

PERÍODO 2019



## TRABAJO AUTÓNOMO 7

### DESARROLLO DE UNA APLICACIÓN MÓVIL USANDO CONTROLES BÁSICOS Y REPOSITORIOS DE GITHUB

#### INTEGRANTES

HUGO MARTIN ANDERICA URQUIZO  
MARIA DEL CISNE FEIJOO JARAMILLO  
DARLY LISSETTE GUERRERO GALVEZ  
GUIDO ALEJANDRO RAMIREZ ESPINOZA

#### PARALELO 1 GRUPO 4

FECHA DE INICIO: 23 DE JULIO DE 2019.  
FECHA DE ENTREGA: 06 DE AGOSTO DE 2019.

ELABORADO POR: ADRIANA COLLAGUAZO JARAMILLO

MATERIA: PROGRAMACIÓN DE SISTEMAS TELEMÁTICOS

CARRERA DE INGENIERÍA EN MECATRÓNICA

FIEC-ESPOL

## ACTIVIDADES

### Paso 1. Creacion de Repositorio en GitHub

Link de repositorio: [https://github.com/CisneFeijoo1998/PST\\_TA7\\_G4.git](https://github.com/CisneFeijoo1998/PST_TA7_G4.git)

CisneFeijoo1998 / PST\_TA7\_G4

Code Issues Pull requests Projects Wiki Security Insights Settings

Este proyecto es creado para el desarrollo del trabajo autónomo 7

Manage topics

26 commits 3 branches 0 releases 2 contributors

Branch: master New pull request Create new file Upload files Find File Clone or download

Hugo Correccion medidas

.idea Creacion de diseño de main activity 16 hours ago  
app Corrección medidas 1 hour ago  
gradle/wrapper Commit inicial 3 days ago  
.gitignore Commit inicial 3 days ago  
build.gradle Commit inicial 3 days ago  
gradle.properties Commit inicial 3 days ago  
gradlew Commit inicial 3 days ago  
gradlew.bat Commit inicial 3 days ago  
settings.gradle Commit inicial 3 days ago

Latest commit s1ec814 1 hour ago

Add a README

© 2019 GitHub, Inc. Terms Privacy Security Status Help



Contact GitHub Pricing API Training Blog About

### Paso 2: Invitar a otros miembros del grupo a mi proyecto

CisneFeijoo1998 / PST\_TA7\_G4

Code Issues Pull requests Projects Wiki Security Insights Settings

Options Collaborators Push access to the repository

diguerre

guirespi

Handerica

Search by username, full name or email address  
You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

Add collaborator

Moderation Interaction limits

© 2019 GitHub, Inc. Terms Privacy Security Status Help



Contact GitHub Pricing API Training Blog About

## Paso 3: Crear una rama [branch]

The screenshot shows a GitHub repository page for 'CisneFeijoo1998 / PST\_TA7\_G4'. The 'Overview' tab is selected. In the 'Default branch' section, the 'master' branch is listed with an update from 2 hours ago. A 'Default' button and a 'Change default branch' link are present. In the 'Your branches' section, a new branch 'Maria\_Feijoo\_Jaramillo' is shown, updated 2 days ago. It has 28 commits and 0 pull requests. Buttons for 'New pull request' and a trash icon are available. In the 'Active branches' section, three other branches are listed: 'Darly\_Guerrero\_Galvez' (17 commits, 2 pull requests), 'Guido\_Ramirez' (17 commits, 0 pull requests), and 'Maria\_Feijoo\_Jaramillo' (28 commits, 0 pull requests). Each branch row includes a 'New pull request' button and a trash icon.

© 2019 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

## Paso 4: Unir ramas al proyecto principal [branch].

```
Usuario@DESKTOP-FPASOG4 MINGW64 ~/Documents/UNIVERSIDAD/u/SEPTIMO SEMESTRE/PST/P
ST_TA7_G4 (master)
$ git pull origin master
From https://github.com/CisneFeijoo1998/PST_TA7_G4
 * branch      master      -> FETCH_HEAD
Auto-merging app/src/main/res/values/strings.xml
CONFLICT (content): Merge conflict in app/src/main/res/values/strings.xml
Auto-merging app/src/main/java/com/example/pst_ta7_g4/MainActivity.java
CONFLICT (content): Merge conflict in app/src/main/java/com/example/pst_ta7_g4/M
ainActivity.java
Automatic merge failed; fix conflicts and then commit the result.

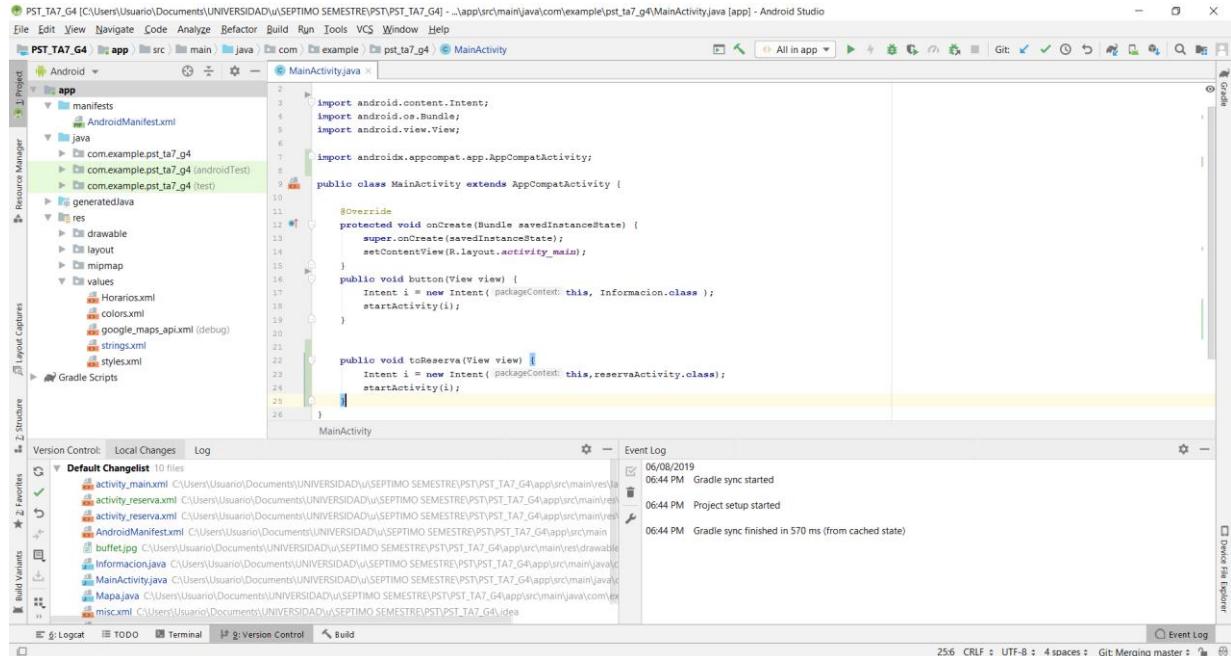
Usuario@DESKTOP-FPASOG4 MINGW64 ~/Documents/UNIVERSIDAD/u/SEPTIMO SEMESTRE/PST/P
ST_TA7_G4 (master|MERGING)
$ git add .

Usuario@DESKTOP-FPASOG4 MINGW64 ~/Documents/UNIVERSIDAD/u/SEPTIMO SEMESTRE/PST/P
ST_TA7_G4 (master|MERGING)
$ git commit -m "Implementacion de activity Informacion, donde presenta datos del
restaurante como ubicacion, telefonos y horarios de atencion"
[master 7dd0234] Implementacion de activity Informacion, donde presenta datos del
restaurante como ubicacion, telefonos y horarios de atencion

Usuario@DESKTOP-FPASOG4 MINGW64 ~/Documents/UNIVERSIDAD/u/SEPTIMO SEMESTRE/PST/P
ST_TA7_G4 (master)
$ git push origin master
Enumerating objects: 56, done.
Counting objects: 100% (56/56), done.
Delta compression using up to 4 threads
Compressing objects: 100% (22/22), done.
Writing objects: 100% (28/28), 2.28 KiB | 776.00 KiB/s, done.
Total 28 (delta 18), reused 0 (delta 0)
remote: Resolving deltas: 100% (18/18), completed with 9 local objects.
To https://github.com/CisneFeijoo1998/PST_TA7_G4.git
  ee2aa64..7dd0234  master -> master

Usuario@DESKTOP-FPASOG4 MINGW64 ~/Documents/UNIVERSIDAD/u/SEPTIMO SEMESTRE/PST/P
ST_TA7_G4 (master)
$
```

## Paso 5: Crear un nuevo proyecto en Android Studio



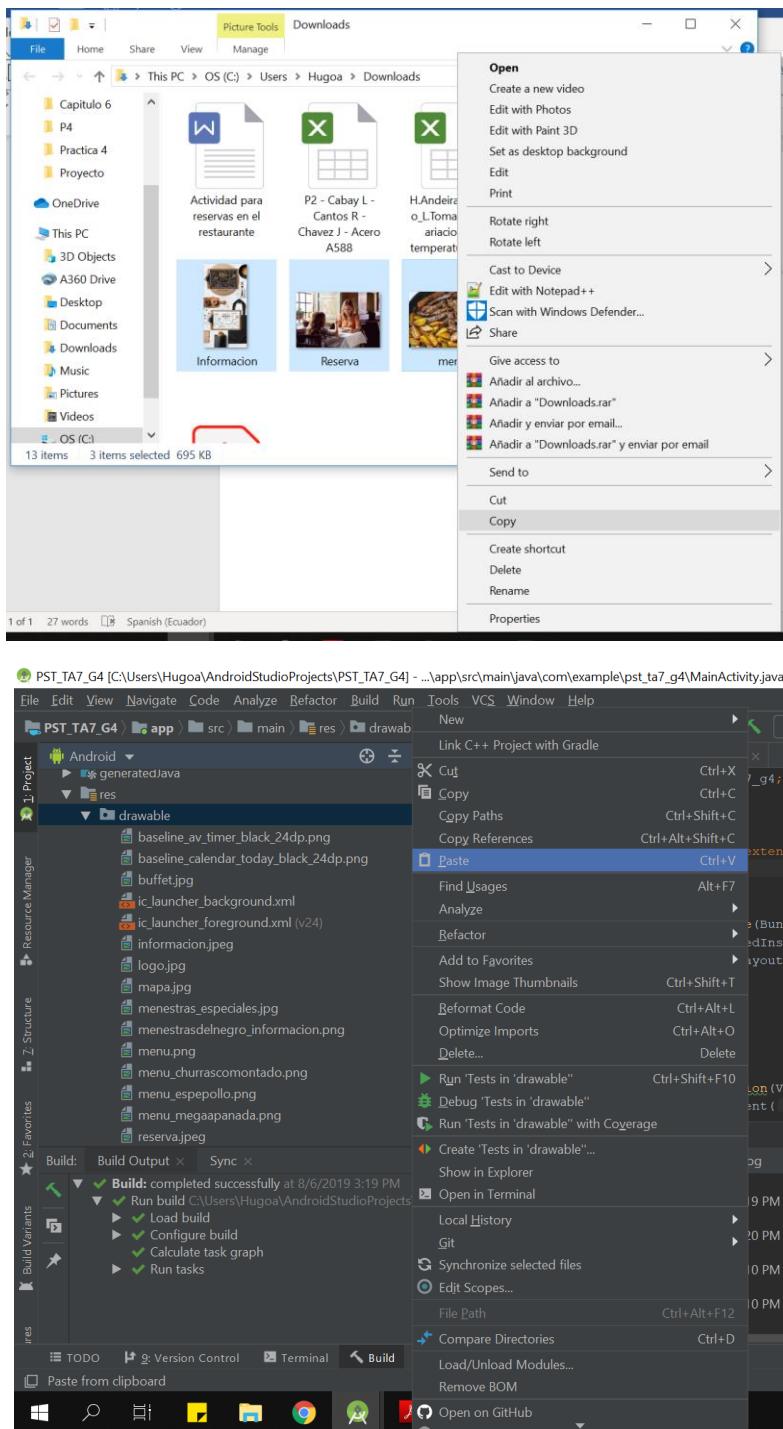
## Paso 6. Implementación de la Aplicación

El siguiente trabajo consistió en el desarrollo de una aplicación para el restaurante “Menestras del Negro”, un restaurante dedicado principalmente a la venta de comida ecuatoriana tradicional. Para el desarrollo de la misma utilizamos varios controles básicos como Button, ImageButton, ArrayAdapter, Spinner, entre otros, los cuales nos permitieron poder implementar nuestra app de forma sencilla.

La aplicación consiste en una página principal la cual contiene varias opciones para el usuario, una de ellas es la sección Información, la cual nos permite consultar la ubicación de sus restaurantes, sus horarios de atención y el teléfono de contacto. La siguiente opción permite observar los distintos menú que ofrece el restaurante, con una breve descripción de cada, y finalmente como última opción se permite al usuario una alternativa para que pueda realizar sus reservas desde la app, para lo cual se muestra un calendario y un reloj, donde según el día y la hora seleccionada se agendará una reserva, además como método de confirmación que la reserva ha sido realizada exitosamente, llegará un SMS al celular del usuario con la información de su reserva.

## Implementación del MainActivity

1. Copiamos las imágenes que queremos incluir en el diseño de la aplicación dentro de la carpeta drawable utilizando el direccionamiento de Android Studio.



2. En la sección de diseño (xml) del MainActivity escribimos el código para agregar un ImageView al activity y reemplazamos con cada uno de sus parámetros los requerimientos de dicho ImageView. Es recomendable establecer el prefijo “img\_nombreImagen” en cada ImageView para así saber de qué tipo de componente se trata y a que imagen está referida. Para efectos de diseño se modificó la opacidad de las imágenes agregadas.

MainActivity.java

```
3 <ConstraintLayout ...>
4     xmlns:android="http://schemas.android.com/apk/res/android"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".MainActivity">
9
10    <ImageView
11        android:id="@+id/img_Encabecada"
12        android:layout_width="wrap_content"
13        android:layout_height="wrap_content"
14        android:layout_marginTop="16dp"
15        android:onClick="toInformacion"
16        app:layout_constraintStart_toStartOf="parent"
17        app:layout_constraintTop_toTopOf="parent"
18        app:srcCompat="@drawable/logo" />
19
20    <ImageView
21        android:id="@+id/img_menu"
22        android:layout_width="414dp"
23        android:layout_height="246dp"
24        android:layout_marginTop="144dp"
25        android:onClick="toMenu"
26        android:tint="#33FFFFFF"
27        app:layout_constraintStart_toStartOf="parent"
28        app:layout_constraintTop_toTopOf="parent"
29        app:srcCompat="@drawable/menu" />
30
31    <ImageView
32        android:id="@+id/img_reserva"
33        android:layout_width="414dp"
34        android:layout_height="283dp"
35        android:layout_marginTop="396dp"
36        android:background="#1AFFFFFF"
37        android:onClick="toReserva"
38        android:tint="#33FFFFFF"
39        app:layout_constraintStart_toStartOf="parent"
40        app:layout_constraintTop_toTopOf="parent"
41        app:srcCompat="@drawable/reserva" />
42
43    <TextView
44        android:id="@+id/lbl_reserva"
45        android:layout_width="162dp"
46        android:layout_height="51dp"
47        android:layout_marginTop="604dp"
48        android:layout_marginEnd="232dp"
49
50    androidx.constraintlayout.widget.ConstraintLayout > ImageView
```

activity\_main.xml

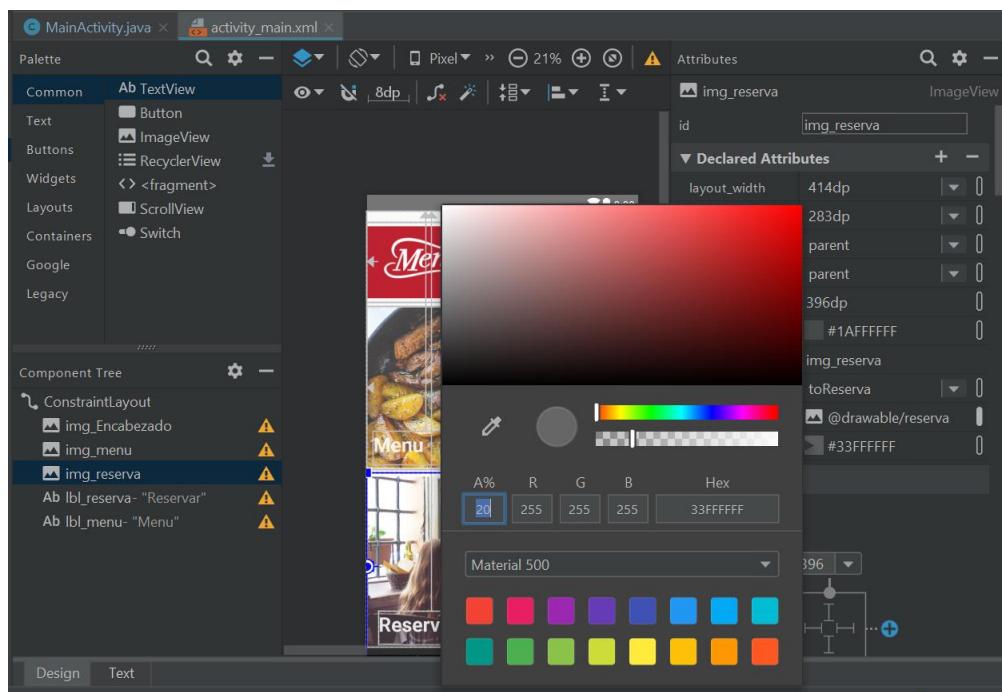
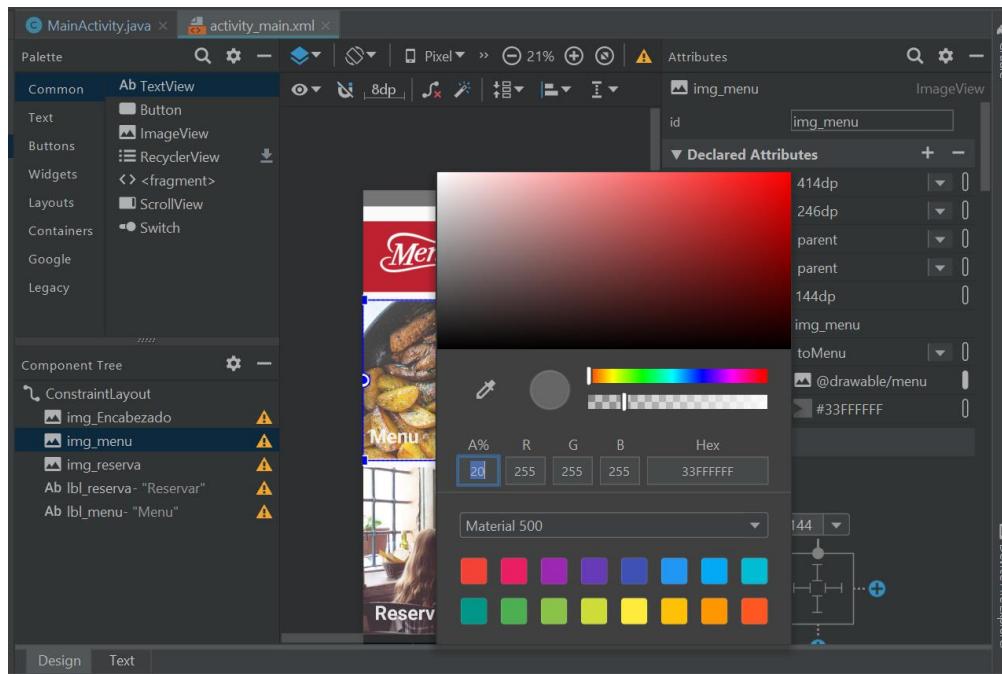


MainActivity.java

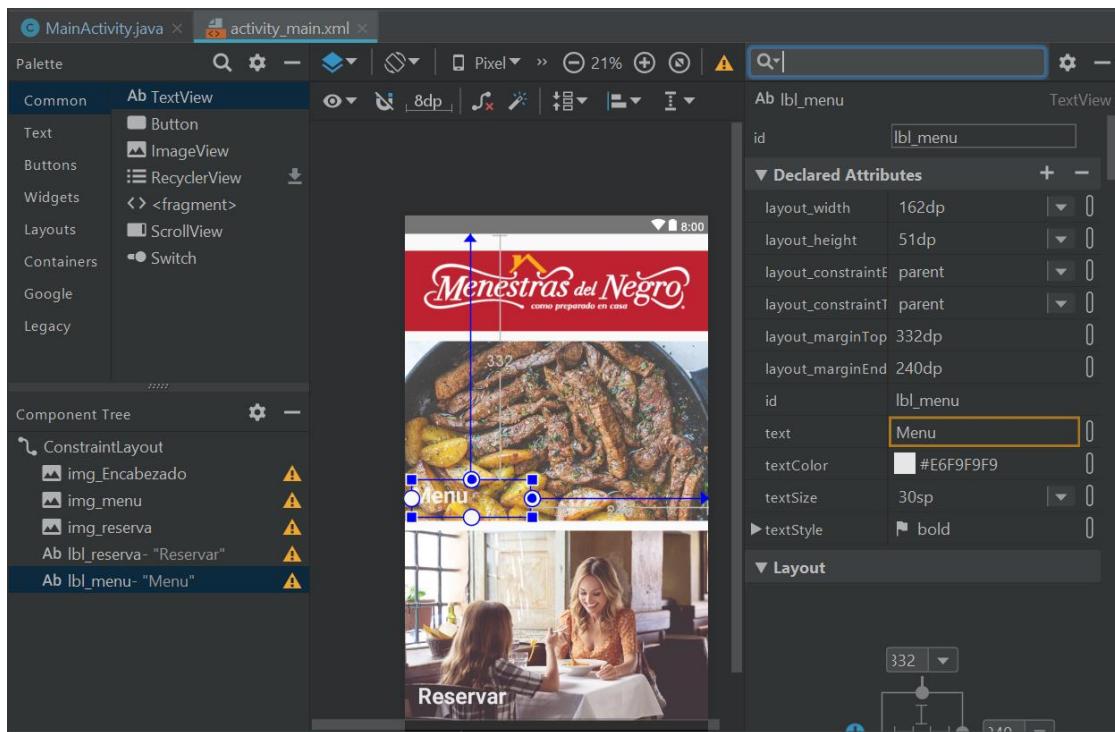
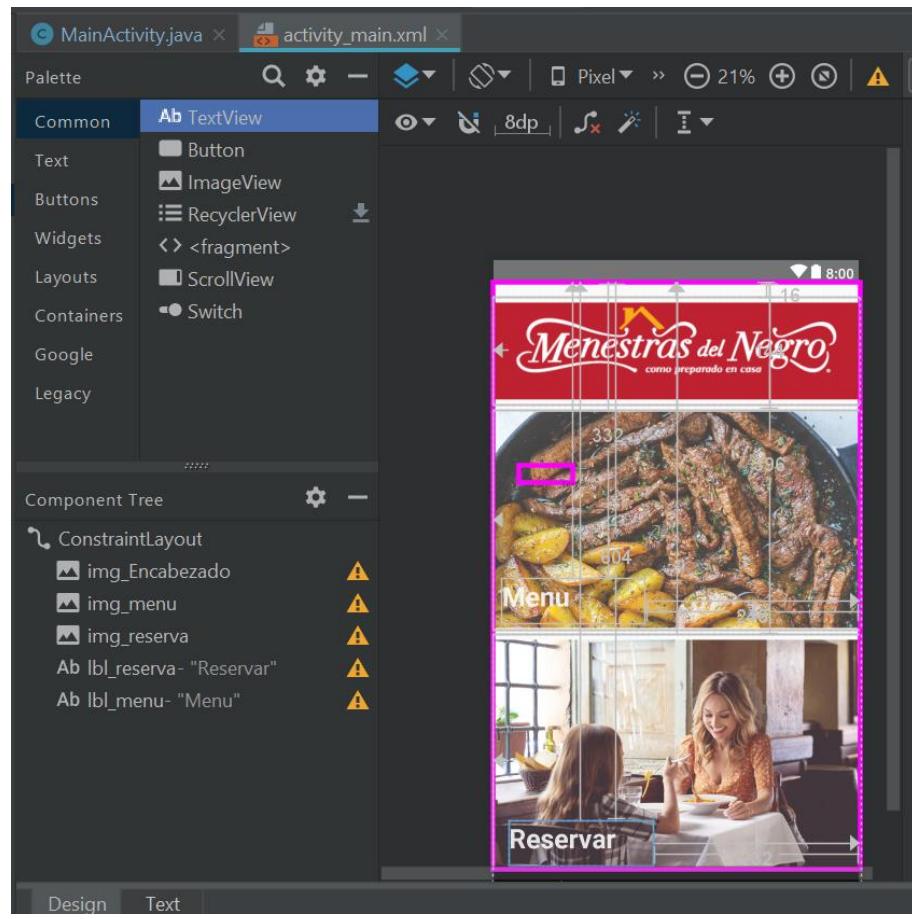
```
20 <ConstraintLayout ...>
21     android:id="@+id/img_menu"
22     android:layout_width="414dp"
23     android:layout_height="246dp"
24     android:layout_marginTop="144dp"
25     android:onClick="toMenu"
26     android:tint="#33FFFFFF"
27     app:layout_constraintStart_toStartOf="parent"
28     app:layout_constraintTop_toTopOf="parent"
29     app:srcCompat="@drawable/menu" />
30
31    <ImageView
32        android:id="@+id/img_reserva"
33        android:layout_width="414dp"
34        android:layout_height="283dp"
35        android:layout_marginTop="396dp"
36        android:background="#1AFFFFFF"
37        android:onClick="toReserva"
38        android:tint="#33FFFFFF"
39        app:layout_constraintStart_toStartOf="parent"
40        app:layout_constraintTop_toTopOf="parent"
41        app:srcCompat="@drawable/reserva" />
42
43    <TextView
44        android:id="@+id/lbl_reserva"
45        android:layout_width="162dp"
46        android:layout_height="51dp"
47        android:layout_marginTop="604dp"
48        android:layout_marginEnd="232dp"
49
50    androidx.constraintlayout.widget.ConstraintLayout > ImageView
```

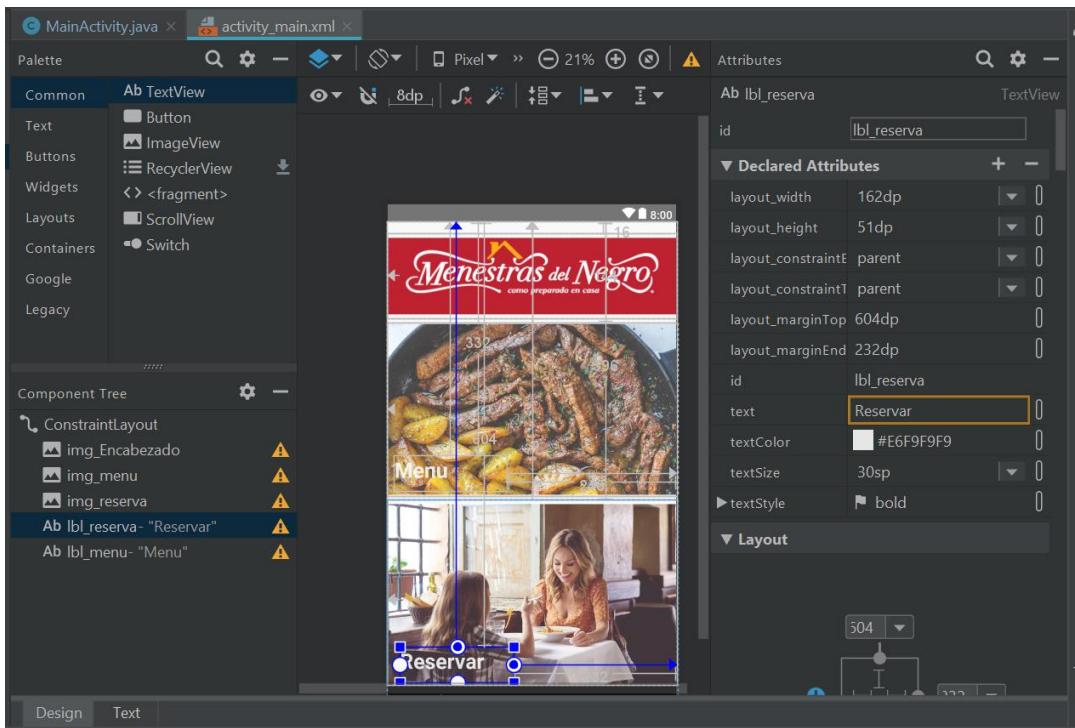
activity\_main.xml





3. En la pestaña de diseño del archivo xml se procede a insertar dos TextViews y se modifica su tamaño, color y se establece que sea texto en negrita. Se recomienda usar el formato “lbl\_nombreTextView” cuando se ingrese este tipo de componentes, para luego saber que componente se esta inicializando en la parte lógica.





