



**Universidad
Europea**

LAUREATE INTERNATIONAL UNIVERSITIES

UNIVERSIDAD EUROPEA DE MADRID



ESCUELA ARQUITECTURA INGENIERÍA Y DISEÑO

CICLO FORMATIVO DE GRADO SUPERIOR
DESARROLLO DE APLICACIONES MULTIPLATAFORMA

PROYECTO FIN DE CICLO

EAT WELL

Autores

CURSO 2017-18

TÍTULO:

AUTORES: GUILLERMO REBUFFO CÉSPEDES

JUAN CORTES

LEÓN KARAGISHEV

TUTOR DEL PROYECTO: EASY FOOD

FECHA DE LECTURA:

CALIFICACIÓN

Fdo.

Ernesto Ramiro Córdoba
Tutor/a del Proyecto

RESUMEN

Cada vez más, llevar una alimentación equilibrada es primordial para las personas. Partiendo del actual contexto tecnológico imperante en nuestras sociedades, la utilización de una aplicación móvil permitiría atender y satisfacer eficientemente dicha demanda. Por todo ello, el presente proyecto ha sido pensado para fusionar la venta de comida saludable y su reparto a domicilio. Por un lado, definiendo un proveedor que elabore comida saludable y, por otra, delimitando un público objetivo interesado en adquirir hábitos alimenticios saludables pero cuya actividad diaria no les permite adquirirlos.

Esta aplicación está formada de diversas categorías: por un lado, permitirá elegir del proveedor de comida saludable una oferta variada – dietas proteicas, bajas en calorías, veganas, sin gluten, etc. – y unirlo inmediatamente al carrito de compra. Por otra parte, se podrá elegir la forma de pago que se desee, ya sea vía online o cuando la compra llegue al domicilio. Asimismo, contará con una funcionalidad que es el tracking GPS, que consiste en ver el pedido en tiempo real gracias a la API de Google Maps. Uno de los objetivos de esta aplicación es que el cliente tenga una sensación de confianza, inmediatez y tranquilidad a la hora de hacer cualquier compra.

ABSTRACT

This project has been designed for any type of audience, it is focused on a task that we carry out every day in our daily life and it is very important, called: "The time to eat", which this can help to do it from a more practical and quick way, you will not have to bother to cook or go running to the supermarket to make purchases for your meal of the day, week or month, this application solves all those problems.

We are talking about this application that is formed by various categories of food that you can choose, these meals are based on being healthy for the customer, we will have different types of categories such as protein-based, carbohydrate, low-calorie, vegan, gluten-free, meals for coeliacs, you will have the facility to choose any type of food and join it to your shopping cart, you have two payment options either online or when the purchase arrives at your home, we have added a functionality that is GPS tracking, this is that you can see your order in real time thanks to the Google Maps API, we are interested in the customer having this feeling of confidence and tranquility at the time of making any purchase.

AGRADECIMIENTOS

Un agradecimiento especial a nuestros profesores que nos apoyaron a lo largo de nuestra formación y particularmente a aquellos que no dejaron de incentivarnos en seguir adelante con este proyecto, a pesar de las dificultades que se nos presentaron.



Esta obra se distribuye bajo una licencia Creative Commons.

Se permite la copia, distribución, uso y comunicación de la obra si se respetan las siguientes condiciones:

- Se debe reconocer explícitamente la autoría de la obra incluyendo esta nota y su enlace.
- La copia será literal y completa
- No se podrá hacer uso de los derechos permitidos con fines comerciales, salvo permiso expreso de los autores.

El texto precedente no es la licencia completa sino una nota orientativa de la licencia Original completa (jurídicamente válida) que puede encontrarse en:
<http://creativecommons.org/licenses/by-nc-nd/3.0/deed.es>

INDICE DE CONTENIDOS

1. Introducción	1
1.1. Objetivos	7
1.2. Motivación	8
1.3. Antecedentes	8
2. Desarrollo de la aplicación	12
2.1. Funcionalidades Aplicación	14
2.2. Herramientas Tecnológicas	15
2.3. Descripción del trabajo realizado	19
2.4. Resultados y Validación	23
2.5. Capturas de la Aplicación Cliente	41
2.6. Capturas Aplicación Servidor	53
3. Conclusiones	55
3.1. Innovación	55
3.2. Trabajo futuro	55
4. Biblioteca y Webgrafía	56

1. Introducción

El propósito de este proyecto es que las personas puedan acceder a una aplicación de fácil manejo que les permita comprar comida saludable, online desde sus dispositivos Android.

1.1. Objetivos

1. Contará con una base de datos en **Firestore** donde se almacenarán categorías de las diferentes comidas ofertadas, y los datos personales de los usuarios registrados.
2. Un "Modelo" de clases como estructura de nuestra aplicación: User, Food, Order, Category, Request.
3. Registro actualizado de los usuarios de la aplicación.
4. El usuario podrá realizar un seguimiento en tiempo real de su pedido. En este caso, utilizaremos la *API de Maps Google*.
5. Una aplicación de fácil manejo y utilitaria.

El logro de estos objetivos, entre otras cosas, permitirá tener una relación fluida y de retroalimentación con los clientes, a través de nuestra ventana de *Rating*.

1.2. Motivación

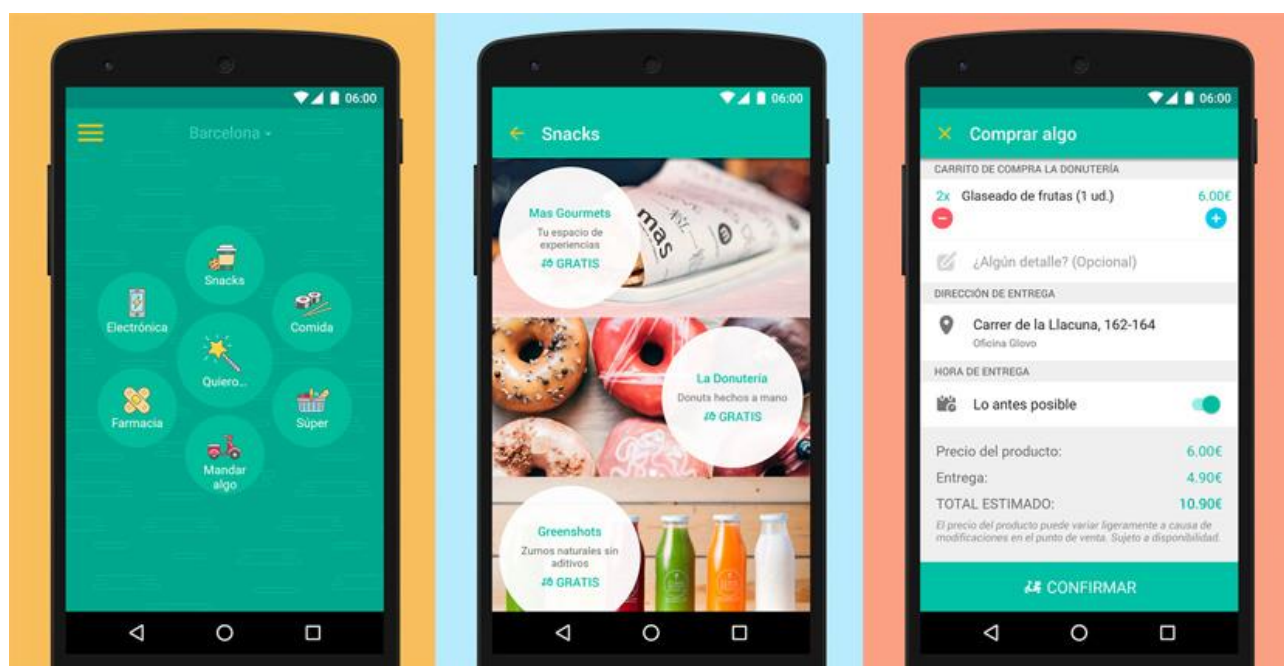
Proveer una aplicación innovadora de "fácil manejo y utilitaria", al margen de las ya existentes, que permitirá comprar alimentos saludables garantizados por un proveedor reputado.

1.3. Antecedentes

En la actualidad existen muchas aplicaciones con prestaciones "delivery", que ofertan la venta de comida, pero no específicamente de comida saludable con un proveedor garantizado. La aplicación "Glovo", cuenta con una categoría denominada "Fit Food", que permite elegir, entre sus diversos proveedores, comida sana. Sin embargo, nuestra aplicación iría construyendo a corto y mediano plazo una cartera de clientes con rutinas de alimentación establecidas y de proveedores de productos alimenticios garantizados.

Glovo

Presentación de la aplicación GLOVO.



Categoría Fit Food (Aplicación/Website)



Fit Food

Look good naked

Compra de las comidas

Fit Food

TOP VENTAS

Ofertas Combinaciones

Poke Bowl

Salad Bar

Bowls



TOP VENTAS

Bowl Açaí Peanut Power (500ml)

Açaí, plátano, mantequilla de cacahuete, leche de ...

7,00 €

Bowl El Favorito (500ml)

Açaí, plátano, leche de anacardos, miel, crumb... d ...

7,50 €

Bowl Purple Youth (500ml)

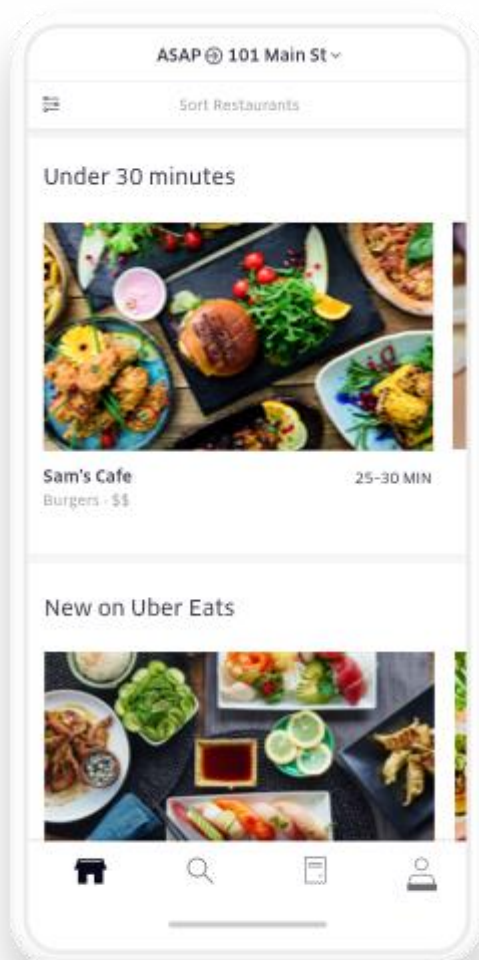
Arándanos, plátano, leche de anacardos, crumble de ...

