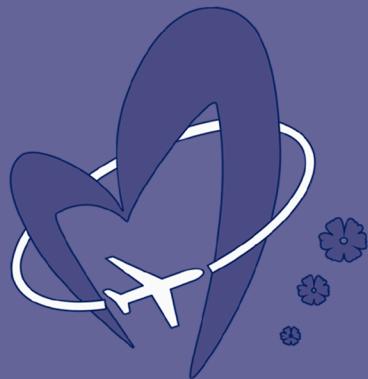


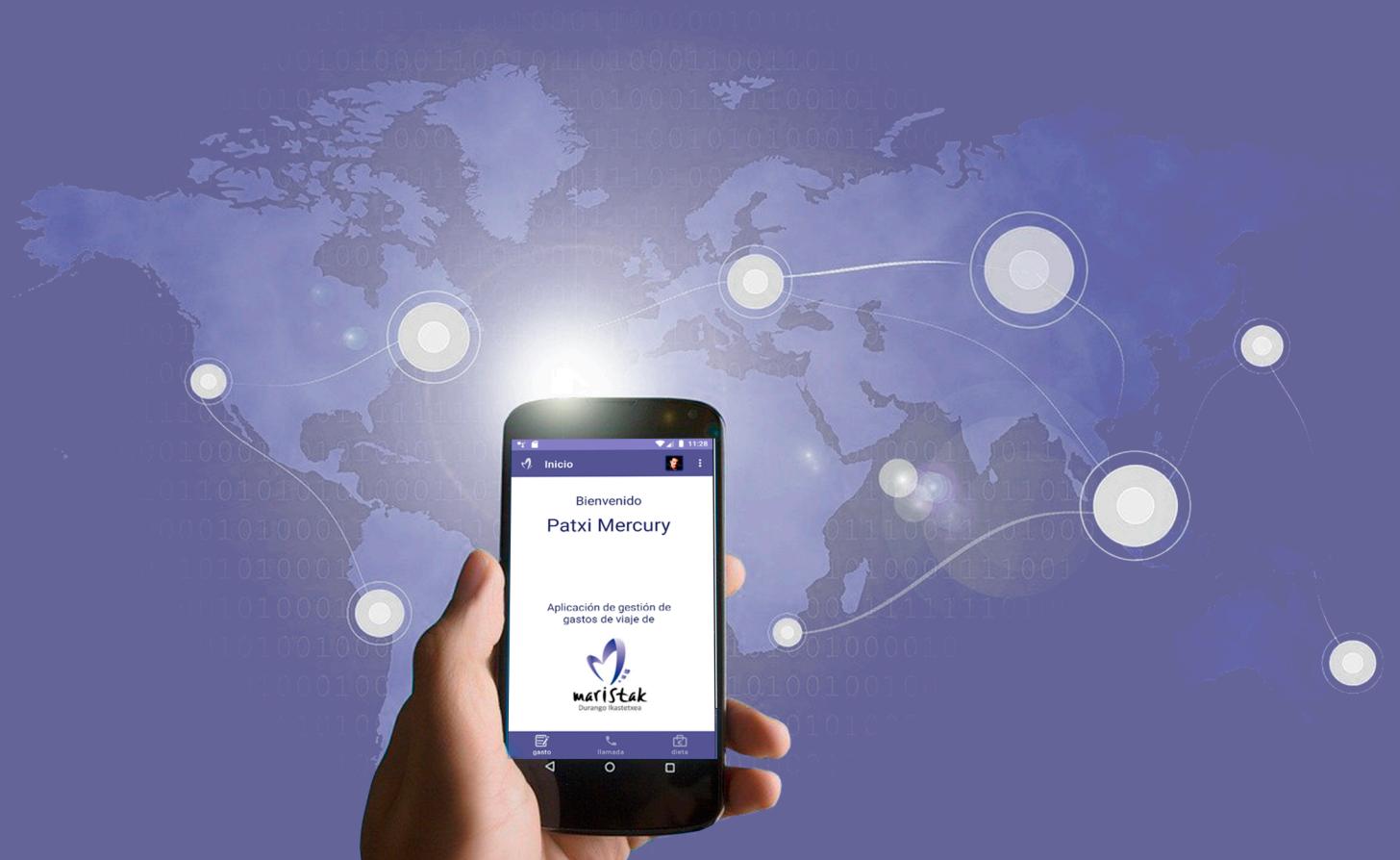
MEMORIA TÉCNICA

Agustín Sardón Cárcamo - Mikel Oceja Acha

1 de Junio del 2020



APLICACIÓN DE GESTIÓN DE VIAJES DE MARISTAK





1 INTRODUCCIÓN

2 OBJETIVOS

2.1 Especificaciones de Maristak

2.2 Nuestros objetivos

- » Interfaz de usuario orientada a la usabilidad y experiencia de usuario.
- » Código ordenado: reparto de responsabilidades, mantenibilidad, escalabilidad, seguridad.

3 APLICACIÓN MÓVIL

3.1 Características técnicas

- » Sistema operativo 8.0 o superior de Android.
- » Conexión a Base de Datos SQLite.
- » Interfaz gráfica con Material Design.
- » Navegación superior, inferior, menú.
- » Adaptado a diferentes densidades de pantalla.
- » Validaciones.
- » Consumo web service del banco para tarjetas de crédito.
- » Validación de tarjetas por geolocalización.
- » Adaptado al modo landscape del dispositivo
- » Autenticación y autorización.
- » Dispositivos en los que se ha probado la App.

3.2 Funcionamiento de la aplicación

- » Registro de dietas y gastos.
- » Listado de dietas y gastos.
- » Actualización de dietas y gastos.
- » Consultas de dietas y gastos.
- » Llamada telefónica de ayuda sobre la App.
- » Información del Usuario.
- » Gestión de tarjetas de crédito.
- » Login



3.3 Plataforma de desarrollo de software.

- » IDE Android Studio.
- » IDE Eclipse como taller de pruebas.
- » Illustrator y Photoshop para diseño interfaz gráfica.
- » SQLite Browser.
- » sqlite-jdbc-3.30.1.jar
- » Navegador web para peticiones GET al web service.
- » Control de versiones Git.
- » Repositorio para control de versiones: GitHub.

