocompletar cada vez las credenciales.

Figura A3.1: Pantalla de menú. Modo trabajador. Interfaz principal.

Esta es la pantalla principal del menú de un usuario de tipo trabajador. Como se puede observar los iconos variarán con respecto al otro tipo de usuario.
Ambos iconos tienen las funciones de abrir las ventanas responsables de realizar las tareas que indica, las cuales se verán más adelante en detalle:

- El icono de **Mostrar trabajos cercanos** corresponde a la Figura A7 y sus respectivas descendientes.
- El icono de **Mostrar estado de mis trabajos** corresponde a la Figura A8 y sus respectivas descendientes.

Figura A3.2: Pantalla de menú. Modo trabajador. Menú lateral.



Este es el menú lateral de la aplicación, que permite mantener organizado el menú, y no sobrecargarlo con elementos de uso reducido, aquellos que se usan con menos frecuencia.
Cada elemento enlaza con su correspondiente acción asociada:

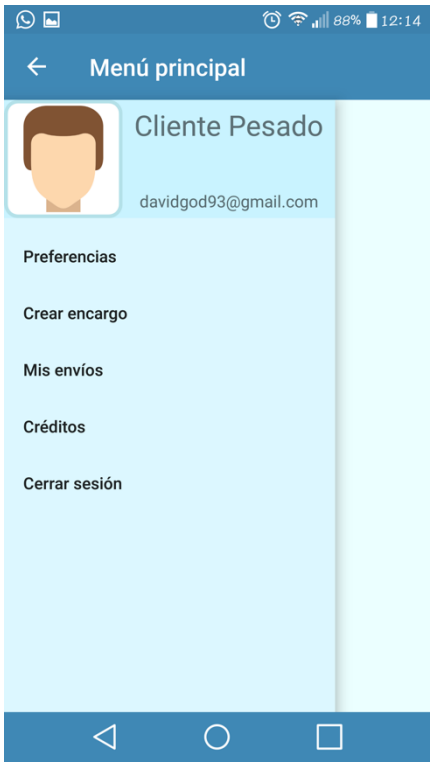
- El elemento **Preferencias** corresponde a la Figura A4 y sus respectivas descendientes.
- El elemento de **Trabajos cercanos** corresponde a la Figura A7 y sus respectivas descendientes.
- El elemento de **Mis trabajos** corresponde a la Figura A8 y sus respectivas descendientes asociadas.
- El elemento **Créditos** corresponde a la Figura A6.
- El elemento **Cerrar sesión** corresponde a la Figura 10.

Figura A3.3: Pantalla de menú. Modo cliente. Interfaz principal.

Esta pantalla plantea la misma filosofía que la explicada en la Figura A3.1, pero aplicada al uso para los clientes. En esta ventana la asignación corresponde del siguiente modo:

- El icono de **Organizar una recogida** corresponde a la Figura A7 y sus respectivas descendientes.
- El icono de **Mostrar estado de mis trabajos** corresponde a la Figura A8 y sus respectivas descendientes.

Figura A3.4: Pantalla de menú. Modo cliente. Menú lateral.



El menú mostrado seguirá el mismo planteamiento que el explicado en la Figura A3.2, por lo que la asignación quedará como se plantea a continuación:

- El elemento **Preferencias** corresponde a la Figura A4 y sus respectivas descendientes.
- El elemento de **Crear un encargo** corresponde a la Figura A9 y sus respectivas descendientes.
- El elemento de **Mis envíos** corresponde a la Figura A8 y sus respectivas descendientes asociadas.
- El elemento **Créditos** corresponde a la Figura A6.
- El elemento **Cerrar sesión** corresponde a la Figura 10.

