

**Ingeniería en Sistemas**

**Computacionales**

**Raúl Tostado Blázquez I.D. 145901**

**Materia: Sistemas Distribuidos**

**Nombre del Trabajo: Práctica Mashup**

**Profesora: Ofelia D. Cervantes Villagómez**

**San Andrés Cholula, Puebla a**

**22 de abril de 2015**

Reporte práctica Mashup

**Introducción a la práctica**

Un mashup es una forma de integración y reutilización. Ocurre cuando una aplicación web o servicio es usado o llamado desde otra aplicación con el fin de reutilizar su contenido y/o funcionalidad. Utilizando diversas fuentes, es posible para crear nuevos servicios, visualizado en una única interfaz gráfica. Por ejemplo, se pueden combinar las direcciones, fotografías y/o tweets con un mapa de Google Maps, fotografías de Flickr, y la API de Twitter para crear un mashup de un mapa interactivo. **Antes de comenzar:**

El primer paso fue dar una lectura detallada a la práctica para saber cuál es el contenido de la misma, posteriormente se descargaron como Zip los programas con los que se trabajó y otros complementos necesarios para trabajar.

**Actividad 1 – Llamando Servicios Web**

El objetivo de la práctica es implementar los servicios Web previamente desarrollados en la práctica[**Play-WebService**](https://github.com/Innova4DLab/Play-WebService) en una vista implementada en **Play**, utilizando **JavaScript**.

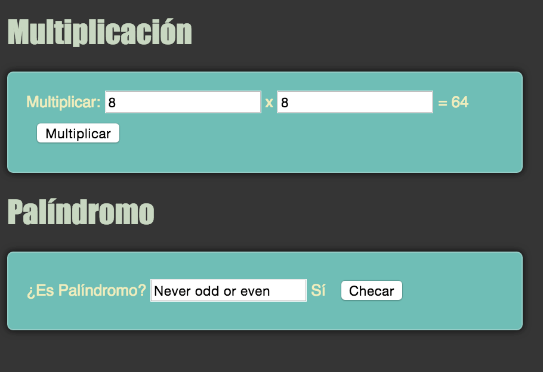
Existen diversas formas de llamar servicios Web, una de ellas es utilizando llamadas asíncronas al servidor utilizando **Ajax** *(Asynchronous Javascript and XML)*, éstas llamadas son muy fáciles de implementar utilizando la librería **jQuery**.

* Para comenzar, **Iniciar** el servidor de **Play Framework** en la terminal, navegando a la carpeta donde se encuentra el proyecto de la práctica [**Play-WebService**](https://github.com/Innova4DLab/Play-WebService) y escribir:

$ cd play-java-rest

$ activator run

* Como en la práctica anterior, **Abrir** el **proyecto de Play** en el editor de su preferencia.
  + Como ya habiamos visto previamente, **Play Framework** utiliza estrictamente el patrón**Modelo-Vista-Controlador**, por lo que la parte de la vista, donde se desarrolla el despliegue de la interfaz, se lleva a cabo dentro de la carpeta **app/views**.
  + Para nuestra aplicación, las llamadas a los **servicios Web** las implementaremos del lado del**cliente**, utilizando llamadas asíncronas, por lo que nuestras actividades se centraran bajo**HTML, CSS y JavaScript**.
* **Copiar** el contenido del archivo en *este repositorio* **main.scala(1).html** archivo del proyecto de**Play** dentro de la ruta **app/views/main.scala.html**.
* **Abrir** el navegador y escribir **localhost:9000** en la barra de direcciones, debería aparecer algo así:



**Explicación:**

* **jQuery** es una librería **Javascript** que nos ayuda a simplificar mucho trabajo. para importarla es recomendable utilizar el servicio **CDN** *(content distribution network)* de **Google Hosted Libraries**, ejemplo:

<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>

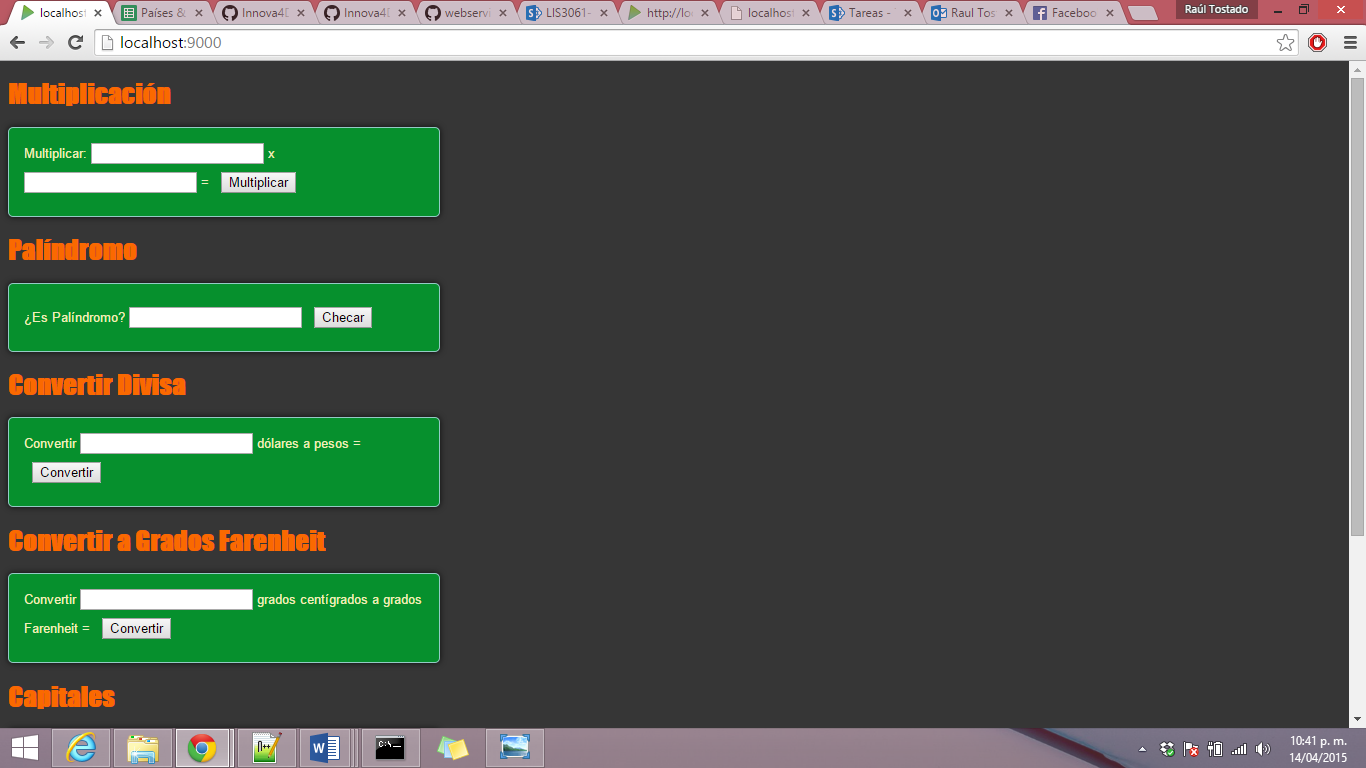
Nuestra aplicación se dive en 3 partes, **(1) JavaScript**, **(2) CSS**, **(3) HTML**.