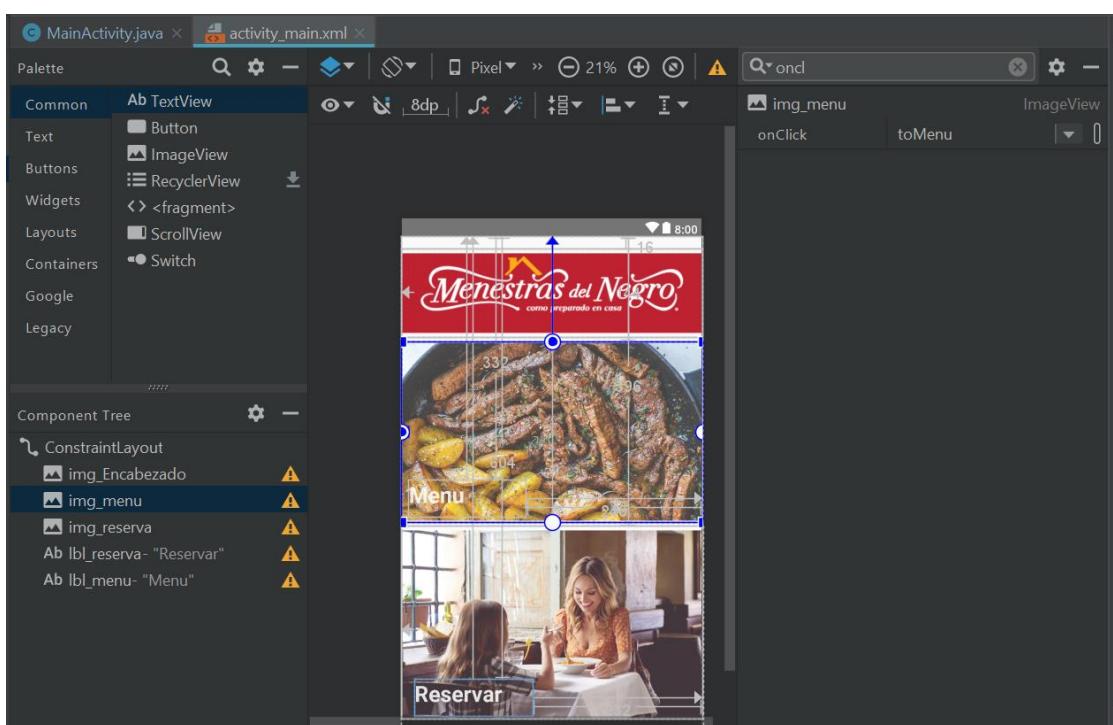
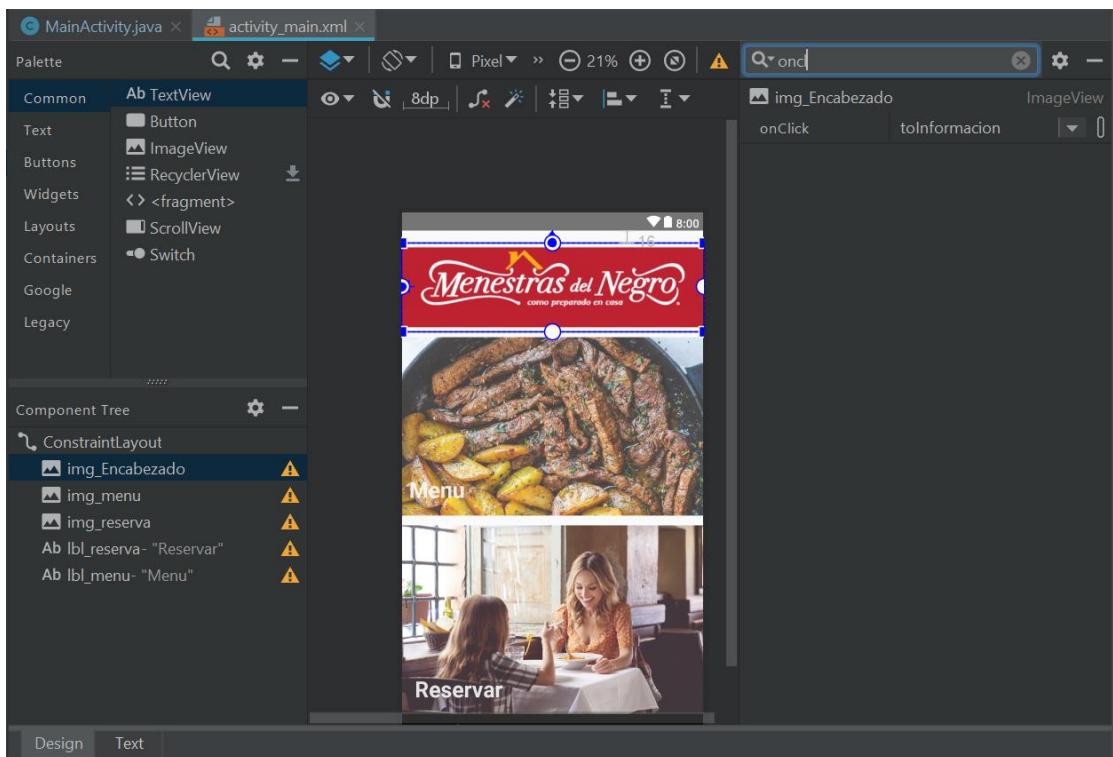
- Una vez terminada la parte de diseño nos dirigimos a la parte lógica del activity (archivo .java) donde se van a crear los métodos para acceder a otras activities de la app con objetos de tipo Intent, usando el siguiente código.

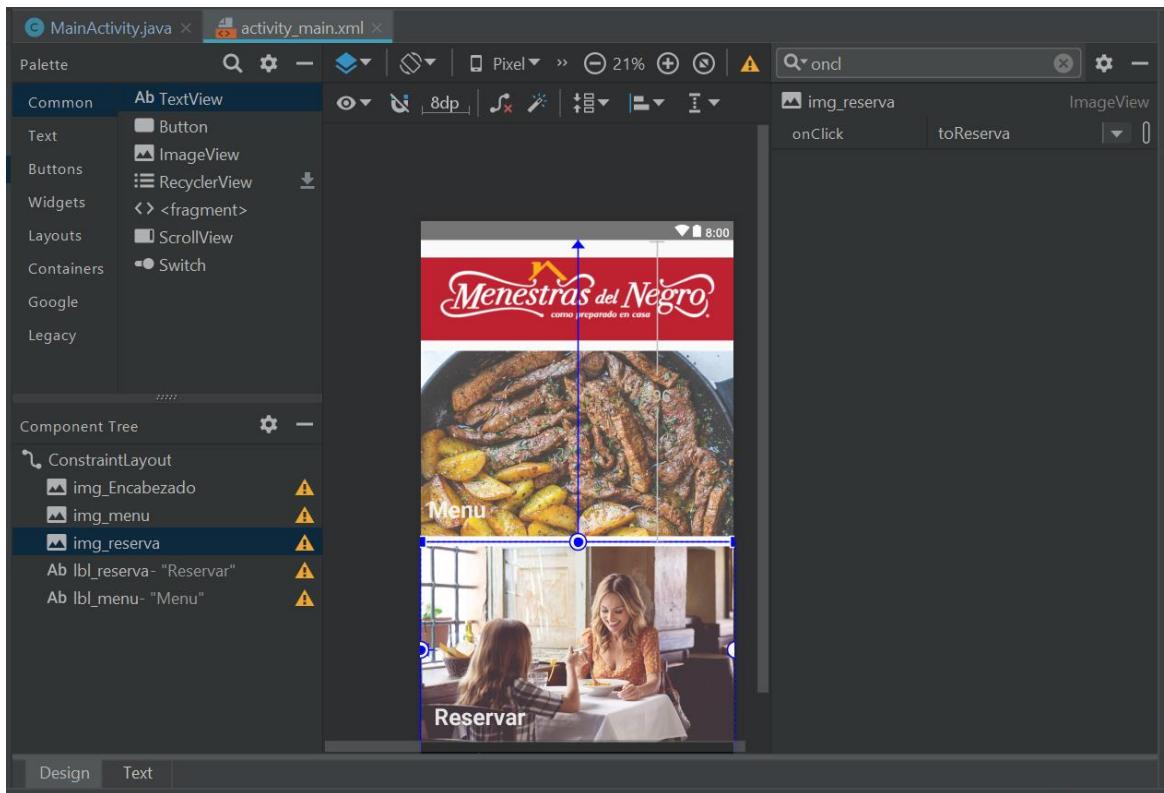
```

1 package com.example.pst_ta7_g4;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_main);
12     }
13
14     public void toInformacion(View view) {
15         Intent i = new Intent( mContext, Informacion.class );
16         startActivity(i);
17     }
18
19     public void toReserva(View view) {
20         Intent i = new Intent( mContext, reservaActivity.class );
21         startActivity(i);
22     }
23
24     public void toMenu(View view) {
25         Intent i = new Intent( mContext, Menu.class );
26         startActivity(i);
27     }
28
29 }
30
31
32
33
34

```

- Finalmente asignamos cada uno de los eventos creados a las imágenes de la parte de diseño del activity, utilizando el atributo onClick.

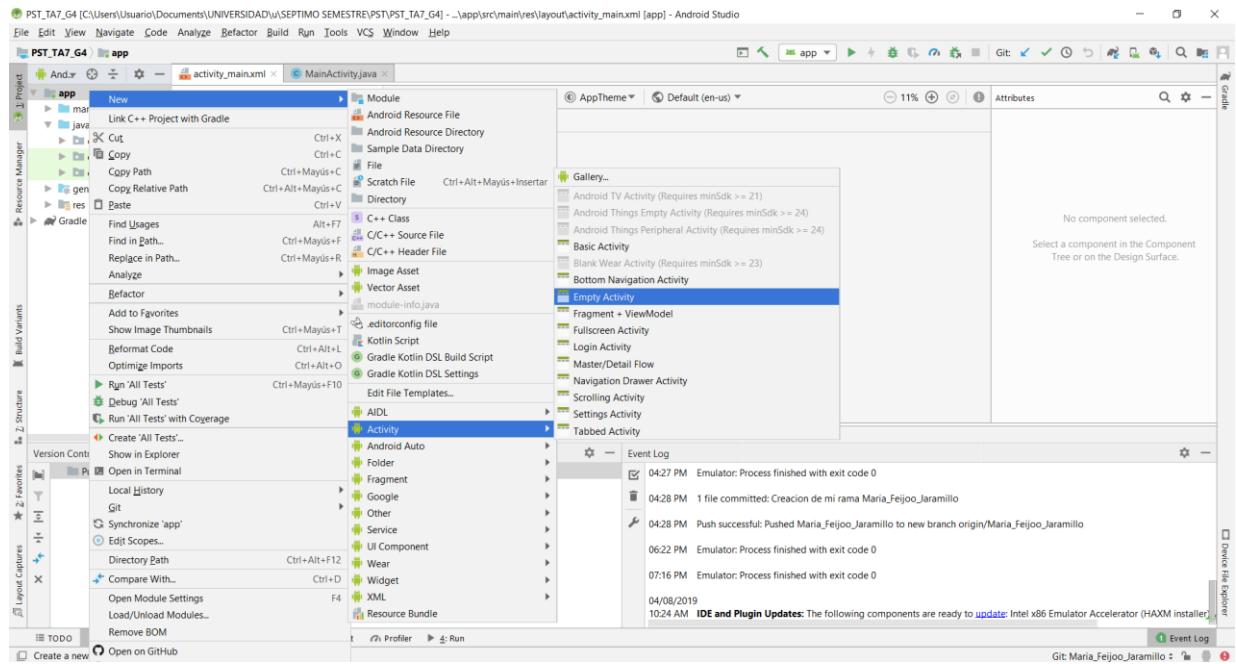
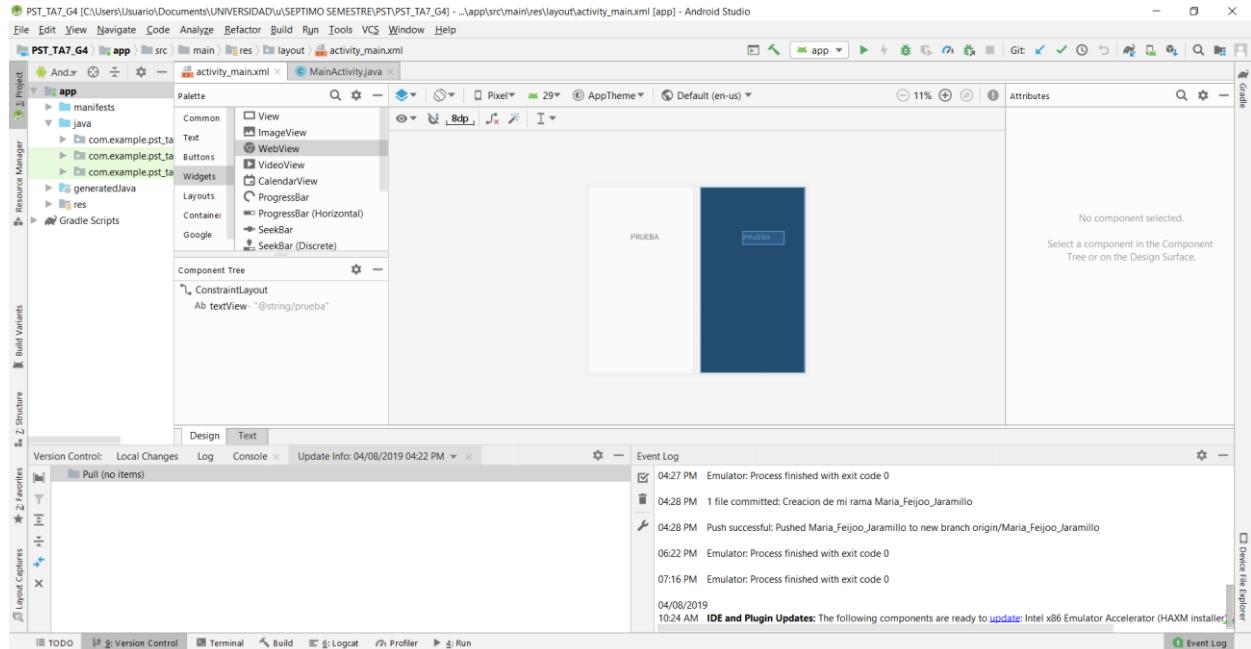




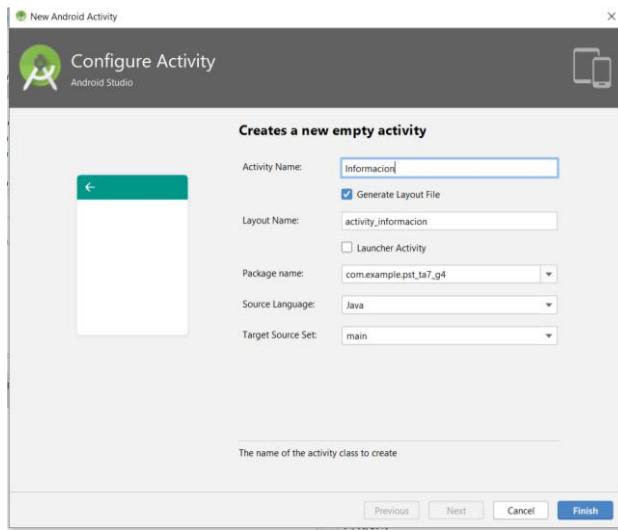
6. De esta forma al correr la aplicación se abrirá como ventana inicial el MainActivity y mediante toques a las diferentes imágenes se puede acceder a otras activities. Se utilizan ImageViews para así obtener un mejor diseño de presentación, pues un botón tiene muchas limitaciones en cuanto a su diseño.

## Implementación de la Actividad “Información”

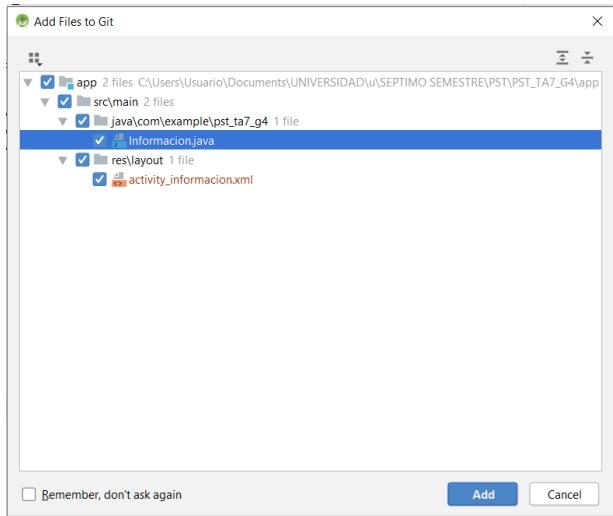
1. En primer lugar, abrimos el proyecto en el que vamos a trabajar, y procedemos a crear una nueva actividad, para lo cual nos vamos a app, se da clic derecho y se selecciona la opción New>Activity>Empty Activity.



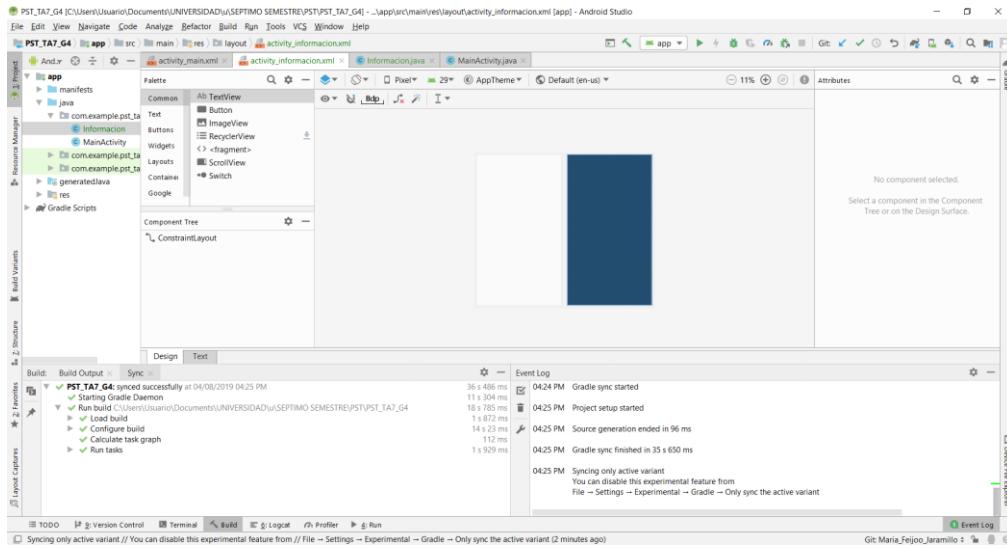
2. En la ventana que aparece a continuación, colocamos el nombre de la actividad, en este caso Información, y damos clic en “Finish”.



3. En la siguiente ventana, se da clic en “Add”, para agregar la actividad a nuestro proyecto.

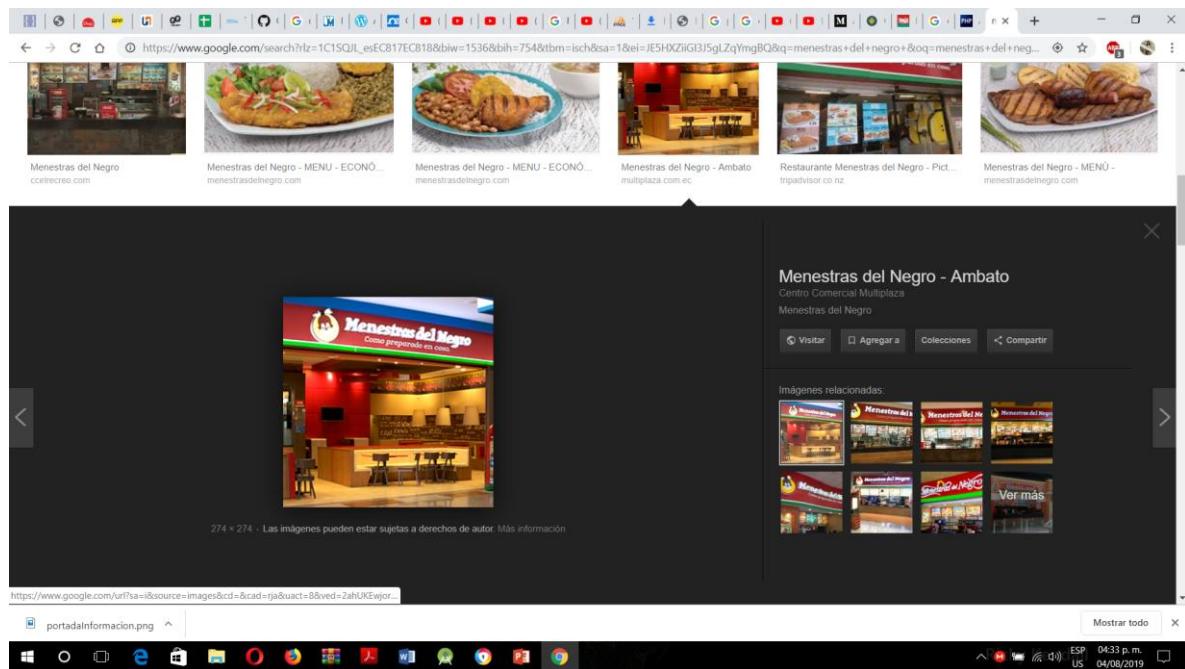


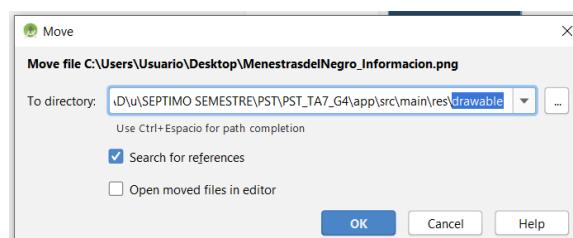
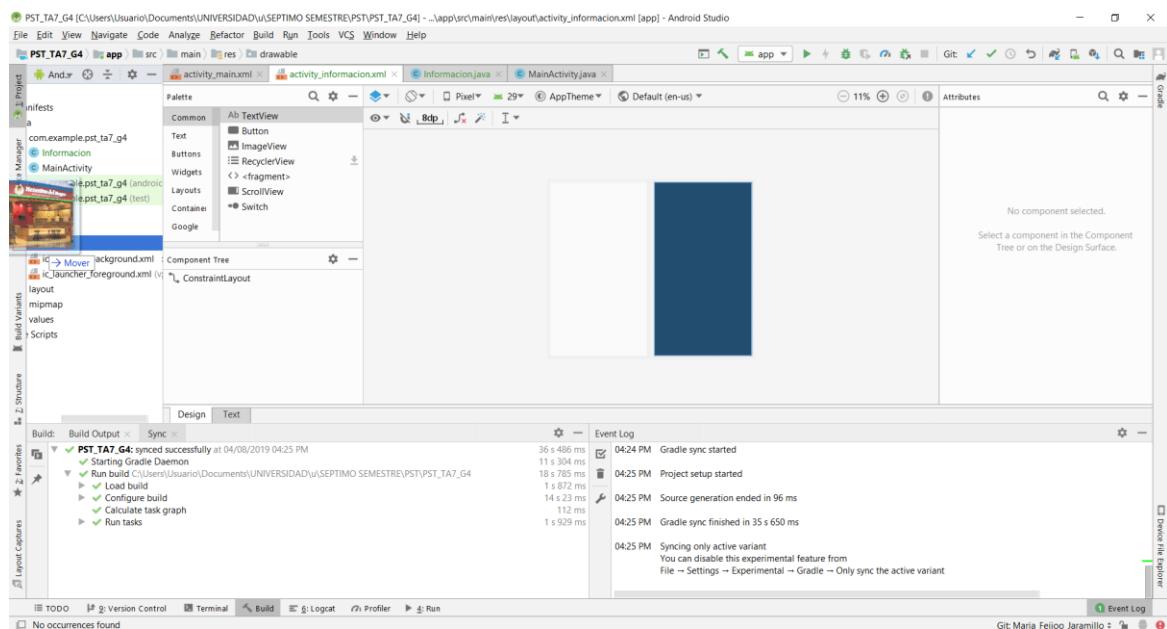
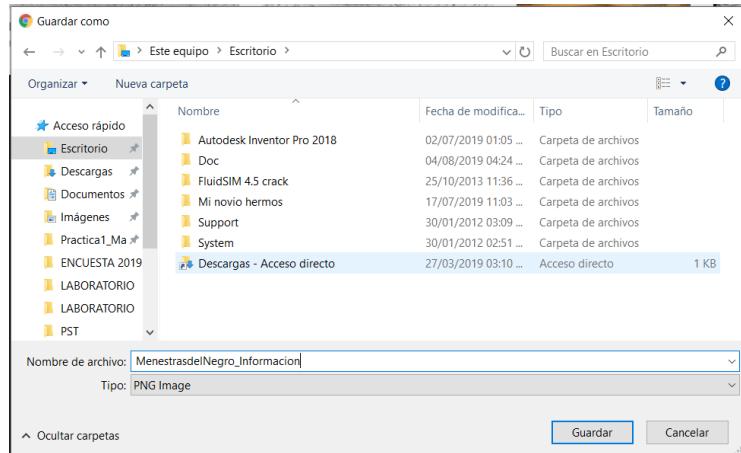
4. Seguidamente, ya creada nuestra actividad, abrimos el archivo `activity_informacion.xml`, para comenzar con el diseño de la actividad creada.

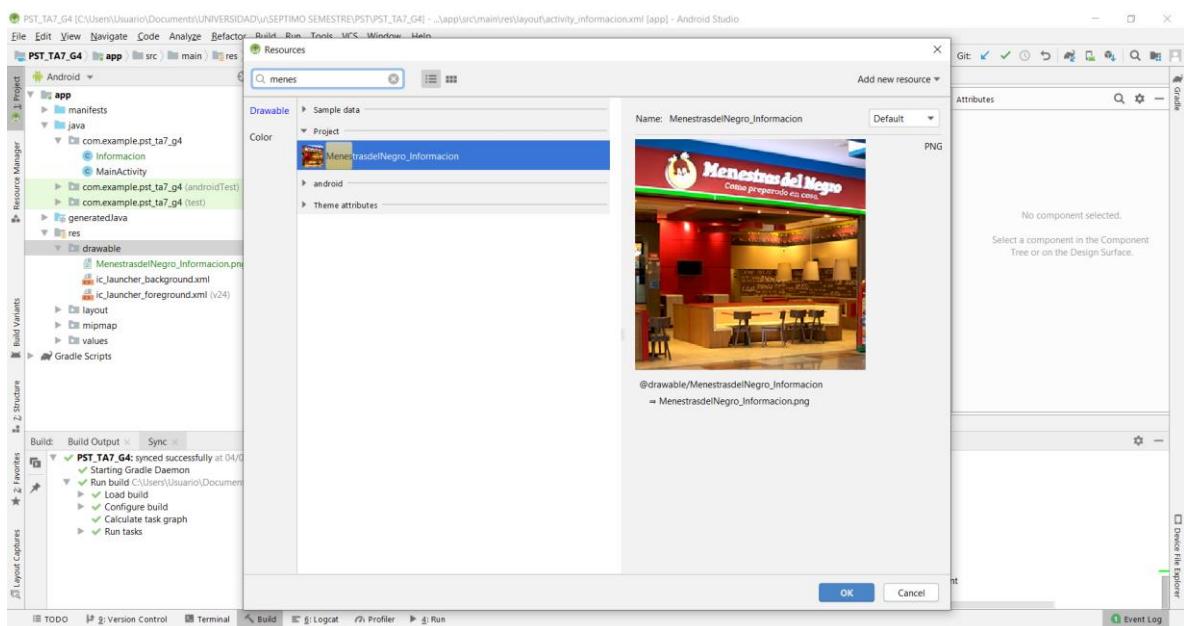
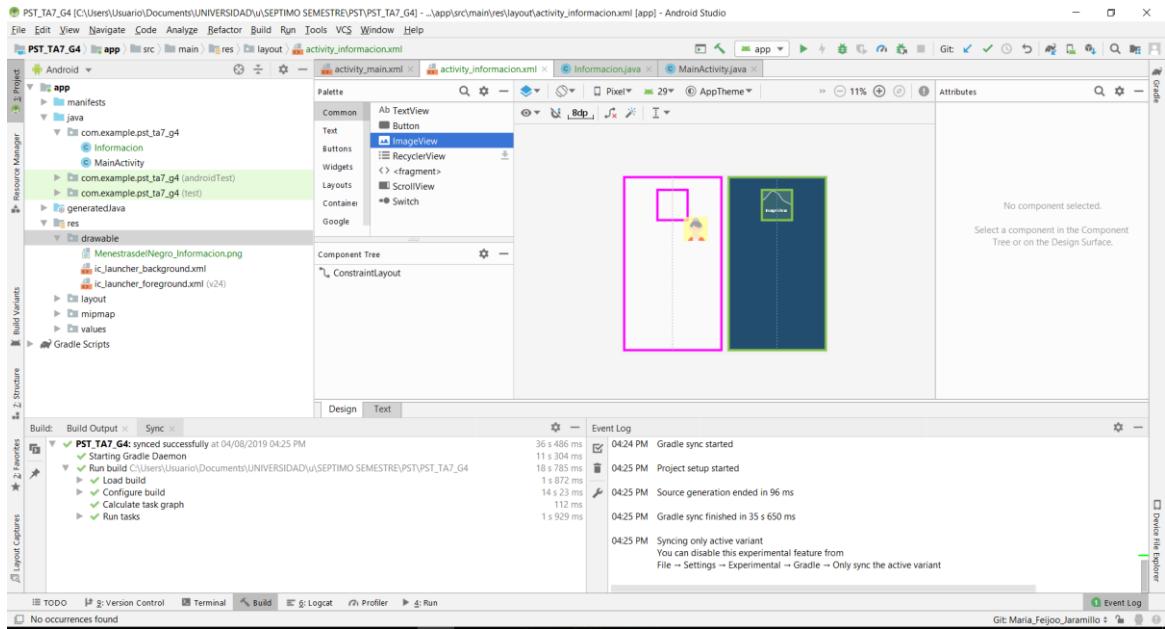


