UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Título del Proyecto:

ShoppingList

Autor:

Asaustre García, Oscar

Director:

Tomás Gironés, Jesús

TESINA PARA LA OBTENCIÓN DEL TÍTULO DE: Diploma de Especialización en Desarrollo de Aplicaciones para Android

Septiembre del 2017

Contenido

Introducción

Descripción del problema

Realizar la lista de la compra ha sido siempre un acto muy habitual en los hogares, El proceso normal ha sido escribir en una hoja de papel los productos necesarios que se tiene que comprar en el supermercado, tienda, ...

Cada día más lo dispositivos móviles pasan a formar parte de nuestra vida cotidiana, empleándolos para realizar fotografías, leer el correo,.... Por este motivo surgió la idea de realizar una aplicación que controle la lista de la compra en una APP Móvil, pero aprovechando las tecnologías para ofrecer algunos servicios de valor añadido a la lista de la compra tradicional, como almacenar todas las lista de las compras, realizar el control de gastos, compartir listas entre varias personas,...

Objetivos

Permite organizar de una manera ágil la lista de la compra. Los productos que se agregan a la lista se puede realizar de tres formas diferentes:

- Introduciendo el texto en el cuadro de texto de la aplicación. Éste a partir de 3 caracteres te sugiere productos que ya están registrados (conforme vamos introduciendo productos estos quedan registrados para otras listas).
- A través del lector de código de barras. Se utiliza la cámara como lector de código de barras y localiza si está registrado en nuestro sistema ese producto recuperándolo en caso de existir. Si el producto no existe el sistema te sugiere que lo registres de manera manual con su código de barras para en posteriores ocasiones poder recuperarlo por este medio.
- A través del reconocedor de voz. Con la voz se puede introducir el producto, el sistema te sugiere varias palabras que encajan con lo dictado a través de la voz y es el usuario quien selecciona la palabra exacta.

Se puede gestionar múltiples listas muy útil para los casos que se realizan compras en diferentes comercios o se quiere tener registrado la compra de distintos días.

Cada mes, si el usuario ha activado las notificaciones en el menú de preferencias (el proceso se lanza todos los días 1 de cada mes) el sistema

lanza un servicio que envía una notificación informando de los gastos registrados en los últimos 6 meses agrupados por meses. El control de gastos también se puede acceder desde la aplicación, en donde se visualizan los gastos de los últimos 6 meses cumplidos, es decir el gasto del mes actual no se podrá visualizar hasta el mes siguiente.

- Poder compartir una lista de la compra entre varias personas, de tal manera que se pueda componer ésta en tiempo real por varios integrantes. Para ello se aprovechará de herramientas que se proporciona en la nube, en este caso firebase, para poder compartir

- Utilizar las aplicaciones externas más comunes (Gmail, Facebook y Twitter) para poder logarse en la aplicación.

Motivación

Durante la realización del curso pensando en que proyecto podía realizar me di cuenta que había un acto que realiza muy a menudo y era cada vez que iba a comprar anotaba en una libreta o en algunos casos en una aplicación de notas del móvil de la necesidades de compra del hogar.

De ahí surgió la idea de realizar una aplicación que controlara los elementos que me hacían falta cada vez que iba al supermercado a comprar pero encima darte la posibilidad de registrar lo que cuesta cada elementos comprado y te controlara el total de gastos de la lista, además de servicios de valor añadido como utilizar el control de voz para registrar elementos o recuperar los elementos por el código de barras de estos una vez que lo hayamos registrado en otra ocasión.

Situación de... / Tecnologías utilizadas

Incluir la información necesaria que facilite la comprensión del proyecto. Solo se incluirá este punto para describir alguna tecnología, paradigma de diseño, plataforma, ... no descrita durante el Diploma o Master. Su extensión será siempre reducida.

Algunas tecnologías utilizadas no descritas durante el curso son las siguientes:

- Existen dos maneras de almacenar la lista de la compra en el dispositivo.