3.4 Recursos y librerías de terceros.

3.5 Análisis de la aplicación.

4 ENTREGA DEL PROYECTO

4.1 Archivos y documentos entregados

4.2 Credenciales de cuentas

4.3 Propuesta de autenticación para un caso real

4.4 Guía para cambios de parámetros de la aplicación

4.5 Registros para pruebas

5 CONCLUSIONES



INTRODUCCIÓN



1 INTRODUCCIÓN

El presente documento es la memoria técnica realizada por los alumnos Agustín Sardón y Mikel Ocea para la defensa del reto presentado por Maristak Ikastetxea Durango.

Se trata de un proyecto de carácter académico llevado a cabo durante la realización del curso de DESARROLLO DE APLICACIONES PARA DISPOSITIVOS ANDROID de Lanbide, impartido por Maristak como centro colaborador de Lanbide.

El curso se ha impartido a través de la metodología ETHAZI, promoviendo el aprendizaje colaborativo basado en retos y la responsabilidad sobre el propio aprendizaje.

Este reto tiene un carácter global, y relaciona todas las asignaturas fomentando, durante su ejecución, la asimilación de competencias tanto técnicas como transversales.

En este documento se recogen las especificaciones técnicas y principales características del proyecto.





2 OBJETIVO

El objetivo del proyecto es la creación de una App para la gestión de los gastos de viajes de una empresa que satisfaga las especificaciones funcionales y no funcionales exigidas en el enunciado del reto.

La aplicación debe gestionar los gastos y dietas derivados de los viajes de los empleados y la activación de las tarjetas de crédito corporativas, europea o internacional.

2.1 Especificaciones de Maristak

Especificaciones generales

- » Desarrollo aplicación móvil
- » Información de usuarios (login)
- » Navegación: menú, lista....
- » Barra navegación inferior
- » Toolbar: configuración / acerca de / info. Usuario (Icono usuario) → login
- » Gestión de gastos
- » Gestión de dietas
- » Gestión de tarjetas (web services)
- » Auditoría móvil (seguridad)

Especificaciones no funcionales

- » Versiones de Android 8.0 para arriba.
- » Que incluya conexión a una BBDD y acceso a la misma.
- » Que incluya un diseño moderno basado en Material Design.
- » Que incluya geolocalización.



OBJETIVO



2.2 Nuestros objetivos

Por parte de los desarrolladores, para complementar las especificaciones, hemos establecido los siguientes objetivos:

Crear una interfaz que aporte una experiencia de usuario agradable y facilite la navegación:

Hemos diseñado la App teniendo en mente la usabilidad. Hemos procurado que la app sea fácil de usar y que la navegación sea cómoda. Por ejemplo que se pueda pasar de visualizar a editar con los mínimos pasos posibles (en una pulsación).

Detalle	Valor
fecha:dd/mm/aaaa	14/10/2019
medio transporte	0.0
kilometraje	300.0
Precio km	0.3
peaje	16.0
parking	15.0
comida	40.0

También a la hora de recuperar información optamos por la mayor flexibilidad posible en las consultas y a ser posible desde un mismo sitio.

Filtro	Valor
desde fecha:dd/mm/aaaa	
hasta fecha:dd/mm/aa...	
desde importe	
hasta importe	



OBJETIVO

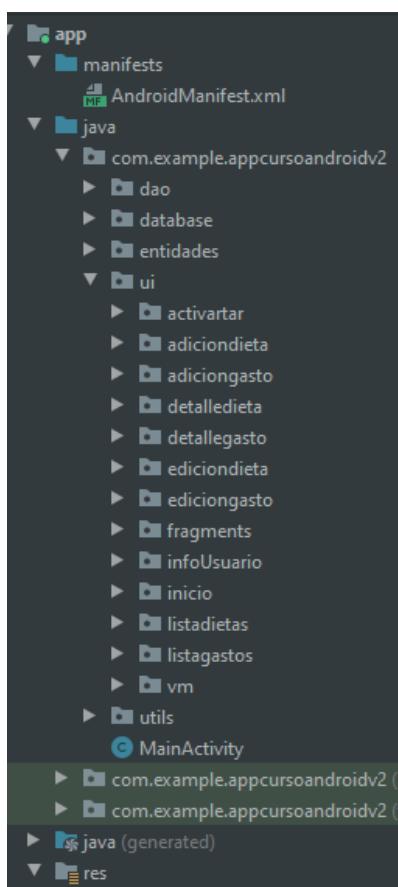


Por ejemplo, si queremos obtener registros desde una fecha concreta hasta la actualidad no obliguemos al usuario a meter la fecha actual, puede que eso responda a una lógica de un estilo de programación que se conforma con cumplir con los requisitos funcionales, pero desde luego no responde a la lógica del usuario que acusa el desgaste de trabajar diariamente con las mismas herramientas y con las mismas tareas repetitivas.

Hagámosle un favor, suavicémosle esas rutinas.

Crear una estructura de proyecto mantenable y escalable teniendo en cuenta la seguridad

Otro de los objetivos que nos hemos impuesto es el de crear un código que sea mantenable y escalable.



Para ello hemos seguido el principio de reparto de responsabilidades o separación de responsabilidades. Lo que significa que no se hace cualquier cosa en cualquier parte del código, sino que cada parte del código tiene una responsabilidad concreta. Se habla de lo mismo cuando se habla de desarrollo por capas. Así, por ejemplo, tenemos una capa de acceso a datos en la que tenemos las clases que gestionan las operaciones con la base de datos. También tenemos clases que mapean o reproducen campos de tablas con propiedades de objetos, los famosos pojos, que son las encargadas de transportar los datos que maneja la aplicación entre las diferentes capas. *Ver algún ejemplo en carpeta entidades.*

Esta combinación de clases manejadoras de entidades o tablas y objetos simples como transporte de datos se conoce como patrón DAO (Data Access Object). Esto además de mantener un código ordenado y clasificado nos permite reutilizar código y nos evita escribir código redundante, por ejemplo repetir consultas sql.



OBJETIVO



Y a la hora de hacer test o pruebas es más fácil encontrar los errores en tiempo de ejecución o errores lógicos porque sabemos qué parte del código es responsable de lo que está fallando, o al menos, sabemos en qué capa o clase se encuentra el código responsable de ese error.

