

PROYECTO MY CARTERA

Desarrollado con Apache/Cordova

José Javier García Romero DAW 2015/2016

Introducción

- My cartera es una aplicación que pretende ser la agenda de un equipo de comerciales
- La aplicación permite almacenar un listado de comerciales, un listado de empresas e ir grabando las visitas de los comerciales a estas empresas para posteriormente visualizar esas rutas en un mapa.
- Esta aplicación está disponible para ser utilizada en dispositivos Android, y en breve estará disponible como WebApp.
- Los desarrollos Web y móvil son paralelos y están relacionados.

LA BASE DE DATOS

| empresas | |
|------------|-------------|
| id | INT(11) |
| cif | VARCHAR(20) |
| nombre | VARCHAR(80) |
| direccion | VARCHAR(80) |
| provincia | VARCHAR(30) |
| poblacion | VARCHAR(30) |
| cp | VARCHAR(5) |
| tif | INT(11) |
| comercial | INT(11) |
| fecha_alta | DATETIME |
| contacto | VARCHAR(50) |
| Indexes | |

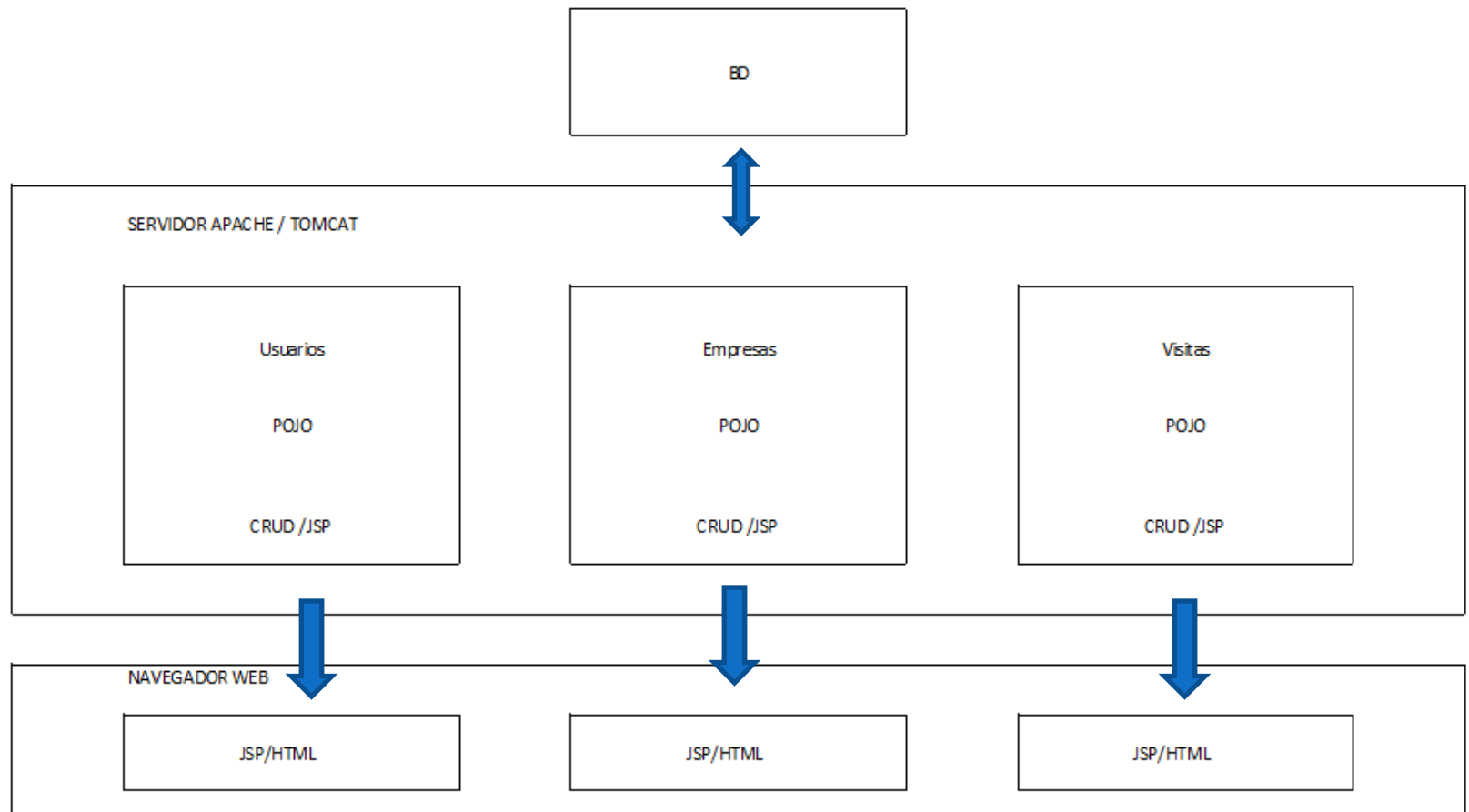
| usuarios | |
|-----------|-------------|
| id | INT(11) |
| login | VARCHAR(15) |
| nombre | VARCHAR(30) |
| apellidos | VARCHAR(30) |
| fnac | DATETIME |
| fu | DATETIME |
| pass | VARCHAR(40) |
| nif | VARCHAR(45) |
| es_adm | BIT(1) |
| Indexes | |

