

Práctica 3: TUS Santander

Alejandro Pérez Ruiz
Grupo de Ingeniería Software y Tiempo Real
perezruiza@unican.es

Objetivo (I)

- Crear una **aplicación Android** que nos proporcione información sobre las líneas de autobús disponibles en Santander.
- No vamos a partir de cero: tenemos una **aplicación creada con** gran parte de la funcionalidad implementada que contiene **dos errores importantes** que debemos corregir, además de añadir algunas mejoras.


Objetivo (II)

- Aspecto (básico) que debería tener la aplicación:



¿De dónde obtenemos los datos?


- El ayuntamiento de Santander tiene un servicio donde nos proporcionan diversos datos relativos a transportes, agenda cultural, comercios, tráfico...
 - <http://datos.santander.es/data/>
 - Para esta práctica nos interesan los relativos al servicio municipal de autobuses:
<http://datos.santander.es/dataset/?id=lineas-bus>
 - El servidor nos puede devolver los datos en diferentes formatos:



Líneas Bus
Este recurso contiene la información referente a las líneas de Transporte Urbano en uso dentro del Municipio de Santander.

[RDF](#) [HTML](#) [JSON](#) [N3](#) [XML](#) [TURTLE](#) [CSV](#) [ATOM](#) [JSONLD](#)

👍 15 🗨️ 15
Visitas: 2566



Paradas por línea
Este recurso contiene información de las Paradas programadas dentro de cada Línea.

[RDF](#) [HTML](#) [JSON](#) [N3](#) [XML](#) [TURTLE](#) [CSV](#) [ATOM](#) [JSONLD](#)

👍 14 🗨️ 21
Visitas: 2099

¿Cómo se obtienen los datos? (I)

- Al dataset se accede gracias a un “**servicio REST**”.
 - REST es un estilo de arquitectura para desarrollar servicios web que presenta las siguientes características:
 - Cliente/Servidor: roles bien definidos y separados.
 - Sin estado: no hay estado asociado al cliente.
 - Servicio uniforme: todos los servicios comparten una forma de invocación uniforme utilizando los métodos GET, POST, PUT y DELETE.
- En este servicio como queremos consultar datos únicamente necesitaremos el método GET del servicio REST.

¿Cómo se obtienen los datos (II)?

- Los datos de los autobuses los obtendremos en formato **JSON**
- El servicio tiene muy poca documentación de cómo se utiliza y que significan cada uno de los datos que nos devuelven:



LINEAS BUS

Contiene la información referente a las líneas de Transporte Urbano en uso dentro del Municipio de Santander.

Condiciones de uso: [Atribución 4.0 Internacional \(CC BY 4.0\)](#)

Publicador: Ayuntamiento de Santander

Fecha de creación: 04/07/2013 10:25:22

Frecuencia de actualización: Diaria

Obteniendo los datos con el navegador

- Los datos se encuentran identificados con una URL
 - URL con las líneas de autobuses para Santander:
http://datos.santander.es/api/rest/datasets/lineas_bus
 - Si utilizamos esa URL para acceder con un navegador web veremos los datos de una manera poco “amigable” (depende del navegador que se utilice).
- Existen algunas aplicaciones para Chrome como “Advanced Rest API Client” que nos permiten ver correctamente los datos.

Obteniendo los datos con Android

```
public void getJSON(String urlJSON) throws IOException{  
    URL url = new URL(urlJSON);  
    HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();  
    urlConnection.setRequestProperty("Accept", "application/json");  
    bufferedDataGasolinas = new BufferedInputStream(urlConnection.getInputStream());  
}
```


JSON (I)

- JSON “JavaScript Object Notation”: formato ligero de intercambio de datos.
- Su objetivo es que sea fácil leerlo y escribirlo para los humanos y que a la vez sea fácil para que las máquinas lo interpreten y generen.
- Ejemplo sencillo de modelado de datos con JSON:
 - Tenemos una tienda de videojuegos con los siguientes productos:
 - Juegos: GTA V, FIFA 17 y NBA 2K17.
 - Consolas: PS4, XBOX ONE y Wii U.
 - Un objeto JSON se identifica entre llaves:

```
{  "NombreJuego": "GTA V",  "Cantidad": 20}
```

Nombre Valor Nombre Valor

JSON (II)

