

Introducción a la Computación Móvil Parse Platform

Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas
Profesor: Carlos Andrés Parra
E-mail: ca.parraa@javeriana.edu.co



Parse Platform

The Complete Application Stack

Build applications faster with **object** and **file** storage,
user authentication, **push** notifications, dashboard and more out of the box.

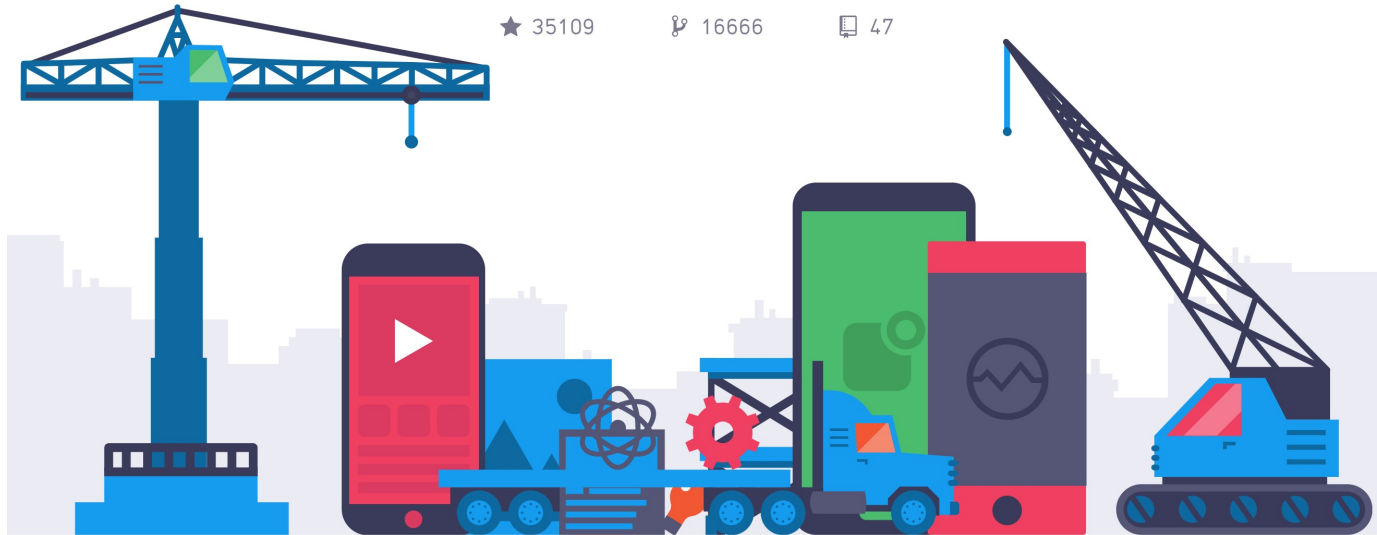
GET STARTED

COMMUNITY FORUM

★ 35109

🔗 16666

📄 47



[Parse](#)

Parse

- Plataforma open source para el desarrollo de aplicaciones móviles (Backend as a Service)
- Completamente gratis y open source pero sin infraestructura.
 - [Back4App](#) ofrece infraestructura para Parse con un costo asociado.
- Dentro de sus funcionalidades incluye:
 - Base de datos basada en MongoDB o PostgreSQL
 - Autenticación
 - LiveQueries (Equivalente a Realtime DB)
 - Push notifications (A través de FCM)

Parse

- Tiene soporte y SDKs para
 - iOS Nativo (Objective C y Swift)
 - Android Nativo (Java y Kotlin)
 - Frameworks Híbridos Web (JavaScript, PHP)
 - Frameworks Híbridos Nativos (Flutter, Unity, Xamarin)
 - Arduino

Parse

[Parse Server Guide](#) | [Parse \(parseplatform.org\)](#)

- Si no se cuenta con infraestructura, se deben aprovisionar una o varias máquinas para su funcionamiento.
- Pre-requisitos
 - Node 8 o superior, Mongodb 3.6, Python 2.x
- Si se quiere correr en una máquina local:

```
$ sh <(curl -fsSL https://raw.githubusercontent.com/parse-community/parse-server/master/bootstrap.sh)
$ npm -g mongodb-runner
$ mongodb-runner start
$ npm start
```

- Salvar el primer objeto

```
curl -X POST \-H "X-Parse-Application-Id: APPLICATION_ID" \-H "Content-Type: application/json" \-d
'{"score":123,"playerName":"Sean Plott","cheatMode":false}' \ http://localhost:1337/parse/classes/GameScore
```

Parse

- Si se quiere contar con Parse en alguna infraestructura diferente al *localhost*, es mejor aprovisionar una máquina en algún tipo de proveedor:
 - Google Cloud Platform, AWS, Azure
 - Crear una instancia con las características deseadas, e instalar alguna distribución de Linux, Docker y Docker-compose
 - Nube Javeriana
 - Solicitar una máquina virtual con alguna distribución de Linux e instalar Docker y Docker-compose