| visitas | |
|-------------|--------------|
| usuarios_id | INT(11) |
| empresas_id | INT(11) |
| fecha | DATETIME |
| resultado | MEDIUMTEXT |
| motivo | VARCHAR(150) |
| id | INT(11) |
| Indexes | |

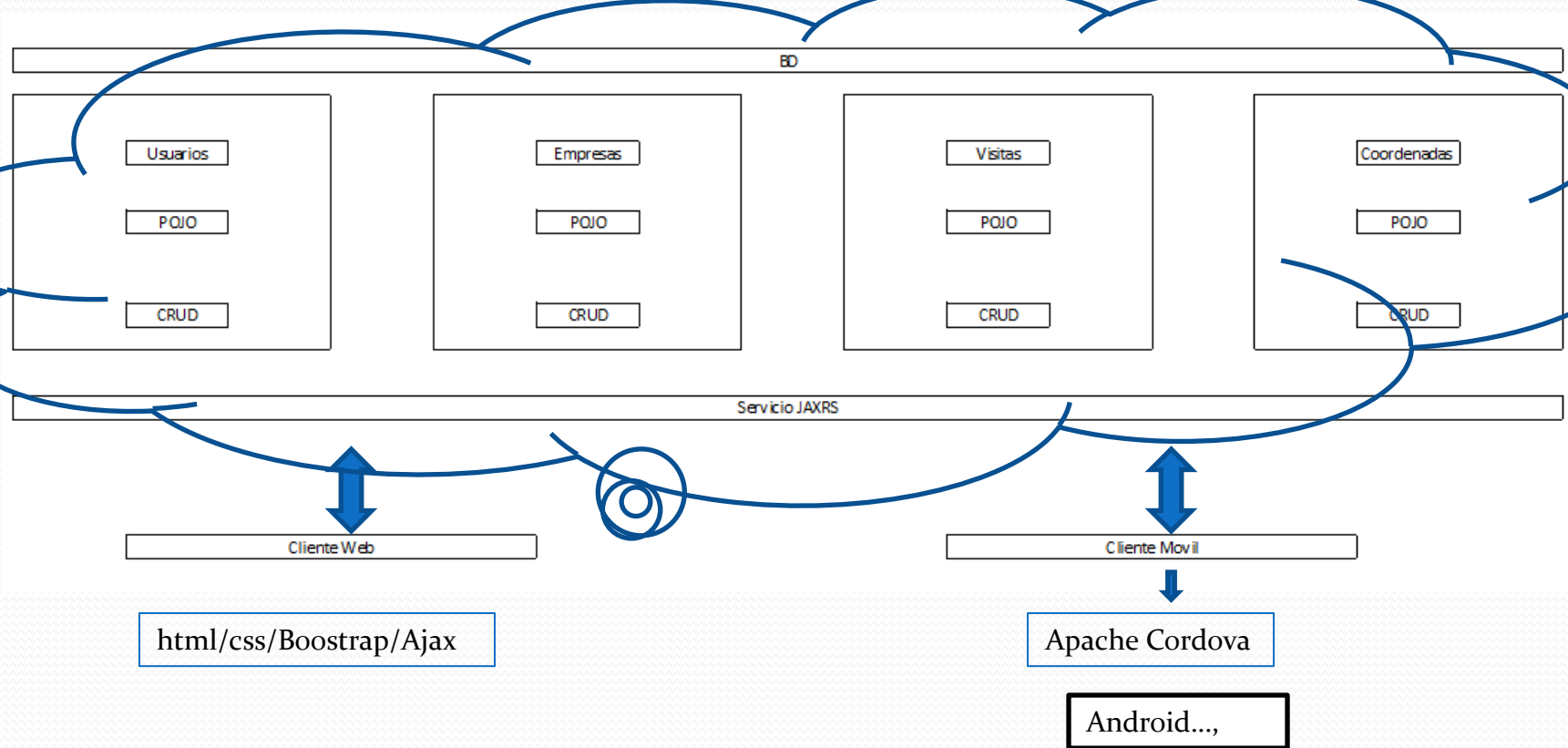
| coordenadas | |
|-------------|-------------|
| GLatitud | VARCHAR(50) |
| GLongitud | VARCHAR(50) |
| Date | TIMESTAMP |
| usuarios_id | INT(11) |
| id | INT(11) |
| Indexes | |

Añadida en 2016

Resumen Arquitectura Antiguo Proyecto 2011



Nueva Arquitectura



¿Por qué Apache Cordova?