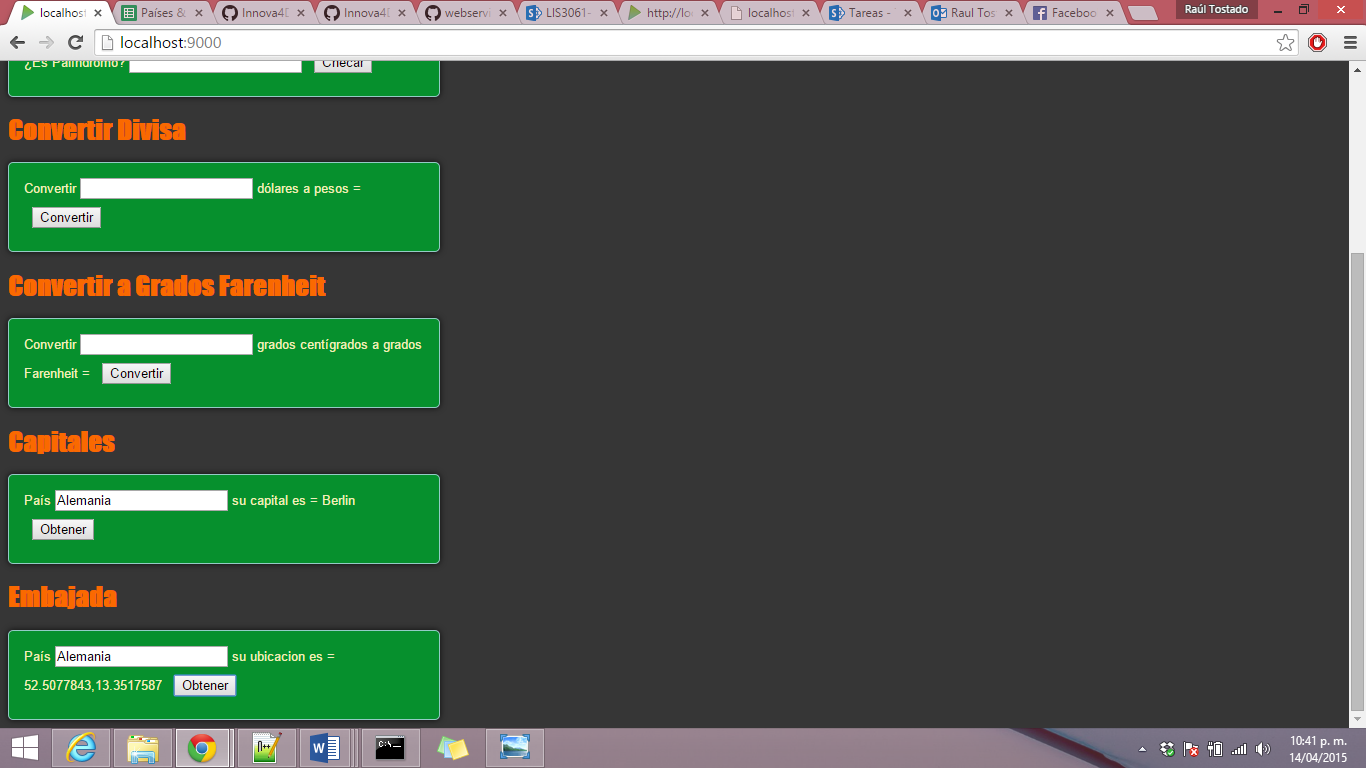
* En la sección de **Javascript** se definen métodos que escuchan eventos y ejecutan llamadas asíncronas para desplegar los resultados en el **DOM**. Estas funciones se implementan de manera simple gracias a **jQuery**.
  + Analizar el código **JavaScript**.
  + Presta atención a la **documentación**.
  + **Analizar** el método [Click](https://api.jquery.com/click/) de jQuery.
  + **Analizar** el método [Ajax](http://api.jquery.com/jquery.ajax/) de jQuery.
  + **Analizar** el método [Append](https://api.jquery.com/append/) de jQuery.
  + **Explicar** en el reporte las funcionalidades de estos métodos y su uso en la aplicación.
* En la sección **< body >** del código **HTML** se definen algunos elementos del **DOM**, aquí es importante tener en cuenta los atributos **id** y **class** puesto que éstos son referenciados en el código **JavaScript** y **CSS**.
  + **Analizar** la relación entre el código **HTML** descrito dentro del tag **body** y el código **CSS** y**Javascript**.
  + **Explicar** en el reporte el uso de **id** y **class** dentro de los elementos **HTML**.
* La sección **CSS** está definida dentro del tag **< style >** en el documento **HTML**, aquí se define el[*look & feel*](http://en.wikipedia.org/wiki/Look_and_feel) de la aplicación.
  + **Analizar** el código CSS.
  + **Modificar** el código CSS, añadiendo tu propio estilo a la aplicación.
    - Una buena referencia es [W3Schools](http://www.w3schools.com/css/default.asp).
    - [CSS3 Cheat Sheet](http://www.smashingmagazine.com/wp-content/uploads/images/css3-cheat-sheet/css3-cheat-sheet.pdf)
  + **Explicar** en el reporte los atributos **CSS** utilizados en la aplicación.

**Actividad:**

En el documento **main.scala(1).html** de este repositorio se han implementado dos de los **servicios Web** previamente elaborados en la práctica [**Play-WebService**](https://github.com/Innova4DLab/Play-WebService), (**Multiplicación y Palíndromo**), a partir de ésto deberás:

* **Desarrollar** en JavaScript las llamadas asíncronas a los servicios faltantes:
  + **Convertir** Pesos a Dólares.
  + **Convertir** grados Centigrados a Farenheit.
  + **Obtener** la capital dado un país.
  + **Obtener** la embajada mexicana dado un país.
* **Desarrollar** en HTML los elementos necesarios para input/ouput del servicio Web.
* **Desarrollar** en CSS un estilo para presentar de manera adecuada los servicios al usuario.
* **Explicar** ampliamente en el reporte el desarrollo de la actividad.