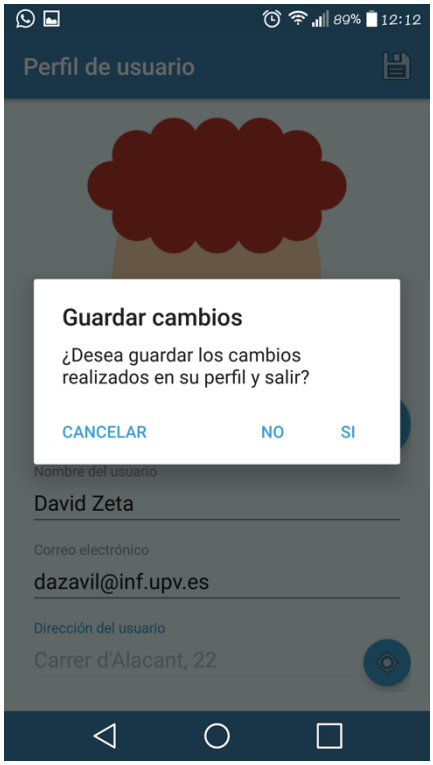
Figura A4.1: Pantalla de perfil. Información almacenada.

Esta pantalla muestra la información del usuario que está almacenada en el servidor de Firebase.
Como se puede observar esta pantalla tiene la opción de cambiar la imagen del usuario a través del botón azul en la esquina inferior derecha del avatar.
Tanto el nombre como el correo son editables.
La dirección de usuario está deshabilitada, porque se selecciona desde el botón azul de su derecha.
A través de este icono se despliega la pantalla del mapa, para seleccionar una ubicación.
Una vez seleccionada a través de los servicios de Google Maps, obtendrá el nombre de la calle equivalente al punto seleccionado en el mapa.
El botón de guardar de arriba a la derecha en la barra lanzará el diálogo mostrado en la Figura A4.3

Figura A4.2: Pantalla de perfil. Actualizando información.

Como se puede apreciar en esta pantalla, se ha hecho un cambio en la dirección, la cual ha calculado a través de Google Maps la dirección equivalente al punto seleccionado.

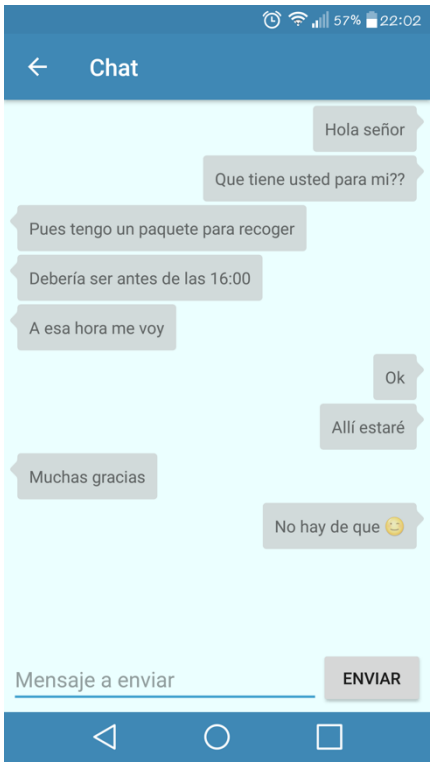
Figura A4.3: Pantalla de perfil. Almacenando cambios.



Una vez se acciona el botón de guardar, muestra este diálogo. El comportamiento variará dependiendo de la respuesta recibida:

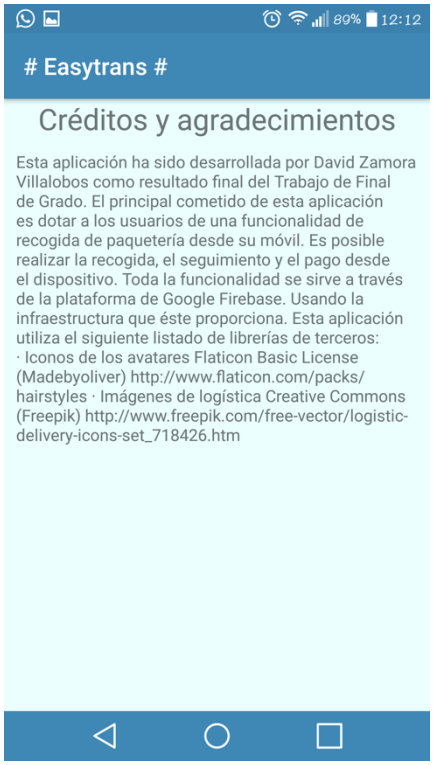
- SI: Guardará los cambios y te devolverá al menú.
- NO: Saldrá sin guardar ningún cambio.
- CANCELAR: Cerrará el diálogo para continuar editando la información.

