



RESTAPP

Alumna : Ana Maria Pascual Galiana

Tutora: Alicia González Canet

CFGS: Desarrollo de Aplicaciones Multiplataforma

INDICE

Introducción	3
Objetivos	4
Tecnologías usadas	5
Android Studio	6
Google maps	7
Canva	8
Firebase	9
Base de datos	10
Autenticación	12
Localización	14
Permisos de ubicación	16
Estructura de la aplicación	18
Código	20
Mejoras	46
Conclusiones	47
Bibliografía	48

Introducción

La finalidad de este trabajo de fin de ciclo es el servicio de comida a domicilio a través de la creación de una aplicación para smartphones y tablets.

Para su implementación tenemos la empresa “x”. Esta organización se encuentra situada en la ciudad de Gandía y esta especializada en el sector hostelero desde el año 2021. Concretamente se dedica a la producción y venta de comida a través de un restaurante situado en la ciudad anteriormente mencionada, además cuenta con una aplicación, en la cual nos vamos a centrar en este proyecto.

(Evidencia en estudios, datos)

Por otra parte y debido a la aparición del Covid19 se ha percibido una variación en los intereses y las preocupaciones de la sociedad, ya que cada vez es más notoria la preferencia o necesidad de pasar más tiempo de ocio en el hogar con las personas pertenecientes a los círculos más íntimos. Por este motivo las empresas se preguntan cómo reorganizar sus funciones y que pueden hacer para adaptarse a las exigencias del entorno y la sociedad.

Porque Android ?

Es un sistema operativo de código abierto basado en el kernel de Linux, significa que cualquier persona es libre de mejorarlo, cambiarlo y adaptarlo, lo que hace de Android un sistema operativo para dispositivos móviles extendido por lo tanto significa que diferentes fabricantes pueden integrarlo en sus aparatos sin necesidad de pagar licencia, lo que hace de ello que el número de dispositivos móviles Android sea muy alto.

Objetivos

OBJETIVO GENERAL

El objetivo vital de la aplicación es ofrecer un servicio de comida a domicilio a través de una aplicación.

OBJETIVOS ESPECÍFICOS

- Elaborar una aplicación como solución a las necesidades sociales actuales.
- Describir el proceso de desarrollo de dicha aplicación.
- Elaborar una pequeña base de datos en tiempo real donde encontrar la carta actualizada a diario.
- Recibir y analizar las características de los pedidos para poder ser tramitados momentáneamente.

Tecnologías utilizadas

Las tecnologías que he se han utilizado han sido las siguientes:

Para el desarrollo de la aplicación :

- Android Studio
- Firebase
- Google Maps

Para la parte de diseño:

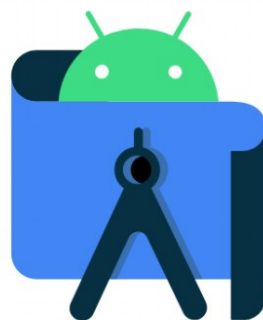
- Canva

Android Studio

Android Studio es el entorno de desarrollo integrado oficial para el desarrollo de apps para Android . Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece incluso más funciones que aumentan tu productividad cuando desarrollas apps para Android, como las siguientes:

- Un sistema de compilación flexible basado en Gradle
- Un emulador rápido y cargado de funciones
- Un entorno unificado donde puedes desarrollar para todos los dispositivos Android
- Aplicación de cambios para insertar cambios de código y recursos a la app en ejecución sin reiniciarla
- Integración con GitHub y plantillas de código para ayudarte a compilar funciones de apps comunes y también importar código de muestra
- Variedad de marcos de trabajo y herramientas de prueba
- Herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de versiones, entre otros

android studio



Google Maps es el nombre de una aplicación desarrollada por Google que se encarga de ofrecer a los usuarios toda la información que necesiten sobre su ubicación actual, como también la de cualquier dirección específica, así como el trazado de recorridos para llegar al lugar que estos deseen desde donde se encuentran.

El uso de Google maps ha sido la implementación de una vista del mapa focalizado en la ubicación actual.



Google Maps

Canva es una web de diseño gráfico y composición de imágenes para la comunicación fundada en 2012, y que ofrece herramientas online para crear tus propios diseños, tanto si son para ocio como si son profesionales. Su método es el de ofrecer un servicio freemium, que puedes utilizar de forma gratuita, pero con la alternativa de pagar para obtener opciones avanzadas.

Sirve tanto para diseñadores aficionados como para los más experimentados, incluyendo su propio banco de imágenes y una serie de herramientas variadas. Si eres un diseñador experimentado podrás obtener muy buenos resultados de forma rápida y sencilla, y si eres un aficionado no necesitarás conocimientos para obtener resultados decentes.



Es una plataforma digital que se utiliza para facilitar el desarrollo de aplicaciones web o móviles de una forma efectiva, rápida y sencilla, la cual es utilizada por sus diversas funciones como una técnica de Marketing Digital para aumentar la base de usuarios y generar mayores beneficios económicos.

Su principal objetivo, es mejorar el rendimiento de las apps mediante la implementación de diversas funcionalidades que van a hacer de la aplicación en cuestión, mucho más manejable, segura y de fácil acceso para los usuarios.

Con Firebase, podemos:

- lograr una integración dinámica de los usuarios usando Firebase Authentication;
- lograr que nuestras aplicaciones sean visualizadas y utilizadas utilizando la herramienta de compartir o Dynamic Links;
- tener nuestra base de datos alojada en Realtime Database
- crear análisis de resultados con Analytics;



Base de datos

Realtime database como base de datos de la aplicación.

Es una base de datos sencilla donde solamente se almacena la carta, el menú y las comandas realizadas.

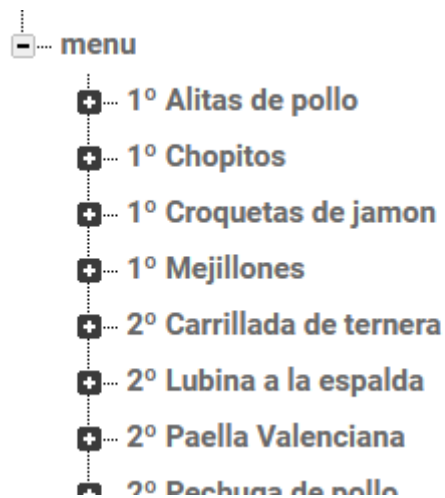


El formato almacenado de la carta seria este :



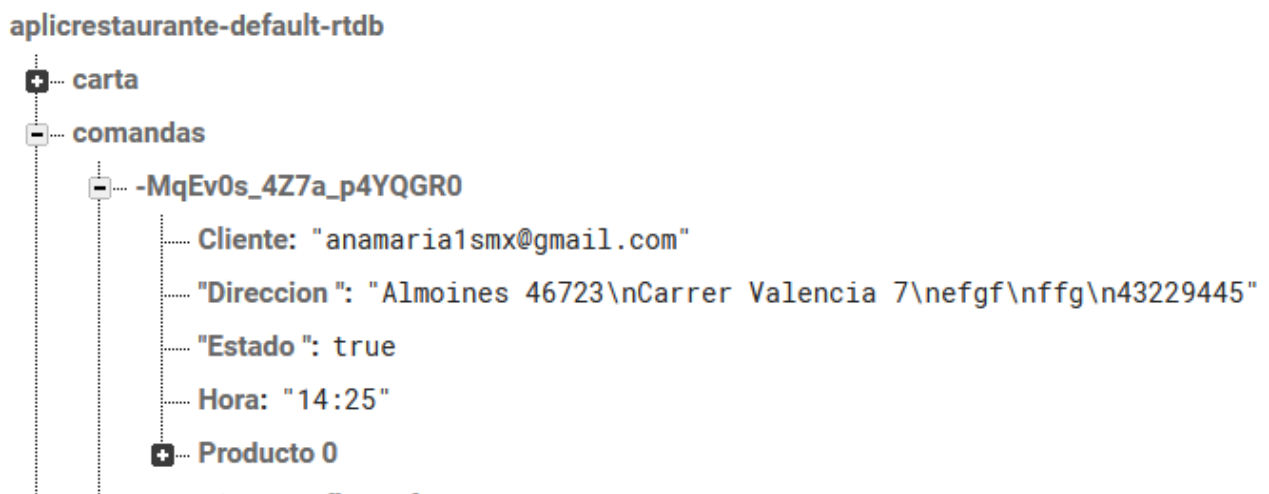
Como se puede apreciar a través de la imagen las fotos de la carta están alojadas en firebase storage y se carga a traver de la base de datos mediante una url.

El formato del menú:



Es una base de datos sencilla sin entidad relación ya que como se puede apreciar era innecesaria.

Formato comanda:



A la base de datos cuando se sube la comanda, lleva una serie de datos como que cliente ha echo el pedido, la dirección, el estado, la hora y el listado con las unidades y precio de los productos.