Applications Types

Native Application



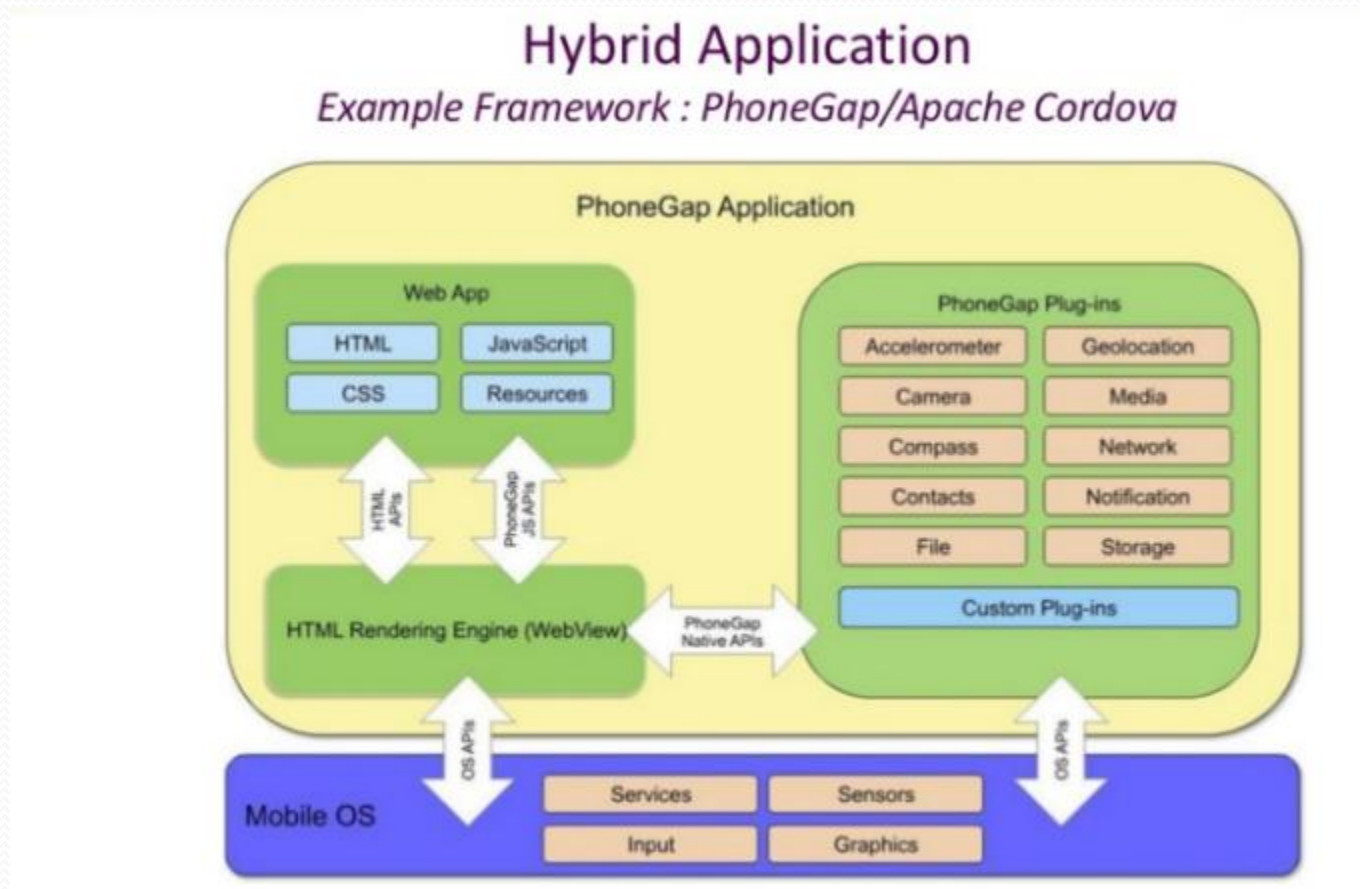
Web Application



Hybrid Application



¿Por qué Apache Cordova?



¿Por qué Apache Cordova?