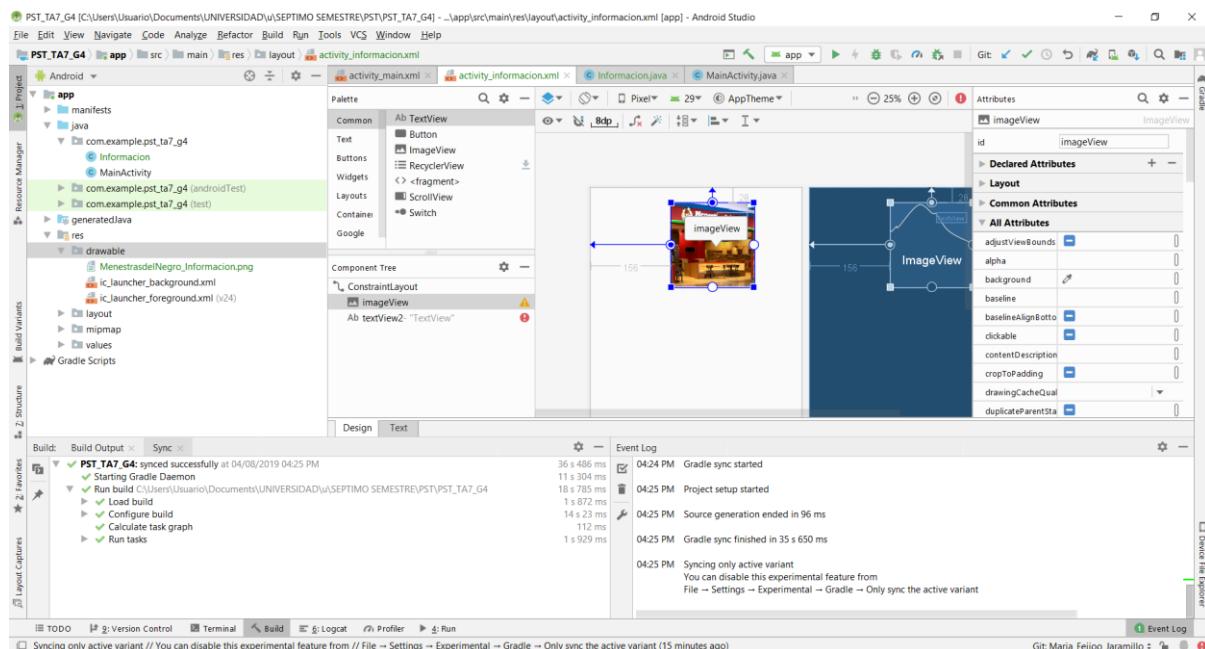
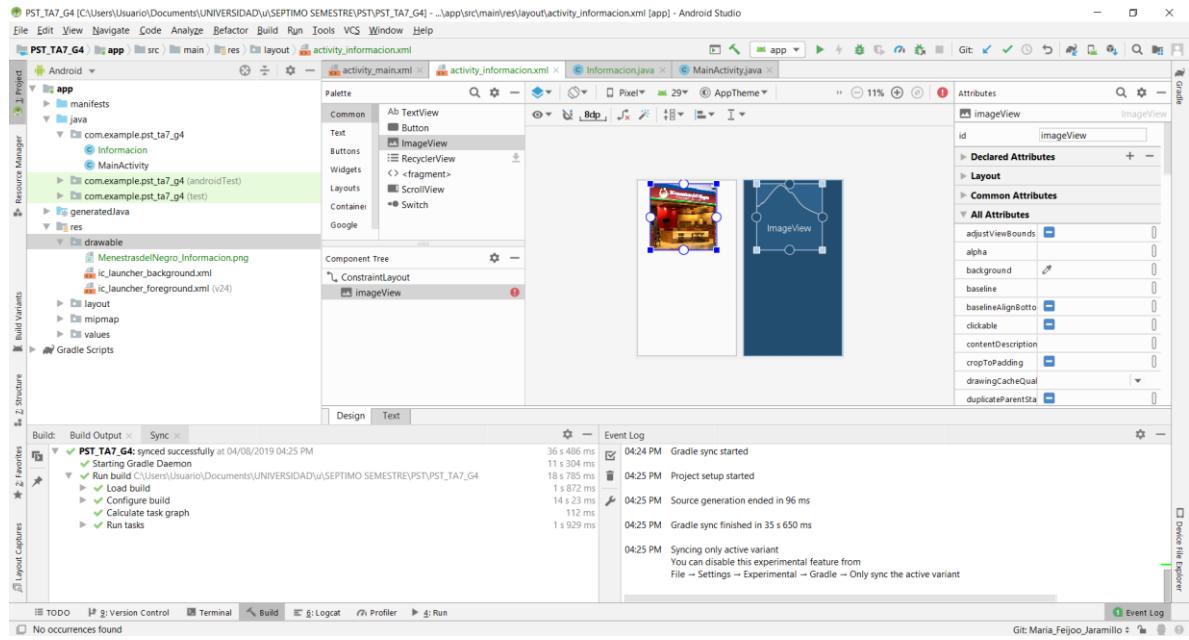
Para el diseño de la actividad Información se realizaron las siguientes configuraciones:

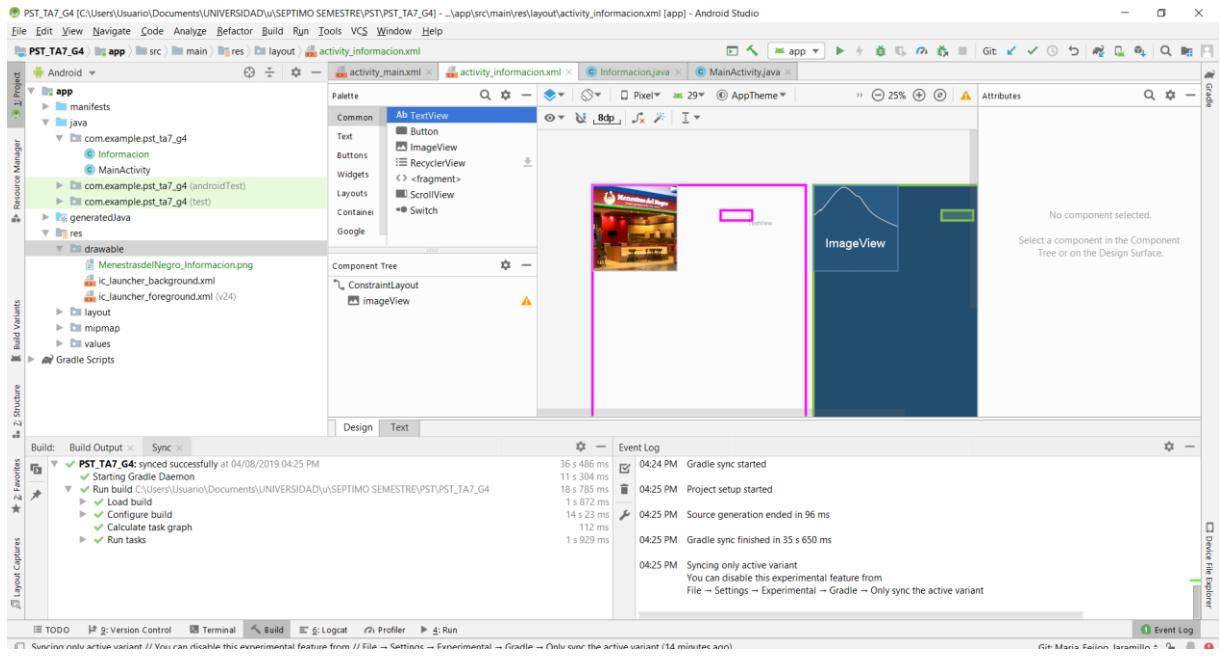
- Se agrego un ImageView, con la imagen del restaurante, para dar una mejor presentación. Para este proceso primero debemos descargar una imagen, seguidamente copiamos esta imagen en la carpeta res>drawable, y finalmente vamos al listado de componentes y seleccionamos un ImageView, aparecerá una nueva ventana en donde buscaremos el nombre de la imagen agregada y damos clic en “OK”. Finalmente, referimos nuestro componente al Layout, y lo colocamos en la esquina superior izquierda de la pantalla.



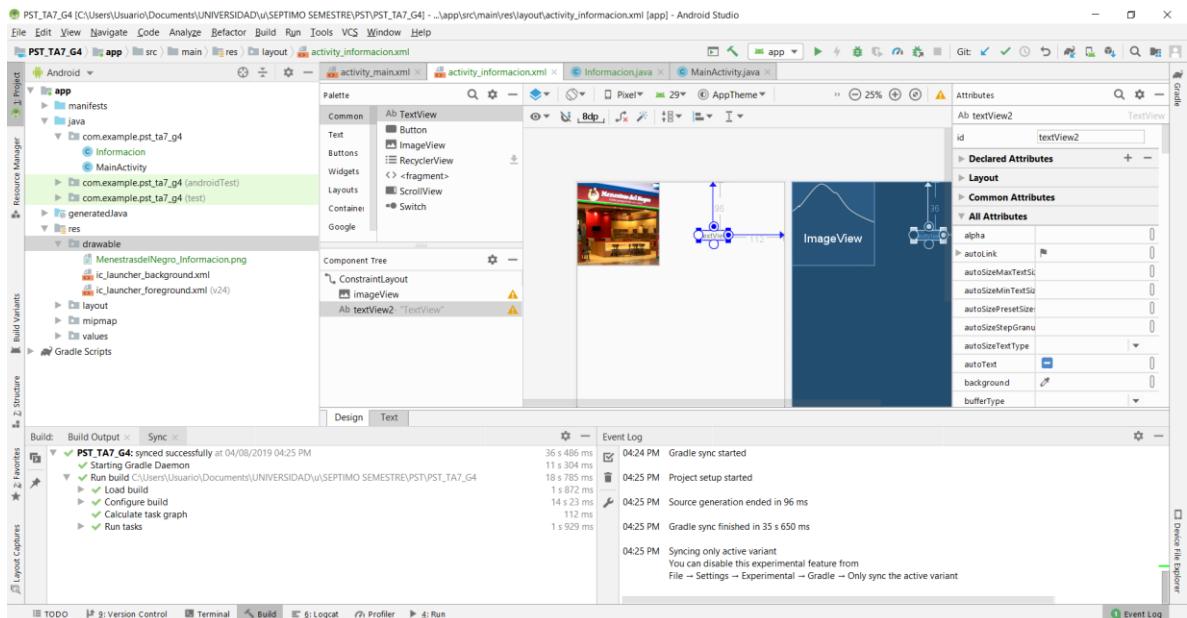


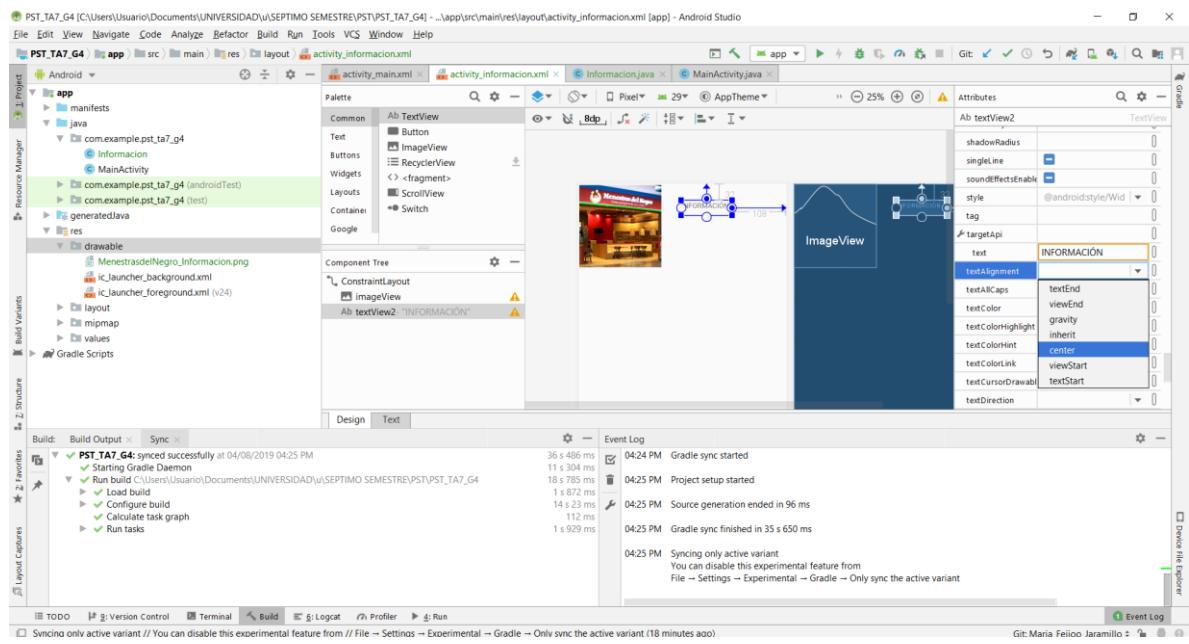
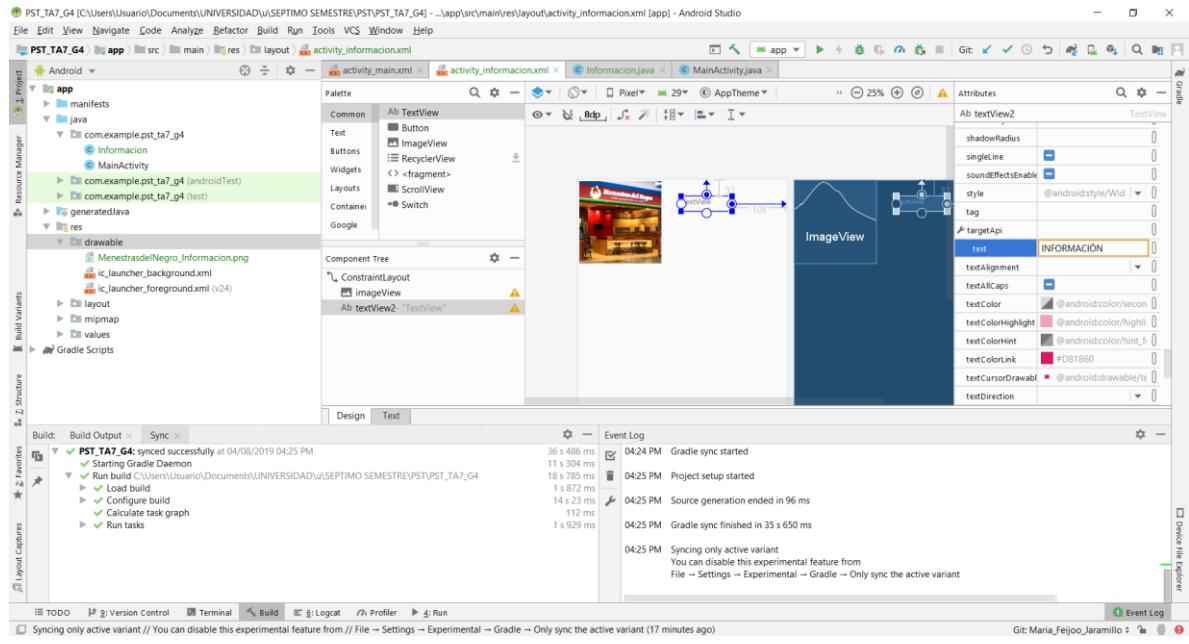


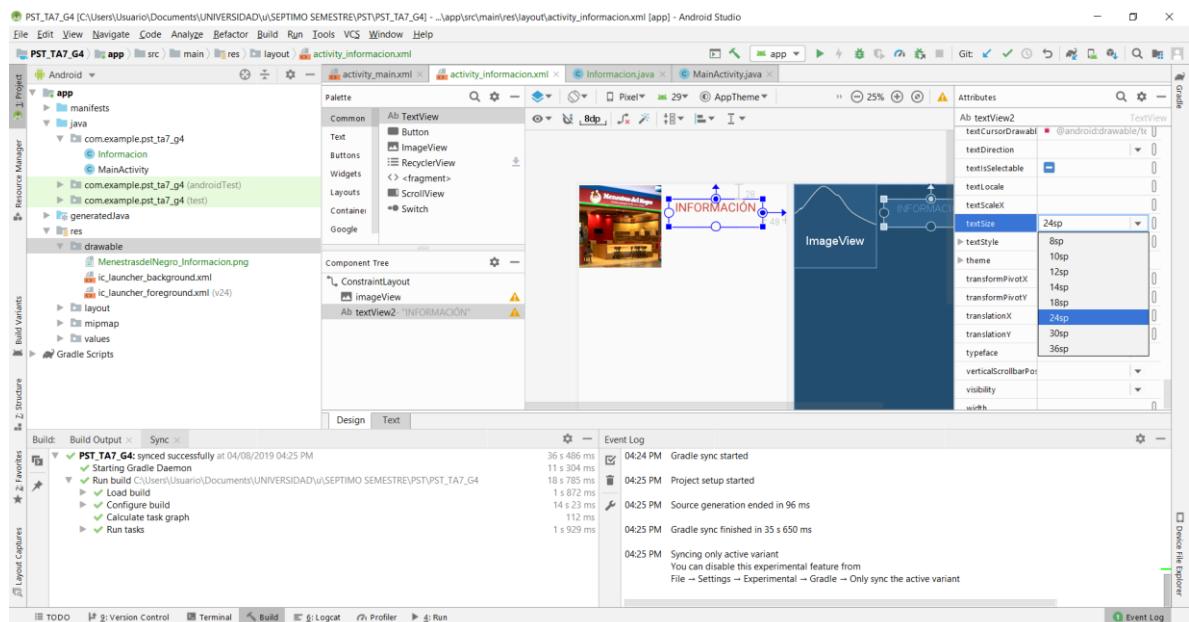
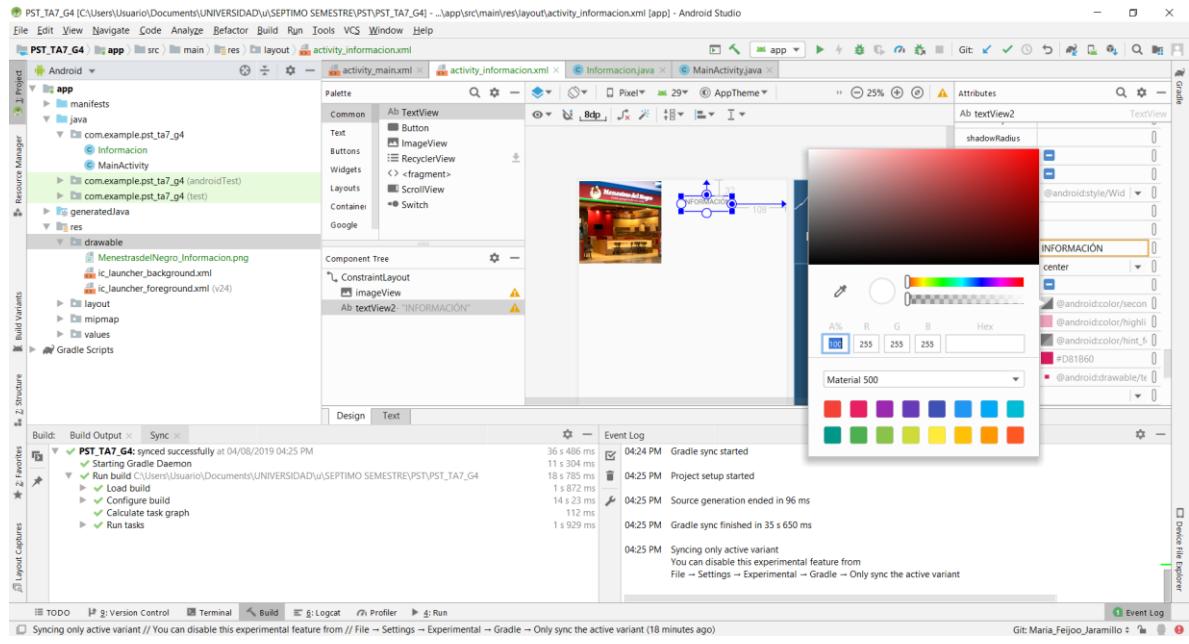


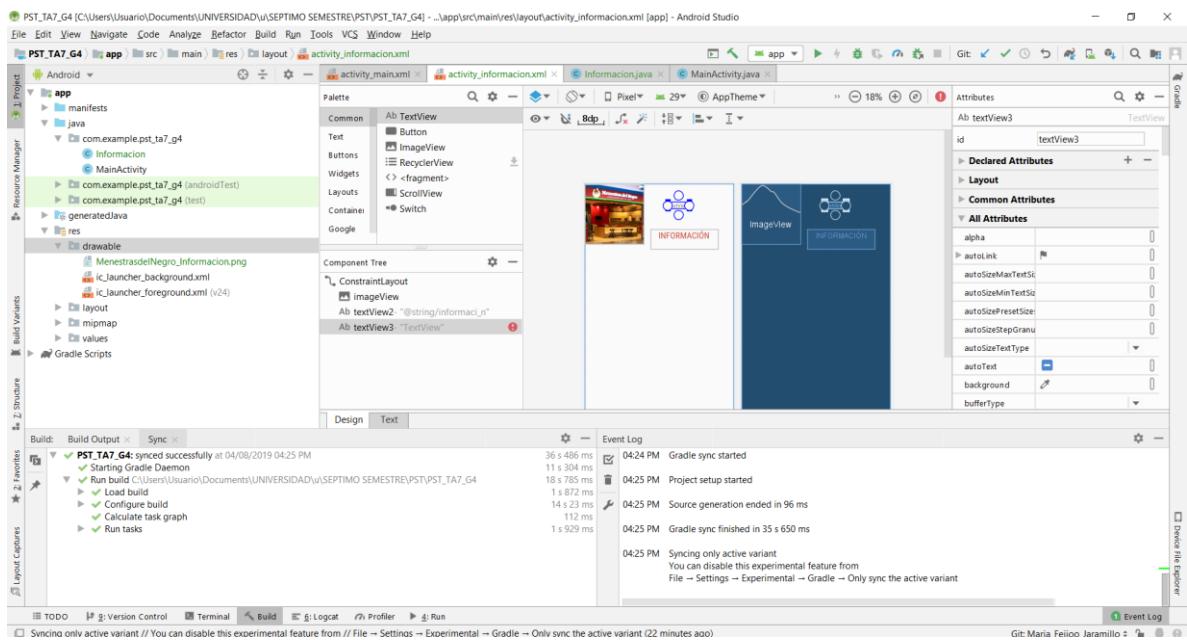
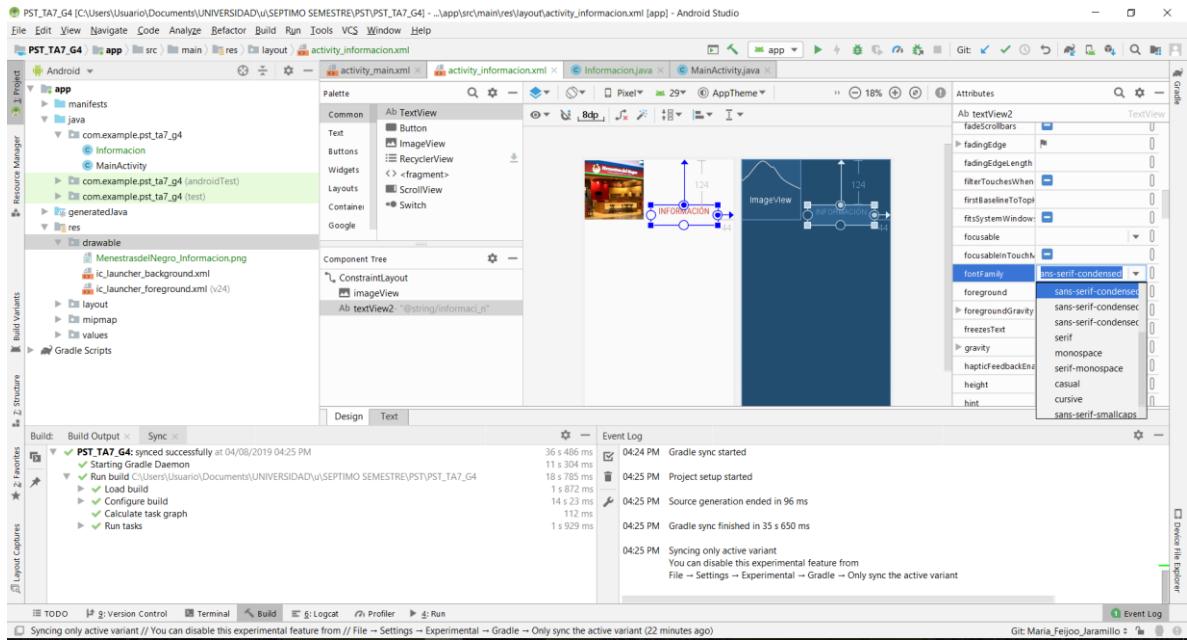


- b. Para colocar el título y todos los textos que queramos mostrar en la pantalla, utilizamos el componente TextView, modificando sus propiedades para cada caso, como se muestra a continuación. Para todos los textos se realiza un proceso muy similar al que se muestra a continuación, para el título, al cual modificamos su relleno, tipo, color y tamaño de letra.

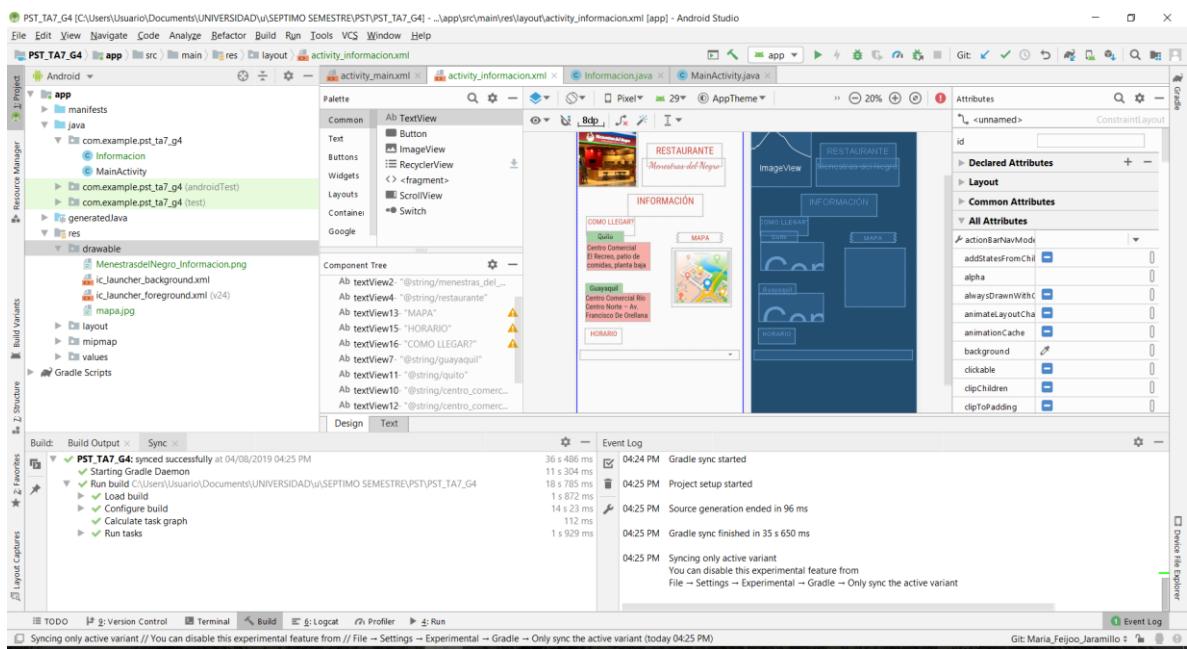
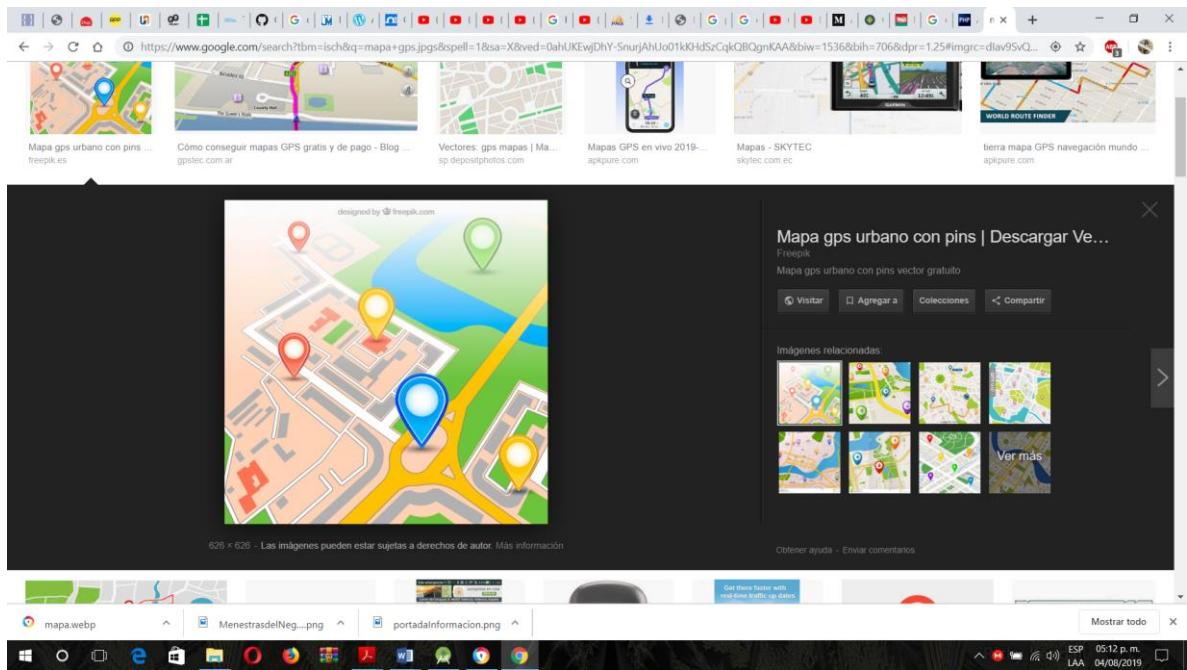




