

**METODOLOGIA DE PROGRAMACION:**

En [ingeniería de software](http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software) es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.

**SCRUM**

Scrum es un proceso en el que se aplican de manera regular [un conjunto de buenas prácticas](http://www.proyectosagiles.org/fundamentos-de-scrum) para trabajar colaborativamente, en equipo, y obtener [el mejor resultado posible](http://www.proyectosagiles.org/beneficios-de-scrum) de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un [estudio de la manera de trabajar de equipos altamente productivos](http://www.proyectosagiles.org/historia-de-scrum).

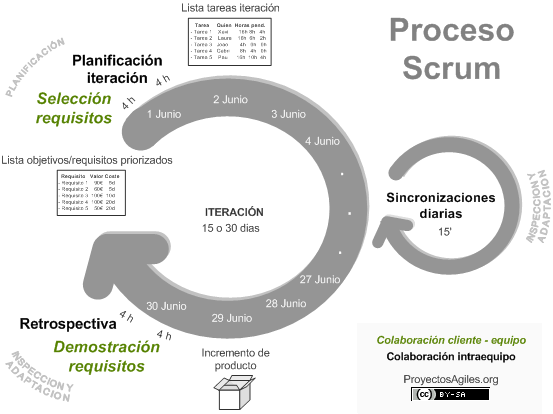
En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.

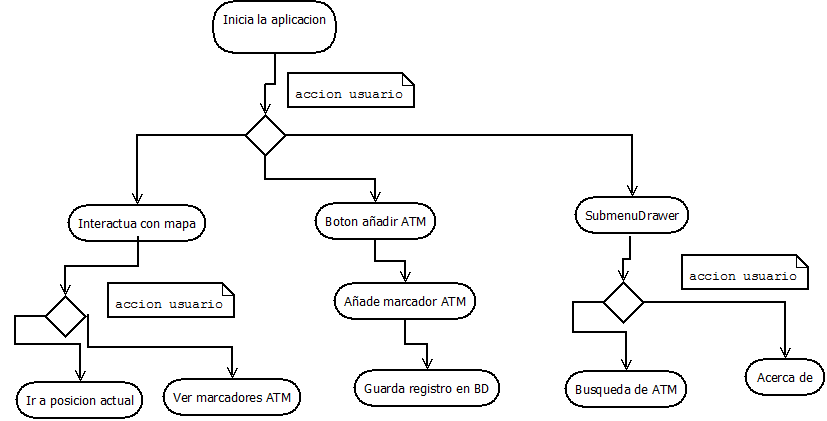
**El Proceso**

En Scrum un proyecto se ejecuta en bloques temporales cortos y fijos ([iteraciones](http://www.proyectosagiles.org/desarrollo-iterativo-incremental) de un mes natural y hasta de dos semanas, si así se necesita). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

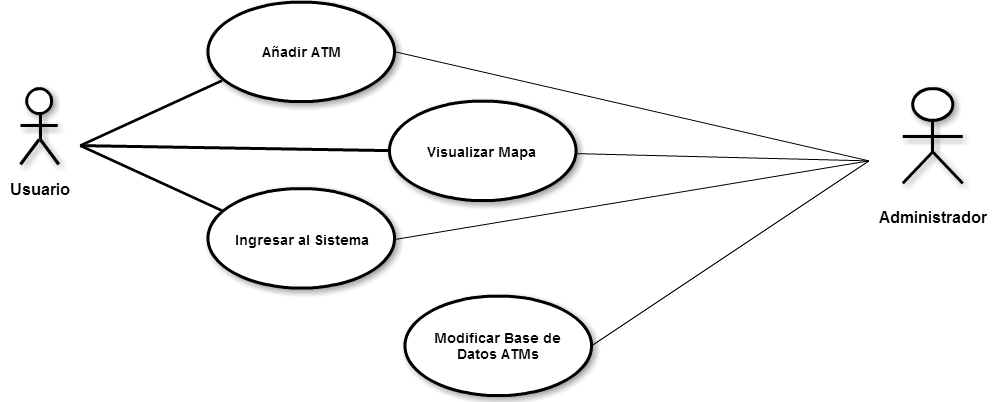
El proceso parte de la [lista de objetivos/requisitos priorizada](http://www.proyectosagiles.org/lista-requisitos-priorizada-product-backlog) del producto, que actúa como plan del proyecto. En esta lista **e**l [cliente](http://www.proyectosagiles.org/cliente-product-owner) prioriza los objetivos balanceando el valor que le aportan respecto a su coste y quedan repartidos en iteraciones y entregas. De manera regular el cliente puede [maximizar la utilidad de lo que se desarrolla](http://www.proyectosagiles.org/beneficios-de-scrum#flexibilidad-adaptacion) y el [retorno de inversión](http://www.proyectosagiles.org/beneficios-de-scrum#gestion-roi) mediante la [re planificación de objetivos](http://www.proyectosagiles.org/replanificacion-proyecto) del producto, que realiza durante la iteración con vista a las siguientes iteraciones.