Why mobile development is so complicated

Multiple OS, Multiple languages

```

<include from="http://schemas.android.com/apk/res/android"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimens/activity_horizontal_margin"
    android:paddingRight="@dimens/activity_horizontal_margin"
    android:paddingTop="@dimens/activity_vertical_margin"
    android:paddingBottom="@dimens/activity_vertical_margin"
    android:background="@drawable/bg04"/>

```

Copyright

```

width: layout_width*mag_offset;
width: layout_height*mag_offset;
width: h*mag_offset;

```



כוננים

```

<Grid>
  <ItemsSource="{Binding Path=Grid.Rows}">
    <Grid.Resources>
      <DataTemplate x:Key="TweetList">
        <Grid>
          <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
          </Grid.RowDefinitions>
          <TextBlock Grid.Row="0" TextWrapping="Wrap" Text="{Binding Text}" />
          <TextBlock Grid.Row="1" HorizontalAlignment="Right" Text="{Binding CreatedDate}" />
          <Grid>
            <Grid.RowDefinitions>
              <RowDefinition/>
            </Grid.RowDefinitions>
            <TextBlock Text="Tweet List" FontSize="24" HorizontalAlignment="Center" Margin="0" />
          </Grid>
          <Grid>
            <Grid.RowDefinitions>
              <RowDefinition/>
            </Grid.RowDefinitions>
            <TextBlock VerticalContentAlignment="Stretch"
              Opacity="0.5"
              FontSize="24" />
          </Grid>
        </Grid>
      </DataTemplate>
    </ItemsSource>
  </Grid>
</Grid>

```



Windows Phone

```

- (CGRect)getUIViewRectOfCapCircleSize {
    if (CGRect)
        return CGRectMake(0, 0, 0, 0);

    float r;
    CGRectGetViewRectOfCapCircleSize(r);
    return CGRectMake(r, r, r, r);
}

- (CGRect)getUIViewRectOfCapLong {
    if (CGRect)
        return CGRectMake(0, 0, 0, 0);

    float r;
    CGRectGetViewRectOfCapCircleSize(r);
    return CGRectMake(r, r, r, r);
}