Autenticación

Firebase authentication, es el modo que he utilizado para todo el sistema de login con correo y contraseña.

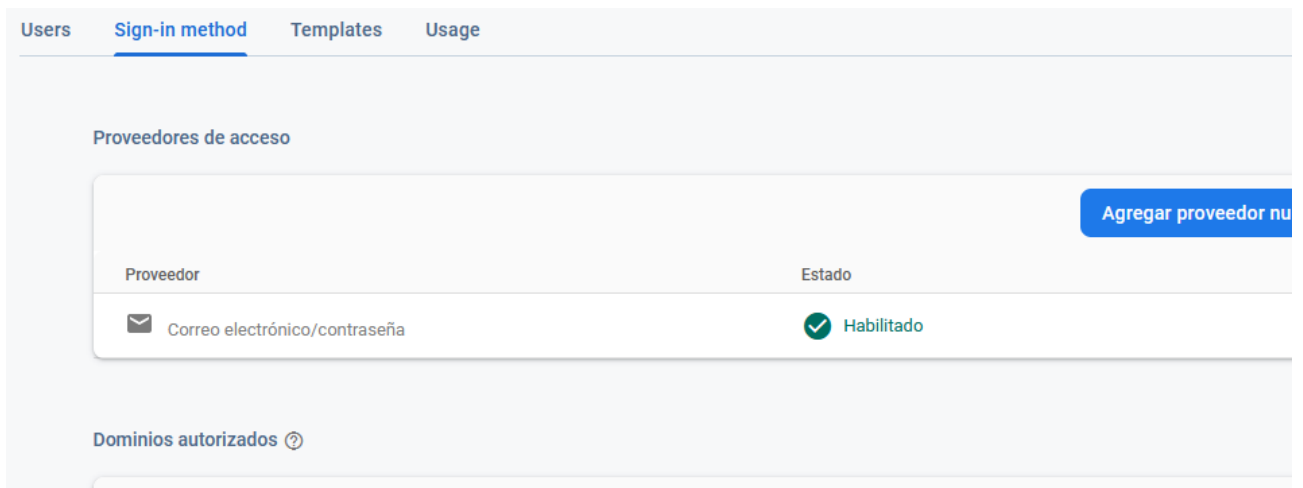


The screenshot shows the 'Users' tab in the Firebase Authentication console. At the top, there are tabs for 'Users', 'Sign-in method', 'Templates', and 'Usage'. Below the tabs, there is a search bar with the placeholder text 'Buscar por dirección de correo electrónico, número de teléfono o UID de usuario' and a blue button labeled 'Agregar usuario'. Below the search bar is a table with the following columns: 'Identificador', 'Proveedores', 'Fecha de creación', 'Fecha de acceso', and 'UID de usuario'. The table contains four rows of user data.

Identificador	Proveedores	Fecha de creación	Fecha de acceso	UID de usuario
legennd87spain@gmail.com	✉	2 dic. 2021	2 dic. 2021	RQhx6oT3HsSr73LMWgL6YeBNo...
saraypascualferlin@gmail.com	✉	19 nov. 2021	19 nov. 2021	aWhCmTuLU9UDmZFthvNW2RnV...
anamaria1dam@gmail.com	✉	19 nov. 2021	19 nov. 2021	mMa31un5qFNGP1DIW0ZxAsJ6D...
izanpfpro@gmail.com	✉	12 nov. 2021	12 nov. 2021	FaoA4nPx3zMejGkhgODISsPVsEP2

Firebase authentication nos ofrece una vista desde la consola con el registro de creación, correo y un id unico de cada usuario registrado en la aplicación.

Tambien contamos con la pestaña Método de acceso.



The screenshot shows the 'Sign-in method' tab in the Firebase Authentication console. At the top, there are tabs for 'Users', 'Sign-in method', 'Templates', and 'Usage'. Below the tabs, there is a section titled 'Proveedores de acceso'. Below this section is a table with the following columns: 'Proveedor' and 'Estado'. The table contains one row of data.

Proveedor	Estado
✉ Correo electrónico/contraseña	✓ Habilitado

Below the table, there is a section titled 'Dominios autorizados' with a help icon.

Como se aprecia en la imagen tengo habilitado el método de acceso con correo electrónico y contraseña. La plataforma cuenta con varios métodos de acceso, desde Facebook hasta el teléfono móvil entre ellos.

Ademas nos ofrece unas plantillas para que cuando un cliente se registra o a olvidado su contraseña mediante los formularios de registro o olvide mi contraseña de la aplicación, reciba un correo electrónico para poder verificar el correo o cambiar su contraseña.

[Users](#) [Sign-in method](#) [Templates](#) [Usage](#)

Plantillas

Correo electrónico ⓘ

✉ Verificación de dirección de correo...

✉ Restablecer contraseña

✉ Cambio de dirección de correo ele...

⚙ Configuración del SMTP

SMS

📱 Verificación por SMS

ⓘ Para cambiar el idioma en que se muestra esta plantilla, restablece su contenido a los valores predeterminados

Restablecer

Verificación de dirección de correo electrónico

Cuando un usuario se registra con una dirección de correo electrónico y una contraseña, puedes enviarle un mensaje de confirmación para verificar la dirección de correo electrónico que registró. [Más información](#) ⓘ

Nombre del remitente

De

no proporcionado

noreply@aplicrestaurante.firebaseio.com

✎

Responder a

Localización

Google maps contiene métodos a los que poder llamar para obtener la localización exacta o en su defecto lo mas precisa posible

Para poder acceder a la localización en la aplicación el usuario debe conceder permisos de ubicación a través del dialogo que salta cada vez que se inicia la aplicación y esta no tiene dichos permisos. Ademas es necesario el consentimiento de la ubicación para poder utilizar la aplicación.

A continuación muestro la parte del código donde cargo el mapa y refresco la ventana para que una vez concedido el permiso de ubicación el mapa se recargue con la ubicación instantáneamente:

```
SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
    .findFragmentById(R.id.map);
mapFragment.getMapAsync( onMapReadyCallback: this);
flpc = LocationServices.getFusedLocationProviderClient( activity: this);

if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION, Manifest.permission.ACCESS_COARSE_LOCATION}, 1);
    super.onRestoreInstanceState(savedInstanceState);
}
```

Esta es la función donde recojo la ubicación:

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

    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED && ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 1) != null) {
        // Requesting location permission
    }

