Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Reporte final : Proyecto tecnológico

Arquitectura híbrida para la gestión de visitantes de la UAM Azcapotzalco.

Rodrigo Ortiz Martínez
Matricula 207331462

Asesora
Dra. Maricela Claudia Bravo Contreras

Profesor Asociado Departamento de Sistemas

Trimestre 2016 Otoño

Yo, Maricela Claudia Bravo Contreras, declaro que aprobé el contenido del presente reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Maricela Claudia Bravo Contreras
Firma

Yo,Rodrigo Ortiz Martinez , doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Rodrigo Ortiz Martinez
Firma

TABLA DE CONTENIDO Página

1.- RESUMEN

La tecnología de servicios Web es una distribución rápida de información entre clientes, proveedores, socios comerciales y sus diferentes plataformas.
Por otro lado es bien sabido que actualmente el mundo de las aplicaciones móviles está en su pleno apogeo, una de las ventajas más evidentes de su uso es la facilidad y rapidez con la que se accede a la información, ya que las aplicaciones están presentes en sus terminales en todo momento.
Bajo esta perspectiva se ha propuesto el desarrollo de una aplicación móvil que consuma una serie de servicios web, la principal ventaja es que un servicio web puede comunicar aplicaciones creadas con distintas tecnologías (sistemas operativos y lenguajes).
La aplicación desarrollada es una herramienta que sirve para una gestión de visitantes y de sus perfiles a la UAM Azcapotzalco, para ello el visitante puede desde la aplicación móvil registrar y dar de alta a visitantes, configurar sus datos, registrar cada visita así como consultar su historial visitas. Desde la base de datos el administrador puede visualizar los datos registrados y en un futuro hacer una evaluación y estadísticas de los perfiles de los visitantes y sus visitas.

Se utilizaron diferentes tecnologías como programación en Android, MySQL, lenguaje PHP para el desarrollo de los servicios Web así como el uso de un servidor para lograr la arquitectura híbrida propuesta como proyecto de integración.

2.- INTRODUCCIÓN

Los servicios Web están estandarizados, ya que utilizan los estándares XML y HTTP para la transferencia de datos y son compatibles con muchos otros entornos de desarrollo. Son por lo tanto independiente de plataformas. Es en este contexto existe un interés muy particular que se atribuye al diseño de servicios web que permiten a las empresas a ofrecer aplicaciones a las que se accede de forma remota por otras compañías.
Una ventaja de los servicios web es que no están ligados a ningún sistema operativo o lenguaje de programación y estos pueden pueden comunicarse entre diferentes lenguajes.

Por otro lado, es bien sabido que la tecnología evoluciona rápido, con ello crece el mundo de las aplicaciones móviles, en este contexto empresas, industrias y centros educativos entre otros recurren a las aplicaciones móviles que les reportan un mayor beneficio ya que es fácil y accesible disponer y acceder a la información desde cualquier lugar.
En la actualidad el uso de los servicios web para Android comienza a hacerse común e indispensable ya que con frecuencia necesitamos la comunicación y la transferencia de datos entre un dispositivo móvil y un sistema manejador de base de datos.

3.- ANTECEDENTES

3.1 Proyectos terminales

-En el proyecto “Implementación de un sistema web para dar seguimiento a las visitas realizadas a entidades financieras por parte de la Comisión Nacional Bancaria y de Valores” [1] se propone la implementación de un sistema web para registrar, consultar y actualizar información relevante de

visitas que se realizan a instituciones financieras, en dicho trabajo el cliente interactúa con el sistema mediante un navegador web, es algo similar a lo que se pretende sin embargo nuestra propuesta se enfoca a una aplicación para móvil así como realizar un historial de visitas.

-El proyecto [2] "Aplicación web para la visualización y análisis de datos extraídos del sistema ODBII para la gestión de una flota de camiones" tiene como objetivo el desarrollo de una aplicación web que despliegue información referente a la gestión de una flota de transportes y obtener también control sobre el personal que opera las unidades, nuestro sistema tiene también como objetivo gestionar información pero sobre visitantes a la universidad, a diferencia de este proyecto, se realizará una aplicación para un dispositivo móvil.

-En la actualidad existe un proyecto terminal [3] "Sistema de gestión escolar en Android" en el cual se implementa entre otros módulos un módulo para la consulta de datos académicos y personales de alumnos de la UAM Azcapotzalco en una aplicación móvil, nuestra propuesta empleará también aplicación para un dispositivo móvil pero a a diferencia se gestionaran perfiles de visitantes de la institución.

3.2 Artículos de investigación

-Existe un artículo de investigación "*Research and implementation of Web Services in Android network communication framework Volley*"[4] donde se prevé que la combinación de los Servicios Web y dispositivos móviles promoverá el desarrollo de aplicaciones móviles. Se propone el Volley framework Google 2013 para apoyar a los Servicios Web, que pueden facilitar el desarrollo de aplicaciones de servicios web, pero también puede mejorar el rendimiento del acceso a los servicios Web.

-"Mobile Web Service Provisioning and Discovery in Android Days"[5] es un artículo en el cual se propone una nueva arquitectura para el host móvil, actualizándolo a las tecnologías y los estándares de hoy. Por otra parte, se menciona que el anfitrión móvil es muy adecuado para el consumo y la orquestación de servicios en la nube. Los clientes pueden acceder a los datos del a través de HTTP o el protocolo XMPP usando la filosofía REST. Esto es una de las ideas principales de nuestra propuesta.

-El trabajo [6] "Método ágil híbrido para desarrollar software en dispositivos móviles" presenta un nuevo método ágil e híbrido para desarrollar software en dispositivos móviles, la evaluación de cuán ágil es, si es o no adecuada para el desarrollo de aplicaciones en dispositivos móviles y su aplicación en un caso real para evaluar sus bondades. Es un marco de referencia para nuestra propuesta, se propone a diferencia incluir servicios web.

3.3 Tesis

-En el trabajo de tesis [7],"Desarrollo de un directorio usando un servicio web" se plantea el diseño de una aplicación que utilice servicios Web para consultar, agregar y modificar datos de profesores de la base de datos del directorio del departamento de horarios de la ESIME Zacatenco del Instituto Politécnico Nacional. A diferencia de este trabajo, se propone una aplicación para un dispositivo móvil enfocado a perfiles de visitantes.

-Existe un trabajo de tesis del año 1992 [8] “Sistema de gestión automatizada de personal para la comisión de vialidad y transporte urbano” en la cual se propone la creación de una aplicación para el control de empleados del departamento del distrito federal en un registro único y conservando su historial a partir de su ingreso, se usa el lenguaje cobol y fue diseñada para un solo puesto de trabajo. A gran diferencia nuestro sistema utilizará servicios web y nuevas tecnologías.

-Un trabajo de tesis [9] *“Android Application Development”* tiene como objetivo investigar el proceso de diseño e implementación de una aplicación para Android que utilice un servicio web.
La entrada de la aplicación puede ingresar y recuperar datos de él y mostrar información al usuario, dicha tesis no menciona una aplicación y tipos de datos en particular. Es algo similar a lo que pretendemos y enfocarlo a una gestión de visitantes de la UAM Azcapotzalco.