```

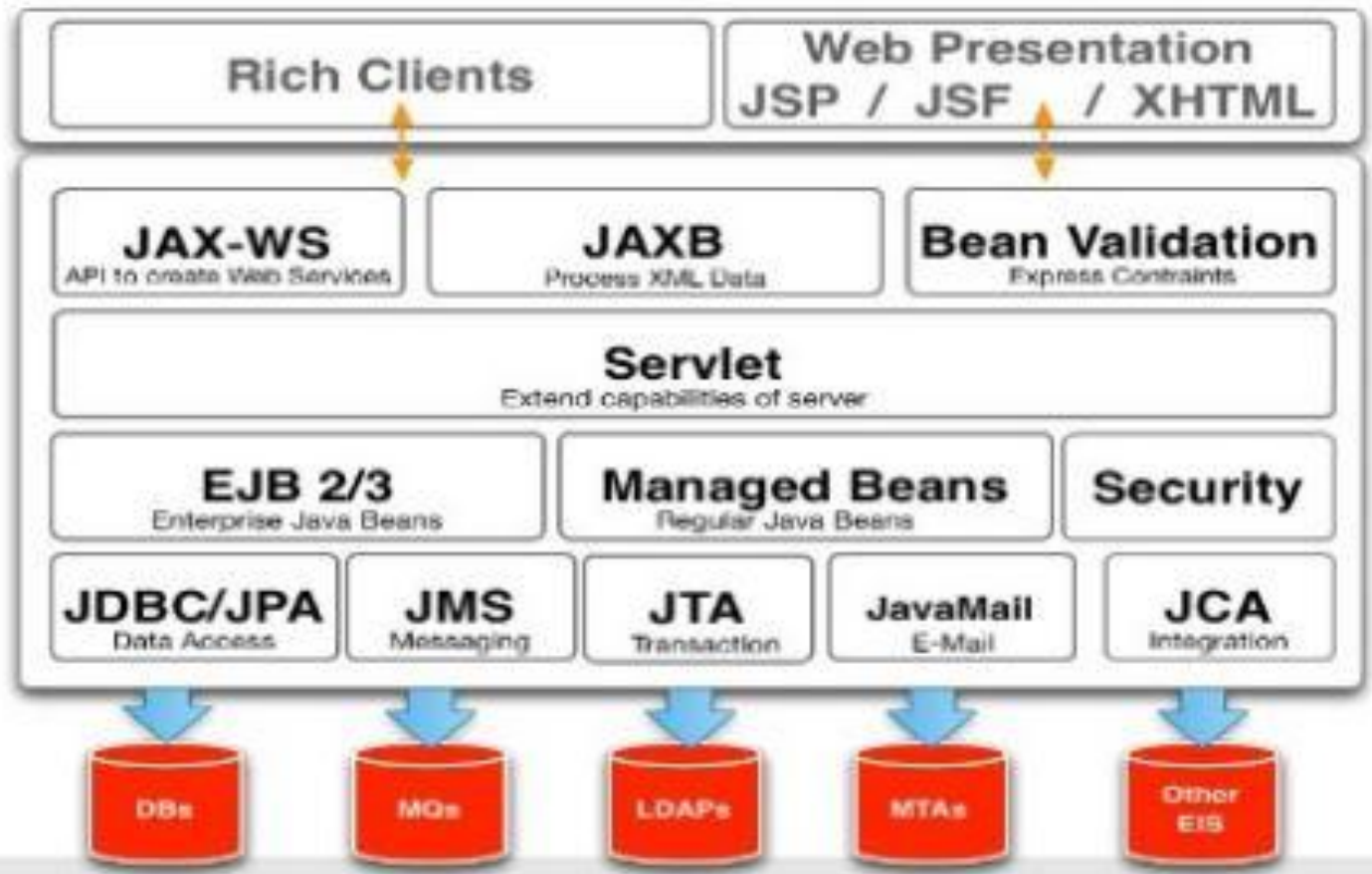


Not a problem for
Chuck Norris. What
about you?



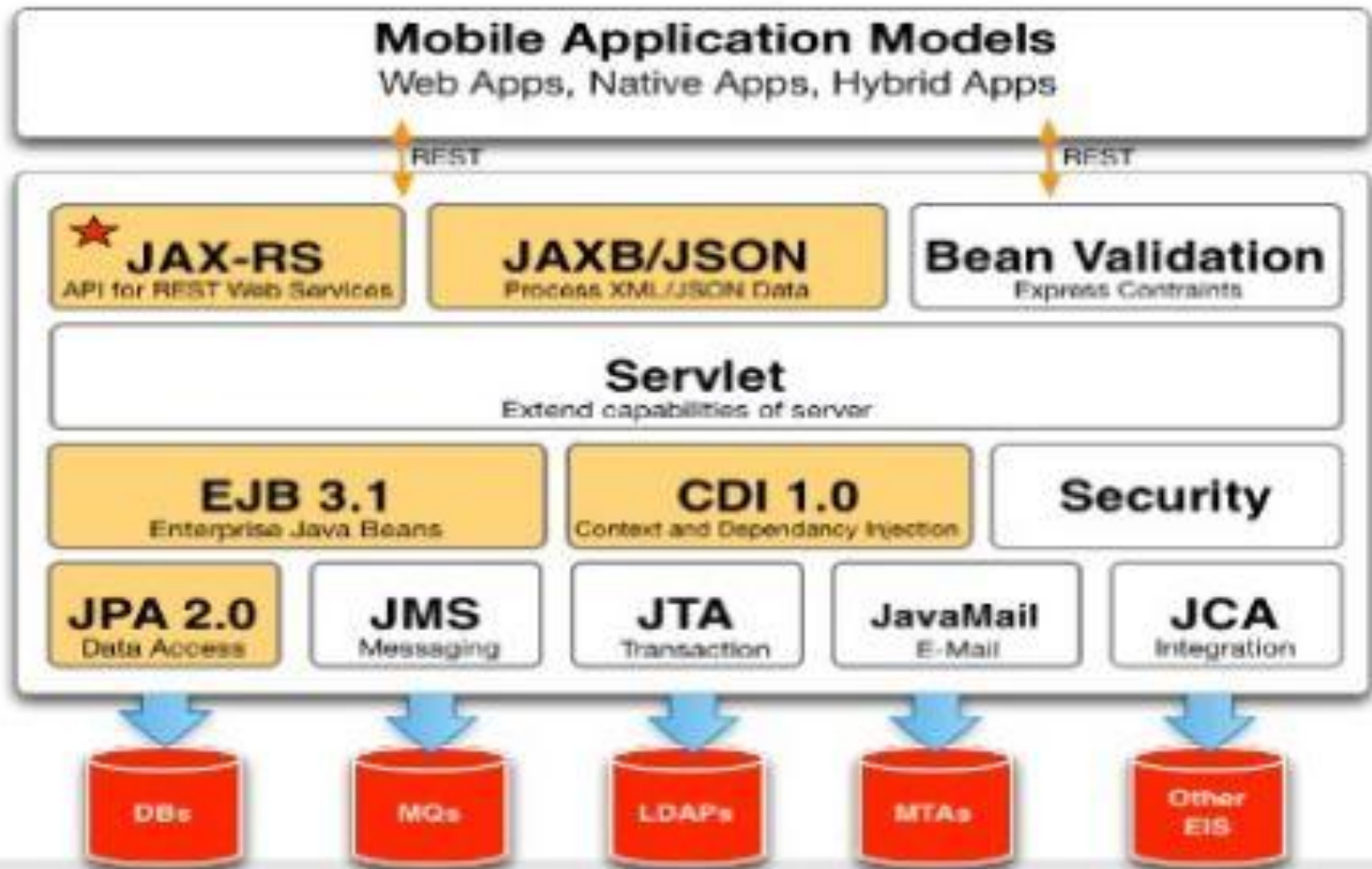
Traditional Java (2) EE technologies usage