Figura A5: Pantalla de chat.



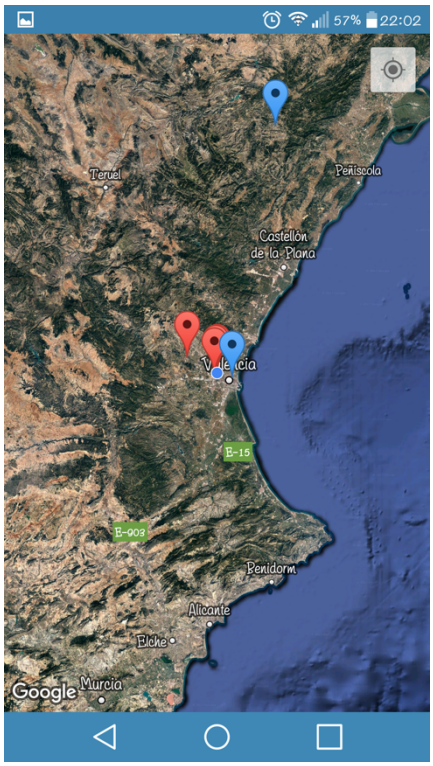
Esta es la ventana que te permite interaccionar con otro usuario del sistema.
Esta ventana es un sistema de mensajería en tiempo real. Los elementos de la derecha corresponden a los mensajes enviados por el usuario de este dispositivo. Los elementos de la izquierda corresponden a los mensajes recibidos de otros usuarios.
El campo editable de la parte inferior de la pantalla es el hueco donde se escribirá el mensaje que se desee enviar. Cuando se haga *click* sobre el botón Enviar, este enviará el mensaje.

Figura A6: Pantalla de créditos y agradecimientos.



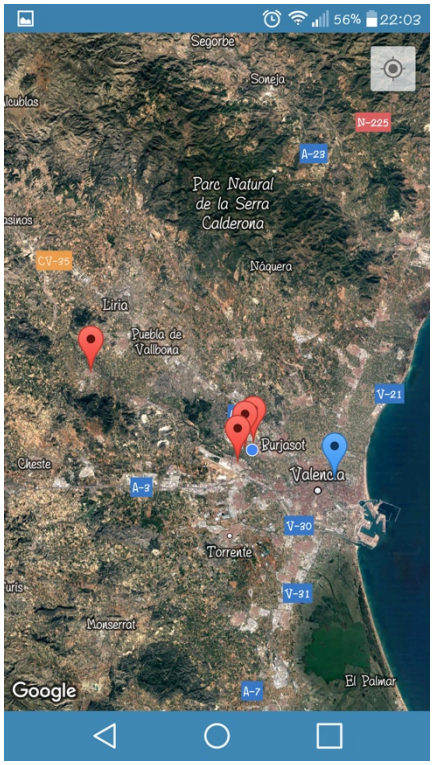
Esta es la pantalla de créditos de la app.
En ella simplemente se muestra un texto estático que no varía de referencias y ayudas que ha usado esta aplicación.

Figura A7.1: Modo trabajador. Pantalla de encargos cercanos.