Esta primera parte de la práctica se desarrolló de manera sencilla, ya que solo fue necesario dar el id correspondiente a cada una de las acciones que iban a ser llamadas y declarar los id de tipo result, los cuales son el resultado de nuestras operaciones

# Actividad 2 - Desarrollando un Mashup

Ahora que ya saben implementar un servicio Web y algunos conceptos básicos de **HTML y CSS** en la actividad 2 vamos a implementar un **Mashup**. Un **Mashup** es un conjunto de servicios, ya sean propios o de terceros que se unen para crear una aplicación completamente diferentes, algunos servicios de terceros pueden son ofrecidos por **Flickr, Wikipedia, WeatherReport, Google Maps**, etc.

Vamos a generar una aplicación que utilice todos estos servicios. Nuestra aplicación será una**WebApp** que permitirá ofrecer información sobre un país a las personas de negocios. Para esto ofreceremos un pequeño parrafo introductorio con la información del país **(Wikipedia)**, un pequeño mapa con la ubicación de la embajada **(Google Maps)**, el clima en la capital del país **(OpenWeather)**, una serie de fotografías relacionadas al país **(Flickr)**. Para esta actividad vamos a desarrollar una nueva vista, para esto es deberás seguir las siguientes instrucciones:

* **Cambiar** el nombre del archivo **main.scala.html** a **main.scala(act1).html**
* **Crear** un nuevo archivo **main.scala.html**
  + (Si están en **Atom**, pueden dar click derecho y seleccionar New file, ó pueden duplicar el anterior **main.scala.html**),
* **Copiar** el contenido del archivo **main.scala(2).html** que se encuentra **dentro de este repositorio** a **main.scala.html** del proyecto de Play.

En esta actividad vamos a **separar asuntos**, es decir, vamos a utilizar **CSS** para manejar únicamente el estilo, **HTML** para crear los elementos y **JavaScript** para implementar la lógica de la aplicación. Para hacer más claro el uso de *[Separation of concerns](http://en.wikipedia.org/wiki/Separation_of_concerns)* vamos a utilizar de la carpeta **public** de Play. La carpeta **public** es un directorio en donde el contenido es accesible por cualquiera que tenga acceso al servidor, es un lugar ideal para agregar imágenes que deseemos mostrar en la aplicación, código JavaScript y hojas de estilo CSS, ahora deberás:

* **Copiar** el contenido de la hoja de estilo **main(2).css** que se encuentra en este repositorio y pegar el contenido dentro de **public/stylesheets/main.css**
* **Analizar** el código, la principal diferencia en estos nuevos servicios es la integración de **API** de terceros, en algunos casos, como en la **API** de **Wikipedia** por cuestiones de seguridad se utiliza**JSONP**. Es por esto que al utilizar una API de un tercero es importante conocer la documentación. La **API** mas extensa es la API de **Flickr**, a continuación se presentan enlaces a la documentación así como definiciones de algunos conceptos presentes en esta aplicación:
  + **API** es el conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas.
  + **JSON**, acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.
  + **JSONP** es una técnica de comunicación utilizada en los programas JavaScript para realizar llamadas asíncronas a dominios diferentes. JSONP es un método concebido para suplir la limitación de AJAX entre dominios, que únicamente permite realizar peticiones a páginas que se encuentran bajo el mismo dominio y puerto por razones de seguridad.
  + **AJAX**, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano.
  + **Access-Control-Allow-Origin** Es un mecanismo que por cuestiones de seguridad permite o niega que varios recursos en un sitio web puedan ser solicitados desde otro dominio fuera del domino en donde la llamada fue solicitada.