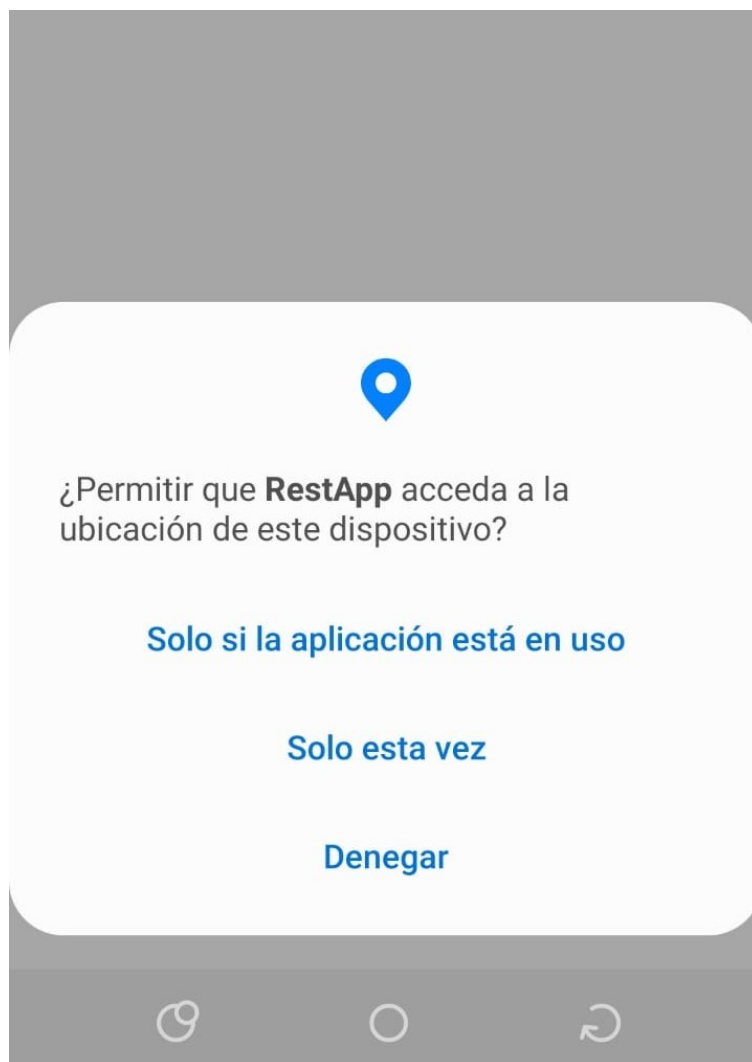
    flpc.getLastLocation()
        .addOnSuccessListener( activity: this, new OnSuccessListener<Location>() {
            @Override
            public void onSuccess(Location location) {
                // Got last known location. In some rare situations this can be null.
                if (location != null) {
                    // Add a marker in Sydney and move the camera
                    LatLng ubica = new LatLng(location.getLatitude(), location.getLongitude());
                    mMap.addMarker(new MarkerOptions().position(ubica).title("Mi ubicación"));
                    mMap.moveCamera(CameraUpdateFactory.newLatLng(ubica));
                    mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(ubica, 18));
                }
            }
        });
}
```

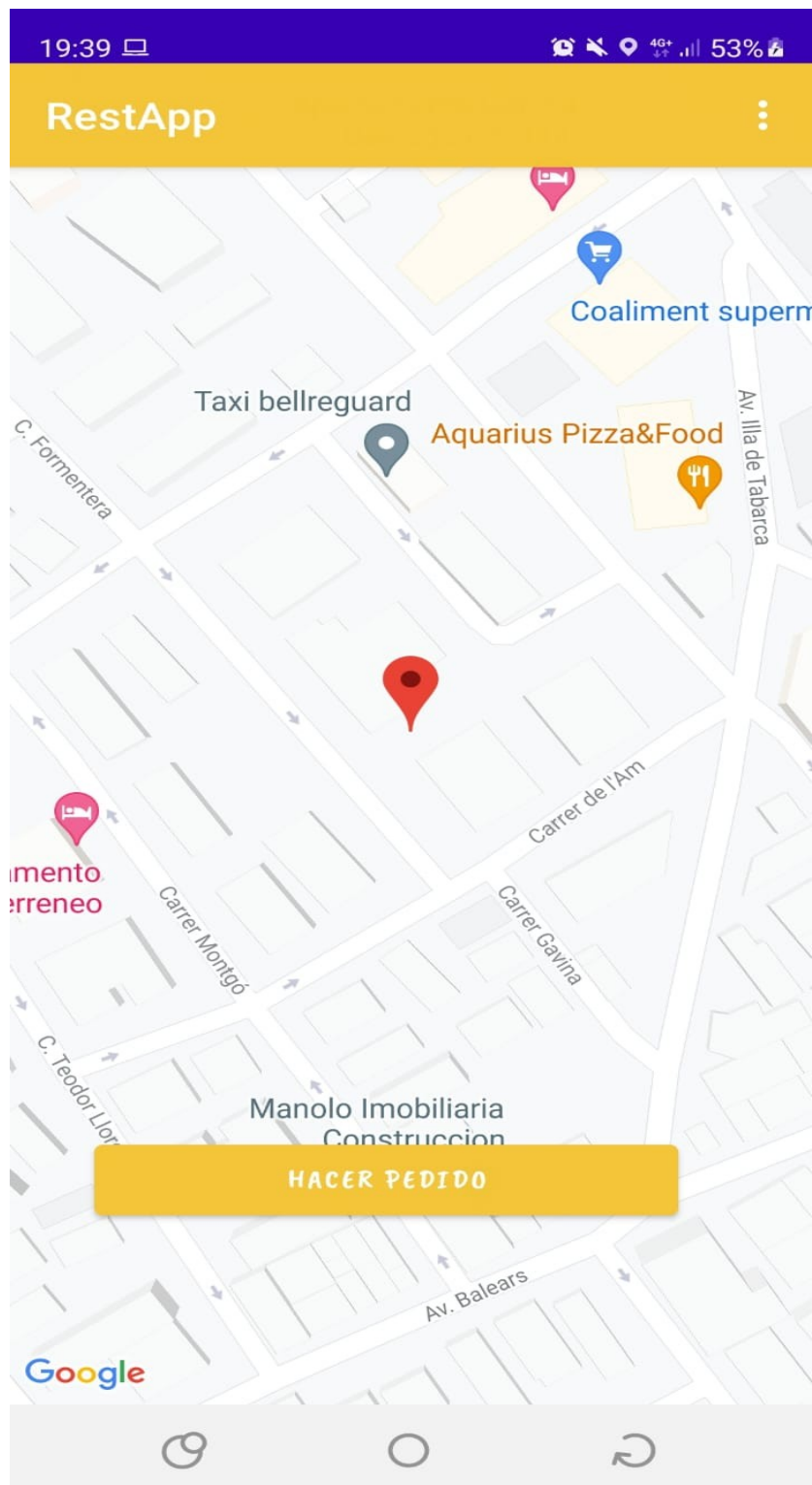
Antes de recoger la ubicación dentro de la función tengo incluido el dialogo para conceder permisos.

Permisos de ubicación

El método para conceder permisos de ubicación en una aplicación ya es existente en Android Studio por decirlo de algún modo.

A continuación muestro el dialogo para conceder los permisos y la visualización de que después de habérselos concedido nos muestra el mapa con nuestra ubicación.



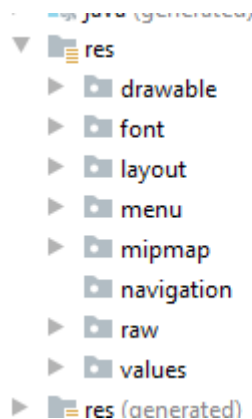


Nos muestra el mapa con con la ubicación marcada y un botón para navegar por la aplicación.

Estructura de la aplicación

La dividiremos en 2 partes en esta se explicara la estructura de directorios como se dividen los ficheros del back con los de fronted, como se ha construido toda la parte visual y por otra parte en 'código' mostraremos lo que pertenece a la parte del back, la parte lógica.

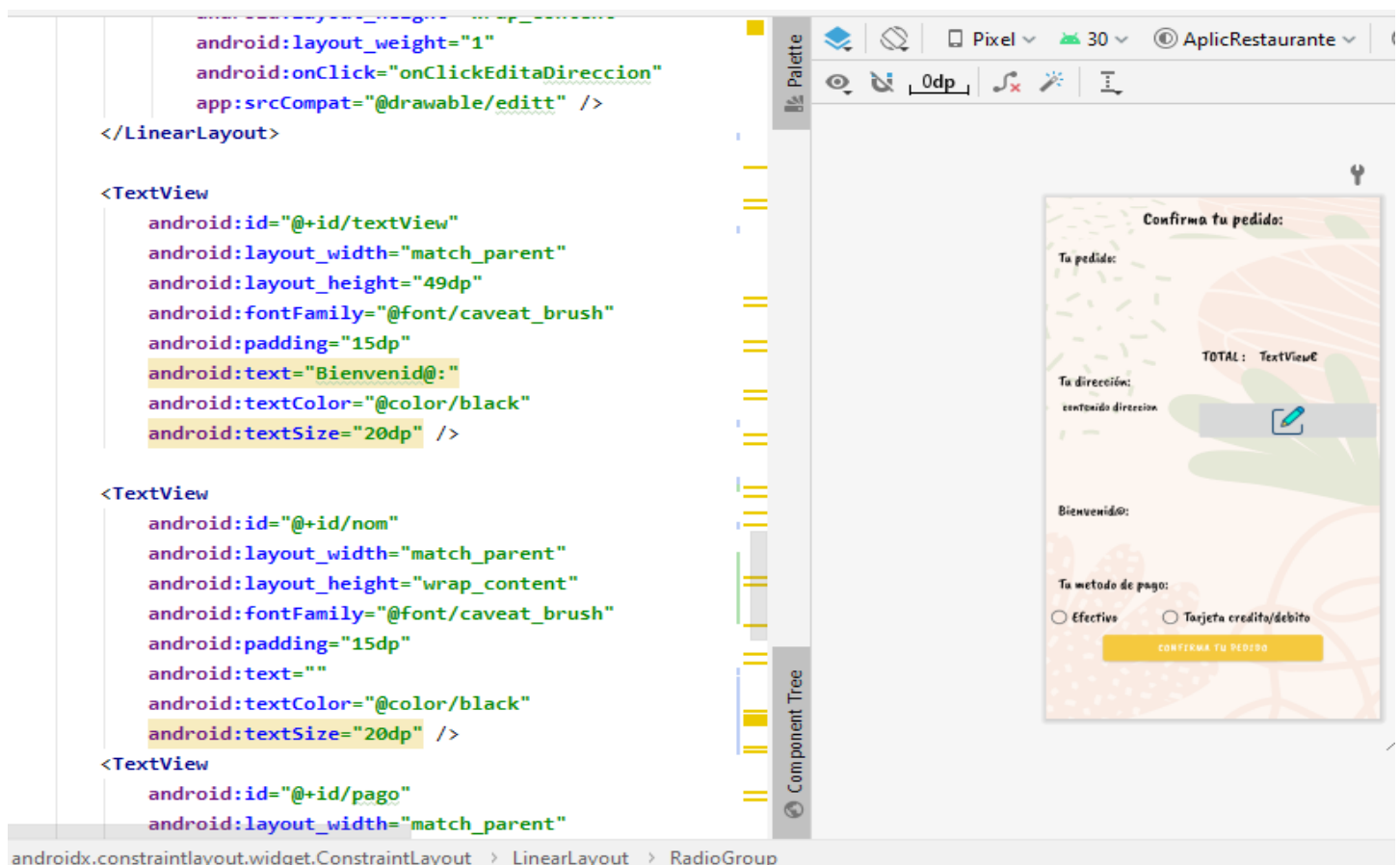
Esta es la estructura de directorios que formarían la parte visual.



Estos son los directorios que yo he utilizado para mi aplicación:

- Directorio drawable es el que contiene todas las imágenes alojadas en la aplicación excepto las de la carta que están alojadas en firebase storage.
- Directorio font es el que contiene todos los formatos de letra descargados en la aplicación, en mi caso utilicé uno descargado.
- Directorio layout es el directorio principal de la parte visual de la aplicación pues es el que contiene todas las vistas de la aplicación, donde se alojan todos los componentes en cada vista.
- Directorio menú es el que contiene la vista del menú desplegable que hay arriba a la derecha de la aplicación.
- Directorio values es el que contiene los ficheros de colores, textos, certificados de fuentes o de la api de google mas por ejemplo.

Este seria un ejemplo de como se construye la parte visual de una pagina:



Los enlaces a los componentes en la parte del código se consiguen gracias a sus 'id' en el caso de los botones se pone el nombre de la función a la que hace referencia la funcionalidad de dicho botón.