4.- JUSTIFICACIÓN

Actualmente no existe un sistema de registro automático de visitantes externos a la UAM Azcapotzalco. El personal de vigilancia registra los datos en bitácoras y permite el acceso a las instalaciones, sin embargo no hay registro de perfiles ni historial de visitas.
Aunque este método es percibido como rápido y fácil, casi no proporciona ninguna seguridad y puede permitir que todos puedan ver la información sobre los visitantes. La posibilidad de ofrecer a una aplicación móvil de apoyo para el registro de los visitantes en la universidad hace de esto un enfoque rentable. Un sistema automatizado para la gestión de visitantes da una apariencia de una institución más segura y profesional; cuidando que se cumpla con las obligaciones en términos de de protección de datos de los visitantes, por otro lado permite dar seguimiento y gestión del tráfico de visitantes a las instalaciones de la universidad. Entre las posibles aplicaciones futuras de esta propuesta es su uso para ofrecer mecanismos de filtrado de visitantes.

5.- OBJETIVOS

5.1 Objetivo General

Diseñar e implementar una arquitectura híbrida formada por una base de datos y una colección de servicios Web para la gestión de datos de perfiles de personas que visitan la UAM Azcapotzalco y que éstos datos se adquieran a través de una aplicación móvil desarrollada con Android.

5.2 Objetivos Específicos

Desarrollar una arquitectura híbrida formada por una base de datos y una colección de servicios Web para que a través de ésta arquitectura se realice la gestión de visitantes de la UAM Azcapotzalco.

Diseñar e implementar una aplicación móvil a través de la cual se puedan adquirir de forma semiautomática los datos del visitante para generar un historial de visitas.

6.- MARCO TEÓRICO

El Protocolo de Transferencia de HiperTexto (*Hypertext Transfer Protocol*) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP.

Comunicación entre el navegador y el servidor [10]

La comunicación entre el navegador y el servidor se lleva a cabo en dos etapas: (figura 1)

-El navegador realiza una solicitud HTTP

-El servidor procesa la solicitud y después envía una respuesta HTTP

Solicitud HTTP

Una solicitud HTTP es un conjunto de líneas que el navegador envía al servidor. Incluye:

- Una línea de solicitud: es una línea que especifica el tipo de documento solicitado, el método que se aplicará y la versión del protocolo utilizada. La línea está formada por tres elementos que deben estar separados por un espacio:

- El método

- La dirección URL

- La versión del protocolo utilizada por el cliente (por lo general, HTTP/1.0)

- Los campos del encabezado de solicitud: es un conjunto de líneas opcionales que permiten aportar información adicional sobre la solicitud y/o el cliente (navegador, sistema operativo, etc.). Cada una de estas líneas está formada por un nombre que describe el tipo de encabezado, seguido de dos puntos (:) y el valor del encabezado.

Respuesta HTTP

Una respuesta HTTP es un conjunto de líneas que el servidor envía al navegador. Está constituida por:

- Una línea de estado**:** es una línea que especifica la versión del protocolo utilizada y el estado de la solicitud en proceso mediante un texto explicativo y un código. La línea está compuesta por tres elementos que deben estar separados por un espacio: La línea está formada por tres elementos que deben estar separados por un espacio:

- La versión del protocolo utilizada.

- El código de estado

- El significado del código

- Los campos del encabezado de respuesta: es un conjunto de líneas opcionales que permiten aportar información adicional sobre la respuesta y/o el servidor. Cada una de estas líneas está compuesta por un nombre que califica el tipo de encabezado, seguido por dos puntos (:) y por el valor del encabezado Cada una de estas líneas está formada por un nombre que describe el tipo de encabezado, seguido de dos puntos (:) y el valor del encabezado.

- El cuerpo de la respuesta: contiene el documento solicitado.

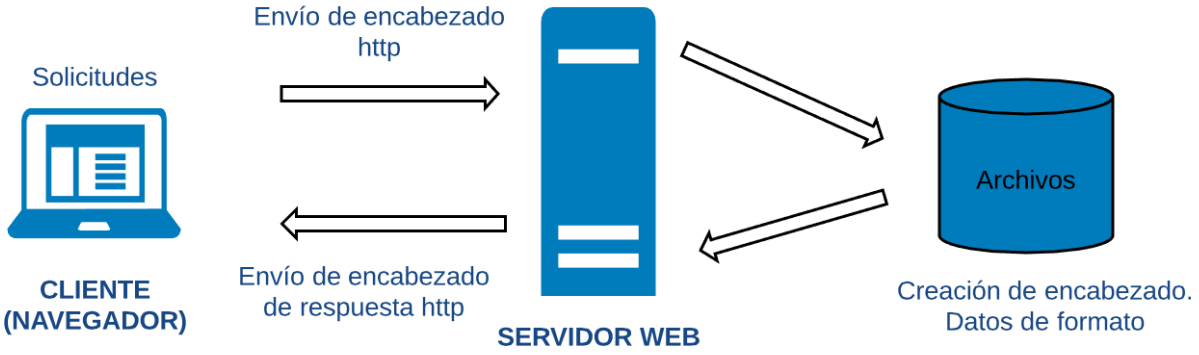


Figura 1. Arquitectura cliente servidor y uso de protocolo HTTP

Servicios web RESTful

REST (Representational State Transfer) es un estilo de arquitectura de software para sistemas hipermedias distribuidos tales como la Web. El término fue introducido en la tesis doctoral de Roy Fielding en 2000, quien es uno de los principales autores de la especificación de HTTP. [11]

Una implementación concreta de un servicio web REST sigue cuatro principios de diseño fundamentales:

• utiliza los métodos HTTP de manera explícita

• no mantiene estado

• expone URIs con forma de directorios

• transfiere XML, JavaScript Object Notation (JSON), o ambos

Una de las características claves de los servicios web REST es el uso explícito de los métodos HTTP, siguiendo el protocolo definido por RFC 2616. Por ejemplo, HTTP GET se define como un método productor de datos, cuyo uso está pensado para que las aplicaciones cliente obtengan

recursos, busquen datos de un servidor web, o ejecuten una consulta esperando que el servidor web la realice y devuelva un conjunto de recursos.

REST hace que los desarrolladores usen los métodos HTTP explícitamente de manera que resulte consistente con la definición del protocolo. Este principio de diseño básico establece una asociación uno-a-uno entre las operaciones de crear, leer, actualizar y borrar y los métodos HTTP. De acuerdo a esta asociación:

• Se usa POST para crear un recurso en el servidor.

• Se usa GET para obtener un recurso.

• Se usa PUT para cambiar el estado de un recurso o actualizarlo.

• Se usa DELETE para eliminar un recurso.

Android HTTP

A continuación se describen algunas clases utilizadas en el desarrollo del proyecto.

HttpURLConnection

La clase HttpURLConnection es sumamente ligera y forma parte del paquete java.net

Permite recibir y enviar información a través de la Web, lo que hace mas potente las aplicaciones Android y d ala ventaja de comunicarse con los maneadores de bases de datos los cuales no pueden ser conectados de forma directa ya sea el caso de SQL, MySQL, Oracle entre otros

Permite implementar la comunicación cliente-servidor con la API de Android [12]

Clase InputString

Esta clase abstracta es la superclase de todas las clases que representan un flujo de entrada de bytes. Esta clase permite obtener el flujo de datos asociados al recurso que se encuentra en la dirección especificada.