Teniendo presente la seguridad en el código SQL hemos utilizado “sentencias preparadas” o “parametrizadas” cuando en ellas incluimos parámetros o datos introducidos desde los formularios.

```
public int modify(Gasto gasto) throws Exception {  
    String[] args = new String[]{String.valueOf(gasto.getId())};  
    return db.update(Constantes.TABLA_GASTO, getContentValues(gasto),  
        Constantes.GASTO_ID + "=?", args);  
}
```



3 APLICACIÓN MÓVIL

3.1 CARACTERÍSTICAS TÉCNICAS

3.1.1 Sistema operativo

Hemos desarrollado la aplicación para el sistema operativo Android 8.0 o superior

3.1.2 Base de Datos SQLite

- » Elegimos SQLite como base de datos por varios motivos:
- » Es código abierto y libre.
- » Guarda los datos en un archivo de texto del dispositivo y no necesita servidor ni drivers de conexión como JDBC, ODBC, etc.
- » Admite todas las características de las bases de datos relacionales.
- » Android viene con implementación de base de datos SQLite incorporada.

Sin embargo SQLite tiene un pequeño inconveniente, que no tiene tipo de dato fecha. Las alternativas al tipo fecha son String y Long. String no nos sirve porque vamos a tener que hacer cálculos con fechas, en concreto calcular los días que hay entre dos fechas para después en función de los días calcular el total de las dietas.

Recogemos un String de los campos fecha de los formularios, lo convertimos a un tipo fecha, en concreto Date, y de fecha a milisegundos (milisegundos desde el 1 de enero de 1970). A la hora de presentar fechas en la vista hacemos el proceso inverso; cogemos milisegundos de la base de datos, los pasamos a fecha y de fecha a String. Para ello utilizamos la clase Date porque tiene un constructor al que se le puede pasar milisegundos y la clase SimpleDateFormat. También hemos creado una clase propia DateFormat que simplifica las operaciones, con un método parse(String fecha) y un método getDateInTextFormat.

3.1.3 Interfaz gráfica con Material Design.

Utilizamos componentes gráficos de material design como los TextInputLayout que ofrecen características adicionales a los EditText como validaciones y labels animados, las barras de navegación o los cuadros de diálogo. Material Design aporta un look and feel moderno a la interfaz de usuario y se basa en principios de experiencia de usuario o UX que están muy estudiados y que mejoran la experiencia del usuario.





3.1.4 Navegación superior, inferior, menú.

En cuanto a los componentes de navegación hemos optado por no sobrecargar la aplicación de este tipo de componentes para no crear confusión al usuario. Y como en las especificaciones ya nos piden un barra superior y una barra inferior no hemos creído necesario añadir más. Siguiendo esa filosofía de menos es más, hemos aprovechado la barra superior para colocar en ella un menú desplegable y hemos utilizado la barra inferior para el acceso rápido a las opciones que podríamos decir son el núcleo de la aplicación como son el registro de gastos y el registro de dietas. También incluimos en ella la llamada de asistencia o ayuda para que esté accesible en todo momento y sea fácilmente localizable.



Para que la interfaz de usuario sea coherente pensamos que las barras de navegación superior e inferior tienen que estar visibles en todas las pantallas de la aplicación y no nos parece una solución elegante duplicar los componentes de la vista y el código asociado en todas las activities. Ni es elegante ni es mantenible. Que un cambio en el mismo componente, aunque sea en todas las pantallas, implique tocar el mismo código duplicado en múltiples sitios es más un problema que una solución. Por eso optamos por implementar los componentes de navegación y todo el código asociado en fragmentos y luego reutilizar esos fragmentos desde las diferentes activities. Es decir tenemos un fragmento para la barra superior y un fragmento para la barra inferior. Y luego lo que hacemos es incluir esos fragmentos en todas las activities. Así, el código que gestiona el menú está en el fragmento del toolbar y si queremos añadir una opción nueva al menú sólo tenemos que añadir esa opción en un sitio.

De la misma manera tenemos un único formulario para gastos, un único formulario para dietas y un único formulario para consultas. El formulario dietas por ejemplo se carga desde tres activities, una de creación, una de edición y otra de visualización.

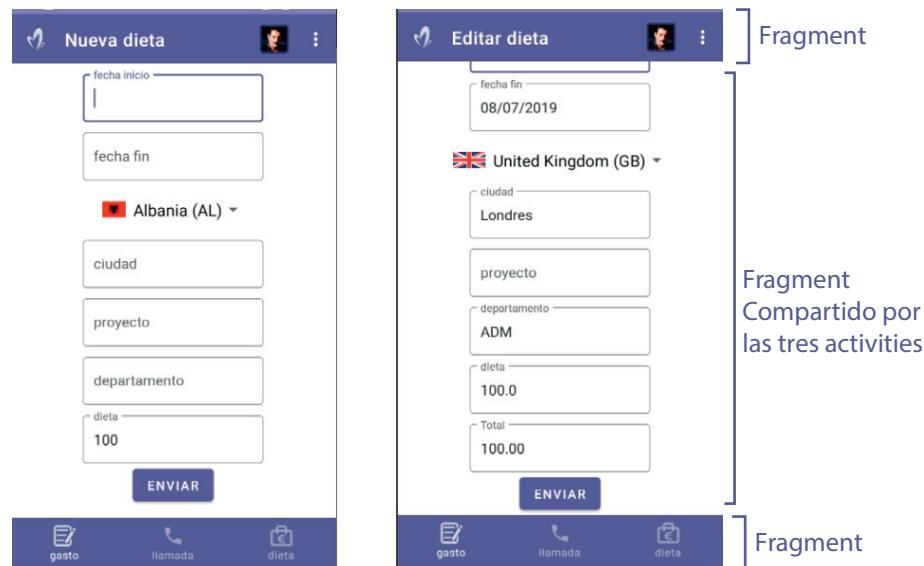
En visualización los campos no son editables, en creación y modificación sí. En modificación pasamos datos desde la base de datos a la vista, en creación no. El botón de enviar es único (está en el fragmento) pero en la activity de creación el método onClick crea una nueva dieta llamando a la capa correspondiente, en la activity de edición modifica la dieta.

Resumiendo, aunque sea el mismo formulario, el mismo fragment, cada activity tiene su estado sin afectar al resto y podemos hacer algo que se asemeja al polimorfismo.



