PROGRAMACIÓN DE SISTEMAS TELEMATICOS

***TEMA***

SISTEMA DE PEDIDOS DE JUGUETES FUNKO POP A DOMICILIO

***DOCENTE***

*MSIG. Adriana Collaguazo Jaramillo*

***Grupo Número 4***

* *Andrés Alfonso Alcívar Albán*
* *José Luis Chuchuca Suarez*
* *Carlos Anthony Cárdenas Vanoni*
* *David Daniel Lara Pazmiño*

*AGOSTO 2019*

MANUAL TÉCNICO

SISTEMA DE PEDIDOS DE JUGUETES FUNKO POP A DOMICILIO

# RESUMEN EJECUTIVO

Los productos de películas de acción tienen una alta demanda en niños, jóvenes y adultos por lo que varios establecimientos de la ciudad de guayaquil han iniciado con este emprendimiento, y es por esto que el presente proyecto tiene la finalidad de construir un aplicativo móvil que vincule a clientes, tiendas especializadas en la comercialización de juguetes Funko Pop y conductores motorizados, con la finalidad de que el cliente seleccione la tienda y los productos de interés, y a la vez el conductor recepta el pedido de la tienda y lo entregue donde se disponga. Esto aumentará la venta de los establecimientos ya que abre un abanico de clientes que prefieren realizar compras a domicilio, para evitar la movilización a la tienda además que pueden visualizar de una mejor manera los productos y tomar mejores decisiones. Los conductores también ven en esta aplicación un nuevo modelo de negocio mejorando sus ingresos mensuales. El login es muy sencillo por medio del correo Gmail y el cliente como dueño de la tienda puede observar el perfil del conductor, así como también otorgarle una calificación que busca mejorar los servicios prestados.

# INTRODUCCIÓN

Las tendencias actuales nos llevan a utilizar nuestro dispositivo móvil para cualquier actividad de nuestra vida cotidiana, una necesidad detectada es que al momento de requerir juguetes temáticos únicamente se comercializan en tiendas locales, además el crecimiento de coleccionistas y de requerimientos de este tipo de objetos a crecido exponencialmente por las películas estrenadas en los meses anteriores, por tal motivo se busca suplir esta necesidad y se realiza la aplicación denominada SUPER SHOP, en donde usuarios guayaquileños podrán adquirir productos Geek y que un repartidor lo lleve a su sitio requerido, evitando la movilización a centros comerciales que conllevarían otros gastos y pérdidas de tiempo.

Esto ayudará también a las tiendas ya que aumentarán sus ventas y llegarán a más personas de lo habitual, incluso colando ofertas que solo tengan validez para el uso de la aplicación, y los motorizados tendrán otra fuente de servicios aumentando sus ingresos mensuales.

Descripción del problema

El desarrollo y popularidad de varias películas de acción en esta última década han abierto oportunidad de negocios en otras industrias recaudando un gran margen de utilidad, los juguetes infantiles han potenciado sus ventas con los muñecos Funko Pop, los cuales son altamente deseables por niños, jóvenes y adultos. En la ciudad de guayaquil existen locales especializadas en estos productos, pero desean potenciar sus ventas aliviando a los clientes movilizarse a sus tiendas, por esta razón es necesario elaborar un aplicativo móvil que vincule a los clientes, establecimientos y conductores motorizados.

# Objetivos Específicos

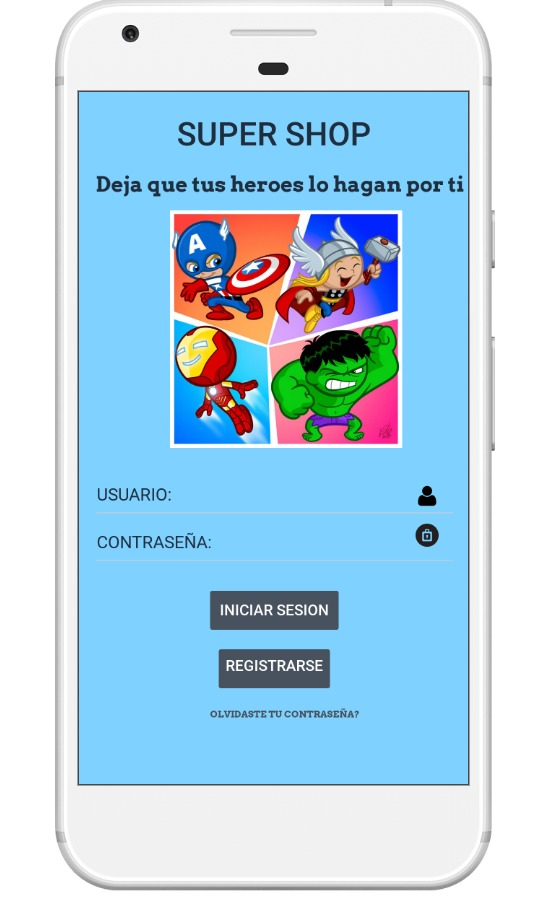
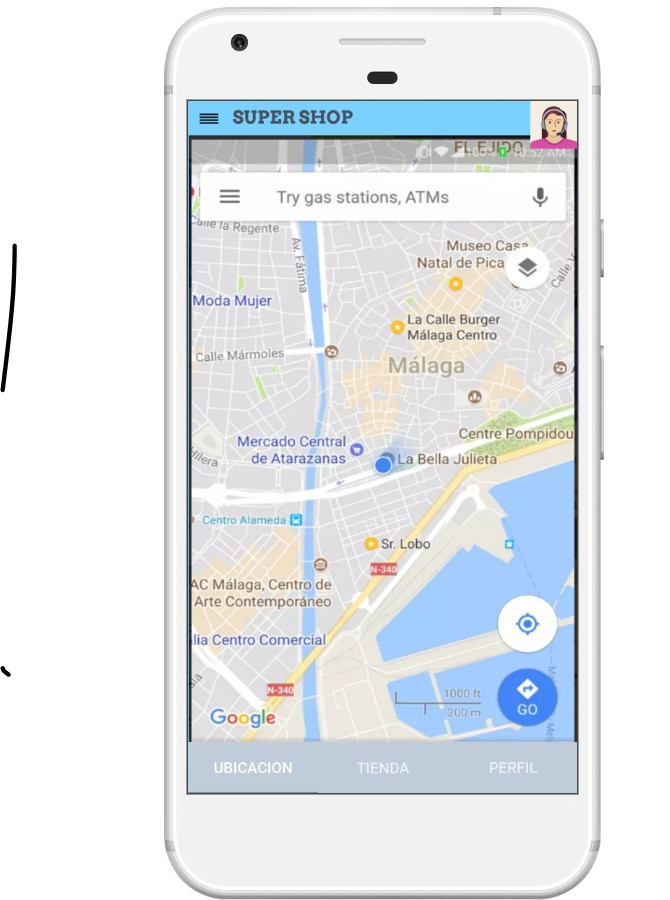
* Realizar un aplicativo móvil en Android Studios que logre vincular a tiendas geek, clientes interesados y conductores motorizados, para realizar entregas a domicilio de muñecos coleccionables Funko Pop.
* Realizar pruebas con tiendas reales en la ciudad de guayaquil, verificando la operatividad del aplicativo y determinar ventajas y desventajas de su uso.
* Vincular el aplicativo móvil con una base de datos remota que permita optimizar recursos para el cliente, tienda y motorizados.

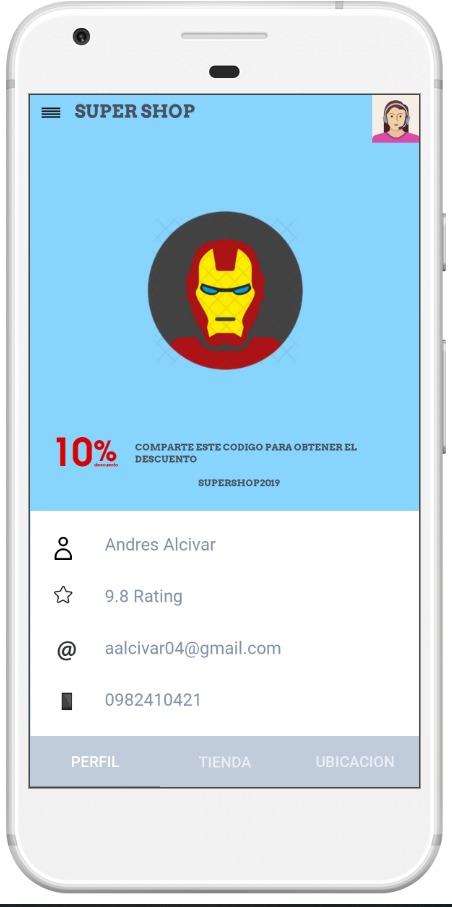
¿Cómo funciona la solución?

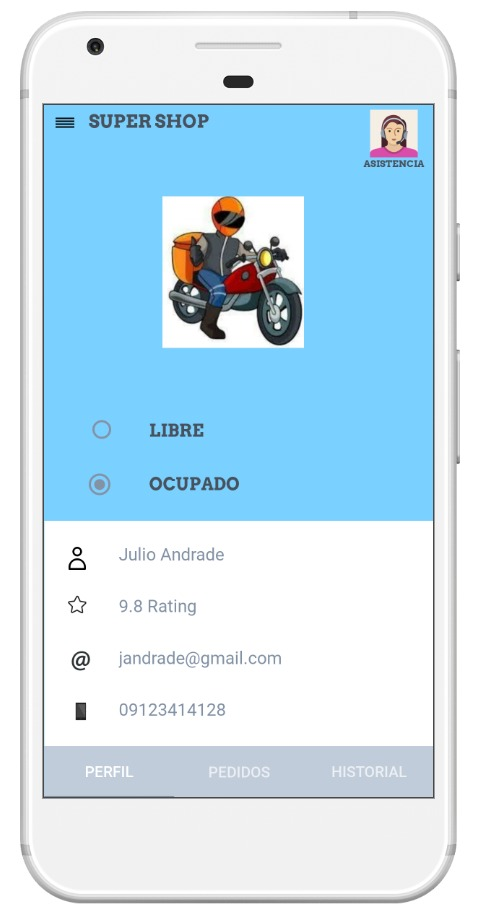
Para la resolución del problema se ha desarrollado un aplicativo móvil llamado SUPER SHOP, el mismo tiene la finalidad de vincular a los clientes con las tiendas especializadas en la comercialización de Funko Pop, para luego entregar el pedido a conductores motorizados registrados en la aplicación que llevarán el producto al lugar deseado, esto logrará abrir el abanico de ventas para las tiendas, y evitará la movilización innecesaria del cliente.

¿Qué van a construir para resolver el problema?

Se va a realizar un aplicativo móvil llamado SUPER SHOP, para este proyecto no aplica la construcción física de componentes. Para tener una idea general del producto final, se utiliza la ayuda de Marvel App para realizar el diseño del aplicativo.

.





Recursos de hardware y de software

Hardware

* Celular con sistema operativo Android para realizar las pruebas.
* 04 computadoras personales para realizar la programación en Android Studios

Software

* Asana
* Android Studios
* Marvel App
* Draw io
* Base de Datos: CloudStore del proyecto Firebase de Google

Diagrama de Grant

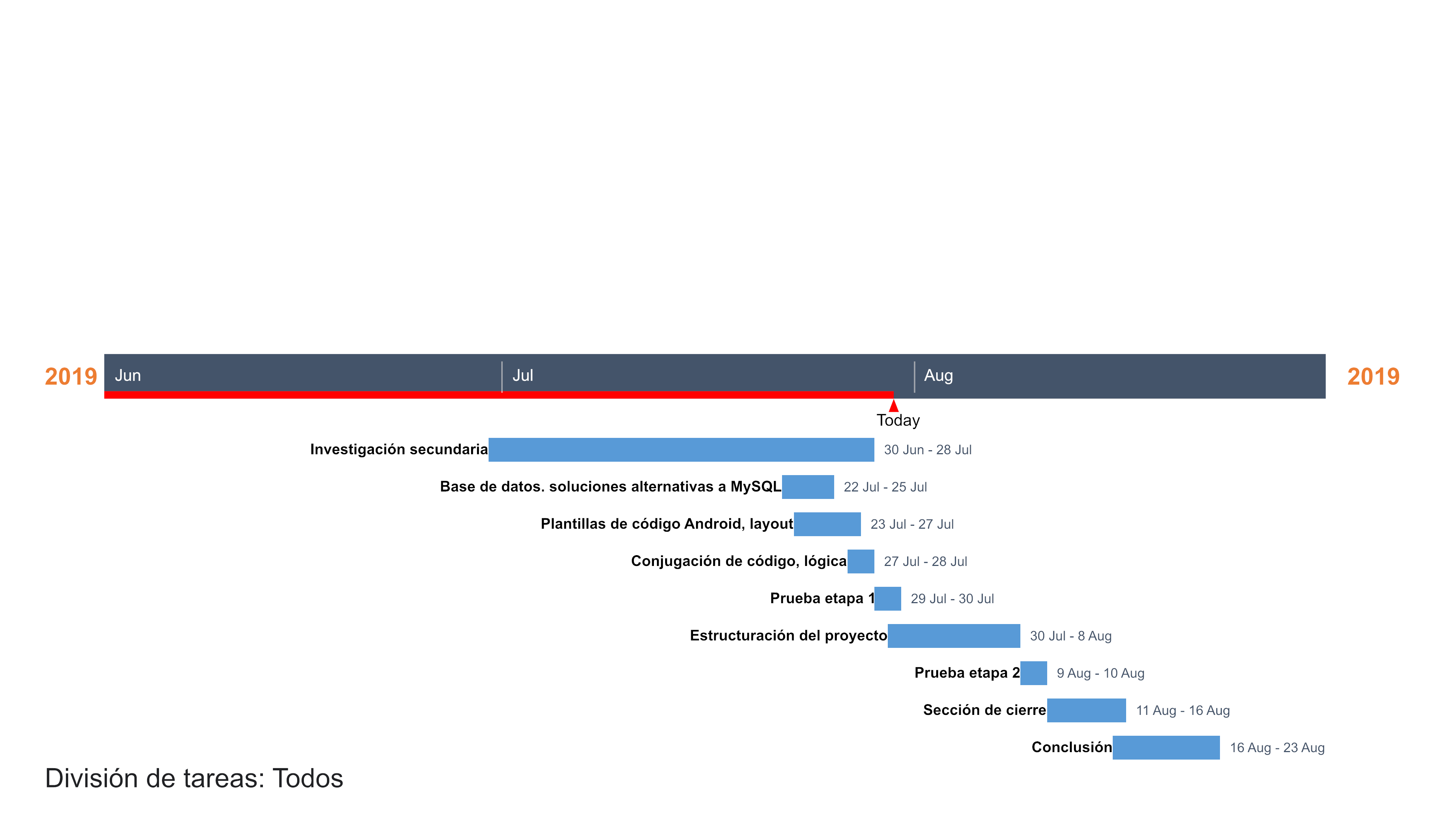


Diagrama Entidad Relación

Este modelo representa a la realidad por medio de un esquema gráfico, identificará los objetos principales del problema a resolver y se diferencian con los otros diagramas que se colocan atributos, se los une por enlaces que muestran la relación que existe entre entidades.