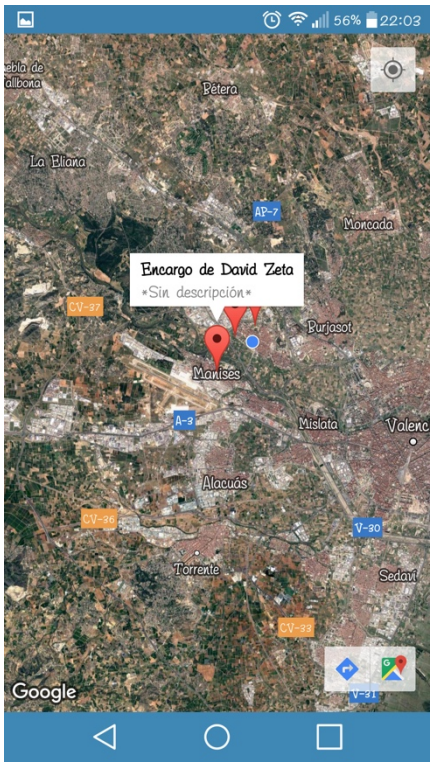
Esta es la primera pantalla especial para el usuario de tipo trabajador. Un usuario cliente no la verá.
Los iconos azules corresponden a encargos ya asignados, y los rojos a encargos aún por asignar.
Esta es la vista por defecto sin hacer zoom.
El punto azul corresponde a la ubicación del usuario.

Figura A7.2: Modo trabajador. Pantalla de encargos cercanos.



Esta es la misma pantalla que la de la Figura A7.1, pero se ha hecho un poco más de zoom sobre la zona más cercana, para obtener detalles más precisos de la ubicación.

Figura A7.3: Modo trabajador. Pantalla de encargos cercanos.

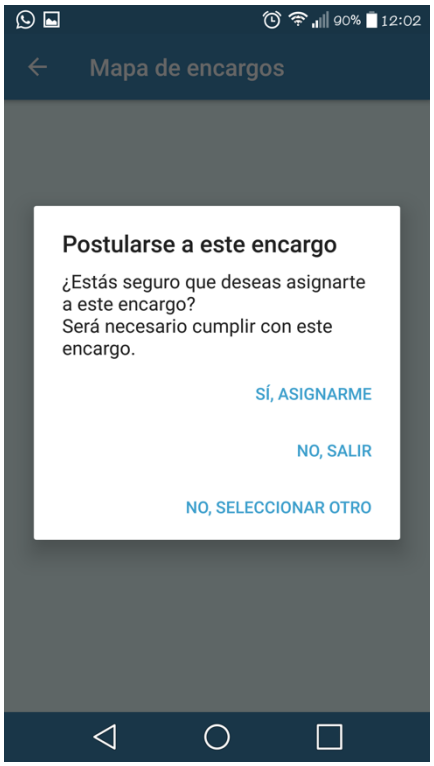


Aquí en esta pantalla, que sigue siendo la misma que la de las Figuras A7.1 y A7.2 pero seleccionando un encargo en concreto.
Al apretar sobre él, se despliega una etiqueta como la que se ve, que también genera una acción al apretar sobre ella.
Esta acción será visible en la Figura A7.4 de más adelante.

Figura A7.4: Modo trabajador. Pantalla de encargos cercanos.

Este diálogo se mostrará tras la selección de una etiqueta desplegada de un marcador del mapa.
Si se selecciona el botón cancelar, se cierra el diálogo devolviéndote el control sobre el mapa para volver a seleccionar otro encargo distinto.

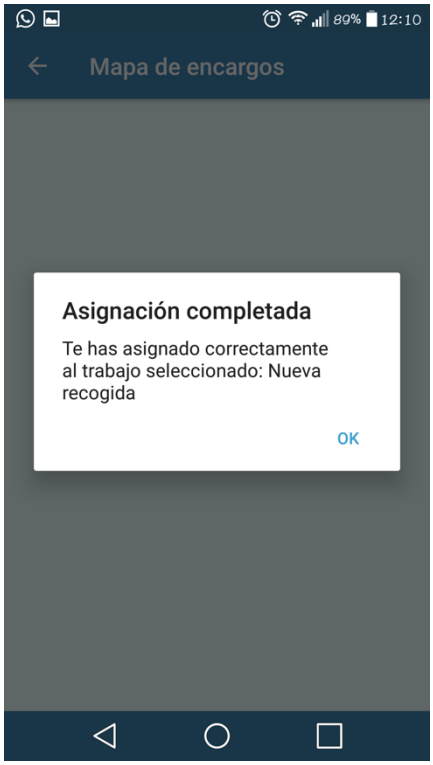
Figura A7.5: Modo trabajador. Pantalla de encargos cercanos.