7,00 €

Uber Eats

Otra aplicación muy parecida que aplica la misma metodología se llama Uber Eats donde puedes buscar las comidas por los restaurantes pero no especifica justamente el tipo de comidas, en si es más general.

Uber Eats



La aplicación tiene una sección llamada *Healthy Delivery* donde podremos encontrar los restaurantes que tienen productos de comida saludable ya tu allí puedes combinar los platos que más de adapten a ti y al tipo de deporte o ejercicio que estés haciendo.

Healthy Delivery in Madrid



Panela & Co
€ • Saludable

30-40 min 4.4 ★ (68)



Ohanasana
€ • Saludable

35-45 min 4.3 ★ (100+)



Magasand - Retiro
€ • Saludable

25-35 min 4.5 ★ (66)



Juicy Avenue - Fuencarral
€ • Saludable

20-30 min 4.3 ★ (100+)



Juicy Avenue - Hortaleza
€ • Saludable

35-45 min 3.4 ★ (27)



Magasand - Chueca
€ • Saludable

30-40 min

2. DESARROLLO DEL PROYECTO

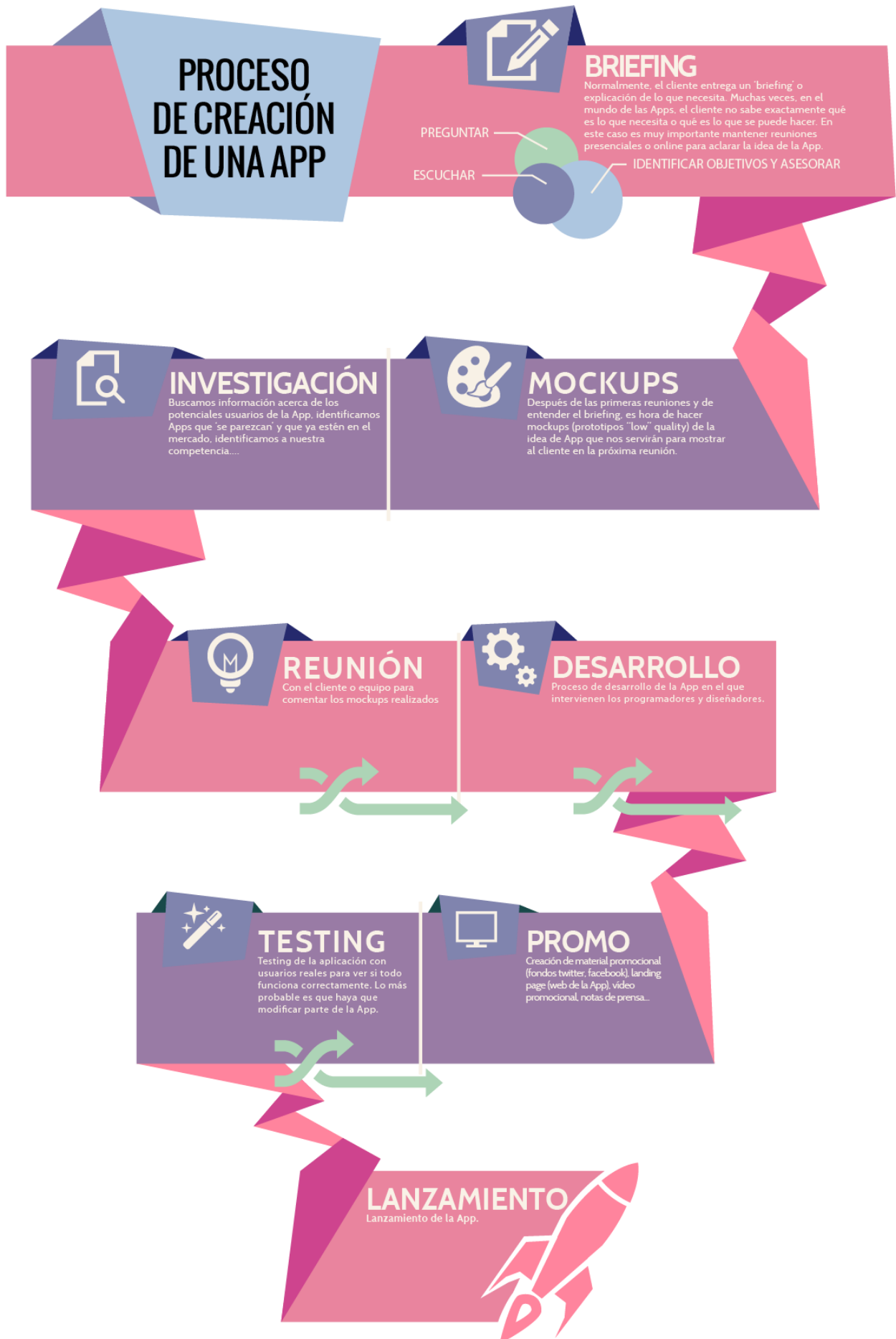
El desarrollo de nuestra aplicación tiene diferentes fases, la primera es una pantalla de *Splash* que desde aquí ya reflejaremos de lo que consiste nuestra aplicación mostrando una *imagen (Logo)*, esto será un transición de unos segundos y a continuación pasaremos a la siguiente pantalla donde tendremos dos botones, uno para *registrarnos* y el otro para el *Login*, en caso que elijamos registrarnos tendremos que introducir tres datos, que será nuestro nombre, teléfono y contraseña todos estos datos se guardaran en nuestra base de datos en caso que quieras recuperar tu contraseña, en la pantalla de *Login* solo necesitaras introducir tu teléfono y contraseña.

Al iniciar sesión tendrás la opción de Recordar contraseña para no tener la molestia de hacer *Login* repetidas veces.

Una vez que iniciamos sesión nos mostrara directamente las categorías de nuestras comidas, cada categoría dispondrá una lista de comidas, haciendo click en algunas de ellas tendrás la opción de adquirir cuantas quieras y visualizar su precio, en el lado izquierdo superior (toolbar), tendrás un icono de la aplicación que desde ahí podrás desplegar un home drawer donde verás 4 botones: Menú, Carta, Pedidos, Salir. Al seleccionar una comida al carrito tienes que introducir tu dirección donde quieras que se te envíe la/s compra, esto orden será enviada al repartidor y al servidor automáticamente.

Nosotros para gestionar este proyecto hemos construido 3 Aplicaciones, cada una de ellas tiene su funcionalidad: (*Cliente, Servidor*) y el lado del *Shipper o repartidor*.

PROCESO DE CREACIÓN DE UNA APP



2.1. ***Funcionalidades Aplicación***

El lado del Cliente

En este lado tendrás las opciones de registro y acceso para usar la aplicación, mostrará en pantalla las categorías de las comidas que dispondrás para elegir, puedes añadir tus comidas favoritas al carrito de las compras, esto sumara todas tus órdenes y te dará la opción de pagar con tarjeta de crédito o dinero en efectivo al llegar tu pedido, también podrás ver el tracking de tu pedido en tiempo real e incluso cancelar la orden máximo en un cierto tiempo, tienes la opción de poder restablecer tu contraseña con el código que introduzcas al principio del registro.

El lado del Servidor

Este lado viene a ser el más importante de la aplicación ya que desde aquí gestionaremos el lado del Cliente y Repartidor, aquí tendremos los privilegios de modificar la aplicación Cliente como por ejemplo:

- Modificar Categorías
- Actualizar Imágenes
- Borrar Categorías
- Enviar Ordenes
- Cancelar Ordenes
- Enviar Notificaciones en tiempo Real

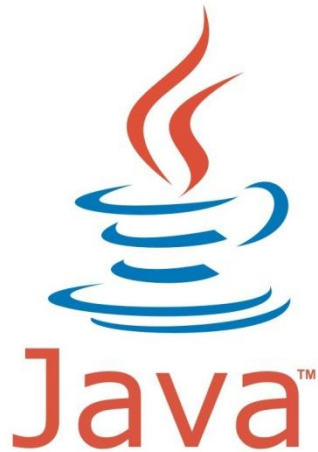
En si tenemos el control total y para acceder dispondremos de un número y una contraseña que solo tendrán acceso los administradores, esto es gracias a que hemos declarado una variable en la clase de Usuarios con permisos "true" para solo el lado del servidor.

El lado del Shipper

Este lado viene a ser el lado del repartidor, estará encargado de llevar los pedidos, en su pantalla visualizara las órdenes de los clientes para poder llegar al destino. No tiene privilegios de administración del sistema.

2.2. *Herramientas Tecnológicas*

JAVA



Java es el lenguaje base con lo que trabajaremos y está construida nuestra aplicación, este lenguaje es multiplataforma, está orientado a objetos, es un lenguaje robusto y de alto nivel.

Lo habitual es que las aplicaciones Java se encuentren compiladas en un **bytecode** (un fichero binario que tiene un programa ejecutable), aunque también pueden estar compiladas en código máquina nativo.



Es un sistema de gestión de base de datos relacional compatible con **ACID** (son las características de los parámetros que permiten clasificar las transacciones de los sistemas de gestión de base de datos).

Este sistema de gestión estará incorporado en nuestra aplicación para poder gestionar nuestras bases de datos.

Consideramos como puntos clave para la utilización de este motor de base de datos los siguientes:

Configuración sencilla: Una vez instalado este motor de base de datos no requiere configuración de rutas, tamaños, puertos, entre otros puntos que por lo general configuramos al inicio de una instalación de cualquiera otro motor. Por ejemplo: SQL Server, MySQL y Oracle DB, reduciendo de forma significativa todos aquellos esfuerzos sobre la administración.

No demanda el soporte de un servidor: Implementa una serie de librerías que se encargan de la gestión y por ende no ejecuta procesos para administrar la información.

Es Software Libre: Por ser de código abierto, tanto los archivos de compilación como las instrucciones de escalabilidad, se encuentran disponibles para toda la comunidad de desarrolladores.

Genera un archivo para el esquema: SQLite almacena toda la base de datos en un archivo único multiplataforma, siendo este punto una gran ventaja en cuanto a temas de seguridad y migración, puesto que los datos de las apps desarrolladas para Android no son accedidos por contextos externos, así mismo simplifica las copias de seguridad y los procesos de migración.