    1. La primera es que el usuario realice el registro de los elementos de la lista de la compra de manera anónima (sin logarse en la aplicación). En este caso se utiliza una base de local en SQLite

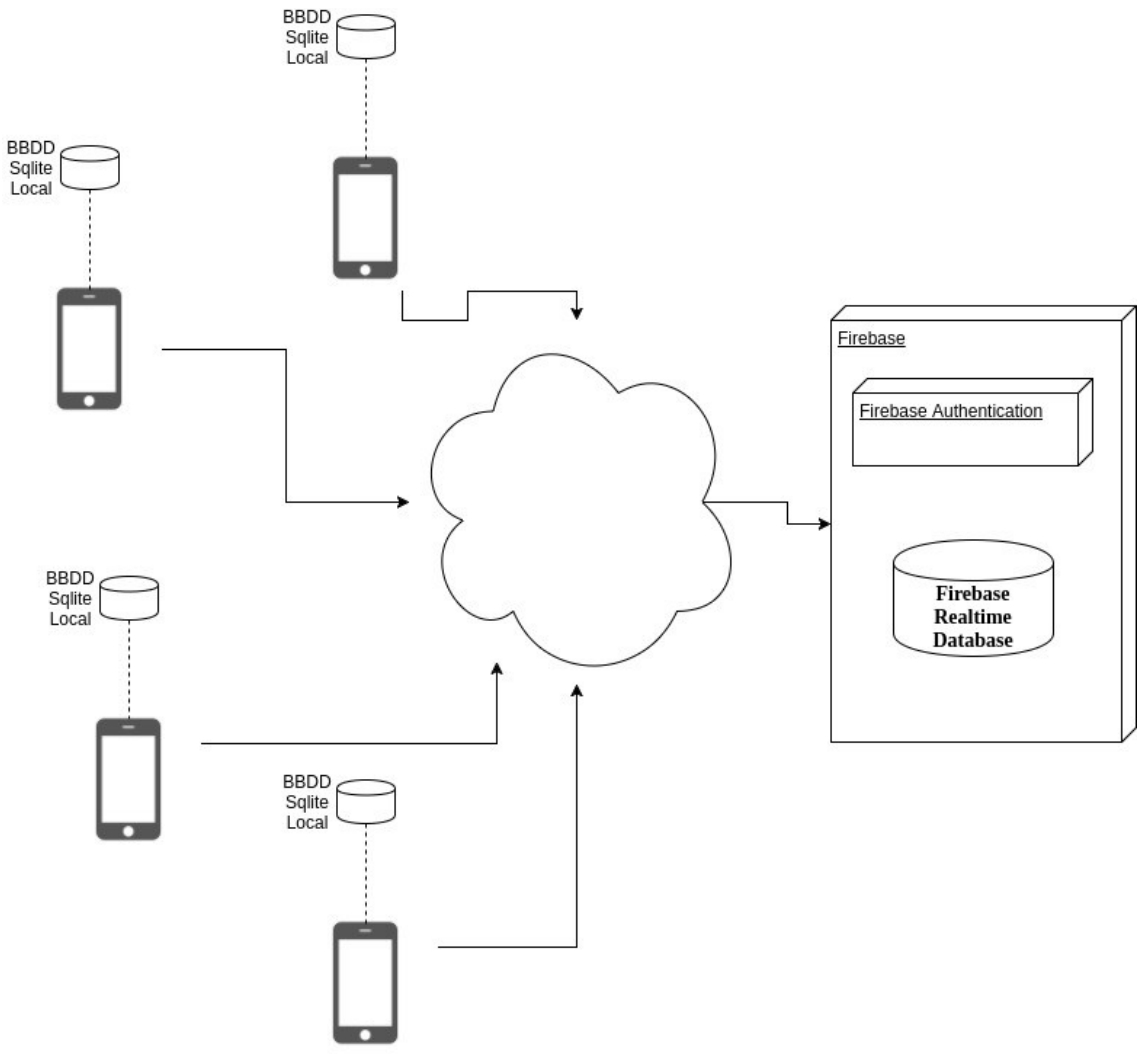
para almacenar la información. Para controlar y facilitar las operaciones que se realiza en la base de datos local se hace uso del **framework ORM GreenDao** (http://greenrobot.org/greendao/).