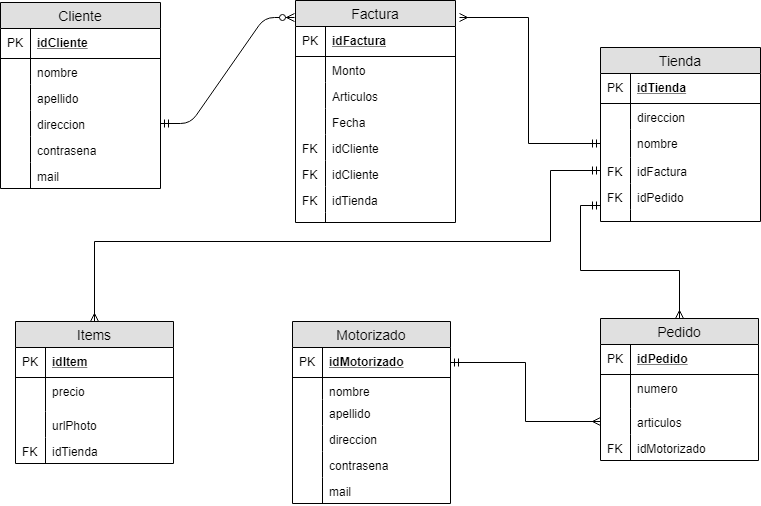


diagrama de casos UML

El diagrama de casos de uso UML es una forma de representar de forma gráfica una vista general de las acciones que pueden realizar los actores dentro del aplicativo. Se lo denomina UML porque viene del Lenguaje de Modelado Unificado.

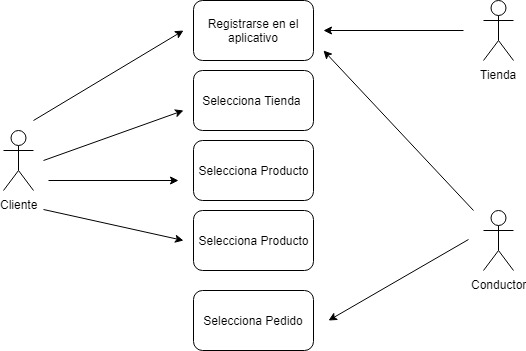
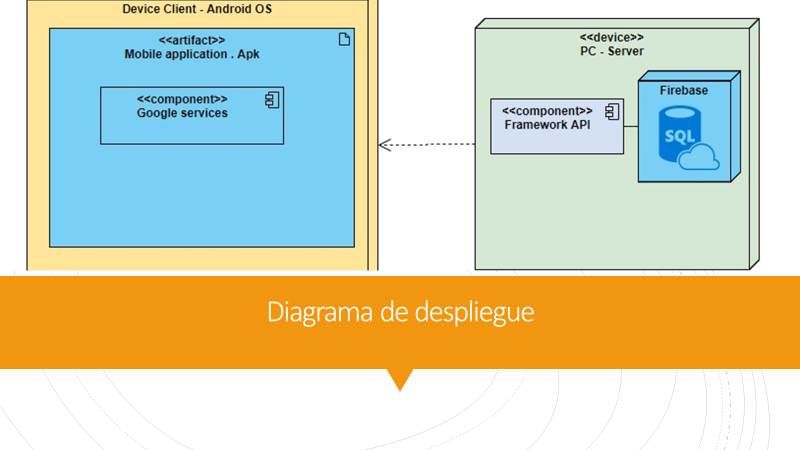


Diagrama de Despliegue

Descripción de los campos, tipos de datos creados en la base de datos.

Base de datos

La base de datos utilizada para el proyecto es Firebase-Database, esta proporciona la información en tiempo real y en forma de árbol json. Es ideal para el desarrollo de aplicaciones porque permite que la información esté sincronizada y almacenada en la nube.

|  |  |  |  |
| --- | --- | --- | --- |
| TABLA | VARIABLE | DESCRIPCIÓN | TIPO DE DATO |
| Pedido | idCliente | Clave única de usuario | String |
| latitud | Latitud Geográfica de la tienda | Double |
| longitud | Longitud Geográfica de la tienda | Double |
| precio | Valor en dólares de los productos comprados | Double |

|  |  |  |  |
| --- | --- | --- | --- |
| TABLA | VARIABLE | DESCRIPCIÓN | TIPO DE DATO |
| Producto | fotoUrl | Fotografía del producto | String |
| nombre | Latitud Geográfica de la tienda | String |
| Precio | Valor en dólares del producto | Double |

|  |  |  |  |
| --- | --- | --- | --- |
| TABLA | VARIABLE | DESCRIPCIÓN | TIPO DE DATO |
| Tienda | latitud | Fotografía del producto | String |
| longitud | Latitud Geográfica de la tienda | String |
| logoURL | Valor en dólares del producto | Double |
| Nombre | nombre de la tienda | String |

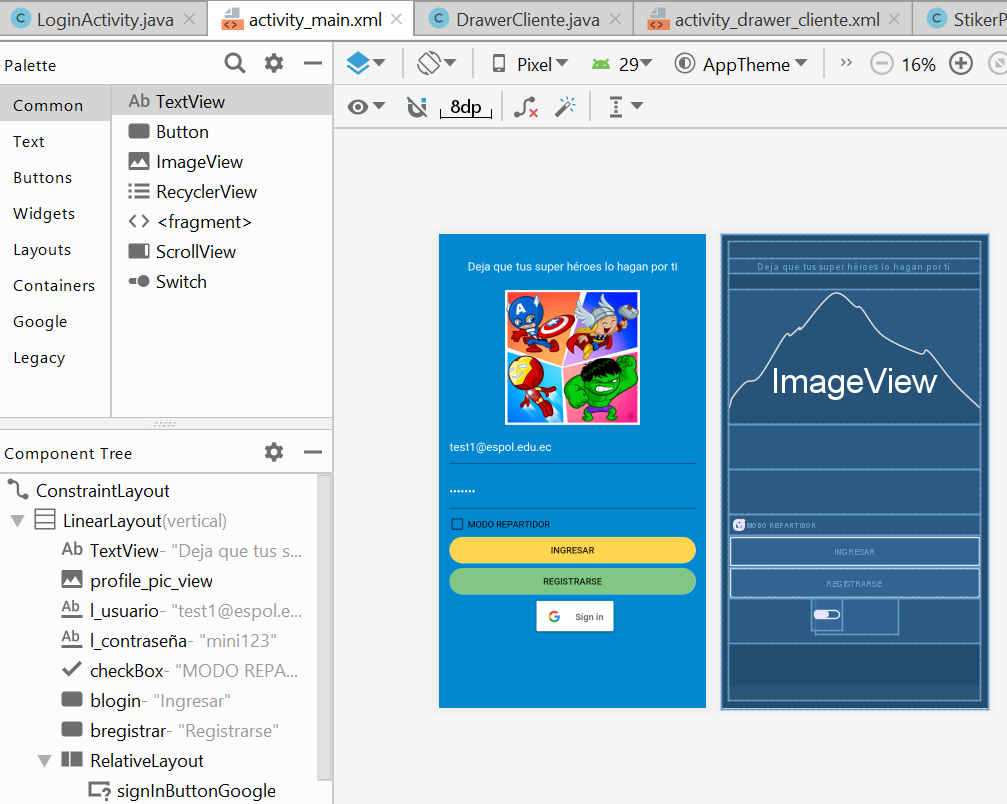
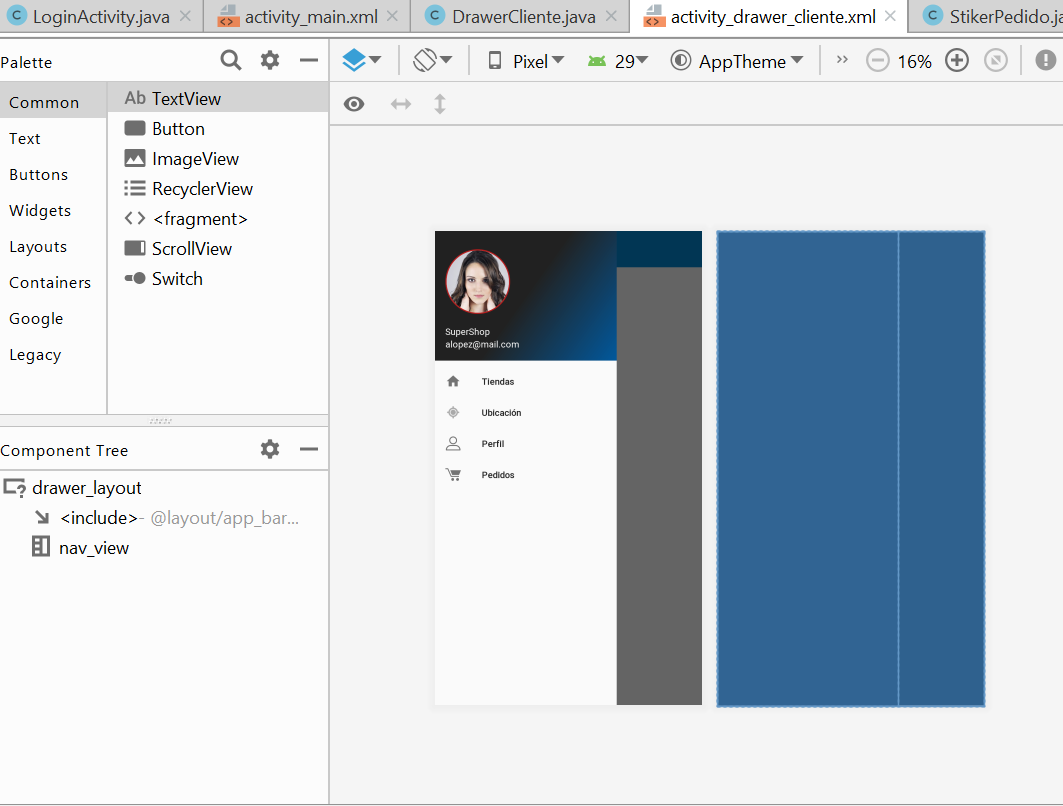
Explicación del código fuente completo desarrollado con los comentarios correspondientes

CLASE DRAWER CLIENTE

Es la pantalla donde se muestra la pantalla principal del cliente, se denomina en el archivo como DrawerCliente porque parte de un tipo de activity Drawer. El objetivo es mantener varias pantallas accesibles a través de un navegador lateral a especie de menú. Además posee métodos que verifica el encedido del GPS del móvil e instancian el objeto “user” para validar si está ya ingresado en el sistema y de no ser el caso, llevar al usuario a la pantalla de Login.