Almacena los datos de forma persistente: Permitiendo que aunque se apague el dispositivo una vez se encienda los datos persistan y se encuentren correctos en la aplicación.



Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA . Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones que aumentan tu productividad durante la compilación de apps para Android, como las siguientes:

- Un sistema de compilación basado en Gradle flexible
- Un emulador rápido con varias funciones
- Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android
- Instant Run para aplicar cambios mientras tu app se ejecuta sin la necesidad de compilar un nuevo APK
- Integración de plantillas de código y GitHub para ayudarte a compilar funciones comunes de las apps e importar ejemplos de código
- Gran cantidad de herramientas y frameworks de prueba
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versión, etc.
- Compatibilidad con C++ y NDK
- Soporte incorporado para Google Cloud Platform, lo que facilita la integración de Google Cloud Messaging y App Engine
- En esta página encontrarás una introducción a las funciones básicas de Android Studio. Para acceder a un resumen de los últimos cambios, consulta Notas de la versión de Android Studio.



Firestore se trata de una plataforma móvil creada por Google, cuya principal función es desarrollar y facilitar la creación de apps de elevada calidad de una forma rápida, con el fin de que se pueda aumentar la base de usuarios y ganar más dinero. La plataforma está subida en la nube y está disponible para diferentes plataformas como iOS, Android y web. Contiene diversas funciones para que cualquier desarrollador pueda combinar y adaptar la plataforma a medida de sus necesidades.

Firestore se inició cuando Google la compró en 2014, y seguidamente la fue mejorando mediante la compra del equipo de Divshot.

Comunicación

Nuestra comunicación para este proyecto se ha dado gracias a 3 aplicaciones que son: Slack, WhatsApp, y Skype.

Slack. Lo hemos utilizado para seguir el transcurso de nuestro proyecto con nuestro Profesor Ernesto y los pasos a seguir en nuestra aplicación.



Slack es una herramienta de comunicación en equipo creada por Stewart Butterfield, Eric Costello, Cal Henderson, y Serguei Mourachov. Slack surge como una herramienta interna utilizada por la compañía Tiny Speck en el desarrollo de Glitch, un juego en línea actualmente obsoleto. Slack es lanzado al mercado en agosto del 2013 y consiguió un registro de 8000 clientes en las primeras 24 horas.

WhatsApp._ Esta aplicación la hemos utilizado para subir nuestros avances que hemos estado haciendo en la aplicación.

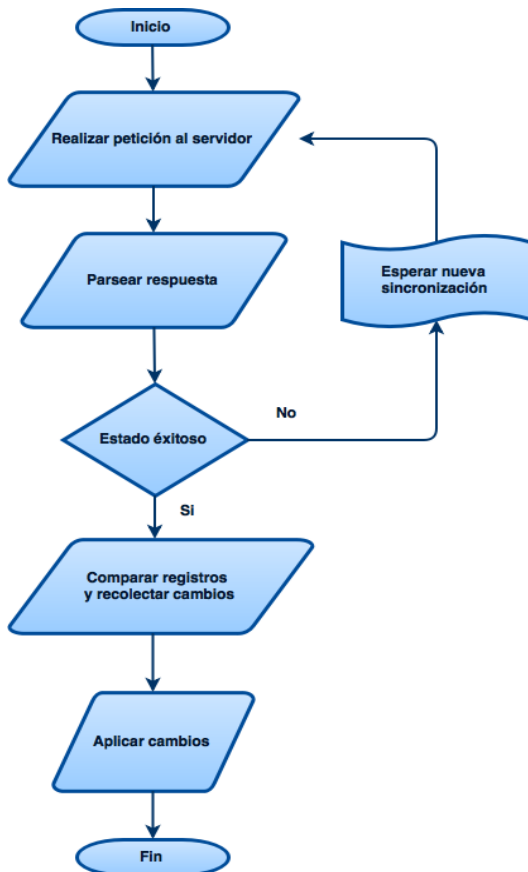


@Actualapp

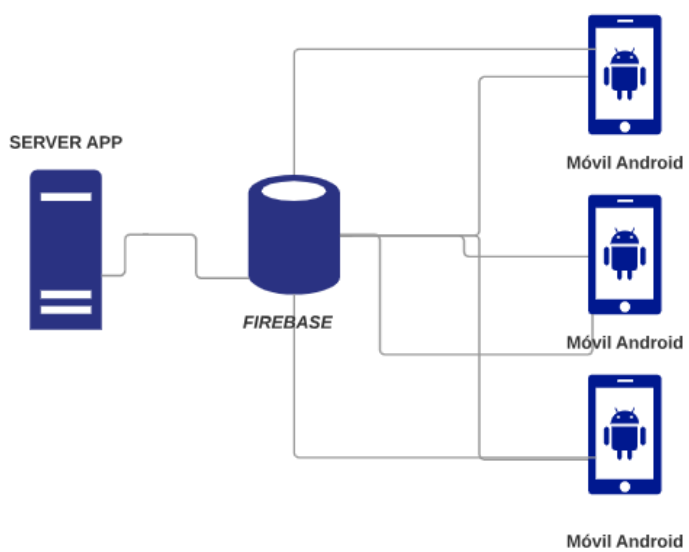
Skype._ Esta aplicación la hemos utilizado para reuniones del grupo y aclarar temas, ideas, ayudas entre nosotros.

2.3. Descripción del trabajo realizado

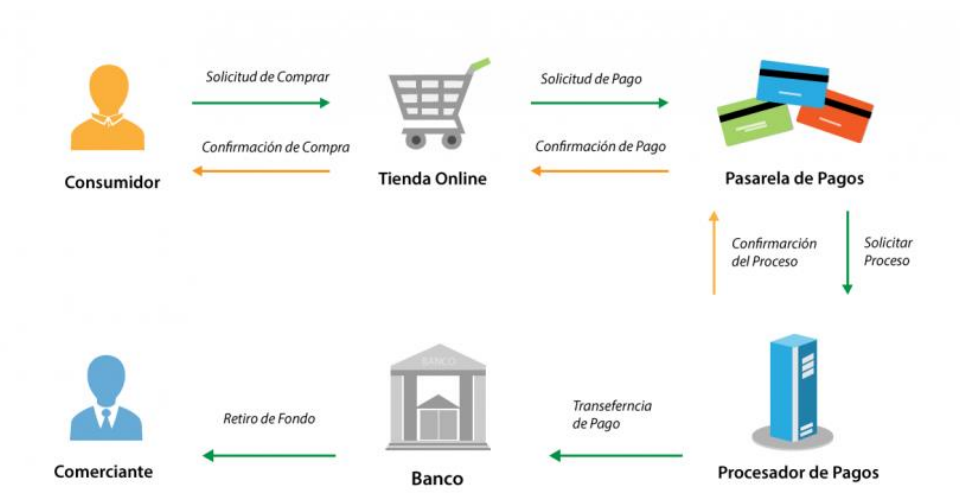
Diagrama de Flujo



Esquema de datos



Esquema de pagos, (PASARELA)



Clases Utilizadas

El MODELO

Para entender la base de la aplicación empezaremos por la **clase User** que es donde declaramos los atributos que utilizarán los clientes que usen nuestra aplicación, estos serán: nombre, password, teléfono, código de seguridad; en caso que tengamos que restablecer la contraseña. Aquí también tenemos un atributo que es el **isStaff** que hace referencia a la Aplicación Servidor, donde este se encarga de que tengamos acceso como administrador ya que está con un usuario y contraseña declarado en modo "true" para nosotros.

La **clase Categoría** es donde se guardan los nombres e imágenes de las comidas de la aplicación; esto estará compuesto por un CardView donde se mostrarán las diferentes categorías de comidas que tenemos, esto estará enlazado con la **clase Food** donde

están nuestros platos con sus respectivos nombres, precios, imágenes, Id y descuento. La **clase Order** guarda la compra para enviarla a través de la **clase Request**. Tenemos la **clase Notification** que se encarga de enviar las notificaciones entre Cliente y Servidor sobre pedidos y ordenes esto es gracias a una **Clase Sender** que sera la encargada de enviar y recibir las notificaciones, **La clase Rating** es para puntuar las comidas que hemos pedido y ayudar a los demas usuarios a base de la valoración de los clientes.

DATABASE

La clase Database esta compuesta por una lista de ordenes y con los atributos ProductName, ProductId,Quantity,Discount que estas seran los que interactuaran a la hora de realizar los metodos que hemos implementado como por ejemplo :

- Añadir a cartas
- Borrar carta
- Añadir a favoritos
- Borrar Favoritos
- Es favorite

COMMON

En esta clase una de las cosas mas importantes es definir la API de Google que usaremos para el tracking del pedido, tambien hemos utilizamos un metodo para chequear la conexion a internet en caso que el telefono no este recibiendo datos o no tenga conexion WI-FI y detallar el estatus del pedido si esta en proceso, en camino o hecho.

ItemClickListener(Interface)

Esta clase que hemos implementado hace como referencia a clases donde se utilizan los metodos de *OnItemClickListener* para seleccionar objetos, en si cualquier objeto que seleccionaremos llama a esta clase.

Remote

Aqui tenemos clase RetrofitClient que esta sera la encargada de hacer las peticiones GET, POST, PUT, PATCH, DELETE, Y HEAD, y gestionar diferentes tipos de parametros y parsear automaticamente la respuesta a un POJO.

Service

En service tenemos las clases *MyFirebaseIDService* que crea un token del usuario para luego usarlo guardando su número telefonico en la base de datos y luego referenciarlo

con la clase *MyFirebaseMessaging* que se encarga de enviar las notificaciones a los usuarios.

ViewHolder

Patrón de diseño de *ViewHolder* se utiliza para acelerar la representación de tu *ListView* – en realidad para que funcione sin problemas, *findViewById* es bastante costoso (se analiza el DOM) cuando se utiliza cada vez que se representa un elemento de lista, debe desplazar la jerarquía de diseño y también instanciar objetos. Dado que las listas pueden volver a dibujar sus elementos con bastante frecuencia durante el desplazamiento, tales gastos generales podrían ser sustanciales. Estas clases nos ayudara a mejorar el performance y la experiencia de nuestros usuarios.