2. EL usuario se ha registrado y logado en la aplicación con un usuario y en este caso automáticamente la base de datos se almacena en la nube utilizando la plataforma Firebase y concretamente Firebase Realtime Database

- Se incorpora la librería de terceros **Zxing** (https://github.com/zxing/zxing), para la utilización de la cámara como lector de código de barras

Arquitectura de la aplicación

Esquema del diseño



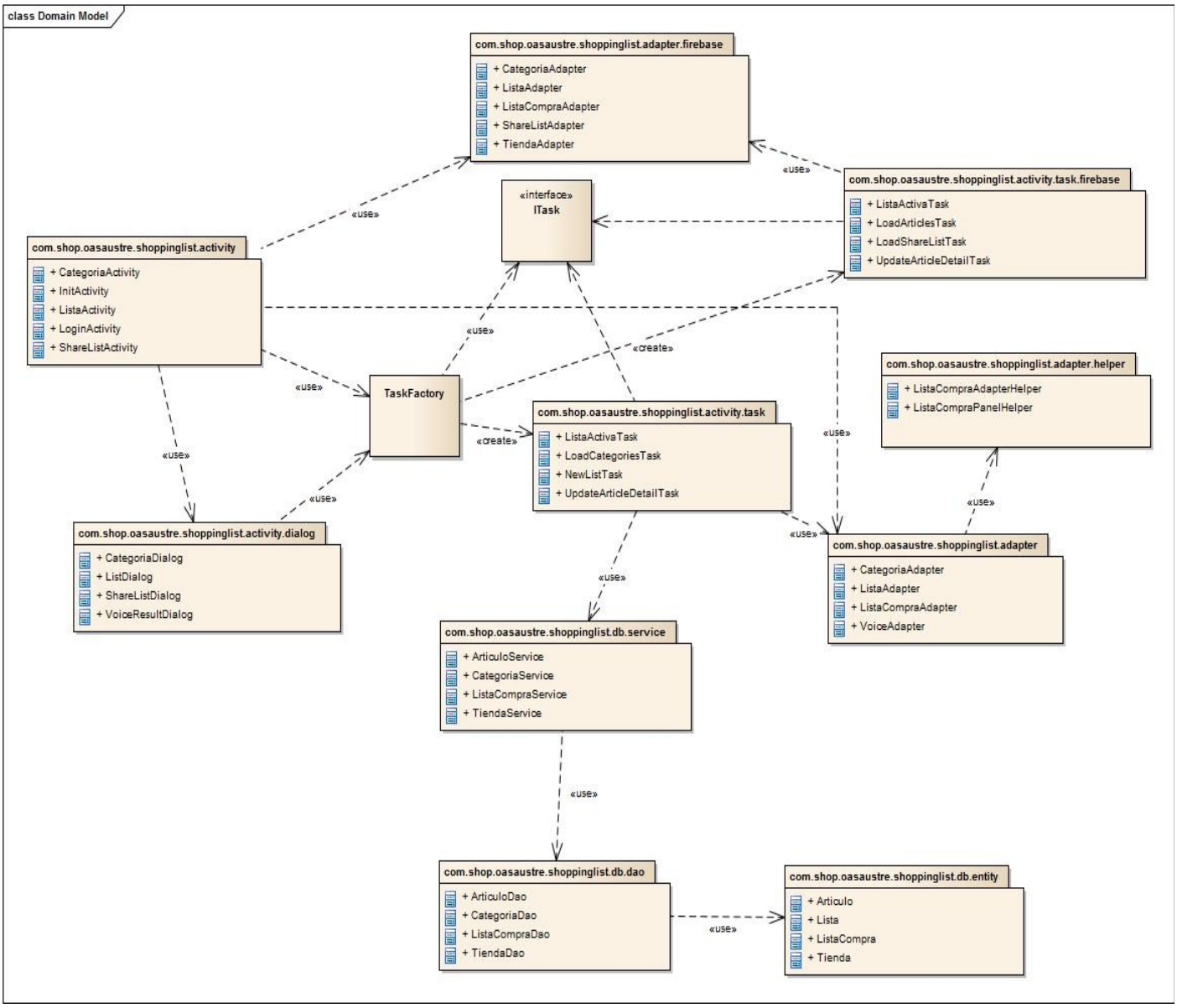
La aplicación puede trabajar en dos modos:

- El usuario no realiza el login en la aplicación, en este caso se utiliza una base de datos relacional (Sqlite) en el dispositivo del usuario para registrar los artículos en la lista de la compra y llevar el control de gastos. Para este caso se utiliza un framework ORM, greenDao, para realizar las operaciones de borrado, actualizar, insertar y consulta de entidades. Facilita el desarrollo y se elimina el código repetitivo de acceso y conexión a la base de datos.
- El usuario realiza el login en la aplicación. En este instante se hace uso del servicio Firebase Authentication para gestionar los usuarios y los datos son almacenados en la nube en la base de datos Firebase Realtime Database. Esta base de datos es común para todos los usuarios que han realizado el login en la aplicación, dando la posibilidad que varios usuarios puedan compartir su lista de la compra siendo actualizadas en tiempo real.

Además del almacenamiento de la información (en sqlite o firebase) existen otras series de características que ofrece la aplicación independiente del tipo de almacenamiento

- Incorpora una librería de terceros, Zxing, para la utilización de la cámara como lector de código de barras (es necesario conceder el permiso para trabajar con la cámara)
- La aplicación ha sido diseñada para verse correctamente en vertical y horizontal.
- Aunque no es necesario porque no se va a manejar gran volumen de información los accesos a la base de datos (local o firebase) se realizan desde hilos independientes al hilo principal existiendo AsynTask o Runnable según las necesidades.
- Se hace uso de Intenciones para la llamada entre actividades y funcionalidades incorporadas en el sistema, como por ejemplo el reconocedor de voz.
- Se hace uso de cuadro de diálogos (clase Dialog) personalizados.
- Se utilizan tanto actividades como fragment.
- Se hace uso del control NavigationView para la barra de menú lateral.
- Los iconos de la aplicación están en formato SVG.
- Diferentes Recyclerview según donde se recuperen los datos (local o firebase). Además estos están personalizados para que puedan realizar agrupaciones.

- Se hace uso de la clase ScheduledThreadPoolExecutor para el lanzamiento de la tarea periódica (1 vez al mes) del control de gastos.
- El diseño de la base de datos sqlite (dispositivos locales), se ha realizado desde una herramienta externa y la primera vez que se arranca la aplicación se copia automáticamente el fichero de base de datos que está en el directorio assets a la memoria interna de la aplicación. Con esto facilita los cambios en el diseño y la precarga de datos previos sin tener que hacerlo mediante código.
- Se pueden compartir listas entre varios usuarios que utilicen la aplicación en dispositivos independientes (a través del usuario de Firebase) de tal manera que las actualizaciones de éstas se haran en tiempo real en los dispositivos implicados.
- Se hace uso de Themes y estilos personalizados tanto para los textos, fondos, menús y controles.
- Se utiliza un servicio que es el encargado de enviar una notificación avisando que ya está disponible el control de gasto de último mes (se muestra la información de los últimos 6 meses).



La aplicación está compuesta de varias actividades que se encuentran localizadas en el paquete **com.shop.oasaustre.shoppinglist.activity**. Entre las actividades se destacan:

- InitActivity: Es la actividad principal y la primera que se visualiza al arrancar la aplicación. Es la que tiene el mayor peso de la aplicación y contiene el menú lateral, realiza la llamada a las otras actividades a través de Intenciones y recibe la respuestas del resultado del lector de código de barras y del reconocimiento de voz.

En cuanto el layout que carga este actividad (context_init.xml) está compuesto por un cuadro de texto de la pantalla principal es el núcleo de interacción de la aplicación y es donde se introducen los nuevos artículos o se recuperan artículos previamente registrados (el control es un AutoCompleteTextView que realiza sugerencias de artículos ya registrados). El resultado se guarda en base de datos a la lista de la compra y se incorpora al recyclerview que está debajo del cuadro de texto.

La comunicación tanto con el framework Zxing para el lector de código de barras como el reconocedor de voz se realiza con llamadas a través de intenciones con respuesta de resultados que son recogidas en la actividad InitActivity a través del método **onActivityResult.**

- LoginActivity: Es la actividad que realiza el login de la aplicación. Se hace uso de Firebase Authentication para autenticación y autorización de los usuarios. En la aplicación se ofrece logarse mediante 3 formas: Gmail, Facebook y Twitter.

- SettingsActivity: Es la actividad que contiene las preferencias de la aplicación.