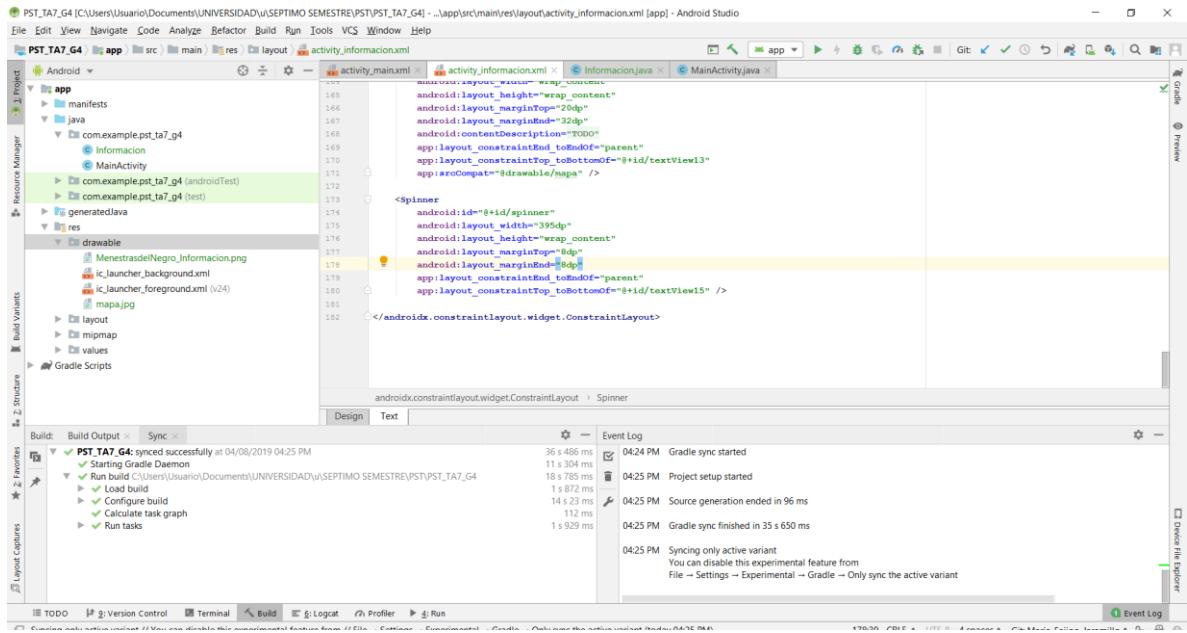




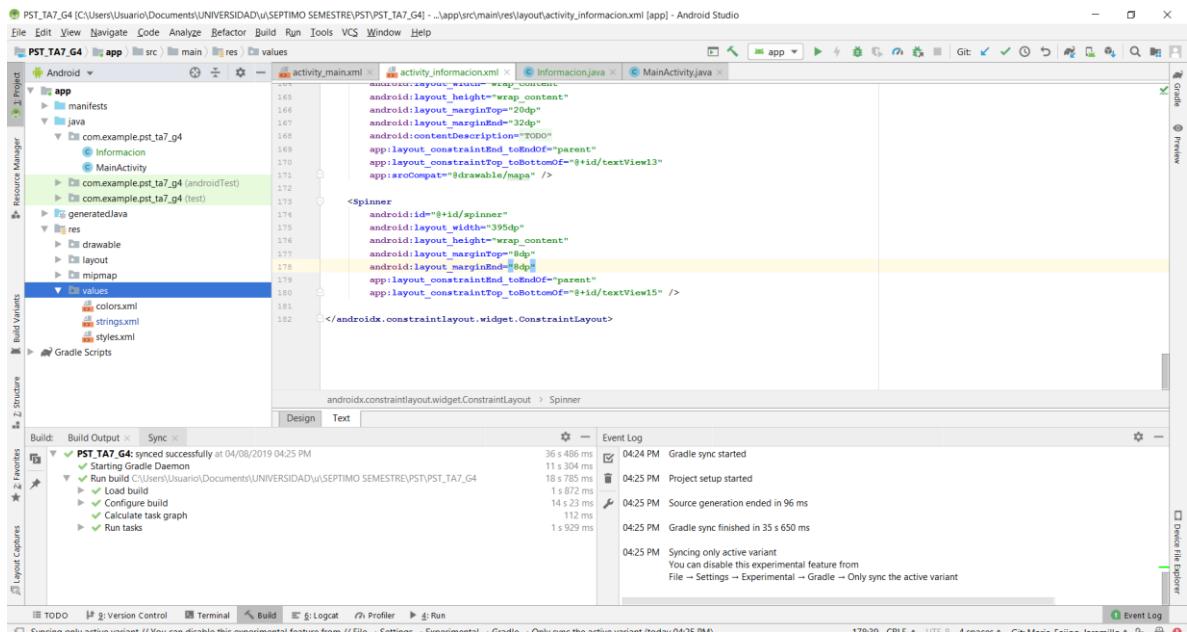
- c. Seguidamente, procedemos a agregar un ImageButton para que el usuario, pueda ingresar al mapa y ver la localizacion de los restaurantes en Ecuador. Al igual que como se realizo con el componente ImageView, descargamos una imagen y la copiamos en la carpeta res>drawables, luego seleccionamos el componente ImageButton y lo arrastramos a la pantalla, en la ventana que aparecerá seleccionamos la imagen que agregamos y damos clic en “OK”.

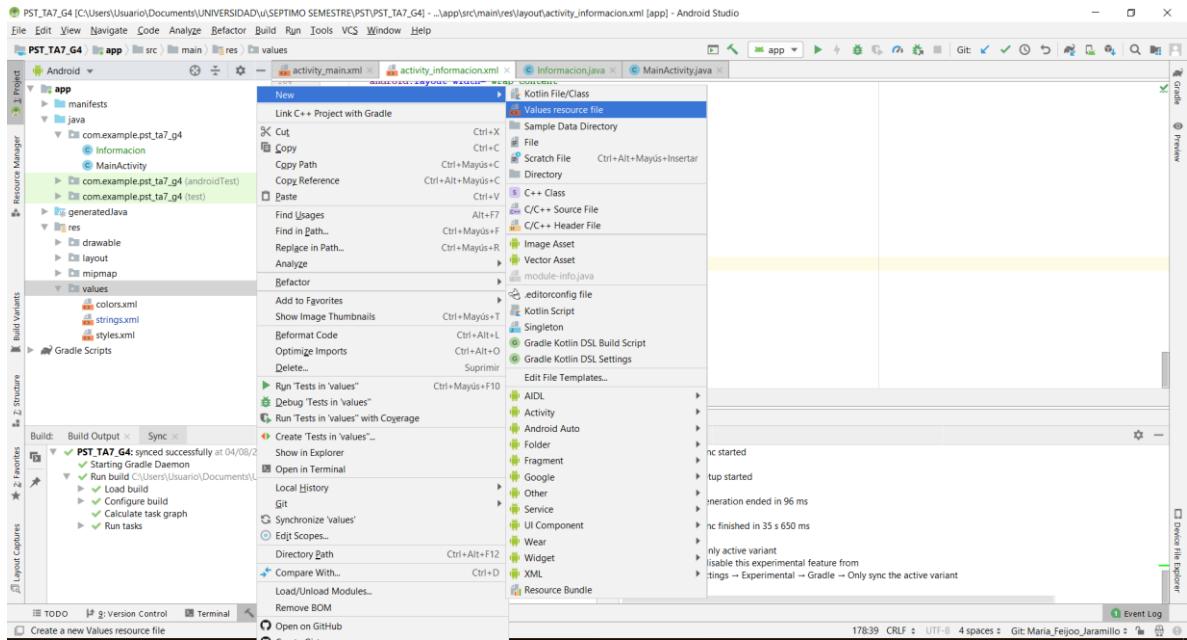


- d. Para poder mostrar todos los horarios de atención del restaurante, utilizamos el componente Spinner, el cual nos permite poder desplazar una lista con las opciones que queremos mostrar, en este caso los distintos horarios. Para lo cual se siguieron los pasos a continuación.
- i. Se agrega el componente ya sea arrastrándolo en Design, o definiéndolo manualmente en el archivo .xml, opción text.

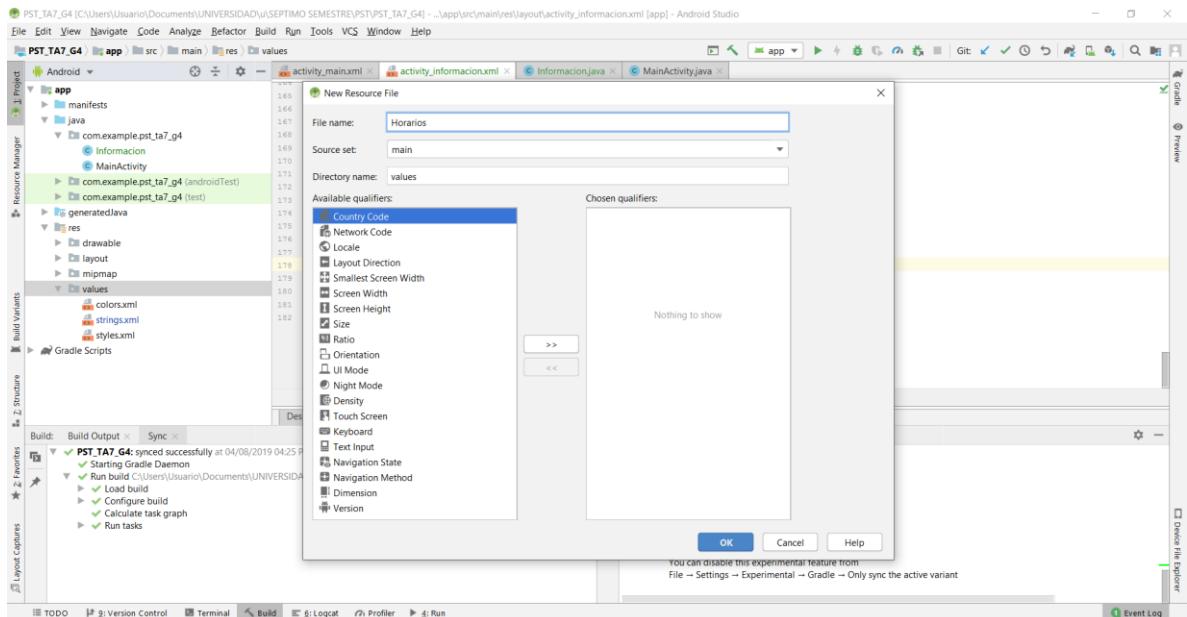


- ii. Para llenar nuestro spinner vamos a la carpeta de values y creamos un arreglo, para lo cual damos click derecho en la carpeta, y seleccionamos la opcion New>Values Resource File

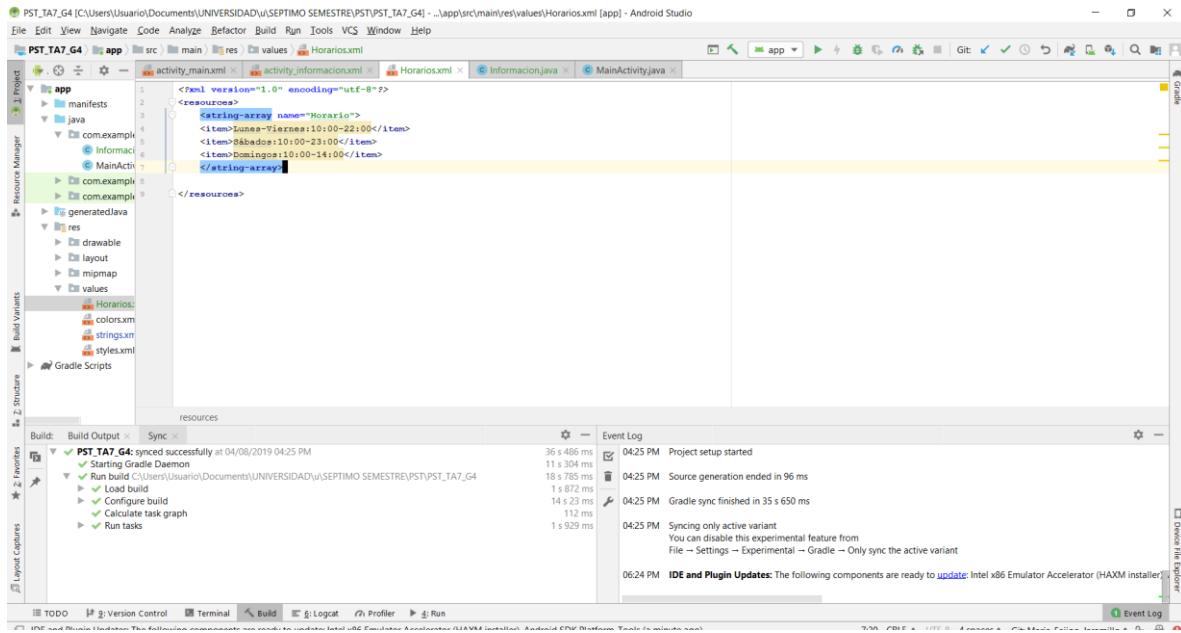




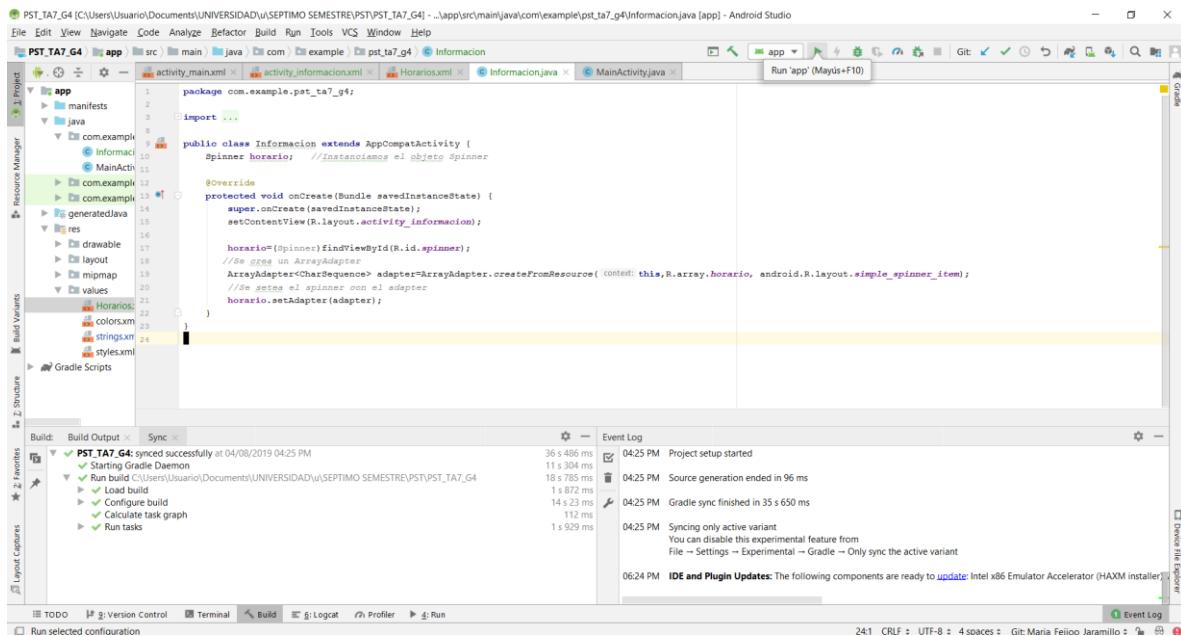
- iii. A continuación, aparecerá la siguiente ventana en donde debemos designar el nombre que queremos que tenga nuestro arreglo y damos clic en OK

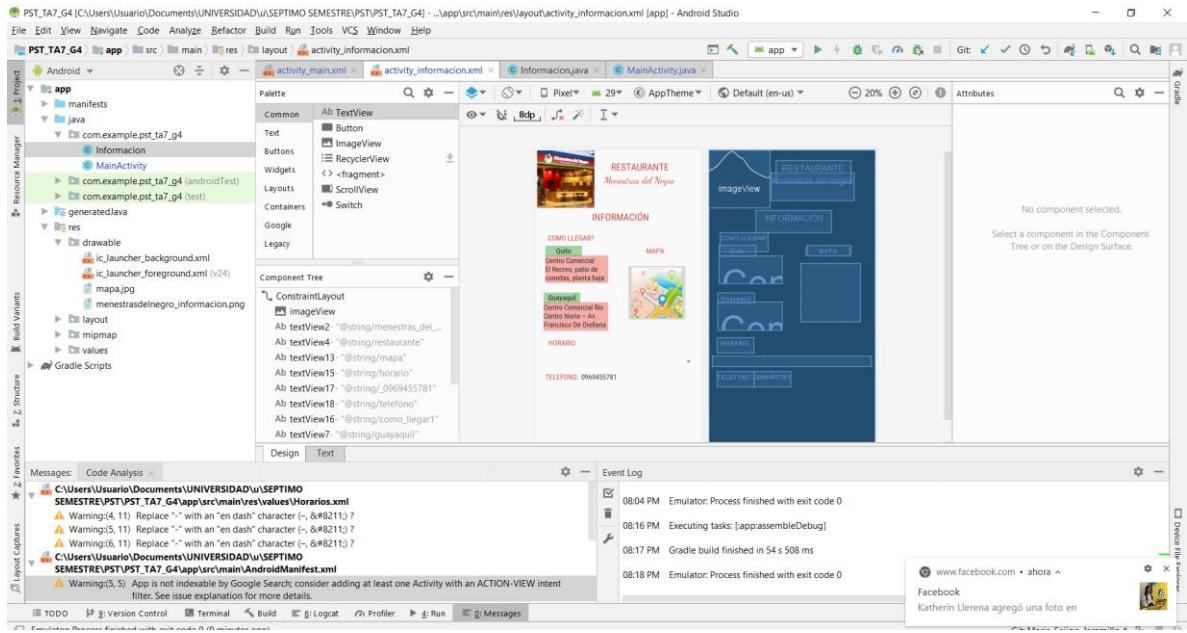


- iv. En el archivo creado, se procede a crear un arreglo de String donde definimos las opciones que queremos se muestren en nuestro Spinner.

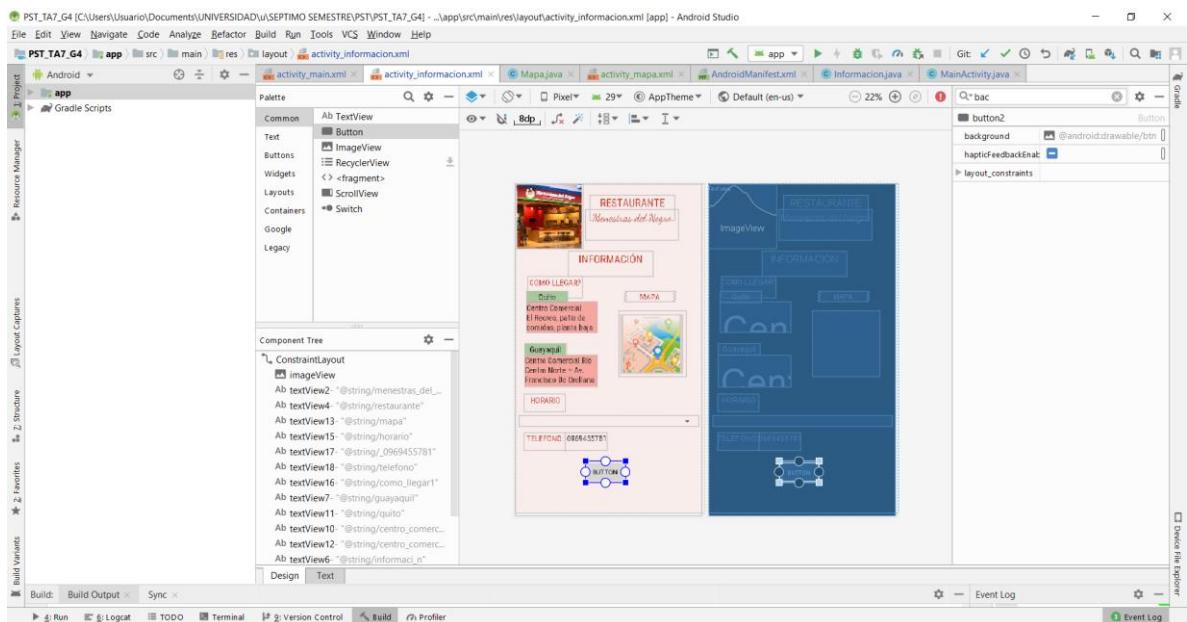


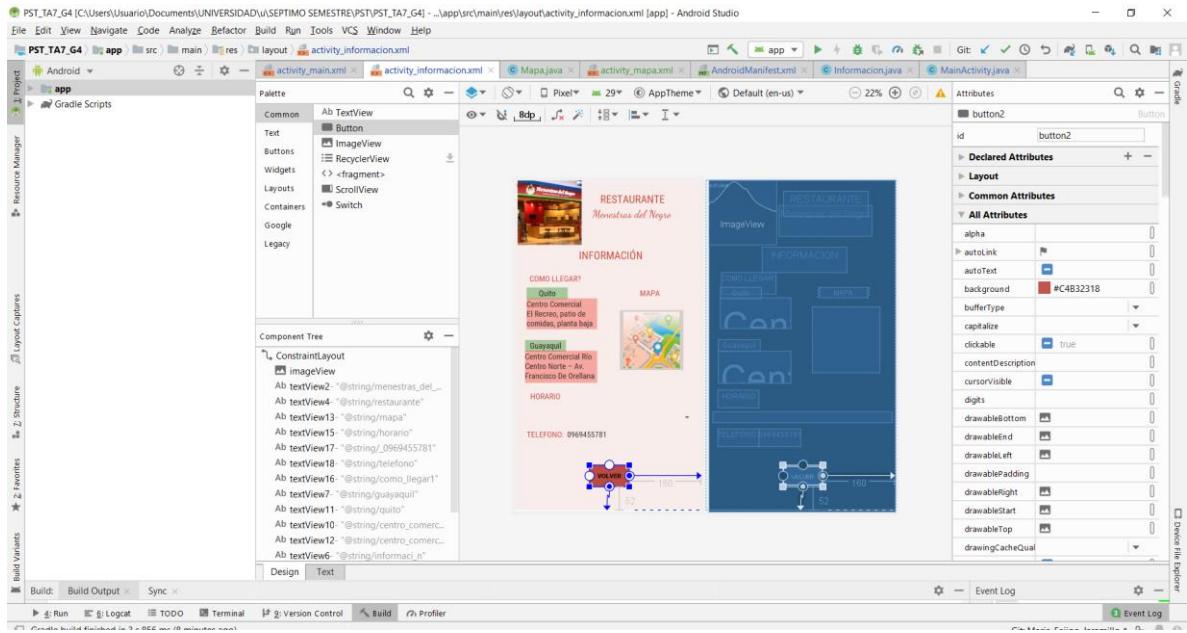
v. Luego vamos al archivo Información.java e instanciamos el elemento Spinner, además también creamos un ArrayAdapter, para llenar el Spinner y poder mostrarlo en la pantalla.





5. Continuando con el diseño, para poder regresar a la página inicial, agregamos un nuevo botón, con el componente Button, modificando algunas de sus características, definiendo su función en la clase Informacion.java y agregando en la propiedad Onclick la función creada, este botón nos permite regresar a la pagina principal de la app.





```

PST_TA7_G4 [C:\Users\Usuario\Documents\UNIVERSIDAD\SEMESTRE\PST\PST_TA7_G4] - app\src\main\java\com\example\pst_ta7_g4\Informacion.java - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
PST_TA7_G4 app src main layout activity_informacion.xml
activity_main.xml activity_informacion.xml Mapa.java activity_mapa.xml AndroidManifest.xml Informacion.java MainActivity.java
Project app Gradle Scripts
Resource Manager
Palette Common Ab TextView Text Button ImageView Buttons Recycler View Widgets ScrollView Layouts Containers Google Legacy
Component Tree ConstraintLayout imageView
Ab textView2 : @string/menestras_del_
Ab textView4 : @string/restaurante"
Ab textView13 : @string/mapa"
Ab textView15 : @string/horario"
Ab textView17 : @string/_099455781"
Ab textView18 : @string/telefono"
Ab textView16 : @string/como_llegar"
Ab textView7 : @string/guayaquil"
Ab textView11 : @string/quito"
Ab textView10 : @string/centro_comerc_
Ab textView12 : @string/centro_comerc_
Ab textView6 : @string/informaci_n"
Build: Build Output Sync x
Run Log TODO Terminal Version Control Build Profiler
Create build finished in 3 s 856 ms (10 minutes ago)
Event Log