Docker

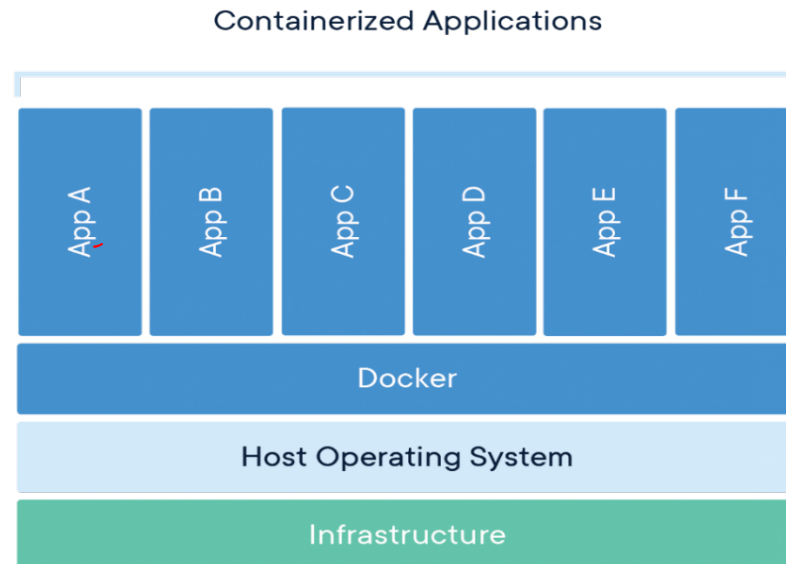


- Es una herramienta que se utiliza para automatizar la implementación de aplicaciones en contenedores livianos.
- Las aplicaciones pueden ser desplegadas en diferentes entornos sin los problemas de dependencias y de forma automatizada.



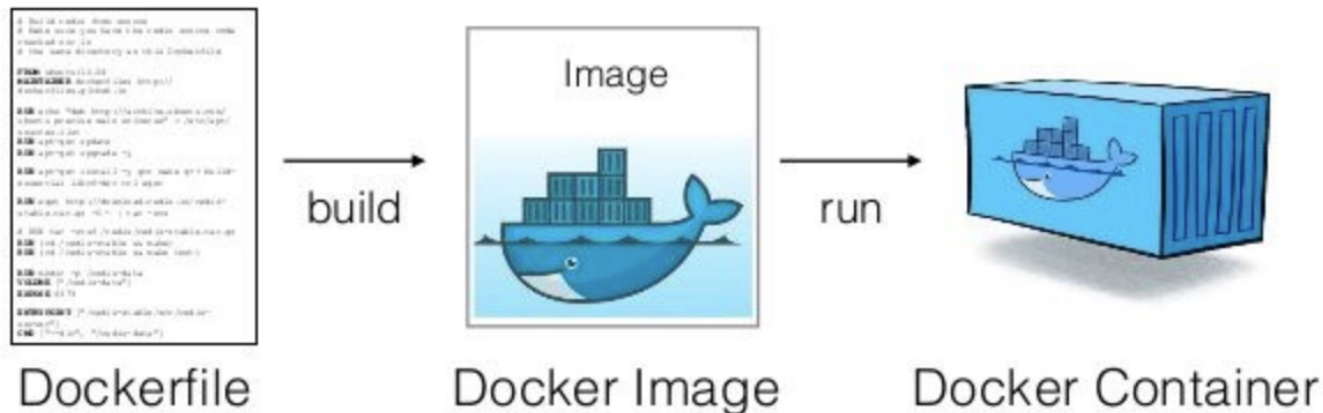
Docker

- Fácil de transportar a través de diferentes plataformas.
- La aplicación se ejecuta de forma aislada.
- Tiempo de actividad de arranque corto.
- Alta escalabilidad y eficiencia.
- Los volúmenes de datos se pueden reutilizar.



Docker

- Imagen -> Descriptor de los elementos necesarios para un contenedor. Se construye a partir de un dockerfile
- Contenedor -> Contenedor obtenido a partir de la construcción de una imagen que se puede ejecutar



