ACADEMIK

Próximos cursos

Junio 2020

Martes 2

Introducción al desarollo web con CSS, HTML 5 Y JavaScript

Aprende los fundamentos de creación de sitios web

- **Duración: 36 horas**
- **Clases: Martes y Jueves - 19:00 - 21:00**

Martes 2

Aplicaciones empresariales con Angular

El framework más popular en entornos empresariales

- **Duración: 36 horas**
- **Clases: Martes y Jueves - 19:00 - 21:00**

Martes 2

Introducción a la programación con Java 11

El lenguaje más popular desde 1996

- **Duración: 36 horas**
- **Clases: Sábados - 14:00 - 18:00**

Martes 2

Aplicaciones web con Java

Aprende los fundamentos de la programación backend

- **Duración: 36 horas**
- **Clases: Sábados - 14:00 - 18:00**

Programación de aplicaciones Android Java con Backends REST

ING. MAX ALEJANDRO ANTONIO CERNA FLORES

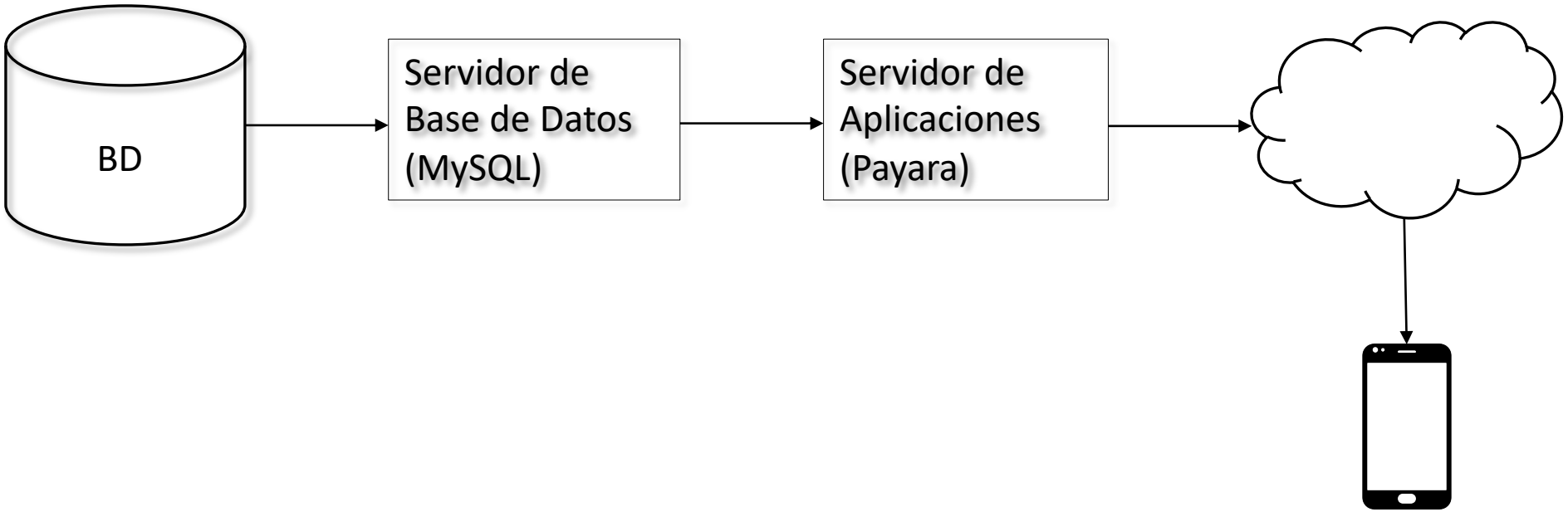
DATABASE ADMINISTRATOR Y JAVA DEVELOPER

TWITTER: @MAXEIN

Ingredientes

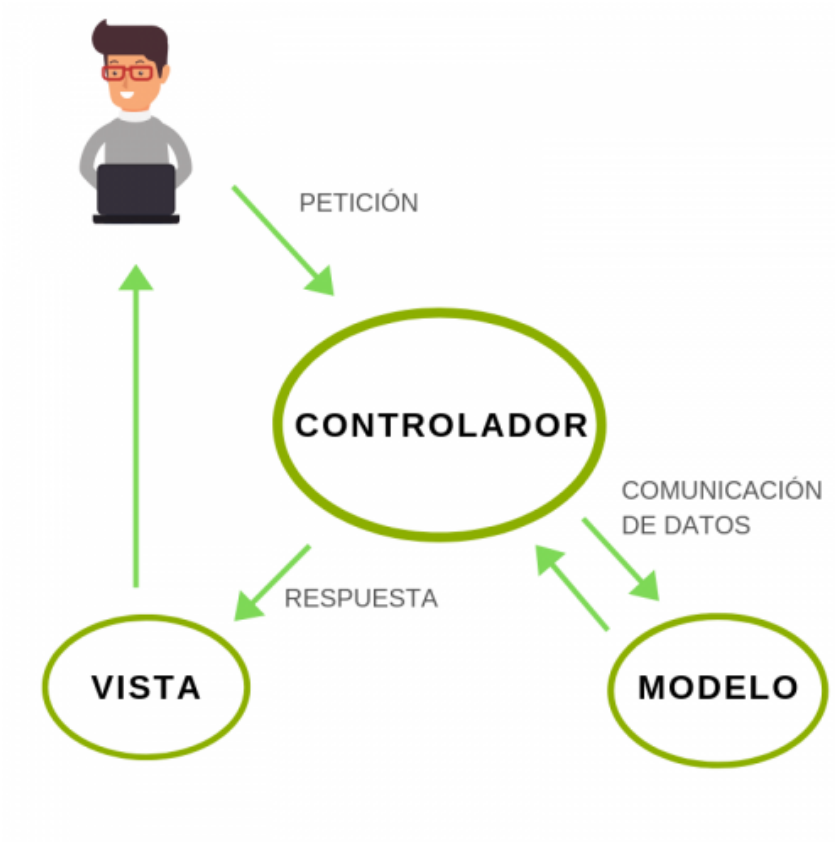
- IDE Java (Netbeans)
- Base de datos relacional (MySQL)
- Driver JDBC (JAVA-MySQL)
- IDE Android (Android Studio)
- ORM(Hibernate)
- WebServer (Payara)

Arquitectura Hardware



Arquitectura de Software

Patrón MVC



Descargar las herramientas

MySQL - https://dev.mysql.com/downloads/installer/

Apache Netbeans - https://netbeans.apache.org/download/

Android Studio - developer.android.com/AndroidStudio/Download

Payara - https://www.payara.fish/software/downloads

JDBC Driver - https://dev.mysql.com/downloads/connector/j

Java - https://www.oracle.com/java/technologies/java-ee-sdk-download.html

Creando la base de datos

```
create database phonedb;

use phonedb;

create table persona (nombre varchar(100), edad int,telefono varchar(15));

alter table persona add constraint persona_pk primary key (nombre);

insert into persona values("Max",34,"55555555");

insert into persona values("Alejandro",34,"11111111");

Commit;
```