**public class** DrawerCliente **extends** AppCompatActivity  
 **implements** NavigationView.OnNavigationItemSelectedListener,  
 GoogleApiClient.OnConnectionFailedListener{  
 LocationManager **locationManager**;  
 AlertDialog **alert** = **null**;  
  
 *// firebase auth y google api client* **private** GoogleApiClient **googleApiClient**;  
  
 **private** FirebaseAuth **firebaseAuth**;  
 **private** FirebaseAuth.AuthStateListener **firebaseAuthListener**;  
  
 *//Firbase variables* FirebaseDatabase **database**;  
 DatabaseReference **reference**;  
 **private** FragmentTiendas **fragmentTiendas**;  
 **private** pedidosCliente **fragmentPedidos**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_drawer\_cliente***);  
 Toolbar toolbar = findViewById(R.id.***toolbar***);  
 setSupportActionBar(toolbar);  
 DrawerLayout drawer = findViewById(R.id.***drawer\_layout***);  
 NavigationView navigationView = findViewById(R.id.***nav\_view***);  
 ActionBarDrawerToggle toggle = **new** ActionBarDrawerToggle(  
 **this**, drawer, toolbar, R.string.***navigation\_drawer\_open\_cliente***, R.string.***navigation\_drawer\_close\_cliente***);  
 drawer.addDrawerListener(toggle);  
 toggle.syncState();  
 navigationView.setNavigationItemSelectedListener(**this**);  
  
 GoogleSignInOptions gso = **new** GoogleSignInOptions.Builder(GoogleSignInOptions.***DEFAULT\_SIGN\_IN***)  
 .requestEmail()  
 .build();  
  
 **googleApiClient** = **new** GoogleApiClient.Builder(**this**)  
 .enableAutoManage(**this**, **this**)  
 .addApi(Auth.***GOOGLE\_SIGN\_IN\_API***, gso)  
 .build();  
  
 **firebaseAuth** = FirebaseAuth.*getInstance*();  
 *// el oyente llama las acciones pertinentes para que el usuario acceda* **firebaseAuthListener** = **new** FirebaseAuth.AuthStateListener() {  
 @Override  
 **public void** onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {  
 FirebaseUser user = firebaseAuth.getCurrentUser();  
 **if** (user != **null**) {  
 setUserData(user);  
 } **else** {  
 goLogInScreen();  
 }  
 }  
 };  
  
 *// inicializar conexion firebase* inicializarFirebase();  
  
 **fragmentTiendas** = **new** FragmentTiendas(**reference**);  
 **fragmentPedidos** = **new** pedidosCliente(**reference**);  
  
  
 **if**(savedInstanceState==**null**){  
 getSupportFragmentManager().beginTransaction().replace(R.id.***fragment\_container***,  
 **fragmentTiendas**).commit();  
 navigationView.setCheckedItem(R.id.***nav\_home***);  
 }  
 **locationManager** = (LocationManager) getSystemService(***LOCATION\_SERVICE***);  
 **if** ( !**locationManager**.isProviderEnabled( LocationManager.***GPS\_PROVIDER*** ) ) {  
 AlertNoGps();  
 }  
 }  
  
 **private void** setUserData(FirebaseUser user) {  
 *//* ***TODO: Setear usuario e id cliente*** }  
  
 **private void** inicializarFirebase() {  
 FirebaseApp.*initializeApp*(**this**);  
 **database** = FirebaseDatabase.*getInstance*();  
 **reference** = **database**.getReference();  
 *//Toast.makeText(this, "Se inicializó firebase ", Toast.LENGTH\_LONG).show();* }  
  
 @Override  
 **public void** onBackPressed() {  
 DrawerLayout drawer = findViewById(R.id.***drawer\_layout***);  
 **if** (drawer.isDrawerOpen(GravityCompat.***START***)) {  
 drawer.closeDrawer(GravityCompat.***START***);  
 } **else** {  
 **super**.onBackPressed();  
 }  
 }  
  
 @Override  
 **public boolean** onCreateOptionsMenu(Menu menu) {  
 *// Inflate the menu; this adds items to the action bar if it is present.* getMenuInflater().inflate(R.menu.***menu\_cliente***, 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.***logout\_session***) {  
 logOut();  
 **return true**;  
 }  
  
 **return super**.onOptionsItemSelected(item);  
 }  
  
 @SuppressWarnings(**"StatementWithEmptyBody"**)  
 @Override  
 **public boolean** onNavigationItemSelected(MenuItem item) {  
 *// Handle navigation view item clicks here.* **int** id = item.getItemId();  
  
 **if** (id == R.id.***nav\_home***) {  
 getSupportFragmentManager().beginTransaction().replace(R.id.***fragment\_container***,**fragmentTiendas**).commit();  
 } **else if** (id == R.id.***nav\_ubicacion***) {  
 Intent intent=**new** Intent(getApplicationContext(),MapsActivity.**class**);  
 startActivity(intent);  
 } **else if** (id == R.id.***nav\_perfil***) {  
 getSupportFragmentManager().beginTransaction().replace(R.id.***fragment\_container***,**new** FragmentPerfil()).commit();  
 } **else if** (id == R.id.***nav\_pedidos***) {  
 getSupportFragmentManager().beginTransaction().replace(R.id.***fragment\_container***,**fragmentPedidos**).commit();  
  
 }  
  
 DrawerLayout drawer = findViewById(R.id.***drawer\_layout***);  
 drawer.closeDrawer(GravityCompat.***START***);  
 **return true**;  
 }  
  
 **private void** AlertNoGps() {  
 **final** AlertDialog.Builder builder = **new** AlertDialog.Builder(**this**);  
 builder.setMessage(**"El sistema GPS esta desactivado, ¿Desea activarlo?"**)  
 .setCancelable(**false**)  
 .setPositiveButton(**"Si"**, **new** DialogInterface.OnClickListener() {  
 **public void** onClick(@SuppressWarnings(**"unused"**) **final** DialogInterface dialog, @SuppressWarnings(**"unused"**) **final int** id) {  
 startActivity(**new** Intent(android.provider.Settings.***ACTION\_LOCATION\_SOURCE\_SETTINGS***));  
 }  
 })  
 .setNegativeButton(**"No"**, **new** DialogInterface.OnClickListener() {  
 **public void** onClick(**final** DialogInterface dialog, @SuppressWarnings(**"unused"**) **final int** id) {  
 dialog.cancel();  
 }  
 });  
 **alert** = builder.create();  
 **alert**.show();  
  
 }  
  
  
 @Override  
 **protected void** onStart() {  
 **super**.onStart();  
  
 **firebaseAuth**.addAuthStateListener(**firebaseAuthListener**);  
 }  
  
 **private void** goLogInScreen() {  
 Intent intent = **new** Intent(**this**, MainActivity.**class**);  
 intent.addFlags(Intent.***FLAG\_ACTIVITY\_CLEAR\_TOP*** | Intent.***FLAG\_ACTIVITY\_CLEAR\_TASK*** | Intent.***FLAG\_ACTIVITY\_NEW\_TASK***);  
 startActivity(intent);  
 }  
  
 */\*\*  
 \* Accion para cerrar sesion  
 \*/* **public void** logOut() {  
 **firebaseAuth**.signOut();  
  
 Auth.***GoogleSignInApi***.signOut(**googleApiClient**).setResultCallback(**new** ResultCallback<Status>() {  
 @Override  
 **public void** onResult(@NonNull Status status) {  
 **if** (status.isSuccess()) {  
 goLogInScreen();  
 } **else** {  
 Toast.*makeText*(getApplicationContext(), R.string.***not\_close\_session***, Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 });  
 }  
  
 */\*\*  
 \* M'etodo que revoca (o elimina) las credenciales otorgadas a la app  
 \** ***@param view*** *\*/* **public void** revoke(View view) {  
 **firebaseAuth**.signOut();  
  
 Auth.***GoogleSignInApi***.revokeAccess(**googleApiClient**).setResultCallback(**new** ResultCallback<Status>() {  
 @Override  
 **public void** onResult(@NonNull Status status) {  
 **if** (status.isSuccess()) {  
 goLogInScreen();  
 } **else** {  
 Toast.*makeText*(getApplicationContext(), R.string.***not\_revoke***, Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 });  
 }  
  
 @Override  
 **public void** onConnectionFailed(@NonNull ConnectionResult connectionResult) {  
 }  
  
 @Override  
 **protected void** onStop() {  
 **super**.onStop();  
  
 **if** (**firebaseAuthListener** != **null**) {  
 **firebaseAuth**.removeAuthStateListener(**firebaseAuthListener**);  
 }  
 }  
  
}

LAYOUT DE LA CLASE DRAWERCLIENTE

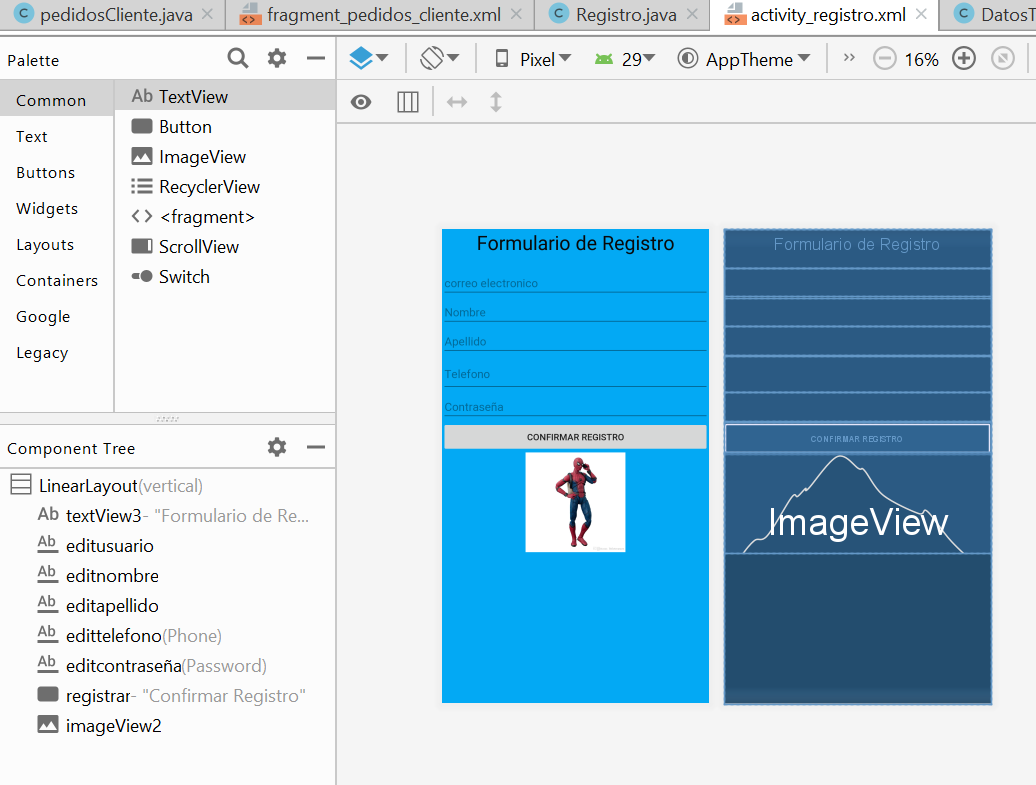
 

**CLASE REGISTRO**

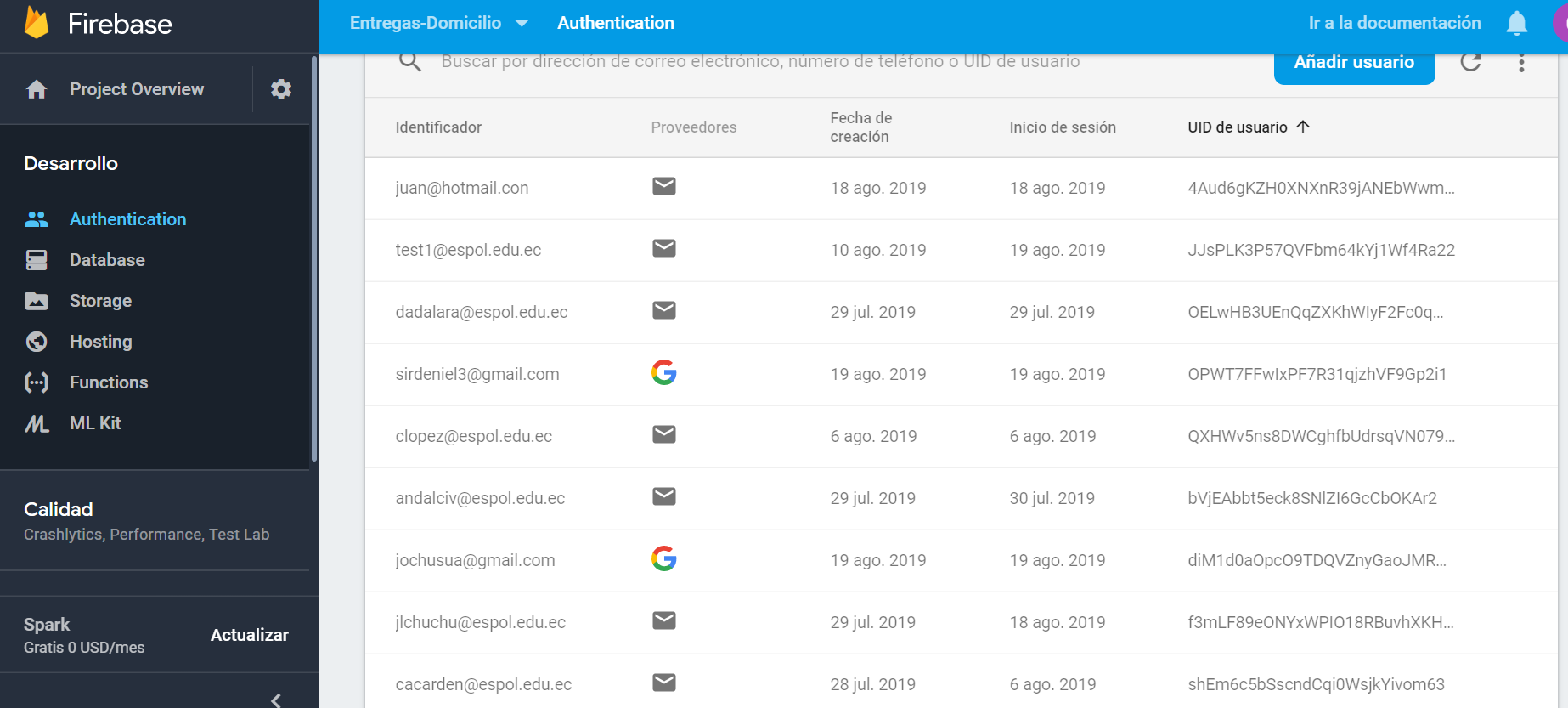
Su objetivo es permitir al usuario crear una cuenta en la base de datos de la aplicación. Implementa condiciones necesarias para un registro normalizado en todos los usuarios.