Ejemplo dockerfile

```
FROM maven:3.6.0-jdk-11-slim

# application placed into /opt/app
RUN mkdir -p /home/apps/jerseyserver
WORKDIR /home/apps/jerseyserver

COPY pom.xml /home/apps/jerseyserver
RUN mvn install

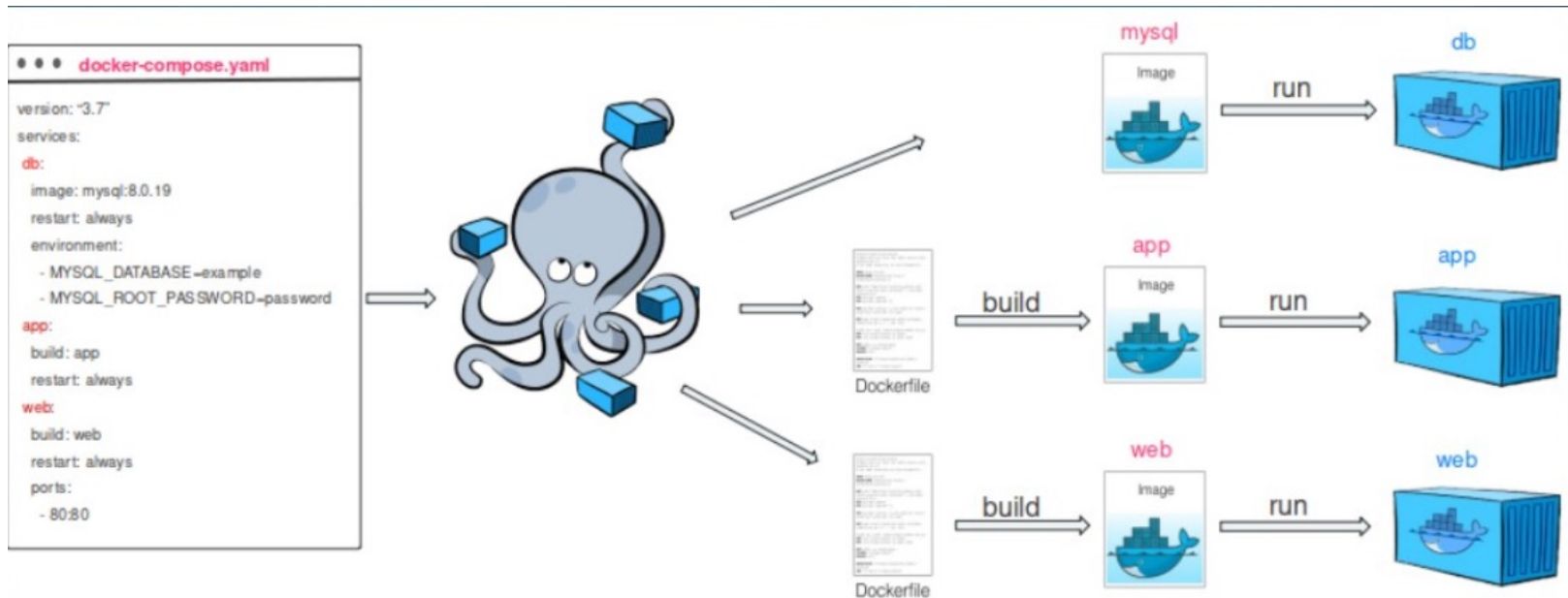
# rest of the project
COPY src /home/apps/jerseyserver/src
RUN mvn package

# local application port
EXPOSE 8181

CMD ["mvn", "exec:java"]
```

Docker-compose

- Herramienta para la ejecución de varios contenedores a partir de un archivo descriptivo (yaml)



Instalar Parse a través de Docker

- Primero instalar Docker y Docker-compose en la maquina host
- Docker
 - `sudo apt update`
 - `sudo curl -sSL https://get.docker.com/ | sh`
- Docker compose
 - `sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose`
 - `sudo chmod +x /usr/local/bin/docker-compose`
- Verificar que la instalación se hizo de forma exitosa
 - `sudo docker --version`
 - `sudo docker-compose --version`

<https://docs.docker.com/compose/install/>

Instalar Parse a través de docker

- Copiar y correr el archivo *parse-compose.yml* disponible en BrightSpace
- Antes de ejecutar el comando, definir en las líneas 12 y 20 los valores de llave maestra (**masterKey**) y aplicación (**appId**). Además en la línea 12 agregar los objetos que se quieran usar en tiempo real (LiveQuery)
- Para subir parse:
 - docker-compose -f parse-platform.yaml up
- Para detener parse y borrar los contenedores
 - docker-compose -f parse-platform.yaml down

Docker-compose Parse

```
services:
  mongo:
    image: mongo
    container_name: my-mongo
  parse-server:
    image: parseplatform/parse-server
    container_name: my-parse-server
    ports:
      - 1337:1337
    links:
      - mongo:my-mongo
    command: --appId myappid --masterKey mymasterkey --databaseURI
      mongodb://mongo/test --startLiveQueryServer
      --liveQuery "{\"classNames\":[\"SmartUser\"]}"
    depends_on:
      - mongo
  parse-dashboard:
    image: parseplatform/parse-dashboard
    container_name: my-parse-dashboard
    ports:
      - 4040:4040
    command: --dev --appId myappid --masterKey mymasterkey
      --serverURL "http://localhost:1337/parse"
    depends_on:
      - parse-server
```

Parse Dashboard

PARSE DASHBOARD 3.1.1

myappid

Core

Browser

- Role 0
- User 0

Webhooks

Jobs

Logs

Config

API Console

Push

Open Source Hub | GitHub | Docs

CLASS

User 0 objects • Public Read and Write enabled

<input checked="" type="checkbox"/>	objectId	String	emailVerified	Boolean	ACL	ACL	updatedAt	Date	authData	Object	username	String	createdAt
-------------------------------------	----------	--------	---------------	---------	-----	-----	-----------	------	----------	--------	----------	--------	-----------

No data to display

Add a row to store an object in this class.