Creando los WebServices

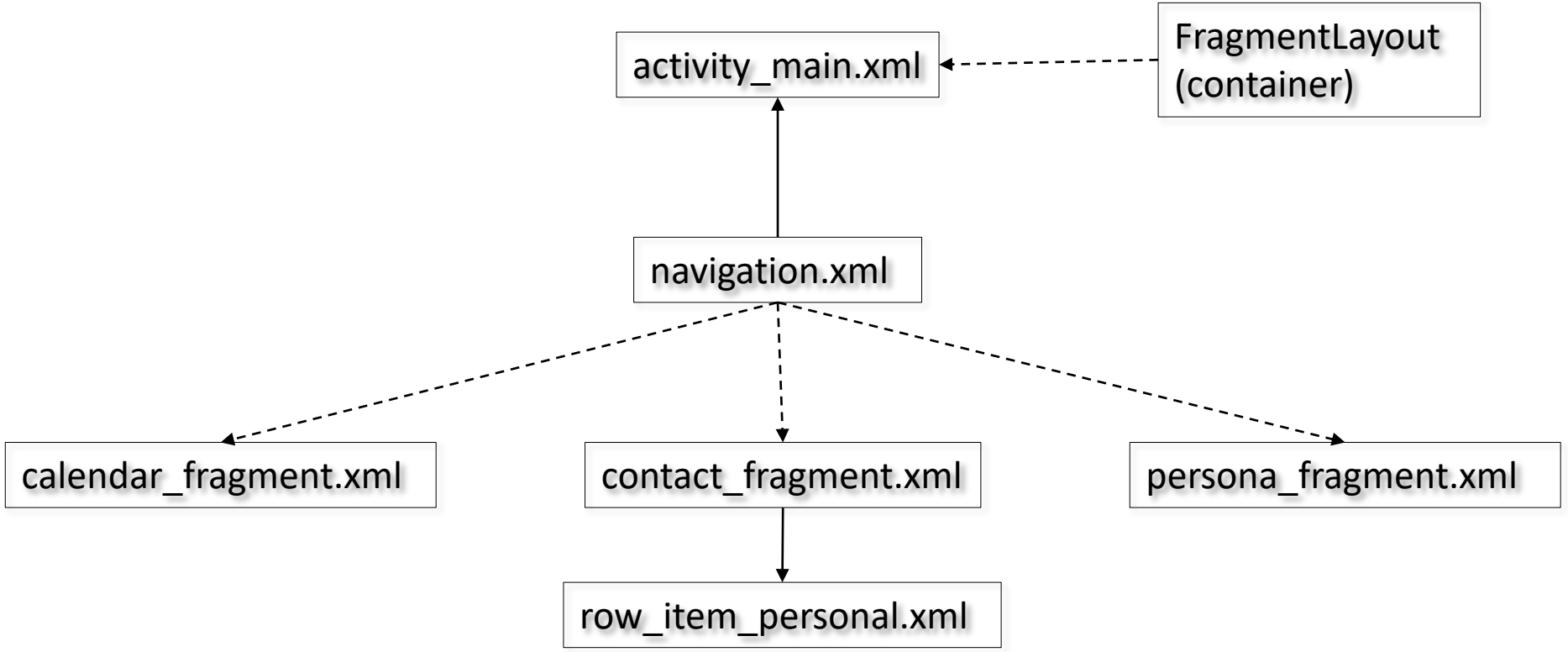
1. Crear un Maven Web Application Project en Netbeans
   - Indicar nombre del proyecto y path
   - Agregar un nuevo WebServer Application Payara
2. Crear packages en el proyecto para los Entities y WebServices
3. Crear los Entities a partir una conexión de base de datos a MySQL.
   - Definir nuestro DataSource agregando nuestro JDBC driver para MySQL
   - Mapear las tablas que necesitemos.
4. Crear WebServices RESTful usando los entities creados como guias.
   - Modificar el WebServices para entregar respuesta en formato Json.

Mientras tanto… en Android Studio

1. Crear un proyecto con la plantilla “Empty Activity”
   - Definimos el package del proyecto
   - Lenguaje a utilizar en este caso Java.
   - El API Android minimo (por defecto es la versión 15)

2. En AndroidManifest.xml damos los permisos correspondientes
   - <uses-permission android:name="android.permission.INTERNET" />
   - En caso dada la versión Maxima del API Android sea 28 o superior también debemos agregar:
   - <base-config cleartextTrafficPermitted="true" />

Estructura XMLs



Definiendo Activities y Fragments (XML)

Main Activity (activity_main.xml)

1. Agregar un FrameLayout que soporte Scrollbar para poder listar contactos.

app:layout_behavior="@string/appbar_scrolling_view_behavior“

2. Agregar un BottomNavigationView para un menú practico en la parte inferior de nuestro proyecto, los componentes del menú estarán definidos en un archivo xml a parte al que llamaremos navigation.xml

app:menu="@menu/navigation"

3. Para los iconos de la aplicación agregamos los archivos png o jpg en el path: res/drawable para ser utilizados en el archivo navigation.xml

Definiendo Activities y Fragments (XML)

navigation.xml

1. En este archivo se definen los componentes del menú BottomNavigationView, por ejemplo:

```
<?xml version="1.0" encoding="utf-8"?>
  <menu xmlns:android=http://schemas.android.com/apk/res/Android...>
  <item
        android:id="@+id/navigation1"
        android:icon="@drawable/ic_persona"
        android:title="Persona"/>
  </menu>
```

Con la opción “Android:icon” podemos especificar el icono que queremos utilizar que previamente colocamos en la carpeta de imágenes.

Definiendo Activities y Fragments (XML)

contact_fragment.xml (fragmento una opción del menú en activity_main.xml)

Se crea un ***RecyclerView*** el cual nos servirá de componente base para listar los contactos recuperados al consumir el WebServices que creamos previamente.