El resto de actividades se utilizan fundamentalmente para realizar las operaciones básicas CRUD sobre los diferentes elementos que componen una lista de la compra.

Los accesos a la base de datos se realizan en hilos secundarios por medio de un AsynTask que se comunicará con el hilo principal para actualizar la interfaz de usuario. Aquí se distinguen dos paquetes:

- com.shop.oasaustre.shoppinglist.activity.task: Son las tareas que realizan el acceso a la base de datos Sqlite cuando el usuario realiza las operaciones en local.

- com.shop.oasaustre.shoppinglist.activity.task.firebase: Son las tareas que realizan el acceso a la base de datos Firebase Realtime Database para realizar las operaciones sobre ella cuando el usuario está registrado en la aplicación con lo que su información es almacenada en la nube.

Todos las clases de los dos paquetes descritos anteriormente implementan la clase **ITask** haciendo uso del patrón de diseño Command para que te de una implementación de la clase concreta según sea haga uso de base de datos en local o en la nube.

Para la instanciar las clases que implementan la interfaz ITask se ha hecho uso del patrón de diseño Factoría para que te proporcione la clase concreta. La clase **TaskFactory** es la encargada de la creación de los objetos.

Para el acceso a la base de datos SQLite en los dispositivos locales se ha utilizado el framework ORM GreenDao organizando el acceso a datos en los paquetes:

- **com.shop.oasaustre.shoppinglist.db.entity:** Son las entidades que representan las tablas de la base de datos.

- com.shop.oasaustre.shoppinglist.db.dao: Contiene los métodos para el acceso a los datos y se generan automáticamente cuando se compila el código (lo realiza el framework GreenDao).

- **com.shop.oasaustre.shoppinglist.db.service**: Contiene los servicios que hacen uso de los DAO’s y son los que se llaman en la aplicación para el acceso a la información de la base de datos.

Como se ha comentado anteriormente estos tres paquetes sólo se usan y tienen sentido cuando se realiza el acceso a la base de datos local Sqlite del dispositivo.

- com.shop.oasaustre.shoppinglist.activity.dialog: Contiene la lógica para crear las pantallas para crear las entidades auxiliares categorías, tiendas y listas de la compra que no contienen mucha información son emergentes y heredan de DialogFragment. Estos Dialog’s son formularios personalizados.

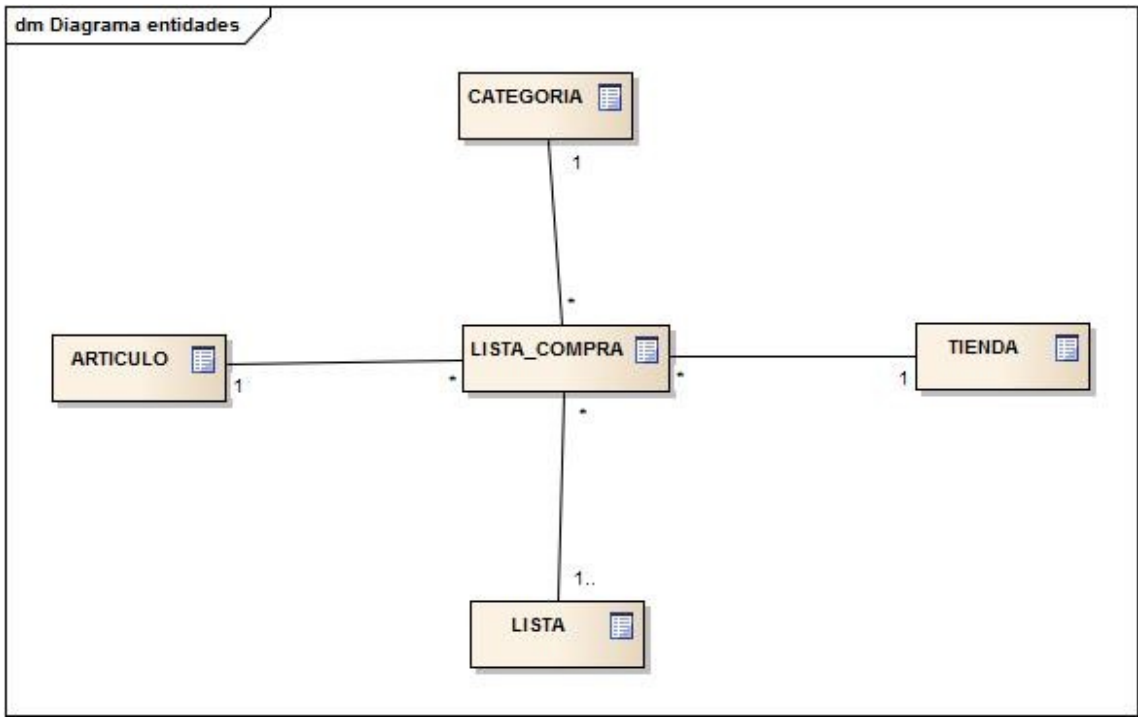
- com.shop.oasaustre.shoppinglist.adapter: Contiene los adaptadores que manejan los datos en los recyclerView. Estos RecyclerView son sólo utilizados para las base de datos en local (sqlite).