Add a row

Habilitar reglas para firewall Google GPC

- Ir a la sección de reglas de firewall


INSTANCES INSTANCE SCHEDULE


VM instances are highly configurable virtual machines for running workloads on Google infrastructure. [Learn more](#)


Filter Enter property name or value ? ⋮


<input type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
<input type="checkbox"/>	✓	parse	us-west4-b			10.182.0.3 (nic0)	34.125.109.172 ↗	SSH ▾ ⋮


Related actions ⤴ HIDE

 **View billing report**
View and manage your Compute Engine billing

 **Monitor VMs**
View outlier VMs across metrics like CPU and network

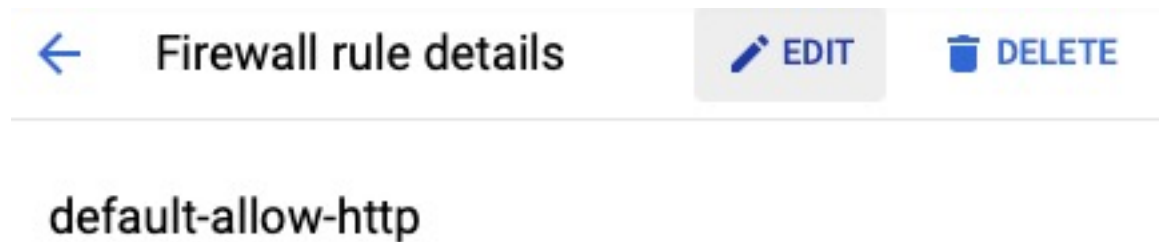
 **Explore VM logs**
View, search, analyze, and download VM instance logs

 **Set up firewall rules**
Control traffic to and from a VM instance


 **Patch management**
Schedule patch updates and view patch compliance on VM instances

Habilitar reglas para firewall Google GPC

- Seleccionar o crear una regla para http y editarla:



- Agregar la excepción para el puerto 1337 usado por Parse:

Protocols and ports 

☐ Allow all

☒ Specified protocols and ports

☒ tcp :

☐ udp :

☐ Other protocols

Demo!

- Máquina virtual
 - ~~- NubePUJ~~
 - Google
- Revisar instalación de docker
- Correr script de docker-compose
- Verificar instalación

Conectando Android con Parse

- Crear una aplicación
- Agregar las dependencias
 - `implementation "com.github.parse-community.Parse-SDK-Android:parse:1.26.0"`
 - `implementation 'com.squareup.okhttp3:logging-interceptor:3.8.1'`
 - `implementation 'com.github.parse-community:ParseLiveQuery-Android:1.2.2'`
- Crear una clase que extienda de aplicación y registrarla en el *manifest*
 - Una clase que extiende aplicación se puede usar para definir las propiedades para toda la aplicación y no sólo para una actividad en particular.
 - En este caso, se usa esta clase para definir la conexión con Parse

Clase Application

```
public class MyApp extends Application {  
    public static final String PARSE_SERVER = "http://{server_address}:1337/parse";  
  
    @Override  
    public void onCreate() {  
        super.onCreate();  
        Parse.initialize(new Parse.Configuration.Builder(this)  
            .applicationId("myappid").clientKey("")  
            // should correspond to Application Id env variable  
            .server(PARSE_SERVER )  
            .build());  
    }  
}
```

Clase Application Manifest

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.sesion2">
    <uses-permission android:name="android.permission.INTERNET"/>
    ...
    <application
        android:name=".MyApp"
        ...
```

Android y Parse

- Crear una actividad con dos campos de texto, un botón para persistir los datos de un usuario.
- Adicionalmente, cree un espacio en la pantalla, para mostrar una lista con los datos consultados desde la plataforma.
- En este espacio se van a ver en tiempo real las modificaciones sobre estos datos, al igual que en Firebase Realtime DB

The mockup shows a mobile application interface. At the top, there are two text input fields labeled "name..." and "last name...". Below these fields is a grey button with the text "SAVE". A horizontal line separates the input section from the list section. The list section has a header "USERS" in red text. Below the header is a large, empty white rectangular area, which is intended to display a list of users.

Escribir datos en la base de datos

- En vez de un ParseObject, se puede guardar un objeto propio, si se anota con @ParseObject, y se define un constructor por defecto y getters para cada atributo.