- Un array en JSON se crea con []:

```
{ "Juegos": [  
  { "NombreJuego": "GTA V", "Cantidad": 20 },  
  { "NombreJuego": "FIFA 16", "Cantidad": 13 },  
  { "NombreJuego": "NBA 2K16", "Cantidad": 16 } ]  
}
```

- Por lo tanto nuestra tienda de videojuegos queda del siguiente modo:

```
{ "Tienda":  
  [  
    { "Juegos":  
      [ { "Nombre": "GTA V", "Cantidad": 20 },  
        { "Nombre": "FIFA 16", "Cantidad": 13 },  
        { "Nombre": "NBA 2K16", "Cantidad": 16 } ]  
    },  
    { "Consolas":  
      [ { "Nombre": "PS4", "Cantidad": 2 },  
        { "Nombre": "XBOX ONE", "Cantidad": 4 },  
        { "Nombre": "Wii U", "Cantidad": 1 } ]  
    }  
  ]  
}
```

¿Cómo se “parsea” un JSON? (I)

JSON con las líneas de autobuses:

Objeto → {

```
  "summary": {"items": 33, "items_per_page": 1000, "pages": 1, "current_page": 1},
  "resources": [ ← Array
    {
      "ayto:numero": "E30",
      "dc:name": "SE ESTACIONES-FERIAS",
      "dc:modified": "2017-10-12T23:00:00.112Z",
      "dc:identifier": "30",
      "uri": "http://datos.santander.es/api/datos/lineas_bus/30.json",
      ...
    },
    {
      "ayto:numero": "E31",
      "dc:name": "MANUEL LLANO-SARDINERO",
      ...
    },
    ...
  ]
}
```

Objeto → {

Objeto → {

¿Cómo se “parsea” un JSON? (II)

```
{
  "summary": {"items": 33...},
  "resources": [
    {
      "ayto:numero": "E30",
      "dc:name": "SE ESTACIONES-FERIAS",
      "dc:modified": "2017-10-12T23:00:00.112Z",
      ...
    },
    {
      "ayto:numero": "E31",
      "dc:name": "MANUEL LLANO-SARDINERO",
      ...
    }
  ]
}
```

- Android proporciona clases para “parsear” un JSON
 - `JsonReader`

```
JsonReader reader = new JsonReader(new InputStreamReader(in, "UTF-8"));
List<Linea> listLineasBus = new ArrayList<Linea>();

reader.beginObject();
while(reader.hasNext()){
    String name = reader洗洗Name();
    if(name.equals("resources")){
        reader.beginArray();
        while (reader.hasNext()){
            listLineasBus.add(readLinea(reader));
        }//while
    }else{
        reader.skipValue();
    } //if
} //while
```

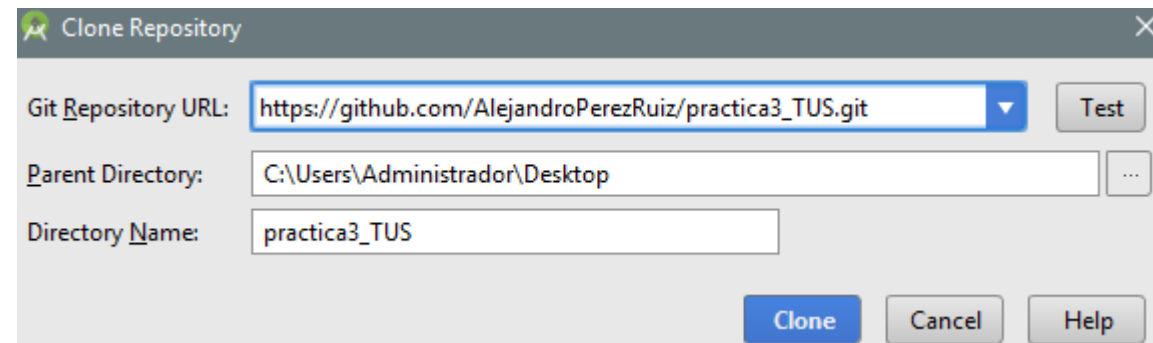
¿Cómo se “parsea” un JSON? (III)

```
{
  "summary":{"items":33...},
  "resources": [
    {
      "ayto:numero":"E30",
      "dc:name":"SE ESTACIONES-FERIAS",
      "dc:modified":"2017-10-12T23:00:00.112Z",
      ...},
    ...
    {
      "ayto:numero":"E31",
      "dc:name":"MANUEL LLANO-SARDINERO",
      ...
    }
  ]
}
```

```
public static Linea readLinea (JsonReader reader) throws IOException {
    reader.beginObject(); //Leemos un object
    String name = "", numero="";
    int identifier=-1;
    while(reader.hasNext()) {
        String n = reader洗洗Name();
        if (n.equals("ayto:numero")) {
            numero = reader.readString();
        } else if (n.equals("dc:name")) {
            name = reader.readString();
        } else if (n.equals("dc:identifier")) {
            identifier = reader.nextInt();
        } else {
            reader.skipValue();
        }
    }
    reader.endObject();
    return new Linea(name,numero,identifier);
}
```