2.4.Resultados y Validación

Desarrollo Aplicación Cliente

Clase Database.java

```
public class Database extends SQLiteOpenHelper {
    private static final String DB_NAME="BBDDSQL.db";
    private static final int DB_VERSION=1;

    public Database(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
    }

    public List<Order> getCarts(){
        SQLiteDatabase db = getReadableDatabase();
        SQLiteQueryBuilder qb = new SQLiteQueryBuilder();

        String [] sqlSelect = {"ProductName", "ProductId", "Quantity", "Price", "Discount"};
        String sqlTable = "OrderDetail";

        qb.setTables(sqlTable);
        Cursor c = qb.query(db, sqlSelect, null, null, null, null, null);
```

```

final List<Order> result = new ArrayList<>();
if(c.moveToFirst()){
    do{
        result.add(new Order(c.getString(c.getColumnIndex("ProductId")),
            c.getString(c.getColumnIndex("ProductName")),
            c.getString(c.getColumnIndex("Quantity")),
            c.getString(c.getColumnIndex("Price")),
            c.getString(c.getColumnIndex("Discount"))));

        } while (c.moveToNext());
    }
    return result;
}

public void addToCart(Order order){
    SQLiteDatabase db = getReadableDatabase();
    String query = String.format("INSERT INTO OrderDetail(ProductId, ProductName, Quantity,
Price, Discount) VALUES ('%s', '%s', '%s', '%s', '%s');",
        order.getProductId(),
        order.getProductName(),
        order.getQuantity(),
        order.getPrice(),
        order.getDiscount());
    db.execSQL(query);
}

public void cleanCart(){
    SQLiteDatabase db = getReadableDatabase();
    String query = String.format("DELETE FROM OrderDetail");
    db.execSQL(query);
}

//Favorites
public void addToFavorites (String foodId){
    SQLiteDatabase db = getReadableDatabase();
    String query = String.format("INSERT INTO Favorites (FoodId) VALUES (%s)", foodId);
    db.execSQL(query);
}

public void reomoveToFavorites (String foodId){
    SQLiteDatabase db = getReadableDatabase();
    String query = String.format("DELETE FROM Favorites WHERE FoodId='%s'", foodId);
    db.execSQL(query);
}

public boolean isFavorite (String foodId){
    SQLiteDatabase db = getReadableDatabase();
    String query = String.format("SELECT * FROM Favorites WHERE FoodId='%s'", foodId);
    Cursor cursor = db.rawQuery(query, null);
    if (cursor.getCount() <= 0){

```



```

        cursor.close();
        return false;
    }
    cursor.close();
    return true;
}
}

```

Clase ItemClickListener.java

```

public interface ItemClickListener {
    void onClick (View view, int position, boolean isLongClick);
}

```

Clase Common.java

```

public class Common {
    public static User currentUser;

    private static final String BASE_URL = "https://fcm.googleapis.com/";

    public static APIService getFCMService(){
        return RetrofitClient.getClient(BASE_URL).create(APIService.class);
    }

    public static String convertCodeToStatus(String status){
        if(status.equals("0"))
            return "Orden hecha";
        else if(status.equals("1"))
            return "En camino";
        else
            return "Enviado";
    }

    public static final String DELETE = "Delete";
    public static final String USER_KEY = "User";
    public static final String PWD_KEY = "Password";

    public static boolean isConnectedToInternet(Context context) {

        ConnectivityManager connectivityManager = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);

        if (connectivityManager != null) {

            NetworkInfo[] info = connectivityManager.getAllNetworkInfo();
            if (info != null) {
                for (int i = 0; i < info.length; i++) {
                    if (info[i].getState() == NetworkInfo.State.CONNECTED)
                        return true;
                }
            }
        }
    }
}

```

```

    }
    }
    }

    return false;
}
}

```

Clase *APIService.java*

```

public interface APIService {
    @Headers(
        {
            "Content-Type:application/json",
            "Authorization:key=AAAAATcp9ZjU:APA91bFF07z4XsykX-
6wPYGfVx_dTHSe22oOQS8ATXuMo77k_0eA3mXtFCXkq93nKXQSIkqlpzVUkpwnMiuUSNCtvwfNz
N5KGaoubSWdeE29M2PJ36gLRoyFgOw6Cwf6KOIG0gQyx5jv"
        }
    )
    @POST("fcm/send")
    Call<MyResponse> sendNotification(@Body Sender body);
}

```

Clase *RetrofitClient.java*

```

public class RetrofitClient {
    private static Retrofit retrofit=null;

    public static Retrofit getClient(String baseUrl){
        if(retrofit == null){
            retrofit = new Retrofit.Builder()
                .baseUrl(baseUrl)
                .addConverterFactory(GsonConverterFactory.create())
                .build();
        }
        return retrofit;
    }
}

```

Clase *MyFirebaseIDService.java*

```

public class MyFirebaseIdService extends FirebaseInstanceIdService {
    @Override
    public void onTokenRefresh() {
        super.onTokenRefresh();
        String tokenRefreshed = FirebaseInstanceId.getInstance().getToken();
    }
}

```

```

        if(Common.currentUser != null)
            updateTokenToFirebase(tokenRefreshed);
    }

    private void updateTokenToFirebase(String tokenRefreshed) {
        FirebaseDatabase db = FirebaseDatabase.getInstance();
        DatabaseReference tokens = db.getReference("Tokens");
        Token token = new Token(tokenRefreshed, false); //false because this token send from
Client app
        tokens.child(Common.currentUser.getPhone()).setValue(token);
    }
}

```

Clase MyFirebaseMessaging.java

```

public class MyFirebaseMessaging extends FirebaseMessagingService {
    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        super.onMessageReceived(remoteMessage);
        sendNotification(remoteMessage);
    }

    private void sendNotification(RemoteMessage remoteMessage) {
        RemoteMessage.Notification notification = remoteMessage.getNotification();
        Intent intent = new Intent(this, MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent,
PendingIntent.FLAG_ONE_SHOT);

        Uri defaultSoundUri =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
        NotificationCompat.Builder builder = new NotificationCompat.Builder(this)
            .setSmallIcon(R.mipmap.ic_launcher_round)
            .setContentTitle(notification.getTitle())
            .setContentText(notification.getBody())
            .setAutoCancel(true)
            .setSound(defaultSoundUri)
            .setContentIntent(pendingIntent);

        NotificationManager noti =
(NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
        noti.notify(0, builder.build());
    }
}

```

Clase CartViewHolder.java

```

class CartViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener
,View.OnCreateContextMenuListener {

    public TextView txt_cart_name, txt_price;
    public ImageView img_cart_count;

```

```

private ItemClickListener itemClickListener;

public void setTxt_cart_name(TextView txt_cart_name) {
    this.txt_cart_name = txt_cart_name;
}

public CartViewHolder(View itemView) {
    super(itemView);
    txt_cart_name = (TextView) itemView.findViewById(R.id.cart_item_name);
    txt_price = (TextView) itemView.findViewById(R.id.cart_item_Price);
    img_cart_count = (ImageView) itemView.findViewById(R.id.cart_item_count);

    itemView.setOnCreateContextMenuListener(this);
}

@Override
public void onClick(View v) {

}

@Override
public void onCreateContextMenu(ContextMenu contextMenu, View view,
ContextMenuItem contextMenuItem) {
    contextMenu.setTitle("Select Action");
    contextMenu.add(0,0,getAdapterPosition(),Common.DELETE);

}
}

public class CartAdapter extends RecyclerView.Adapter<CartViewHolder>{

    private List<Order> listData = new ArrayList<>();
    private Context context;

    public CartAdapter(List<Order> listData, Context context) {
        this.listData = listData;
        this.context = context;
    }

    @Override
    public CartViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        LayoutInflater inflater = LayoutInflater.from(context);
        View itemView = inflater.inflate(R.layout.cart_layout, parent, false);
        return new CartViewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(CartViewHolder holder, int position) {
        TextDrawable drawable = TextDrawable.builder()
            .buildRound(""+listData.get(position).getQuantity(), Color.RED);
    }
}

```

```

        holder.img_cart_count.setImageDrawable(drawable);

        Locale locale = new Locale("en", "US");
        NumberFormat fmt = NumberFormat.getCurrencyInstance(locale);
        int price =
(Integer.parseInt(listData.get(position).getPrice()))*(Integer.parseInt(listData.get(position).getQuantity()));
        holder.txt_price.setText(fmt.format(price));

        holder.txt_cart_name.setText(listData.get(position).getProductName());
    }

    @Override
    public int getItemCount() {
        return listData.size();
    }
}

```

Clase CartAdapter.java

class CartViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener, View.OnCreateContextMenuListener {

```

    public TextView txt_cart_name, txt_price;
    public ImageView img_cart_count;

    private ItemClickListener itemClickListener;

    public void setTxt_cart_name(TextView txt_cart_name) {
        this.txt_cart_name = txt_cart_name;
    }

    public CartViewHolder(View itemView) {
        super(itemView);
        txt_cart_name = (TextView) itemView.findViewById(R.id.cart_item_name);
        txt_price = (TextView) itemView.findViewById(R.id.cart_item_Price);
        img_cart_count = (ImageView) itemView.findViewById(R.id.cart_item_count);

        itemView.setOnCreateContextMenuListener(this);
    }

    @Override
    public void onClick(View v) {

    }

    @Override
    public void onCreateContextMenu(ContextMenu contextMenu, View view, ContextMenu.ContextMenuInfo contextMenuInfo) {
        contextMenu.setHeaderTitle("Select Action");
        contextMenu.add(0,0,getAdapterPosition(),Common.DELETE);
    }
}

```

```

    }
}

```

```

public class CartAdapter extends RecyclerView.Adapter<CartViewHolder>{

    private List<Order> listData = new ArrayList<>();
    private Context context;

    public CartAdapter(List<Order> listData, Context context) {
        this.listData = listData;
        this.context = context;
    }

    @Override
    public CartViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        LayoutInflater inflater = LayoutInflater.from(context);
        View itemView = inflater.inflate(R.layout.cart_layout, parent, false);
        return new CartViewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(CartViewHolder holder, int position) {
        TextDrawable drawable = TextDrawable.builder()
            .buildRound("" + listData.get(position).getQuantity(), Color.RED);
        holder.img_cart_count.setImageDrawable(drawable);

        Locale locale = new Locale("en", "US");
        NumberFormat fmt = NumberFormat.getCurrencyInstance(locale);
        int price =
(Integer.parseInt(listData.get(position).getPrice()))*(Integer.parseInt(listData.get(position).getQuantity()));
        holder.txt_price.setText(fmt.format(price));

        holder.txt_cart_name.setText(listData.get(position).getProductName());
    }

    @Override
    public int getItemCount() {
        return listData.size();
    }
}