```
public void saveData(View v) {
    if(validateForm()) {
        Log.i(TAG, "Attempt to write on parse");
        ParseObject firstObject = new ParseObject(USER_CN);
        String name = this.name.getText().toString();
        String lastName = this.lastName.getText().toString();

        firstObject.put("name", name);
        firstObject.put("lastName", lastName);
        firstObject.saveInBackground(new SaveCallback() {
            @Override
            public void done(ParseException e) {
                if (e != null) {
                    Log.e(TAG, e.getLocalizedMessage());
                } else {
                    Log.d(TAG, "Object saved.");
                }
            }
        });
    }
}
```

Leer datos de la base de datos (una vez!)

- Para leer datos desde la base de datos, se puede hacer una consulta de la siguiente manera:

```
private void loadUsers(){
    ParseQuery<ParseObject> query = ParseQuery.getQuery(USER_CN); //SmartUser
    query.findInBackground(new FindCallback<ParseObject>() {
        @Override
        public void done(List<ParseObject> objects, ParseException e) {
            if(objects!=null) {
                listUsers.removeAllViews();
                for (ParseObject row : objects) {
                    String name = (String) row.get("name");
                    String lastName = (String) row.get("lastName");
                    TextView listItem = new TextView(getApplicationContext());
                    listItem.setText(name + " " + lastName);
                    listItem.setTextSize(20);listItem.setHeight(200);
                    listUsers.addView(listItem);
                }
            }
        }
    });
}
```


Suscribirse a cambios en la base de datos

//Parse live query attributes

```
ParseLiveQueryClient parseLiveQueryClient;
```

```
ParseQuery<ParseObject> parseQuery; //suscription to changes
```

//en onCreate

```
parseLiveQueryClient = ParseLiveQueryClient.Factory.getClient();
```

```
parseQuery = ParseQuery.getQuery(USER_CN);
```

```
SubscriptionHandling<ParseObject> subscriptionHandling = parseLiveQueryClient.subscribe(parseQuery);
```

//Reaccionar a cualquier evento

```
subscriptionHandling.handleEvents(new SubscriptionHandling.HandleEventsCallback<ParseObject>() {
```

```
    @Override
```

```
    public void onEvents(ParseQuery<ParseObject> query, SubscriptionHandling.Event event, ParseObject object) {
```

```
        dataChanged(query); //Método propio que reacciona al evento
```

```
    }
```

```
});
```

//Reaccionar a un evento de creación

```
subscriptionHandling.handleEvent(SubscriptionHandling.Event.CREATE, new
```

```
    SubscriptionHandling.HandleEventCallback<ParseObject>() {
```

```
    @Override
```

```
    public void onEvent(ParseQuery<ParseObject> query, ParseObject object) {
```

```
        dataCreated(object); //Método propio que reacciona al evento
```

```
    }
```

```
});
```

Reaccionar a cambios

```
private void dataChanged(ParseQuery<ParseObject> query) {  
    Log.i(TAG, "An event happened!!");  
    query.findInBackground(new FindCallback<ParseObject>() {  
        @Override  
        public void done(List<ParseObject> objects, ParseException e) {  
            listUsers.removeAllViews();  
            for(ParseObject row : objects){  
                String name = (String) row.get("name");  
                String lastName = (String) row.get("lastName");  
                TextView listItem = new TextView(getApplicationContext());  
                listItem.setText(name+ " "+lastName);  
                listItem.setTextSize(20);  
                listItem.setHeight(200);  
                listUsers.addView(listItem);  
            }  
        }  
    });  
}
```

//Actualiza la lista de
usuarios en tiempo real

Demo!!

Notificaciones

Android

Notificaciones

- Una notificación es un mensaje que se produce fuera de los layouts típicos de la aplicación, e incluso puede ocurrir cuando la aplicación no esta corriendo.
- Los elementos esenciales son un ícono, un titulo y un texto para mostrar la notificación.
- La notificación puede tener asociada una acción para lanzar una actividad propia de la aplicación

Notificaciones

- Desde Android 26 (Oreo) es necesario construir un canal para las notificaciones

```
private void createNotificationChannel() {  
    // Create the NotificationChannel, but only on API 26+ because  
    // the NotificationChannel class is new and not in the support library  
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {  
        CharSequence name = "channel";  
        String description = "channel description";  
        int importance = NotificationManager.IMPORTANCE_DEFAULT;  
        //IMPORTANCE_MAX MUESTRA LA NOTIFICACIÓN ANIMADA  
        NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name, importance);  
        channel.setDescription(description);  
        // Register the channel with the system; you can't change the importance  
        // or other notification behaviors after this  
        NotificationManager notificationManager = getSystemService(NotificationManager.class);  
        notificationManager.createNotificationChannel(channel);  
    }  
}
```

Notificaciones

- Crear la notificación con datos básicos. CHANNEL_ID es un string único que identifica al canal, necesario a partir de Android 26, OREO
- Atributos de la clase

```
public static String CHANNEL_ID = "MyApp";  
int notificationId = 0;
```

- Creación de la notificación