APLICACIÓN MÓVIL

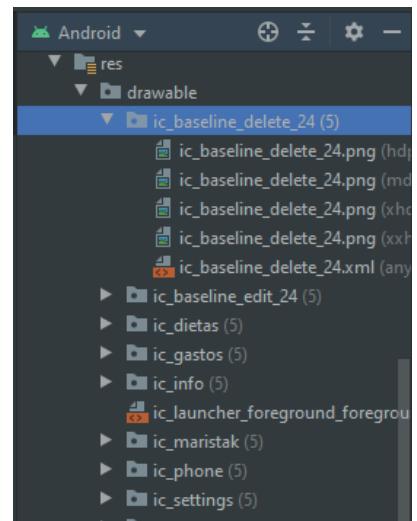
En la siguiente imagen vemos la pantalla correspondiente a la edición de dietas. La barra superior es un fragmento, la barra inferior otro fragmento, lo que está entre ambas es otro fragmentos y la activity es el marco que los contiene con su estado independiente que hace que los campos sean editables o no y que el botón al ser pulsado cree una dieta o la modifique.



3.1.5 Adaptado a diferentes densidades de pantalla.

Diferentes dimensiones de iconos para diferentes densidades de pantalla.

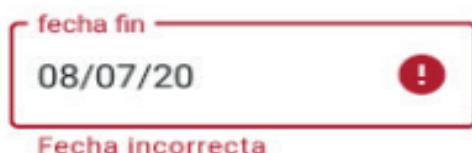
Los componentes visuales evitar en lo posible dar dimensiones fijas y jugar con los constraints.



3.1.6 Validaciones.

También hemos cuidado las validaciones o el filtrado de introducción de datos en los formularios mediante una expresión regular en el caso de las fechas y el tipo de input numérico en los EditText que introducen datos numéricos.

Estas validaciones o comprobaciones impiden que la aplicación se cierre por una excepción al meterse un tipo de dato que no es el esperado como una fecha incorrecta





o inexistente o un carácter alfabético donde se espera un número. Esto, junto con las sentencias parametrizadas en las consultas a la base de datos cuando se pasan parámetros introducidos desde los formularios, también añade un plus de seguridad.

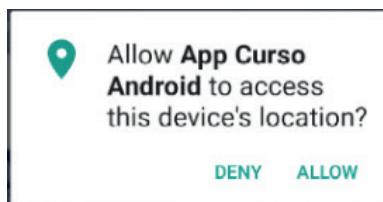
3.1.7 Consumo de Web Service

La aplicación simula el consumo de un Web Service que ofrece el banco ficticio para la activación de tarjetas europea e internacional. Para ello hacemos uso de una librería externa llamada Volley y utilizamos las clases de la API de java JSONObject y JSONArray para parsear JSON.

Relacionado con el web service para la activación de tarjetas está el siguiente epígrafe.

3.1.8 Validación de la activación de las tarjetas por geolocalización.

Utilizamos la API de Geolocalización de Google para obtener las coordenadas terrestres del dispositivo (longitud y latitud). A partir de las coordenadas terrestres y con el método getFromLocation de la clase Geocoder obtenemos una lista de objetos de tipo Address (dirección) y a partir de una dirección (la primera de la lista) obtenemos el código de país. Previamente requerimos los permisos necesarios del usuario.



3.1.9 Adaptado al modo landscape del dispositivo

A modo de complemento a las especificaciones técnicas que nos han sido solicitadas, hemos creído conveniente adaptar la aplicación al modo landscape de los dispositivos. Esto mejora la experiencia de usuario permitiendo al usuario usar el modo que más cómodo le resulte.

También hemos gestionado los problemas que se generan con el ciclo de vida de la actividad al cambiar este tipo de configuraciones. Hemos usado tanto ViewModels como los bundles que se guardan al detener la aplicación y se recargan al volver a ella.

3.1.10 Autenticación y autorización.

Comprobamos que el usuario y la contraseña que ha metido el usuario coinciden con los datos del usuario de la tabla usuario.

Si falla el login en el mensaje de validación decir que los datos introducidos no son correctos pero no dar más información.



APLICACIÓN MÓVIL



Guardar en la base de datos la fecha y hora en que hace login, que será la de la última conexión la próxima vez que haga login.

3.1.11 Dispositivos en los que se ha probado la App.

- » Emulador 4 WVGA (Nexus S) API 27
- » Emulador 4.7 WXGA API 27
- » Emulador 5.1 WVGA API 27

3.2 FUNCIONAMIENTO DE LA APLICACIÓN

3.2.1 Registro de dietas y gastos.

Para registrar un gasto o una dieta accedemos al formulario correspondiente desde el ícono de gasto y el ícono de dieta de la barra de navegación inferior. Una vez en el formulario rellenamos los datos y le damos al botón enviar. La aplicación nos mostrará un mensaje de que se ha creado el gasto o la dieta y el número de identificación del registro creado.

Si estamos creando un gasto la aplicación nos traerá automáticamente el precio del kilómetro de la base de datos (es un parámetro de la aplicación) y si estamos creando una dieta nos traerá automáticamente el precio de la dieta en función del código de país que hayamos introducido en el campo país. El código de país se introduce desde un desplegable de códigos de países.

3.2.2 Listado de dietas y gastos.

Para obtener un listado de dietas o gastos accedemos desde el menú desplegable en la barra de navegación superior.



APLICACIÓN MÓVIL

3.2.3 Actualización y borrado de dietas y gastos.

Para actualizar o cambiar algún valor de una dieta o de un gasto accedemos desde el listado de dietas o listado de gastos. Para ello pulsamos el icono del lápiz en la línea correspondiente a la dieta o al gasto y nos llevará al formulario de edición. Una vez allí cambiamos el valor o valores que necesitemos y le damos al botón enviar. La aplicación nos lanzará un mensaje de que los cambios han sido guardados. Otra forma de llegar a la pantalla de edición del gasto o la dieta es desde la pantalla de detalle del gasto o dieta pulsando el botón ACTUALIZAR.

Exactamente la misma navegación se utiliza para borrar una dieta o un gasto, pero en este caso el icono es el de la papelera y el botón el de BORRAR.

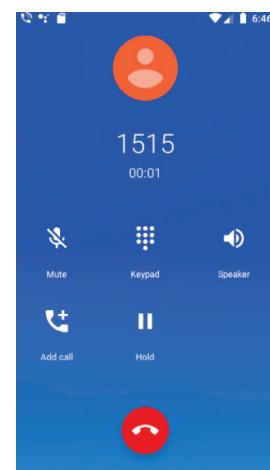
3.2.4 Consultas de dietas y gastos.