Aplicación proporcionada - Descarga

- En un repositorio de GitHub tenéis la aplicación sobre la que debéis realizar la práctica:
https://github.com/AlejandroPerezRuiz/practica3_TUS.git
- Debéis clonar este repositorio en vuestra máquina para empezar a trabajar; para ello:
 - Tenéis que tener instalado git en vuestro sistema (<https://git-scm.com/downloads>)
 - Desde Android Studio tenéis que ir a “File” → “New” → “Project from Version Control” → “Git”



Aplicación proporcionada – Errores (I)

- Imaginad que el código de la aplicación anterior lo ha desarrollado alguien que lo dejó sin terminar:
 - Se quedó con **dos errores** importantes que hacen que no se visualice ningún dato.
 - La muestra de los resultados quedó en un estado muy “primitivo”. Con los dos errores anteriores resueltos la app se mostrará del siguiente modo:



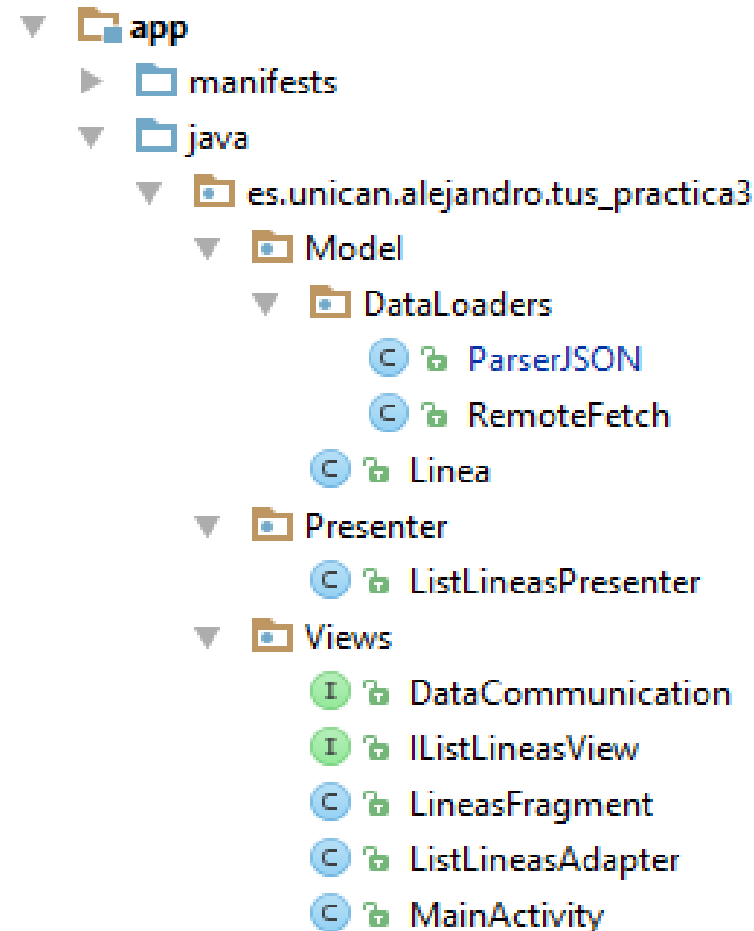
Aplicación proporcionada – Errores (II)

Después de corregir los dos errores más críticos falta información por mostrar en la app (nombre de la línea):