```

```

package com.example.pst_ta7_g4;
import android.app.appcompat.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Spinner;

public class Informacion extends AppCompatActivity {
    Spinner horario; //Instanciamos el objeto spinner

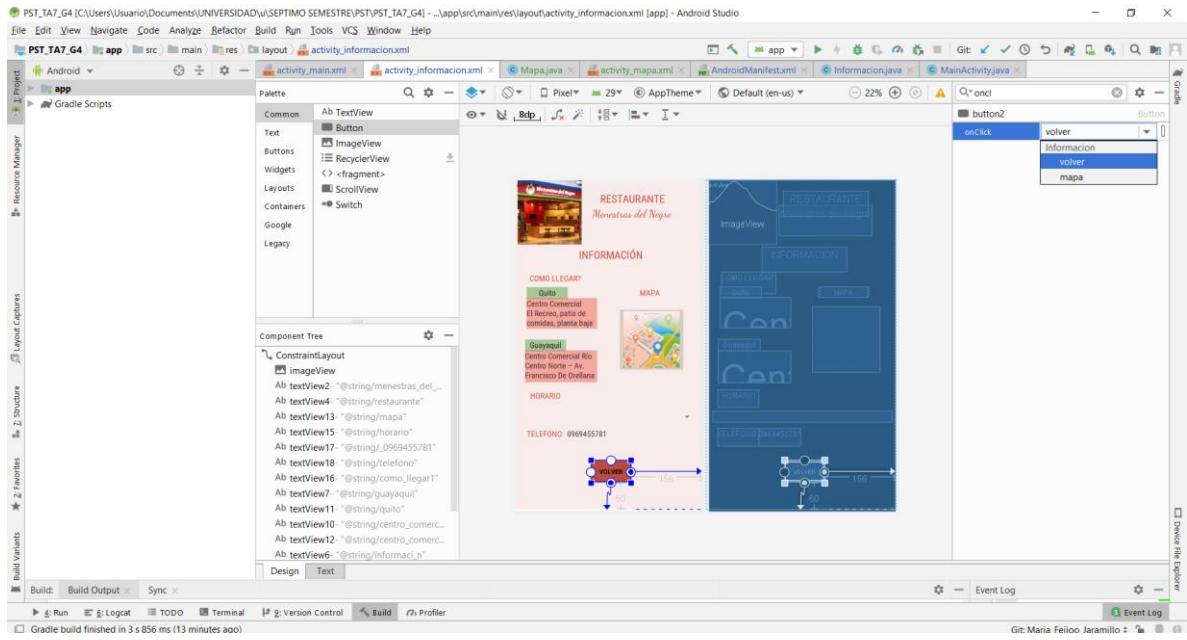
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_informacion);

        horario=(Spinner)findViewById(R.id.spinner);
        //Se crea un ArrayAdapter
        ArrayAdapter<CharSequence> adapter=ArrayAdapter.createFromResource(getApplicationContext(), R.array.horario, android.R.layout.simple_spinner_item);
        //Se pone el spinner con el adapter
        horario.setAdapter(adapter);
    }

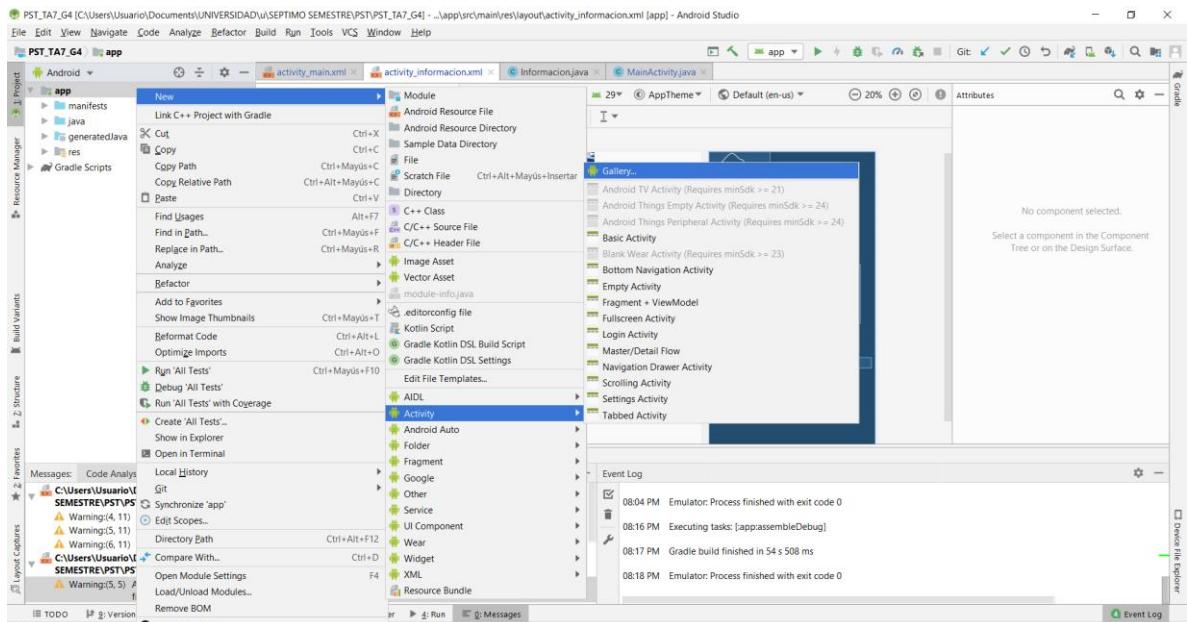
    public void volver(View view) {
        Intent i = new Intent(getApplicationContext(), MainActivity.class);
        startActivity(i);
    }

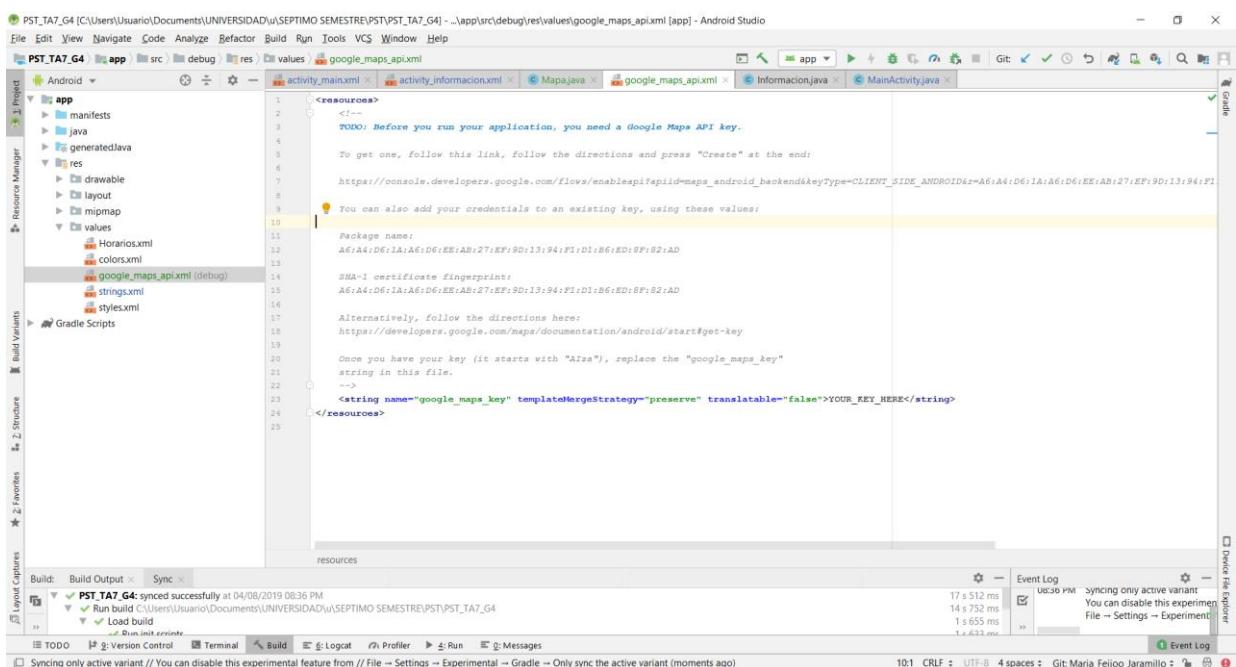
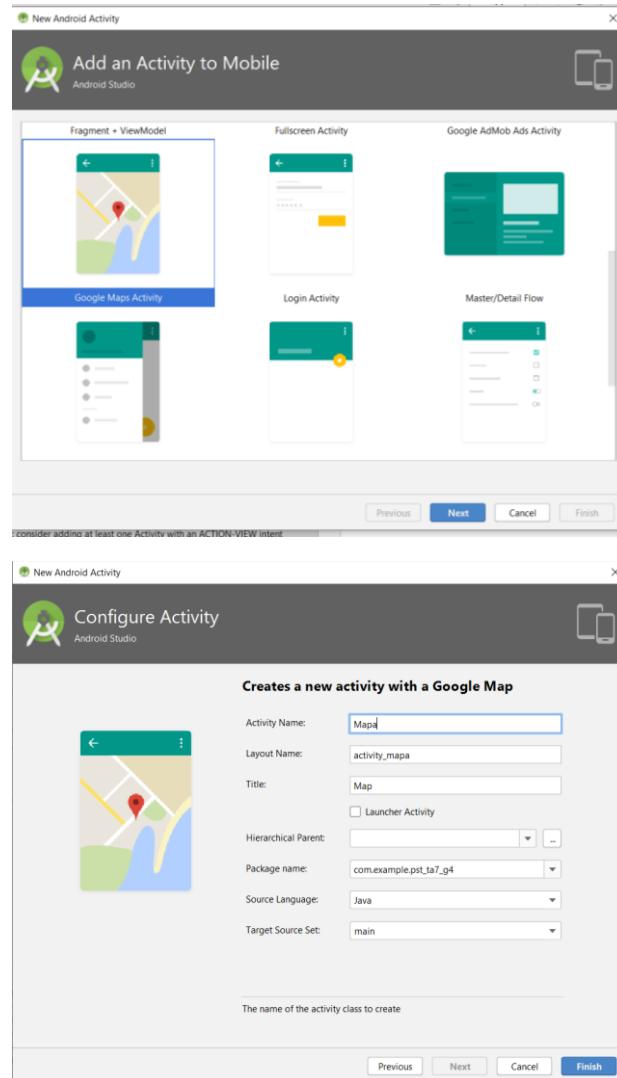
    public void mapa(View view) {
        Intent i = new Intent(getApplicationContext(), Mapa.class);
        startActivity(i);
    }
}

```

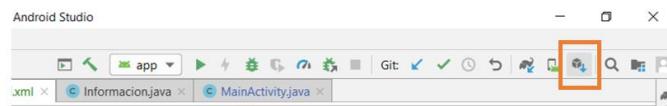


6. Ya finalizado el diseño de nuestra pantalla, procedemos a la configuración de nuestro ImageButton, para lo cual creamos una nueva actividad en donde mostraremos la ubicación de los restaurantes.
  - i. En primer lugar, creamos la nueva Activity, para lo cual lo hacemos igual que el caso anterior, con la diferencia que ahora seleccionaremos una Activity tipo Google Maps Activity.

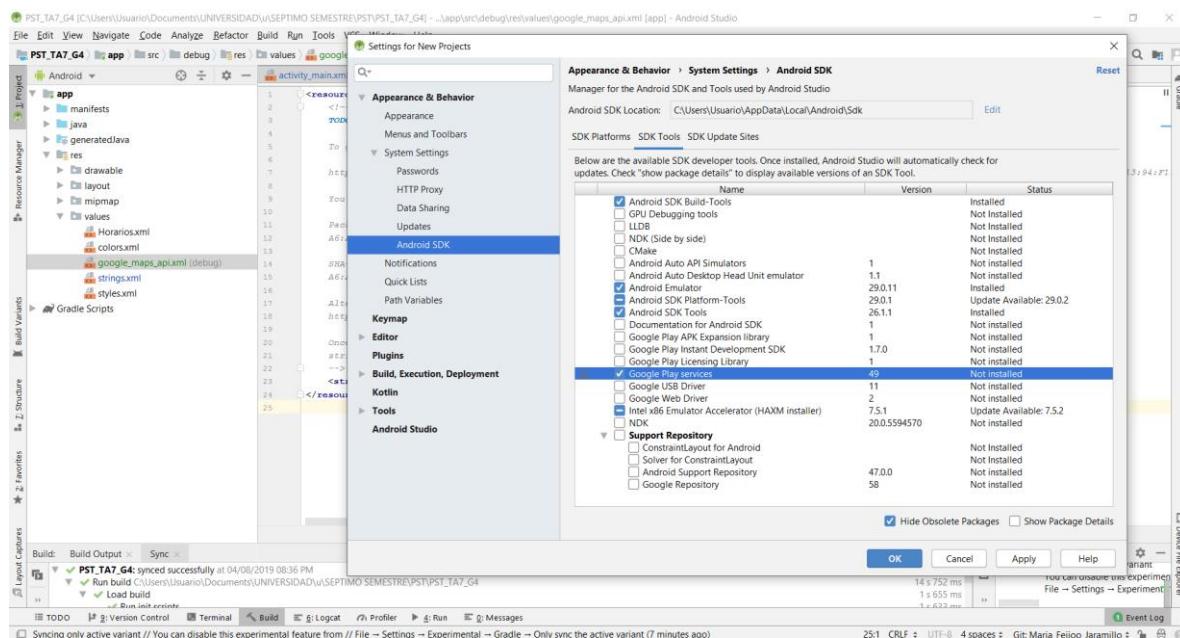
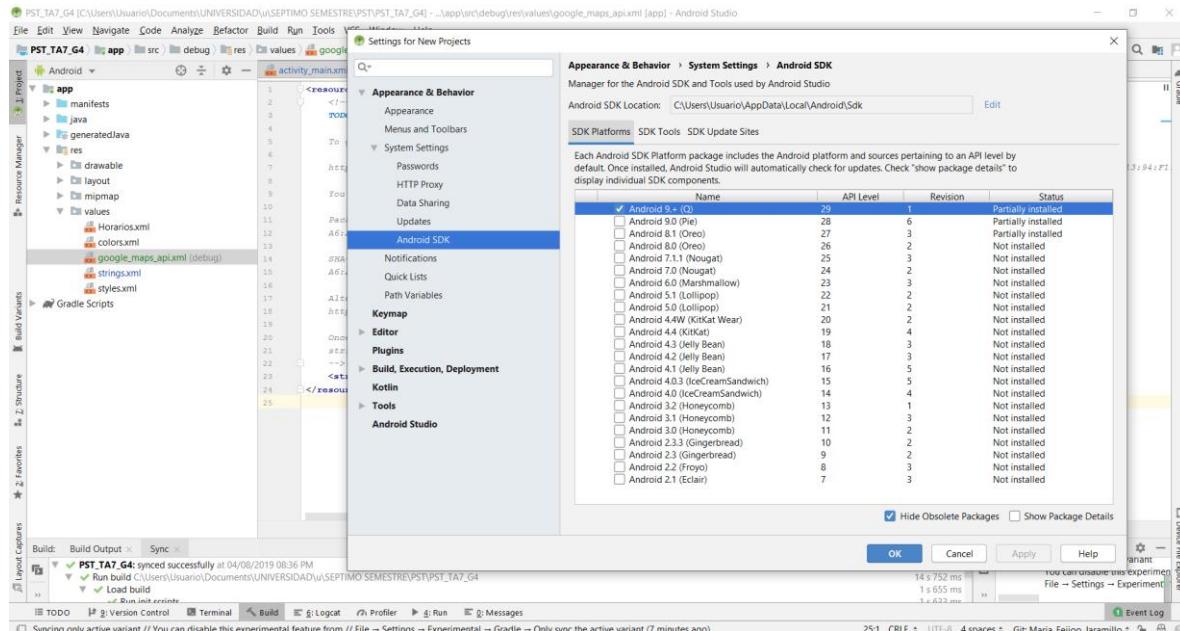


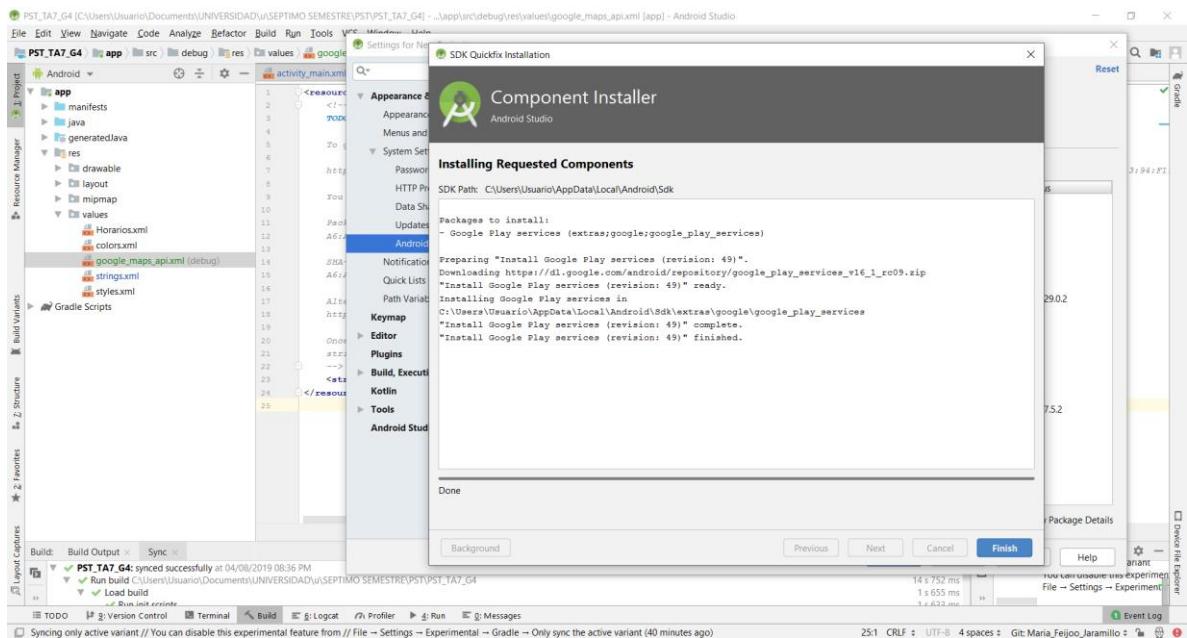
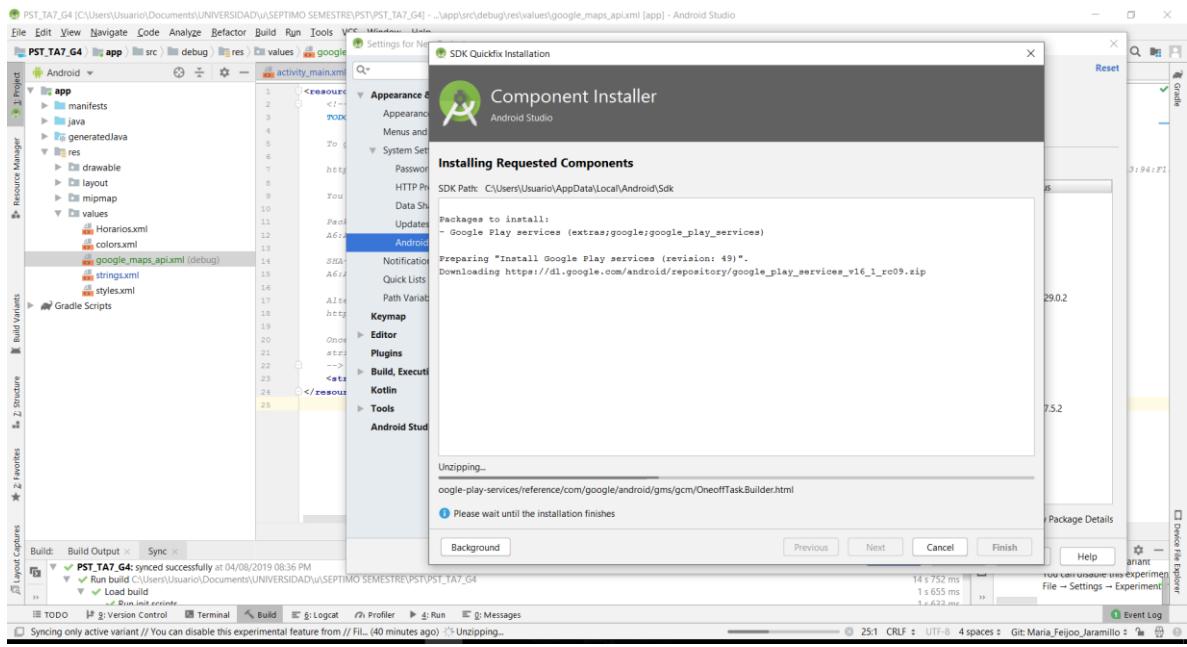


- ii. Ya creada la actividad, debemos realizar distintas configuraciones para asegurar que el mapa pueda parecer en la pantalla, primero vamos a SDK Manager, colocado en la parte superior derecha de la pantalla, y damos clic.

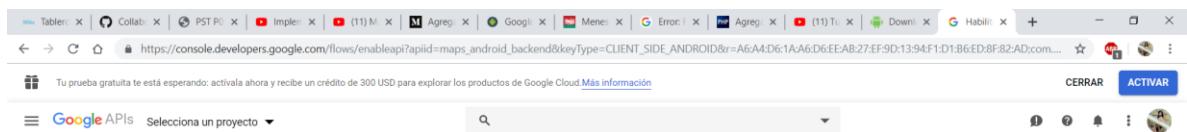
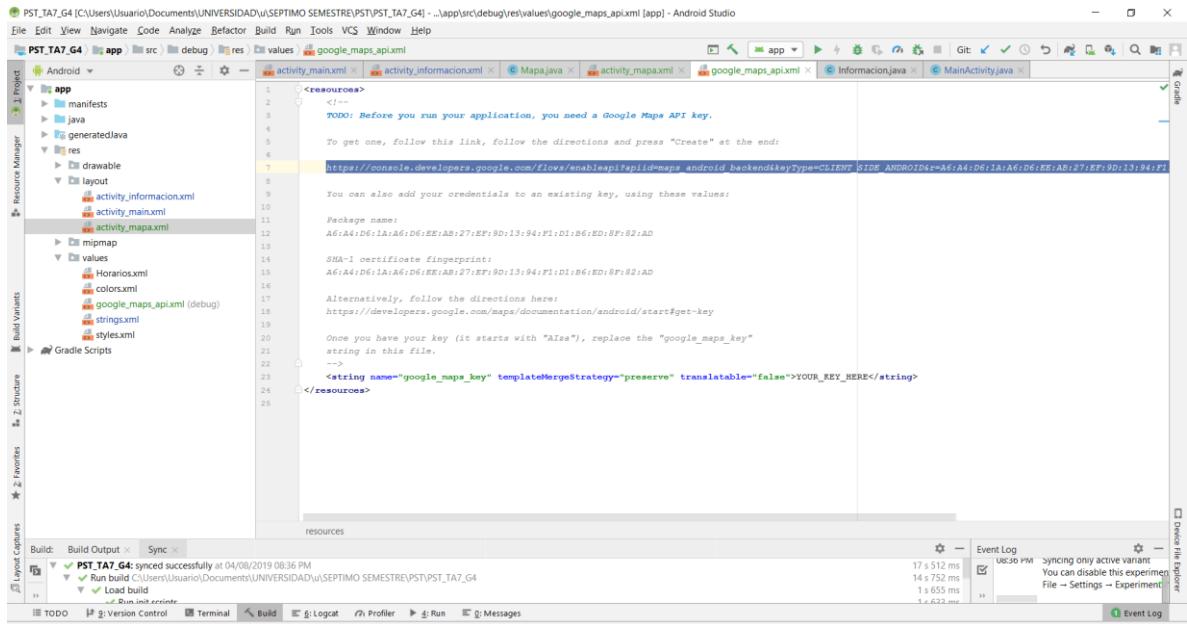


- iii. A continuación, aparecerá una nueva ventana donde debemos ir a la opción SDK Tools y seleccionar la opción Google Play Services para instalarla, luego damos clic en “OK”, apareciendo una nueva ventana donde se inicia el proceso de descarga, que tardará algunos minutos, finalizado este proceso se da clic en “Finish”.





- vi.** Luego vamos al archivo .xml de actividad del mapa creada y copiamos el link que aparece en nuestro navegador. Este nos enviara a la página de Google's API, para que podamos obtener nuestra clave API, que necesitamos para poder ejecutar la aplicación.



- vii. Siguiendo los pasos que nos detallan, obtenemos la Clave API para nuestro proyecto, la cual debemos copiar y pegarla en el archivo .xml en la sección del código que dice “YOUR\_KEY\_HERE”.

Tu prueba gratuita te está esperando: activala ahora y recibe un crédito de 300 USD para explorar los productos de Google Cloud. [Más información](#)

CERRAR ACTIVAR

Google APIs Selecciona un proyecto ▾

La API está habilitada

El proyecto se ha creado y Undefined parameter - API\_NAMES se ha habilitado.

A continuación, tienes que crear una clave de API para llamar a la API.

[Crear clave de API](#)

Mostrar todo

Logo.png mapa (1).jpg mapa.jpg mapa.jpg mapa.png mapa.jpg mapa.jpg mapa.jpg Mostrar todo

Tu prueba gratuita te está esperando: activala ahora y recibe un crédito de 300 USD para explorar los productos de Google Cloud. [Más información](#)

CERRAR ACTIVAR

Google APIs My Project ▾

API APIs y servicios Credenciales

Panel de control Credenciales Pantalla de consentimiento de OAuth Verificación de dominio

Biblioteca Crear credenciales Eliminar

Credenciales Crea credenciales para acceder a tus API y servicios

Claves de API

Nombre Fecha de creación Última actividad

Clave de API 1 4 ago. 2018 11:11 AM

AlzaS9yAKCArg\_55301817wDobHtaBIVJAgnz...

Para usar esta clave en tu aplicación, transfírela como un parámetro `key=API_KEY`

Tu clave de API

AlzaS9yAKCArg\_55301817wDobHtaBIVJAgnz...

Restringir clave

CERRAR RESTRINGIR CLAVE

Mostrar todo

Logo.png mapa (1).jpg mapa.jpg mapa.jpg mapa.png mapa.jpg mapa.jpg mapa.jpg Mostrar todo

```

<resources>
    <!--
        TODO: Before you run your application, you need a Google Maps API key.

        To get one, follow this link, follow the directions and press "Create" at the end:
        https://console.developers.google.com/flows/enableapi?apiId=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&key=A6:D6:1A:D6:ED:8F:B2:AD

        You can also add your credentials to an existing key, using these values:

        Package name:
        AG:AA:D6:1A:61:D6:ED:EF:BD:13:94:F1:D1:B6:ED:8F:B2:AD

        SHA-1 certificate fingerprint:
        AG:AA:D6:1A:61:D6:ED:EF:BD:13:94:F1:D1:B6:ED:8F:B2:AD

        Alternatively, follow the directions here:
        https://developers.google.com/maps/documentation/android/start#get-key

        Once you have your key (it starts with "Alisa"), replace the "google_maps_key"
        string in this file.

        <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">AlisaSyARC4zqL86Xbt81Fw6BoheHtmZBVJA5guC</string>
    </resources>

```

The screenshot shows the Android Studio interface with the project 'PST\_TA7\_G4' open. The code editor displays the 'activity\_mapa.xml' file under the 'res/values' directory. A specific line of code is highlighted in yellow: '`<string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">AlisaSyARC4zqL86Xbt81Fw6BoheHtmZBVJA5guC</string>`'. The bottom status bar indicates the current time as 13 chars 23:103 CRLF and the file path as Git: Maria\_Feijoo\_Jaramillo.

```

<resources>
    <!--
        TODO: Before you run your application, you need a Google Maps API key.

        To get one, follow this link, follow the directions and press "Create" at the end:
        https://console.developers.google.com/flows/enableapi?apiId=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&key=A6:A4:D6:1A:61:D6:ED:EF:BD:13:94:F1:D1:B6:ED:8F:B2:AD

        You can also add your credentials to an existing key, using these values:

        Package name:
        AG:AA:D6:1A:61:D6:ED:EF:BD:13:94:F1:D1:B6:ED:8F:B2:AD

        SHA-1 certificate fingerprint:
        AG:AA:D6:1A:61:D6:ED:EF:BD:13:94:F1:D1:B6:ED:8F:B2:AD

        Alternatively, follow the directions here:
        https://developers.google.com/maps/documentation/android/start#get-key

        Once you have your key (it starts with "Alisa"), replace the "google_maps_key"
        string in this file.

        <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">AlisaSyARC4zqL86Xbt81Fw6BoheHtmZBVJA5guC</string>
    </resources>

```

This screenshot is identical to the one above, showing the same code editor view of the 'activity\_mapa.xml' file. The highlighted line of code is the same: '`<string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">AlisaSyARC4zqL86Xbt81Fw6BoheHtmZBVJA5guC</string>`'. The bottom status bar shows the time as 17 s 512 ms and the file path as Git: Maria\_Feijoo\_Jaramillo.

- viii. Posteriormente, ingresamos al siguiente link <http://www.coordenadas-gps.com/>, para poder obtener las coordenadas de las direcciones que deseamos marcar en el mapa. Y las pegamos en los lugares correspondientes de longitud y latitud. Para nuestro caso agregamos dos marcas, una en la ciudad de Guayaquil y otra en la ciudad de Quito.

No es seguro | www.coordenadas-gps.com

login | registrarse | Follow | Tweet

Inicio | Como llegar | Sistema de Coordenadas | Convertidor | My Location | País | Estados | Mapa Personalizado

Haz clic en el mapa para obtener su dirección y coordenadas GPS directamente. La latitud y longitud se muestran en la columna de la izquierda y en el mapa.

Dirección: Centro Comercial El Recreo, Avenida Pedro Vicente Maldonado

GD (grados decimales)\*

Latitude: -0.2525943 | Longitude: -78.5224697

GMS (grados, minutos, segundos)\*

Latitude: 0° N 15' 9.339" | Longitude: 78° E 31' 20.89"

Obtener Dirección

Obtener Altitud

Latitud: -0.252594 | Longitud: -78.52247

Obtener Dirección

Las cookies nos permiten ofrecer nuestros servicios. Al utilizar nuestros servicios, aceptas el uso que hacemos de las cookies.

OK!