**DIAGRAMA DE ACTIVIDADES:**



**DIAGRAMA DE USOS:**



**CODIGO**

**Clase Acerca De:**

**package** com.example.thom.googlemapstest;  
  
**import** android.support.v7.app.ActionBarActivity;  
**import** android.os.Bundle;  
**import** android.view.Menu;  
**import** android.view.MenuItem;  
  
  
**public class** Acerca\_De **extends** ActionBarActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_acerca\_\_de***);  
 }  
  
  
 @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\_acerca\_\_de***, menu);  
 **return true**;  
 }  
  
 @Override  
 **public boolean** onOptionsItemSelected(MenuItem item) {  
 *// Handle action bar item clicks here. The action bar will  
 // automatically handle clicks on the Home/Up button, so long  
 // as you specify a parent activity in AndroidManifest.xml.* **int** id = item.getItemId();  
  
 *//noinspection SimplifiableIfStatement* **if** (id == R.id.***action\_settings***) {  
 **return true**;  
 }  
  
 **return super**.onOptionsItemSelected(item);  
 }  
}

**Clase activity\_ingreso\_atm:**

**package** com.example.thom.googlemapstest;  
  
**import** android.support.v7.app.ActionBarActivity;  
**import** android.os.Bundle;  
**import** android.view.Menu;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.widget.AdapterView;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.EditText;  
**import** android.widget.Spinner;  
**import** android.\*;  
**import** android.widget.Toast;  
  
**import** com.google.android.gms.maps.GoogleMap;  
**import** com.google.android.gms.maps.model.BitmapDescriptorFactory;  
**import** com.google.android.gms.maps.model.LatLng;  
**import** com.google.android.gms.maps.model.Marker;  
**import** com.google.android.gms.maps.model.MarkerOptions;  
  
**import** java.util.ArrayList;  
**import** java.util.HashMap;  
**import** java.util.List;  
  
  
**public class** activity\_Ingreso\_atm **extends** ActionBarActivity {  
  
 **private** EditText **longitud**,**latitud**,**nombre**;  
 **private** Spinner **banco**;  
 **private** ArrayList<customMarker> **customMarkersArray** = **new** ArrayList<customMarker>();  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_activity\_\_ingreso\_atm***);  
  
 Spinner spinner1=(Spinner) findViewById(R.id.***Bancossp***);  
  
 ArrayAdapter<CharSequence> adaptadorDatos= ArrayAdapter.*createFromResource*(**this**,R.array.***lista\_bancos***,android.R.layout.***simple\_spinner\_item***);  
 adaptadorDatos.setDropDownViewResource(android.R.layout.***simple\_spinner\_dropdown\_item***);  
 spinner1.setAdapter(adaptadorDatos);  
 spinner1.setOnItemSelectedListener(**new** AdapterView.OnItemSelectedListener() {  
  
 **public void** onItemSelected(AdapterView<?> parent, View view, **int** position, **long** id) {  
 Toast.*makeText*(getBaseContext(), parent.getItemAtPosition(position) + **"selected"**, Toast.***LENGTH\_LONG***);  
 }  
  
 **public void** onNothingSelected(AdapterView<?> parent) {  
  
 }  
 });  
  
  
 **longitud** = (EditText) findViewById(R.id.***editText3***);  
 **latitud** = (EditText) findViewById(R.id.***editText2***);  
 **nombre**=(EditText) findViewById(R.id.***editText***);  
 **banco**=(Spinner) findViewById(R.id.***Bancossp***);  
 Locacion();  
 }  
  
 **public void** Locacion()  
 {  
 GPSDetector detector=**new** GPSDetector(**this**.getApplicationContext());  
 **longitud**.setText(Double.*toString*(detector.getLongitud()));  
 **latitud**.setText(Double.*toString*(detector.getLatitud()));  
 }  
 @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\_activity\_\_ingreso\_atm***, menu);  
 **return true**;  
 }  
  
 @Override  
 **public boolean** onOptionsItemSelected(MenuItem item) {  
 *// Handle action bar item clicks here. The action bar will  
 // automatically handle clicks on the Home/Up button, so long  
 // as you specify a parent activity in AndroidManifest.xml.* **int** id = item.getItemId();  
  
 *//noinspection SimplifiableIfStatement* **if** (id == R.id.***action\_settings***) {  
 **return true**;  
 }  
  
 **return super**.onOptionsItemSelected(item);  
 }  
 **private** HashMap<Marker, customMarker> **markersHashMap**;  
 **public void** agregar(View view)  
 {  
 String nombre,bancoa;  
 **double** longituda,latituda;  
 GPSDetector localizacion=**new** GPSDetector(**this**.getApplicationContext());  
 nombre=**this**.**nombre**.getText().toString();  
 bancoa=String.*valueOf*(**this**.**banco**.getSelectedItem());  
 longituda=localizacion.getLongitud();  
 latituda=localizacion.getLatitud();  
 ConexionBD ObjCnx = **new** ConexionBD(**this**);  
  
 ObjCnx.abrirConexion();  
  
 **if**(ObjCnx.insertar(nombre,bancoa,latituda,longituda)==**true**)  
 {  
 String texto = **"Elemento Agregado Correctamente"**;  
 Toast toast = Toast.*makeText*(**this**, texto,Toast.***LENGTH\_LONG***);  
 toast.show();  
 }  
 **else** {  
 String texto = **"Error al Agregar Elemento"**;  
 Toast toast = Toast.*makeText*(**this**,texto,Toast.***LENGTH\_LONG***);  
 toast.show();  
 }  
 ObjCnx.cerrarConexiones();  
 GPSDetector a=**new** GPSDetector(**this**.getBaseContext());  
 **markersHashMap** = **new** HashMap<Marker, customMarker>();  
 **customMarkersArray**.add(**new** customMarker(a.getLatitud(), a.getLongitud()));  
 addMarker();  
 plotMarkers(**customMarkersArray**);  
 }  
 GoogleMap **googleMap**;  
 Marker **marcador**;  
 **private void** addMarker(){  
  
 **if**(**null** != **googleMap**)  
 {  
 GPSDetector lol=**null**;  
 **marcador** = MainActivity.*getMapa*().addMarker(**new** MarkerOptions()  
 .position(**new** LatLng(lol.getLatitud(), lol.getLongitud()))  
 );  
  