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/lista_persona“…/>
```

La estructura de cada fila que compone el RecyclerView se define en otro xml al cual se le llamara como ***row_item_personal.xml*** para este caso.

Definiendo Activities y Fragments (XML)

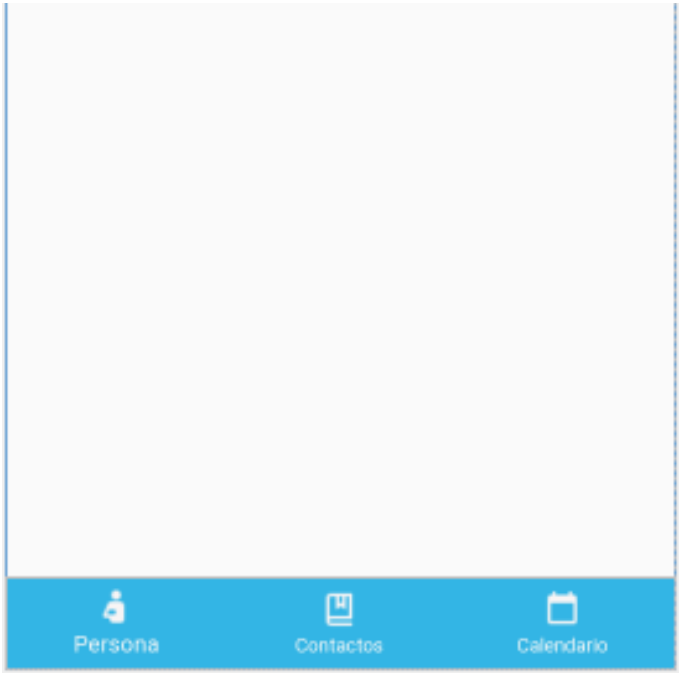
row_item_personal.xml

Definimos los componentes de cada row en un LinearLayout inicial.

La estructura interna puede cambiar según lo que se desea mostrar. Para este ejemplo únicamente se agregaran un TextView para el nombre del contacto y un Button que podría utilizarse para realizar la llamada.

```
<LinearLayout
   xmlns:android=http://schemas.android.com/apk/res/android…>
  <TextView
       android:id="@+id/persona_nombre“../>
  <Button
       android:id="@+id/persona_telefono“../>
</LinearLayout>
```

Vista Previa



Pojo Persona(.java)

Persona.java

Clase simple java que tiene como atributo los mismos atributos que tiene el Entity creado en el proyecto de Web Service pero sin las anotaciones especiales.

```
public class Persona {

private String nombre;
private Integer edad;
private String telefono;
....
}
```

Definiendo Activities y Fragments (.java)

MainActivity.java

Metodos a utilizar:

setupNavigationView(): metodo para enlazar BottomNavigationView e incluir el listener de seleccion con clic, invocara al metodo *selectFragment*.

selectFragment(MenuItem item): Metodo para seleccionar el Fragment segun el parametro enviado.

pushFragment(Fragment fragment): Recibe el fragmento seleccionado, si el fragmento no es null se obtiene el FragmentManager.

Definiendo Activities y Fragments (.java)

ContactFragment.java

Se define un objeto “Adapter” que será el que manejara los datos que se colocaran en el RecyclerView, este objeto es una clase especial que se describirá posteriormente.

Se define una clase interna llamada ***ObtenerDataWS*** que será nuestro hilo asíncrono para consumir el WebService. Hereda de *AsyncTask*

Predeterminado esta clase debe utilizar un método ***doInBackground*** que ejecutara la tarea del hilo tras bambalinas en este caso conectarse al Web Service y analizar el resultado Json.

urlConnection(String link): Método que hará la conexión al web Service, invocado desde doInBackground.

readJsonPersona(JsonReader jsonReader): Metodo que recorrerá el resultado Json del web services y lo acoplara en un objeto del tipo Persona.

onPostExecute(String result): Metodo especial de AsynTask que permite que una vez completada la actividad doInBackground haga una tarea, en este caso es poder asignar el Adapter que se creo y relleno durante el doInBackground.

Definiendo Activities y Fragments (.java)

ContactAdapter.java

Es la clase que hace la magia de enlazar los datos previamente capturados con los objetos definidos en los xml para mostrar los resultados, es un adaptador para el RecyclerView.

Internamente hay una clase ContactViewHolder la cual definirá los componentes TextView y Button que se enlazaran con los creamos en el row_item_personal.xml.

ContactAdapter(List<Persona> contactos, Context context): Constructor que recibirá la activity origen y la lista de contactos que se obtuvieron del web services.

onBindViewHolder(ContactViewHolder holder, final int position): Metodo que recorrerá cada posición de la lista y lo agregara al recyclerView.

Muchas gracias por su atención

Preguntas?