Logo.png | mapa (1).jpg | mapa.jpg | mapa.jpg | mapa.png | mapa.jpg | mapa.jpg | Mostrar todo

PST\_TA7\_G4 [C:\Users\Usuario\Documents\UNIVERSIDAD\U\SEPTIMO SEMESTRE\PST\_TA7\_G4] -> app\src\main\java\com\example\pst\_ta7\_g4\Mapa.java [app] - Android Studio

```

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

/**
 * Manipulates the map once available.
 * This callback is triggered when the map is ready to be used.
 * This is where we can add markers or lines, add listeners or move the camera.
 * In this case, we just add a marker near Sydney, Australia.
 * If Google Play services is not installed on the device, the user will be prompted to install
 * it inside the SupportMapFragment. This method will only be triggered once the user has
 * installed Google Play services and returned to the app.
 */
@Override
public void onMapReady(GoogleMap googleMap) {
    GoogleMap map;
    // Add a marker in Sydney and move the camera
    LatLng quito = new LatLng(-0.2525943, -78.5224697);
    map.addMarker(new MarkerOptions().position(quito).title("Marker in Sydney"));
    map.moveCamera(CameraUpdateFactory.newLatLng(quito));
}

```

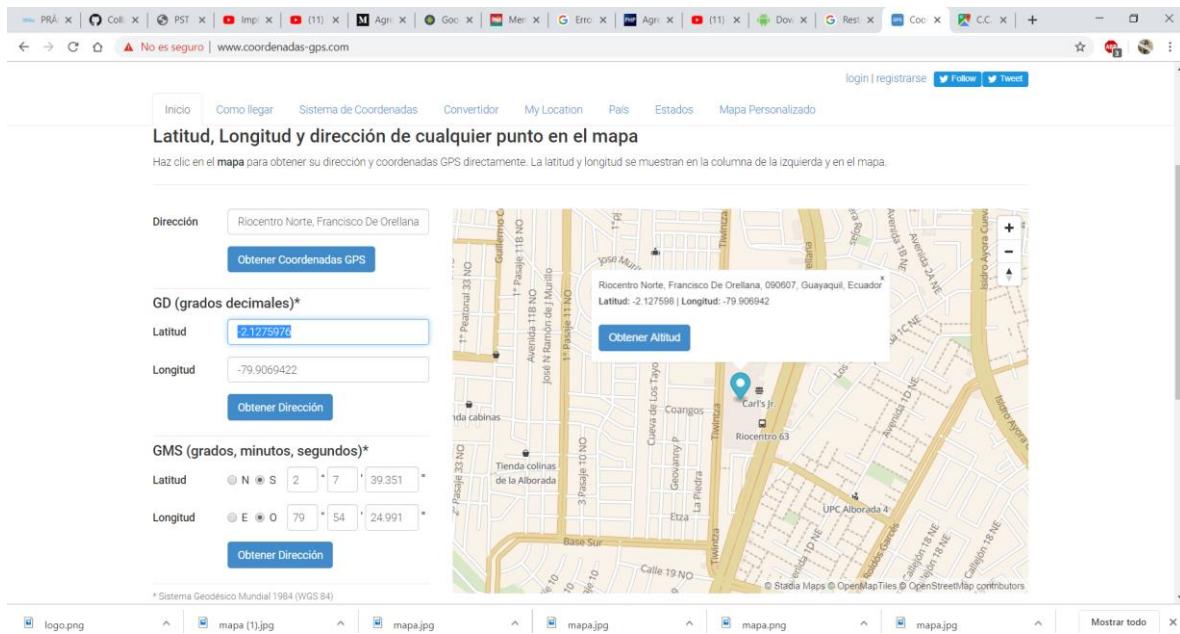
IDE and Plugin Updates

The following components are ready to update:  
Intel x86 Emulator Accelerator (HAXM installer)

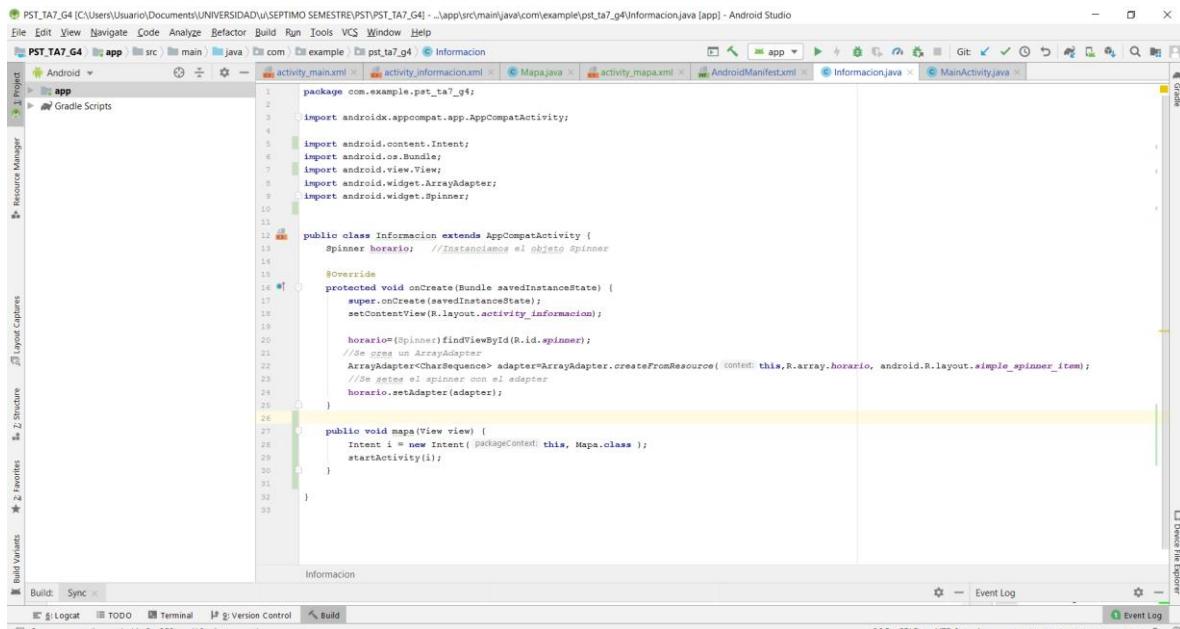
Build: Sync

Logcat | TODO | Terminal | Version Control

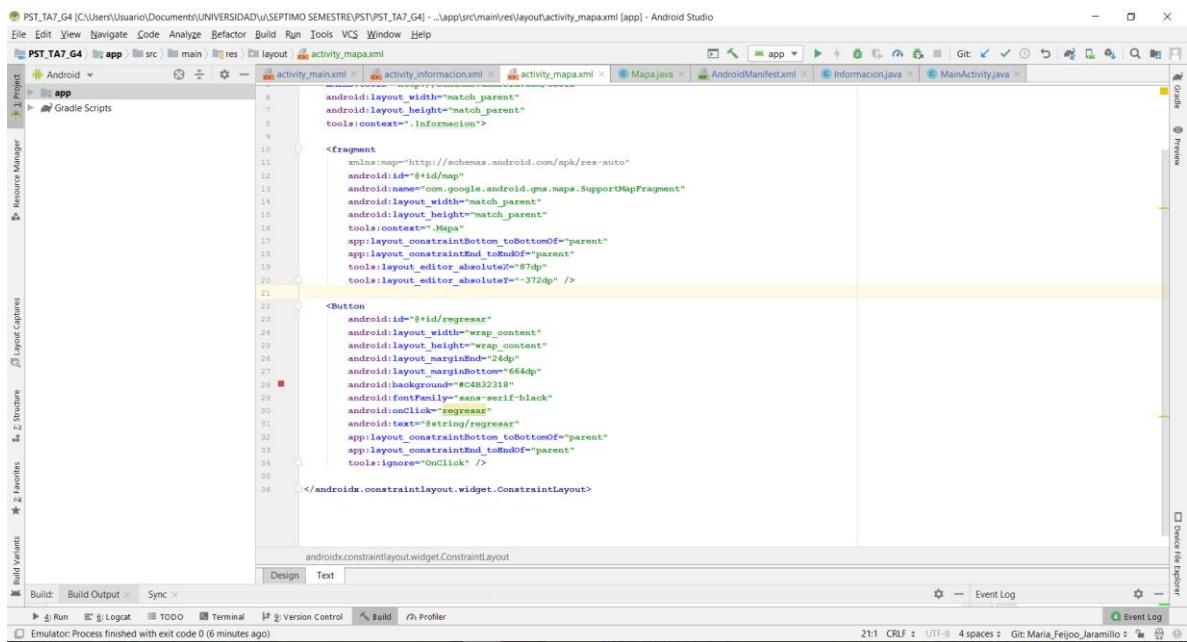
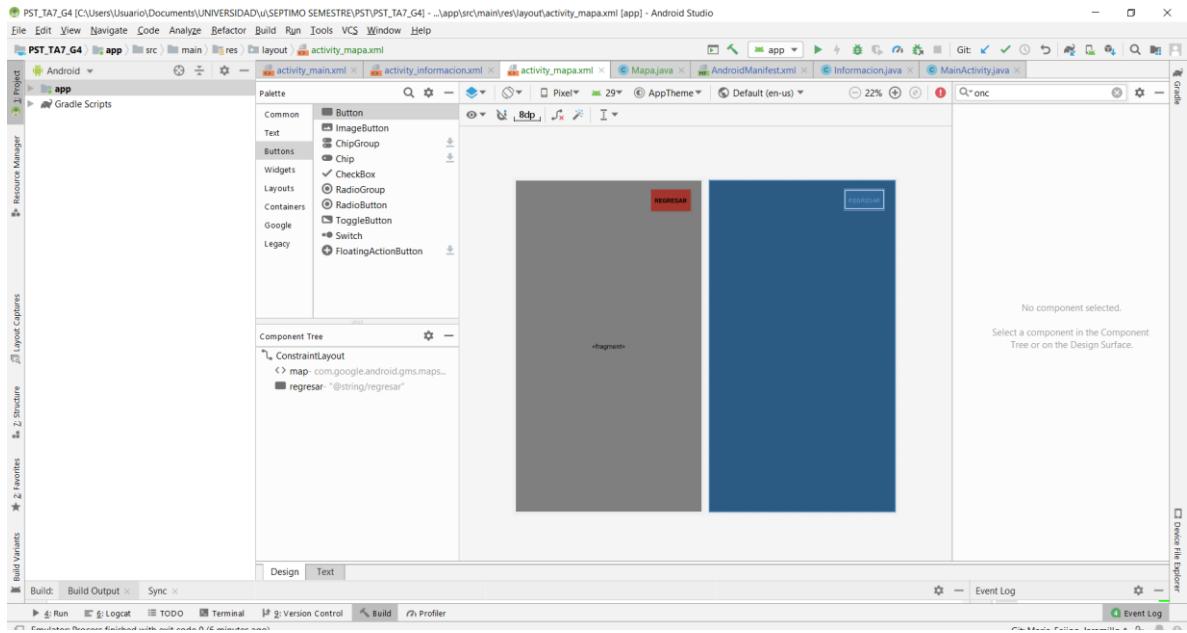
Source generation ended in 2 s 352 ms (3 minutes ago)

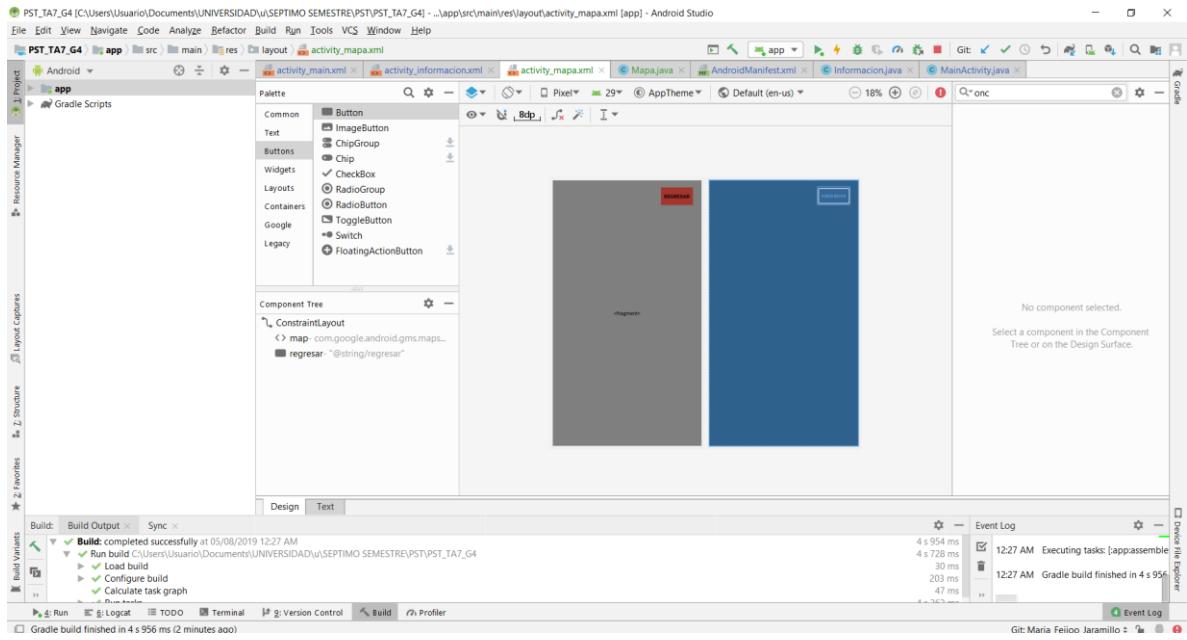


- ix.** Procedemos a crear la función que hará que al presionar el botón, podamos ir a la Activity del mapa, para lo cual en el documento Informacion.java, describimos la función, y en el .xml, modificamos la propiedad onClick, para que se active con esta función denominada mapa.



- x.** Finalmente, agregamos un nuevo botón, que nos permite poder regresar a la ventana anterior, desde el mapa, para lo cual utilizamos el componente Button, y definimos su función en activity\_mapa.java





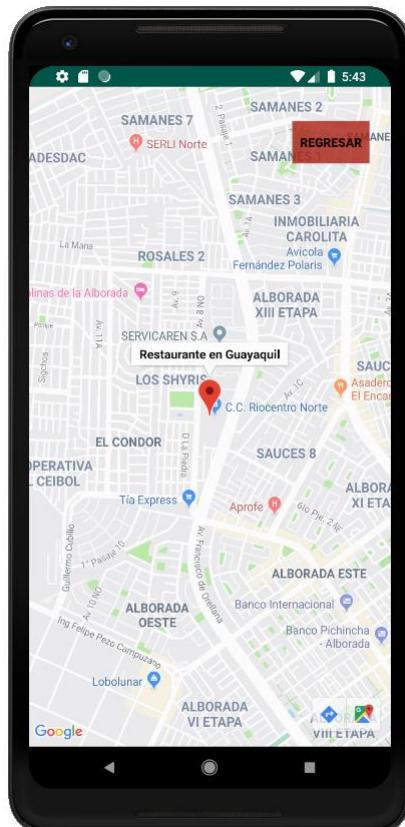
- Finalizando la creación de la Actividad Información, ejecutamos para poder observar los resultados y verificar su correcto funcionamiento.



- Spinner



- Mapa



Ya comprobado su funcionamiento procedemos a subir los cambios realizados en nuestra rama, a la rama master, para lo cual utilizamos los siguientes comandos.

```
MINGW64 ~/Users/Usuario/Documents/UNIVERSIDAD/u/SEPTIMO SEMESTRE/PST/PST_TA7_G4
$ git pull
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details.

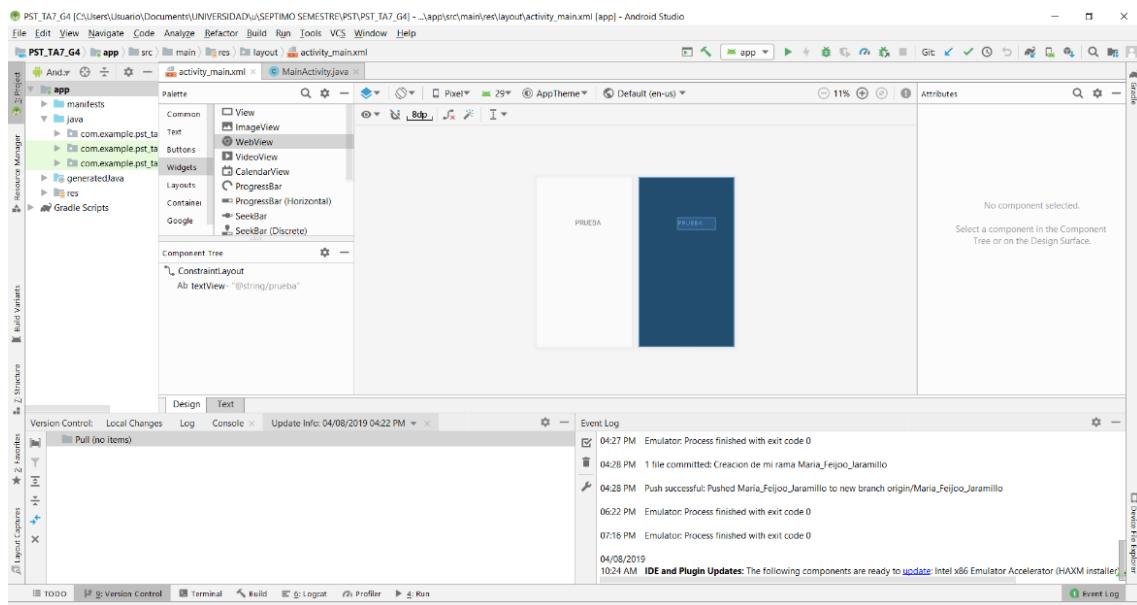
git pull <remote> <branch>
If you wish to set tracking information for this branch you can do so with:
  git branch --set-upstream-to=<origin/><branch> master

User:DESKTOP-FPASOGA MINGW64 ~/Documents/UNIVERSIDAD/u/SEPTIMO SEMESTRE/PST/PST_TA7_G4 (master)
$ git pull origin master
From https://github.com/CisneFeijo0998/PST_TA7_G4
 * branch            master      -> FETCH_HEAD
Auto-merging app/src/main/res/values/strings.xml
CONFLICT (content): Merge conflict in app/src/main/res/values/strings.xml
Auto-merging app/src/main/java/com/example/pst_ta7_g4/MediaActivity.java
CONFLICT (content): Merge conflict in app/src/main/java/com/example/pst_ta7_g4/MediaActivity.java
Automatic merge failed; fix conflicts and then commit the result.

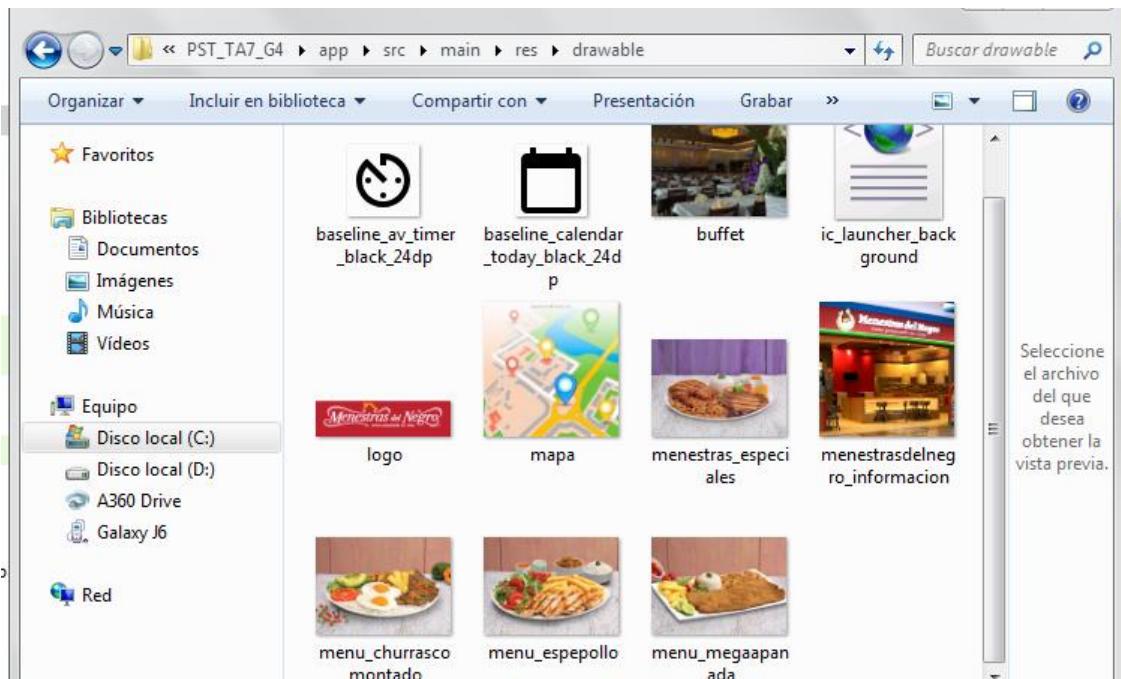
User:DESKTOP-FPASOGA MINGW64 ~/Documents/UNIVERSIDAD/u/SEPTIMO SEMESTRE/PST/PST_TA7_G4 (MERGING)
$ git add .
User:DESKTOP-FPASOGA MINGW64 ~/Documents/UNIVERSIDAD/u/SEPTIMO SEMESTRE/PST/PST_TA7_G4 (MERGING)
$ git commit -m "Implementacion de actividad Informacion, donde presenta datos del restaurante como ubicacion, telefonos y horarios de atencion"
[master 7d60234] Implementacion de actividad Informacion, donde presenta datos del restaurante como ubicacion, telefonos y horarios de atencion
User:DESKTOP-FPASOGA MINGW64 ~/Documents/UNIVERSIDAD/u/SEPTIMO SEMESTRE/PST/PST_TA7_G4 (master)
$ git push
Enumerating objects: 56, done.
Counting objects: 100% (56/56), done.
Delta compression using up to 4 threads
Compressing objects: 100% (28/28), done.
Writing objects: 100% (28/28), 2.28 KiB | 776.00 KiB/s, done.
Total 28 (delta 18), reused 0 (delta 0)
remote: Resolving deltas: 100% (18/18), completed with 9 local objects.
To https://github.com/CisneFeijo0998/PST_TA7_G4.git
 ! [remote rejected] master -> master
error: failed to push some refs to 'https://github.com/CisneFeijo0998/PST_TA7_G4.git'
```

## Implementación de la Actividad “Menú”

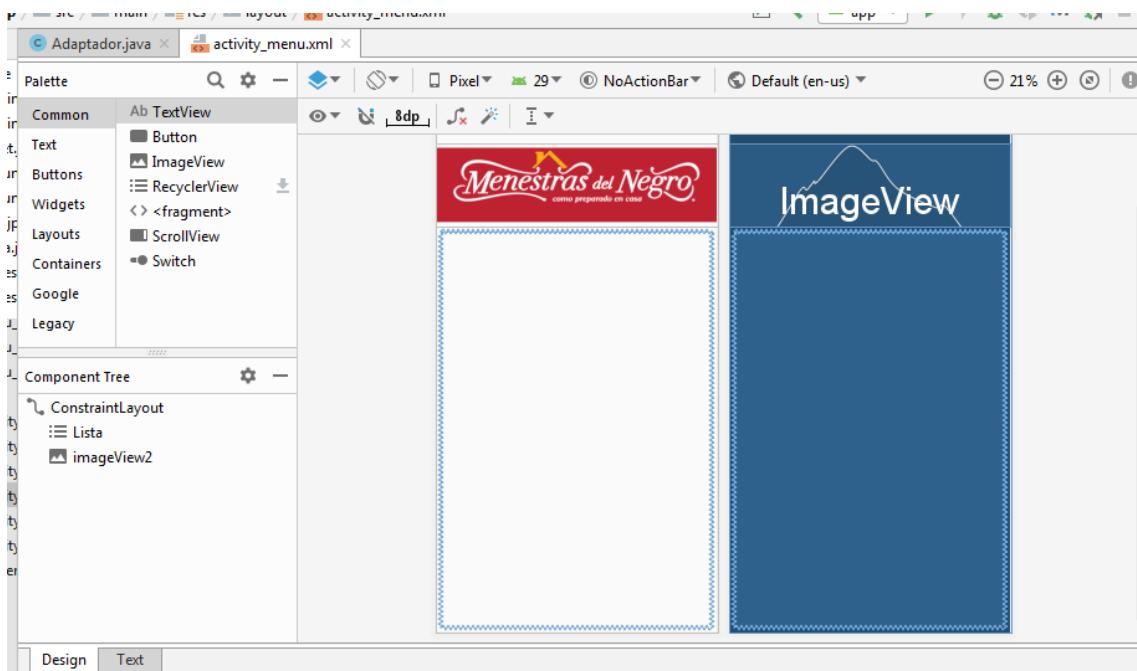
1. Para la creacion del menu, comenzamos con una actividad vacia, la cual llamaremos “Menu”.



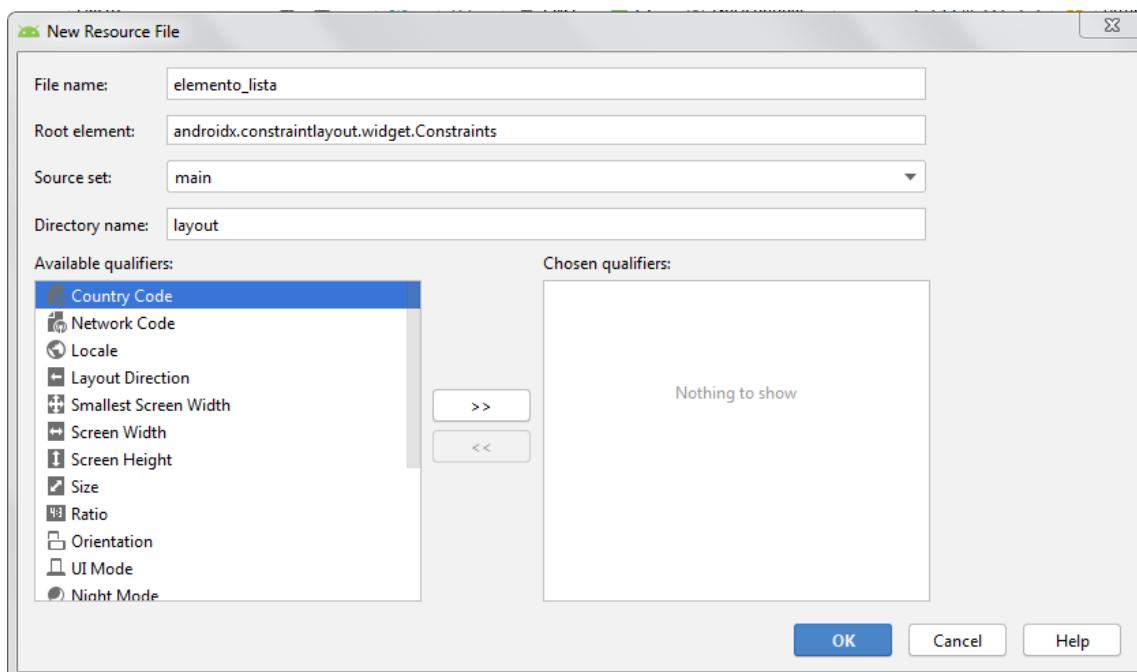
2. Guardamos todas las imágenes que vamos a utilizar para el menú en la carpeta “drawable” de nuestro proyecto C:\Users\Fastline\AndroidStudioProjects\PST\_TA7\_G4\app\src\main\res\drawable



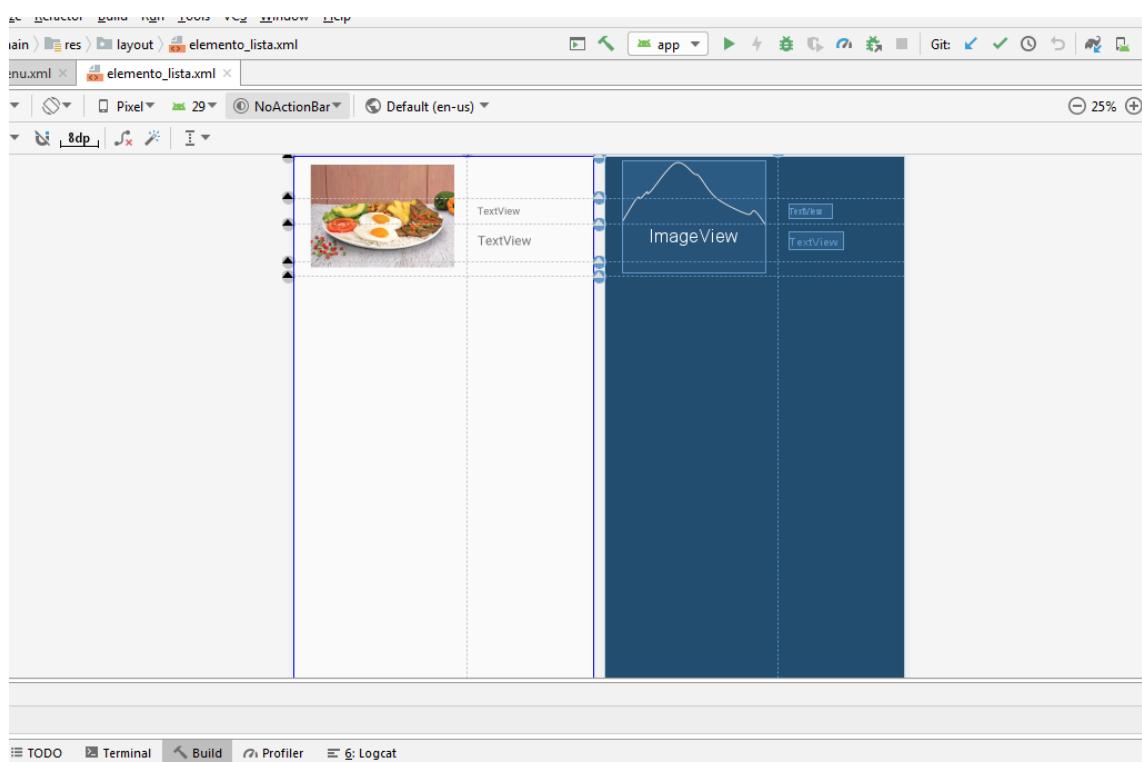
3. En nuestro archivo activity\_menu.xml, en la parte de Design insertamos un listView y un imageView, y las anclamos al constraint layout.



4. En la opción “layout” que se encuentra en la parte derecha del programa, creamos un nuevo layout resource file con el nombre “elemento\_lista.xml” y como root element un constraint layout “androidx.constraintlayout.widget.Constraints”. En este archivo, crearemos lo que ira en cada una de las filas de nuestro listView.



5. Para ayudarnos en el diseño, agregamos líneas de guía (guideline), para este caso 4 horizontales y 1 vertical. Insertamos 1 ImageView (Aquí se presentará el plato) y 2 TextView (Aquí presentaremos el nombre del plato y su precio) y las alineamos y anclamos a las líneas de guía.



- Para nuestro menú, este será el diseño a mostrar para cada uno de los platos del restaurante.