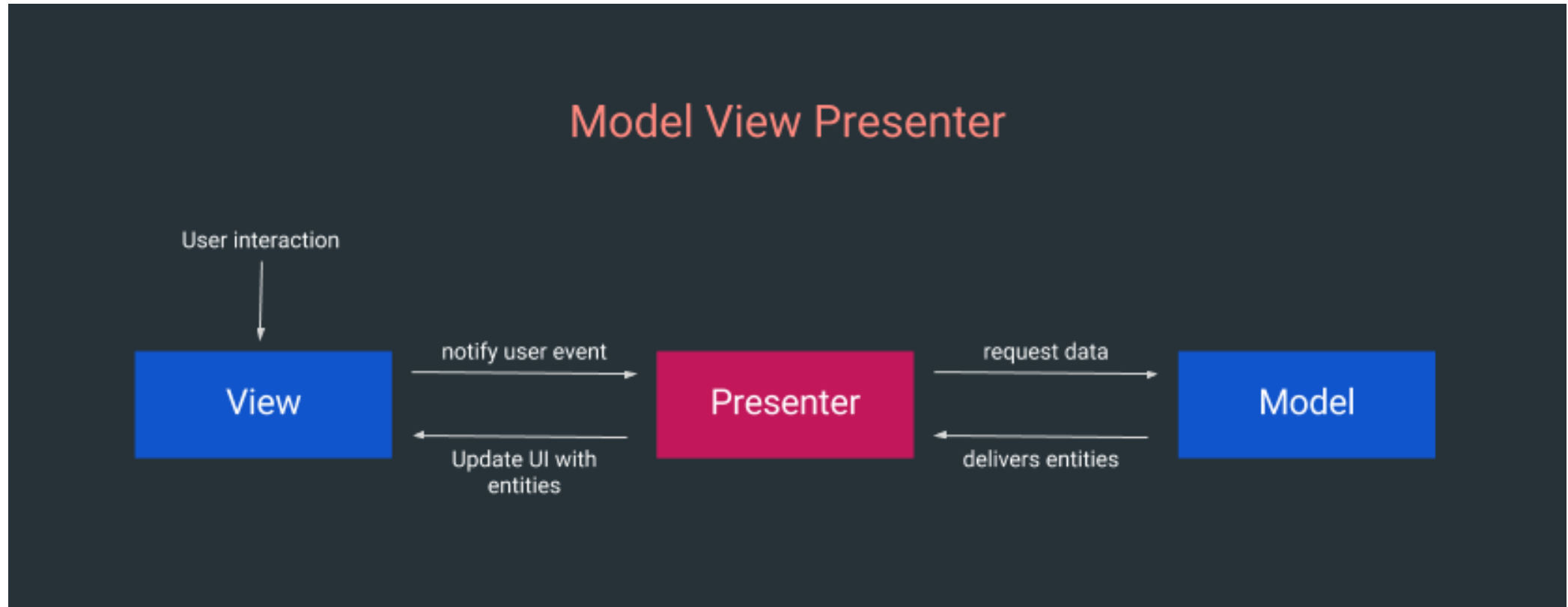
- Esto es provocado por un pequeño error en el método que “parsea” el JSON.
- Habiendo corregido este error obtendremos toda la información correctamente:



Aplicación proporcionada - Estructura



Aplicación proporcionada - MVP



Tareas de la práctica (I)

1. Identificar y corregir los dos errores críticos que hacen que no se visualicen los datos.
 - En el momento que hayáis corregido los errores debéis subir un breve informe (en PDF) al Moodle de la asignatura donde indiquéis en qué consistían los errores y cómo los habéis solucionado.
 - Esta primera tarea será recompensada con hasta 0.15 puntos extras en la nota final de la asignatura.
 - El primero que identifique los errores y los corrija: 0.15 puntos extras.
 - El segundo: 0.14 puntos extras.
 - El tercero: 0.13 puntos extras.
 - El cuarto: 0.12 puntos extras.
 - El quinto: 0.11 puntos extras.
 - El sexto: 0.10 puntos extras.

Tareas de la práctica (II)

2. Para detectar el error que hay en el “parseo” del JSON se debe crear un test que utilice un JSON local y compruebe si el método **readArrayLineasBus(InputStream in)** devuelve una lista con la información esperada de las líneas.
 - Una vez detectado el error corregirlo en el código del método **readArrayLineasBus**
3. Se deben visualizar los datos de un modo más amigable:
 - Mientras se descargan y procesan los datos se debe mostrar un ProgressDialog



- Cuando se finalice la carga de datos mostrar un Toast con el siguiente texto si se ha producido exitosamente: “Datos obtenidos con éxito”.
- Siguiendo el patrón de colores de las líneas de autobuses de Santander de la siguiente web: <http://www.tusantander.es/red-lineas> mostrar el número/identificador de la línea con su color correspondiente.