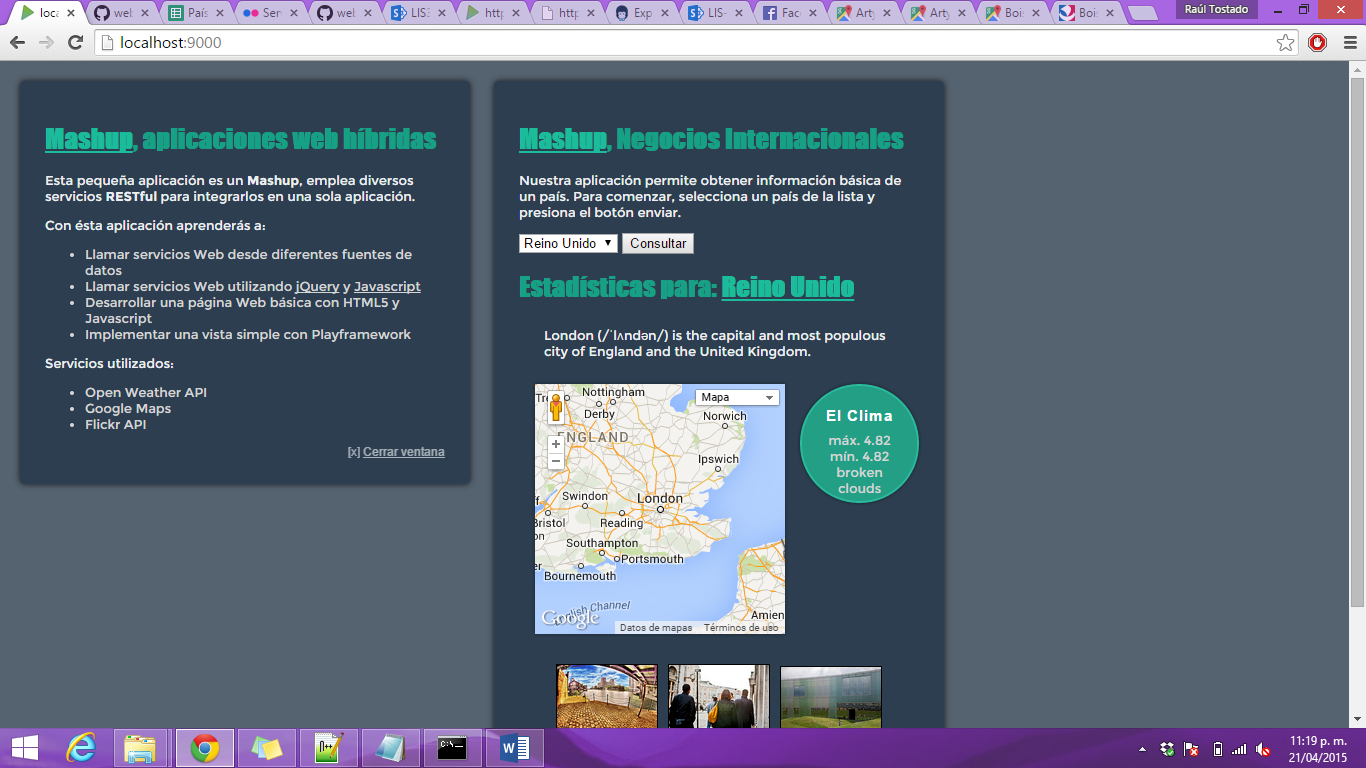
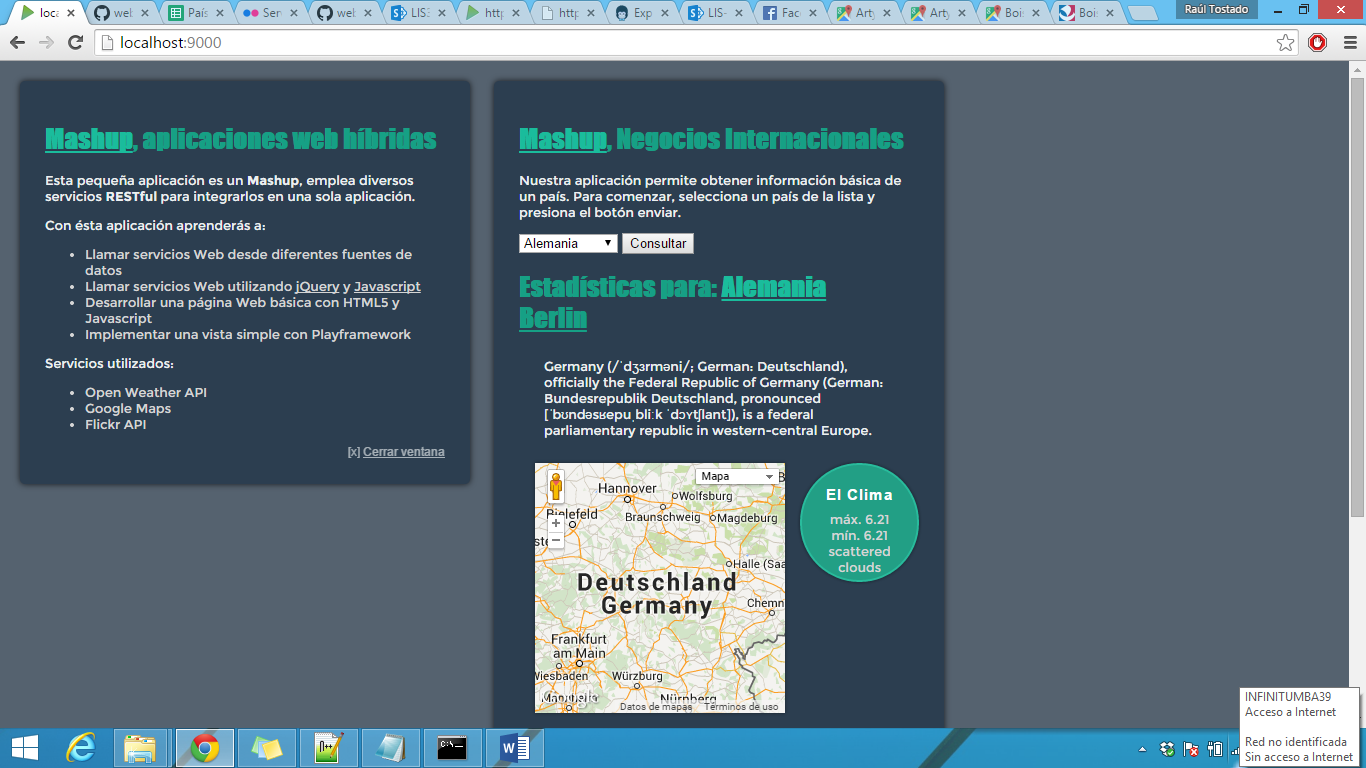
### Documentación:

* Documentación [Flickr](https://www.flickr.com/services/api/flickr.photos.search.html)
* Documentación [Wikipedia](https://www.mediawiki.org/wiki/API:Main_page)
* Documentación [OpenWeather](http://openweathermap.org/api)
* Documentación [GoogleMaps](https://developers.google.com/maps/documentation/javascript/)

### Actividad:

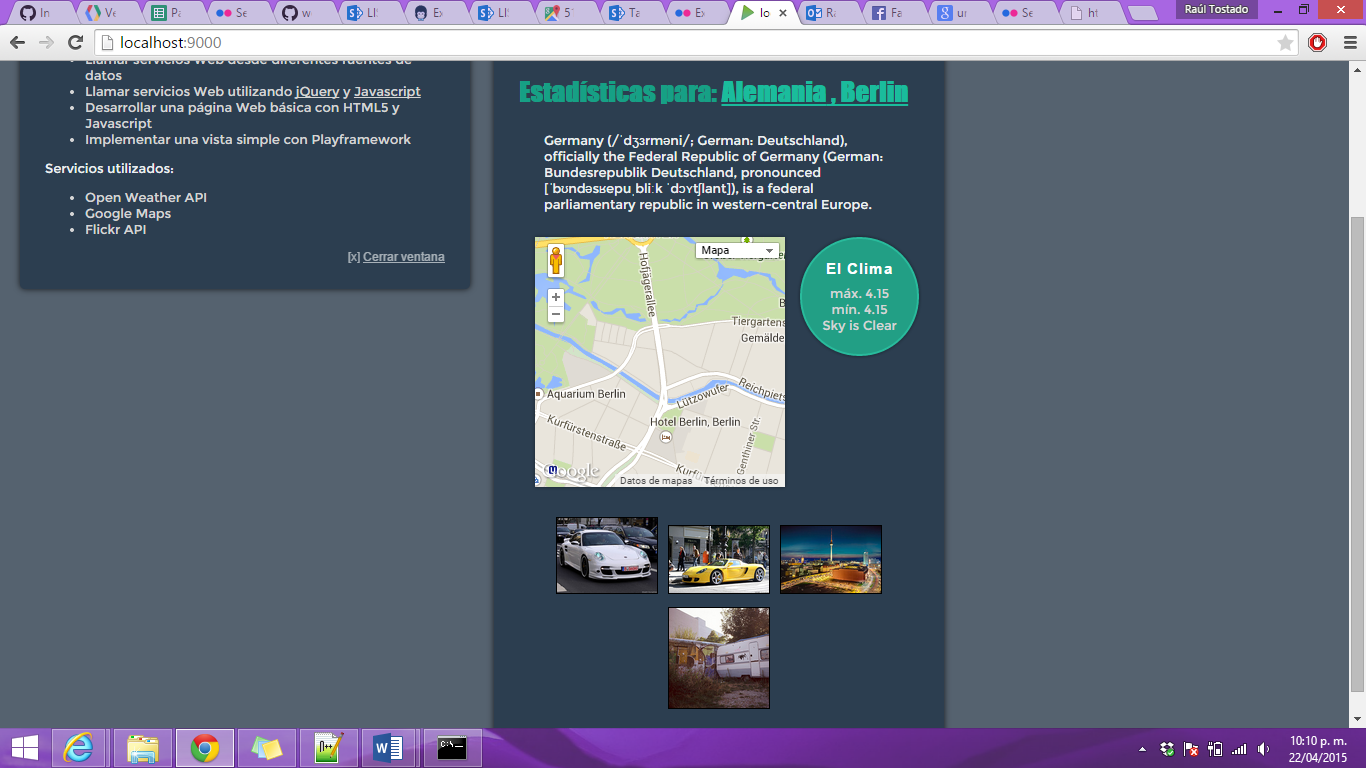
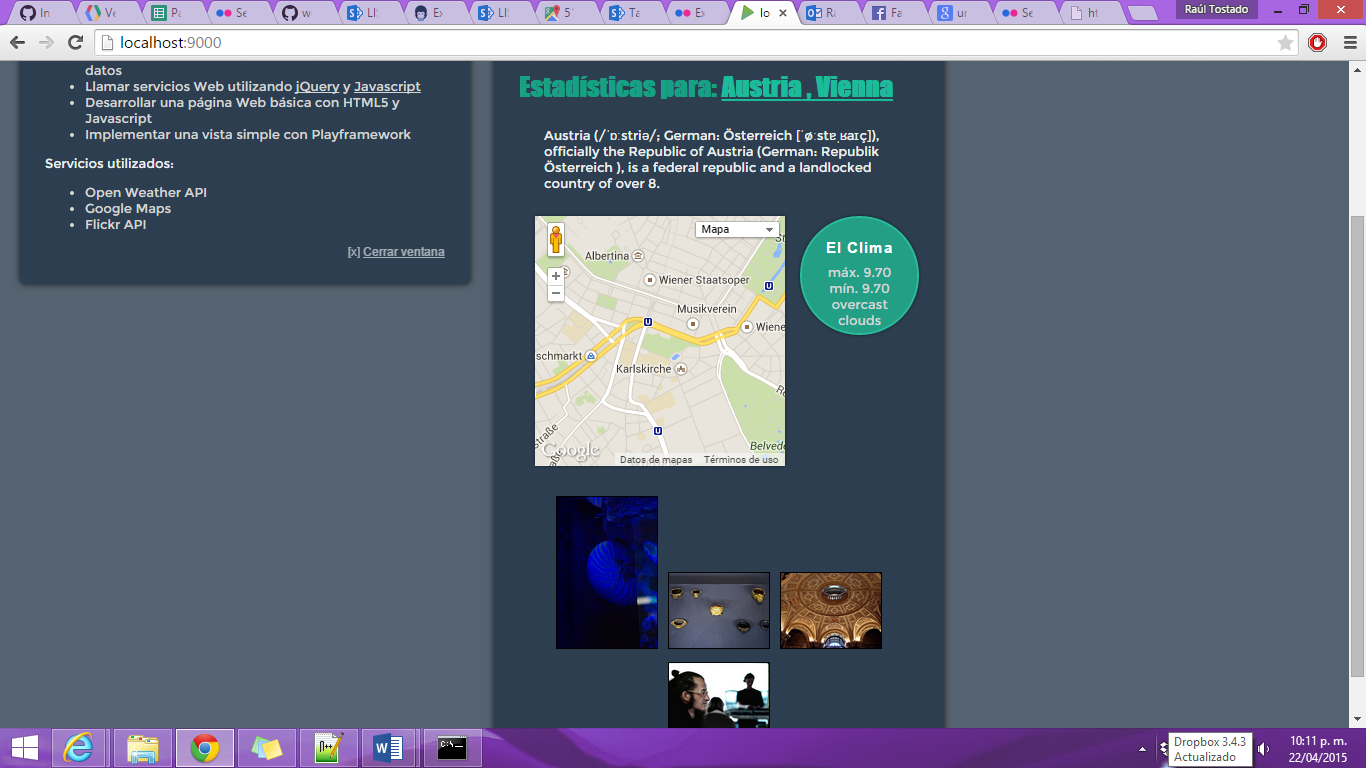
En el documento **main.scala(2).html** de este repositorio se encuentra un Mashup en donde se han implementado algunos **servicios Web** desarrollados por terceros, **Google**, **Flickr**, **OpenWeather**, etc. El objetivo de la práctica es integrar los servicios Web que ya desarrollaste en esta aplicación:

* Utilizando **los servicios Web** previamente desarrollados en la actividad anterior, la nueva aplicación deberá:
  + **Mostrar** la capital al lado del país.

* Mostrar en el mapa la ubicación de la embajada.
* Mostrar la temperatura en grados Celsius.
* Mostrar fotografías cercanas a la embajada mexicana para cada país.

Revisar la documentación de Flickr para implementar la API utilizando latitud y longitud (Geolocalización). Explicar ampliamente en el reporte los pasos seguidos para integrar los servicios.