Para consultar o buscar dietas o gastos accedemos desde el menú desplegable de la barra de navegación superior, opciones “Consulta dietas” y “Consulta gastos”. Una vez en el formulario nos presentará los campos para introducir los criterios de filtro o de búsqueda. Podemos acotar la fecha por abajo, la fecha por arriba, el importe por abajo, el importe por arriba y cualquier combinación de los cuatro parámetros, desde dejar los cuatro en blanco a llenar los cuatro.

Cuando pulsamos “enviar” se ejecutará una consulta con los parámetros introducidos en el formulario y nos mostrará un listado con las filas que cumplan los criterios indicados. Si no hay ninguna coincidencia el listado se mostrará en blanco.

3.2.5 Llamada telefónica de ayuda sobre la App.

La aplicación dispone de un servicio de ayuda o asistencia sobre el funcionamiento de la aplicación que consiste en un número de teléfono al que se puede llamar en cualquier momento pulsando el icono de llamada de la barra de navegación inferior.





APLICACIÓN MÓVIL



3.2.6 Información del Usuario.

Podemos acceder a la información del usuario pulsando el avatar del usuario en la barra de navegación superior. Esto nos mostrará una pantalla con la foto y alias del usuario, nombre y apellido, DNI y la fecha y hora de la última vez que se conectó o usó la aplicación.



3.2.7 Gestión de tarjetas de crédito.

Para acceder a la gestión de las tarjetas de crédito hay que desplegar el menú del toolbar y seleccionar la opción Gestión tarjetas. En esta actividad hay dos ilustraciones que simulan las tarjetas. En ellas podemos encontrar la información obtenida del webservice: Número de la tarjeta, fecha de la última activación y, representado de forma gráfica, el estado de la activación de las tarjetas. Cabe destacar que la aplicación realiza una validación a la hora de activar la tarjeta, impidiendo dicha activación en caso de encontrarse en una ubicación en la que la tarjeta a activar no sea de aplicación. Hay que tener en cuenta que la activación de una tarjeta supone la desactivación de la otra.



3.2.8 Login

Para el acceso a la aplicación es preciso introducir las credenciales del usuario correctas.



Si el usuario no introduce alguno de los datos e intenta enviar el formulario, la aplicación notificará al usuario de la falta del campo en cuestión. Si alguno de los datos introducidos por el usuario no es correcto, la aplicación notificará de la incorrección sin dar información sobre el campo o campos erróneos o correctos. De esta manera una persona o programa ajeno no tiene información que le facilite la posible entrada ilegítima a nuestra aplicación.

En esta actividad hemos integrado un toolbar específico para ella. Entre las especificaciones había que hacer un toolbar con un botón de información acerca de la versión de la App y otro de configuración que por ahora no debía hacer nada.



Sabemos que en un caso real la configuración de la App no debería estar accesible antes de que el usuario se loguease, pero ya que no va a tener funcionalidad hemos preferido integrarlo en este toolbar para mantener el toolbar general con una funcionalidad total.

3.3 Plataforma de desarrollo de software.

IDE Android Studio

El ide que hemos utilizado para desarrollar el código final de la aplicación es android studio. Lo hemos elegido porque es apropiada para desarrollar la aplicación en el lenguaje nativo de android. Además, es una herramienta cuyo propietario es google, que también es propietario de android, con lo que la máxima compatibilidad está asegurada.

IDE Eclipse

Para hacer pruebas del código y decidir cómo solucionar los diferentes retos que se nos han presentado hemos utilizado el IDE eclipse ya que nos a sido más sencillo hacer pruebas sencillas en ella.

Illustrator y Photoshop

Los logotipos, el icono de lanzamiento, algún icono personalizado y algún otro elemento han sido diseñados mediante estas herramientas de la familia adobe. La mayoría de los iconos que hemos usado son de material design, pero hemos modificado algunos iconos de material design y los hemos subido como SVG para que el propio android studio los adapte a las diferentes densidades de pantalla.

También hemos usado estas herramientas y otras como google docs o pencil para la elaboración de la planificación inicial, wireframe y mockup.

SQLite Browser

A la hora de hacer pruebas con la base de datos hemos utilizado esta herramienta porque nos permite ver los datos de la base de datos y contrastarlo con los resultados que obtenemos en la aplicación.

sqlite-jdbc-3.30.1.jar

Conector para SQLite para hacer pruebas desde Eclipse



Navegador web

Para hacer pruebas de acceso al webservice hemos necesitado modificar datos del web service mediante peticiones Get del navegador. En concreto, esta necesidad viene derivada de la comprobación que realiza nuestra App de la localización del usuario. El emulador detecta nuestra posición en US, por lo que la aplicación no nos permite activar la tarjeta europea.

Control de versiones Git

Hemos trabajado usando Git como herramienta de control de versiones. Android studio ofrece la posibilidad de trabajar con esta herramienta de forma integrada, y por ello nos hemos decantado por esta opción.

Repositorio para control de versiones: GitHub

El código de la aplicación ha sido desarrollado en el repositorio que hemos creado en GitHub. Hemos creado un único usuario y hemos trabajado conjuntamente en el master.

3.4 Recursos y librerías de terceros.

Volley

<https://github.com/google/volley>

Librería para hacer peticiones http asíncronas

Volley ofrece justo lo que se necesita para consumir un servicio web tipo REST:

- » Poder hacer peticiones http asíncronas y usando los diferentes métodos del protocolo http.
- » Método onResponse para tratar la respuesta obtenida en caso de éxito y método onError para gestionar los errores en caso de fallo.
- » Permite el envío de parámetros por POST sobreescribiendo el método getParams.
- » Permite el envío de cabeceras sobreescribiendo el método getHeaders.
- » No impone un manera estricta o rígida de hacer las cosas sino que te da flexibilidad.
- » Es una librería creada por Google (le presuponemos solvencia técnica o que estará bien hecha).
- » Proyecto libre y presente en GitHub.