**public class** Registro **extends** AppCompatActivity **implements** View.OnClickListener {  
  
 *// Permite al ususario agregar su nombre, apellido , telefono y contrasena* EditText **etusuario**, **etnombre**, **etapellido**, **ettelefono**, **etcontraseña**;  
 Button **btnRegistrar**;  
  
 *//Declaramos un objeto firebaseAuth* **private** FirebaseAuth **firebaseAuth**;  
  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_registro***);  
  
 *//inicialización de las variables* **etusuario**=(EditText)findViewById(R.id.***editusuario***);  
 **etnombre** = (EditText)findViewById(R.id.***editnombre***);  
 **etapellido**=(EditText)findViewById(R.id.***editapellido***);  
 **ettelefono** = (EditText)findViewById(R.id.***edittelefono***);  
 **etcontraseña**=(EditText)findViewById(R.id.***editcontraseña***);  
  
 **btnRegistrar** =(Button)findViewById(R.id.***registrar***);  
  
 *//inicializamos el objeto firebaseAuth* **firebaseAuth** = FirebaseAuth.*getInstance*();  
  
 *//attaching listener to button* **btnRegistrar**.setOnClickListener(**this**);  
 }  
  
  
  
 **private void** Registrar(){  
  
 *//Obtenemos el email y la contraseña desde las cajas de texto* String email = **etusuario**.getText().toString().trim();  
 String password = **etcontraseña**.getText().toString().trim();  
  
 *//Verificamos que las cajas de texto no esten vacías* **if**(TextUtils.*isEmpty*(email)){  
 Toast.*makeText*(**this**,**"Se debe ingresar un email"**,Toast.***LENGTH\_LONG***).show();  
 **return**;  
 }  
  
 **if**(TextUtils.*isEmpty*(password)){  
 Toast.*makeText*(**this**,**"Falta ingresar la contraseña"**,Toast.***LENGTH\_LONG***).show();  
 **return**;  
 }  
  
  
  
 *//creating a new user* **firebaseAuth**.createUserWithEmailAndPassword(email, password)  
 .addOnCompleteListener(**this**, **new** OnCompleteListener<AuthResult>() {  
 @Override  
 **public void** onComplete(@NonNull Task<AuthResult> task) {  
 *//checking if success* **if**(task.isSuccessful()){  
 Escritura();  
 Toast.*makeText*(Registro.**this**,**"Se ha registrado el usuario con el email: "**+ **etusuario**.getText(),Toast.***LENGTH\_LONG***).show();  
 }**else**{  
  
 **if** (task.getException() **instanceof** FirebaseAuthUserCollisionException) {*//si se presenta una colisión* Toast.*makeText*(Registro.**this**, **"Ese usuario ya existe "**, Toast.***LENGTH\_SHORT***).show();  
 } **else** {  
 Toast.*makeText*(Registro.**this**, **"No se pudo registrar el usuario "**, Toast.***LENGTH\_LONG***).show();  
 }  
 }  
  
 }  
 });  
  
 }  
  
  
 **public void** Escritura(){  
 DatabaseReference referencia = FirebaseDatabase.*getInstance*().getReference();  
 Usuario usuario = **new** Usuario();  
 usuario.setIdentificacion(**etusuario**.getText().toString());  
 usuario.setNombre(**etnombre**.getText().toString());  
 usuario.setApellido(**etapellido**.getText().toString());  
 **double** celular = Integer.*valueOf*(**ettelefono**.getText().toString());  
 usuario.setTelefono(celular);  
 referencia.child(**"usuarios"**).child(UUID.*randomUUID*().toString()).setValue(usuario);  
 Toast.*makeText*(**this**,**"El registro fue exitoso"**,Toast.***LENGTH\_SHORT***).show();  
 finish();  
 }  
  
 @Override  
 **public void** onClick(View view) {  
 *//Invocamos al método:* Registrar();  
  
 }  
  
  
}

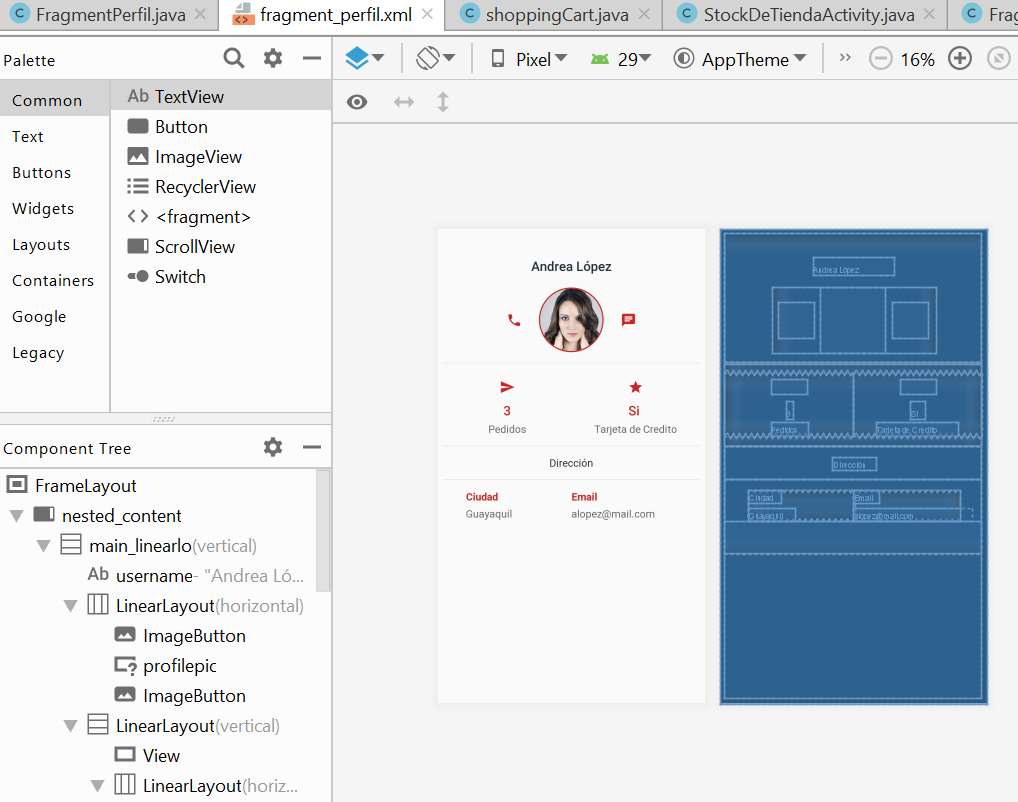
LAYOUT DE LA CLASE REGISTRO



**Base de datos en FIREBASE con los usuarios registrados**



LAYOUT FRAGMENT PERFIL

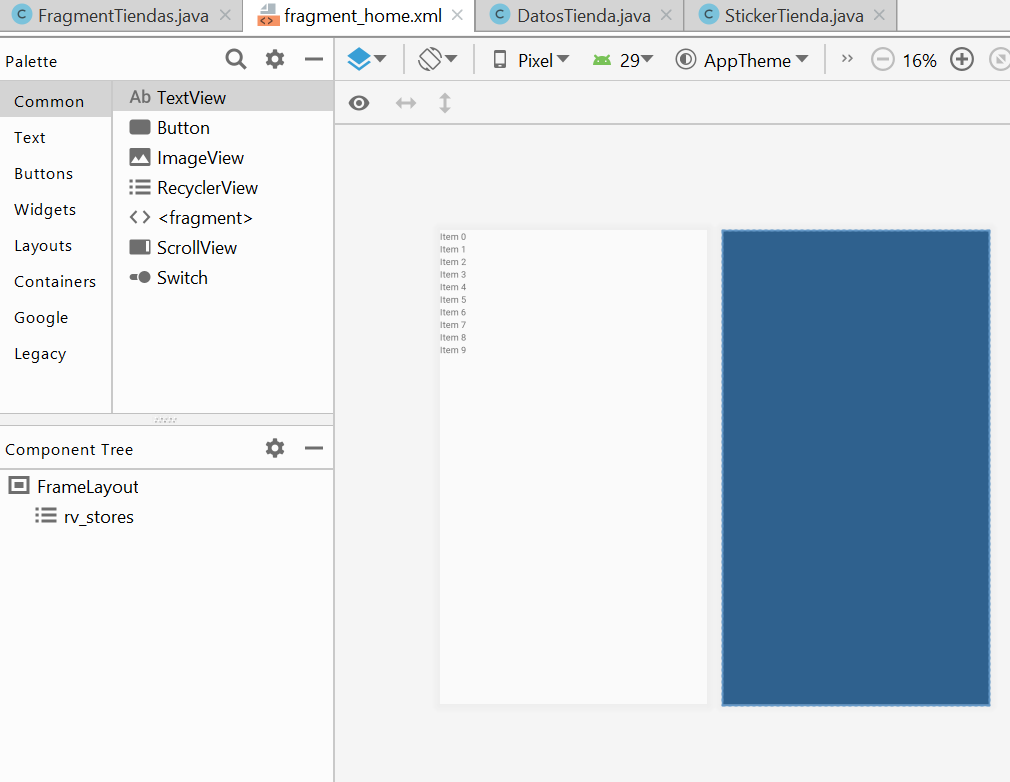


CLASE FRAGMENT TIENDAS

Clase que aparece en una de las pantallas accesibles al cliente, toma la lista de tiendas de la base de datos y los muestra en el RecyclerView que posee, junto con toda la lógica necesaria cuando entra auna de estas tiendas.

**public class** FragmentTiendas **extends** Fragment {  
  
  
 **private** List<DatosTienda> **listStores** = **new** ArrayList<DatosTienda>();  
 ArrayAdapter<DatosTienda> **arrayAdapterStoreUnit**;  
  
 *//private FirebaseDatabase database;* **private** DatabaseReference **refTiendas**;  
 **private** Context **context**;  
 **private** RecyclerView **rv\_stores**;  
 **private** StickerTienda **stickerTienda**;  
  
 **public** FragmentTiendas() {  
 *// Required empty public constructor* }  
  
 **public** FragmentTiendas( DatabaseReference refTiendas) {  
 *//this.database = database;* **this**.**refTiendas** = refTiendas;  
 }  
  
 @Override  
 **public** View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 *// Inflate the layout for this fragment* **final** View inputFragment = inflater.inflate(R.layout.***fragment\_home***, container, **false**);  
 **context** = getContext();  
 **rv\_stores** = (RecyclerView)inputFragment.findViewById(R.id.***rv\_stores***);  
 **rv\_stores**.setLayoutManager(**new** LinearLayoutManager(**context**));  
 **rv\_stores**.setAdapter(**null**);  
  
 *// populate tiendas frome refTiendas* listarTiendas();  
 **return** inputFragment;  
 }  
  
 **private void** listarTiendas() {  
 **refTiendas**.child(**"tiendas"**).addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(@NonNull DataSnapshot dataSnapshot) {  
 **listStores**.clear(); *// para persistencia ?* **for** (DataSnapshot objSnapshot: dataSnapshot.getChildren()){ *// recorre nodo tiendas* DatosTienda su = objSnapshot.getValue(DatosTienda.**class**);  
 su.setUid(objSnapshot.getKey()); *// identificador o uid  
 //Toast.makeText(context, "Tienda>"+su.getNombre(), Toast.LENGTH\_LONG).show();* **listStores**.add(su);  
 }  
 **stickerTienda** = **new** StickerTienda(**listStores**);  
 **rv\_stores**.setAdapter(**stickerTienda**);  
 *//Toast.makeText(context, "stickerTienda", Toast.LENGTH\_LONG).show();* }  
  
 @Override  
 **public void** onCancelled(@NonNull DatabaseError databaseError) {  
 Toast.*makeText*(**context**, databaseError.getMessage(), Toast.***LENGTH\_LONG***).show();  
 }  
 });  
 }  
  
  
}

LAYOUT DE FRAGMENT TIENDAS



CLASE MAPS ACTIVITY

Maps Activity con métodos para mostrar marcadores de tiendas y clientes. Accesible para el conductor y cliente.