```

<ImageButton
    android:id="@+id/imageButton"
    style="@style/Widget.AppCompat.ImageButton"
    android:layout_width="176dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:onClick="onClickEditaDireccion"
    app:srcCompat="@drawable/editt" />

</LinearLayout>

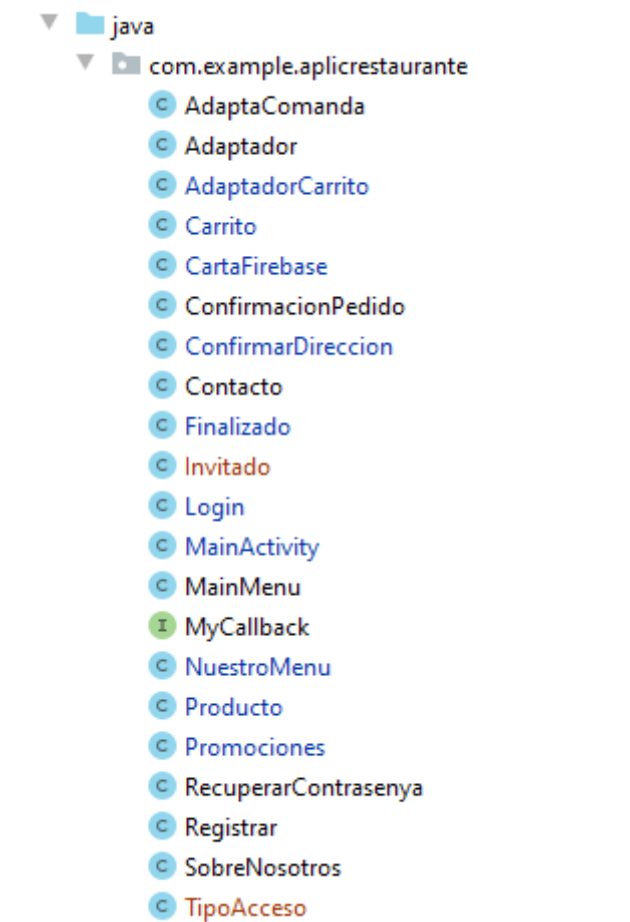
<TextView
    android:id="@+id/nom"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:fontFamily="@font/caveat_brush"
    android:padding="15dp"
    android:text=""
    android:textColor="@color/black"
    android:textSize="20dp" />

<TextView
    android:id="@+id/pago"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:fontFamily="@font/caveat_brush"
    android:padding="15dp"
    android:text="Tu metodo de pago:"
    
```

Código

Aquí vamos a mostrar y explicar toda la parte del back, la lógica que le da funcionalidad a la parte del fronted.

Este sería todo el conjunto de ficheros que componen toda la parte de código de la aplicación:



Voy a ir en orden de ejecución de la aplicación.

Primero empezare por la parte común de todos los ficheros que componen las ventanas de la aplicación.

Todos contienen el método onCreate, la clase que extiende de MainMenu y el enlace de contenido de vista con su layout.

```
public class ConfirmarDireccion extends MainMenu {
    private FusedLocationProviderClient fl;
    EditText pobInp, calleInp, numInp, codeInp, info, tlf;
    String name, nombre;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_confirmar_direccion);
        fl = LocationServices.getFusedLocationProviderClient(this);
    }
}
```

MainMenu en realidad no es una ventana sino el menú desplegable de la parte derecha de arriba de la aplicación.

```
public class MainMenu extends AppCompatActivity {

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

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_inicio) {
            Intent intent = new Intent( packageContext: this, MainActivity.class);
            startActivity(intent);
        }

        if (id == R.id.action_sobreNosotros) {
            Intent intent = new Intent( packageContext: this, SobreNosotros.class);
            startActivity(intent);
        }

        if (id == R.id.action_contacto) {
            Intent intent = new Intent( packageContext: this, Contacto.class);
            startActivity(intent);
        }
    }
}
```

MainActivity es la ventana donde empieza nuestra aplicación.

Es donde esta alojado el código anteriormente mostrado cuando he explicado toda la parte de recoger la ubicación pero otra parte de código interesante que nos ofrece esta clase es la función que contiene para cuando pulsamos la flecha de retroceso de nuestro teléfono móvil salte un dialogo preguntando si realmente queremos salir de la aplicación.

```
@Override
public void onBackPressed() {
    new AlertDialog.Builder( context: this).setIcon(android.R.drawable.ic_dialog_alert).setTitle("Cerrar Aplicación")
        .setMessage("Seguro que quieres cerrar la aplicación?")
        .setPositiveButton( text: "SI", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(
                DialogInterface dialog, int which) {
                a.signOut();
                Intent intent = new Intent(Intent.ACTION_MAIN); intent.addCategory(Intent.CATEGORY_HOME);
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK); startActivity(intent);
                finish();
            }
        }).setNegativeButton( text: "NO", listener: null).show(); }
```

ConfirmarDirección es la clase que contiene todo el formulario respecto la dirección de entrega del pedido como del teléfono móvil si se quiere añadir.

Este seria el método onCreate que contiene con todos los enlaces a los componentes del formulario como la llamada a la función de cargar datos que nos carga el formulario con los datos básicos de nuestra ubicación por si ese es el lugar de entrega.

```
public class ConfirmarDireccion extends MainMenu {
    private FusedLocationProviderClient fl;
    EditText pobInp, calleInp, numInp, codeInp, info, tlf;
    String name, nombre;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_confirmar_direccion);
        fl = LocationServices.getFusedLocationProviderClient( activity: this);

        getSupportActionBar().setTitle("RestApp");
        String bb = "#E4F4C536";
        getSupportActionBar().setBackgroundDrawable(new ColorDrawable(Color.parseColor(bb)));

        pobInp = (EditText) findViewById(R.id.poblacioninput);
        calleInp = (EditText) findViewById(R.id.calleinput);
        numInp = (EditText) findViewById(R.id.numeroinput);
        codeInp = (EditText) findViewById(R.id.codigopostalinput);
        info = (EditText) findViewById(R.id.infoadicionalinput);
        tlf = (EditText) findViewById(R.id.telefonoinput);

        cargarDatos();
    }
}
```

La función cargar datos detallada :

```
public void cargarDatos() {

    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {

        return;
    }
    fl.getLastLocation().addOnSuccessListener( activity: this, new OnSuccessListener<Location>() {
        @Override
        public void onSuccess(Location location) {
            // Got last known location. In some rare situations this can be null.
            if (location != null) {
                Geocoder geocoder = new Geocoder( context: ConfirmarDireccion.this, Locale.getDefault());
                try {
                    List<Address> addresses = geocoder.getFromLocation(location.getLatitude(), location.getLongitude(), 1);
                    if (addresses != null) {
                        String localidad = addresses.get(0).getLocality();
                        String calle = addresses.get(0).getThoroughfare();
                        String numero = addresses.get(0).getFeatureName();
                        String codigoPostal = addresses.get(0).getPostalCode();
                        pobInp.setText(localidad);
                        calleInp.setText(calle);
                        numInp.setText(numero);
                        codeInp.setText(codigoPostal);
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    });
}
```

La clase geocoder es la encargada de ayudarnos a recoger los datos de la dirección, luego creamos el list de tipo Address al que le pasamos nuestra latitud y longitud para que podamos cargar dichos datos, una vez conseguido eso tan solo tenemos que guardar en variables de tipo string los datos que nos interesan y cambiar el texto de las partes del formulario implicadas.

Vamos a mostrar la función del botón que nos lleva a la carta o bien si hemos accedido a esa pantalla a través del botón de editar dirección de la pantalla de confirmación de pedido nos devuelva allí.