Clase BufferedReader

Permite leer el flujo de caracteres que ingresaron. Lee el texto de una secuencia de entrada de caracteres, proporciona una lectura eficiente de caracteres, matrices y líneas.

Clase StringBuilder

Permite trabajar con cadenas de caracteres sobre las cuales se va a realizar alguna modificación.

JSON

Es el acrónimo para JavaScript Object Notation. Es un estándar basado en texto plano que sirve para intercambio de información entre dos sistemas que no necesariamente comparten el mismo lenguaje.

JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

7.- DESARROLLO DEL PROYECTO

7.1 Descripción del Problema

Actualmente no existe un sistema de registro automático de visitantes externos a la UAM Azcapotzalco. El personal de vigilancia registra los datos en bitácoras y permite el acceso a las instalaciones, sin embargo no hay registro de perfiles ni historial de visitas.
Aunque este método es percibido como rápido y fácil, casi no proporciona ninguna seguridad y puede permitir que todos puedan ver la información sobre los visitantes. La posibilidad de ofrecer a una aplicación móvil de apoyo para el registro de los visitantes en la universidad hace de esto un enfoque rentable.

7.2 Descripción detallada de la solución software

Para la resolución óptima de dicho problema se propuso la siguiente solución software, que a continuación se describe basándose en los siguientes aspectos:

Descripción de la información a tratar, en la que describiremos la información que va a tratar este sistema de forma clara y concisa.

Descripción de la funcionalidad de la solución, en la que que se describe toda la funcionalidad de la que va a disponer el sistema a desarrollar.

7.3 Descripción de la información a tratar

La solución propuesta considerara la siguiente información acerca del problema:

Datos personales del visitante a la UAM Azcapotzalco

-Nombre y apellidos del visitante

-Edad del vistante

-Genero del visitante

-Correo electrónico del visitante

-Institución de procedencia

-Dispositivo móvil

Datos del registro de una visita a la UAM Azcapotzalco

-Fecha de visita

-Hora de entrada y hora de salida

-Puerta de acesso a la UAM Azcapotzalco

-Motivo de la visita

-Destino o persona a visitar

Cualquier visitante que descargue la aplicación Android puede:

- Darse de alta en el sistema rellenando un formulario con sus datos personales y el correo electrónico así como su dispositivo móvil que desea dar de alta.
- Modificar cualquier dato dado de alta en el sistema.Para ello deberá proporcionar su código de visitante que tiene registrado
- Registrar una nueva visita proporcionando los datos requeridos en un formulario
- Consultar su historial de visitas

El Administrador del sistema puede:

- Proporcionar o modificar el código de visitante
- Consultar la base de datos
- En un futuro realizar un análisis de perfiles de los visitantes, análisis de las visitas o generar reportes gracias a los datos obtenidos en la capa de persistencia de datos (Base de datos)

7.4 Recursos hardware y software

-Sistema de desarrollo Android Studio

-SDK de Java versión 1.7.0.

-JDK (java Development kit)

-Servidor Apache (En el caso de implementación se usará un servidor en la nube)

-Editor de texto para programar (Sumblime Text, Java Netbeans, Coda2 etc).

7.5 Análisis del sistema software.

El objetivo inicial de este proyecto fue el de diseñar e implementar una aplicación móvil a través de la cual se puedan adquirir de forma semiautomática los datos del visitante vía una colección de servicios web para poder generar posteriormente un historial de visitas.

Para esto se proposo una arquitectura híbrida la cual se compone de una colección de servicios Web basado en la API REST que permitan ofrecer una capa de tipo "*middleware*"para lograr un interoperabilidad entre una aplicación móvil y una capa de persistencia de datos.

La siguiente figura muestra los módulos de la arquitectura desarrollados en el proyecto (figura 2).

Figura 2. Diagrama general de la arquitectura

A continuación se describen cada uno de los módulos desarrollados en la arquitectura.

7.5.1 Lado del servidor

Durante el desarrollo del software se instaló el servidor XAMP, como se menciona anteriormente , se puede usar un servidor en la nube

Capa de persistencia de datos

Representa la base de datos del sistema, que será consultada y modificada por los procesos "Alta visitante", "Proceso de registro de visita", "Consulta del historial de visitas", "Configuración o modificación de datos de visitante".

En la figura 3 se muestra el modelo entidad relación de dicha capa. La capa de persistencia de datos se ha realizado en MysQL.

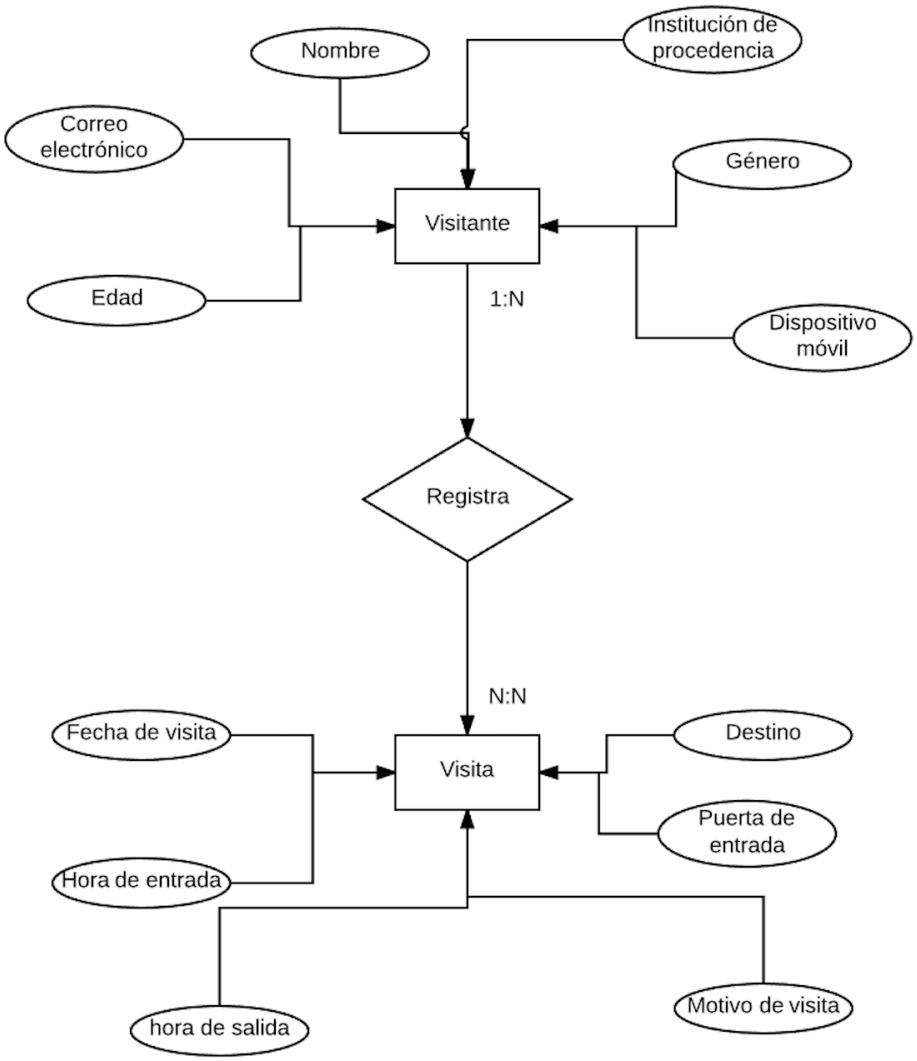


Figura 3. Modelo entidad relación

Colección de servicios Web

Los Servicios Web tienen como función enviar consultas SQL, recuperar e ingresar datos a la base de datos a través de la aplicación en Android .
Podemos crear los servicios Web en Java Netbeans o alguien editor de texto para programar, en nuestro caso usamos el editor coda para Macbook.
Creamos una carpeta para guardar el código de nuestros servicios .
Creamos la carpeta "Proyecto" y la guardamos en la carpeta htdocs perteneciente al servidor, como hemos utilizado un servidor local para el desarrollo del proyecto lo guardamos así:
XAPP->htdocs->proyecto, (véase Figura 4).