```
NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this,  
CHANNEL_ID);  
mBuilder.setSmallIcon(R.drawable.noticon);  
mBuilder.setTitle("Titulo Notificación");  
mBuilder.setText("Contenido de la Notificación");  
mBuilder.setPriority(NotificationCompat.PRIORITY_DEFAULT);
```

Notificaciones

- Asociar una actividad que se lanza como resultado de tocar la notificación:

//Acción asociada a la notificación

```
Intent intent = new Intent(this, NotificationTappedActivity.class);  
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |  
Intent.FLAG_ACTIVITY_CLEAR_TASK);  
PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent, 0);  
mBuilder.setContentIntent(pendingIntent);  
mBuilder.setAutoCancel(true); //Remueve la notificación cuando se toca
```


Notificaciones

- Mostrar la notificación a través del manager

```
int notificationId = 001;
```

```
NotificationManagerCompat notificationManager =
```

```
NotificationManagerCompat.from(this);
```

```
// notificationId es un entero unico definido para cada notificacion que se lanza
```

```
notificationManager.notify(notificationId, mBuilder.build());
```

Servicios

Android

Servicios

- Un servicio es otro componente de Android que no tiene asociada una interfaz gráfica y se encarga de hacer tareas en primero y segundo plano.
- Se puede usar para por ejemplo, recibir notificaciones de cambios en el Backend y notificar al usuario.
- Un servicio puede seguir activo a pesar de que la aplicación se haya cerrado.

Servicios

- Foreground
 - A foreground service performs some operation that is noticeable to the user. For example, an audio app would use a foreground service to play an audio track. Foreground services continue running even when the user isn't interacting with the app.
- Background
 - A background service performs an operation that isn't directly noticed by the user. For example, if an app used a service to compact its storage, that would usually be a background service.

Tipos de Servicio

- Service
 - Se usa cuando se deben atender muchas peticiones al tiempo
- ServiceIntent
 - Se usa cuando no es necesario atender múltiples peticiones al tiempo, más simple de implementar, sólo se sobrescriben los métodos necesarios

Servicio

```
public class HelloIntentService extends IntentService {
    public HelloIntentService() {
        super("HelloIntentService");
    }
    @Override
    protected void onHandleIntent(Intent intent) {
        // Trabajo que debe hacer el servicio
        // Por ahora solo esperar 5 segundos
        try {
            Thread.sleep(5000);
            Log.i(TAG, "Servicio en ejecución" );
        } catch (InterruptedException e) {
            // Restore interrupt status.
            Thread.currentThread().interrupt();
        }
    }
}
```

Código del Servicio

```
Intent intent = new Intent(MainActivity.this, HelloIntentService.class);
startService(intent);
```

Lanzar el Hilo

```
<service android:name=".HelloIntentService" />
```

Manifest, dentro
del tag application

Cambios desde Android O

Desde android 8 se limitan los tiempos de ejecución en background.

- **Service** – Tarea que se ejecuta en background. Corre en el mismo hilo de la actividad que lo invoca. Si es muy demorado debería crearse un hilo internamente en el servicio para no afectar el rendimiento de quien lo llama
 - **Desventaja**: Corre en el mismo hilo
- **IntentService** – Tarea que se ejecuta en background pero crea su propio hilo.
 - **Desventaja** : El trabajo que se le asigne se pierde si quien lo llama se Cierra. **Deprecated since Android 11!**
- **JobIntentService** – Similar al IntentService, pero quien lo ejecuta puede terminarlo en cualquier momento y puede reiniciarlo cuando él mismo reinicie.

JobIntentService

<https://developer.android.com/reference/androidx/core/app/JobIntentService>

```
public class HelloJobIntentService extends JobIntentService {
```

```
    private static final int JOB_ID = 12;
```

Id para el Servicio

```
    public static void enqueueWork(Context context, Intent intent) {  
        enqueueWork(context, HelloJobIntentService.class, JOB_ID, intent);  
    }
```

Método auxiliar
para manejar una
cola de tareas

@Override

```
    protected void onHandleWork(@NonNull Intent intent){  
        int miliSeconds = intent.getIntExtra("miliSeconds", 10000);  
        try {  
            Thread.sleep(miliSeconds);  
            Log.i(HomeActivity.TAG, "Service Finished Waiting" );  
        } catch (InterruptedException e) {  
            Thread.currentThread().interrupt();  
        }  
    }
```

Código a ejecutar
en background

```
}
```


JobIntentService

- Hay que utilizar el permiso WAKE_LOCK para versiones inferiores a Oreo

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

- Hay que definir el permiso BIND_JOB_SERVICE dentro del tag de servicio para versiones Android Oreo y posterior

```
<application>
```

```
<!-- ... -->
```

```
<service
```

```
    android:name=".services.HelloJobIntentService"
```

```
    android:permission="android.permission.BIND_JOB_SERVICE" />
```

```
</application>
```

JobIntentService

- Y para invocarlo:

```
Intent intent = new Intent(HomeActivity.this, HelloJobIntentService.class);  
intent.putExtra("miliSeconds", 5000);  
HelloJobIntentService.enqueueWork(HomeActivity.this, intent);  
Log.i(TAG, "After the call to the service");
```

- Si la aplicación se cierra, el servicio también!

Servicios en Background y en el Boot del SO

- Si se quieren recibir cambios del servidor aún sin la aplicación en ejecución, es necesario crear un servicio en background y lanzarlo en cada arranque del Sistema Operativo

```
public class Autostart extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent arg) {
        if(arg.getAction() == Intent.ACTION_BOOT_COMPLETED) {
            Intent intent = new Intent(context, BackgroundBootService.class);
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
                context.startForegroundService(intent);
            } else {
                context.startService(intent);
            }
            Log.i("Autostart", "started");
        }
    }
}
```

Registro en el manifest

```
<receiver android:name=".boot.Autostart" android:exported="true">  
  <intent-filter>  
    <action android:name="android.intent.action.BOOT_COMPLETED" />  
  </intent-filter>  
</receiver>
```

- Autostart es el nombre de la clase que extiende de *BroadcastReceiver* y en este caso se encuentra en el paquete *boot*

Background Boot Service

```
public class BackgroundBootService extends Service {
```

```
    @Override
```

```
    public void onCreate() {  
        super.onCreate();  
        createNotificationChannel();  
    }
```

```
    @Override
```

```
    public IBinder onBind(Intent intent) {return null;}
```

```
    public void onDestroy() {
```

```
        Log.i(TAG, "BOOT Service has been stopped");
```

```
        Toast.makeText(this, "BOOT service stopped", Toast.LENGTH_LONG).show();
```

```
        if (parseQuery != null && parseLiveQueryClient != null)
```

```
            parseLiveQueryClient.unsubscribe(parseQuery);
```

```
    }
```

```
//..
```

Background Boot Service

Requiere una notificación!!

@Override

```
public int onStartCommand(Intent intent, int flags, int startId) {  
    startForeground(2, buildComplexNotification("Service Started", "Connected to  
        NubePUJ", R.drawable.ic_baseline_anchor_24, MainActivity.class));
```

```
    Toast.makeText(this, "SmartPUJ Started", Toast.LENGTH_LONG).show();  
    parseLiveQueryClient = ParseLiveQueryClient.Factory.getClient();  
    parseQuery = ParseQuery.getQuery("SmartUser");  
    subscriptionHandling = parseLiveQueryClient.subscribe(parseQuery);  
    subscriptionHandling.handleEvents(new  
        SubscriptionHandling.HandleEventsCallback<ParseObject>() {  
        @Override  
        public void onEvents(ParseQuery<ParseObject> query,  
            SubscriptionHandling.Event event, ParseObject object) {  
            dataChanged(query);  
        }  
    });  
    return START_STICKY;  
}
```

Demo!

Notificaciones PUSH

- Aunque se puede usar de la forma que se presentó, hay varios inconvenientes:
 - Cada aplicación que requiera notificaciones tendría un servicio en background lo que iría en detrimento del rendimiento del dispositivo
 - Si se quieren escuchar los cambios en el backend sin que la aplicación este corriendo, se debe usar un ForeGround service (desde android 8), lo que implica que siempre existirá una notificación que no se puede quitar de la barra de notificaciones, a menos que se detenga el servicio.
 - No es fácil hacer este proceso desde plataformas híbridas y algunos fabricantes pueden impedir la ejecución de estos servicios

Notificaciones PUSH

¿Qué alternativas hay?

- En Android y Apple
 - Firebase Cloud Messaging (FCM), incluido en la capa gratis de Firebase
- Sólo en Apple
 - Apple Push Notifications Service (APNs) -> FCM lo usa

- Ventajas

- Se utilizan los procesos que ya corren en el dispositivo para recibir notificaciones de distintas aplicaciones y backends (Google APIs)
- Implementación mucho más simple, no es necesario correr procesos en background ni crear servicios en el arranque del dispositivo.

- Desventajas

- Se crea una dependencia fuerte con el proveedor del servicio de notificaciones.
- Costos y limitaciones pueden cambiar a futuro

Servicios REST

REST usando Volley

- Volley es una librería construida por Google para consumir servicios REST

- Para usarla se debe incluir la dependencia en gradle:

implementation **'com.android.volley:volley:1.1.1'**

- Mas información de documentación y versiones:

<https://github.com/google/volley/releases>

REST usando Volley

```
public void consumeRESTVolley(){
    RequestQueue queue = Volley.newRequestQueue(this);
    String url = "https://restcountries.eu/rest/v2/";
    String path = "currency/cop";
    String query = "?fields=name;capital";
    StringRequest req = new StringRequest(Request.Method.GET, url+path+query,
        new Response.Listener() {
            @Override
            public void onResponse(Object response) {
                String data = (String)response;
                restResponse.setText(data);
            },
            new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {
                    Log.i("TAG", "Error handling rest invocation"+error.getCause());
                }
            }
        );
    queue.add(req);
}
```

Requiere el
permiso de
internet en el
manifest!!

REST usando apache.http

- Forma Antigua, hasta Android 6
- Primero agregar la dependencia