```

Clase MainActivity.java

```

public class MainActivity extends AppCompatActivity {
    FButton btnSignIn, btnSignUp;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}

```

```

setContentView(R.layout.activity_main);

btnSignIn = (FButton) findViewById(R.id.btnSignIn);
btnSignUp = (FButton) findViewById(R.id.btnSignUp);

//Init Paper
Paper.init(this);

btnSignIn.setOnClickListener(new View.OnClickListener() { //llamamos al botón
    @Override
    public void onClick(View v) {

    }
});

btnSignUp.setOnClickListener(new View.OnClickListener() { //llamamos al botón
    @Override
    public void onClick(View v) {
        Intent signUp = new Intent (MainActivity.this,SignUp.class);
        startActivity(signUp);

    }
});

btnSignIn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent signIn = new Intent(MainActivity.this,SignIn.class);
        startActivity(signIn);

    }
});

//Check Recordar
String user = Paper.book().read(Common.USER_KEY);
String pwd = Paper.book().read(Common.PWD_KEY);
if(user != null && pwd != null){
    if(!user.isEmpty() && !pwd.isEmpty())
        login(user, pwd);
}
}

private void login(final String phone, final String pwd) {
    //Iniciamos Firebase
    final FirebaseDatabase database = FirebaseDatabase.getInstance();
    //uso final para definir una entidad que solo se puede asignar una vez
    final DatabaseReference table_user = database.getReference("User");

    if (Common.isConnectedToInternet(getBaseContext())){
        final ProgressDialog mDialog = new ProgressDialog(MainActivity.this);
        mDialog.setMessage("Porfavor espera...");
        mDialog.show();

        table_user.addValueEventListener(new ValueEventListener() {

```

```

@Override
public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
    //Chequemos si el usuario no existe en el db
    if (dataSnapshot.child(phone).exists()) {
        //Get User Information
        mDialog.dismiss();
        User user = dataSnapshot.child(phone).getValue(User.class);
        user.setPhone(phone); //setPhone
        if (user.getPassword().equals(pwd)) {
            Intent homeIntent = new Intent(MainActivity.this, Home.class);
            Common.currentUser = user;
            startActivity(homeIntent);
            finish();
        } else {
            Toast.makeText(MainActivity.this, "Password Incorrecto",
Toast.LENGTH_SHORT).show();
        }
    } else {
        Toast.makeText(MainActivity.this, "El usuario no existe",
Toast.LENGTH_SHORT).show();
    }
}

@Override
public void onCancelled(@NonNull DatabaseError databaseError) {

}

});

}
else
{
    Toast.makeText(MainActivity.this, "Please Check your
connection!", Toast.LENGTH_SHORT).show();
    return;
}
}
}
}

```

Clase SignIn.java

```

public class SignIn extends AppCompatActivity {
    EditText edtPhone, edtPassword;
    Button btnSignIn;
    CheckBox ckbRemember;
    TextView txtForgotPwd;

    FirebaseDatabase database;
    DatabaseReference table_user;

    @Override

```



```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_sign_in);

    edtPassword=(MaterialEditText)findViewById(R.id.edtPassword);
    edtPhone=(MaterialEditText)findViewById(R.id.edtPhone);
    btnSignIn=(Button)findViewById(R.id.btnSignIn);
    ckbRemember=(CheckBox) findViewById(R.id.ckbRemember);
    txtForgotPwd = (TextView) findViewById(R.id.txtForgotPwd);

    //Init Paper
    Paper.init(this);

    //Recibimos datos
    Intent i = getIntent();
    String telefono = i.getStringExtra("Telefono");
    String password = i.getStringExtra("password");
    edtPhone.setText(telefono);
    edtPassword.setText(password);

    //Iniciamos Firebase

    database = FirebaseDatabase.getInstance();

    //uso final para definir una entidad que solo se puede asignar una vez
    table_user = database.getReference("User");

    txtForgotPwd.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            showForgotPwdDialog();
        }
    });

    btnSignIn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            if (Common.isConnectedToInternet(getBaseContext())){

                //Guardar usuario y contraseña
                if(ckbRemember.isChecked()){
                    Paper.book().write(Common.USER_KEY, edtPhone.getText().toString());
                    Paper.book().write(Common.PWD_KEY, edtPassword.getText().toString());
                }

                final ProgressDialog mDialog = new ProgressDialog(SignIn.this);
                mDialog.setMessage("Porfavor espera...");
                mDialog.show();
            }
        }
    });
}

```

```

table_user.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        //Chequeamos si el usuario no existe en el db
        if (dataSnapshot.child(edtPhone.getText().toString()).exists()) {
            //Get User Information
            mDialog.dismiss();
            User user = dataSnapshot.child(edtPhone.getText().toString()).getValue(User.class);
            user.setPhone(edtPhone.getText().toString()); //setPhone
            if (user.getPassword().equals(edtPassword.getText().toString())) {
                Intent homeIntent = new Intent(SignIn.this, Home.class);
                Common.currentUser = user;
                startActivity(homeIntent);
                finish();
            } else {
                Toast.makeText(SignIn.this, "Password Incorrecto",
                    Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText(SignIn.this, "El usuario no existe",
                Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {

    }
});

}
else
{
    Toast.makeText(SignIn.this, "Please Check your
connection!", Toast.LENGTH_SHORT).show();
    return;
}

});

}

private void showForgotPwdDialog() {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Forgot Password");
    builder.setMessage("Enter your secure code");

    LayoutInflater inflater = this.getLayoutInflater();
    View forgot_view = inflater.inflate(R.layout.forgot_password_layout, null);

```

```

        builder.setView(forgot_view);
        builder.setIcon(R.drawable.ic_security_black_24dp);

        final MaterialEditText edtPhone = (MaterialEditText)
forgot_view.findViewById(R.id.edtPhone);
        final MaterialEditText edtSecureCode = (MaterialEditText)
forgot_view.findViewById(R.id.edtSecureCode);

        builder.setPositiveButton("SI", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                //Comprobar si el usuario esta disponible
                table_user.addValueEventListener(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                        User user =
dataSnapshot.child(edtPhone.getText().toString()).getValue(User.class);

                        if (user.getSecureCode().equals(edtSecureCode.getText().toString())){
                            Toast.makeText(SignIn.this, "Tu contraseña: " + user.getPassword(),
Toast.LENGTH_LONG).show();
                        } else {
                            Toast.makeText(SignIn.this, "Error del codigo de seguridad",
Toast.LENGTH_SHORT).show();
                        }
                    }

                }

                @Override
                public void onCancelled(@NonNull DatabaseError databaseError) {

                }
            }
        });

        builder.setNegativeButton("NO", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {

            }
        });

        builder.show();
    }

```

Clase SignUp.java

```

public class SignUp extends AppCompatActivity {

    MaterialEditText edtPhone, edtName, edtPassword, edtSecureCode;
    Button btnSignUp;

```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_sign_up);

    edtName = (MaterialEditText)findViewById(R.id.edtName);
    edtPhone=(MaterialEditText)findViewById(R.id.edtPhone);
    edtPassword=(MaterialEditText)findViewById(R.id.edtPassword);
    edtSecureCode=(MaterialEditText)findViewById(R.id.edtSecureCode);

    //Boton Registrarse
    btnSignUp = (Button)findViewById(R.id.btnSignUp);

    //Iniciamos Firebase

    final FirebaseDatabase database = FirebaseDatabase.getInstance();
    //uso final para definir una entidad que solo se puede asignar una vez
    final DatabaseReference table_user = database.getReference("User");

    btnSignUp.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            //Comprobamos conexion a internet
            if(Common.isConnectedToInternet(getBaseContext())) {

                final ProgressDialog mDialog = new ProgressDialog(SignUp.this);
                mDialog.setMessage("Porfavor espera...");
                mDialog.show();
                //Enviamos datos
                Intent i = new Intent(SignUp.this, SignIn.class);
                i.putExtra("Telefono", edtPhone.getText().toString());
                i.putExtra("password", edtPassword.getText().toString());
                startActivity(i);

                table_user.addValueEventListener(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                        //Chqueamos si esta el telefono movil del user
                        if (dataSnapshot.child(edtPhone.getText().toString()).exists()) {

                            mDialog.dismiss();
                            Toast.makeText(SignUp.this, "Numero ya registrado",
                                Toast.LENGTH_SHORT).show();

                        } else {

```

```
mDialog.dismiss();
User user = new User(edtName.getText().toString(),
    edtPassword.getText().toString(),
    edtSecureCode.getText().toString());
table_user.child(edtPhone.getText().toString()).setValue(user);
Toast.makeText(SignUp.this, "Registro Completo",
    Toast.LENGTH_SHORT).show();
finish();
}
}

@Override
public void onCancelled(@NonNull DatabaseError databaseError) {

}
});
}
else{
    Toast.makeText(SignUp.this,"Please Check your
connection!",Toast.LENGTH_SHORT).show();
    return;
}
}
});
}
}
```

Clase OrderStatus.java

```
public class OrderStatus extends AppCompatActivity {
    public RecyclerView recyclerView;
    public RecyclerView.LayoutManager layoutManager;

    FirebaseRecyclerAdapter<Request, OrderViewHolder> adapter;

    FirebaseDatabase database;
    DatabaseReference requests;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_order_status);

        //Firebase
        database = FirebaseDatabase.getInstance();
        requests = database.getReference("Requests");

        recyclerView = (RecyclerView) findViewById(R.id.listOrders);
        recyclerView.setHasFixedSize(true);
        layoutManager = new LinearLayoutManager(this);
        recyclerView.setLayoutManager(layoutManager);

        //If we start OrderStatus activity from Home Activity
        //We will not put any extra, so we just loadOrder by phone from Common
```

```

        if (getIntent() == null){
            loadOrders(Common.currentUser.getPhone());
        } else {
            loadOrders(getIntent().getStringExtra("userPhone"));
        }
    }

    private void loadOrders(String phone){
        adapter = new FirebaseRecyclerAdapter<Request, OrderViewHolder>(
            Request.class,
            R.layout.order_layout,
            OrderViewHolder.class,
            requests.orderByChild("phone")
                .equalTo(phone)
        ) {
            @Override
            protected void populateViewHolder(OrderViewHolder viewHolder, Request model, int position) {
                viewHolder.txtOrderId.setText(adapter.getRef(position).getKey());

                viewHolder.txtOrderStatus.setText(Common.convertCodeToStatus(model.getStatus()));
                viewHolder.txtOrderAddress.setText(model.getAddress());
                viewHolder.txtOrderPhone.setText(model.getPhone());
            }
        };
        recyclerView.setAdapter(adapter);
    }
}

```

Clase Home.java