 **googleMap**.setOnMarkerClickListener(**new** GoogleMap.OnMarkerClickListener() {  
 @Override  
 **public boolean** onMarkerClick(Marker marker) {  
 **return false**;  
 }  
 });  
 }  
 }  
 **private void** addMarker( **double** latitud,**double** longitud) {  
 */\*\* Make sure that the map has been initialised \*\*/* **if**(**null** != **googleMap**)  
 {  
 GPSDetector lol=**null**;  
 **marcador** = MainActivity.*getMapa*().addMarker(**new** MarkerOptions()  
 .position(**new** LatLng(latitud, longitud))  
 );  
  
 **googleMap**.setOnMarkerClickListener(**new** GoogleMap.OnMarkerClickListener() {  
 @Override  
 **public boolean** onMarkerClick(Marker marker) {  
 **return false**;  
 }  
 });  
 }  
 }  
 **private void** plotMarkers(ArrayList<customMarker> markers) {  
 **if** (markers.size() > 0) {  
  
 **for** (customMarker myCustomMarker : markers) {  
 MarkerOptions markerOption = **new** MarkerOptions()  
 .position(**new** LatLng(myCustomMarker.getLat(), myCustomMarker.getLng()));  
 markerOption.icon(BitmapDescriptorFactory.*fromResource*(R.drawable.atm));  
  
 Marker currentMarker = MainActivity.getMapa().addMarker(markerOption);  
 markersHashMap.put(currentMarker, myCustomMarker);  
 }  
 }  
 }  
}

**Clase Añadir\_Marcadores**

**package** com.example.thom.googlemapstest;  
  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.support.v7.app.ActionBarActivity;  
**import** android.os.Bundle;  
**import** android.view.Menu;  
**import** android.view.MenuItem;  
**import** android.widget.TextView;  
  
**import** com.google.android.gms.maps.GoogleMap;  
**import** com.google.android.gms.maps.model.BitmapDescriptorFactory;  
**import** com.google.android.gms.maps.model.LatLng;  
**import** com.google.android.gms.maps.model.Marker;  
**import** com.google.android.gms.maps.model.MarkerOptions;  
  
**import** java.util.ArrayList;  
**import** java.util.HashMap;  
  
  
**public class** Anadir\_Marcadores **extends** ActionBarActivity {  
  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_marc***);  
 */\* \*/* Crear\_Lista();  
 }  
 @Override  
 **public boolean** onCreateOptionsMenu(Menu menu) {  
  
 getMenuInflater().inflate(R.menu.***menu\_anadir\_\_marcadores***, menu);  
 **return true**;  
}  
 @Override  
 **public boolean** onOptionsItemSelected(MenuItem item) {  
 *// Handle action bar item clicks here. The action bar will  
 // automatically handle clicks on the Home/Up button, so long  
 // as you specify a parent activity in AndroidManifest.xml.* **int** id = item.getItemId();  
  
 *//noinspection SimplifiableIfStatement* **if** (id == R.id.***action\_settings***) {  
 **return true**;  
 }  
  
 **return super**.onOptionsItemSelected(item);  
 }  
 **public void** Crear\_Lista()  
 {  
 *// Cursor c = ConexionBD.db.rawQuery("SELECT \* FROM Cajeros", null);* ((TextView)**this**.findViewById(R.id.***txt\_res***)).setText(**"Holalals"**);  
 }  
 }

**Clase BD:**

**package** com.example.thom.googlemapstest;  
**import** android.content.Context;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.database.sqlite.SQLiteOpenHelper;  
  
**public class** BD **extends** SQLiteOpenHelper {  
  
 String **query**=**"CREATE TABLE Cajeros(Nombre TEXT,Banco TEXT, latitud DOUBLE, longitud DOUBLE)"**;  
 **public** BD(Context context, String name, SQLiteDatabase.CursorFactory factory, **int** version)  
 {  
 **super**(context,name,factory,version);  
 }  
  
 **public void** onCreate(SQLiteDatabase db)  
 {  
 db.execSQL(**query**); *//Ejecuta query para crear la tabla* }  
  
 **public void** onUpgrade(SQLiteDatabase db, **int** i, **int** i2)  
 {  
 db.execSQL(**"DROP TABLE IF EXISTS Cajeros"**); *//\*hace actualizaciones de la tabla* db.execSQL(**query**);  
 }  
  
}

**Clase Conexión BD**

**package** com.example.thom.googlemapstest;  
**import** android.content.Context;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.database.sqlite.SQLiteDatabaseLockedException;  
**import** android.view.View;  
**import** android.widget.Toast;  
  
**public class** ConexionBD {  
  
 **public static** SQLiteDatabase *db*;  
 **private static** Context *nContext*;  
 **private static** BD *objBD*;  
  
 **public** ConexionBD(Context o)  
 {  
 *nContext*=o;  
 }  
  
 **public** SQLiteDatabase abrirConexion()  
 {  
 *objBD* = **new** BD(*nContext*,**"BDatmRepo"**,**null**,1);  
 *db* = *objBD*.getWritableDatabase();  
 **return** *db*;  
 }  
  
 **public void** cerrarConexiones()  
 {  
 *db*.close();  
 }  
  
 **public boolean** insertar(String nombre,String banco, **double** latitud, **double** longitud)  
 {  
 **boolean** resultado = **false**;  
 **try**{  
 String query=**"INSERT INTO Cajeros (Nombre,banco,latitud,longitud) VALUES('"**+nombre+**"','"**+banco+**"',"**+latitud+**","**+longitud+**")"**;  
  
 *db*.execSQL(query);  
 resultado = **true**;  
 **return** resultado;  
  
 }**catch**(Exception e)  
 {  
 resultado = **false**;  
 **return** resultado;  
 }  
 }  
}