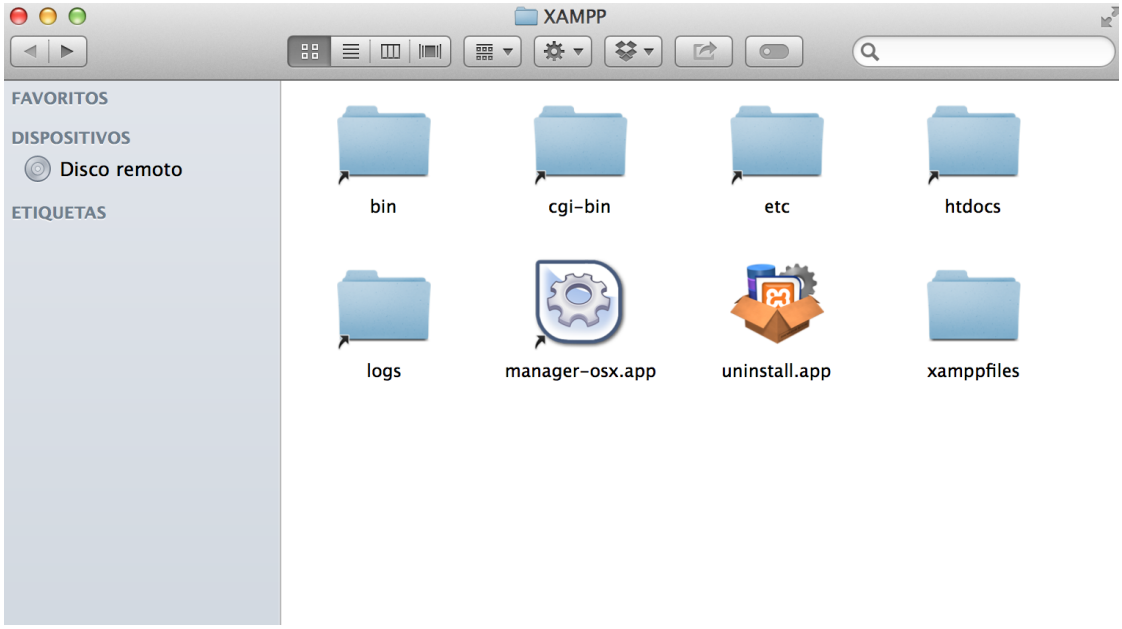


Figura 4. Carpeta htdocs en servidor local para guardar los servicios web

Servicio web para ingresar al sistema

El servicio de ingreso al sistema verifica que existe un usuario en nuestra base de datos,
se recibe el usuario y su contraseña
Creamos variables para usuario y contraseña, $_REQUEST recibe tanto el método GET como el método Post.

$usu=$_REQUEST['usu']; usuario
$pas=$_REQUEST['pas']; contraseña

$cnx=new PDO("mysql:host=localhost;dbname=Visitantes_UAM","root","");
Aquí ponemos el servidor, en nuestro caso es un servidor local, el nombre de la base de datos, el usuario que es "root" en servidor local así como una contraseña que hemos omitido pero que iría en las comillas en blanco.
La extensión *Objetos de Datos de PHP* (PDO por sus siglas en inglés) define una interfaz ligera para poder acceder a bases de datos en PHP.PDO proporciona una capa de abstracción de *acceso a datos*, lo que significa que, independientemente de la base de datos que se esté utilizando, se emplean las mismas funciones para realizar consultas y obtener datos [13].

Al objeto PDO se le indica la conexión a MySQL, el host, como es un servidor local le damos un localhost y en nombre de la base de datos, el usuario que en nuestro caso es "root" así como la contraseña a la base de datos, todos estos datos se deberán cambiar cuando el servicio se instale en un servidor en la nube.

Hecha la conexión se debe que ejecutar la respectiva consulta *select* bajo un comando query devuelve un conjunto de resultados como un objeto PDO.

*$res=$cnx->query("select * from visitante where codVis='$usu' and pasUsu='$pas'");*

El resultado se almacena en un arreglo para posteriormente enviarlos al objeto JSON.

$datos=array();* una vez que tenemos los datos los enviamos al objeto JSON para poder consumirlo en Android con *echo json_encode($datos);

Si el usuario existe *$res = 1*, si no existe *$res = 0*, (véase Figura 5).

```php
<?php

//recibiendo usuario y pas
$usu=$_REQUEST['usu'];
$pas=$_REQUEST['pas'];

$cnx=new PDO("mysql:host=localhost;dbname=Visitantes_UAM","root","");
$res=$cnx->query("select * from visitante where codVis='$usu' and pasUsu='$pas'");

$datos=array();

foreach ($res as $row){
    $datos[]=$row;
}

echo json_encode($datos);


?>
```

Figura 5. Servicio Web par autenticarse al sistema.

Prueba servicio de ingreso al sistema

Ahora se realiza una verificación del funcionamiento del servicio, para ello ingresamos algunos datos a nuestra base de datos (Figura 6).

4	A0003	Marco perez Ruiz	30	Masculino	UNAM-Aragon	Huawei 334	marcos@hotmail.com	333333
5	A0004	Joaquin Gonzalez Ortega	30	Masculino	Personal	Huawei 334	marcos@hotmail.com	444444
10	A0001	Jose Perez Diaz	51	Masculino	ESIME Zacatenco	Motorol	Jos222@live.com	111111

Figura 6. Datos de visitantes ingresados en la base de datos

Se entra a la página del servidor local y en la URL llamamos al servicio, para verificar que funciona

Ingresamos un usuario y su respectiva contraseña, en este caso dimos como usuario A0001 y contraseña 111111, estos datos los ingresamos arbitrariamente a la base de datos tal como se muestra en la siguiente figura (Figura 7).

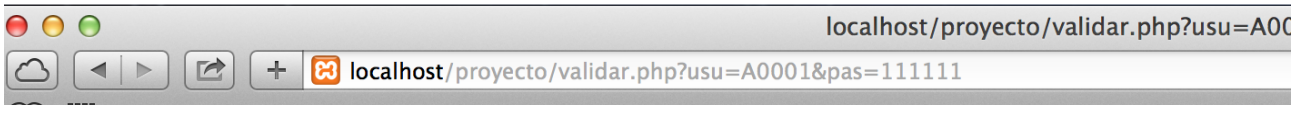


Figura 7. URL para verificar la funcionalidad del servicio web de ingreso al sistema

Se verifica el correcto funcionamiento del servicio web en el servidor local (véase figura 8).