- com.shop.oasaustre.shoppinglist.adapter.firebase: Contiene los adaptadores de los RecyclerVIew que manejan los datos obtenidos en Firebase Realtime Database. Su uso es sólo cuando se realiza se está trabajando con datos en la nube.

Modelo de datos

En cuanto al modelo de datos siguiendo el diseño de la aplicación se puede distinguir entre el modelo de datos en SQLITE y el diseñado en la base de datos Firebase Realtime Database (NoSQL).

Modelo de datos SQLite



Entidad: Articulo	
CAMPO	Tipo de datos
ID	INTEGER PRIMARY KEY AUTOINCREMENT
NOMBRE	TEXT
CODIGOBARRAS	TEXT

Entidad: Categoría	
CAMPO	Tipo de datos
ID	INTEGER PRIMARY KEY AUTOINCREMENT

NOMBRE	TEXT

Entidad: Lista	
CAMPO	Tipo de datos
ID	INTEGER PRIMARY KEY AUTOINCREMENT
NOMBRE	TEXT
FECHA	INTEGER
ACTIVO	INTEGER

Entidad: Tienda	
CAMPO	Tipo de datos
ID	INTEGER PRIMARY KEY AUTOINCREMENT
NOMBRE	TEXT
DIRECCION	TEXT
POBLACION	TEXT
PROVINCIA	TEXT

Entidad: Tienda	
CAMPO	Tipo de datos
ID	INTEGER PRIMARY KEY AUTOINCREMENT

IDLISTA	INTEGER (FOREIGN KEY LISTA)
IDARTICULO	INTEGER (FOREIGN KEY ARTICULO)
IDCATEGORIA	INTEGER (FOREIGN KEY CATEGORIA)
IDTIENDA	INTEGER (FOREIGN KEY TIENDA)

Modelo de datos Firebase Realtime Database (NoSQL)