**Clase Custom Marker**

**package** com.example.thom.googlemapstest;  
  
*/\*\*  
 \* Created by Thom on 4/19/2015.  
 \*/***public class** customMarker {  
  
  
 **private** Double **lat**;  
 **private** Double **lng**;  
  
 **public** customMarker( Double lat, Double lng) {  
  
 **this**.**lat** = lat;  
 **this**.**lng** = lng;  
 }  
 **public** Double getLat() {  
 **return lat**;  
 }  
 **public** Double getLng() {  
 **return lng**;  
 }  
  
}

**Clase GPS detector**

**package** com.example.thom.googlemapstest;  
  
**import** android.app.AlertDialog;  
**import** android.app.Service;  
**import** android.content.Context;  
**import** android.content.DialogInterface;  
**import** android.content.Intent;  
**import** android.location.Location;  
**import** android.location.LocationListener;  
**import** android.location.LocationManager;  
**import** android.os.Bundle;  
**import** android.os.IBinder;  
**import** android.provider.Settings;  
**import** android.util.Log;  
  
*/\*\*  
 \* Created by Personal on 23/04/2015.  
 \*/***public class** GPSDetector **extends** Service **implements** LocationListener {  
  
 **private final** Context **mContext**;  
  
 *//Estado GPS* **boolean isGPSEnabled** = **false**;  
  
 *//Estado de Red* **boolean isNetworkEnabled** = **false**;  
  
 *// GPS puede obtener locacion* **boolean canGetLocation** = **false**;  
  
 Location **locacion** = **null**; *// locacion* **double latitud**; *// latitud* **double longitud**; *// longitud  
  
 // Distancia minima para actualizar posicion* **private static final long *MIN\_DISTANCE\_CHANGE\_FOR\_UPDATES*** = 0; *// 10 metros  
  
 // Tiempo minimo para actualizar posicion en milisegundos* **private static final long *MIN\_TIME\_BW\_UPDATES*** = 60 \* 1; *// 1 minut0  
  
 // Declaracion de un Manejador de locacion* **protected** LocationManager **locationManager**;  
  
 **public** GPSDetector(Context context) {  
 **this**.**mContext** = context;  
 getLocacion();  
 }  
  
 **public** Location getLocacion() {  
 **try** {  
 **locationManager** = (LocationManager) **mContext**.getSystemService(Context.***LOCATION\_SERVICE***);  
  
 *// Obtencion del estado del GPS* **isGPSEnabled** = **locationManager**.isProviderEnabled(LocationManager.***GPS\_PROVIDER***);  
  
 *// Obtencion del estado de la red* **isNetworkEnabled** = **locationManager**.isProviderEnabled(LocationManager.***NETWORK\_PROVIDER***);  
  
 **if** (!**isGPSEnabled** && !**isNetworkEnabled**) {  
 *// Red no activada* showSettingsAlert();  
 *// GPS activado* **if** (**isGPSEnabled**) {  
 **if** (**locacion** == **null**) {  
 **locationManager**.requestLocationUpdates(LocationManager.***GPS\_PROVIDER***,***MIN\_TIME\_BW\_UPDATES***,***MIN\_DISTANCE\_CHANGE\_FOR\_UPDATES***, **this**);  
 Log.*d*(**"GPS"**, **"GPS Enabled"**);  
 **if** (**locationManager** != **null**) {  
 **locacion** = **locationManager**.getLastKnownLocation(LocationManager.***GPS\_PROVIDER***);  
 **if** (**locacion** != **null**) {  
 **latitud** = **locacion**.getLatitude();  
 **longitud** = **locacion**.getLongitude();  
 }  
 }  
 }  
 }  
  
 }  
 **else** {  
 **this**.**canGetLocation** = **true**;  
 *// GPS activado* **if** (**isGPSEnabled**) {  
 **if** (**locacion** == **null**) {  
 **locationManager**.requestLocationUpdates(LocationManager.***GPS\_PROVIDER***,***MIN\_TIME\_BW\_UPDATES***,***MIN\_DISTANCE\_CHANGE\_FOR\_UPDATES***, **this**);  
 Log.*d*(**"GPS"**, **"GPS Activado"**);  
 **if** (**locationManager** != **null**) {  
 **locacion** = **locationManager**.getLastKnownLocation(LocationManager.***GPS\_PROVIDER***);  
 **if** (**locacion** != **null**) {  
 **latitud** = **locacion**.getLatitude();  
 **longitud** = **locacion**.getLongitude();  
 }  
 }  
 }  
 }  
  
 **if** (**isNetworkEnabled**) {  
 **locationManager**.requestLocationUpdates(LocationManager.***NETWORK\_PROVIDER***,***MIN\_TIME\_BW\_UPDATES***,***MIN\_DISTANCE\_CHANGE\_FOR\_UPDATES***, **this**);  
 Log.*d*(**"Network"**, **"Red Activada"**);  
 **if** (**locationManager** != **null**) {  
 **locacion** = **locationManager**.getLastKnownLocation(LocationManager.***NETWORK\_PROVIDER***);  
 **if** (**locacion** != **null**) {  
 **latitud** = **locacion**.getLatitude();  
 **longitud** = **locacion**.getLongitude();  
 }  
 }  
 }  
  