Servicios de localización de google

Utilizamos los servicios de localización de google (play-services-location)



Country code picker

<https://github.com/hbb20/CountryCodePickerProject/>

En la sección de dietas, a fin de proporcionar la máxima usabilidad, nos encontramos con un pequeño obstáculo. Nuestra intención era que la aplicación aplicase el precio por día correcto dependiendo del país que introdujese. Pero a la hora de introducir el nombre del país las diferentes variantes podían producir que nuestra aplicación no relacionase correctamente el país. Pedir el código de país al usuario era inviable, por lo que buscamos una solución en la red.

United Kingdom (GB) ▾

La solución ha sido esta librería que nos permite desplegar un selector de país con una bandera y el nombre del país en inglés. La librería funciona mediante un ítem XML propio que te permite obtener o mostrar diversa información. En nuestro caso hemos evaluado el código del nombre del país del estándar ISO 3166 para asignar un precio por día a la dieta.

3.5 ANÁLISIS DE LA APLICACIÓN

En esta sección vamos a realizar un análisis de nuestra aplicación partiendo del archivo apk. Simularemos que es una aplicación de terceros y vamos a analizar los archivos resultantes de las diferentes descompresiones, observando principalmente los permisos que se le solicitan al usuario y estudiando el uso que se da de ellos. Obviamente en nuestro proyecto no hay ningún tipo de código malicioso, es una práctica con carácter académico.

Descompresión con apktool

Con esta herramienta vamos a poder acceder a diversos archivos. Uno de los más interesantes es el archivo manifest.

	Nombre	Fecha de modificación	Típ
l	assets	2020/06/23 12:46	Car
1	original	2020/06/23 12:46	Car
l	res	2020/06/23 12:46	Car
l	smali	2020/06/23 12:46	Car
l	unknown	2020/06/23 12:46	Car
	AndroidManifest	2020/06/23 12:46	Doc
	apktool	2020/06/23 12:46	Arc



APLICACIÓN MÓVIL



En el podemos encontrar los permisos que le son solicitados al usuario.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:compileSdkVersion="29" android:compileSdkVersionCodename="10" package="com.example.appcursoandroidv2" platformBuildVersionCode="29"
    platformBuildVersionName="10">
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.CALL_PHONE"/>
    <application android:allowBackup="true" android:appComponentFactory="androidx.core.app.CoreComponentFactory" android:extractNativeLibs="false"
        android:icon="@mipmap/ic_launcher_foreground" android:label="@string/app_name" android:roundIcon="@mipmap/ic_launcher_foreground_round"
        android:supportsRtl="true" android:id="@+id/AppTheme">
        <activity android:name="com.example.appcursoandroidv2.ui.detalldietas.DetalleDietasActivity"/>
        <activity android:name="com.example.appcursoandroidv2.ui.ediciondieta.EdicionDietasActivity"/>
        <activity android:name="com.example.appcursoandroidv2.ui.adiciondieta.AditionDietasActivity"/>
        <activity android:name="com.example.appcursoandroidv2.ui.infousuario.InfoUsuarioActivity"/>
        <activity android:name="com.example.appcursoandroidv2.ui.listagastos.FiltroGastosActivity"/>
        <activity android:name="com.example.appcursoandroidv2.ui.listadietas.FiltroDiетasActivity"/>
        <activity android:name="com.example.appcursoandroidv2.ui.activartar.ActivarTarjetaActivity"/>
        <activity android:name="com.example.appcursoandroidv2.ui.inicio.InicioActivity"/>
        <activity android:name="com.example.appcursoandroidv2.ui.detalliegasto.DetalleGastoActivity"/>
        <activity android:name="com.example.appcursoandroidv2.ui.ediciongasto.EdicionGastoActivity"/>
        <activity android:name="com.example.appcursoandroidv2.ui.listagastos.ListaGastosActivity"/>
        <activity android:name="com.example.appcursoandroidv2.ui.listadietas.ListaDietasActivity"/>
        <activity android:name="com.example.appcursoandroidv2.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity android:exported="false" android:name="com.google.android.gms.common.api.GoogleApiActivity"
            android:theme="@android:style/Theme.Translucent.NoTitleBar"/>
        <meta-data android:name="com.google.android.gms.version" android:value="@integer/google_play_services_version"/>
        <provider android:authorities="com.example.appcursoandroidv2.lifecycle-process" android:exported="false" android:multiprocess="true"
            android:name="androidx.lifecycle.ProcessLifecycleOwnerInitializer"/>
    </application>
</manifest>
```

También se nos han descomprimido los archivos de la carpeta resources. Layouts, menús, archivos,...

aplicaciones_sospechosas > Tools > apktool > app-release > res >	
Nombre	Fecha de modificación
color	2020/06/23 12:46
drawable	2020/06/23 12:46
drawable-anydpi	2020/06/23 12:46
drawable-hdpi	2020/06/23 12:46
drawable-ldpi	2020/06/23 12:46
drawable-ldrtl-hdpi	2020/06/23 12:46
drawable-ldrtl-mdpi	2020/06/23 12:46
drawable-ldrtl-xhdpi	2020/06/23 12:46
drawable-ldrtl-xxhdpi	2020/06/23 12:46
drawable-ldrtl-xxxhdpi	2020/06/23 12:46
drawable-mdpi	2020/06/23 12:46
drawable-nodpi	2020/06/23 12:46
drawable-watch	2020/06/23 12:46
drawable-xhdpi	2020/06/23 12:46
drawable-xxhdpi	2020/06/23 12:46
drawable-xxxhdpi	2020/06/23 12:46
interpolator	2020/06/23 12:46
layout	2020/06/23 12:46
layout-land	2020/06/23 12:46
layout-sw600dp	2020/06/23 12:46
layout-watch	2020/06/23 12:46
menu	2020/06/23 12:46



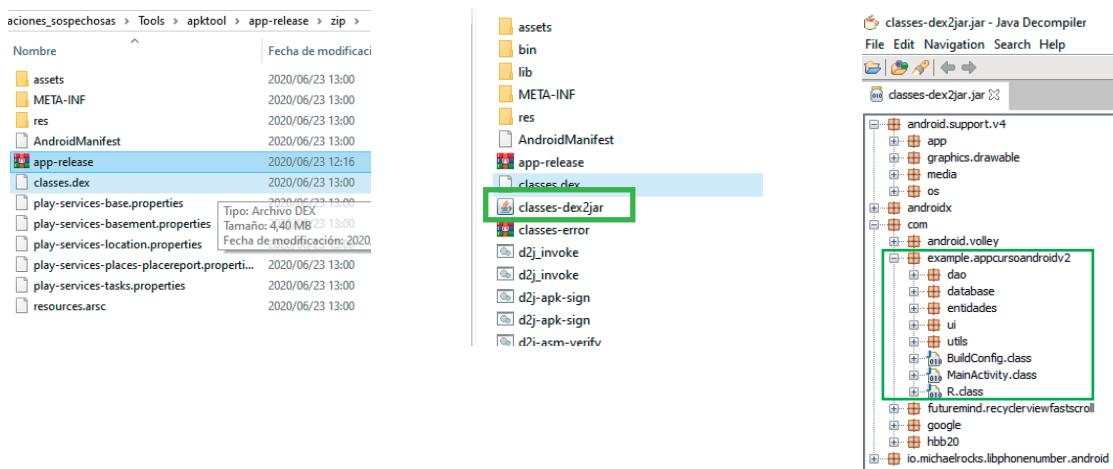
APLICACIÓN MÓVIL



Descompresión del zip

El objetivo de la siguiente descompresión es poder usar las siguientes herramientas que nos dan acceso a los archivos .class de la aplicación.

- » Para ello, comenzaremos cambiando la extensión del archivo .apk a un .zip, y lo descomprimimos.
- » Ahora tenemos los archivos .dex y los vamos a convertir a .jar usando la última versión de la herramienta dex2jar.
- » La herramienta anterior nos ha creado un archivo llamado classes-dex2jar, es el archivo comprimido de java.
- » Con la tercera herramienta, jd-gui-windows, abrimos el archivo classes-dex2jar y ya podemos ver los archivos .java de nuestra aplicación.



Ahora analizaremos el código de nuestra aplicación para ver el uso que se da de los permisos.

El permiso del acceso al teléfono es usado por nuestra aplicación en caso de que el usuario pulse sobre el botón de asistencia telefónica.