Una vez seleccionado desde el diálogo, se despliega esta nueva ventana, que nos permite asegurarnos de que queremos postularnos sobre este encargo realmente.
Tendremos dos acciones distintas en función de nuestra elección:

- SI: Generará el diálogo de la Figura A7.6.
- SALIR: Volverá al menú principal sin asignarte nada.
- OTRO: Volverá al mapa para seleccionar de nuevo otro encargo.

Figura A7.6: Modo trabajador. Pantalla de encargos cercanos.



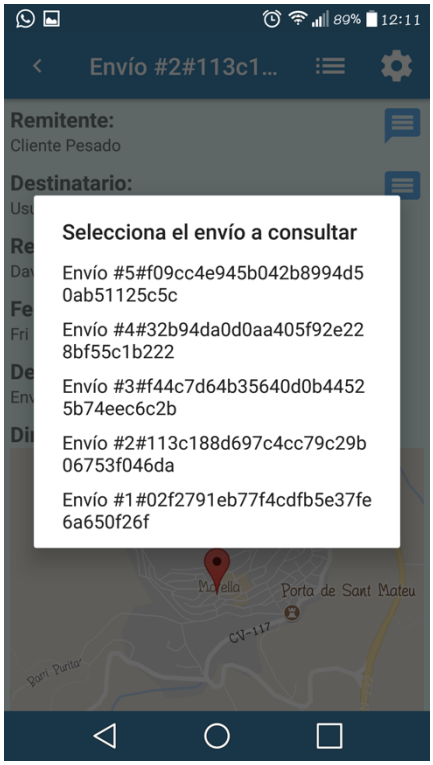
Esta ventana es meramente informativa.
Te informa de que el proceso de asignación como trabajador ha tenido éxito.
Este cambio ha generado una notificación push.
Al aceptarla, volverá el menú.

Figura A8.1: Modo trabajador. Pantalla de listado de encargos.

Esta ventana te permite gestionar varias acciones.
La primera y principal funcionalidad es la de Mostrar la información del encargo.
Se representa en los textos que muestra esta vista.
Cada usuario distinto a ti, y existente, tendrá un icono de un chat. Este icono desplegará la ventana de chat (Figura A5) con el usuario de su izquierda.
El mapa representa la ubicación exacta del pedido en el mapa.
En la parte inferior aparece el estado del encargo.
Es posible que algún encargo no tenga asignado destinario, el destinatario solo aparece si existe un usuario con cuenta en el sistema que vaya a recibir este pedido.
Los dos iconos de arriba despliegan cada uno acciones distintas:

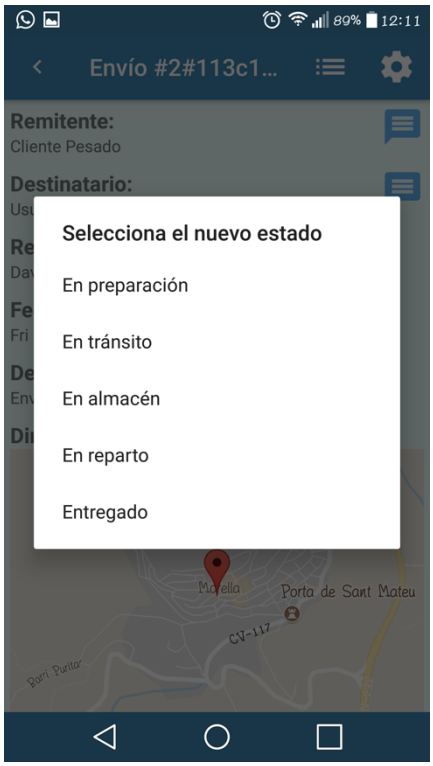
- Rueda dentada: Ajusta el estado del encargo.
- Tres líneas: Cambia de encargo.