```
public void onClick(View view) {
    if (name.equals("Finalizado")) {
        Intent i = new Intent( packageContext: this, Finalizado.class);
        String direccion = pobInp.getText().toString()+" "+ codeInp.getText().toString() + "\n";
        i.putExtra( name: "dir", direccion);
        i.putExtra( name: "nombre", nombre);
        startActivity(i);
    } else {
        Intent intent = new Intent( packageContext: this, TipoAcceso.class);
        String direccion = pobInp.getText().toString()+" "+ codeInp.getText().toString() + "\n";
        intent.putExtra( name: "dir", direccion);
        startActivity(intent);
    }
}
```

TipoAcceso es una ventana bastante sencilla es la que contiene 2 botones donde decidimos si nos logeamos en la aplicación o entramos como invitados

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_tipo_acceso);
    invitado = findViewById(R.id.invitado);
    logear = findViewById(R.id.registrado);

    getSupportActionBar().setTitle("RestApp");
    String bb = "#E4F4C536";
    getSupportActionBar().setBackgroundDrawable(new ColorDrawable(Color.parseColor(bb)));

    cargarDireccion();
}

public void onClickSesion(View view) {
    Intent i = new Intent( packageContext: this, Login.class);
    i.putExtra( name: "dir", direccion);
    startActivity(i);
}

public void onClickInvitado(View view) {
    Intent i = new Intent( packageContext: this, Invitado.class);
    i.putExtra( name: "dir", direccion);
    startActivity(i);
}

public void cargarDireccion(){

    Bundle extras = getIntent().getExtras();
    direccion =extras.getString( key: "dir");
}
```

De ahí accedemos o bien a la clase Invitado o a la Clase login, de la de invitado no voy a entrar en detalle pues simplemente contiene un formulario donde introducir un nombre y al pulsar el botón nos lleva a la carta.

Voy a seguir cual seria el proceso de toda la parte destinada a el login.

Login esta clase contiene el formulario donde poner el correo y contraseña en caso de estar ya registrado el botón para logearnos que nos muestra un aviso de que nos hemos logeado si los datos son correctos y nos lleva a la carta.

A parte de ello hay 2 botones mas que nos llevaría 1 a la pantalla de registro y otro a la pantalla de recuperar la contraseña.

El método onCreate contiene la conexión a firebase, la instancia de la clase AwesomeValidation para un correcto formato de correo y contraseña. Y el enlace a los campos de edición y botones.

```
public class Login extends MainMenu {
    private FirebaseAuth mAuth;
    FirebaseUser user;
    AwesomeValidation awesomeValidation;
    Button inicio, registro, recuperar;
    EditText correo, contraseña;
    String direccion;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        getSupportActionBar().setTitle("RestApp");
        String bb = "#E4F4C536";
        getSupportActionBar().setBackgroundDrawable(new ColorDrawable(Color.parseColor(bb)));

        cargarDireccion();

        mAuth = FirebaseAuth.getInstance();
        awesomeValidation = new AwesomeValidation(ValidationStyle.BASIC);
        awesomeValidation.addValidation( activity: this, R.id.correofinput, Patterns.EMAIL_ADDRESS, "El email no es valido");
        awesomeValidation.addValidation( activity: this, R.id.contraseñainput, s: ".{6,}", "Debe contener almenos 6 digitos");

        inicio=findViewById(R.id.iniciarsesion);
        registro = findViewById(R.id.registrarse);
        recuperar = findViewById(R.id.olvide);
        correo = findViewById(R.id.correoelectronicoinput);
        contraseña = findViewById(R.id.contraseñainput);
    }
}
```

Android Studio Arctic Fox | 2020.1.2
Update...

Esta sería la función que valida los datos introducidos en el formulario, verifica que la cuenta existe en firebase authentication y si todo es correcto nos lleva a la carta.

```
public void onClickIniciar(View view) {
    if (correo.getText().length() != 0 && contraseña.getText().length() != 0) {
        if (awesomeValidation.validate()) {
            String mail = correo.getText().toString();
            String pass = contraseña.getText().toString();

            mAuth.signInWithEmailAndPassword(mail, pass).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()) {
                        Toast.makeText(context, Login.this, text: "Login correcto.", Toast.LENGTH_SHORT).show();
                        Intent intent = new Intent(packageContext, CartaFirebase.class);
                        intent.putExtra(name: "dir", direccion);
                        intent.putExtra(name: "nombre", mAuth.getCurrentUser().getEmail().toString());
                        startActivity(intent);
                        finish();
                    } else {
                        String error = ((FirebaseAuthException) task.getException()).getErrorCode();
                        dameToastdeerror(error);
                    }
                }
            });
        } else {
            Toast.makeText(context, this, text: "Completa todos los datos..", Toast.LENGTH_SHORT).show();
        }
    }
}
```

Registro esta ventana contiene un código muy similar al oncreate del login nos centraremos en la función que recoge los datos introducidos nos crea la cuenta nos manda el email para que validemos la cuenta y nos devuelve al login.

```
public void onClickRegistrofinal(View view) {

    String mail = correo.getText().toString();
    String pass = contraseña.getText().toString();
    String pass2 = repcontr.getText().toString();
    if (pass.equals(pass2)) {
        if (awesomeValidation.validate()) {
            mAuth.createUserWithEmailAndPassword(mail, pass).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()) {
                        Toast.makeText(context: Registrar.this, text: "Usuario creado con éxito", Toast.LENGTH_SHORT);
                        FirebaseUser user = mAuth.getCurrentUser();
                        user.sendEmailVerification();
                        finish();
                    } else {
                        String error = ((FirebaseAuthException) task.getException()).getErrorCode();
                        dameToastdeerror(error);
                    }
                }
            });
        } else {
            Toast.makeText(context: this, text: "Completa todos los datos..", Toast.LENGTH_SHORT);
        }
    } else {
        Toast.makeText(context: Registrar.this, text: "Las contraseñas no coinciden", Toast.LENGTH_SHORT);
    }
}
```

RecuperarContraseña de esta ventana también muestro la función que recoge el correo electronico valida que este registrado y posteriormente mande un correo para cambiar dicha contraseña.

```
public void onClickOlvidePass(View view) {
    String email = mail.getText().toString();

    firebaseAuth.sendPasswordResetEmail(email).addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            if (task.isSuccessful()) {
                Toast.makeText(context: RecuperarContrasenya.this, text: "Hemos enviado un email", Toast.LENGTH_SHORT);
                finish();
            } else {
                String error = ((FirebaseAuthException) task.getException()).getErrorCode();
                dameToastdeerror(error);
            }
        }
    })
}
```

La clase CartaFirebase y su adaptador

Esta es la clase que contiene el código de el acceso a la base de datos de firebase y la carga y funcionalidad de dicha carta, también contiene un botón que nos lleva al resumen de la comanda y otro que nos lleva al menú diario.

Función que contiene la carga de la carta:

```
public void cargar(){

    myRef.child("carta/entrantes").addValueEventListener(new ValueEventListener() {

        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            if (snapshot.exists()){
                Log.e( tag: "firebase", msg: "dins del if ");
                for (DataSnapshot producteSnapshot : snapshot.getChildren()) {

                    String n = producteSnapshot.child("nombre").getValue().toString();
                    String pr = producteSnapshot.child("precio").getValue().toString();
                    String url = producteSnapshot.child("foto").getValue().toString();
                    double preu = Double.parseDouble(pr);
                    p = new Producto(n, preu, unidades: 1, url);

                    listaEntrantes.add(p);
                }
                subcategor.put(categor.get(0), listaEntrantes);
            }
        }

        @Override
```

Esta función tiene 5 llamadas distintas dentro una por cada categoría de la carta. La variable myref se inicia en el método onCreate recogiendo la instancia a la base de datos y la referencia.

Función que añade al arraylist que mandaremos al resumen de la comanda con cada producto al que hayamos pulsado.

```
myListView.setOnChildClickListener(new ExpandableListView.OnChildClickListener() {
    @Override
    public boolean onChildClick(ExpandableListView parent, View v, int groupPosition, int childPosition, long id) {
        Producto p = null;
        if (groupPosition == 0) {
            p = new Producto(listaEntrantes.get(childPosition).nombre, listaEntrantes.get(childPosition).precio, unidades: 1, foto: '
            comanda.add(p);
        } else if (groupPosition == 1) {
            p = new Producto(listaPaellas.get(childPosition).nombre, listaPaellas.get(childPosition).precio, unidades: 1, foto: "");
            comanda.add(p);
        } else if (groupPosition == 2) {
            p = new Producto(listaFideuas.get(childPosition).nombre, listaFideuas.get(childPosition).precio, unidades: 1, foto: "");
            comanda.add(p);
        } else if (groupPosition == 3) {
            p = new Producto(listaPostres.get(childPosition).nombre, listaPostres.get(childPosition).precio, unidades: 1, foto: "");
            comanda.add(p);
        } else if (groupPosition == 4) {
            p = new Producto(listaVinos.get(childPosition).nombre, listaVinos.get(childPosition).precio, unidades: 1, foto: "");
            comanda.add(p);
        }
        Context context = getApplicationContext();
        CharSequence text = "Producto añadido";
        int duration = Toast.LENGTH_SHORT;

        Toast toast = Toast.makeText(context, text, duration);
        toast.show();
        return false;
    }
});
```

Esta parte de código sería la que nos pone toda la información de la carta cargada en la vista.

```
adapta = new Adaptador(categor, subcategor, this);
myListView.setAdapter(adapta);
```

Para la vista de la carta tuve que crear una clase que extiende de BaseAdapter para darle el formato visual de la carta.