```

paramAnonymousMenuItem = new Intent( android.intent.action.CALL );
paramAnonymousMenuItem.setData(Uri.parse("tel:1515"));
if (ActivityCompat.checkSelfPermission(BottomNavigationFragment.this.getActivity(), "android.permission.CALL_PHONE") == PackageManager.PERMISSION_GRANTED)
    BottomNavigationFragment.this.requestPermissions(new String[] { "android.permission.CALL_PHONE" }, 101);
else {
    BottomNavigationFragment.this.startActivity(paramAnonymousMenuItem);
}
break;
case 2131230873:
    paramAnonymousMenuItem.setChecked(true);
    BottomNavigationFragment.this.intent = new Intent(BottomNavigationFragment.this.context, AdicionGastoActivity.class);
    paramAnonymousMenuItem = BottomNavigationFragment.this;
    paramAnonymousMenuItem.startActivity(paramAnonymousMenuItem.intent);
    break;
case 2131230872:
    paramAnonymousMenuItem.setChecked(true);
    BottomNavigationFragment.this.intent = new Intent(BottomNavigationFragment.this.context, AdicionDietaActivity.class);
    paramAnonymousMenuItem = BottomNavigationFragment.this;
    paramAnonymousMenuItem.startActivity(paramAnonymousMenuItem.intent);
}
return false;
);
}
return paramViewGroup;

if (paramInt == 101) {
    if ((paramArrayOfInt.length > 0) && (paramArrayOfInt[0] == 0))
    {
        paramArrayOfString = new Intent("android.intent.action.CALL");
        paramArrayOfString.setData(Uri.parse("tel:1515"));
        startActivity(paramArrayOfString);
    }
}

```



APLICACIÓN MÓVIL



Los permisos de localización son utilizados para la validación la localización del usuario a la hora de activar las tarjetas. El usuario solo podrá activar una tarjeta si está localizado en el ámbito de aplicación territorial de la tarjeta a activar. Es decir, si el usuario está en europa e intenta activar la tarjeta internacional, la aplicación comprobará la localización del usuario y al detectar que se halla en europa denegara la solicitud de activación de tarjeta internacional.

```
this.tvCardEurope.setText((CharSequence)this.records.get("card_europe"));
this.tvCardInternational.setText((CharSequence)this.records.get("card_international"));
str2 = (String)this.records.get("current_card");
if (str2.equals("EUROPE"))
{
    this.ivTarjetaEurope.setImageAlpha(255);
    this.ivTarjetaInternational.setImageAlpha(170);
}
if (str2.equals("INTERNATIONAL"))
{
    this.ivTarjetaEurope.setImageAlpha(170);
    this.ivTarjetaInternational.setImageAlpha(255);
}

private void obtenerCoordenadas()
{
    this.fusedLocationProviderClient = LocationServices.getFusedLocationProviderClient(this);
    if ((ActivityCompat.checkSelfPermission(this, "android.permission.ACCESS_FINE_LOCATION") != 0) && (ActivityCompat.checkSelfPermission(this, new String[] { "android.permission.ACCESS_FINE_LOCATION", "android.permission.ACCESS_COARSE_LOCATION" }) == 0))
    {
        ActivityCompat.requestPermissions(this, new String[] { "android.permission.ACCESS_FINE_LOCATION", "android.permission.ACCESS_COARSE_LOCATION" });
    }
    else
    {
        this.permisosLocation = true;
    }
    this.fusedLocationProviderClient.getLastLocation().addOnSuccessListener(this, new OnSuccessListener<Location>()
    {
        public void onSuccess(Location paramAnonymousLocation)
        {
            ...
        }
    });
}
```

Además de estos permisos, la aplicación también solicita el acceso a internet. Nuestra aplicación usa internet para diversos cometidos.

- » La comunicación con el webservice se efectúa mediante una API Rest utilizando el protocolo http

```
private void sendHttpRequest(String paramString)
{
    Volley.newRequestQueue(this).add(new JsonObjectRequest(0, paramString, null, new Response.Listener<JSONObject>()
    {
        public void onResponse(JSONObject paramAnonymousJSONObject)
        {
            ActivarTarActivity.this.parseJSON(paramAnonymousJSONObject);
            ActivarTarActivity.this.loadInfo();
        }
    }), new Response.ErrorListener()
    {
        public void onErrorResponse(VolleyError paramAnonymousVolleyError)
        {
            Toast.makeText(ActivarTarActivity.this, paramAnonymousVolleyError.getMessage(), 0).show();
        }
    });
}
```



- » La imagen del usuario que aparece en el toolbar y la activity de información del usuario está colgada en la red. En nuestra base de datos tenemos guardada la url de la imagen la cual es llamada mediante los servicios de internet.