La base de datos se puede consultar en la siguiente url

https://shoppinglist-92c34.firebaseio.com/


```
{
 "articulo" : {
  "-Kuj0z7aOE09QPuUXqxw" : {
   "nombre" : "patatas",
   "uid" : "-Kuj0z7aOE09QPuUXqxw"
  }
 },
 "categoria" : {
  "-Ku_KBXT6sQ6DD7nn8HX" : {
   "nombre" : "tuberculos",
   "uid" : "-Ku_KBXT6sQ6DD7nn8HX"
  }
 },
 "compartida" : {
  "-Ku_DUGbDliLP7LHSMZc" : {
   "j8EPe3Pj6UWwAQYV7ylnkXzNcal2" : true
  },
  "-Ku_DtGI0dMtFNbbSicz" : {
   "j8EPe3Pj6UWwAQYV7ylnkXzNcal2" : false,
   "lF19DtEMAoMlWSwbr2hbNi4rHtE2" : true
```

```
  },
  "-Kua0quIsUEy4Ut3ML29" : {
    "Fn6fRuG933QKukT2Gpx0NgIKAGH2" : true
  }
},
"lista" : {
  "-Ku_DUGbDliLP7LHSMZc" : {
    "fecha" : 1506010591831,
    "nombre" : "Lista Compra",
    "uid" : "-Ku_DUGbDliLP7LHSMZc"
  },
  "-Ku_DtGI0dMtFNbbSicz" : {
    "fecha" : 1506010701457,
    "nombre" : "Lista Compra",
    "uid" : "-Ku_DtGI0dMtFNbbSicz"
  },
  "-Kua0quIsUEy4Ut3ML29" : {
    "fecha" : 1506024061337,
    "nombre" : "Lista Compra",
    "uid" : "-Kua0quIsUEy4Ut3ML29"
  }
},
"lista_compra" : {
  "-Kuj0z7ftEs_cVG5wNcY" : {
    "articulo" : {
      "articleName" : "patatas",
      "barcode" : "",
      "idArticle" : "-Kuj0z7aOE09QPuUXqxw"
```

```
    },
    "categoria" : {
      "categoryName" : "tuberculos",
      "idCategory" : "-Ku_KBXT6sQ6DD7nn8HX"
    },
    "lista" : {
      "idLista" : "-Ku_DtGI0dMtFNbbSicz",
      "listaName" : "Lista Compra"
    },
    "precio" : 15,
    "uid" : "-Kuj0z7ftEs_cVG5wNcY",
    "unidades" : 2
  }
},
"usuario" : {
  "Fn6fRuG933QKukT2Gpx0NgIKAGH2" : {
    "compartida" : {
      "-Kua0quIsUEy4Ut3ML29" : true
    },
    "name" : "Oscar Asaustre García",
    "provider" : "facebook.com",
    "uid" : "Fn6fRuG933QKukT2Gpx0NgIKAGH2"
  },
  "j8EPe3Pj6UWwAQYV7ylnkXzNcal2" : {
    "compartida" : {
      "-Ku_DUGbDliLP7LHSMZc" : true
    },
    "email" : "ginopalaka@gmail.com",
```

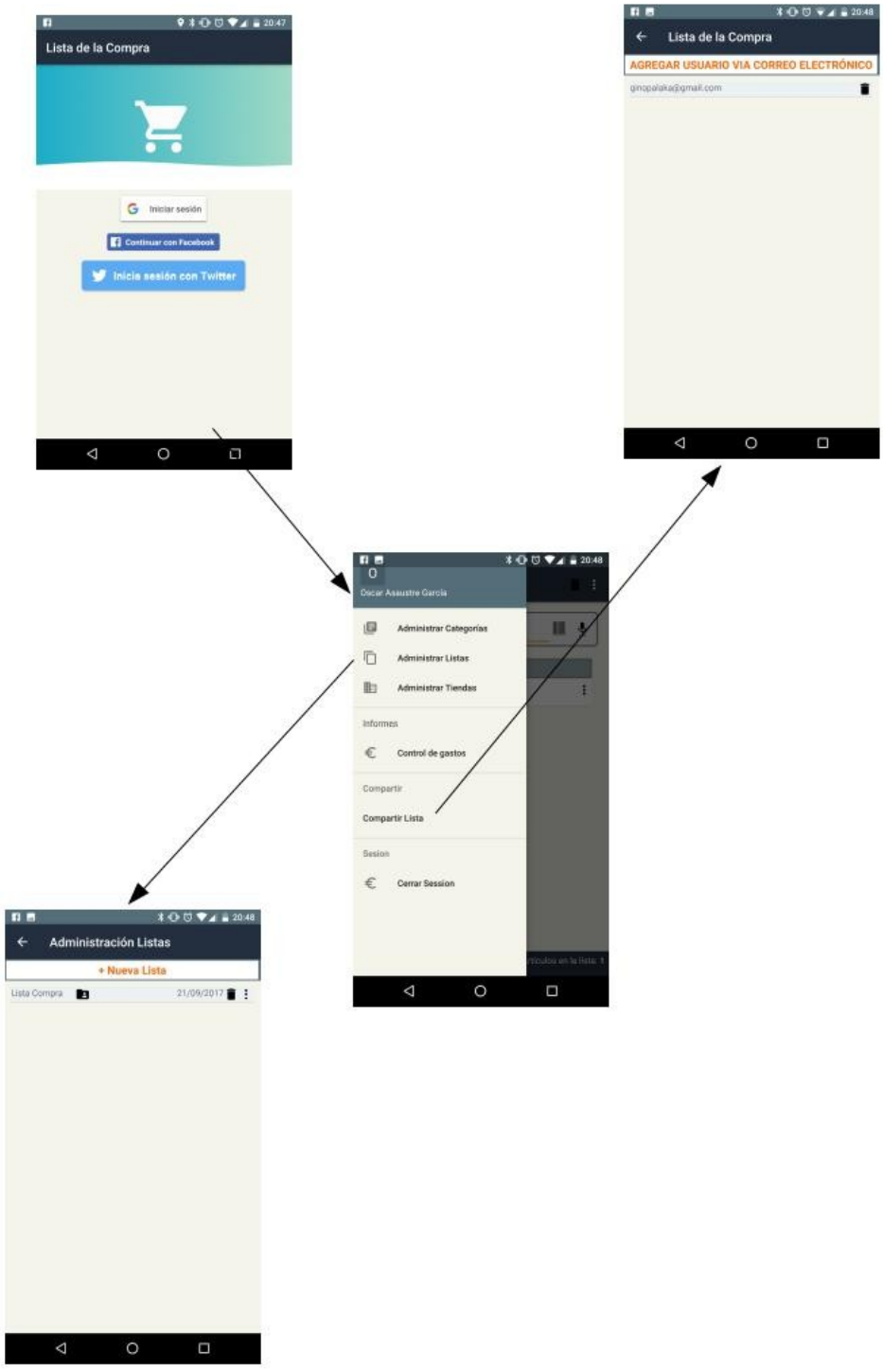
```
    "name" : "Gino Palaka",
    "provider" : "google.com",
    "uid" : "j8EPe3Pj6UWwAQYV7ylnkXzNcal2"
  },
  "lF19DtEMAoMlWSwbr2hbNi4rHtE2" : {
    "compartida" : {
      "-Ku_DtGI0dMtFNbbSicz" : true
    },
    "email" : "oscar.asaustre@gmail.com",
    "name" : "Oscar Asaustre García",
    "provider" : "google.com",
    "uid" : "lF19DtEMAoMlWSwbr2hbNi4rHtE2"
  }
 }
}
```