La clase Adaptador

```
public class Adaptador extends BaseExpandableListAdapter {

    private ArrayList<String> categorias;
    private Map<String, ArrayList<Producto>> subcategoria;
    private Context context;

    public Adaptador(ArrayList<String> categorias, Map<String, ArrayList<Producto>> subcategoria, Context context) {
        this.categorias = categorias;
        this.subcategoria = subcategoria;
        this.context = context;
    }
}
```

Nos centraremos en las funciones que cargan la vista de las categorías y subcategorías.

```
@Override
public View getGroupView(int groupPosition, boolean isExpanded, View convertView, ViewGroup parent) {
    String titulo = (String) getGroup(groupPosition);
    convertView = LayoutInflater.from(context).inflate(R.layout.categoriasvista, root: null);
    TextView categoria = (TextView) convertView.findViewById(R.id.categoria);
    categoria.setText(titulo);
    return convertView;
}

@Override
public View getChildView(int groupPosition, int childPosition, boolean isLastChild, View convertView, ViewGroup parent) {
    Producto item = (Producto) getChild(groupPosition, childPosition);

    convertView = LayoutInflater.from(context).inflate(R.layout.subcategoriavista, root: null);
    TextView subcategoria = (TextView) convertView.findViewById(R.id.nombres);
    subcategoria.setText(item.nombre);

    TextView precio = (TextView) convertView.findViewById(R.id.precios);
    precio.setText(String.valueOf(String.format("%.2f", item.precio)));
    ImageView img = (ImageView) convertView.findViewById(R.id.imagenes);
    Log.e( tag: "foto", msg: "adaptador foto: " + item.getFoto());
    Glide.with(context).load(item.getFoto()).into(img);
    //img.setImageResource(item.foto);
}
```

La clase NuestroMenu contiene la vista del menu que esta formada por tres grupos de 'radiobuttons' uno para los entrantes otro para los principales y otro para los postres, tambien contiene dos botones uno para añadir el menu y otro para volver a la carta.

Función que recoge los platos de realtime database

```
public void getLlistaProductes(final MyCallback myCallback){
    myRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange( DataSnapshot dataSnapshot) {
            llistaProductes.clear();
            for (DataSnapshot producteSnapshot: dataSnapshot.getChildren())
                producte = (String) producteSnapshot.getKey();
            llistaProductes.add(producte);
        }
        // Into onDataChange method!!
        myCallback.onCallback(llibraProductes);
    }

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

    }

});
```

Función que carga los el nombre de los platos en los radiobutons.

```
getLlistaProductes(new MyCallback() {
    @Override
    public void onCallback(ArrayList products) {
        String e1 = llistaProductes.get(0);
        entrante1.setText(e1);
        String e2 = (String) llistaProductes.get(1);
        entrante2.setText(e2);
        String e3 = (String) llistaProductes.get(2);
        entrante3.setText(e3);
        String e4 = (String) llistaProductes.get(3);
        entrante4.setText(e4);

        String e5 = llistaProductes.get(4);
```

Función que añade el menú seleccionado a la comanda con sus platos elegidos.

```
public void OnClickAnadir(View view){
    if (entrante1.isChecked() || entrante2.isChecked() || entrante3.isChecked() || entrante4.isChecked()){
        if(principal1.isChecked() || principal2.isChecked() || principal3.isChecked() || principal4.isChecked()){
            if(postre1.isChecked() || postre2.isChecked() || postre3.isChecked() || postre4.isChecked()){
                if(entrante1.isChecked()==true){
                    primero= entrante1.getText().toString();
                }
                if(entrante2.isChecked()==true){
                    primero= entrante2.getText().toString();
                }
            }
            descripcion = primero+"\n"+segundo+"\n"+postre;
            Toast toast = Toast.makeText( context: this, text: "Menu añadido", Toast.LENGTH_SHORT);
            toast.show();
            Producto p = new Producto( nombre: "Menu Diario", precio: 12,descripcion, unidades: 1);
            llistaMenus.add(p);
        }else{
            Context context = getApplicationContext();
            CharSequence text = "Debes seleccionar 3 platos";
            int duration = Toast.LENGTH_SHORT;

            Toast toast = Toast.makeText(context, text, duration);
            toast.show();
        }
    }
}
```

Función para volver a la carta y mandar la información del menú añadido para su correcta visualización en el resumen del pedido.

```
public void onClickCarta(View view) {
    Intent intent = new Intent( packageContext: this, CartaFirestore.class);
    int q = llistaMenus.size();

    intent.putExtra( name: "quant", q);
    intent.putExtra( name: "dir",direccion);
    intent.putExtra( name: "nombre", nombre);
    for (int i = 0; i < llistaMenus.size();i++) {
        intent.putExtra( name: "prod"+i, llistaMenus.get(i).getNombre());
    }
    for (int i = 0; i < llistaMenus.size();i++) {
        intent.putExtra( name: "preu"+i, llistaMenus.get(i).getPrecio());
    }
    for (int i = 0; i < llistaMenus.size();i++) {
        intent.putExtra( name: "unit"+i, llistaMenus.get(i).getUnidades());
    }
    for (int i = 0; i < llistaMenus.size();i++) {
        intent.putExtra( name: "descripcion"+i, llistaMenus.get(i).getDescripcion());
    }
    for (int j = 0; j < llistaMenus.size();j++){
        llistaMenus.remove(j);
    }
    startActivity(intent);
}
```


La Clase Carrito y su adaptador:

Función que nos carga el carrito cuando iniciamos la aplicación esta función es llamada en el onCreate.

```
public void cargarCarrito(){
    Bundle extras = getIntent().getExtras();
    int num = extras.getInt( key: "quant");
    ArrayList<String> prod = new ArrayList<>();
    ArrayList<String> descr = new ArrayList<>();
    ArrayList<Double> preu = new ArrayList<>();
    ArrayList<Integer> uni = new ArrayList<>();
    for (int i = 0; i < num;i++) {

        prod.add(extras.getString( key: "prod"+i));
        preu.add(extras.getDouble( key: "preu"+i));
        uni.add(extras.getInt( key: "unit"+i));
        descr.add(extras.getString( key: "descripcion"+i));
        Log.e( tag: "obj", msg: "cargar carrito "+prod.get(i));
    }
    direccion =extras.getString( key: "dir");
    nombre =extras.getString( key: "nombre");
    for (int i = 0; i < prod.size();i++) {
        Producto p = new Producto(prod.get(i), preu.get(i),descr.get(i), uni.get(i));
        prodComanda.add(p);
    }

    adaptan = new AdaptadorCarrito( context: this, prodComanda);
    myListView.setAdapter(adaptan);
}
```

Función que muestra el total de la comanda :

```
public static void totalDinero(){
    double tt = 0;
    for (int i = 0;i < prodComanda.size();i++){
        tt += prodComanda.get(i).getPrecio()*prodComanda.get(i).getUnidades();
    }
    String t =String.valueOf(tt);
    total.setText(String.valueOf(String.format("%.2f",tt)));
}
```

Función que enviá el contenido de la comanda del carrito,nombre y dirección a la siguiente pantalla:

```

    }
    boton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent( packageContext: Carrito.this, Finalizado.class);
            int q = prodComanda.size();
            intent.putExtra( name: "nombre", nombre);
            intent.putExtra( name: "dir",direccion);
            intent.putExtra( name: "quant", q);
            for (int i = 0; i < prodComanda.size();i++) {
                intent.putExtra( name: "prod"+i, prodComanda.get(i).getNombre());
                intent.putExtra( name: "preu"+i, prodComanda.get(i).getPrecio());
                intent.putExtra( name: "unit"+i, prodComanda.get(i).getUnidades());
                intent.putExtra( name: "descripcion"+i,prodComanda.get(i).getDescripcion());
            }
            /*for (int j = 0; j < prodComanda.size();j++){
                prodComanda.remove(j);
            }*/
            startActivity(intent);
        }
    });

    totalDinero();