```

public class Home extends AppCompatActivity
    implements NavigationView.OnNavigationItemSelectedListener {

    FirebaseDatabase database;
    DatabaseReference category;

    TextView txtFullName;
    RecyclerView recycler_menu;
    RecyclerView.LayoutManager layoutManager;

    FirebaseRecyclerAdapter<Category, MenuViewHolder> adapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        toolbar.setTitle("Menu");
        setSupportActionBar(toolbar);

        //Init Firebase
        database = FirebaseDatabase.getInstance();
    }
}

```

```

category = database.getReference("Category");

Paper.init(this);

FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent cartIntent = new Intent(Home.this, Cart.class);
        startActivity(cartIntent);
    }
});

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
    this, drawer, toolbar, R.string.navigation_drawer_open,
    R.string.navigation_drawer_close);
drawer.addDrawerListener(toggle);
toggle.syncState();

NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
navigationView.setNavigationItemSelectedListener(this);

//Set Name for user
View headerView = navigationView.getHeaderView(0);
txtFullName = (TextView)headerView.findViewById(R.id.txtFullName);
txtFullName.setText(Common.currentUser.getName());

//Load menu
recycler_menu = (RecyclerView) findViewById(R.id.recycler_menu);
recycler_menu.setHasFixedSize(true);
layoutManager = new LinearLayoutManager(this);
recycler_menu.setLayoutManager(layoutManager);

if(Common.isConnectedToInternet(this))
    loadMenu();

else {
    Toast.makeText(Home.this, "Please Check your connection!",
    Toast.LENGTH_SHORT).show();
    return;
}

updateToken(FirebaseInstanceId.getInstance().getToken());
}

private void updateToken(String token) {
    FirebaseDatabase db = FirebaseDatabase.getInstance();
    DatabaseReference tokens = db.getReference("Tokens");
    Token data = new Token(token, false); //false because this token send from Client app
    tokens.child(Common.currentUser.getPhone()).setValue(data);
}

```

```

    }

    private void loadMenu(){
        adapter = new FirebaseRecyclerAdapter<Category, MenuViewHolder>(Category.class,
        R.layout.menu_item, MenuViewHolder.class, category) {
            @Override
            protected void populateViewHolder(MenuViewHolder viewHolder, Category model, int
            position) {
                viewHolder.txtMenuName.setText(model.getName());
                Picasso.with(getBaseContext()).load(model.getImage())
                    .into(viewHolder.imageView);
                final Category clickItem = model;
                viewHolder.setItemClickListener(new ItemClickListener() {
                    @Override
                    public void onClick(View view, int position, boolean isLongClick) {
                        //Get CategoryId and send to new Activity
                        Intent foodList = new Intent(Home.this, FoodList.class);
                        //Because CategoryId is key, so we just get key of this item
                        foodList.putExtra("CategoryId", adapter.getRef(position).getKey());
                        startActivity(foodList);
                    }
                });
            }
        };
        recycler_menu.setAdapter(adapter);
    }

    @Override
    public void onBackPressed() {
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.home, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if(item.getItemId() == R.id.refresh)
            loadMenu();

        return super.onOptionsItemSelected(item);
    }

```



```

@SuppressWarnings("StatementWithEmptyBody")
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    // Handle navigation view item clicks here.
    int id = item.getItemId();

    if (id == R.id.nav_menu) {

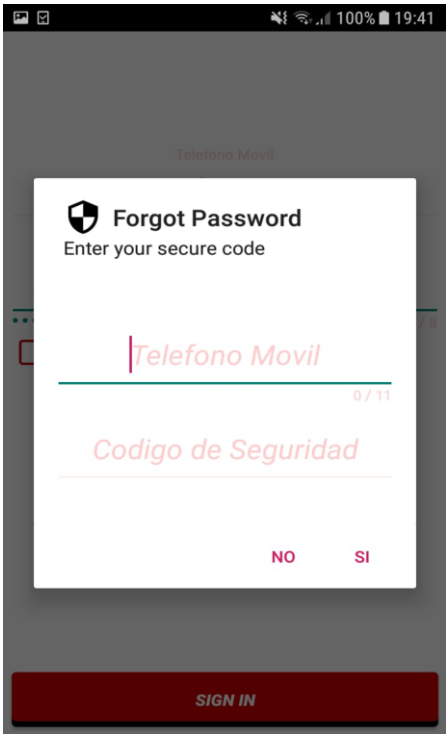
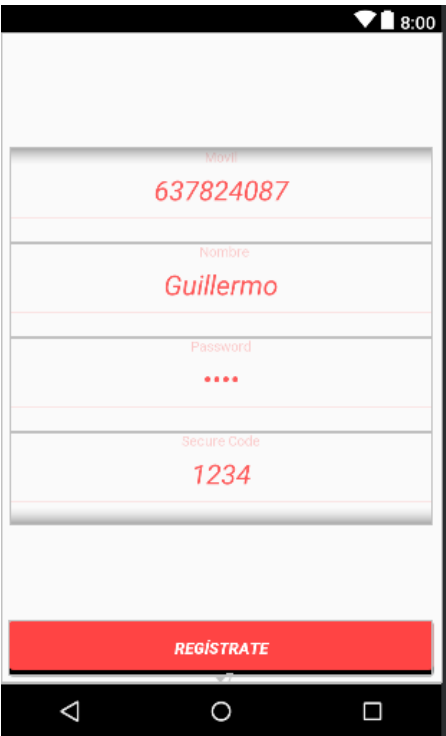
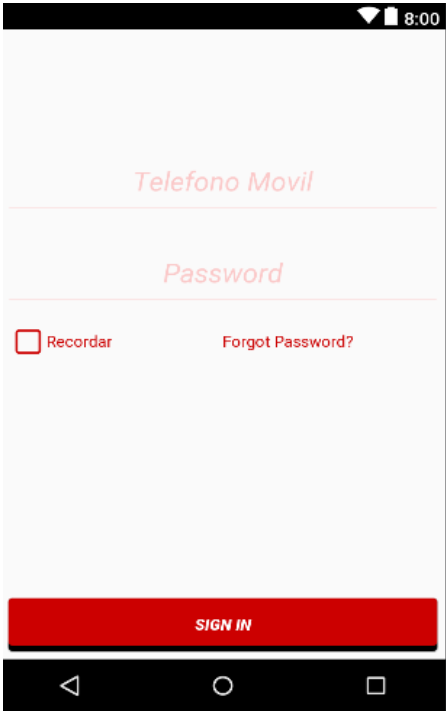
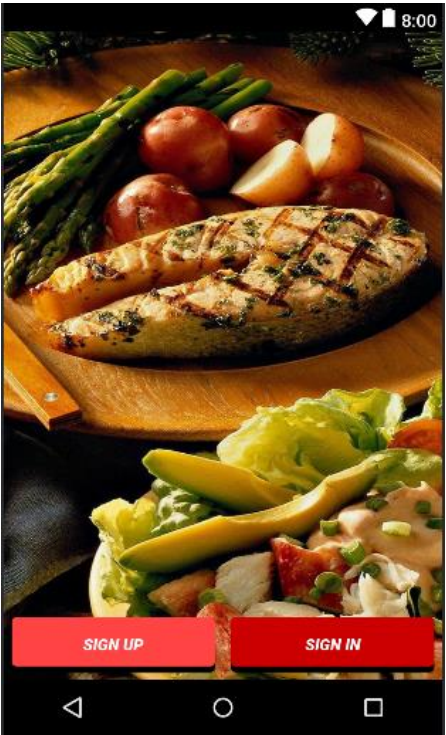
    } else if (id == R.id.nav_cart) {
        Intent cartIntent = new Intent(Home.this, Cart.class);
        startActivity(cartIntent);
    } else if (id == R.id.nav_orders) {
        Intent orderIntent = new Intent(Home.this, OrderStatus.class);
        startActivity(orderIntent);
    } else if (id == R.id.nav_log_out) {
        //Borrar guardado usuario y contraseña
        Paper.book().destroy();

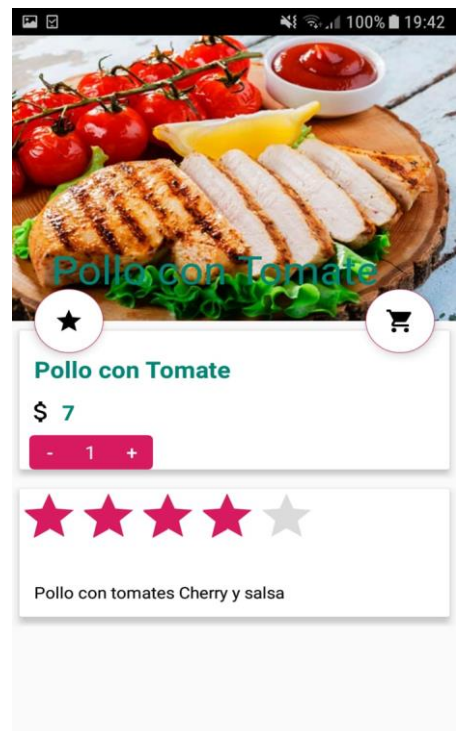
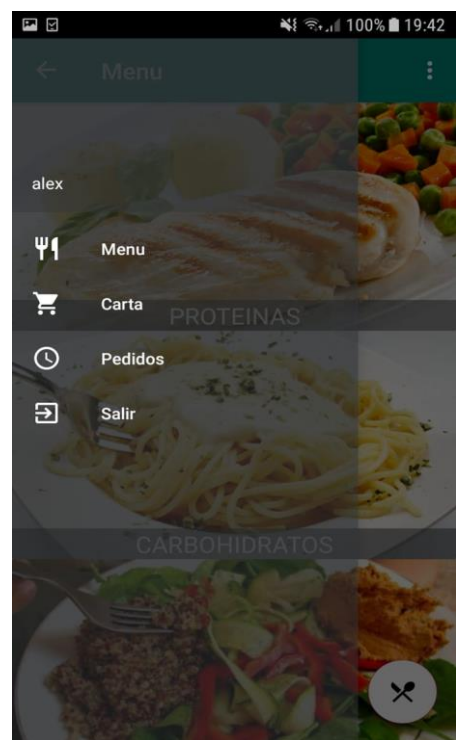
        //Logout
        Intent signIn = new Intent(Home.this, SignIn.class);
        signIn.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(signIn);
    }