Current situation



Modernize to Java EE 6 technologies

Future situation to enable mobile capabilities

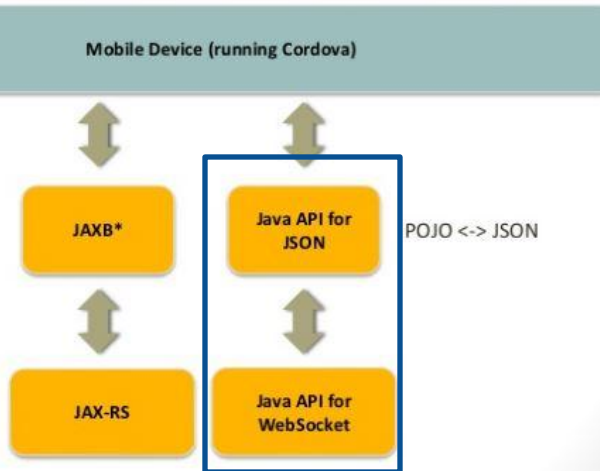


REST fits well with mobile strategy

- REST is flexible and adequate for Client-Server communication
- Communication happens over HTTP using REST styles
- JSON over HTTP/HTTPS should be used for data-interchange
- Stateless, Cacheable, Lightweight, Scalable
- Using HTTP verbs (DELETE, GET, POST, PUT)
- Using Web technologies and security standards
- Fully supported in JBoss EAP (Java EE 6 compliant environment) by a JAX-RS implementation (RESTEasy)

Java EE 7 + Cordova

Relating Java EE 7 to Mobile



Java EE 7 + Cordova

REST

- JAX-RS 2.0 is the REST development API for Java
- Major upgrade with Java EE 7. (JSR-339)
 - Client API, Async, Validation, Filters/Handlers, Interceptors, and Content Negotiation.
- Server and client
- Integrates with JAXB and Bean Validation
- Annotation based, declarative
 - `@Path`, `@GET`, `@POST`, `@PUT`, `@DELETE`,
- Pluggable and extensible.

Java EE 7 + Cordova

REST Example with JAX-RS

```
@Path("/race")
public class RaceServiceRS {

    @POST
    @Path("/join/{raceId}/{raceClass}")
    public Response joinRace(@HeaderParam("x-secure-token")
        String token,
        @PathParam("raceId") long raceId,
        @PathParam("raceClass") long raceClass) {
        logger.log(Level.INFO, "Joining the race: {0}", raceId);
        AuthorizationInfo info
            = authorizationBean.checkAuthorization(token);
        raceBean.joinRace(info.getAccount(), raceId, raceClass);
        return Response.ok().build();
    }
}
```

HTTP POST to `http://<server>/n34/race/join/<id>/<class>`

PROBLEMAS

Presentation layer migration

Standard like **JSP** with **Servlet** or **JSF 2.0/2.1** and non-standard web frameworks like **Struts**, **Tapestry**, **GWT**, ... do not have a **migration path** in a **mobile strategy**

Client side needs a clean / fresh start

Mobile-First strategy

creating pages, UI components that address the constraints of mobile, then progressively enhances the experience to other screen spaces, features, and more

Mobile Overview

Native HTML5 vs. Web Apps

~~JSF/JSP/Facelets~~

~~HTTP Session~~

ACCESS TO DEVICE CAPABILITIES

NATIVE APPS

- Single platform affinity
- Written with platform SDKs
- Must be written for each platform
- Access to all native APIs
- Faster graphics performance
- AppStore distribution

HYBRID APPS

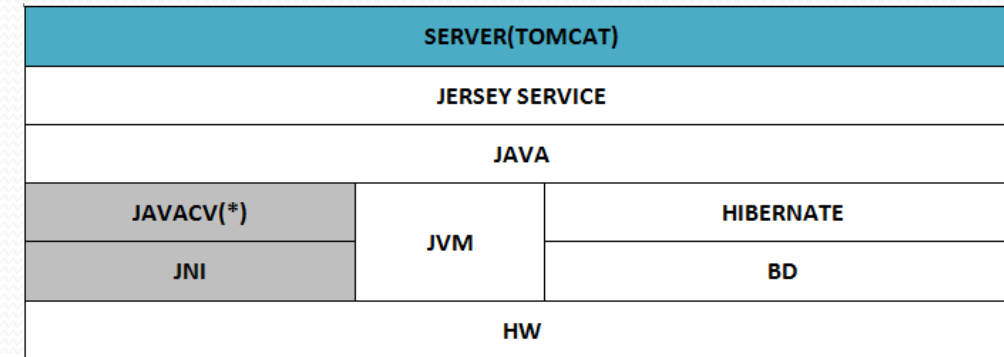
- Cross-platform affinity
- Written with web technologies (HTML5, CSS3 and JavaScript)
- Runs locally on the device, supports offline
- Access to native APIs
- AppStore distribution