Figura A8.2: Modo trabajador. Pantalla de listado de encargos.



Este diálogo lo despliega el icono de las tres líneas.
Permite seleccionar un encargo para mostrar.
Este es el método para cambiar la información del encargo y poder verla.

Figura A8.3: Modo trabajador. Pantalla de listado de encargos.

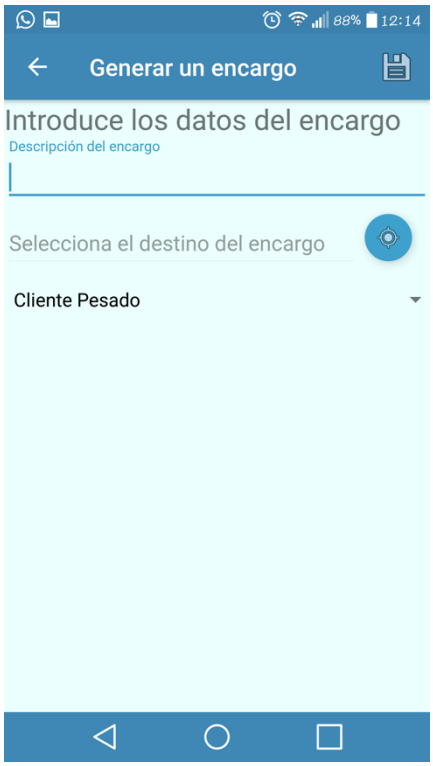


Este diálogo lo despliega el icono de la rueda dentada, que permite cambiar en tiempo real el estado de un pedido.
Este cambio genera una notificación push.

Figura A8.4: Modo cliente. Pantalla de listado de encargos.

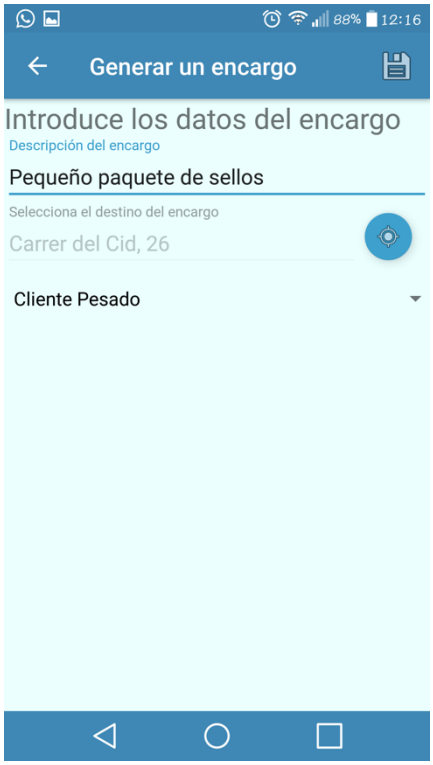
Como puede observarse esta ventana es idéntica a las anteriores de listado, pero como es en modo cliente, no permite modificar el estado del envío.

Figura A9.1: Modo cliente. Pantalla de generar un encargo.



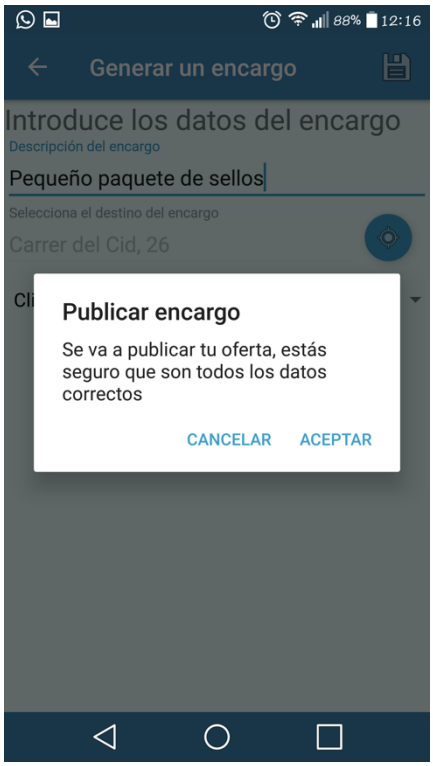
Esta es la primera ventana exclusiva en modo cliente. Esta permite generar un encargo a partir de los datos facilitados.
Tiene campos de introducción, selección y el de mapa. Además un botón de guardado que es igual que los demás.