En las imágenes de la parte superior, podemos ver los métodos solicitados anteriormente, nos da el país junto con su capital, la ubicación de la embajada (se aumentó el zoom para que fuera más claro de observar), se encuentra la temperatura en grados celcius y finalmente fotos que se encuentran cerca de la embajada.

Para implementar todos los servicios fue necesario revisar el API de los servicios solicitados, ya que con la misma podemos ver de manera clara como implementar los métodos necesarios, nos apoyamos de los servicios creados en la parte uno, en este caso se complementaron con los propuestos mediante el API de Wikipedia, Google y Flickr.

# Actividad 3 - Limpieza

En el desarrollo de aplicaciones Web es importante mantener un estilo al programar, por esto deberás:

* **Nombrar** adecuadamente las variables y métodos.
* **Agregar** un comentario antes de un método para indicar su funcionamiento.
* **Comentar** las líneas de código que podrían ser no muy claras.
* **Realizar** un identado adecuado del código.
* **Revisar** el código para detectar usos inadecuados de variables ó errores al escribir.

Recordando que el estilo debe ser siguiendo las guías de [Google](https://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml) para **JavaScript**.

R= En el código se encuentra la documentación necesaria para poder trabajar de manera precisa.

# Actividad 4 - Preguntas

Responder ampliamente a las siguientes preguntas en el reporte, tomando en cuenta todas las actividades realizadas en este repositorio.

* ¿Qué es un **Mashup**?

R= Es cuando en una aplicación o servicio web se llama a otra aplicación o servicio para ser reutilizado y no tener que crear algo que ya existe, por ejemplo si ya tenemos la herramienta de Google maps, es innecesario crear una nueva aplicación que haga lo mismo, ya que será tiempo perdido al igual que nunca será tan eficiente como la primera.

* ¿Cuál es la diferencia entre **Get** y **Post**?

R= Get envía variables dentro de la dirección URL y Post se envía información, pero no es visible para el usuario.

* Describe la utilidad de Ajax para llamar **servicios Web**.

R= En que si el cliente realiza cambios, esos cambios son visibles para él ya que son ejecutados en su navegador, si se quiere solicitar algo basta con escribirlo dentro de la página sin necesidad de recargar la página.

* Explica la utilidad del formato **Json**.

R= Json como tal es un formato ligero para el intercambio de datos; al no usar mucha memoria, es eficiente al momento de hacer llamadas y posteriormente consultas en caso de que se maneje una base de datos.

* + Explica la diferencia entre **Json** y **Jsonp**.

R= Como su nombre lo indica Json es un formato para el intercambio de datos, en cambio Jsonp es la técnica de comunicación utilizada en los programas JavaScript para realizar llamadas asíncronas a dominios diferentes. Ayuda a suplir las carencias de Ajax

* Explica la **separación de asuntos** en términos de desarrollo Web.

R= Es necesario, ya que al separarlos hay una mayor comprensión del problema, es decir, se entiende más fácil cuales son los elementos con los que se trabaja, ya sea si van a ser llamados los archivos o las hojas de estilo para la presentación.

* ¿Cómo se implementa el **patrón MVC** en **Play**?

R= Al momento de generar la carpeta, se implementa este patrón por defecto, es muy útil ya que permite la separación de las distintas partes de la aplicación.

* + ¿Qué se hace en el **controlador**?

R= Es donde se encuentra el núcleo de la aplicación, todos los métodos y acciones que se van a realizar, son declarados en este punto

* + ¿Cómo se implementa la **vista**?

R= Usamos el formato HTML para dar de alta uno por uno los títulos, encabezados y cajas que estarán dentro de la aplicación, se designan los elementos y las hojas de estilo ayudarán para tener una mejor interfaz

* + ¿Cuál es la utilidad de la carpeta **public** en **Play**?

R= Son los archivos que serán visibles por el usuario, en este caso encontramos las hojas de estilo e imágenes, las cuales serán visibles dentro del navegador

* ¿Cómo se implementa la API de Google Maps?
  + ¿Por qué su implementación es diferente?

R= Se implementa siguiendo la documentación de google, utilizando los servicios y métodos que nos piden, es diferente al resto ya que es necesario especificar más valores para que pueda trabajar, también que al ser una aplicación externa tiene diferentes controles para cuidar la integridad de la misma

# Conclusiones

Esta fue la práctica más interesante hasta el momento del semestre, ya que gracias a la misma fuimos capaces de entender cómo se crean Servicios Web y como deberán ser llamados los mismos para su correcto uso.