 }  
  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
  
 **return locacion**;  
 }  
 */\*\*  
 \* Stop using GPS listener Calling this function will stop using GPS in your  
 \* app  
 \* \*/* **public void** detenerGPS() {  
 **if** (**locationManager** != **null**) {  
 **locationManager**.removeUpdates(GPSDetector.**this**);  
 }  
 }  
 */\*\*  
 \* Function to get latitude  
 \* \*/* **public double** getLatitud() {  
 **if** (**locacion** != **null**) {  
 **latitud** = **locacion**.getLatitude();  
 }  
 **return latitud**;  
 }  
 */\*\*  
 \* Function to get longitude  
 \* \*/* **public double** getLongitud() {  
 **if** (**locacion** != **null**) {  
 **longitud** = **locacion**.getLongitude();  
 }  
  
 **return longitud**;  
 }  
  
 */\*\*  
 \* Function to check GPS/wifi enabled  
 \*  
 \** ***@return*** *boolean  
 \* \*/* **public boolean** canGetLocation() {  
 **return this**.**canGetLocation**;  
 }  
  
 */\*\*  
 \* Function to show settings alert dialog On pressing Settings button will  
 \* lauch Settings Options  
 \* \*/* **public void** showSettingsAlert() {  
 AlertDialog.Builder alertDialog = **new** AlertDialog.Builder(**mContext**);  
  
 *// Setting Dialog Title* alertDialog.setTitle(**"Ojo"**);  
  
 *// Setting Dialog Message* alertDialog  
 .setMessage(**"GPS Desactivado. ¿Ir al menu de ajustes?"**);  
  
 *// On pressing Settings button* alertDialog.setPositiveButton(**"Ajustes"**,  
 **new** DialogInterface.OnClickListener() {  
 **public void** onClick(DialogInterface dialog, **int** which) {  
 Intent intent = **new** Intent(  
 Settings.***ACTION\_LOCATION\_SOURCE\_SETTINGS***);  
 **mContext**.startActivity(intent);  
 }  
 });  
  
 *// on pressing cancel button* alertDialog.setNegativeButton(**"Cancelar"**,  
 **new** DialogInterface.OnClickListener() {  
 **public void** onClick(DialogInterface dialog, **int** which) {  
 dialog.cancel();  
 }  
 });  
  
 *// Showing Alert Message* alertDialog.show();  
 }  
  
 @Override  
 **public void** onLocationChanged(Location location) {  
 }  
  
 @Override  
 **public void** onProviderDisabled(String provider) {  
 }  
  
 @Override  
 **public void** onProviderEnabled(String provider) {  
 }  
  
 @Override  
 **public void** onStatusChanged(String provider, **int** status, Bundle extras) {  
 }  
  
 @Override  
 **public** IBinder onBind(Intent arg0) {  
 **return null**;  
 }  
}

**Clase Main Activity**

**package** com.example.thom.googlemapstest;  
  
**import** android.content.Intent;  
**import** android.graphics.Color;  
**import** android.support.v7.app.ActionBarActivity;  
**import** android.support.v7.app.ActionBar;  
**import** android.support.v4.app.FragmentManager;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.view.Gravity;  
**import** android.view.Menu;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.support.v4.widget.DrawerLayout;  
**import** android.widget.SearchView;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
  
**import** com.google.android.gms.maps.MapFragment;  
**import** com.google.android.gms.maps.model.BitmapDescriptorFactory;  
**import** com.google.android.gms.maps.model.LatLng;  
**import** com.google.android.gms.maps.model.Marker;  
**import** com.google.android.gms.maps.model.MarkerOptions;  
  
**import** android.widget.ImageButton;  
**import** android.view.View.OnClickListener;  
  
**import** java.util.ArrayList;  
**import** java.util.HashMap;  
  
**import** com.google.android.gms.maps.GoogleMap;  
  
  
**public class** MainActivity **extends** ActionBarActivity  
 **implements** Menu\_Navegacion.NavigationDrawerCallbacks {  
  
 */\*\*  
 \* Fragment managing the behaviors, interactions and presentation of the navigation drawer.  
 \*/* **private** Menu\_Navegacion **mMenuNavegacion**;  
 DrawerLayout **mDrawerLayout**;  
  
  
  
 */\*\*  
 \* Used to store the last screen title. For use in {****@link*** *#restoreActionBar()}.  
 \*/* **private** CharSequence **mTitle**;  
 **private static** GoogleMap *googleMap*;  
  
  
 *//Added for custom buttons* ImageButton **meetUpButton**;  
 ImageButton **menuButton**;  
 ImageButton **addButton**;  
  
 ImageButton **searchBarButton**;  
  
 SearchView **searchView**;  
 **boolean hide** = **true**;  
  
 **public void** Ingresar\_atm(View view) {  
 Intent nombre = **new** Intent(**this**,activity\_Ingreso\_atm.**class**);  
 startActivity(**new** Intent(getApplicationContext(),activity\_Ingreso\_atm.**class**));  
 }  
 **public static** GoogleMap getMapa()  
 {  
  
 **return** *googleMap*;  
 }  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 getSupportActionBar().hide();  
  
 setContentView(R.layout.***activity\_main***);  
  
 **mMenuNavegacion** = (Menu\_Navegacion)  
 getSupportFragmentManager().findFragmentById(R.id.***navigation\_drawer***);  
 **mTitle** = getTitle();  
  
 **mMenuNavegacion**.setUp(  
 R.id.***navigation\_drawer***,  
 (DrawerLayout) findViewById(R.id.***drawer\_layout***));  
  
 **mDrawerLayout** = (DrawerLayout) findViewById(R.id.***drawer\_layout***);  
  
  
  
  
  
  
 createMapView();  
 *//addMarker();  
 //plotMarkers(customMarkersArray);  
  
 googleMap*.setMapType(GoogleMap.***MAP\_TYPE\_NORMAL***);  
 *googleMap*.setMyLocationEnabled(**true**);  
  
 addListenerOnMenuButton();  
  