**public class** MapsActivity **extends** AppCompatActivity **implements** OnMapReadyCallback {  
  
 @Override  
 **public void** onMapReady ( GoogleMap googleMap ) {  
 Toast.*makeText*( **this**, **"Mapa cargado"**, Toast.***LENGTH\_SHORT*** ).show();  
 Log.*d*( ***TAG***, **"onMapReady: map is ready"** );  
 **mMap** = googleMap;  
 **mMap**.getUiSettings().setZoomControlsEnabled(**true**);  
  
  
 **double** latitude = getIntent().getDoubleExtra(**"latitude"**,0);  
 **double** longitude = getIntent().getDoubleExtra(**"longitude"**,0);  
  
 **if** (latitude == 0 || longitude == 0){  
 *// no es pedido* }  
 **else**{  
 *// si es pedido* LatLng cliente = **new** LatLng(latitude, longitude);  
 **mMap**.setMapType( GoogleMap.***MAP\_TYPE\_NORMAL*** );  
 **mMap**.addMarker( **new** MarkerOptions().position( cliente ).title( **"Cliente"** ).icon( BitmapDescriptorFactory.*defaultMarker*( BitmapDescriptorFactory.***HUE\_GREEN*** ) ) );  
 }  
  
 *//HobbieCars* LatLng hobbiecars = **new** LatLng(-2.154778, -79.892423);  
 **mMap**.setMapType( GoogleMap.***MAP\_TYPE\_NORMAL*** );  
 **mMap**.addMarker(**new** MarkerOptions().position(hobbiecars).title(**"Hobbiecars Mall del Sol"**));  
  
 *//GeekStore* LatLng geekstore = **new** LatLng(-2.154969, -79.891575);  
 **mMap**.setMapType( GoogleMap.***MAP\_TYPE\_NORMAL*** );  
 **mMap**.addMarker(**new** MarkerOptions().position(geekstore).title(**"GeekStore Mall del Sol"**));  
  
 *//Funko* LatLng funko = **new** LatLng( -2.109114, -79.948691 );  
 **mMap**.addMarker( **new** MarkerOptions().position( funko ).title( **"Funko Mall El Fortin"** ).icon( BitmapDescriptorFactory.*defaultMarker*( BitmapDescriptorFactory.***HUE\_VIOLET*** ) ) );  
 **mMap**.moveCamera( CameraUpdateFactory.*newLatLng*( funko ) );  
  
 *//Zacatrus* LatLng zacatrus = **new** LatLng( -2.228982, -79.898401);  
 **mMap**.addMarker( **new** MarkerOptions().position( zacatrus ).title( **"Zacatrus Mall del Sur"** ).icon( BitmapDescriptorFactory.*defaultMarker*( BitmapDescriptorFactory.***HUE\_VIOLET*** ) ) );  
 **mMap**.moveCamera( CameraUpdateFactory.*newLatLng*( zacatrus ) );  
  
 *//GeekStore2* LatLng geekstore2 = **new** LatLng(-2.176001, -79.943634);  
 **mMap**.setMapType( GoogleMap.***MAP\_TYPE\_NORMAL*** );  
 **mMap**.addMarker(**new** MarkerOptions().position(geekstore2).title(**"GeekStore Riocentro Ceibos"**));  
  
 **if** (**mLocationPermissionsGranted**) {  
 getDeviceLocation();  
  
 **if** (ActivityCompat.*checkSelfPermission*( **this**, Manifest.permission.***ACCESS\_FINE\_LOCATION*** ) != PackageManager.***PERMISSION\_GRANTED*** && ActivityCompat.*checkSelfPermission*( **this**, Manifest.permission.***ACCESS\_COARSE\_LOCATION*** ) != PackageManager.***PERMISSION\_GRANTED***) {  
 **return**;  
 }  
 **mMap**.setMyLocationEnabled( **true** );  
  
 }  
 }  
  
 **private static final** String ***TAG*** = **"MapActivity"**;  
  
 **private static final** String ***FINE\_LOCATION*** = Manifest.permission.***ACCESS\_FINE\_LOCATION***;  
 **private static final** String ***COURSE\_LOCATION*** = Manifest.permission.***ACCESS\_COARSE\_LOCATION***;  
 **private static final int *LOCATION\_PERMISSION\_REQUEST\_CODE*** = 1234;  
 **private static final float *DEFAULT\_ZOOM*** = 15f;  
  
 *//vars* **private** Boolean **mLocationPermissionsGranted** = **false**;  
 **private** GoogleMap **mMap**;  
  
  
 @Override  
 **protected void** onCreate ( @Nullable Bundle savedInstanceState ) {  
 **super**.onCreate( savedInstanceState );  
 setContentView( R.layout.***activity\_maps***);  
 getLocationPermission();  
 }  
  
 **private void** getDeviceLocation () {  
 Log.*d*( ***TAG***, **"getDeviceLocation: getting the devices current location"** );  
  
 FusedLocationProviderClient mFusedLocationProviderClient = LocationServices.*getFusedLocationProviderClient*(**this**);  
  
 **try** {  
 **if** (**mLocationPermissionsGranted**) {  
  
 **final** Task location = mFusedLocationProviderClient.getLastLocation();  
 location.addOnCompleteListener( **new** OnCompleteListener() {  
 @Override  
 **public void** onComplete ( @NonNull Task task ) {  
 **if** (task.isSuccessful()) {  
 Log.*d*( ***TAG***, **"onComplete: found location!"** );  
 Location currentLocation = (Location) task.getResult();  
  
 moveCamera( **new** LatLng( currentLocation.getLatitude(), currentLocation.getLongitude() ), ***DEFAULT\_ZOOM*** );  
  
 } **else** {  
 Log.*d*( ***TAG***, **"onComplete: current location is null"** );  
 Toast.*makeText*( MapsActivity.**this**, **"unable to get current location"**, Toast.***LENGTH\_SHORT*** ).show();  
 }  
 }  
 } );  
 }  
 } **catch** (SecurityException e) {  
 Log.*e*( ***TAG***, **"getDeviceLocation: SecurityException: "** + e.getMessage() );  
 }  
 }  
  
 **private void** moveCamera ( LatLng latLng, **float** zoom ) {  
 Log.*d*( ***TAG***, **"moveCamera: moving the camera to: lat: "** + latLng.**latitude** + **", lng: "** + latLng.**longitude** );  
 CameraPosition cameraPosition = **new** CameraPosition.Builder()  
 .target(latLng) *// Sets the center of the map to User Location View* .zoom(15) *// Sets the zoom* .bearing(0) *// Sets the orientation of the camera to east* .tilt(30) *// Sets the tilt of the camera to 30 degrees* .build(); *// Creates a CameraPosition from the builder* **mMap**.animateCamera(CameraUpdateFactory.*newCameraPosition*(cameraPosition));  
 **mMap**.getUiSettings().setMyLocationButtonEnabled(**true** );  
 }  
  
 **private void** initMap () {  
 Log.*d*( ***TAG***, **"initMap: initializing map"** );  
 SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById( R.id.***map*** );  
  
 mapFragment.getMapAsync( MapsActivity.**this** );  
 }  
  
 **private void** getLocationPermission () {  
 Log.*d*( ***TAG***, **"getLocationPermisszion: getting location permissions"** );  
 String[] permissions = {Manifest.permission.***ACCESS\_FINE\_LOCATION***, Manifest.permission.***ACCESS\_COARSE\_LOCATION***};  
  
 **if** (ContextCompat.*checkSelfPermission*( **this**.getApplicationContext(), ***FINE\_LOCATION*** ) == PackageManager.***PERMISSION\_GRANTED***) {  
 **if** (ContextCompat.*checkSelfPermission*( **this**.getApplicationContext(), ***COURSE\_LOCATION*** ) == PackageManager.***PERMISSION\_GRANTED***) {  
 **mLocationPermissionsGranted** = **true**;  
 initMap();  
 } **else** {  
 ActivityCompat.*requestPermissions*( **this**, permissions, ***LOCATION\_PERMISSION\_REQUEST\_CODE*** );  
 }  
 } **else** {  
 ActivityCompat.*requestPermissions*( **this**, permissions, ***LOCATION\_PERMISSION\_REQUEST\_CODE*** );  
 }  
 }  
  
 @Override  
 **public void** onRequestPermissionsResult ( **int** requestCode, @NonNull String[] permissions, @NonNull **int**[] grantResults ) {  
 Log.*d*( ***TAG***, **"onRequestPermissionsResult: called."** );  
 **mLocationPermissionsGranted** = **false**;  
  
 **switch** (requestCode) {  
 **case *LOCATION\_PERMISSION\_REQUEST\_CODE***: {  
 **if** (grantResults.**length** > 0) {  
 **for** (**int** i = 0; i < grantResults.**length**; i++) {  
 **if** (grantResults[i] != PackageManager.***PERMISSION\_GRANTED***) {  
 **mLocationPermissionsGranted** = **false**;  
 Log.*d*( ***TAG***, **"onRequestPermissionsResult: permission failed"** );  
 **return**;  
 }  
 }  
 Log.*d*( ***TAG***, **"onRequestPermissionsResult: permission granted"** );  
 **mLocationPermissionsGranted** = **true**;  
 *//initialize our map* initMap();  
 }  
 }  
 }  
 }  
}

CLASE PEDIDOS CLIENTE

Muestra los pedidos que realiza el cliente los cuales se registran en la base de datos, es muy similar a la clase FRAGMENT TIENDAS porque tienen un propósito: mostrar información a través de “stickers”.

**public class** pedidosCliente **extends** Fragment {  
  
 **private** List<Pedido> **listPedidos** = **new** ArrayList<Pedido>();  
 ArrayAdapter<Pedido> **arrayAdapterStoreUnit**;  
  
  
 **private** DatabaseReference **refPedidos**;  
 **private** Context **context**;  
 **private** RecyclerView **rv\_stores**;  
 **private** StikerPedido **stickerPedido**;  
  
 **public** pedidosCliente() {  
 *// Required empty public constructor* }  
  
 **public** pedidosCliente( DatabaseReference refPedidos) {  
 *//this.database = database;* **this**.**refPedidos** = refPedidos;  
 }  
  
 @Override  
 **public** View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 *// Inflate the layout for this fragment  
 //return inflater.inflate(R.layout.fragment\_pedidos\_cliente, container, false);* **final** View inputFragment = inflater.inflate(R.layout.***fragment\_pedidos\_cliente***, container, **false**);  
 **context** = getContext();  
  
 **rv\_stores** = (RecyclerView)inputFragment.findViewById(R.id.***rv\_stores2***);  
 **rv\_stores**.setLayoutManager(**new** LinearLayoutManager(**context**));  
 **rv\_stores**.setAdapter(**null**);  
  
 listarPedidos();  
  
  
 **return** inputFragment;  
  
 }  
  
 **private void** listarPedidos() {  
 **refPedidos**.child(**"pedidos"**).addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(@NonNull DataSnapshot dataSnapshot) {  
 **listPedidos**.clear(); *// para persistencia ?* **for** (DataSnapshot objSnapshot: dataSnapshot.getChildren()){ *// recorre nodo pedidos* Pedido pedido = objSnapshot.getValue(Pedido.**class**);  
 pedido.setUid(objSnapshot.getKey()); *// identificador o uid  
 //Toast.makeText(context, "pedido>"+su.getNombre(), Toast.LENGTH\_LONG).show();* **listPedidos**.add(pedido);  
 }  
 **stickerPedido** = **new** StikerPedido(**listPedidos**);  
 **rv\_stores**.setAdapter(**stickerPedido**);  
 *//Toast.makeText(context, "stickerTienda", Toast.LENGTH\_LONG).show();* }  
  
 @Override  
 **public void** onCancelled(@NonNull DatabaseError databaseError) {  
 Toast.*makeText*(**context**, databaseError.getMessage(), Toast.***LENGTH\_LONG***).show();  
 }});}}

LAYOUT STICKER PEDIDO (DENTRO DE CLASE PEDIDOS CLIENTE)

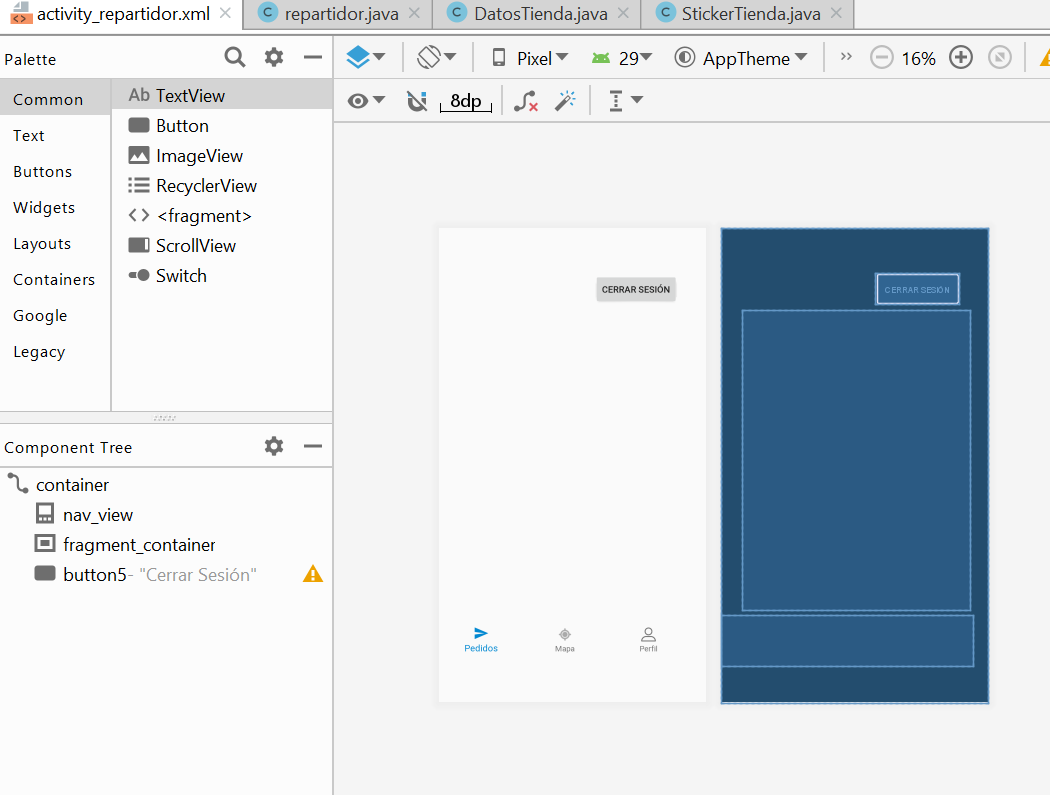
****

CLASE REPARTIDOR

Esta interfaz se abre cuando marca la casilla “Repartidor” en la pantalla de inicio de sesión. Es similar a DrawerCliente.java en los aspectos funcionales y pantallas.