Figura A9.2: Modo cliente. Pantalla de generar un encargo.



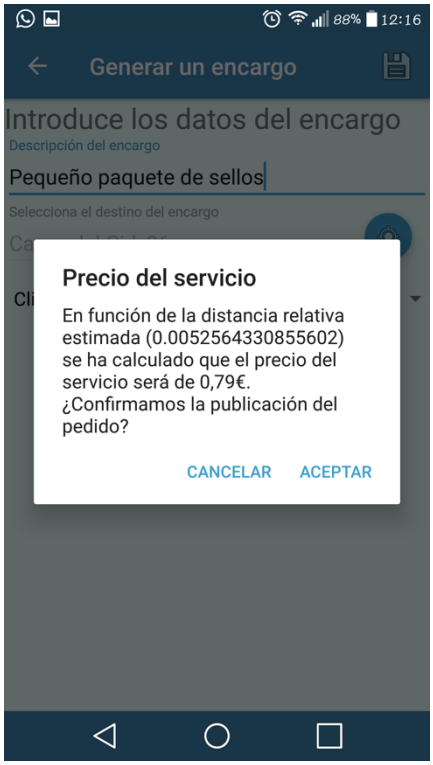
Esta es la misma ventana, pero con los campos rellenos. Al hacer *click* en guardar, se desplegará el diálogo de la Figura A9.3 para confirmar los cambios.

Figura A9.3: Modo cliente. Pantalla de generar un encargo.



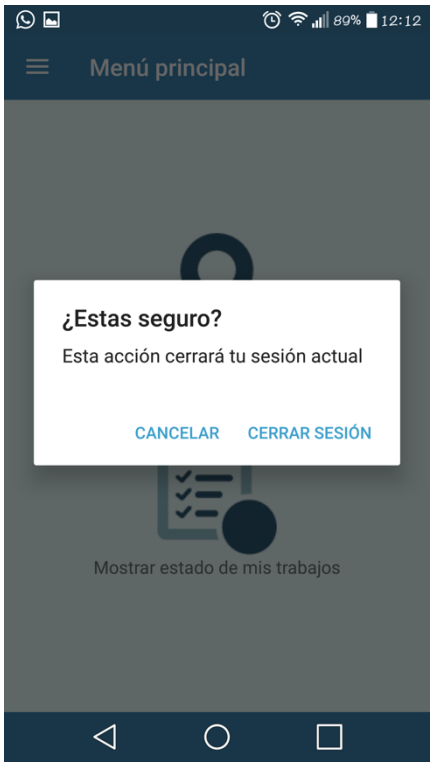
Este diálogo evitará que cometas alguna equivocación al publicar el encargo. Ya que generaría un encargo que después debes ser honrado y cumplir con tu responsabilidad.
Cancelar te devuelve el control sobre el formulario, Aceptar te mandará a la Figura 9.4 informándote del precio del encargo.

Figura A9.4: Modo cliente. Pantalla de generar un encargo.



El precio del encargo ya se ha explicado cómo se calcula, en función de la distancia del trayecto, por lo que, al ser muy corto, el coste es muy reducido.
Aceptar el encargo generará un encargo, mientras que cancelar te devolverá el control sobre el formulario de nuevo.

Figura A10: Diálogo de cerrar sesión.



Esta es la última acción posible a realizar en la aplicación. Que es la de cerrar la sesión actual del usuario.
Esta acción te devuelve a la ventana de introducción de l aplicación.

Este es el final del documento del manual. Para más información puedes acudir al repositorio GitHub o ponerte en contacto con el autor.