```
android { ...  
    useLibrary 'org.apache.http.legacy'  
}
```

- Incluir el permiso de internet en el manifest

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

- Definir un AsyncTask
 - Tareas que permiten consumir datos de un servidor externo de forma asíncrona sin bloquear la actividades de android

AsyncTask

- The three types used by an asynchronous task are the following:
 - Params, the type of the parameters sent to the task upon execution.
 - Progress, the type of the progress units published during the background computation.
 - Result, the type of the result of the background computation.
- Not all types are always used by an asynchronous task. To mark a type as unused, simply use the type Void:

```
private class MyTask extends AsyncTask<Void, Void, Void> { ... }
```

Async Task

- [onPreExecute\(\)](#), invoked on the UI thread before the task is executed. This step is normally used to setup the task, for instance by showing a progress bar in the user interface.
- [doInBackground\(Params...\)](#), invoked on the background thread immediately after [onPreExecute\(\)](#) finishes executing. This step is used to perform background computation that can take a long time.
- [onProgressUpdate\(Progress...\)](#), invoked on the UI thread after a call to [publishProgress\(Progress...\)](#). This method is used to display any form of progress in the user interface while the background computation is still executing.
- [onPostExecute\(Result\)](#), invoked on the UI thread after the background computation finishes. The result of the background computation is passed to this step as a parameter.

AsyncTask

```
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {  
    protected Long doInBackground(URL... urls) {  
        int count = urls.length;  
        long totalSize = 0;  
        for (int i = 0; i < count; i++) {  
            totalSize += Downloader.downloadFile(urls[i]);  
            publishProgress((int) ((i / (float) count) * 100));  
            // Escape early if cancel() is called  
            if (isCancelled()) break;  
        }  
        return totalSize;  
    }  
  
    protected void onProgressUpdate(Integer... progress) {  
        setProgressPercent(progress[0]);  
    }  
  
    protected void onPostExecute(Long result) {  
        showDialog("Downloaded " + result + " bytes");  
    }  
}
```


Consumir Servicios REST

```
private class AsyncRestConsumer extends AsyncTask<String, Void, String> {  
protected String doInBackground(String... urls) {  
    public static final String REST_COUNTRIES = "https://restcountries.eu/rest/v2";  
    String queryParams="";  
    if(urls!=null && urls.length>0 )  
        queryParams= urls[0];  
    String pathParams = "/lang/es";
```

→ urls viene de la invocación del AsyncTask

```
    HttpGet request = new HttpGet(REST_COUNTRIES + pathParams);  
    HttpClient client = new DefaultHttpClient();  
    HttpResponse httpResponse;  
    try {  
        httpResponse = client.execute(request);  
        responseCode = httpResponse.getStatusLine().getStatusCode();  
        message = httpResponse.getStatusLine().getReasonPhrase();  
        HttpEntity entity = httpResponse.getEntity();  
        if (entity != null) {  
            response = EntityUtils.toString(entity);  
        }  
    } catch (Exception e) {  
        Log.d("ERROR", e.toString() + e.getCause());
```

```
    }  
    return response;
```

→ response se pasa a postExecute() como parámetro

Consumir Servicios REST

```
protected void onPostExecute(String response) {
```

```
JSONArray result;
```

```
try {
```

```
    result = new JSONArray(response);
```

```
    for(int i=0; i<result.length(); i++)
```

```
    {
```

```
        JSONObject jo = (JSONObject) result.get(i);
```

```
        Log.d("TAG", "Json Object "+jo.toString());
```

```
        String name = (String)jo.get("name");
```

```
        String capital = (String) jo.get("capital");
```

```
        TextView tv = new TextView(RestClientActivity.this);
```

```
        tv.setText(name+" --> "+capital);
```

```
        lResult.addView(tv);
```

```
    }
```

```
} catch (JSONException e) {
```

```
    e.printStackTrace();
```

```
}
```

Viene del retorno de doInBackground()

Layout de la actividad principal donde se agrega un renglón por cada país cuando se tienen los resultados

Para lanzar la actividad

new AsyncRestConsumer().execute();

Ejercicio 4

- Utilice los servicios de Rest Countries (<http://restcountries.eu/rest/v2>) para consultar la información de países y mostrarla en la pantalla de la siguiente manera:
 - Todos los países (nombre y capital) de la zona económica europea
 - El nombre, la capital y la población de los países cuya moneda sea el yen japonés
 - Los países (nombre) donde se habla francés
- Defina una actividad que permita seleccionar la zona económica y liste todos los países encontrados en una lista (scrollable).
- Finalmente, defina un detalle para cada país, al igual que en el taller 2, pero esta vez, use los datos provistos por el servicio restcountries.