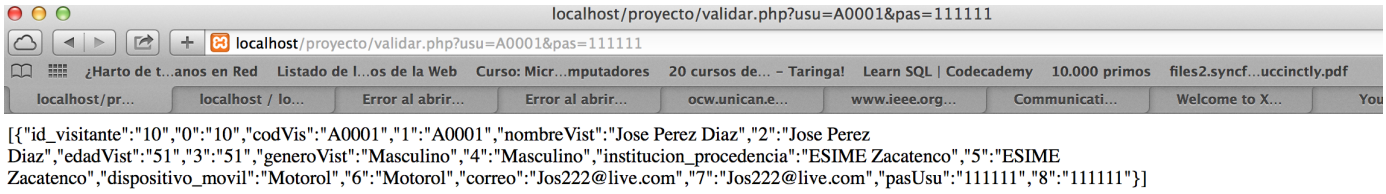


Figura 8. Ejecución del servicio web de ingreso al sistema desde el navegador.

Servicio web para un nuevo registro

Dicho servicio es necesario para poder ingresar a un nuevo visitante a la base de datos desde la aplicación Android.
De igual manera que el servicio para ingresar al sistema, utilizamos *$_REQUEST* para las variables, PDO y un *query* para la sentencia SQL y el envío de datos a la base de datos (figura 9).

```php
<?php


$cod=$_REQUEST['cod'];
$nom=$_REQUEST['nom'];
$eda=$_REQUEST['eda'];
$gen=$_REQUEST['gen'];
$pro=$_REQUEST['pro'];
$dis=$_REQUEST['dis'];
$mal=$_REQUEST['mal'];
$pas=$_REQUEST['pas'];

$cnx=new PDO("mysql:host=localhost;dbname=Visitantes_UAM","root","");
$res=$cnx->query("insert into visitante (codVis,nombreVist,edadVist,generoVist,institucion_procedencia,dispositivo_movil,correo,pasUsu)
values('$cod','$nom','$eda','$gen','$pro','$dis','$mal','$pas')");
```

Figura 9. Servicio web para el registro de un nuevo visitante

Servicio web para listar visitante

Este servicio es necesario para listar los datos del visitante en la aplicación ya sea para mostrar los datos o bien modificarlos vía la aplicación móvil.

```php
<?php

$visit=$_REQUEST['visit'];

$cnx=new PDO("mysql:host=localhost;dbname=Visitantes_UAM","root","");

$res=$cnx->query("select * from visitante where codvis='$visit'");

$datos=array();

foreach ($res as $row){
    $datos[]=$row;
}

echo json_encode($datos);


?>
```

Al igual que en el servicio de ingreso al sistema se utiliza un *query* para la consulta SQL y obtenemos los datos de un visitante con su respectivo código de visitante (figura 10).

Figura 10. Servicio web para listar un visitante

Dicho servicio será consumido por un objeto JSON y enviado a la aplicación móvil.
A continuación se muestra un ejemplo de ejecución del servicio (Figura 11).

Figura 11. Ejecución de la URL del servicio web para listar un visitante

Servicio web para listar historial de visitas

El servicio web para historial de vistas permite visualizar todo el historial de visitas de cada visitante.
El visitante puede acceder a su historial de visitas desde la aplicación móvil, para lograr esto es necesario un servicio web que permita mostrar todas las visitas registradas en la base de datos, al igual que en los casos anteriores se hace una interacción con la base a través de un *query*.
El servicio web de igual manera se ha realizado en lenguaje PHP y se consume a través de un objeto JSON (Figura 12).

```php
<?php

$visit=$_REQUEST['visit'];

$cnx=new PDO("mysql:host=localhost;dbname=Visitantes_UAM","root","");

//$res=$cnx->query("select id_visita,fecha_visita from registro_visita where id_visita='$visit'");
$res=$cnx->query("select id_visita,fecha_visita FROM registro_visita WHERE codVis='$visit'");


$datos=array();

foreach ($res as $row){
    $datos[]=$row;
}

echo json_encode($datos);


?>
```

Figura 12. Servicio web para listar el historial de visitas

Ejemplo de ejecución del servicio de listar historial de visitas dandole valor a la variable del código

de visitante a través de la URL en el navegador web (Figura 13).

Figura 13. Ejecución del servicio web para listar el historial de visitas

Servicio Web para actualizar datos del Visitante

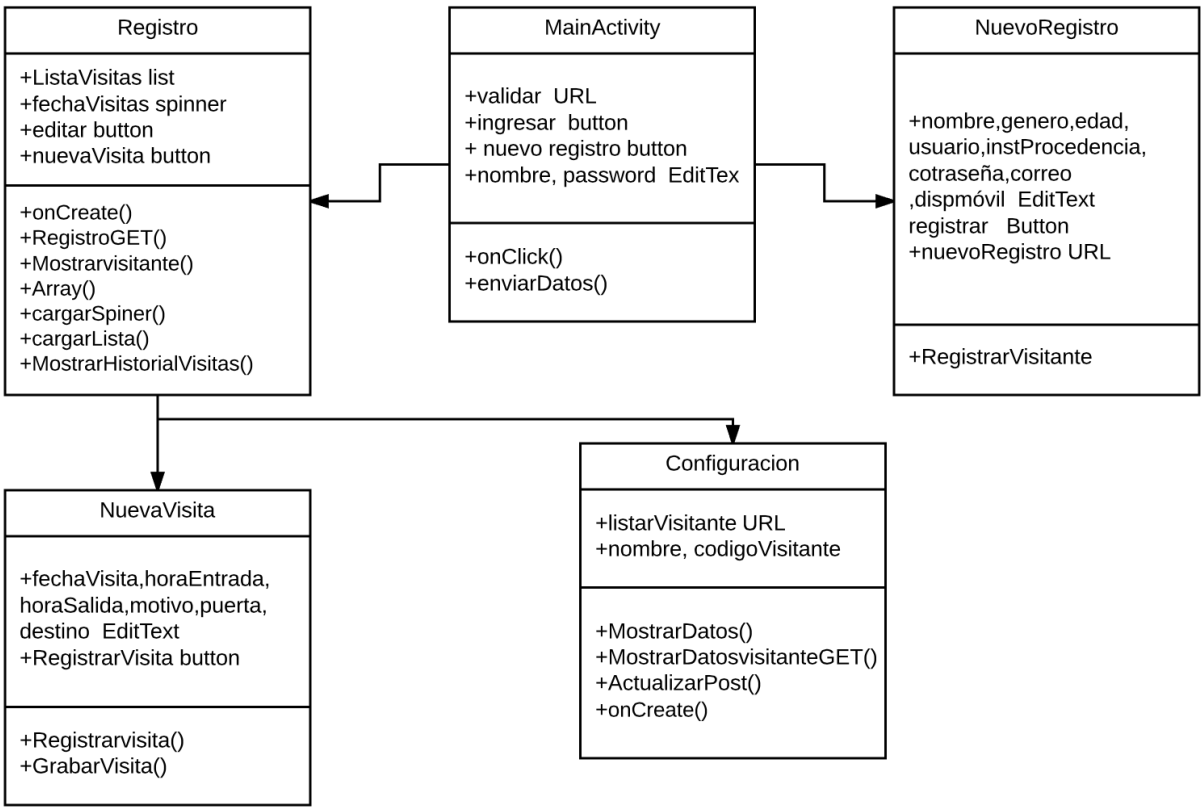
El visitante podrá actualizar sus datos vía la aplicación móvil, para esto es necesario crear un servicio que actualice los datos (Figura14).

```php
<?php

$cod=$_REQUEST['cod'];
$dim=$_REQUEST['dim'];
$eda=$_REQUEST['eda'];
$cor=$_REQUEST['cor'];
$pas=$_REQUEST['pas'];

$cnx=new PDO("mysql:host=localhost;dbname=Visitantes_UAM","root","");



   $res=$cnx->query("update visitante set dispositivo_movil='$dim', edadVist='$eda', correo='$cor', "
   . "pasUsu='$pas' where codVis='$cod'");


 ?>
```

Figura 14. Servicio web para actualizar los datos de un visitante

7.5.2 Lado del cliente

Desarrollo de la aplicación Android



La aplicación Android consta de las siguientes clases (Figura15).