```

La vista del listado de la comanda se construye con un adaptador que extiende de BaseAdapter, como en la de la vista de la carta solo que esta vez no es expandible es una lista única por lo tanto no esta formada con categorías y subcategorías, aunque de todos modos la vista se construye de un método mas extenso ya que había que darle funcionalidad a los botones de sumar y restar, para conseguir dicha funcionalidad me he ayudado de una clase estática llamada holder:

```

static class Holder    {
    public Button botonmas, botonmenos;
    TextView cantidad;

    public Button getBotonmas() { return botonmas; }
    public void setBotonmas(Button botonmas) { this.botonmas = botonmas; }
    public Button getBotonmenos() { return botonmenos; }
    public void setBotonmenos(Button botonmenos) { this.botonmenos = botonmenos; }
    public TextView getCantidad() { return cantidad; }
    public void setCantidad(TextView cantidad) {
        this.cantidad = cantidad;
    }
}

```

Función que carga la vista de la lista del carrito:

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    convertView = LayoutInflater.from(context).inflate(R.layout.comanda, root: null);
    Producto item = (Producto) getItem(position);
    Holder holder = null;

    holder = new Holder();

    // Els botons i la quantitat
    holder.setBotonmas((Button) convertView.findViewById(R.id.sumar));
    holder.setBotonmenos((Button) convertView.findViewById(R.id.restar));
    holder.setCantidad( convertView.findViewById(R.id.contador));

    convertView.setTag(holder);

    TextView prod = (TextView) convertView.findViewById(R.id.productoPedido);
    prod.setText(item.nombre);
    TextView d = convertView.findViewById(R.id.descripcion);
    d.setText(item.descripcion);
    TextView prodprec = (TextView) convertView.findViewById(R.id.productoPedidoPrecio);
    prodprec.setText(String.valueOf(String.format("%.2f", item.precio)));

    TextView tvQuantitat = holder.getCantidad();
    tvQuantitat.setText(String.valueOf(item.unidades));
```

Función que da funcionalidad a los botones de sumar y restar , incrementan las unidades y hace una posterior llamada a la función de calcular el total, para que se actualiza el total cuando añadimos o restamos unidades:

```
holder.getBotonmas().setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        int quantitat = Carrito.prodComanda.get(position).getUnidades();
        quantitat++;

        Carrito.prodComanda.get(position).setUnidades(quantitat);
        tvQuantitat.setText(String.valueOf(Carrito.prodComanda.get(position).getUnidades()));
        Log.e( tag: "unitat ", msg: "esta en mes: "+Carrito.prodComanda.get(position).getUnidades() );
        Carrito.totalDinero();
    }
});
holder.getBotonmenos().setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        int quantitat = Carrito.prodComanda.get(position).getUnidades();
        if (quantitat > 0) quantitat--;

        Carrito.prodComanda.get(position).setUnidades(quantitat);
        tvQuantitat.setText(String.valueOf(Carrito.prodComanda.get(position).getUnidades()));
        Log.e( tag: "unitat", msg: "esta en menos: "+Carrito.prodComanda.get(position).getUnidades() );
        Carrito.totalDinero();
    }
});
```

La clase Finalizado y su adaptador.

Esta clase contiene la vista final de nuestro pedido que ya no es editable, también nos muestra la dirección a la que irá dicho pedido, el nombre del cliente y los métodos de pago.

En el método onCreate principalmente lo que se hace es la llamada a las funciones para cargar toda la información que debe aparecer cuando se carga la pantalla.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_finalizado);
    getSupportActionBar().setTitle("RestApp");
    String bb = "#E4F4C536";
    getSupportActionBar().setBackgroundDrawable(new ColorDrawable(Color.parseColor(bb)));

    finaliza = findViewById(R.id.confirma);
    lw = findViewById(R.id.listComand);
    rbef = findViewById(R.id.efectivo);
    rbtj = findViewById(R.id.tarjeta);

    contenidoDireccion = findViewById(R.id.contenidoDireccion);
    contenidoNombre = findViewById(R.id.nom);
    total = findViewById(R.id.total);

    db = FirebaseDatabase.getInstance().getReference();

    cargarCarrito();
    cargarDireccion();
    totalDinero();
}
```

Función que carga la comanda y la información:

```
public void cargarCarrito(){
    Bundle extras = getIntent().getExtras();
    int num = extras.getInt( key: "quant");
    ArrayList<String> prod = new ArrayList<>();
    ArrayList<String> descr = new ArrayList<>();
    ArrayList<Double> preu = new ArrayList<>();
    ArrayList<Integer> uni = new ArrayList<>();
    direccion = extras.getString( key: "dir");
    nombre = extras.getString( key: "nombre");
    for (int i = 0; i < num;i++) {
        prod.add(extras.getString( key: "prod"+i));
        preu.add(extras.getDouble( key: "preu"+i));
        uni.add(extras.getInt( key: "unit"+i));
        descr.add(extras.getString( key: "descripcion"+i));
        Log.e( tag: "obj", msg: "cargar finalizado "+prod.get(i));
    }
    for (int i = 0; i < prod.size();i++) {
        Producto p = new Producto(prod.get(i), preu.get(i), descr.get(i),uni.get(i));
        prodComanda.add(p);
    }
    adapta = new AdaptaComanda( context: Finalizado.this, prodComanda);
    lw.setAdapter(adapta);
}
```

Funciones que cargan el total de la comanda y la dirección:

```
public void cargarDireccion(){
    contenidoDireccion.setText(direccion);
    contenidoNombre.setText(nombre);
}

public static void totalDinero(){
    double tt = 0;
    for (int i = 0;i < prodComanda.size();i++){
        tt += prodComanda.get(i).getPrecio()*prodComanda.get(i).getUnidades();
    }
    String t =String.valueOf(tt);
    total.setText(String.valueOf(String.format("%.2f",tt)));
}
```

Función de seguro por si el cliente retrocede a la pagina anterior para que cuando vuelva a cambiar a esa pantalla no salga todo el pedido por duplicado:

```

    @Override
    public void onBackPressed() {
        for (int j = 0; j < prodComanda.size();j++){
            prodComanda.remove(j);
        }
        finish();
    }
}
```

Función para que el botón de editar la dirección nos lleve a la pantalla para poder editarla.

```

    public void onClickEditaDireccion(View view) {
        Intent intent = new Intent( packageContext: Finalizado.this, ConfirmarDireccion.class);
        String cl="Finalizado";
        intent.putExtra( name: "nom", cl);
        intent.putExtra( name: "nombre", nombre);
        startActivity(intent);
    }
}
```

Función que sube la comanda a Firebase cuando confirmamos el pedido con el botón y además nos lleva a la pantalla final.

```
public void onClickfin(View v) {
    if ((rbeef.isChecked() || rbty.isChecked()) && contenidoDireccion.getText().length() != 15 && total.getText() != "0.00") {
        if (prodComanda.size() != 0) {
            Map<String, Object> comandaSubida = new HashMap<>();
            int i;
            for (i = 0; i < prodComanda.size(); i++) {
                Producto pp = new Producto(prodComanda.get(i).nombre, prodComanda.get(i).precio, prodComanda.get(i).descripcion,
                    comandaSubida.put("Producto " + i, pp);
            }
            Boolean st = true;
            comandaSubida.put("Estado ", st);
            Time time = new Time(Time.getCurrentTimezone());
            time.setToNow();
            int h = time.hour;
            int m = time.minute;
            String hora = String.valueOf(h) + ":" + String.valueOf(m);
            comandaSubida.put("Hora", hora);
            comandaSubida.put("Direccion ", direccion);
            comandaSubida.put("Cliente", nombre);
            db.child("comandas").push().setValue(comandaSubida);
            prodComanda.clear();
            adapta = new AdaptaComanda(context: Finalizado.this, prodComanda);
            lw.setAdapter(adapta);
            Intent intent = new Intent(packageContext: Finalizado.this, ConfirmacionPedido.class);
            startActivity(intent);
        }
    }
}
```

El adaptador para la vista final de la comanda, este adaptador tambien extiende de Base Adapter, esta vez es la mas sencilla de todas ya que la lista no es desplegable y ademas no tiene ninguna funcionalidad ya que solo sirve para ver la comanda final.