6. En nuestro activity Menu, declaramos las respectivas variables y crearemos una matriz donde almacenaremos la información del menú (nombre del plato, precio, información, etc.), así mismo declaramos un vector de enteros, donde estarán los id de las imágenes en drawable para usarlos después. Cabe recalcar, que deberán estar en el mismo orden de la matriz.

```
public class Menu extends AppCompatActivity {

    ListView lista;

    String[][] datos = {
        {"Churrasco Montado", "$5,99", "La carne y la menestra aportan a nuestra dieta proteínas de alta calidad y aminoácidos esenciales.", "Mega Apanada", "$5,99", "La carne aporta vitamina B, Zinc y proteínas, esto nos ayuda a tener un mejor desarrollo muscular.", "Especialidad del filete", "$7,85", "Al consumir pollo, ensalada y menestra, aportas a tu dieta proteínas de alta calidad, fibra y minerales.", "Menestras Especiales", "$7,25", "Los antioxidantes presentes en la proteína de la carne, pollo o cerdo, se combinan en este plato con verduras y legumbres.", "Recuerda que puedes combinar entre carne, pollo o chuleta.", "$7,25"}
    };

    int[] datosImg = {R.drawable.menu_churrascomontado, R.drawable.menu_megaapanada, R.drawable.menu_espepollo, R.drawable.menestras_especiales};

    private Object Integer;
}
```

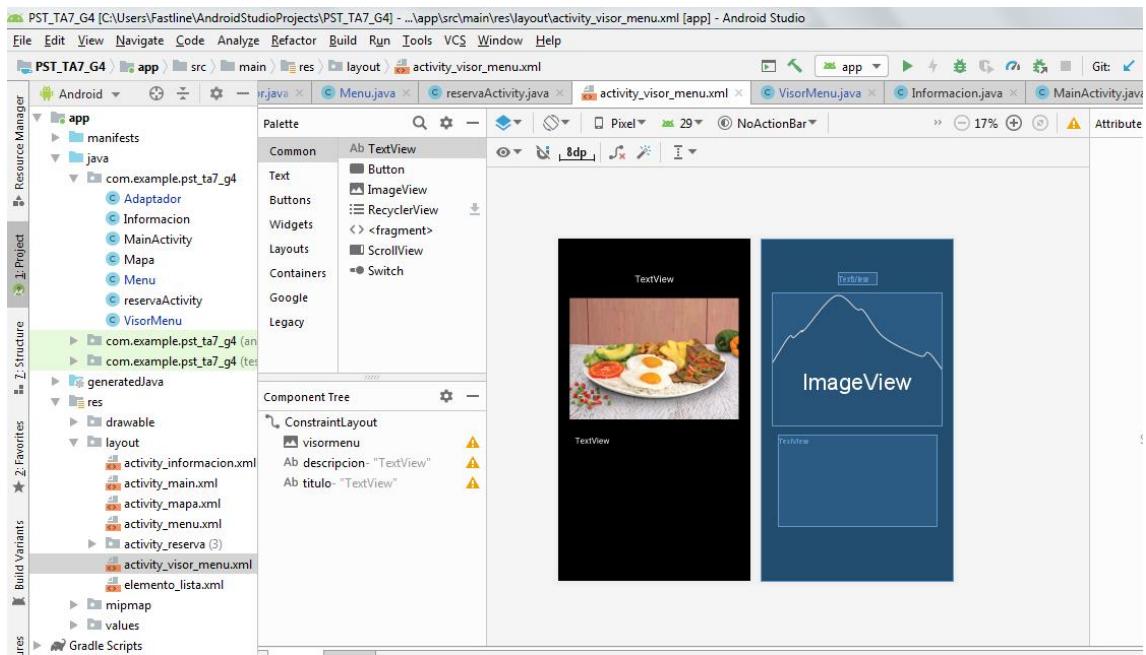
7. Creamos una nueva clase “Adaptador”, esta controlara nuestro adaptador, y se extiende de BaseAdapter. Implementamos los métodos obligatorios, en este caso, solo nos servirá el método getView, el cual se encargara de controlar la posición que nos estén enviando para presentar los datos. Declaramos las respectivas variables; utilizamos el LayoutInflater para instanciar el archivo de diseño xml creado.

```

  c Adaptador.java x  c Menu.java x  c reservaActivity.java x  c VisorMenu.java x  c Informacion.java x  c MainActivity.java x
13
14  //Esta clase controlara nuestro adaptador
15 public class Adaptador extends BaseAdapter {
16
17     private static LayoutInflater inflater = null;
18
19     Context contexto; //entorno de la aplicacion
20     String[][] datos; //matriz de datos principales
21     int[] datosImg; //matriz de datos id imagen
22
23     @
24
25     public Adaptador(Context contexto, String[][] datos, int[] imagenes) {
26
27         this.contexto = contexto;
28         this.datos = datos;
29         this.datosImg = imagenes;
30
31         //sirve para instanciar el archivo de diseño xml creado.
32         inflater = (LayoutInflater)contexto.getSystemService(contexto.LAYOUT_INFLATER_SERVICE);
33     }
34
35     //Controlara la posicion que nos esten enviando para presentar los datos
36     //Este metodo, nos devuelve un contador, por cada elemento que se genere ira incrementando de forma
37     @Override
38     public View getView(int i, View view, ViewGroup viewGroup) {
39
40         final View vista = inflater.inflate(R.layout.elemento_lista, root: null);
41
42         TextView plato = (TextView) vista.findViewById(R.id.nombreplato);
43         TextView valor = (TextView) vista.findViewById(R.id.valorplato);
44         ImageView imagen = (ImageView) vista.findViewById(R.id.imageFood);
45
46         //asignamos los valores, por medio de un contador
47
48     }
49
50     Adaptador > getView()
51
d: Sync x

```

8. Crearemos una nueva actividad “Visormenu” en la cual mostraremos la descripción del plato, en esta ventana añadimos un ImageView, y dos TextView. En el image mostraremos el plato y en los textView el título y descripción respectivamente.



9. En la parte de su código, declaramos los elementos y utilizamos el bundle para obtener los extras y si no es nulo, obtenemos esos datos por medio de un get.String para los textView y get.Int para los id de las imágenes.

```

1 package com.example.pst_ta7_g4;
2
3 import ...
4
5 public class VisorMenu extends AppCompatActivity {
6
7     //cuando se de clic en algun elemento de la lista
8     // nos manda a esta actividad y se muestre los respectivos datos
9     //de cada plato
10    @Override
11    protected void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13        setContentView(R.layout.activity_visor_menu);
14
15        //declararemos los elementos y obtenemos la informacion que se esta obteniendo
16        TextView titulo = (TextView) findViewById(R.id.titulo);
17        TextView detalles = (TextView) findViewById(R.id.descripcion);
18        ImageView img = (ImageView) findViewById(R.id.visormenu);
19
20        Intent intent = getIntent();
21        Bundle b = intent.getExtras();
22
23        //si no es nuelo, obtenemos el titulo, descripcion e imagen
24        if(b!=null) {
25            titulo.setText(b.getString( key: "TIT"));
26            detalles.setText(b.getString( key: "DET"));
27            img.setImageResource(b.getInt( key: "IMG"));
28        }
29    }
30
31}
32
33}
34
35

```

- 10.** Una vez terminada la parte del menú, ejecutamos la aplicación para ver que todo funcione correctamente y subimos a nuestra rama primeramente y luego lo fusionamos a la rama master.

Merge branch 'master' of https://github.com/CisneFeijoo1998/PST\_TA7\_G4 into master

Fastline@Fastline-PC-12 MINGW64 ~/AndroidStudioProjects (master|MERGING)

```

$ git clone https://github.com/jhonlostataza/PST2_1T2019-Tema3_Aplicacion-de-control-de-seguridad-de-la-boveda-de-un-banco-.git
Cloning into 'PST2_1T2019-Tema3_Aplicacion-de-control-de-seguridad-de-la-boveda-de-un-banco-'...
remote: Enumerating objects: 348, done.
remote: Counting objects: 100% (348/348), done.
remote: Compressing objects: 100% (235/235), done.
remote: Total 348 (delta 106), reused 296 (delta 55), pack-reused 0
Receiving objects: 100% (348/348), 917.01 KiB | 1.87 MiB/s, done.
Resolving deltas: 100% (106/106), done.

```

Fastline@Fastline-PC-12 MINGW64 ~/AndroidStudioProjects (master|MERGING)

```

$ git push origin master

```

6:49100%6:50100%



Menestras del Negro  
como preparado en casa

### Churrasco Montado





Churrasco Montado

\$5,99



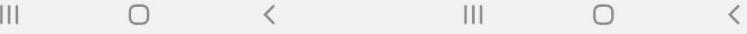
Mega Apanada

\$5,99



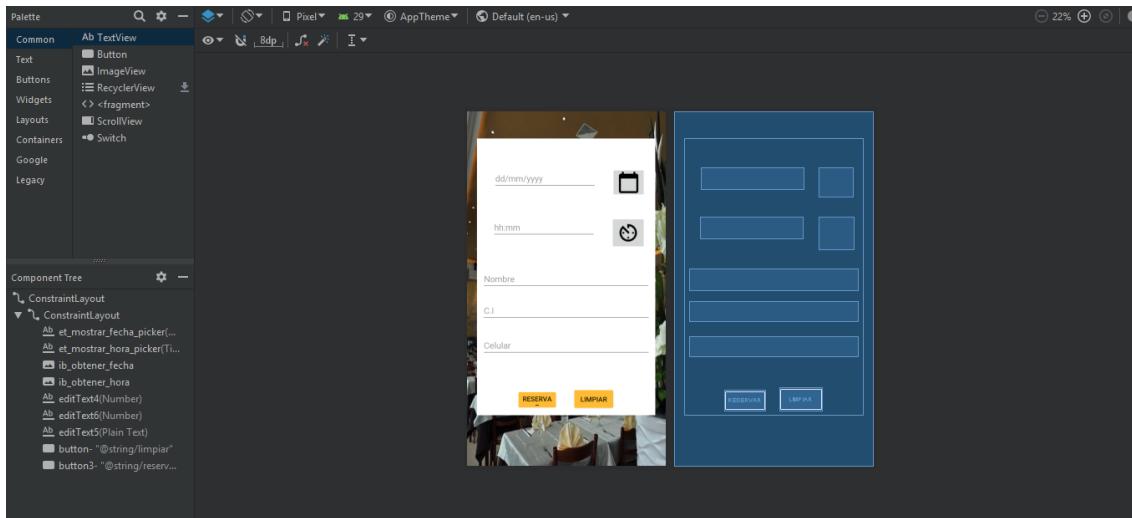
Especialidad del dia

\$7,85



## Implementación de la Actividad “Reserva”

1. Primero diseñamos nuestro archivo *xml* correspondiente a la actividad. En este caso haremos uso de *Image button* para los botones con los logos de calendario y hora a la derecha de su respectivo *Edit Text*. Mismo tipo de *View* que usaremos para los campos de *nombre*, *cédula* y *número celular*, este ultimo lo necesitamos para ser capaces de enviar SMS al dueño del numero y poder dar una alerta de su reserva. De manera global, hacemos uso de dos *constraint layout*.



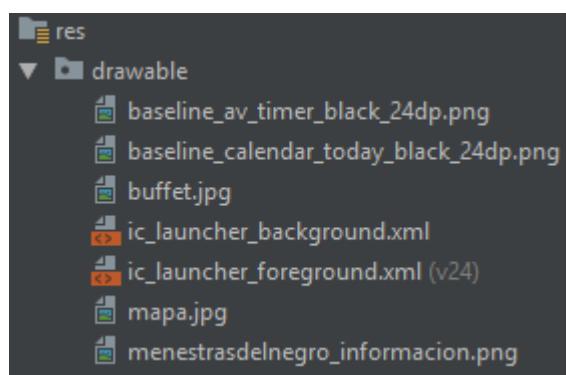
2. Para hacer uso de esta actividad es necesario pedir permisos para enviar SMS, incluyendo cobro de dichos mensajes en un dispositivo real.

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

3. Así mismo en el archivo *strings.xml* declaramos los formatos de fecha y hora con la que queremos que sean presentados en nuestra app.

```
<string name="formato_fecha">dd/mm/yyyy</string>
<string name="formato_hora">hh:mm</string>
```

4. Por otra parte, hace falta agregar en *res/drawable* los iconos que se presentaran en los *Image button* anteriormente mencionados.



5. Para ahondar un poco más en la programación del archivo *xml* correspondiente a la actividad. Un *constraint layout* se usa de contenedor general del formulario para las reservas. Este contiene de fondo una imagen (*buffet.jpg*) anteriormente agregada en la carpeta *drawable*. El segundo *layout* contiene el formulario, y este posee un fondo blanco. Los fondos de los layout se configuran con el atributo *background* en *xml*.

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/buffet"
    tools:context=".ReservaActivity">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="369dp"
        android:layout_height="571dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="56dp"
        android:layout_marginEnd="8dp"
        android:background="@android:color/background_light"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.485"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">
```

6. Hacemos uso de los recursos de imágenes en los *image button* al usarlo en el atributo *src*, característico de la *view* mencionada. Con esta indicamos que imagen queremos que se muestre en nuestro botón.

```
<ImageButton
    android:id="@+id/ib_obtener_fecha"
    android:layout_width="70dp"
    android:layout_height="61dp"
    android:layout_marginTop="60dp"
    android:src="@drawable/baseline_calendar_today_black_24dp"
    app:layout_constraintHorizontal_bias="0.612"
    app:layout_constraintLeft_toRightOf="@+id/et_mostrar_fecha_picker"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:ignore="ContentDescription" />
```

7. En el archivo *java* de nuestra actividad, declaramos los *edit text* necesarios, y declaramos algunas variables tipo *string* necesarias para hacer menos laboriosa la tarea de programar más adelante. Algo peculiar que se realiza es llamar una instancia de *Calendar*, y de este obtener un variables *int* de mes, dia y año. Así también de fecha y hora. Esto se realiza para más adelante usar en los cuadros de dialogo de selección de *fecha y hora* en sus respectivos métodos.

```

public class reservaActivity extends AppCompatActivity implements View.OnClickListener {

    private static final String CERO = "0";
    private static final String BARRA = "/";
    private static final String DOS_PUNTOS = ":";
    private static final int SMS_PERMISSION_CONSTANT = 0;

    public final Calendar c = Calendar.getInstance();

    final int mes = c.get(Calendar.MONTH);
    final int dia = c.get(Calendar.DAY_OF_MONTH);
    final int anio = c.get(Calendar.YEAR);

    EditText etFecha;
    ImageButton ibObtenerFecha;

    final int hora = c.get(Calendar.HOUR_OF_DAY);
    final int minuto = c.get(Calendar.MINUTE);

    EditText etHora;
    ImageButton ibObtenerHora;

    EditText etNumero;
    EditText etNombre;
    EditText etCedula;
}

```

- En el método *onCreate*, correspondemos cada variable inicializada a su respectiva *view*. Notamos que en las primeras líneas del método pedimos los permisos para enviar mensajes desde el celular del usuario. También como iniciamos un *listener* al momento de presionar los *image button*.

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_reserva);
    ActivityCompat.requestPermissions(activity: reservaActivity.this, new String[]{Manifest.permission.SEND_SMS}
        , SMS_PERMISSION_CONSTANT);
    //Widget EditText donde se mostrara la fecha obtenida
    etFecha = (EditText) findViewById(R.id.et_mostrar_fecha_picker);
    //Widget ImageButton del cual usaremos el evento clic para obtener la fecha
    ibObtenerFecha = (ImageButton) findViewById(R.id.ib_obtener_fecha);
    //Evento setOnClickListener - clic
    ibObtenerFecha.setOnClickListener(this);

    //Widget EditText donde se mostrara la hora obtenida
    etHora = (EditText) findViewById(R.id.et_mostrar_hora_picker);
    //Widget ImageButton del cual usaremos el evento clic para obtener la hora
    ibObtenerHora = (ImageButton) findViewById(R.id.ib_obtener_hora);
    //Evento setOnClickListener - clic
    ibObtenerHora.setOnClickListener(this);

    //EditText que contienen datos del cliente.
    etNumero = (EditText) findViewById(R.id.editText4);
    etNombre = (EditText) findViewById(R.id.editText5);
    etCedula = (EditText) findViewById(R.id.editText6);
}

```

- Los *listener* anteriormente creados, dispuestos a cada botón de imagen, son debidamente procesados en el método *onClick*, donde al obtener el *Id* del botón aplastado, podemos definir casos y empezar el método respectivo.

```

public void onClick(View v) {
    switch (v.getId()) {
        case R.id.ib_obtener_fecha:
            obtenerFecha();
            break;
        case R.id.ib_obtener_hora:
            obtenerHora();
            break;
    }
}

```

10. En el método de obtener fecha, lo importante es resaltar como hacemos uso de la clase *DatePickerDialog* que al usar su constructor recibe como parámetro tres datos de tipo *int* que son el año, mes y día con el que se inicializa el cuadro de dialogo de fechas. Esta clase nos presenta un dialogo emergente donde debemos escoger un dia y un mes. Estos quedan guardados en las variables del constructor por defecto. Luego seremos capaces de modificarlas dándoles el formato que deseemos. Al final, valiéndonos del *edit text* correspondiente, cambiamos su texto con los datos adquiridos. Al final del método *obtenerFecha* mostramos este dialogo que hemos configurado con *.show()*.

```

private void obtenerFecha(){
    DatePickerDialog recogerFecha = new DatePickerDialog(context: this, (view, year, month, dayOfMonth) -> {
        //Esta variable lo que realiza es aumentar en uno el mes ya que comienza desde 0 = enero
        final int mesActual = month + 1;
        //Formateo el dia obtenido: antepone el 0 si son menores de 10
        String diaFormatead0 = (dayOfMonth < 10)? CERO + String.valueOf(dayOfMonth):String.valueOf(dayOfMonth);
        //Formateo el mes obtenido: antepone el 0 si son menores de 10
        String mesFormatead0 = (mesActual < 10)? CERO + String.valueOf(mesActual):String.valueOf(mesActual);
        //Muestro la fecha con el formato deseado
        etFecha.setText(diaFormatead0 + BARRA + mesFormatead0 + BARRA + year);

        },anio, mes, dia);
        recogerFecha.show();
    }
}

```

11. De manera similar funciona el método *obtenerHora*. La diferencia radica en usar la clase *TimePickerDialog* para presentar un dialogo emergente donde debemos escoger la hora con sus respectivos minutos. La esencia de tratamiento de los datos recogidos en este cuadro de dialogo es la misma que en el método anterior.

```

private void obtenerHora(){
    TimePickerDialog recogerHora = new TimePickerDialog(context: this, new TimePickerDialog.OnTimeSetListener() {
        @Override
        public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
            //Formateo el hora obtenido: antepone el 0 si son menores de 10
            String horaFormatada = (hourOfDay < 10)? String.valueOf(CERO + hourOfDay) : String.valueOf(hourOfDay);
            //Formateo el minuto obtenido: antepone el 0 si son menores de 10
            String minutoFormatado = (minute < 10)? String.valueOf(CERO + minute):String.valueOf(minute);
            //obtengo el valor a.m. o p.m., dependiendo de la selección del usuario
            String AM_PM;
            if(hourOfDay < 12) {
                AM_PM = "a.m.";
            } else {
                AM_PM = "p.m.";
            }
            //Muestro la hora con el formato deseado
            etHora.setText(horaFormatada + DOS_PUNTOS + minutoFormatado + " " + AM_PM);
        }
    }, hora, minuto, is24HourView: false);

    recogerHora.show();
}

```

12. Para finalizar, cabe destacar que el método del botón *reservar* mostrado al inicio. Aquí hacemos uso de la clase *SmsManager*. Es por ello que necesitamos permisos para enviar SMS desde la aplicación. *SmsManager* nos permite enviar textos a un numero de celular, que obtendremos cuando el usuario llene el formulario, y a su vez el cuerpo del mensaje que nosotros definimos. Al fallar o realizar con éxito el envío de mensajes, se muestra un cuadro de dialogo con mensajes acordes a la situación con la ayuda de la clase *AlertDialog* y su respectivo constructor.

```

public void reservar(View view) {
    if(!(etNumero.getText().toString().equals("")|etNombre.getText().toString().equals("")|etCedula.getText().toString().equals("")|etFecha.getText().toString().equals("")))
        try {
            SmsManager smsManager = SmsManager.getDefault();
            smsManager.sendTextMessage(etNumero.getText().toString(), null, text: "Su solicitud de reserva ha sido aceptada.\nRecuerde que su cita es el $fecha\n$hora\n$minutos");
            AlertDialog.Builder alerta = new AlertDialog.Builder(context: reservaActivity.this);
            alerta.setMessage("Se ha enviado una confirmación via SMS de su reserva.");
            alerta.setTitle("!Lo esperamos!");
            alerta.setCancelable(true);

            AlertDialog dialogo = alerta.create();
            dialogo.show();

        }catch (Exception e){
            Toast.makeText(context: this, text: "Hubo un error al enviar el mensaje",Toast.LENGTH_LONG).show();
        }
    }else{
        AlertDialog.Builder alerta = new AlertDialog.Builder(context: reservaActivity.this);
        alerta.setMessage("Por favor, brinde los datos solicitados.");
        alerta.setTitle("Campos vacíos detectados");
        alerta.setCancelable(true);

        AlertDialog dialogo = alerta.create();
        dialogo.show();
    }
}

```

13. Se muestran los resultados de lo anteriormente programado.

**PST\_TA7\_G4**

**Campo vacío detectado**  
Por favor, brinde los datos solicitados.

**PST\_TA7\_G4**

**Lun., 5 ago.**  
Agosto de 2019

i	m	m	j	v	s	d
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

**PST\_TA7\_G4**

**12:30 A.M.**

00 05  
50 10  
45 15  
40 20  
35 25  
30

**PST\_TA7\_G4**

**08/08/2019**

**12:30 p.m.**

**Guido**

**0944072396**

**0990736437**

**RESERVAR LIMPIAR**

**PST\_TA7\_G4**

**!Lo esperamos!**  
Se ha enviado una confirmación vía SMS de su reserva.

**PST\_TA7\_G4**

**RESERVAR LIMPIAR**

**099 073 6437**

**03:18**

Su solicitud de reserva ha sido aceptada:  
Fecha: 08/08/2019  
Hora: 12:30 p.m.  
A nombre de: Guido  
C.I.: 0944072396

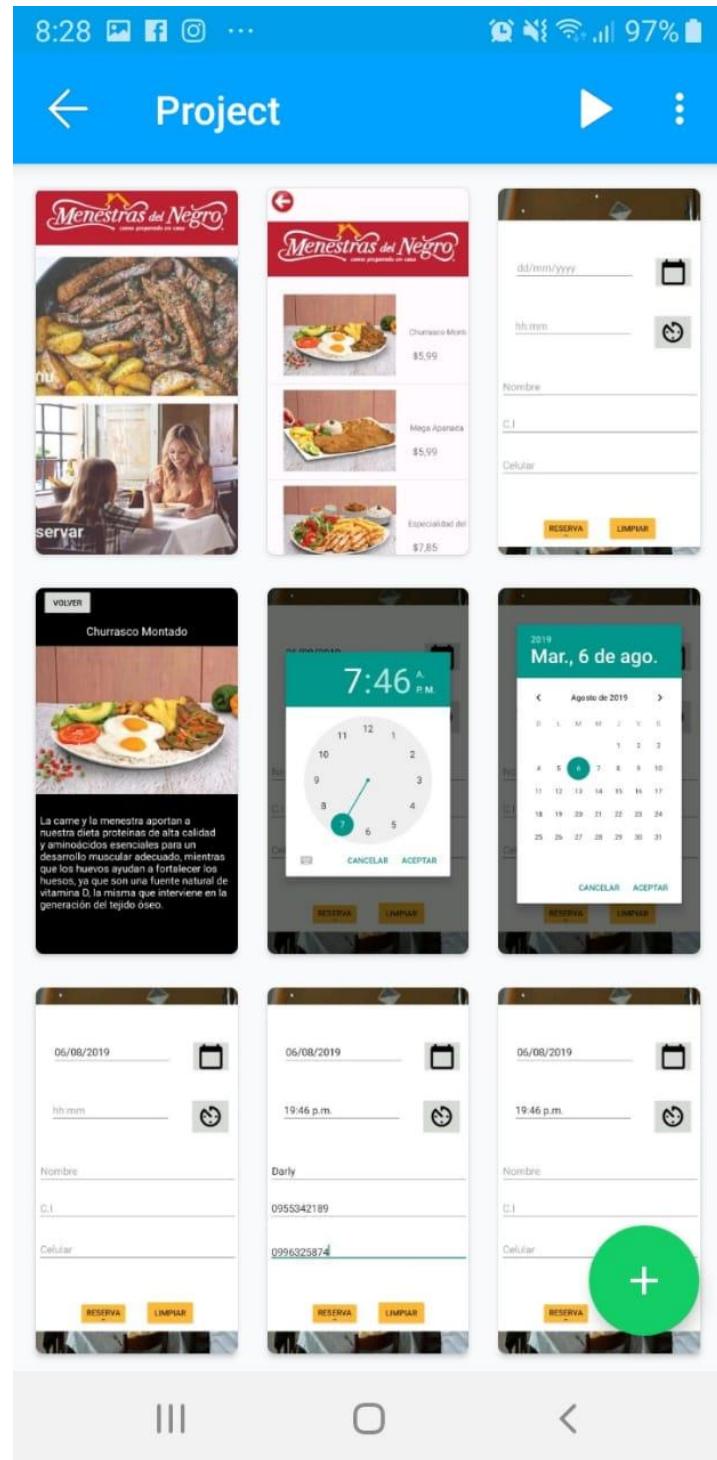
Ahora

Gracias  
Enviando...

Mensaje de texto ➤ SMS

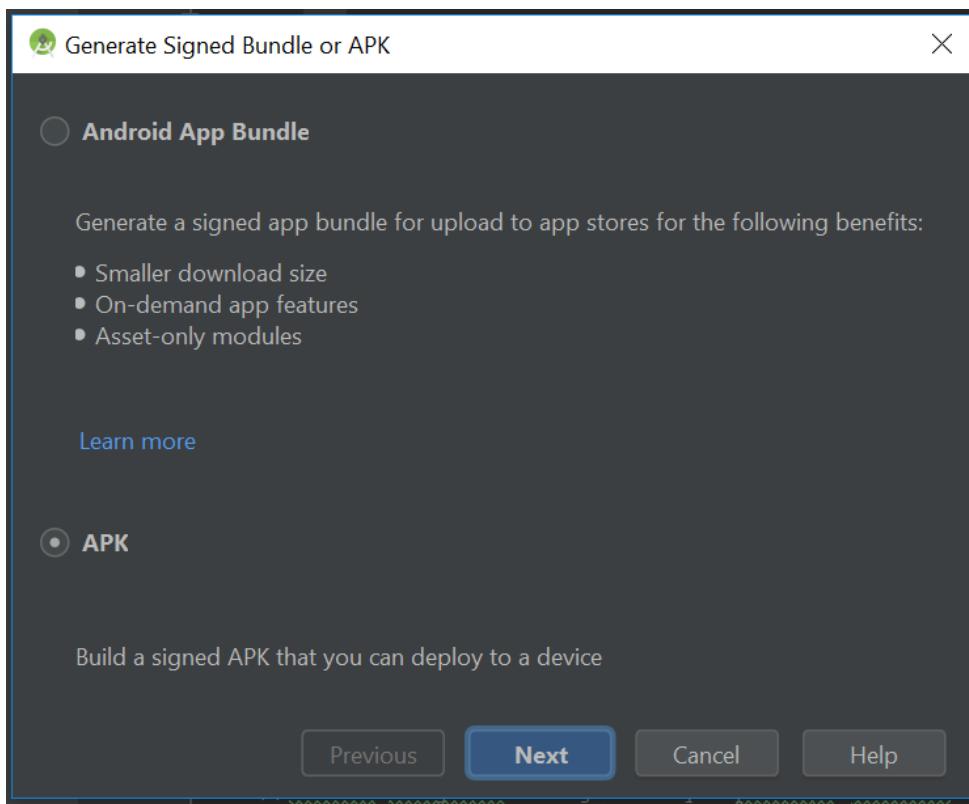
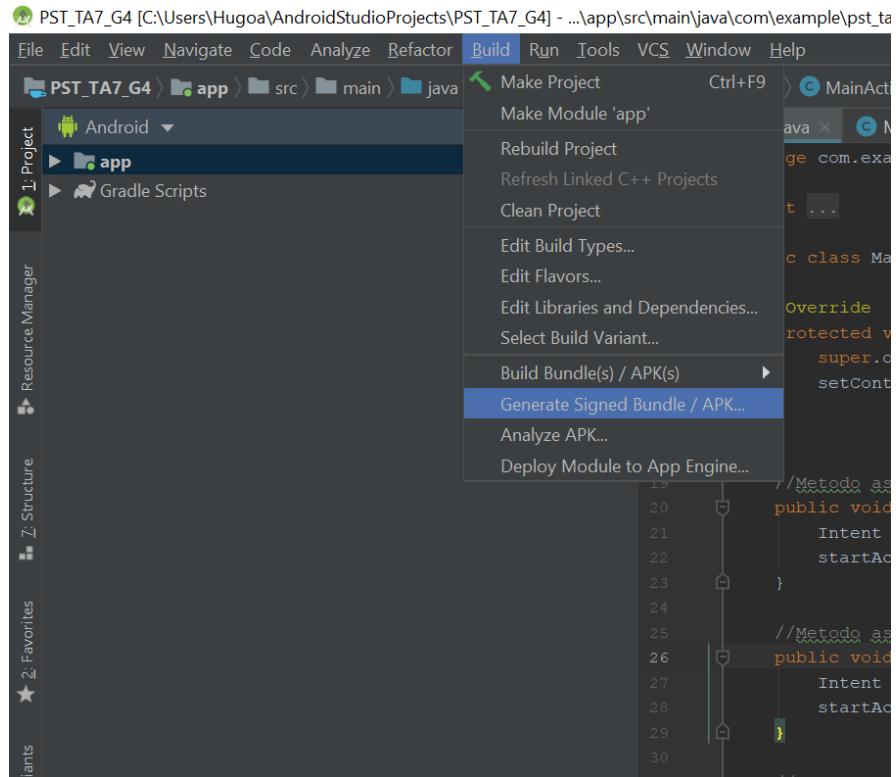
## Paso 7. Prototipo de la Aplicación Marvel