 }  
 */\*Opciones del Mapa de navegacion \*/* @Override  
 **public void** onNavigationDrawerItemSelected(**int** position) {  
  
 **switch** (position) {  
 **case** 0:  
  
 **break**;  
 **case** 1:  
 Intent Consulta = **new** Intent(**this**,Anadir\_Marcadores.**class**);  
 startActivity(**new** Intent(getApplicationContext(),Anadir\_Marcadores.**class**));  
 **break**;  
 **case** 2:  
 Intent nombre = **new** Intent(**this**,Acerca\_De.**class**);  
 startActivity(**new** Intent(getApplicationContext(),Acerca\_De.**class**));  
 **break**;  
  
 }  
  
  
 FragmentManager fragmentManager = getSupportFragmentManager();  
 fragmentManager.beginTransaction()  
 .commit();  
  
 }  
  
 **private void** createMapView(){  
 */\*\*  
 \* Catch the null pointer exception that  
 \* may be thrown when initialising the map  
 \*/* **try** {  
 **if**(**null** == *googleMap*){  
 *googleMap* = ((MapFragment) getFragmentManager().findFragmentById(  
 R.id.***mapView***)).getMap();  
  
 */\*\*  
 \* If the map is still null after attempted initialisation,  
 \* show an error to the user  
 \*/* **if**(**null** == *googleMap*) {  
 Toast.*makeText*(getApplicationContext(),  
 **"Error creando mapa"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 **else** {  
 *googleMap*.setOnMarkerClickListener(**new** GoogleMap.OnMarkerClickListener() {  
 @Override  
 **public boolean** onMarkerClick(com.google.android.gms.maps.model.Marker marker) {  
 marker.showInfoWindow();  
 **return true**;  
 }  
 });  
 }  
 }  
 } **catch** (NullPointerException exception){  
 Log.*e*(**"mapApp"**, exception.toString());  
 }  
 }  
 */\*\*  
 \* Adds a marker to the map  
 \*/***public** GoogleMap get\_Mapa()  
{  
 **return** *googleMap*;  
}  
  
 **private int** manageMarkerPhoto(String photoName) {  
 **return** R.drawable.***cafe***;  
 }  
  
 **private int** manageMarkerRating(String ratingName) {  
 **return** R.drawable.***fourandahalf2***;  
 }  
  
 **public void** onSectionAttached(**int** number) {  
 **switch** (number) {  
 **case** 1:  
 **mTitle** = getString(R.string.***title\_section1***);  
 **break**;  
 **case** 2:  
 **mTitle** = getString(R.string.***title\_section2***);  
 **break**;  
 **case** 3:  
 **mTitle** = getString(R.string.***title\_section3***);  
 **break**;  
  
 }  
 }  
  
 **public void** restoreActionBar() {  
 ActionBar actionBar = getSupportActionBar();  
 actionBar.setNavigationMode(ActionBar.***NAVIGATION\_MODE\_STANDARD***);  
 actionBar.setDisplayShowTitleEnabled(**true**);  
 actionBar.setTitle(**mTitle**);  
 }  
  
  
 @Override  
 **public boolean** onCreateOptionsMenu(Menu menu) {  
  
 **if** (!**mMenuNavegacion**.isDrawerOpen()) {  
 *// Only show items in the action bar relevant to this screen  
 // if the drawer is not showing. Otherwise, let the drawer  
 // decide what to show in the action bar.* getMenuInflater().inflate(R.menu.***main***, menu);  
 restoreActionBar();  
 **return true**;  
 }  
  
 **return super**.onCreateOptionsMenu(menu);  
 }  
  
 @Override  
 **public boolean** onOptionsItemSelected(MenuItem item) {  
 *// Handle action bar item clicks here. The action bar will  
 // automatically handle clicks on the Home/Up button, so long  
 // as you specify a parent activity in AndroidManifest.xml.* **int** id = item.getItemId();  
  
 *//noinspection SimplifiableIfStatement* **if** (id == R.id.***action\_settings***) {  
 **return true**;  
 }  
  
 **return super**.onOptionsItemSelected(item);  
 }  
  
  
  
 **public void** addListenerOnMenuButton() {  
  
 **menuButton** = (ImageButton) findViewById(R.id.***menuButton***);  
  
 **menuButton**.setOnClickListener(**new** OnClickListener() {  
  
 @Override  
 **public void** onClick(View arg0) {  
  
 **mDrawerLayout**.openDrawer(Gravity.***LEFT***);  
  
 }  
  
 });  
  
 }  
  
  
  
 **public void** doLocationSearch(String query) {  
  
 }  
  
 **public void** hide(View v) {  
 **if** (**hide**) {  
 **searchView**.setVisibility(View.***VISIBLE***);  
 **searchView**.setQuery(**""**, **false**);  
 **searchView**.clearFocus();  
 createSearchBarBackground();  
 **hide** = **false**;  
 }  
 **else** {  
 **searchView**.setVisibility(View.***GONE***);  
 **hide** = **true**;  
 }  
 }  
  
 **public void** createSearchBarBackground() {  
 **searchView**.setQueryHint(**"Search for an Address"**);  
 **int** searchPlateId = **searchView**.getContext().getResources().getIdentifier(**"android:id/search\_plate"**, **null**, **null**);  
 View searchPlate = **searchView**.findViewById(searchPlateId);  
  
 **if** (searchPlate!=**null**) {  
 searchPlate.setBackgroundColor(Color.***WHITE***);  
 **int** searchTextId = searchPlate.getContext().getResources().getIdentifier(**"android:id/search\_src\_text"**, **null**, **null**);  
 TextView searchText = (TextView) searchPlate.findViewById(searchTextId);  
  