MOBILE WEB APPS

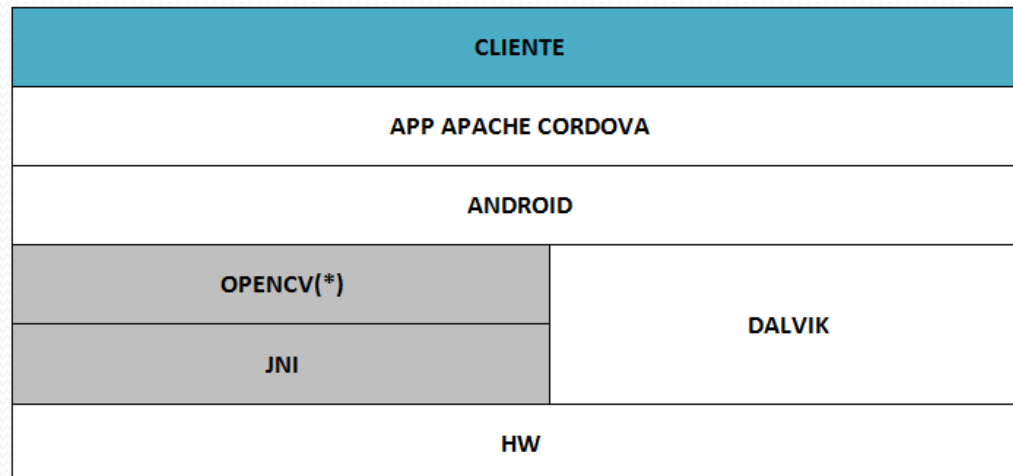
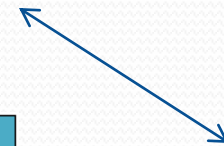
- Cross-platform affinity
- Written with web technologies (HTML, CSS, JavaScript, or Server-side (PHP, ASP.NET, etc.))
- Runs on web server, viewable on multiple devices
- Centralized updates

PLATFORM AFFINITY

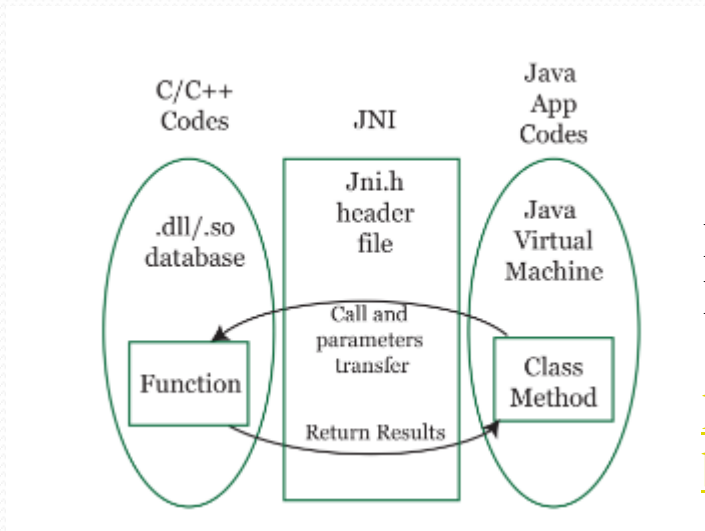
ARQUITECTURA DEL PROYECTO



AJAX / JERSEY



JAVA NATIVE INTERFACE



El soporte JNI de Android Studio todavía es Experimental

<http://tools.android.com/tech-docs/new-build-system/gradle-experimental>

BIBLIOTECAS/APIS DETECCIÓN RECONOCIMIENTO FACIAL

OpenCV

<http://opencv.org/>

OpenCV es una [biblioteca](#) libre de visión artificial originalmente desarrollada por [Intel](#). Se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicaciones de control de procesos donde se requiere reconocimiento de objetos.. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos ([reconocimiento facial](#)), calibración de cámaras, visión estérea y visión robótica.

Google Visión

<https://github.com/googlesamples/android-vision>

The Face API finds human faces in photos, videos, or live streams. It also finds and tracks positions of facial landmarks such as the eyes, nose, and mouth.

Tracking.js

<https://trackingjs.com>

The tracking.js library brings different computer vision algorithms and techniques into the browser environment. By using modern HTML5 specifications, we enable you to do real-time color tracking, face detection and much more — all that with a lightweight core (~7 KB) and intuitive interface.