Link: <https://marvelapp.com/973ag7b>

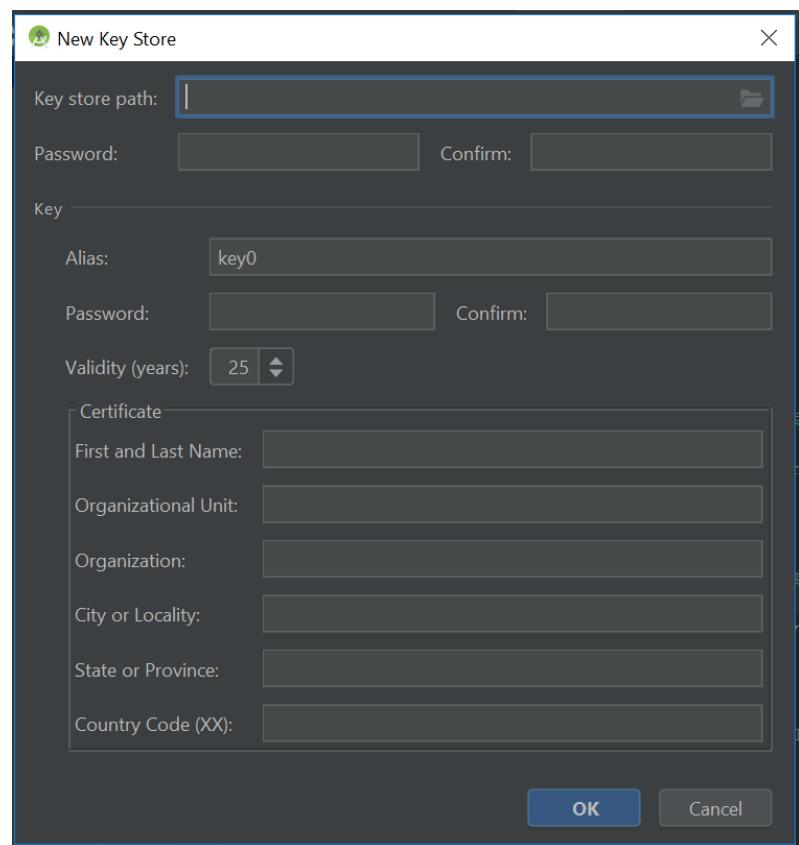
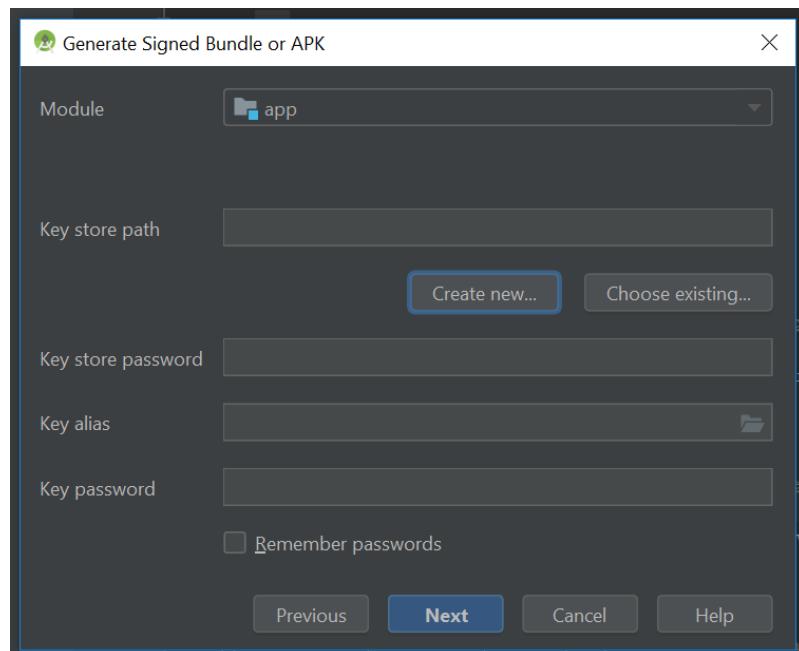


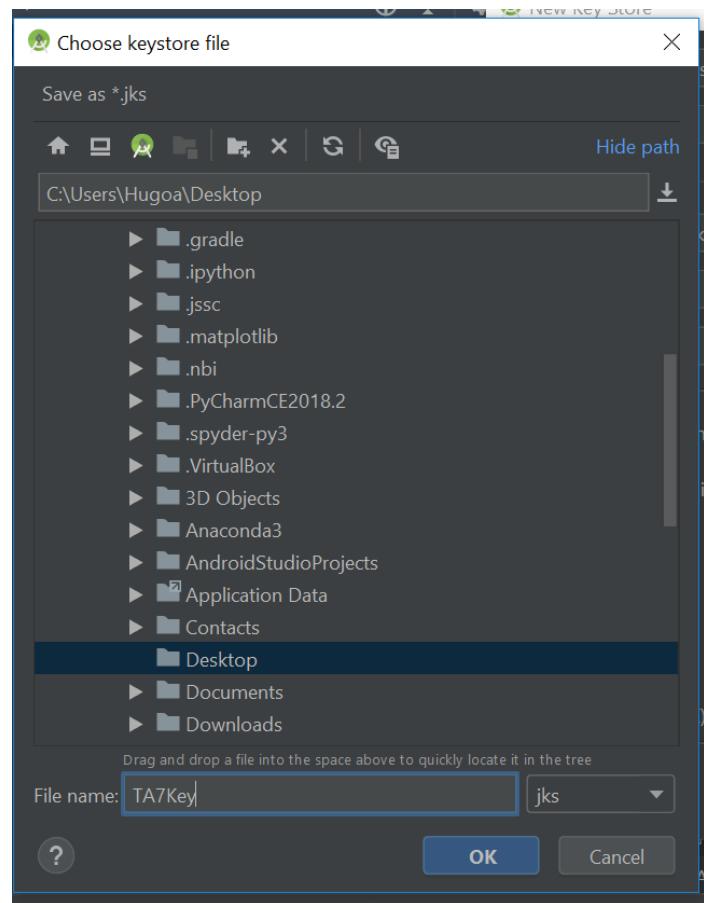
## Paso 8. Generar APK

1. Vamos a la pestaña Build>Generate Signed Bundle APK y hacemos clic. Luego en la ventana que aparece seleccionamos la opción “APK” y clic en Next.



2. En la ventana emergente hacemos clic en Create New. Se deben ingresar los campos requeridos. En directorio es recomendable seleccionar que el archivo apk se genere en el escritorio (por facilidad de localización). Se recomienda, también, que en Key Store Path y en Alias se coloquen los mismos datos, incluido contraseña, para evitar inconvenientes.

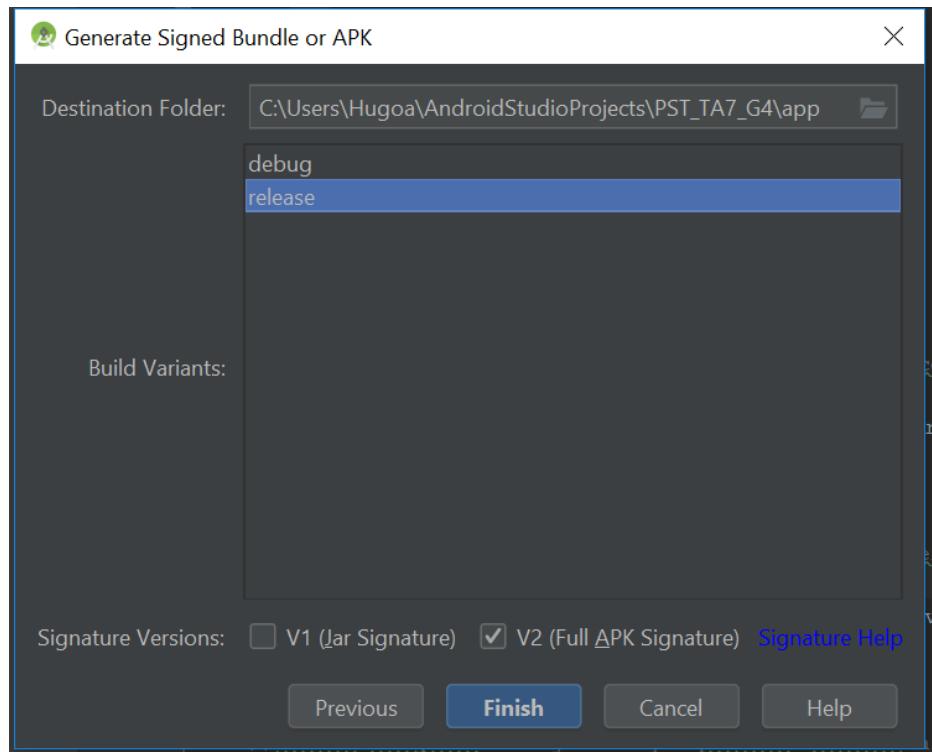
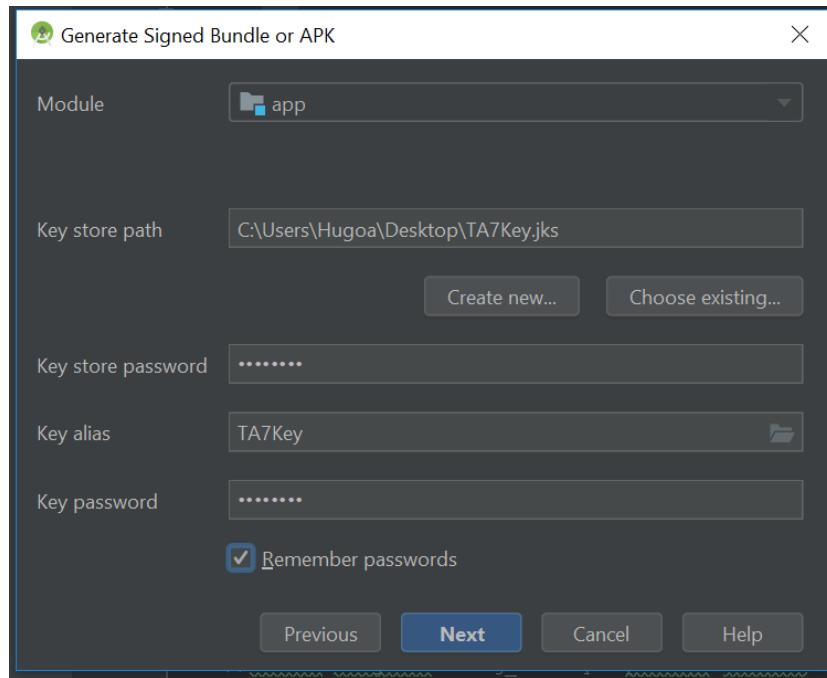




The dialog box is titled "New Key Store". It contains fields for "Key store path" (C:\Users\Hugoa\Desktop\TA7Key.jks), "Password" (\*\*\*\*\*), "Confirm" (\*\*\*\*\*), and "Key". Under "Key", "Alias" is set to "TA7Key". Below "Key" are fields for "Password" (\*\*\*\*\*), "Confirm" (\*\*\*\*\*), and "Validity (years)" (25). A "Certificate" section includes fields for "First and Last Name" (Trabajo Autonomo 7), "Organizational Unit" (PST), "Organization" (ESPOL), "City or Locality" (Guayaquil), "State or Province" (Guayas), and "Country Code (XX)" (Ec). Buttons at the bottom are "OK" and "Cancel".

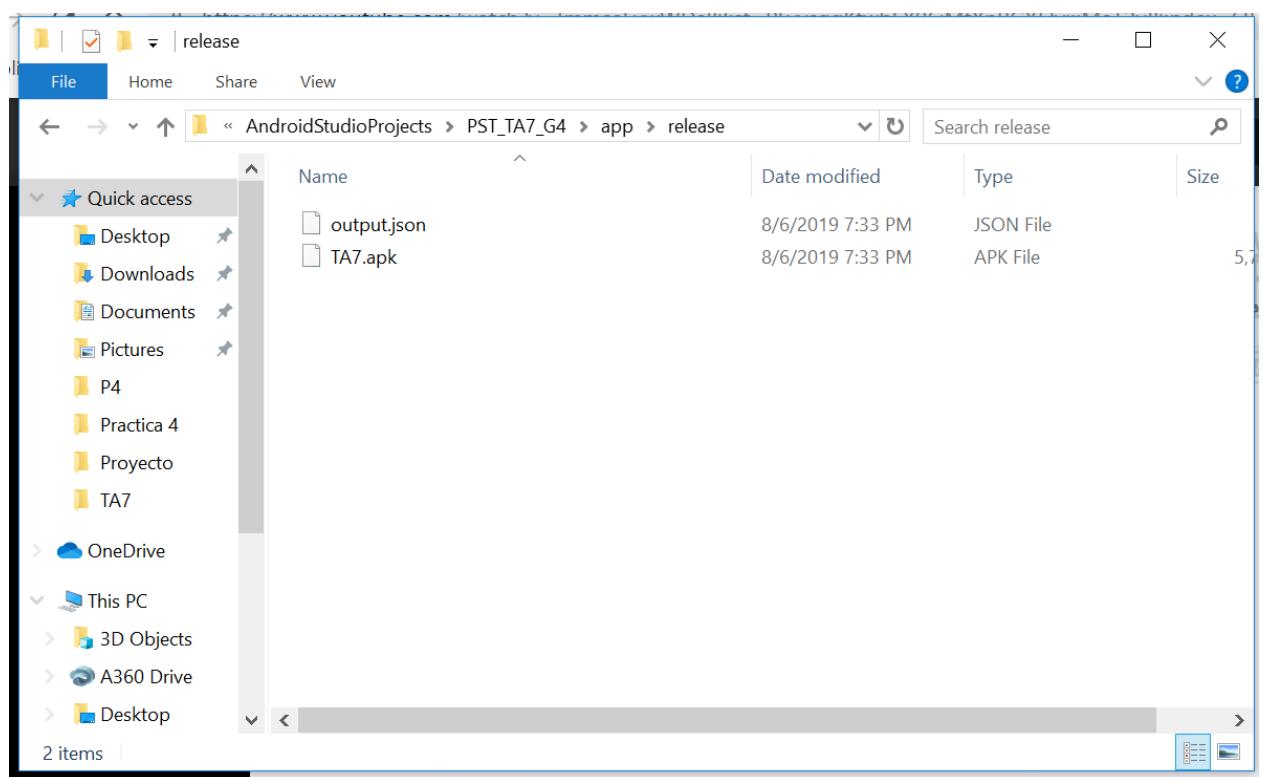
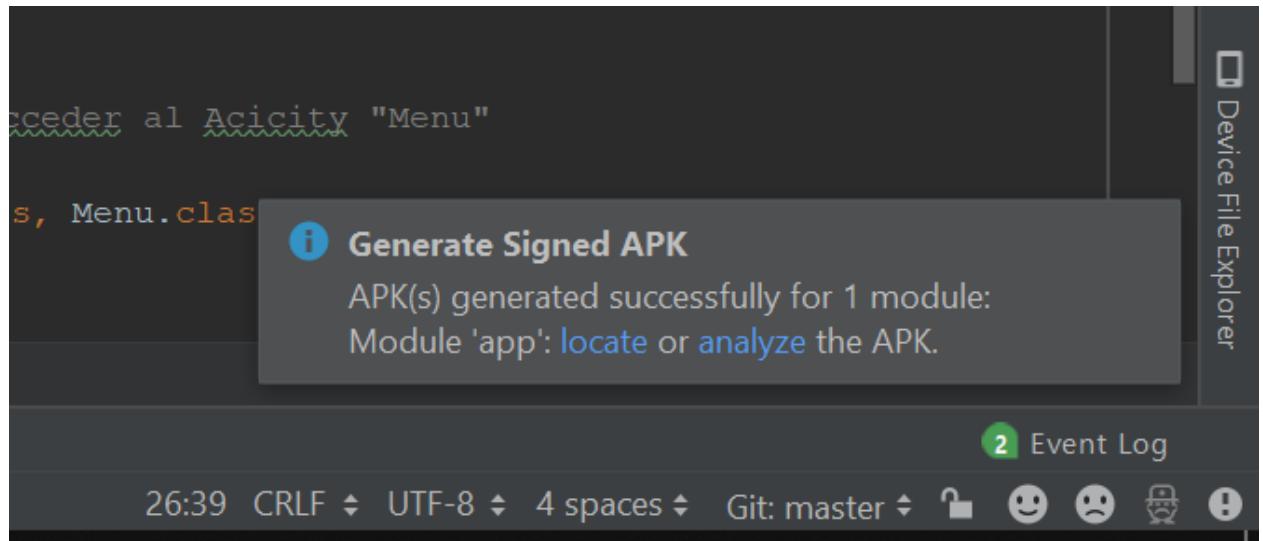
First and Last Name:	Trabajo Autonomo 7
Organizational Unit:	PST
Organization:	ESPOL
City or Locality:	Guayaquil
State or Province:	Guayas
Country Code (XX):	Ec

3. En la ventana nueva confirmamos los datos ingresados y damos clic en Next. Y en la ventana final: en la parte superior seleccionamos la opción reléase y en la parte inferior V2 (Full APK Signature) y clic en Finish.

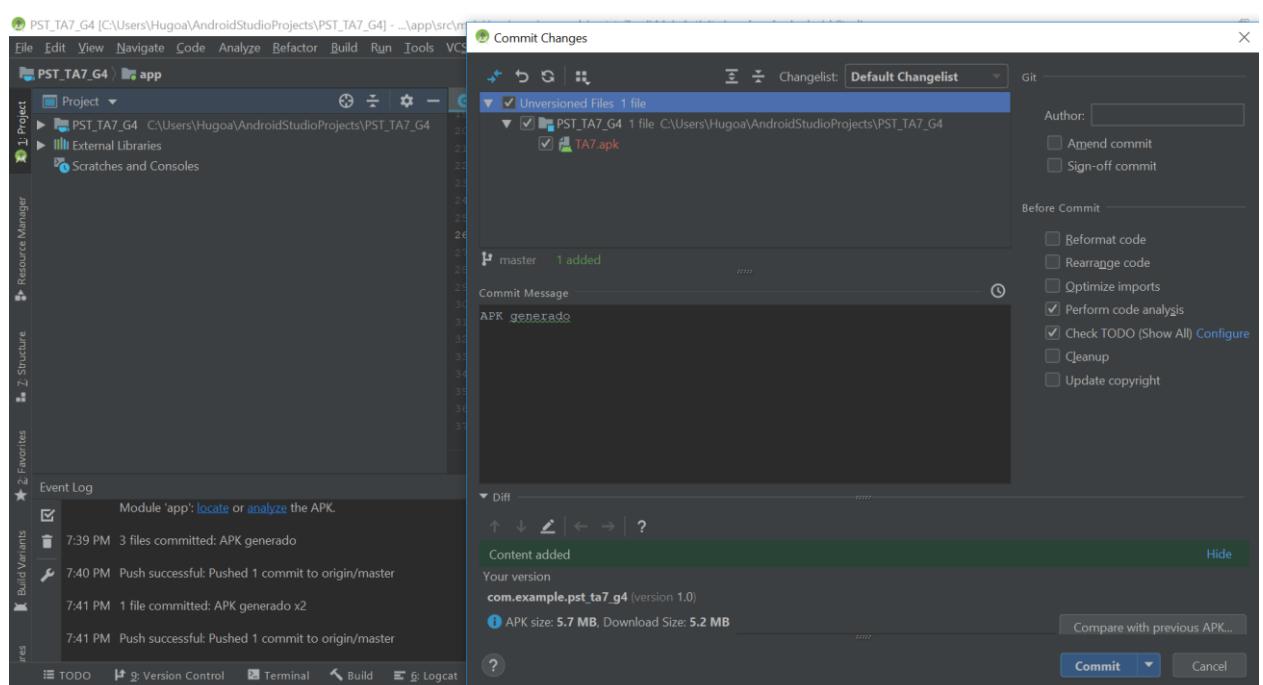
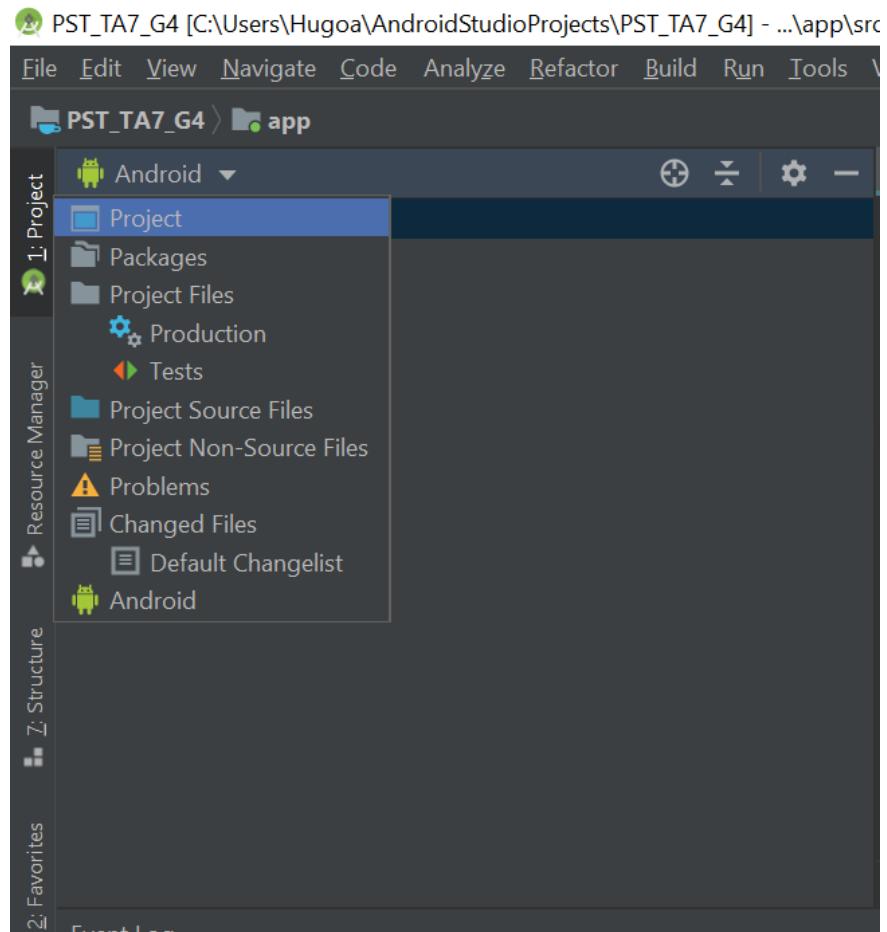


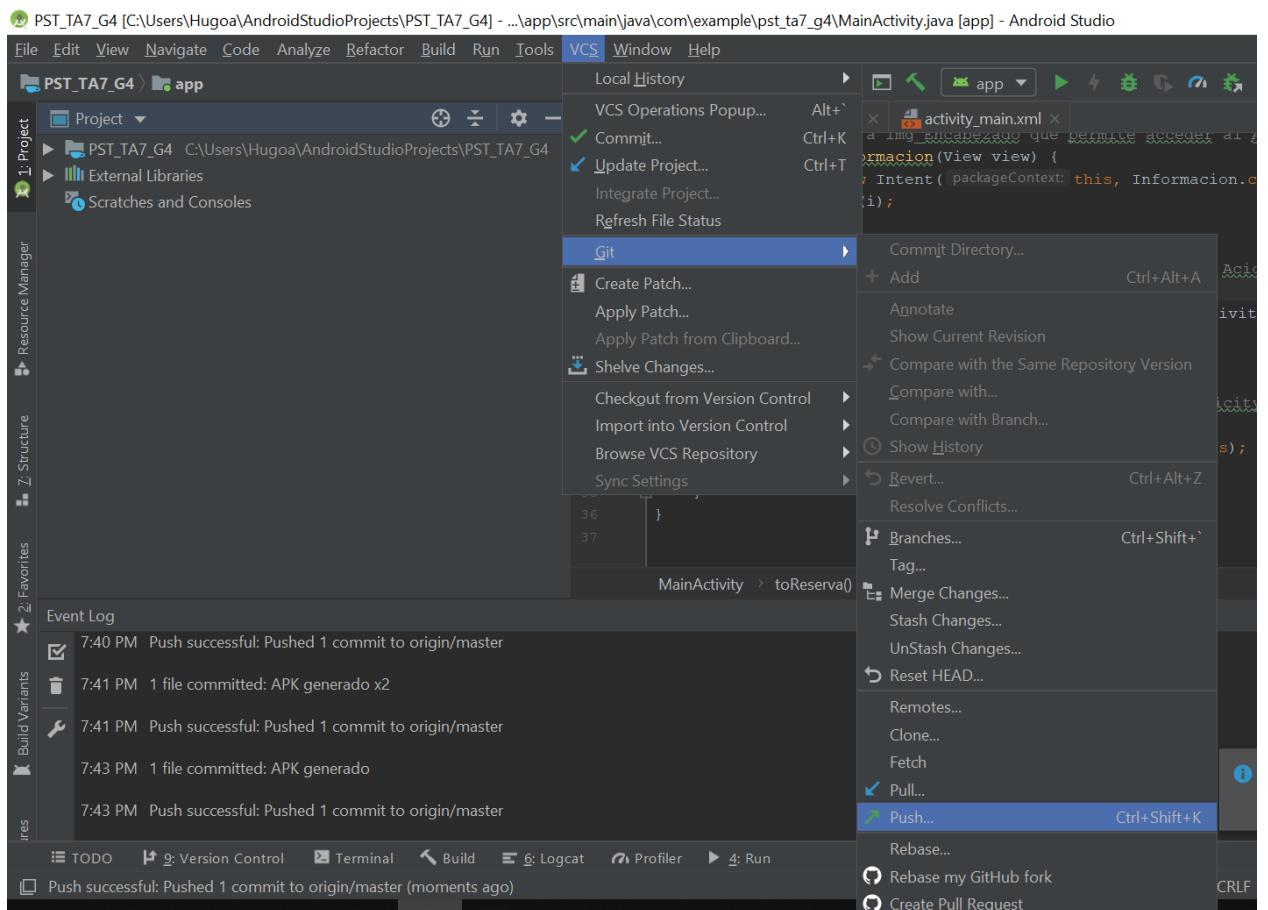
4. Esperar a que el proceso de creación del APK finalice y confirmar la creación del nuevo archivo APK. En la esquina inferior izquierda de la ventana de Android Studio se debe mostrar la notificación de que se generó exitosamente el APK. Para

confirmar la creación también se lo puede hacer accediendo a la carpeta del proyecto en cuestión en el directorio app>release y allí se encontrara el archivo APK con un nombre por default, que es recomendable cambiarlo.



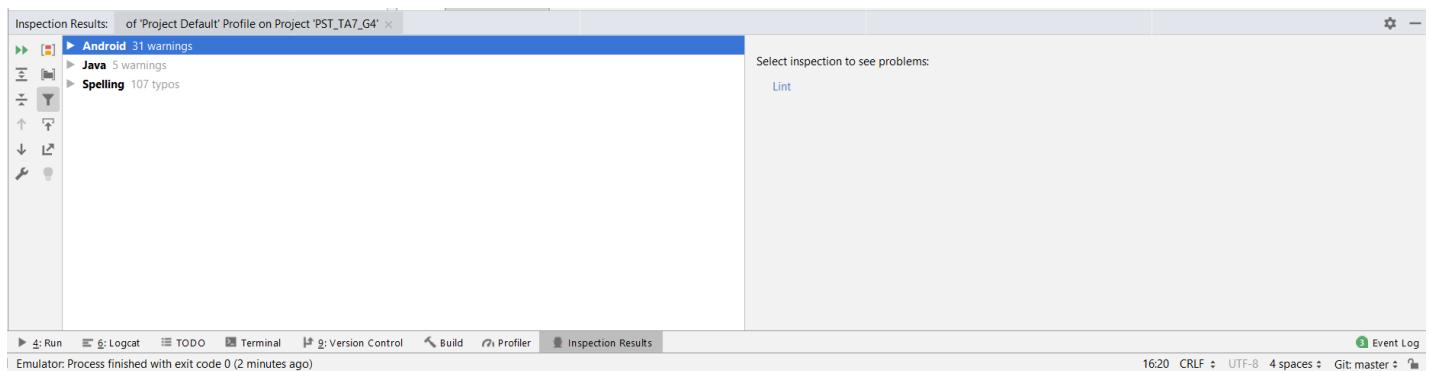
- Para hacer Push con el APK generado es necesario cambiar a la carpeta Project dentro de Android Studio y posteriormente hacer el Commit y posterior Push. De no ser así se omitirán los archivos creados cuando se generó el APK incluyendo el APK en sí mismo.





Branch: master		New pull request	Create new file	Upload files	Find File	Clone or download
	Hugo	APK generado				Latest commit a94937a 44 seconds ago
	.idea	Creacion de diseño de main activity				17 hours ago
	app	APK generado x2				3 minutes ago
	gradle/wrapper	Commit inicial				3 days ago
	.gitignore	Commit inicial				3 days ago
	TA7.apk	APK generado				1 minute ago
	build.gradle	Commit inicial				3 days ago
	gradle.properties	Commit inicial				3 days ago
	gradlew	Commit inicial				3 days ago
	gradlew.bat	Commit inicial				3 days ago
	settings.gradle	Commit inicial				3 days ago

## Paso9. Análisis del código con Lint



## Conclusiones

- ✓ En conclusión, el uso de controles básicos en Android Studio es fundamental para la creación de cualquier aplicación, ya que permiten diseñar apps de una forma más rápida y sencilla.
- ✓ La combinación de distintas views y widgets pueden dar paso a la creación de formularios menos extensos y complejos y de fácil relleno. De manera clara y precisa para evitar confusiones al usuario.
- ✓ Los controles básicos pueden ser mezclados y combinados para permitir mucha flexibilidad de presentación al texto en la pantalla; además es posible personalizarlos ya sea extendiendo su funcionalidad o diseñando uno desde cero, esto nos permite hacer más dinámica y llamativa nuestra aplicación.

## Recomendaciones

- ✓ Se recomienda antes de realizar cualquier cambio en nuestro proyecto, actualizarlo mediante el comando git pull, para evitar problemas cuando intentemos subir nuestra parte del código.
- ✓ Se recomienda que cada integrante del grupo trabaje en su rama, mientras está realizando su actividad para evitar daños en el código principal que pueda afectar a los demás colaboradores.
- ✓ Se recomienda, siempre colocar un id distintivo a cada componente que se ingresa en un Activity conservando un formato fijo para poder identificar tanto: qué tipo de componente es, como la información que contiene.

## **Referencias bibliográficas:**

- Alvarez, M. A. (19 de junio de 2019). *desarrolloweb.com*. Recuperado el 17 de julio de 2019, de Trabajar con ramas en Git: git branch: <https://desarrolloweb.com/articulos/trabajar-ramas-git.html>
- *git*. (s.f.). Obtenido de git: <https://git-scm.com/book/es/v1/Git-en-entornos-distribuidos-Contribuyendo-a-un-proyecto>
- Pascual, J. A. (22 de Julio de 2018). *Computer Hoy*. Obtenido de Computer Hoy: <https://computerhoy.com/reportajes/tecnologia/5-mejores-alternativas-github-2018-275973>
- s.a. (06 de agosto de 2019). *Menestras del Negro*. Recuperado el 01 de agosto de 2019, de <http://www.menestrasdelnegro.com/>