**public class** repartidor **extends** AppCompatActivity {  
 **private** TextView **mTextMessage**;  
 **private** Fragment **selectedFragment**=**null**;  
 FirebaseDatabase **database**;  
 DatabaseReference **reference**;  
 **private** pedidosCliente **fragmentPedidos**;  
 LocationManager **locationManager**;  
 AlertDialog **alert** = **null**;  
  
 **private** BottomNavigationView.OnNavigationItemSelectedListener **mOnNavigationItemSelectedListener** = **new** BottomNavigationView.OnNavigationItemSelectedListener() {  
  
 @Override  
 **public boolean** onNavigationItemSelected(@NonNull MenuItem item) {  
 **switch** (item.getItemId()) {  
 **case** R.id.***navigation\_home***:  
 **selectedFragment**=**fragmentPedidos**;  
 **break**;  
 **case** R.id.***navigation\_dashboard***:  
 Intent intent=**new** Intent(getApplicationContext(),MapsActivity.**class**);  
 startActivity(intent);  
 **case** R.id.***navigation\_notifications***:  
 **selectedFragment**=**new** FragmentPerfil();  
 **break**;  
 }  
 getSupportFragmentManager().beginTransaction().replace(R.id.***fragment\_container***,**selectedFragment**).commit();  
 **return true**;  
 }  
 };  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_repartidor***);  
 BottomNavigationView navView = findViewById(R.id.***nav\_view***);  
 **mTextMessage** = findViewById(R.id.***message***);  
 navView.setOnNavigationItemSelectedListener(**mOnNavigationItemSelectedListener**);  
 inicializarFirebase();  
 **fragmentPedidos** = **new** pedidosCliente(**reference**);  
 getSupportFragmentManager().beginTransaction().replace(R.id.***fragment\_container***,**fragmentPedidos**).commit();  
 **locationManager** = (LocationManager) getSystemService(***LOCATION\_SERVICE***);  
 **if** ( !**locationManager**.isProviderEnabled( LocationManager.***GPS\_PROVIDER*** ) ) {  
 AlertNoGps();  
 }  
 }  
  
 **private void** inicializarFirebase() {  
 FirebaseApp.*initializeApp*(**this**);  
 **database** = FirebaseDatabase.*getInstance*();  
 **reference** = **database**.getReference();  
 *//Toast.makeText(this, "Se inicializó firebase ", Toast.LENGTH\_LONG).show();* }  
  
 **private void** AlertNoGps() {  
 **final** AlertDialog.Builder builder = **new** AlertDialog.Builder(**this**);  
 builder.setMessage(**"El sistema GPS esta desactivado, ¿Desea activarlo?"**)  
 .setCancelable(**false**)  
 .setPositiveButton(**"Si"**, **new** DialogInterface.OnClickListener() {  
 **public void** onClick(@SuppressWarnings(**"unused"**) **final** DialogInterface dialog, @SuppressWarnings(**"unused"**) **final int** id) {  
 startActivity(**new** Intent(android.provider.Settings.***ACTION\_LOCATION\_SOURCE\_SETTINGS***));  
 }  
 })  
 .setNegativeButton(**"No"**, **new** DialogInterface.OnClickListener() {  
 **public void** onClick(**final** DialogInterface dialog, @SuppressWarnings(**"unused"**) **final int** id) {  
 dialog.cancel();  
 }  
 });  
 **alert** = builder.create();  
 **alert**.show();  
  
 }  
 **public void** cerrarSesion(View view) {  
 finish();  
 }  
}

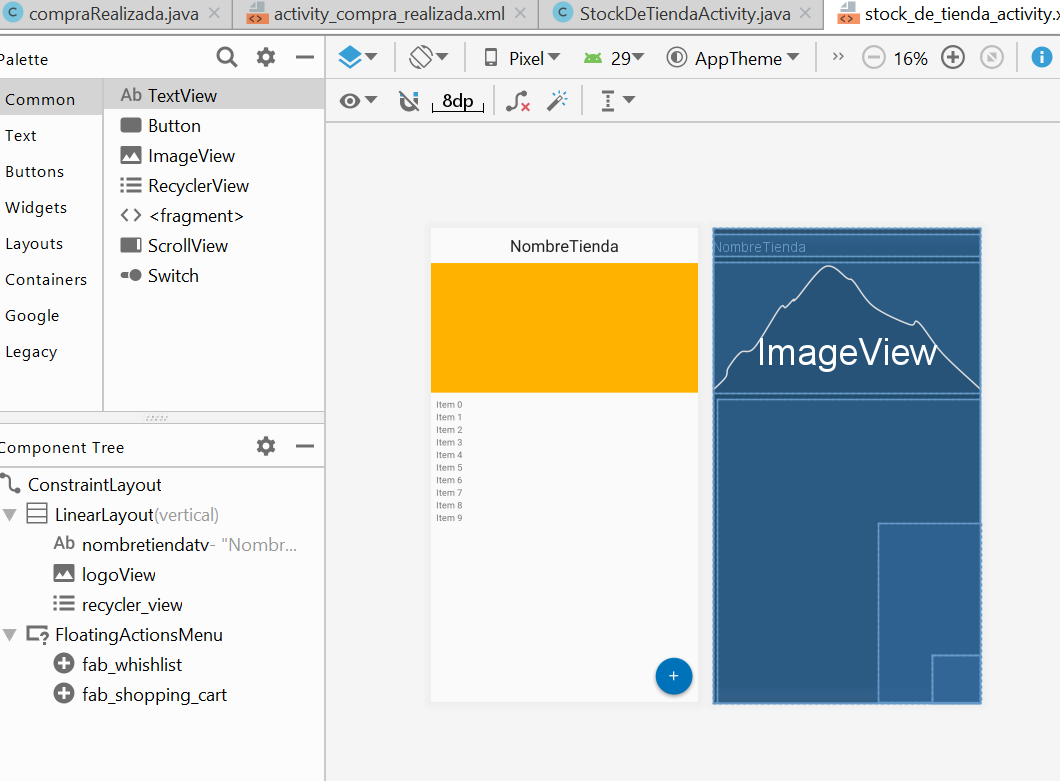
LAYOUT REPARTIDOR



CLASE STOCK DE TIENDA ACTIVITY

En esta clase se muestra métodos para mostrar el stock de la tienda seleccionada que se carga desde base de datos.

**public class** StockDeTiendaActivity **extends** AppCompatActivity {  
  
 **private** List<DatosItem> **listItems** = **new** ArrayList<DatosItem>();  
 **private** List<DatosItem> **listItemsSelected** = **new** ArrayList<DatosItem>();  
 **private** RecyclerView **mRecyclerView**;  
 **private** DatabaseReference **reference**;  
 **private** String **logoUrl**;  
 String **nombreTienda**;  
 **private** Context **context**;  
  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***stock\_de\_tienda\_activity***);  
  
 ImageView logoView = (ImageView) findViewById(R.id.***logoView***);  
 TextView nombretiendaTV = (TextView) findViewById(R.id.***nombretiendatv***);  
  
  
 **mRecyclerView** = (RecyclerView) findViewById(R.id.***recycler\_view***);  
 LinearLayoutManager manager = **new** LinearLayoutManager(StockDeTiendaActivity.**this**);  
 *//mRecyclerView.setHasFixedSize(true);* **mRecyclerView**.setLayoutManager(manager);  
 **mRecyclerView**.setAdapter(**null**);  
  
 **logoUrl** = getIntent().getStringExtra(**"URL\_LOGO"**);  
 **nombreTienda** = getIntent().getStringExtra(**"NOMBRE\_TIENDA"**);  
 **final** String uidTienda = getIntent().getStringExtra(**"UID\_TIENDA"**);  
  
  
 nombretiendaTV.setText(**nombreTienda**);  
  
 Glide.*with*(**this**)  
 .load(**logoUrl**)  
 .fitCenter()  
 .placeholder(R.drawable.***image\_placeholder***)  
 .error(R.drawable.***broken\_image\_blue***)  
 .fallback(R.drawable.***broken\_image\_blue***)  
 .into(logoView)  
 ;  
  
 inicializarFirebase();  
 listarProductos();  
 }  
  
 **private void** listarProductos() {  
 **reference**.child(**"producto"**).addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(@NonNull DataSnapshot dataSnapshot) {  
 **listItems**.clear(); *// para persistencia ?* **for** (DataSnapshot objSnapshot: dataSnapshot.getChildren()){ *// recorre nodo tiendas* DatosItem di = objSnapshot.getValue(DatosItem.**class**);  
 *//Toast.makeText(context, "Tienda>"+su.getNombre(), Toast.LENGTH\_LONG).show();* **listItems**.add(di);  
 }  
 **mRecyclerView**.setAdapter(**new** StickerItem(**listItems**));  
 *//Toast.makeText(context, "stickerTienda", Toast.LENGTH\_LONG).show();* }  
  
 @Override  
 **public void** onCancelled(@NonNull DatabaseError databaseError) {  
 *//Toast.makeText(context, databaseError.getMessage(), Toast.LENGTH\_LONG).show();* }  
 });  
 }  
  
 **private void** inicializarFirebase() {  
 FirebaseApp.*initializeApp*(**this**);  
 FirebaseDatabase database = FirebaseDatabase.*getInstance*();  
 **reference** = database.getReference();  
 *//Toast.makeText(this, "Se inicializó firebase ", Toast.LENGTH\_LONG).show();* }  
  
 **public void** ingresarShoppingCart(View view) {  
 Intent i = **new** Intent(**this**, shoppingCart.**class** );  
 **listItemsSelected**.clear();  
 **for**(DatosItem d:**listItems**){  
 **if**(d.isSelected()){  
 **listItemsSelected**.add(d);  
 }  
 }  
 **if**(**listItemsSelected**.isEmpty()){  
 Toast.*makeText*( **this**, **"No posee productos seleccionados"**, Toast.***LENGTH\_SHORT*** ).show();  
 }**else**{  
 i.putExtra(**"itemsSelecccionados"**, (Serializable) **listItemsSelected**);  
 i.putExtra(**"NOMBRETIENDA"**,(Serializable) **nombreTienda**);  
 i.putExtra(**"URL\_LOGO1"**,(Serializable) **logoUrl**);  
 startActivity(i);  
 }  
 }  
}

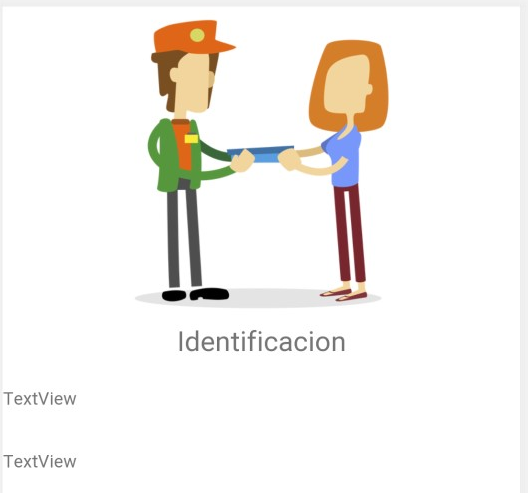


CLASE STICKER PEDIDO

Guarda en una lista todos los objetos “pedido”, posee estados de la compra realizada por el cliente como “Entregado” y “Pendiende”. Da paso a MapsActivity que muestra la ubicación del cliente para realizar el encargo. Solo el repartidor aceptado tiene acceso.