 **if** (searchText != **null**) {  
 searchText.setTextColor(Color.***DKGRAY***);  
 searchText.setHintTextColor(Color.***DKGRAY***);  
 }  
 }  
 }  
  
 }  
  
**Clase Menu\_Navegación:**

**import** android.support.v7.app.ActionBarActivity;  
**import** android.app.Activity;  
**import** android.support.v7.app.ActionBar;  
**import** android.support.v4.app.Fragment;  
**import** android.support.v4.app.ActionBarDrawerToggle;  
**import** android.support.v4.view.GravityCompat;  
**import** android.support.v4.widget.DrawerLayout;  
**import** android.content.SharedPreferences;  
**import** android.content.res.Configuration;  
**import** android.os.Bundle;  
**import** android.preference.PreferenceManager;  
**import** android.view.LayoutInflater;  
**import** android.view.Menu;  
**import** android.view.MenuInflater;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.AdapterView;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.ListView;  
**import** android.widget.Toast;  
  
*/\*\*  
 \* Fragment used for managing interactions for and presentation of a navigation drawer.  
 \* See the <a href="https://developer.android.com/design/patterns/navigation-drawer.html#Interaction">  
 \* design guidelines</a> for a complete explanation of the behaviors implemented here.  
 \*/***public class** Menu\_Navegacion **extends** Fragment {  
  
 */\*\*  
 \* Remember the position of the selected item.  
 \*/* **private static final** String ***STATE\_SELECTED\_POSITION*** = **"selected\_navigation\_drawer\_position"**;  
  
 */\*\*  
 \* Per the design guidelines, you should show the drawer on launch until the user manually  
 \* expands it. This shared preference tracks this.  
 \*/* **private static final** String ***PREF\_USER\_LEARNED\_DRAWER*** = **"navigation\_drawer\_learned"**;  
  
 **private** NavigationDrawerCallbacks **mCallbacks**;  
  
 */\*\*  
 \* Helper component that ties the action bar to the navigation drawer.  
 \*/* **private** ActionBarDrawerToggle **mDrawerToggle**;  
  
 **private** DrawerLayout **mDrawerLayout**;  
 **private** ListView **mDrawerListView**;  
 **private** View **mFragmentContainerView**;  
  
 **private int mCurrentSelectedPosition** = 0;  
 **private boolean mFromSavedInstanceState**;  
 **private boolean mUserLearnedDrawer**;  
  
 **public** Menu\_Navegacion() {  
 }  
  
 @Override  
 **public void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
  
 *// Read in the flag indicating whether or not the user has demonstrated awareness of the  
 // drawer. See PREF\_USER\_LEARNED\_DRAWER for details.* SharedPreferences sp = PreferenceManager.*getDefaultSharedPreferences*(getActivity());  
 **mUserLearnedDrawer** = sp.getBoolean(***PREF\_USER\_LEARNED\_DRAWER***, **false**);  
  
 **if** (savedInstanceState != **null**) {  
 **mCurrentSelectedPosition** = savedInstanceState.getInt(***STATE\_SELECTED\_POSITION***);  
 **mFromSavedInstanceState** = **true**;  
 }  
  
 *// Select either the default item (0) or the last selected item.* selectItem(**mCurrentSelectedPosition**);  
 }  
  
 @Override  
 **public void** onActivityCreated(Bundle savedInstanceState) {  
 **super**.onActivityCreated(savedInstanceState);  
 *// Indicate that this fragment would like to influence the set of actions in the action bar.* setHasOptionsMenu(**true**);  
 }  
  
 @Override  
 **public** View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 **mDrawerListView** = (ListView) inflater.inflate(  
 R.layout.***fragment\_navigation\_drawer***, container, **false**);  
 **mDrawerListView**.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {  
 @Override  
 **public void** onItemClick(AdapterView<?> parent, View view, **int** position, **long** id) {  
 selectItem(position);  
 }  
 });  
 **mDrawerListView**.setAdapter(**new** ArrayAdapter<String>(  
 getActionBar().getThemedContext(),  
 android.R.layout.***simple\_list\_item\_activated\_1***,  
 android.R.id.***text1***,  
 **new** String[]{  
 getString(R.string.***title\_section1***),  
 getString(R.string.***title\_section2***),  
 getString(R.string.***title\_section3***),  
  
 }));  
 **mDrawerListView**.setItemChecked(**mCurrentSelectedPosition**, **true**);  
 **return mDrawerListView**;  
 }  
  
 **public boolean** isDrawerOpen() {  
 **return mDrawerLayout** != **null** && **mDrawerLayout**.isDrawerOpen(**mFragmentContainerView**);  
 }  
  
 */\*\*  
 \* Users of this fragment must call this method to set up the navigation drawer interactions.  
 \*  
 \** ***@param fragmentId*** *The android:id of this fragment in its activity's layout.  
 \** ***@param drawerLayout*** *The DrawerLayout containing this fragment's UI.  
 \*/* **public void** setUp(**int** fragmentId, DrawerLayout drawerLayout) {  
 **mFragmentContainerView** = getActivity().findViewById(fragmentId);  
 **mDrawerLayout** = drawerLayout;  
  
 *// set a custom shadow that overlays the main content when the drawer opens* **mDrawerLayout**.setDrawerShadow(R.drawable.***drawer\_shadow***, GravityCompat.***START***);  
 *// set up the drawer's list view with items and click listener* ActionBar actionBar = getActionBar();  
 actionBar.setDisplayHomeAsUpEnabled(**true**);  
 actionBar.setHomeButtonEnabled(**true**);  
  