```
paramBundle.setUser(this.user);
final ImageLoader localImageLoader = new ImageLoader(Volley.newRequestQueue(this), new ImageLoader.ImageCache()
{
    private final LruCache<String, Bitmap> mCache = new LruCache(10);

    public Bitmap getBitmap(String paramAnonymousString)
    {
        return (Bitmap)this.mCache.get(paramAnonymousString);
    }

    public void putBitmap(String paramAnonymousString, Bitmap paramAnonymousBitmap)
    {
        this.mCache.put(paramAnonymousString, paramAnonymousBitmap);
    }
}
```

Conclusiones del análisis

En nuestra aplicación no hay código malicioso. Los permisos que se le han solicitado al usuario son utilizados solo para los fines descritos en la sección de Funcionamiento de la aplicación.



4 ENTREGA DEL PROYECTO

4.1 Archivos y documentos entregados

En el momento de la entrega de la aplicación se hará entrega de una carpeta en la que se adjuntan los siguientes archivos y documentos.

- » Los archivos Apk-release y Apk-debug que son los ejecutables de la aplicación
- » Esta memoria técnica
- » La presentación del proyecto esta en la url:
<https://view.genial.ly/5eeb2ae9b508bf0d7f24c37b/presentation-mikagu>
- » Una carpeta que contiene diferentes documentos que hemos usado durante la creación del proyecto. Planificación, diseños,...
- » El webservice local que hemos usado para la elaboración de la App. Hay que tener en cuenta que hemos realizado un cambio en la webservice que nos proporcionó Maristik. El objeto response no tenía una propiedad status que devolviera el estado y que permitiera evaluar si la petición ha tenido éxito o no. En su lugar los mensajes de error y los datos venían en la misma propiedad records. Esto nos generaba un problema porque no nos limitamos a presentar los datos en un TextView sino que los separamos en diferentes campos específicos y que luego presentamos en una maquetación de forma gráfica. Si no hubiésemos añadido esa propiedad status, no sabríamos si debemos cargar los datos o no y en caso de error éste se mostraría en el lugar donde deberían ir los datos, estropeando el diseño de la interfaz gráfica. De esta manera podemos evaluar que la petición ha tenido éxito antes de cargar los datos y en caso de fallo, en lugar de cargar los datos lanzamos un mensaje de error. Comando para ejecutar el web service: npm start

4.2 Credenciales de cuentas

Hemos creado un único usuario cuyos datos se encuentran en la tabla usuario de la base de datos. Entendemos que esta aplicación es monousuario, es decir será instalada en un móvil de empresa y usada por un único usuario que será un empleado. Cada empleado tendrá su móvil y tendrá un sistema de autenticación para el caso de que terceras personas puedan acceder de forma no autorizada al dispositivo.

Credenciales ficticias para el primer acceso tras la instalación:

Nombre de usuario: Patxiku

Contraseña: 1



4.3 Propuesta de autenticación para un caso real

En un caso real al instalar la aplicación se crearía un registro en la tabla usuario con unos valores genéricos, por ejemplo nombre de usuario User1, password Admin. La primera vez que el usuario hiciera login se haría contra esos valores genéricos y a continuación se le pediría que modificara los valores con sus datos personales, nombre de usuario, nombre y apellido, dni y la url desde donde cargar su avatar o su foto. La siguiente vez que se logueara ya se haría contra sus credenciales.

4.4 Guía para cambios de parámetros de la aplicación

En parámetros de la aplicación incluimos el precio del kilómetro, el precio de la dieta para desplazamientos dentro de la Unión Europea, el precio de la dieta para desplazamientos fuera de la Unión Europea y las credenciales del usuario.

De nuevo, en un caso real tendríamos un formulario en una activity para cambiar estos parámetros y guardarlos en la tabla precios o en la tabla de usuario; en este proyecto lo hacemos por código porque no nos los piden en los requisitos. El lugar donde habría que ir a cambiar las credenciales de usuario sería la clase DBOpenHelper del paquete database. Y en el INSERT que se hace en la tabla usuario cambiar los valores del INSERT. Y el lugar donde tendríamos que cambiar los precios sería en la misma clase en el INSERT de la tabla precio cambiar los valores.

Para cambiar la Url del web service hay que acceder a la clase ActivarTar dentro del paquete ui y cambiarlo en las líneas 79, 101 y 115. Otra mejora que se podría implementar es introducir el dominio en una constante de aplicación.

4.5 Registros para pruebas

Cuando se instala la aplicación se crean automáticamente unos registros de pruebas, tanto de dietas como de gastos. Esas inserciones se hacen desde la clase DBOpenHelper del paquete database.



5. Conclusiones

Creemos que desde el punto de vista técnico la aplicación es funcional, las especificaciones técnicas han sido superadas y nuestros objetivos han sido cumplidos.

A la hora de acometer el proyecto nos hemos centrado primero en resolver las cuestiones técnicas:

- » Cómo hacer un listado con el componente RecyclerView y cómo asignar eventos a objetos desde las líneas del RecyclerView.
- » Cómo hacer uso de la geolocalización y cómo obtener la ubicación geográfica, concretamente el país.
- » Cómo hacer uso del protocolo http para consumir servicios web.
- » Cómo solventar el tema de guardar las fechas en SQLite que no tiene tipo fecha y hacer conversiones de String a milisegundos (long) y de milisegundos a String.

Tras resolver estos mínimos hemos intentado ir más allá de las especificaciones que nos pedían y aportar un plus de usabilidad haciendo una interfaz de usuario ágil, flexible y dinámica. También hemos hecho que la recuperación de la información sea lo más versátil posible.

Con esta forma de trabajar hemos podido dedicar tiempo a desarrollar nuestros propios objetivos y tratar de implementar una interfaz de usuario lo más realista posible. Creemos que hemos cumplido con los objetivos y estamos muy satisfechos con el resultado.

Como equipo nos hemos integrado bien. Ninguno de los dos teníamos experiencia en el desarrollo de un reto como este, pero nos hemos complementado a la hora de solucionar problemas técnicos y hemos tenido una comunicación fluida y continua.

Por todo ello creemos que hemos cumplido con los objetivos y estamos muy satisfechos con el resultado.