Figura 15. Clases del proyecto en la aplicación móvil

Casos de uso del sistema

A continuación se muestra un diagrama con los principales casos de uso del sistema desarrollado.

El visitante puede registrarse, autenticarse, consultar su historial de visitas, registrar una nueva visita, modificar sus datos desde la aplicación móvil y el administrador puede registrar a un nuevo visitante , borrarlo o nidificarlo, así mismo consultar los perfiles de los visitantes desde la base de datos (Figura16).

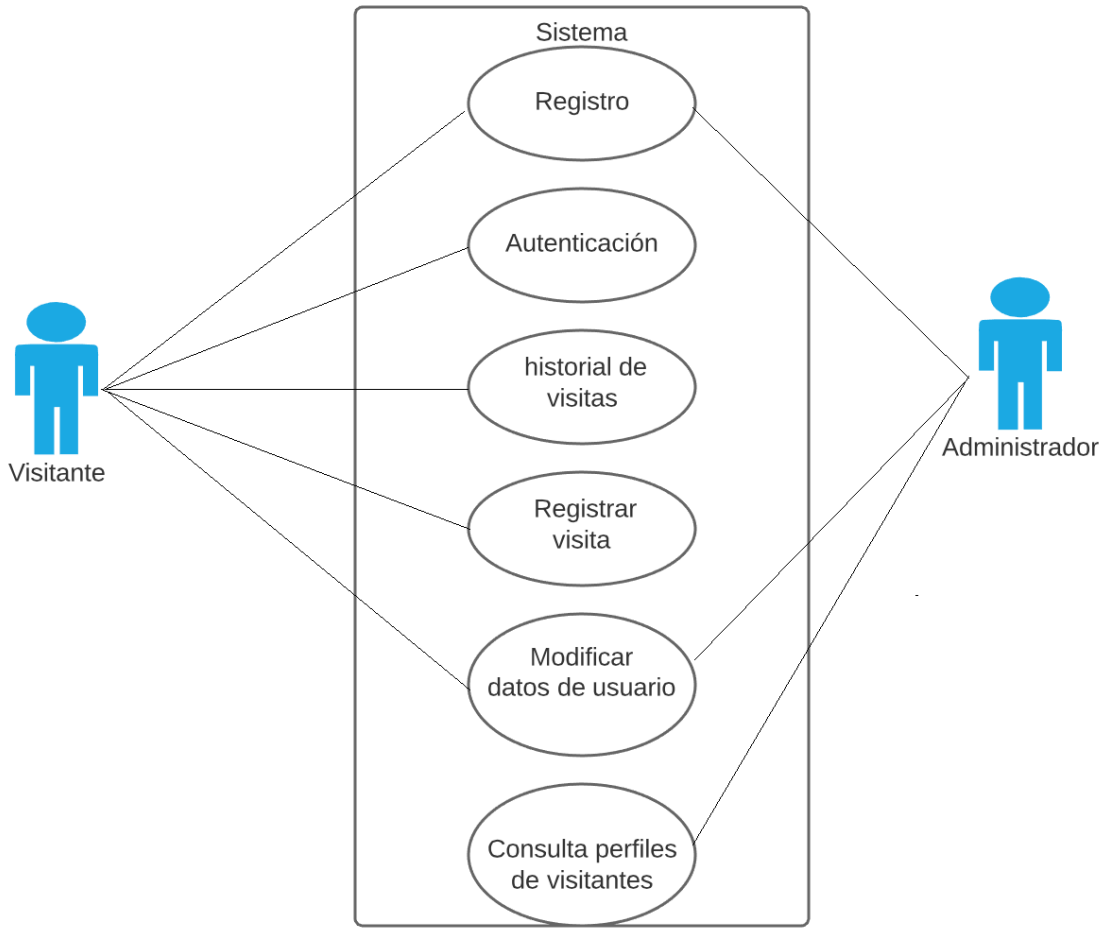


Figura *16.* Casos de uso del sistema

7.6 Manual de uso

Registro del Visitante

Desde la aplicación móvil se podrá registrar un nuevo visitante, los datos introducidos al sistema se almacenarán automáticamente a la base de datos vía el servicio web correspondiente (Figura17).

Figura 17. Registro de un visitante desde la aplicación

Autenticación del visitante registrado y consulta de historial de visitas

El visitante puede ingresar al sistema desde la aplicación mediante la identificación del usuario utilizando su código de visitante y su contraseña.

Una vez ingresado, el visitante puede registrar una nueva visita o bien consultar su historial de visitas (Figura18).

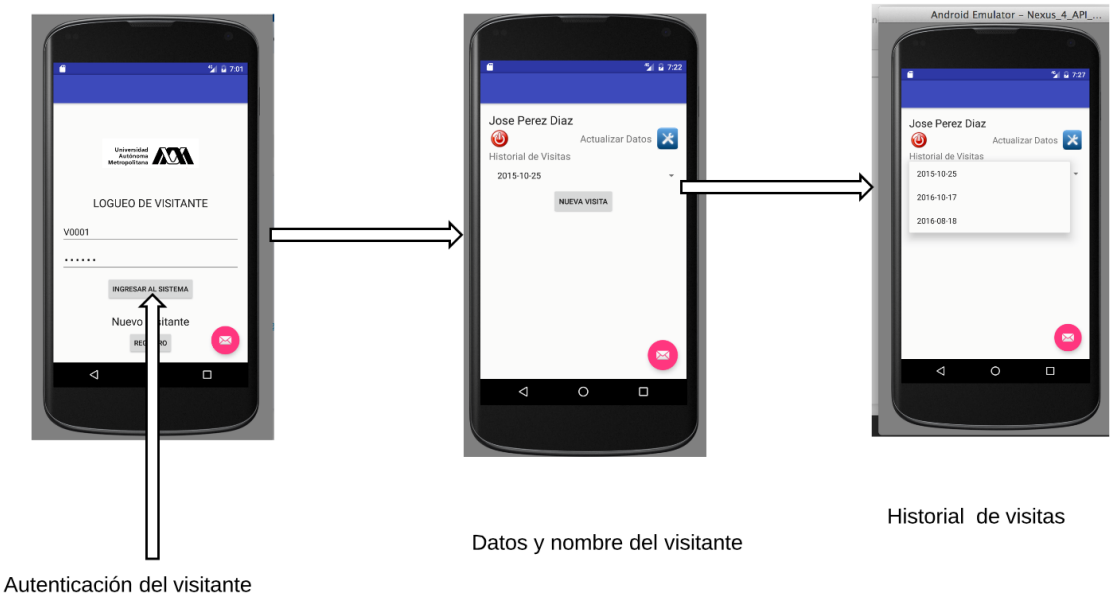


Figura 18. Autenticación de visitante y consulta de historial de visitas

Registro de nueva visita

El visitante registrado puede registrar una nueva visita desde la aplicación, los datos de la visita son registrados automáticamente en la base de datos (Figura19).