 *// ActionBarDrawerToggle ties together the the proper interactions  
 // between the navigation drawer and the action bar app icon.* **mDrawerToggle** = **new** ActionBarDrawerToggle(  
 getActivity(), */\* host Activity \*/* **mDrawerLayout**, */\* DrawerLayout object \*/* R.drawable.***ic\_drawer***, */\* nav drawer image to replace 'Up' caret \*/* R.string.***navigation\_drawer\_open***, */\* "open drawer" description for accessibility \*/* R.string.***navigation\_drawer\_close*** */\* "close drawer" description for accessibility \*/* ) {  
 @Override  
 **public void** onDrawerClosed(View drawerView) {  
 **super**.onDrawerClosed(drawerView);  
 **if** (!isAdded()) {  
 **return**;  
 }  
  
 getActivity().supportInvalidateOptionsMenu(); *// calls onPrepareOptionsMenu()* }  
  
 @Override  
 **public void** onDrawerOpened(View drawerView) {  
 **super**.onDrawerOpened(drawerView);  
 **if** (!isAdded()) {  
 **return**;  
 }  
  
 **if** (!**mUserLearnedDrawer**) {  
 *// The user manually opened the drawer; store this flag to prevent auto-showing  
 // the navigation drawer automatically in the future.* **mUserLearnedDrawer** = **true**;  
 SharedPreferences sp = PreferenceManager  
 .*getDefaultSharedPreferences*(getActivity());  
 sp.edit().putBoolean(***PREF\_USER\_LEARNED\_DRAWER***, **true**).apply();  
 }  
  
 getActivity().supportInvalidateOptionsMenu(); *// calls onPrepareOptionsMenu()* }  
 };  
  
 *// If the user hasn't 'learned' about the drawer, open it to introduce them to the drawer,  
 // per the navigation drawer design guidelines.* **if** (!**mUserLearnedDrawer** && !**mFromSavedInstanceState**) {  
 **mDrawerLayout**.openDrawer(**mFragmentContainerView**);  
 }  
  
 *// Defer code dependent on restoration of previous instance state.* **mDrawerLayout**.post(**new** Runnable() {  
 @Override  
 **public void** run() {  
 **mDrawerToggle**.syncState();  
 }  
 });  
  
 **mDrawerLayout**.setDrawerListener(**mDrawerToggle**);  
 }  
  
 **private void** selectItem(**int** position) {  
 **mCurrentSelectedPosition** = position;  
 **if** (**mDrawerListView** != **null**) {  
 **mDrawerListView**.setItemChecked(position, **true**);  
 }  
 **if** (**mDrawerLayout** != **null**) {  
 **mDrawerLayout**.closeDrawer(**mFragmentContainerView**);  
 }  
 **if** (**mCallbacks** != **null**) {  
 **mCallbacks**.onNavigationDrawerItemSelected(position);  
 }  
 }  
  
 @Override  
 **public void** onAttach(Activity activity) {  
 **super**.onAttach(activity);  
 **try** {  
 **mCallbacks** = (NavigationDrawerCallbacks) activity;  
 } **catch** (ClassCastException e) {  
 **throw new** ClassCastException(**"Activity must implement NavigationDrawerCallbacks."**);  
 }  
 }  
  
 @Override  
 **public void** onDetach() {  
 **super**.onDetach();  
 **mCallbacks** = **null**;  
 }  
  
 @Override  
 **public void** onSaveInstanceState(Bundle outState) {  
 **super**.onSaveInstanceState(outState);  
 outState.putInt(***STATE\_SELECTED\_POSITION***, **mCurrentSelectedPosition**);  
 }  
  
 @Override  
 **public void** onConfigurationChanged(Configuration newConfig) {  
 **super**.onConfigurationChanged(newConfig);  
 *// Forward the new configuration the drawer toggle component.* **mDrawerToggle**.onConfigurationChanged(newConfig);  
 }  
  
 @Override  
 **public void** onCreateOptionsMenu(Menu menu, MenuInflater inflater) {  
 *// If the drawer is open, show the global app actions in the action bar. See also  
 // showGlobalContextActionBar, which controls the top-left area of the action bar.* **if** (**mDrawerLayout** != **null** && isDrawerOpen()) {  
 inflater.inflate(R.menu.***global***, menu);  
 showGlobalContextActionBar();  
 }  
 **super**.onCreateOptionsMenu(menu, inflater);  
 }  
  
 @Override  
 **public boolean** onOptionsItemSelected(MenuItem item) {  
 **if** (**mDrawerToggle**.onOptionsItemSelected(item)) {  
 **return true**;  
 }  
  
 **if** (item.getItemId() == R.id.***action\_example***) {  
 Toast.*makeText*(getActivity(), **"Example action."**, Toast.***LENGTH\_SHORT***).show();  
 **return true**;  
 }  
  
 **return super**.onOptionsItemSelected(item);  
 }  
  
 */\*\*  
 \* Per the navigation drawer design guidelines, updates the action bar to show the global app  
 \* 'context', rather than just what's in the current screen.  
 \*/* **private void** showGlobalContextActionBar() {  
 ActionBar actionBar = getActionBar();  
 actionBar.setDisplayShowTitleEnabled(**true**);  
 actionBar.setNavigationMode(ActionBar.***NAVIGATION\_MODE\_STANDARD***);  
 actionBar.setTitle(R.string.***app\_name***);  
 }  
  
 **private** ActionBar getActionBar() {  
 **return** ((ActionBarActivity) getActivity()).getSupportActionBar();  
 }  
  
 */\*\*  
 \* Callbacks interface that all activities using this fragment must implement.  
 \*/* **public static interface** NavigationDrawerCallbacks {  
 */\*\*  
 \* Called when an item in the navigation drawer is selected.  
 \*/* **void** onNavigationDrawerItemSelected(**int** position);  
 }  
}