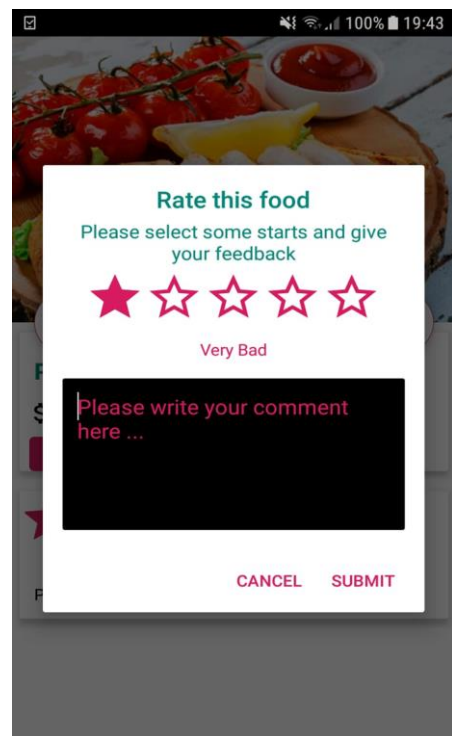
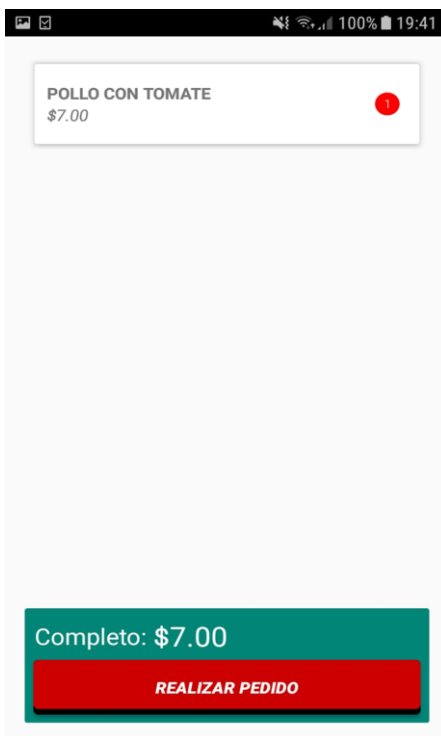
    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    drawer.closeDrawer(GravityCompat.START);
    return true; }

```

2.5.Capturas de la Aplicación Cliente







Desarrollo Aplicación Servidor

Clase Common.java

```
public class Common {
    public static User currentUser;
    public static Request currentRequest;

    public static final String UPDATE = "Update";
    public static final String DELETE = "Delete";

    public static final int PICK_IMAGE_REQUEST = 71;

    public static final String baseUrl = "https://maps.googleapis.com";

    public static final String fcmUrl = "https://fcm.googleapis.com";

    public static String convertCodeToStatus(String code){
        if(code.equals("0")){
            return "Placed";
        } else if (code.equals("1")){
            return "On my way";
        } else {
            return "Shipped";
        }
    }
}

public static APIService getFCMClient(){
    return FCMRetrofitClient.getClient(fcmUrl).create(APIService.class);
}

public static IGeoCoordinates getGeoCodeService(){
```

```

        return RetrofitClient.getClient(baseUrl).create(IGeoCoordinates.class);
    }

    public static Bitmap scaleBitmap(Bitmap bitmap, int newWidth, int newHeight){
        Bitmap        scaledBitmap        =        Bitmap.createBitmap(newWidth,        newHeight,
        Bitmap.Config.ARGB_8888);

        float scaleX = newWidth/(float)bitmap.getWidth();
        float scaleY = newHeight/(float)bitmap.getHeight();
        float pivotX=0, pivotY=0;

        Matrix scaleMatrix = new Matrix();
        scaleMatrix.setScale(scaleX, scaleY, pivotX, pivotY);

        Canvas canvas = new Canvas(scaledBitmap);
        canvas.setMatrix(scaleMatrix);
        canvas.drawBitmap(bitmap, 0, 0, new Paint(Paint.FILTER_BITMAP_FLAG));

        return scaledBitmap;
    }

```

Clase DirectionJSONParser.java

```

public class DirectionJSONParser {

    /**
     * Receives a JSONObject and returns a list of lists containing latitude and longitude
     */
    public List<List<HashMap<String, String>>> parse(JSONObject jObject) {

        List<List<HashMap<String, String>>> routes = new
        ArrayList<List<HashMap<String, String>>>();
        JSONArray jRoutes = null;
        JSONArray jLegs = null;
        JSONArray jSteps = null;

        try {

            jRoutes = jObject.getJSONArray("routes");

            /** Traversing all routes */
            for (int i = 0; i < jRoutes.length(); i++) {
                jLegs = ((JSONObject) jRoutes.get(i)).getJSONArray("legs");
                List path = new ArrayList<HashMap<String, String>>();

                /** Traversing all legs */
                for (int j = 0; j < jLegs.length(); j++) {
                    jSteps = ((JSONObject) jLegs.get(j)).getJSONArray("steps");

```

```

        /** Traversing all steps */
        for (int k = 0; k < jSteps.length(); k++) {
            String polyline = "";
            polyline = (String) ((JSONObject) ((JSONObject)
jSteps.get(k)).get("polyline")).get("points");
            List list = decodePoly(polyline);

            /** Traversing all points */
            for (int l = 0; l < list.size(); l++) {
                HashMap<String, String> hm = new HashMap<String, String>();
                hm.put("lat", Double.toString(((LatLng) list.get(l)).latitude));
                hm.put("lng", Double.toString(((LatLng) list.get(l)).longitude));
                path.add(hm);
            }
            routes.add(path);
        }
    }

    } catch (JSONException e) {
        e.printStackTrace();
    } catch (Exception e) {
    }

    return routes;
}

/**
 * Method to decode polyline points
 * Courtesy : http://jeffreysambells.com/2010/05/27/decoding-polylines-from-google-maps-direction-api-with-java
 */
private List decodePoly(String encoded) {
    List poly = new ArrayList();
    int index = 0, len = encoded.length();
    int lat = 0, lng = 0;

    while (index < len) {
        int b, shift = 0, result = 0;
        do {
            b = encoded.charAt(index++) - 63;
            result |= (b & 0x1f) << shift;
            shift += 5;
        } while (b >= 0x20);
        int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
        lat += dlat;

```

```

    shift = 0;
    result = 0;
    do {
        b = encoded.charAt(index++) - 63;
        result |= (b & 0x1f) << shift;
        shift += 5;
    } while (b >= 0x20);
    int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
    lng += dlng;

    LatLng p = new LatLng((((double) lat / 1E5)),
        (((double) lng / 1E5)));
    poly.add(p);
}

return poly;
}

```

Clase FCMRetrofitClient.java

```

public class FCMRetrofitClient {
    private static Retrofit retrofit = null;

    public static Retrofit getClient(String baseUrl){
        if(retrofit == null){
            retrofit = new Retrofit.Builder()
                .baseUrl(baseUrl)
                .addConverterFactory(GsonConverterFactory.create())
                .build();
        }
        return retrofit;
    }
}

```

Clase IGeoCoordinates.java

```

public interface IGeoCoordinates {
    @GET ("maps/api/geocode/json")
    Call<String> getGeoCode(@Query("address")String address);

    @GET ("maps/api/directions/json")
    Call<String> getDirections(@Query("origin")String origin, @Query("destination") String
    destination);
}

```

Clase TrackingOrder.java

```
public class TrackingOrder extends FragmentActivity implements OnMapReadyCallback,
    GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener,
    LocationListener {

    private GoogleMap mMap;

    private final static int PLAY_SERVICES_RESOLUTION_REQUEST=1000;
    private final static int LOCATION_PERMISSION_REQUEST=1001;

    private Location mLastLocation;

    private GoogleApiClient mGoogleApiClient;
    private LocationRequest mLocationRequest;

    private static int UPDATE_INTERVAL=1000;
    private static int FATEST_INTERVAL=5000;
    private static int DISPLACEMENT=10;

    private IGeoCoordinates mService;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_tracking_order);

        mService = Common.getGeoCodeService();

        if(ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
            &&
            ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED){
            requestRuntimePermission();
        }else{
            if(checkPlayServices()){
                buildGoogleApiClient();
                createLocationRequest();
            }
        }

        displayLocation();

        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
        .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }
}
```



```

private void displayLocation() {
    if(ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
        &&
        ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED){
        requestRuntimePermission();
    }else{
        mLastLocation =
LocationServices.FusedLocationApi.getLastLocation(mGoogleApiClient);
        if(mLastLocation != null){
            double latitude = mLastLocation.getLatitude();
            double longitude = mLastLocation.getLongitude();

            //Add Marker in your location amd move the camera
            LatLng yourLocation = new LatLng(latitude, longitude);
            mMap.addMarker(new MarkerOptions().position(yourLocation).title("Tu ubicación"));
            mMap.moveCamera(CameraUpdateFactory.newLatLng(yourLocation));
            mMap.animateCamera(CameraUpdateFactory.zoomTo(17.0f));

            //After add Marker for you location, Add Marker for this Order and draw route
            drawRoute(yourLocation, Common.currentRequest.getAddress());
        }else{
            Toast.makeText(this, "No se puede obtener la ubicación",
Toast.LENGTH_SHORT).show();
            //Log.d("DEBUG", "No se puede obtener la ubicación");
        }
    }
}

private void drawRoute(final LatLng yourLocation, String address) {
    mService.getGeoCode(address).enqueue(new Callback<String>() {
        @Override
        public void onResponse(Call<String> call, Response<String> response) {
            try{
                JSONObject jsonObject = new JSONObject(response.body().toString());

                String lat = ((JSONArray)jsonObject.get("results"))
                    .getJSONObject(0)
                    .getJSONObject("geometry")
                    .getJSONObject("location")
                    .get("lat").toString();

                String lng = ((JSONArray)jsonObject.get("results"))
                    .getJSONObject(0)
                    .getJSONObject("geometry")
                    .getJSONObject("location")
                    .get("lng").toString();

                LatLng orderLocation = new LatLng(Double.parseDouble(lat),
Double.parseDouble(lng));

```