Figura 19. Registro de una nueva visita

Configuración de datos del visitante

Si el visitante requiere cambiar algún o algunos datos de su perfil, lo puede hacer vía la aplicación móvil. La aplicación muestra los datos registrados del visitante y pueden ser actualizados (Figura20).

Figura 20. Actualización de los datos de un visitante

7.7 Consumo de los Servicios Web

A continuación se describen con detalle el consumo de servicios web desde la aplicación móvil.

Registro de vistantes

Esencialmente se usa el método POST para el envío de datos desde la aplicación movil.

Los datos enviados por el método POST pasa por encabezado HTTP lo que la seguridad depende del protocolo HTTP.

Definimos el método insertar con los diferentes parámetros de entrada, nombre, edad, institución de procedencia etc (Figura 21).

public void insertar(String c, String n, String e, String g, String p, String d, String m, String w)

```
public void insertar( String c, String n, String e, String g, String p, String d, String m, String w) {
    String urlParameters="cod="+c+"&nom="+n+"&eda="+e+"&gen="+g+"&pro="+p+"&dis="+d+"&mal="+m+"&pas="+w;
    HttpURLConnection conection=null;
    try{
        URL url=new URL("http://192.168.1.65/proyecto/nuevo_registro.php");
        conection=(HttpURLConnection)url.openConnection();

        //estableciendo el metodo
        conection.setRequestMethod("POST");

        //longitud de datos que se envian por el método post
        conection.setRequestProperty("Content-Length", "" + Integer.toString(urlParameters.getBytes().length));

        //comando para la salida de datos
        conection.setDoOutput(true);

        DataOutputStream wr=new DataOutputStream(conection.getOutputStream());
        wr.writeBytes(urlParameters);
        wr.close();

        InputStream is=conection.getInputStream();


    }catch (Exception ex){}
```

Figura 20. Método para insertar un nuevo visitante en el sistema

Se declaran los parámetros por separado para enviarlos mediante el método POST y se utiliza la clase *HttpURLConnection* para la conexión a la base de datos, dichos parámetros son los mismos definidos en nuestro servicio web nuevo_registro.php

Se define la variable URL *url=new URL("http://192.168.1.65/proyecto/nuevo_registro.php");*

que es la que nos almacenará la url de nuestro servicio.

Esta URL (http://192.168.1.65) se cambiará por la IP del servidor en la nube en caso de instalarlo y esto en cada variable del mismo tipo.

Dentro del método insertar se declaran comandos como la declaración del método, la longitud y la manera de como enviar los datos.

conection.setRequestMethod("POST"); // declaración del método

Es necesario indicar la longitud de datos que se van a enviar y un comando para la salida de datos.

conection.setRequestProperty("Content-Length", "" +
Integer.toString(urlParameters.getBytes().length));

conection.setDoOutput(true); //salida de datos

Ahora se envía el POST para la salida de datos.

DataOutputStream wr=new DataOutputStream(conection.getOutputStream());

Implementación del botón de Registro

Una vez establecido el método es necesario programar el botón para enviar los datos obtenidos a la base de datos.

Se declara una variable y se enlaza a su respectivo botón en el diseño de la pantalla.

btnRegistrar=(Button)findViewById(R.id.btnRegister);

Se agrega el método click al botón y mediante un hilo llamaremos al método insertar.

Para poder trabajar con los métodos es necesario implementar un hilo.

Los Hilos o los "*Threads*" en Java, son básicamente una forma de poder ejecutar varios procesos simultáneamente en nuestros programas en Java.

Usualmente para poder utilizarlos tenemos que crear clases que extienden a la clase Thread, y reescribir el método principal "*run()*", el cual es el que se va a ejecutar principalmente al iniciar un hilo, *thread* o nuevo proceso en java.

Autenticación

La manera principal de pasar los datos será mediante el método GET el método GET lo que hace es pasar las variables y sus valores a través de una URL.

Para ingresar al sistema se necesita el servicio web validar.php con la siguiente URL:

http://localhost/proyecto/validar.php

Si los datos son correctos se devuelve un objeto JSON con los datos obtenidos de la base de datos, y si no son correctos se regresa un objeto JSON vacío.

En el archivo *Main* del proyecto Android enlazamos las variables a los controles de los archivos del diseño layout XML.

txtUsu=(EditText)findViewById(R.id.txtusu);

txtPas=(EditText)findViewById(R.id.txtpas);

btnIngresar=(Button)findViewById(R.id.btnIngresar);

btnNuevo=(Button)findViewById(R.id.btnNuevoreg);

Método para enviar datos, devuelve un dato que lo tomaremos como un tipo string

public String enviarDatosGET(String usu, String pas){ // recibe el usuario y su contraseña

URL url=null; // se almacena la URL a la cual haremos llamada.

En nuestro caso es la URL del servicio web: http://localhost/proyecto/validar.php.

try{

url=new URL("http://192.168.1.65/WebService/valida.php?usu="+usu+"&pas="+pas)};

Se permite acceder al servicio,tenemos que poner el ip de nuestra máquina o el ip del servidor en la nueve. En Android Studio no funciona el *localhost*, el usuario es un usuario que ingresa como parámetro al igual que la contraseña.

//*try, catch* es un control de errores obligatorio para los métodos en lenguaje java.

para que los datos puedan salir a ese URL necesitamos abrir una conexión que permita enviar los datos

Se necesita la clase *HttpURLConnection* y se le aplica a la variable url un openConnection :

HttpURLConnection conection=(HttpURLConnection)url.openConnection();

Al momento que sale la data,se almacena en una variable de la siguiente manera:

respuesta=conection.getResponseCode();

si hay una respuesta consumimos un objeto JSON del servicio web

resul=new StringBuilder();

Un objeto *StringBuilder* es una secuencia de caracteres mutable, su contenido y capacidad puede cambiar en cualquier momento[6].

```
if(respuesta==HttpURLConnection.HTTP_OK){
        InputStream in=new BufferedInputStream(conection.getInputStream());
        BufferedReader reader=new BufferedReader(new InputStreamReader(in));
         while((linea=reader.readLine())!=null){
           resul.append(linea);
        }
    }
```

Si hay respuesta, tomamos la respuesta, se almacena en la variable in y la clase *BufferedReader* se encarga de leerla de la siguiente manera:

```
BufferedReader reader=new BufferedReader(new InputStreamReader(in));
```

Con w*hile* llenamos la variable resul con el dato que hemos traído y retornamos lo que hemos traído como una cadena *return resul.toString();* lo que retornamos son estos datos (Figura 21).