Vistas

Esquema principal

Lector código de barras,reconocimiento de voz y edición artículos

Compartir listas



Conclusiones

Grado de cumplimiento de los objetivos planteados:

Comencé esta ampliación del proyecto entregado en la asignatura de “Fundamentos de Android” con objetivos muy ambiciosos para mejorar la aplicación. Al final el tiempo ha dejado que el grado de cumplimiento de los objetivos haya dejado el proyecto en un 80% de lo que yo pensaba.

El proyecto partía de una base muy buena, era una aplicación totalmente operativa pero sólo trabajaba con base de datos en local (SQLite). Me planteé la posibilidad de ofrecer la dualidad a los usuarios de poder tener una base de datos en local (el usuario no quiere realizar el registro en la aplicación) y una base de datos en la nube con Firebase Realtime Database (para usuarios logados), con esto podía ofrecer la posibilidad de poder compartir la lista de la compra entre varios usuarios y poder ver los cambios en tiempo real. La migración a la base de datos de Firebase la he conseguido pero me ha consumido más tiempo del esperado y ha hecho que otras mejoras que tenía pensadas las haya dejado en el tintero por falta de tiempo. Aún así estoy totalmente satisfecho con el trabajo porque he logrado una aplicación totalmente operativa y estas mejoras, que a continuación las enumero en las líneas abiertas, son cambios que deben ser bastante menos costosos que la migración a Firebase Realtime Database.

Líneas abiertas:

- Cuando se comparte una lista de la compra enviar con Firebase Invites invitaciones a usuarios no registrados para que sepan que quieren compartir la lista de la compra con ellos.
- Cuando hay una lista compartida y se realiza una notificación, enviar notificaciones push al resto de usuarios implicados (en caso de tener la aplicación en segundo plano) para que sean informado de cambios en la lista compartida.
- Dar un poco más detalle en la listas compartidas, diciendo que usuarios son los que comparten la lista.
- Poder personalizar la interfaz de usuario entre un conjunto de colores predefinidos que se eliga en el menú de preferencias.
- Publicar la aplicación en Google Play Store.

Anexos

Listado de fuentes entregadas / Código fuente en GitHub

El código fuente se encuentra en la siguiente url:
https://github.com/oasaustre/shoppingList.git