**public class** StikerPedido **extends** RecyclerView.Adapter<StikerPedido.MyViewHolder>{  
  
 **private** List<Pedido> **listPedidos**;  
 **private** Context **context**;  
  
 **public** StikerPedido(List<Pedido> listPedidos) {  
 **this**.**listPedidos** = listPedidos;  
 }  
  
 @NonNull  
 @Override  
 **public** StikerPedido.MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, **int** viewType) {  
 **context** = parent.getContext();  
 View view = LayoutInflater.*from*(**context**).inflate(R.layout.***sticker\_pedido***, parent, **false**);  
 **return new** StikerPedido.MyViewHolder(view);  
 }  
  
 @Override  
 **public void** onBindViewHolder(@NonNull StikerPedido.MyViewHolder holder, **int** position) {  
 **final** Pedido pedidos = **listPedidos**.get(position);  
  
  
 String pre= Double.*toString*(pedidos.getPrecio());  
 holder.**identificacion**.setText(**"Cliente: "**+pedidos.getId\_cliente());  
 holder.**precio**.setText(**"Tienda: "**+pedidos.getNombreTienda()+**" Precio: "**+pre+**" Fecha: "**+pedidos.getFecha().toString());  
 holder.**estado**.setText(pedidos.getEstado());  
  
 **if** (pedidos.getEstado().equalsIgnoreCase(**"pendiente"**)){  
 holder.**storePhotoview**.setImageResource(R.drawable.***pendiente2***);  
 }  
  
  
 *// cuando realice click en el logo de tienda, se abrirá StockDeTienda* holder.**storePhotoview**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **if** (pedidos.getEstado().equalsIgnoreCase(**"Entregado"**)){  
 Toast.*makeText*(**context**,**"Ya se realizó este encargo"**,Toast.***LENGTH\_SHORT***).show();  
  
 }**else if** (pedidos.getEstado().equalsIgnoreCase(**"Pendiente"**)){  
 DatabaseReference referencia = FirebaseDatabase.*getInstance*().getReference();  
 Toast.*makeText*(**context**,**"El encargo es aprobado"**,Toast.***LENGTH\_LONG***).show();  
 pedidos.setEstado(**"Entregado"**);  
 referencia.child(**"pedidos"**).child(pedidos.getUid()).setValue(pedidos);  
  
 *//* ***todo: iniciar mapsactivity*** Intent i = **new** Intent(**context**, MapsActivity.**class**);  
 i.putExtra(**"latitude"**,pedidos.getLatitud());  
 i.putExtra(**"longitude"**,pedidos.getLongitud());  
 **context**.startActivity(i);  
 }  
 }  
 });  
  
 }  
  
 @Override  
 **public int** getItemCount() {  
 **return listPedidos** == **null** ? 0 : **listPedidos**.size();  
 }  
  
 **class** MyViewHolder **extends** RecyclerView.ViewHolder {  
  
 **private** View **view**;  
 **private** ImageView **storePhotoview**;  
 **private** TextView **identificacion**;  
 **private** TextView **precio**;  
 **private** TextView **estado**;  
  
  
 **private** MyViewHolder(View itemView) {  
 **super**(itemView);  
 **view** = itemView; *// view o "layout" de stickerTienda  
 // cargar atributos a la clase* **storePhotoview** = (ImageView) itemView.findViewById(R.id.***store\_photoview***);  
 **identificacion** = (TextView) itemView.findViewById(R.id.***id\_pedido***);  
 **precio** = (TextView) itemView.findViewById(R.id.***text\_precio***);  
 **estado** = (TextView) itemView.findViewById(R.id.***textestado***);  
 }  
 }  
}

LAYOUT DE STICKER PEDIDO

****

CLASE STICKER TIENDA

Administra las tiendas obtenidas de la base de datos, pues una vez obtenida la información obtiene la imagen de la tienda desde una URL almacenada en la base de datos, en caso de un error en la misma, coloca una imagen plantilla. Da acceso a “shoppingCart.java”.

**public class** StickerTienda **extends** RecyclerView.Adapter<StickerTienda.MyViewHolder> {  
  
 **private** List<DatosTienda> **listTiendas**;  
 **private** Context **context**;  
  
 **public** StickerTienda(List<DatosTienda> listTiendas) {  
 **this**.**listTiendas** = listTiendas;  
 }  
  
 @NonNull  
 @Override  
 **public** MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, **int** viewType) {  
 **context** = parent.getContext();  
 View view = LayoutInflater.*from*(**context**).inflate(R.layout.***sticker\_tienda***, parent, **false**);  
 **return new** MyViewHolder(view);  
 }  
  
 @Override  
 **public void** onBindViewHolder(@NonNull **final** MyViewHolder holder, **int** position) {  
 **final** DatosTienda datosTienda = **listTiendas**.get(position);  
 *// cargar imagen desde la url, no cuelga la UI porque funciona como un threading* Glide.*with*(**context**)  
 .load(datosTienda.getLogourl())  
 .fitCenter()  
 .placeholder(R.drawable.***image\_placeholder***)  
 .error(R.drawable.***broken\_image\_blue***)  
 .fallback(R.drawable.***broken\_image\_blue***)  
 .into(holder.**storePhotoview**)  
 ;  
  
 holder.**nombre**.setText(datosTienda.getNombre());  
 *// cuando realice click en el logo de tienda, se abrirá StockDeTienda* holder.**storePhotoview**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 *//* ***TODO: abrir "stock personal" a traves de atributos en datosTienda*** Intent intent=**new** Intent(**context**, StockDeTiendaActivity.**class**);  
 intent.putExtra(**"URL\_LOGO"**,datosTienda.getLogourl());  
 intent.putExtra(**"UID\_TIENDA"**,datosTienda.getUid());  
 intent.putExtra(**"NOMBRE\_TIENDA"**,datosTienda.getNombre());  
 **context**.startActivity(intent);  
 }  
 });  
 }  
  
 @Override  
 **public int** getItemCount() {  
 **return listTiendas** == **null** ? 0 : **listTiendas**.size();  
 }  
  
 *// antes esto era public* **class** MyViewHolder **extends** RecyclerView.ViewHolder {  
  
 **private** View **view**;  
 **private** ImageView **storePhotoview**;  
 **private** TextView **nombre**;  
  
 **private** MyViewHolder(View itemView) {  
 **super**(itemView);  
 **view** = itemView; *// view o "layout" de stickerTienda  
 // cargar atributos a la clase* **storePhotoview** = (ImageView) itemView.findViewById(R.id.***store\_photoview***);  
 **nombre** = (TextView) itemView.findViewById(R.id.***nombreTV***);  
  
 }  
 }  
}

LAYOUT DE STICKER TIENDA (DENTRO DE LAYOUT DE FRAGMENT TIENDAS)

****

CLASE STICKER ITEM

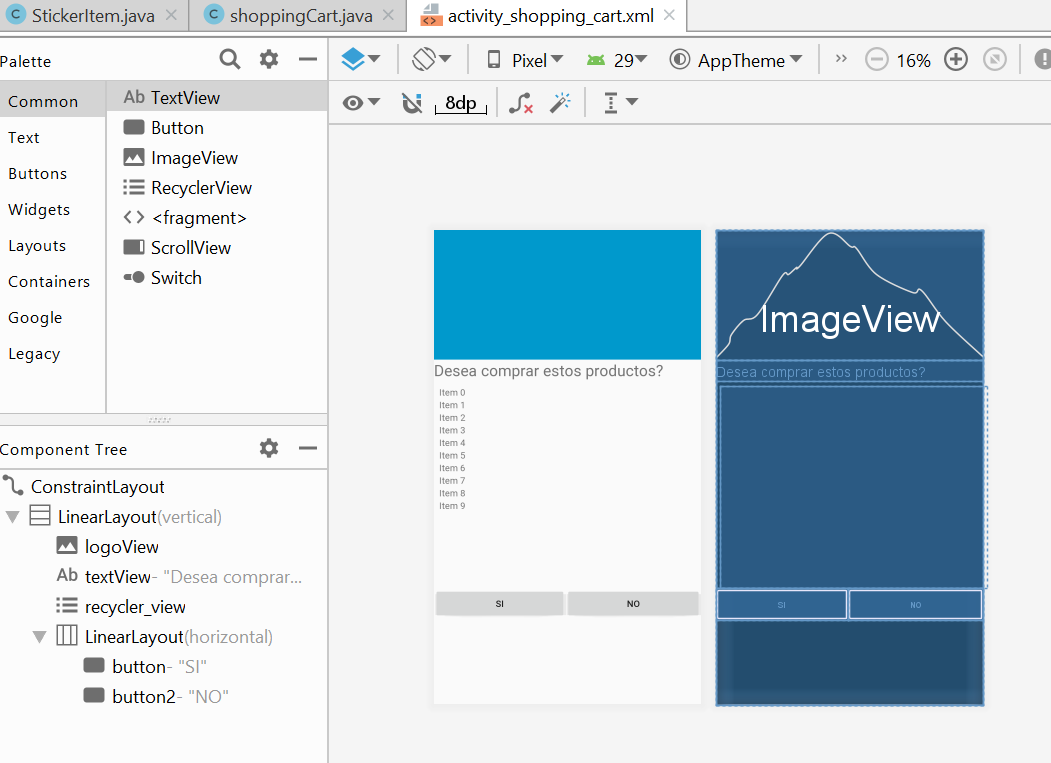
Clase Sticker Item con métodos para el manejo de acciones en los productos que se cargan desde base de datos y se muestran en Recyclerview. Carga de manera correcta las imágenes y precio del producto. En la clase DatosItem almacena un booleano si esta seleccionado.

**public class** StickerItem **extends** RecyclerView.Adapter<StickerItem.MyViewHolder> {  
  
 **private** List<DatosItem> **listDatosItem**;  
 **private int amber**;  
 **private** Context **context**;  
  
 *// here, this func was public* StickerItem(List<DatosItem> datosItemList) {  
 **listDatosItem** = datosItemList;  
 }  
  
 @Override  
 **public** MyViewHolder onCreateViewHolder(ViewGroup parent, **int** viewType) {  
 **context** = parent.getContext();  
 **amber** = ResourcesCompat.*getColor*(**context**.getResources(),R.color.***amber\_300***, **null**);  
 View view = LayoutInflater.*from*(**context**).inflate(R.layout.***sticker\_item***, parent, **false**);  
 **return new** MyViewHolder(view);  
 }  
  
 */\*\*  
 \* Genera la vista cuando el sticker item está mostrándose en el recyclerview  
 \* El recyclerView es óptimo para largas listas de items  
 \** ***@param holder*** *El view del elemento  
 \** ***@param position*** *la posicion en el RecyclerView  
 \*/* @Override  
 **public void** onBindViewHolder(**final** MyViewHolder holder, **int** position) {  
 **final** DatosItem datosItem = **listDatosItem**.get(position);  
 Glide.*with*(**context**)  
 .load(datosItem.getFotourl())  
 .fitCenter()  
 .placeholder(R.drawable.***image\_placeholder***)  
 .error(R.drawable.***broken\_image\_blue***)  
 .fallback(R.drawable.***broken\_image\_blue***)  
 .into(holder.**itemPhoto**)  
 ;  
  
 holder.**itemName**.setText(datosItem.getNombre());  
 holder.**itemPrice**.setText(String.*valueOf*(datosItem.getPrecio()));  
  
 *// configurar fondo cuando aparece en pantalla* holder.**view**.setBackgroundColor(datosItem.isSelected() ? **amber** : Color.***WHITE***);  
  
 *// configurar cuando realiza click en itemPrice, se ejecutará lo siguiente* holder.**itemPhoto**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 datosItem.setSelected(!datosItem.isSelected());  
 holder.**view**.setBackgroundColor(datosItem.isSelected() ? **amber** : Color.***WHITE***);  
 }  
 });  
 }  
  
 @Override  
 **public int** getItemCount() {  
 **return listDatosItem** == **null** ? 0 : **listDatosItem**.size();  
 }  
  
 **class** MyViewHolder **extends** RecyclerView.ViewHolder {  
  
 **private final** ImageView **itemPhoto**;  
 **private** View **view**;  
 **private** TextView **itemPrice**;  
 **private** TextView **itemName**;  
  
 **private** MyViewHolder(View itemView) {  
 **super**(itemView);  
 **view** = itemView;  
 **itemPrice** = (TextView) itemView.findViewById(R.id.***item\_price***);  
 **itemName** = (TextView) itemView.findViewById(R.id.***item\_name***);  
 **itemPhoto** = (ImageView) itemView.findViewById(R.id.***item\_photoview***);  
 }  
 }  
}

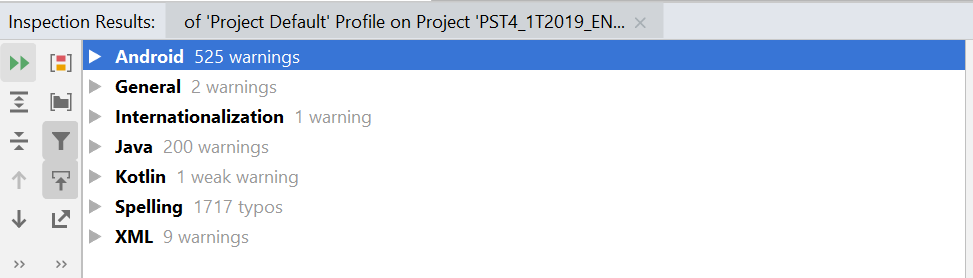
CLASE SHOPPING CART

Gestiona la selección del usuario previo a realizar el pedido. Posee un RecyclerView y un botón de confirmación o cancelación para el compromiso.