```

        Bitmap bitmap = BitmapFactory.decodeResource(getResources(),
R.drawable.box);
        bitmap = Common.scaleBitmap(bitmap, 70, 70);

        MarkerOptions marker = new
MarkerOptions().icon(BitmapDescriptorFactory.fromBitmap(bitmap))
                .title("Order of "+Common.currentRequest.getPhone())
                .position(orderLocation);
        mMap.addMarker(marker);

        //draw woute
        mService.getDirections(yourLocation.latitude+","+yourLocation.longitude,
                orderLocation.latitude+","+orderLocation.longitude)
                .enqueue(new Callback<String>() {
                    @Override
                    public void onResponse(Call<String> call, Response<String> response) {
                        new ParserTask().execute(response.body().toString());
                    }

                    @Override
                    public void onFailure(Call<String> call, Throwable t) {

                    }
                });

    }catch (JSONException e){
        e.printStackTrace();
    }
}

@Override
public void onFailure(Call<String> call, Throwable t) {

}

});
}

private void createLocationRequest() {
    mLocationRequest = new LocationRequest();
    mLocationRequest.setInterval(UPDATE_INTERVAL);
    mLocationRequest.setFastestInterval(FATEST_INTERVAL);
    mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
    mLocationRequest.setSmallestDisplacement(DISPLACEMENT);
}

protected synchronized void buildGoogleApiClient() {
    mGoogleApiClient = new GoogleApiClient.Builder(this)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .addApi(LocationServices.API).build();
    mGoogleApiClient.connect();
}

```

```

private boolean checkPlayServices() {
    int resultCode = GooglePlayServicesUtil.isGooglePlayServicesAvailable(this);
    if(resultCode != ConnectionResult.SUCCESS){
        if(GooglePlayServicesUtil.isUserRecoverableError(resultCode)){
            GooglePlayServicesUtil.getErrorDialog(resultCode, this,
PLAY_SERVICES_RESOLUTION_REQUEST).show();
        }else{
            Toast.makeText(this, "Este dispositivo no es compatible",
Toast.LENGTH_SHORT).show();
            finish();
        }
        return false;
    }
    return true;
}

private void requestRuntimePermission() {
    ActivityCompat.requestPermissions(this, new String[]{
        Manifest.permission.ACCESS_COARSE_LOCATION,
        Manifest.permission.ACCESS_FINE_LOCATION
    }, LOCATION_PERMISSION_REQUEST);
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    switch (requestCode){
        case LOCATION_PERMISSION_REQUEST:
            if(grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED){
                if(checkPlayServices()){
                    buildGoogleApiClient();
                    createLocationRequest();

                    displayLocation();
                }
            }
            break;
    }
}

@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
}

@Override
public void onLocationChanged(Location location) {
    mLastLocation = location;
    displayLocation();
}

```

```

    }

    @Override
    protected void onResume() {
        super.onResume();
        checkPlayServices();
    }

    @Override
    protected void onStart() {
        super.onStart();
        if(mGoogleApiClient != null)
            mGoogleApiClient.connect();
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        displayLocation();
        startLocationUpdates();
    }

    private void startLocationUpdates() {
        if(ActivityCompat.checkSelfPermission(this,
        android.Manifest.permission.ACCESS_FINE_LOCATION) !=
        PackageManager.PERMISSION_GRANTED
        && ActivityCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_COARSE_LOCATION) !=
        PackageManager.PERMISSION_GRANTED){
            return;
        }
        LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
        mLocationRequest, this);
    }

    @Override
    public void onConnectionSuspended(int i) {
        mGoogleApiClient.connect();
    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

    }

    private class ParserTask extends AsyncTask<String, Integer, List<List<HashMap<String,
    String>>>> {
        ProgressDialog mDialog = new ProgressDialog(TrackingOrder.this);

        @Override
        protected void onPreExecute() {
            mDialog.setMessage("Espera por favor");
            mDialog.show();

```

```

    }

    @Override
    protected List<List<HashMap<String, String>>> doInBackground(String... strings) {
        JSONObject jsonObject;
        List<List<HashMap<String, String>>> routes = null;
        try{
            jsonObject = new JSONObject(strings[0]);
            DirectionJSONParser parser = new DirectionJSONParser();

            routes = parser.parse(jsonObject);
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return routes;
    }

    @Override
    protected void onPostExecute(List<List<HashMap<String, String>>> lists) {
        mDialog.dismiss();

        ArrayList points = null;
        PolylineOptions lineOptions = null;

        for (int i=0;i<lists.size();i++){
            points = new ArrayList();
            lineOptions = new PolylineOptions();

            List<HashMap<String, String>> path = lists.get(i);

            for (int j=0; j<lists.size();j++){

                HashMap<String, String> point = path.get(j);

                double lat = Double.parseDouble(point.get("lat"));
                double lng = Double.parseDouble(point.get("lng"));

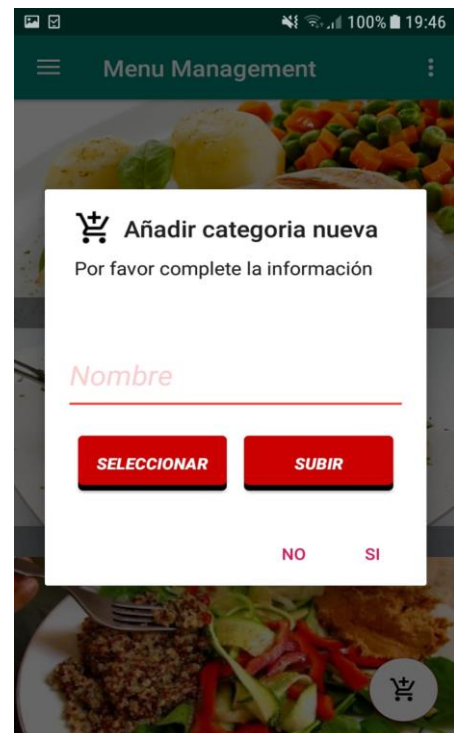
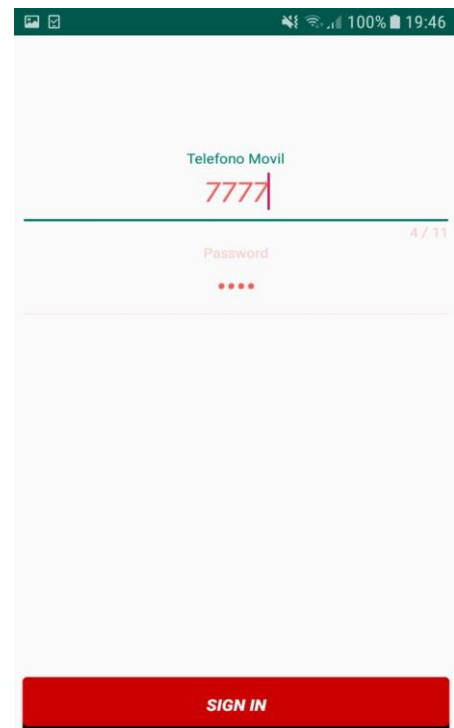
                LatLng position = new LatLng(lat, lng);


                points.add(position);
            }

            lineOptions.addAll(points);
            lineOptions.width(12);
            lineOptions.color(Color.BLUE);
            lineOptions.geodesic(true);
        }
        mMap.addPolyline(lineOptions);
    }
}

```

2.6. Capturas Aplicación Servidor



 **Añadir comida nueva**
Por favor complete la información

Nombre

Descripcion

Precio

Descuento

SELECCIONAR **SUBIR**

NO SI

1543755721891
PLACED
2111
ADDRESS

EDITAR **ELIMINAR** **DETALLES** **DIRECCIÓN**

1543776493681
PLACED
2111
SANTA MONICA 5

EDITAR **ELIMINAR** **DETALLES** **DIRECCIÓN**

1543776648541
PLACED
2111
CALLE ZANKOETA 5, 2B

EDITAR **ELIMINAR** **DETALLES** **DIRECCIÓN**

1543755721891
PLACED
2111
ADDRESS

EDITAR **ELIMINAR** **DETALLES** **DIRECCIÓN**

Subir pedido
Por favor eliga el estado

Placed
On my way
Shipped

CALLE ZANKOETA 5, 2B

EDITAR **ELIMINAR** **DETALLES** **DIRECCIÓN**

1543755721891
2111
\$5.00
ADDRESS
COMMENT

Detail

NAME : TAMAL
QUANTITY : 1
PRICE : 4
DISCOUNT : 0

NAME : STUFFED POTATOES
QUANTITY : 1
PRICE : 1
DISCOUNT : 0

3. CONCLUSIONES

Este proyecto ha sido de mucha práctica para mejorar nuestra destreza a la hora de desarrollar una aplicación y pensar programáticamente con lógica por dónde empezar un proyecto, hemos tenido dificultades ya que se trataba de una aplicación medianamente complicada pero eso no nos ha vencido a la hora de seguir, hemos aprendido mucho y aún lo seguimos haciendo.

3.1. Innovación

La innovación de este proyecto ha sido la creación de nuestra aplicación Servidor donde podremos gestionar de manera más práctica y rápida la aplicación Cliente. Hemos añadido muchas funcionalidades como agregar categorías, agregar platos... explico más en detalle en el punto ([2.1 Ver Anexo](#))

3.2. Trabajo futuro

En un trabajo futuro tenemos en mente que esta aplicación sea conocida, este en muchos países, añadir nuevas funcionalidades, compatibilidad con otros dispositivos como IOS, posiblemente una página web y un almacenamiento de base de datos con mayor capacidad, esperemos que esto sea para el año 2019.

4. Biblioteca y Webgrafía

<https://es.stackoverflow.com/questions/66409/c%C3%B3mo-marcar-ruta-entre-dos-puntos-con-api-de-google-maps>

<https://developers.google.com/maps/documentation/android-sdk/intents>

<https://material.io/tools/color/#!/?view.left=0&view.right=0&primary.color=4CAF50&secondary.color=FDD835>

<https://developers.google.com/android/guides/http-auth>

<https://github.com/rengwuxian/MaterialEditText>

<https://unsplash.com/>

<https://stripe.com/docs/mobile/android/google-pay>

<https://github.com/florent37/MaterialTextField>

<https://stackoverflow.com/questions/15516595/directions-api-on-android>

<https://glovoapp.com/es/mad/category/RESTAURANT/healthy>

<https://www.ubereats.com/en-AU/stores/>