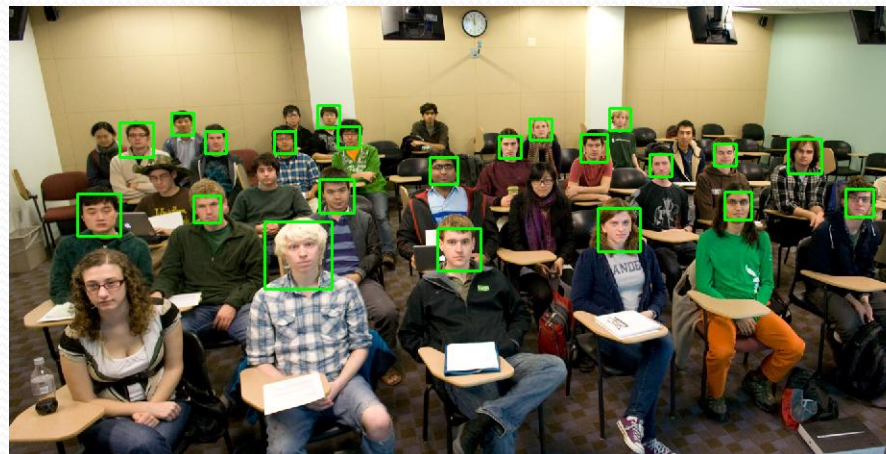
Diferencia entre Detección y Reconocimiento



Face Detection es que la librería es capaz de seguir caras en tiempo real y detectar si se sonríe, si se está triste. Este tipo de algoritmos pueden distinguir otros objetos en una fotografía y seguirlos con la cámara (tracking)



Si seguimos a mas de un objeto cara , lo denominamos multitasking



Face Recognition

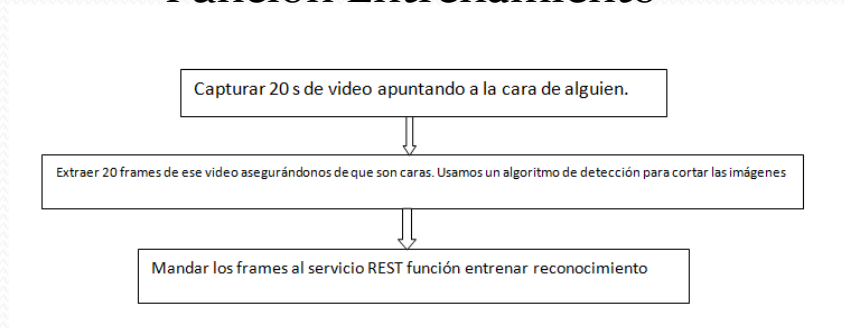
Face Recognition es cuando el algoritmo es capaz de distinguir una cara de otra. Para esto en todas las librerías es preciso entrenar al sistemas con fotografías de la persona a reconocer.

En el caso de OpenCV se recomiendan unas 20 fotografías todas del mismo tamaño y resolución.

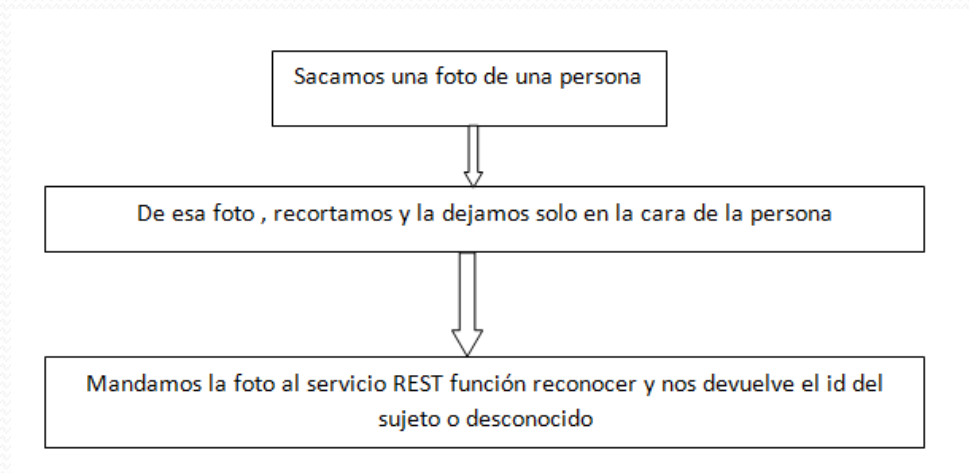


Mi librería no implementada

Función Entrenamiento



Función Reconocimiento





Tooling

Para este proyecto se han usado y están usando.

- Android Studio 2.1.2
- Android Studio NDK r12
- Eclipse Mars 2.
- Netbeans 8.1
- Chrome Version 51.X
- Jdk 1.8
- OpenCV Versiones 2.4 /3.10
- Apache Cordova 6

- com.google.android.gms.vision de google-play-services.
- JavaCV <https://github.com/bytedeco/javacv> / <https://code.google.com/archive/p/javacv/>