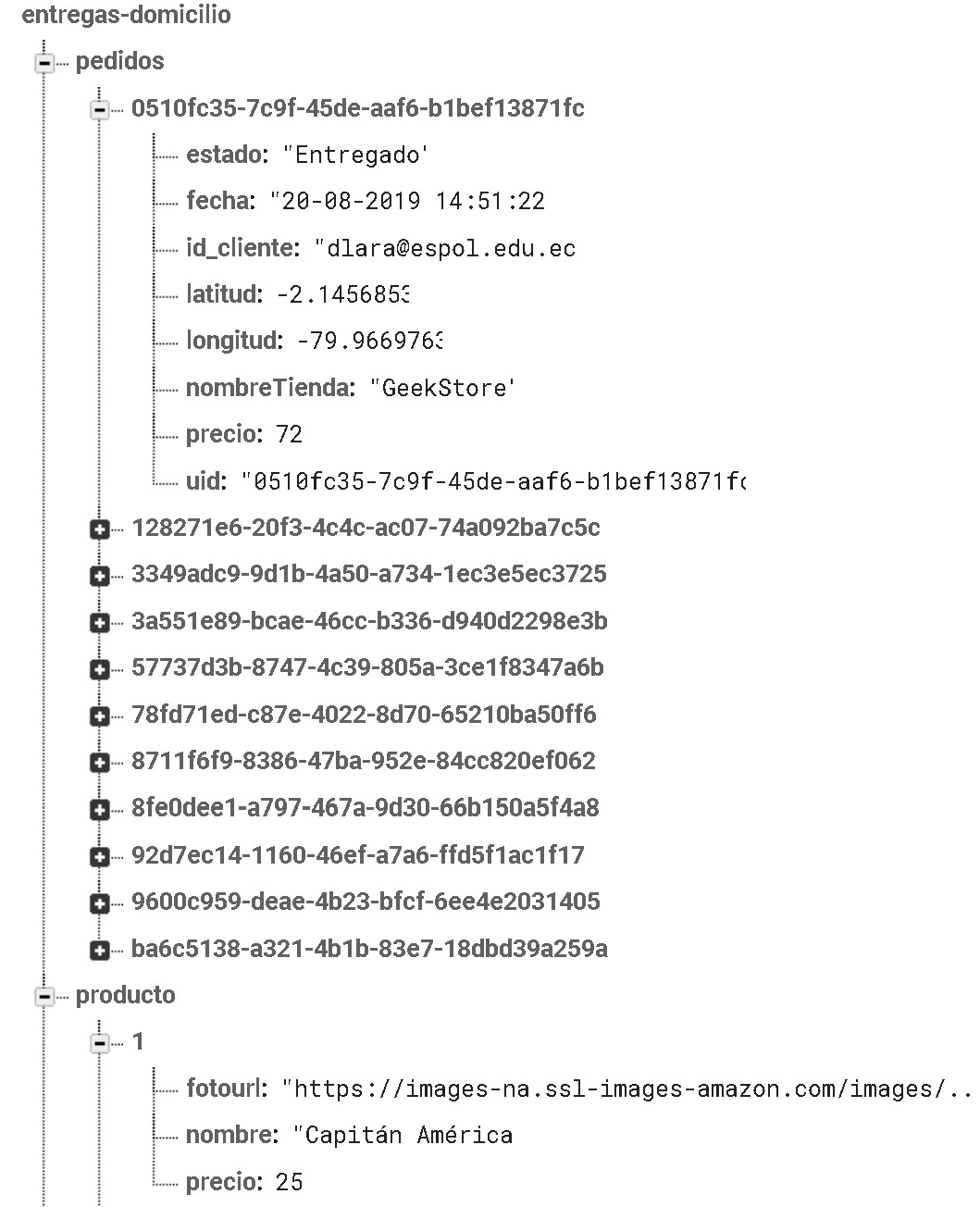
**public class** shoppingCart **extends** AppCompatActivity {  
 **private** List<DatosItem> **mDatosItemList**;  
 **private** RecyclerView **mRecyclerView**;  
 **private** RecyclerView.Adapter **mAdapter**;  
 **private** LocationManager **locManager**;  
 **private** Location **loc**;  
 **private** DatabaseReference **reference**;  
 **double lat**;  
 **double longitud**;  
 String **estado**=**"Pendiente"**;  
 String **nombreTienda**;  
 String **usuario**;  
 String **logoUrl**;  
  
  
 @RequiresApi(api = Build.VERSION\_CODES.***M***)  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_shopping\_cart***);  
 ImageView logoView = (ImageView) findViewById(R.id.***logoView***);  
 **mRecyclerView** = (RecyclerView) findViewById(R.id.***recycler\_view***);  
 **mAdapter** = **new** StickerItem(getListData());  
 LinearLayoutManager manager = **new** LinearLayoutManager(shoppingCart.**this**);  
 **mRecyclerView**.setHasFixedSize(**true**);  
 **mRecyclerView**.setLayoutManager(manager);  
 **mRecyclerView**.setAdapter(**mAdapter**);  
  
 **logoUrl** = getIntent().getStringExtra(**"URL\_LOGO1"**);  
 **nombreTienda** = getIntent().getStringExtra(**"NOMBRETIENDA"**);  
  
 Glide.*with*(**this**)  
 .load(**logoUrl**)  
 .fitCenter()  
 .placeholder(R.drawable.***image\_placeholder***)  
 .error(R.drawable.***broken\_image\_blue***)  
 .fallback(R.drawable.***broken\_image\_blue***)  
 .into(logoView)  
 ;  
 inicializarFirebase();  
  
 **usuario** = getIntent().getStringExtra(**"dato2"**);  
 **locManager** = (LocationManager) getSystemService(Context.***LOCATION\_SERVICE***);  
  
 **if** (checkSelfPermission(Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** && checkSelfPermission(Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***) {  
 *//* ***TODO: Consider calling*** *//* ***Activity#requestPermissions*** *// here to request the missing permissions, and then overriding  
 // public void onRequestPermissionsResult(int requestCode, String[] permissions,  
 // int[] grantResults)  
 // to handle the case where the user grants the permission. See the documentation  
 // for Activity#requestPermissions for more details.* **return**;  
 }  
 **loc** = **locManager**.getLastKnownLocation(LocationManager.***GPS\_PROVIDER***);  
 **lat**=**loc**.getLatitude();  
 **longitud**=**loc**.getLongitude();  
  
 }  
  
 **private** List<DatosItem> getListData() {  
 *//* ***TODO: Listar items desde la base de datos* mDatosItemList** = (ArrayList<DatosItem>)getIntent().getSerializableExtra(**"itemsSelecccionados"**);  
 **return mDatosItemList**;  
 }  
  
 **public void** noComprar(View view) {  
 Toast.*makeText*( **this**, **"Compra no realizada"**, Toast.***LENGTH\_SHORT*** ).show();  
 finish();  
 }  
  
 @RequiresApi(api = Build.VERSION\_CODES.***O***)  
 **public void** comprar(View view) {  
 Intent i = **new** Intent(**this**, compraRealizada.**class** );  
 DatabaseReference referencia = FirebaseDatabase.*getInstance*().getReference();  
 Pedido p= **new** Pedido();  
 p.setEstado(**estado**);  
 p.setNombreTienda(**nombreTienda**);  
 DateTimeFormatter formatter = DateTimeFormatter.*ofPattern*(**"dd-MM-yyyy HH:mm:ss"**);  
 p.setFecha(LocalDateTime.*now*().format(formatter));  
 p.setId\_cliente(**"dlara@espol.edu.ec"**);  
 *//p.setId\_cliente(usuario);* p.setPrecio(totalPrecio());  
 p.setLatitud(**lat**);  
 p.setLongitud(**longitud**);  
 referencia.child(**"pedidos"**).child(UUID.*randomUUID*().toString()).setValue(p);  
 Toast.*makeText*( **this**, **"Compra realizada exitosamente"**, Toast.***LENGTH\_SHORT*** ).show();  
 finish();  
 i.putExtra(**"itemsSelecccionados"**, (Serializable) **mDatosItemList**);  
 i.putExtra(**"total"**, String.*valueOf*(totalPrecio()));  
 startActivity(i);  
 }  
  
 **public double** totalPrecio(){  
 **double** total=0;  
 **for**(DatosItem d:**mDatosItemList**){  
 total+=d.getPrecio();  
 }  
 **return** total;  
 }  
  
 **private void** inicializarFirebase() {  
 FirebaseApp.*initializeApp*(**this**);  
 FirebaseDatabase database = FirebaseDatabase.*getInstance*();  
 **reference** = database.getReference();  
 *//Toast.makeText(this, "Se inicializó firebase ", Toast.LENGTH\_LONG).show();* }  
  
}

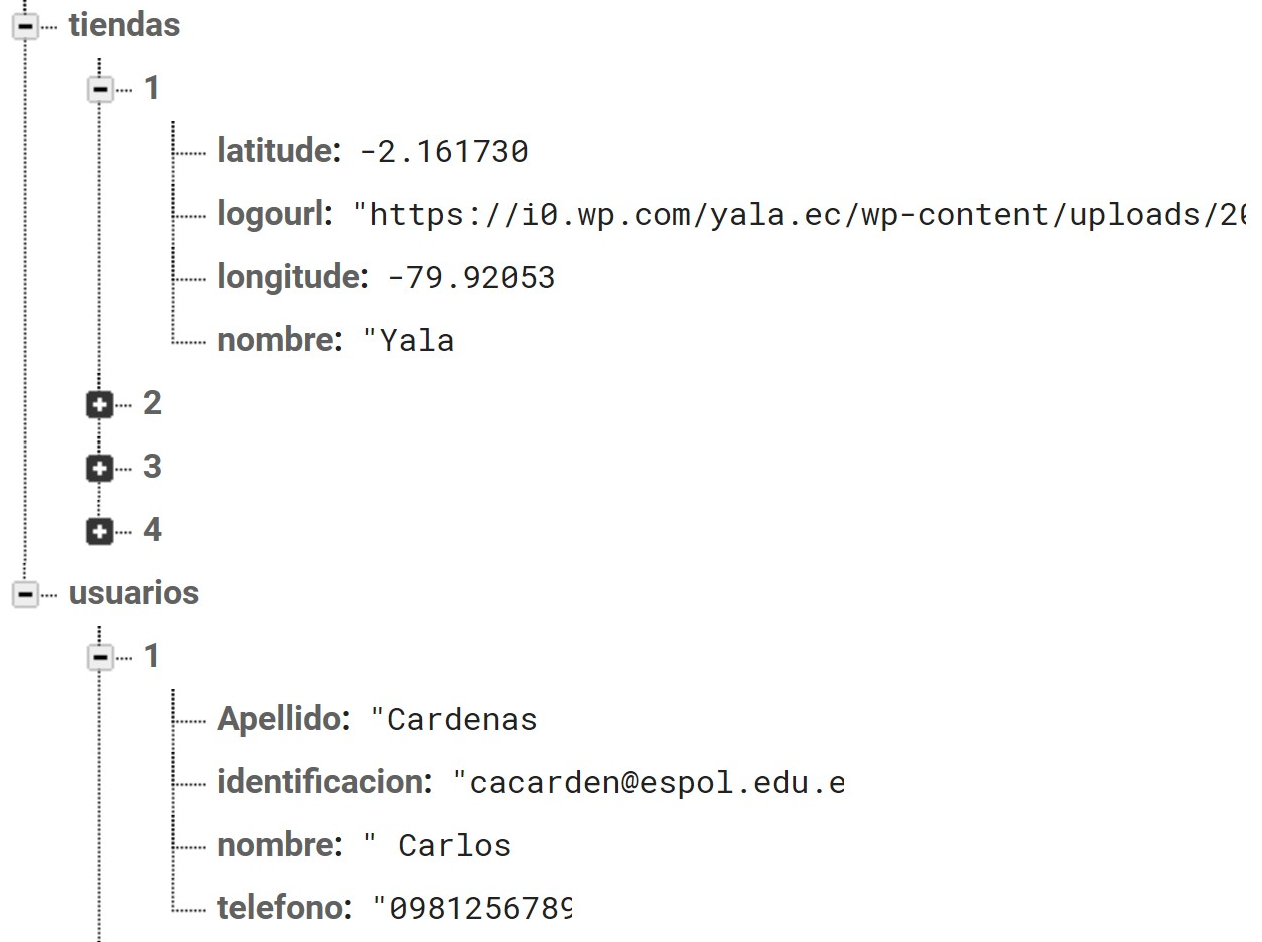


Análisis de Código



ESTRUCTURA DE LA BASE DE DATOS EN FIREBASE REALTIME DATABASE





Análisis de presupuesto

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ITEM | PRODUCTO/SERVICIO | DESCRIPCIÓN | CANTIDAD | PRECIO UNITARIO | SUBTOTAL |
| 1 | Computador Personal | Computador HP | 4 unidades | $ 600 (Pago Único) | $ 2.400 |
| 2 | Servicio Internet | Netlife | 20 megas | $ 10 (Semana) | $ 40 |
| 3 | Jefe de Proyecto | Responsable directo de la ejecución del proyecto | 40 horas | $375 (Semana) | $1.500 |
| 4 | Asistente de Proyecto | Ayudante del Jefe de Proyecto | 40 horas | $250 (Semana) | $1.000 |
| 5 | Programador 1 | Encargado del diseño y operatividad de la aplicación | 40 horas | $175 (Semana) | $700 |
| 6 | Programador 2 | Encargado del diseño y prueba de la aplicación | 40 horas | $175 (Semana) | $700 |
| 7 | Suscripciones | Play Store Android | 1 unidad | $25 | $25 |
| TOTAL | | | | | $6.365 |

Conclusiones

* El desarrollo del aplicativo móvil super shop permitió a los estudiantes desarrollar las habilidades aprendidas en las aulas de clase y proponer solución a un problema real, el cual se lo validó con YALA.EC establecimiento ubicado en la ciudad de Guayaquil.
* Los diagramas de entidad-relación, diagrama de uso y despliegue brindó una guía de inicio al momento de preparar la programación del aplicativo móvil super shop, optimizando el tiempo y funcionalidad del proyecto.
* La correcta documentación y protocolos de programación facilitó la tarea de trabajar con GITHUB repartiendo tareas específicas a cada miembro del grupo de trabajo.

# **Referencias Bibliográficas**

Ágiles, P. (13 de marzo de 2015). *Proyectos Ágiles*. Obtenido de Proyectos Ágiles: https://proyectosagiles.org/que-es-scrum/

LucidChart. (2 de julio de 2016). *LucidChart*. Obtenido de LucidChart: https://www.lucidchart.com/pages/es/que-es-un-diagrama-entidad-relacion

Ruiz, M. (9 de agosto de 2017). *Openwebinars*. Obtenido de Openwebinars: https://openwebinars.net/blog/que-es-firebase-de-google/

(UCM, 2016) (