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    convertView = LayoutInflater.from(context).inflate(R.layout.layout_comandafinal, root: null);
    Producto item = (Producto) getItem(position);

    TextView prod = (TextView) convertView.findViewById(R.id.prodfin);
    prod.setText(item.nombre);
    TextView prodprec = (TextView) convertView.findViewById(R.id.preciouni);
    prodprec.setText(String.valueOf(String.format("%.2f", item.precio)));
    Double tt = item.getPrecio() * item.getUnidades();
    TextView quant = (TextView) convertView.findViewById(R.id.totaluni);
    prodprec.setText(String.valueOf(String.format("%.2f", tt)));
    quant.setText(String.valueOf(item.unidades));

    return convertView;
}
```


La clase ConfirmacionPedido

Esta clase no tiene funcionalidades ya que solo nos sirve para hacer saber al cliente que su pedido de ha confirmado ademas de hacerle saber la hora de llegada de dicho pedido.

Función onCreate donde se obtiene la hora final y la muestra.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_confirmacion_pedido);
    getSupportActionBar().setTitle("RestApp");
    String bb = "#E4F4C536";
    getSupportActionBar().setBackgroundDrawable(new ColorDrawable(Color.parseColor(bb)));

    Date currentTime = Calendar.getInstance().getTime();
    Time time = new Time(Time.getCurrentTimezone());
    time.setToNow();
    String hfin;
    int h = time.hour;
    int m = time.minute;
    int mfinal = m+40;
    if (mfinal >= 60 && mfinal-60 < 10) {
        mfinal = mfinal - 60;
        h = h + 1;
        hfin = String.valueOf(h) + ":0" + String.valueOf(mfinal);
    }else if(mfinal >= 60){
        mfinal = mfinal - 60;
        h = h + 1;
        hfin = String.valueOf(h) + ":" + String.valueOf(mfinal);
    }else{
        hfin = String.valueOf(h)+":"+String.valueOf(mfinal);
    }
    horallegada = findViewById(R.id.horafin);
    horallegada.setText(hfin);
}
```



Las clases Contacto Promociones y Sobre nosotros:

Estas tres clases son muy sencillas tanto que no tienen programación alguna tan solo hay que ponerle la imagen que queremos mostrar en el layout mas adelante mostrare lo necesario para poder reproducir el gif en la clase Promociones.

La clase Producto

Esta clase no es ninguna pantalla en la aplicación tan solo contiene un objeto llamado producto con sus atributos, constructores, getters y setters. Es la encargada de instanciar los productos de la aplicación tanto en la cata como vistas de las listas.

```
public class Producto {
    String nombre;
    String foto;
    double precio;
    String descripcion;
    int unidades;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Producto producto = (Producto) o;
        return nombre.equals(producto.nombre);
    }

    public String getNombre() { return nombre; }

    public void setNombre(String nombre) { this.nombre = nombre; }

    public String getFoto() { return foto; }

    public void setFoto(String foto) { this.foto = foto; }

    public double getPrecio() { return precio; }

    public void setPrecio(double precio) { this.precio = precio; }

    public String getDescripcion() { return descripcion; }

    public void setDescripcion(String descripcion) { this.descripcion = descripcion; }

    public int getUnidades() { return unidades; }

    public void setUnidades(int unidades) { this.unidades = unidades; }
```

```

public Producto(String nombre, double precio, int unidades, String foto) {
    this.nombre = nombre;
    this.precio = precio;
    this.unidades = unidades;
    this.foto = foto;
}

public Producto(String nombre , double precio, String foto) {
    this.nombre = nombre;
    this.precio = precio;
    this.foto = foto;
}

public Producto(String nombre , double precio) {
    this.nombre = nombre;
    this.precio = precio;
}

public Producto(String nombre , double precio, String descripcion, int unidades) {
    this.nombre = nombre;
    this.precio = precio;
    this.unidades = unidades;
    this.descripcion = descripcion;
}

```

Manifest y build.gradle

El archivo Manifest contiene lo necesario para obtener la ubicación, la api-key para google maps y lo necesario para las fuentes de la aplicación.

```

    Location permissions for the "MyLocation" functionality.
-->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

-->
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="AIzaSyD6T0Bdg4cQkmY_ioZH1lA03RT30jFetvU" />
<meta-data
    android:name="preloaded_fonts"
    android:resource="@array/preloaded_fonts" />

```

Build.gradle existen dos ficheros.

El primero contiene las dependencias necesarias para todo lo que hemos implementado en la aplicación.

```
dependencies {  
    implementation "com.google.android.gms:play-services-location:15.0.1"  
    implementation 'androidx.appcompat:appcompat:1.2.0'  
    implementation 'com.google.android.material:material:1.3.0'  
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'  
    implementation 'androidx.navigation:navigation-fragment:2.3.2'  
    implementation 'androidx.navigation:navigation-ui:2.3.2'  
    implementation 'com.google.android.gms:play-services-maps:17.0.0'  
    implementation 'com.google.firebase:firebase-database:19.7.0'  
    implementation 'com.google.firebase:firebase-analytics:18.0.3'  
    implementation 'com.google.firebase:firebase-storage:19.2.2'  
    implementation 'androidx.gridlayout:gridlayout:1.0.0'  
    implementation 'com.google.firebase:firebase-auth:20.0.4'  
    implementation 'com.basgeekball:awesome-validation:4.3'  
    testImplementation 'junit:junit:4.+'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'  
    implementation 'com.github.bumptech.glide:glide:4.11.0'  
    implementation 'pl.droidsonroids.gif:android-gif-drawable:1.2.7'  
}
```

El segundo contiene, repositorio jcenter, dependencias de servicios de google y lo necesario para incluir el AwesomeValidation encargado de dar formato valido a el correo y la contraseña que se introduce en toda la parte referida del login, registro o recuperación de contraseña.

```
buildscript {
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath "com.android.tools.build:gradle:4.1.0"
        classpath 'com.google.gms:google-services:4.3.5'

        // NOTE: Do not place your application dependencies here
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()
        maven { url 'https://jitpack.io' }
    }
}
```

Teniendo en cuenta que el proyecto está basado en una simulación y no en un restaurante real, hay algunos aspectos que no se han llevado a cabo en su completa totalidad. Estas funcionalidades están diferenciadas por ser más o menos necesarias en el caso de llegar a desarrollar el proyecto y por tanto, dejar de ser un supuesto.

En primer lugar y como aspecto ineludible, destaca el hecho de no contar con servicios de pago online. Se trata de la posibilidad de realizar el pago mediante tarjeta de crédito o débito, o a través de las modalidades PayPal y Bizum. Como ya he mencionado anteriormente, en la actualidad no se puede prescindir de estas opciones, ya que son las preferidas por los consumidores por su rapidez y comodidad.

Por otra parte encontramos el tiempo de espera establecido para los pedidos. Como ya hemos visto, este se ha calculado añadiendo 40 minutos a la hora actual en la que el usuario se encuentra al realizar el pedido. Este aspecto es muy genérico, ya que pueden mejorarse las predicciones teniendo en cuenta otras variables considerables como la distancia existente entre el restaurante y el punto de entrega. Además de esto, se puede mejorar la estimación del tiempo si se tiene en cuenta el número total de pedidos en cola y cualquier otro aspecto que el gerente considere oportuno incluir.

Finalmente, otra forma de mejorar la aplicación es una ampliación del número de campos disponibles en el registro de usuario. Esto nos permite obtener un perfil más detallado de cada cliente y por tanto, nos da la oportunidad de ofrecerle una atención personalizada.

Conclusiones

En esta parte me dedico a sacar conclusiones finales y personales del proyecto, extraídas a lo largo de todo el proceso de desarrollo del mismo.

Se han cumplido todos los objetivos y requisitos mencionados. El principal de ellos era crear una aplicación móvil para Android capaz de ofrecer servicio de comida a domicilio a los usuarios por medio de este canal.

La aplicación desarrollada ofrece la posibilidad de que le lleven a los clientes los pedidos tramitados a través de ella, a su domicilio ademas permite la consulta de contacto con la empresa, vía correo o teléfono.

Atendiendo al desarrollo técnico del proyecto, fue difícil sintetizar todas las ideas y requisitos que había sobre la mesa. Una vez realizadas las fases de análisis y diseño, las fases posteriores fueron encauzadas rapidamente. Por lo que las primeras fases de planificación, análisis y diseño resultaron ser de importancia.

Ademas de haber aprendido a desarrollar un proyecto sola de principio a fin, pasando por todas sus fases, he adquirido nuevos conocimientos o consolidado muchos de ellos sobre Java y Android Studio.

Bibliografía

La mayoría de la información necesaria para poder hacer el proyecto la he ido recogiendo de video tutoriales de YouTube o en el propio manual de Android Studio.

Dejo algunos enlaces que me han resultado interesantes para recoger la información:

De este canal concreto a sacado muchísima información ya que su canal esta muy completo de videos referentes a Android Studio.

<https://www.youtube.com/user/neto376>

Enlace a videos utilizados:

https://www.youtube.com/watch?v=V8xjWKR0_0E

<https://www.youtube.com/watch?v=hQCYBeH8sz4&t=658s>

<https://www.youtube.com/watch?v=aVyKywyFOzM>

<https://www.youtube.com/watch?v=k-aMy3t8Lng&t=23s>

<https://www.youtube.com/watch?v=7-LrsDclHeY>

<https://www.youtube.com/watch?v=90G20Z1n7ag&t=1217s>

<https://www.youtube.com/watch?v=lhhhQhE4CVk>

<https://www.youtube.com/watch?v=iMoN414EfSQ>

<https://www.youtube.com/watch?v=9rutlLv0fRQ>

<https://www.youtube.com/watch?v=piW96IV8yls>

Enlace manual Android:

<https://developer.android.com/studio/intro?hl=es-419>