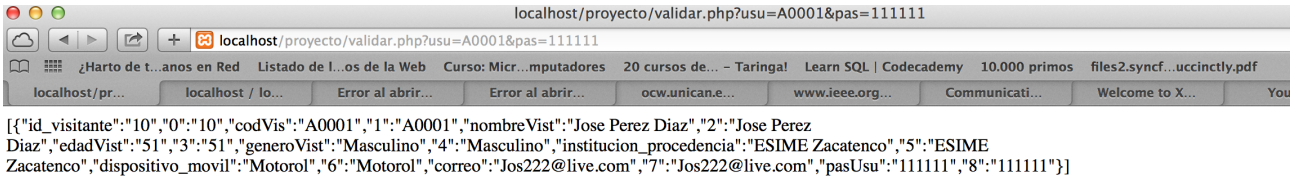


Figura 21. URL del Servicio web para ingresar al sistema

Ahora resta a saber si el objeto JSON tiene o no datos y para ello se crea un método

```
public int obtDatosJSON(String response){  }
```

Si tiene datos quiere decir que el usuario y la contraseña son correctos y se pasa a la ventana de visitante registrado, si no se retorna un 0.

Implementación del botón de ingreso

Se necesitara un método *public void onClick(View v)* para poder implementar uno de los eventos realizados sobre un botón.

Como en el método de registro, es necesario implementar un hilo. Usualmente para poder utilizarlos tenemos que crear clases que extienden a la clase *Thread*, y reescribir el método principal "r*un()*", el cual es el que se ejecuta principalmente al iniciar un hilo, *thread* o nuevo proceso en java.

El método *run ()* nos va a permitir ejecutar el método en el hilo, el hilo nos va a permitir enviar tanto al usuario como su contraseña al método *enviarDatosGET.*

final String resultado=enviarDatosGET(txtUsu.getText().toString(), txtPas.getText().toString());

Debemos implementar un método que nos permita trabajar con la interfaz gráfica del usuario desde el hilo.

Este método verifica si el objeto JSON tiene datos para poder llegar a una nueva ventana e iniciar una nueva actividad Si el JSON no contiene datos , no se entra al sistema y se envía un mensaje de datos (usuario y/o contraseña) incorrectos.

Historial de visitas

El visitante puede consultar el historial de sus visitas, las visitas registradas por el visitante pueden ser visualizadas desde la aplicación

Se utiliza el método *public String RegistroGET(String cod,String historial,String accion){}.*

Este método al igual que los anteriores, si hay una acción de mostrar historial de visita, se usa una variable url que selecciona el servicio web correspondiente, los datos obtenidos gracias a el servicio se almacenan en un objeto JSON para ser consumido por la aplicación móvil

Se le manda como parámetro el código de visitante, su historial y la acción del método

Este método devuelve un objeto JSON, y es necesario un método para transformar el dato obtenido en un arreglo y así poder desplegarlo en una lista.

public ArrayList<String> ArregloSpiner(String response){..}

response es el JSON que devuelve el método, el JSON es descompuesto mediante un bucle y es almacenado en un texto trajeado el campo "fecha visita" y se regresa el listado (Figura 22).

```
for(int i=0; i<json.length();i++){
    texto=json.getJSONObject(i).getString("fecha_visita");
    listado.add(texto);
}
```

Figura 22. Segmento de código para listar visitas.

El método *public void cargarListView(ArrayList<String> datos){..}* permite cargar una lista en Android, esta lista contendrá el historial de vistas del usuario ingresado al sistema.

El adaptador se adapta al spinner de la siguiente manera:

ArrayAdapter<String> adaptador=new ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,datos);

Configuración

En esta parte el visitante puede modificar sus datos a través de la aplicación móvil

Para tal objetivo el visitante debe estar debidamente ingresado al sistema. Al igual que el método de registro visitante, se utiliza el método POST.

Mediante el método *final Bundle recupera=getIntent().getExtras()*; se recupera el nombre y el código del visitante ingresado.

Al igual que en el registro de visitante, el método toma como variables los datos del visitante, estos datos se muestran en la pantalla de configuración y son editables para que el visitante pueda actualizarlos.

Para mostrar los datos del visitante se utiliza la clase *HttpURLConnection* y se consume el servicio web http://192.168.1.65/proyecto/listar_visitante.php con su respectivo objeto JSON y se recuperan los datos registrados en la base de datos.

Se enlazan las variables a sus respectivos *EditText* de la pantalla correspondiente en la aplicación móvil y desde el botón de actualización se hace llamada al método ActualizarPost()

ActualizarPost() inserta los nuevos datos a la base de datos de la misma manera que el registro de visitante y al momento de invocar el evento OnClick del botón de envío, estos usan un hilo para realizar correctamente la transferencia.

Registro de nueva

El visitante puede registrar una nueva visita desde la aplicación móvil.

Los métodos son iguales que en el caso de registro de un visitante al sistema.

Se declara el método *public void registrar_visita( )* y los parámetros son enviados mediante el método POST.

Este método usa como parámetro un URL que es e correspondiente al servicio web para registrar una nueva visita.

URL url=new URL("http://192.168.1.65/proyecto/nueva_visita.php");

Se hace uso de la clase *HttpURLConnection* para el consumo del objeto JSON y finalmente se usa un hilo para poder poder ejecutar y hacer conexión con la base de datos.

7.8 Descarga e instalación de software

Software a descargar:

-JDK

-Android Studio

-Netbeans o algún editor de código

-WampServer o algún servidor en la nube

Descarga del JDK

El Kit de desarrollo de Java (JDK) contiene las herramientas y librerías necesarias para crear y ejecutar applets y aplicaciones en Java.

En el navegador de búsqueda escribimos “descargar JDK” (figura 23)

Damos click en el vinculo y nos envia la página de Oracle donde pondremos descargar Java y JavaNetbeas en caso de necesitarlo (Figura 24).

Figura 23. Búsqueda del JDK en el navegador de google

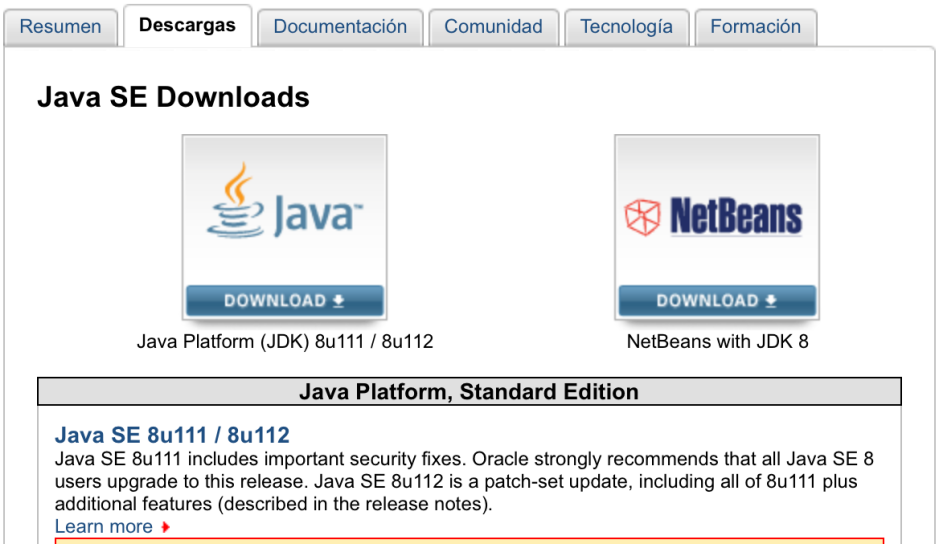


Figura 24. Descarga del JDK de java y de NetBeans

Damos click en java y escogemos la versión adecuada a nuestro equipo de computo (Figura 25).

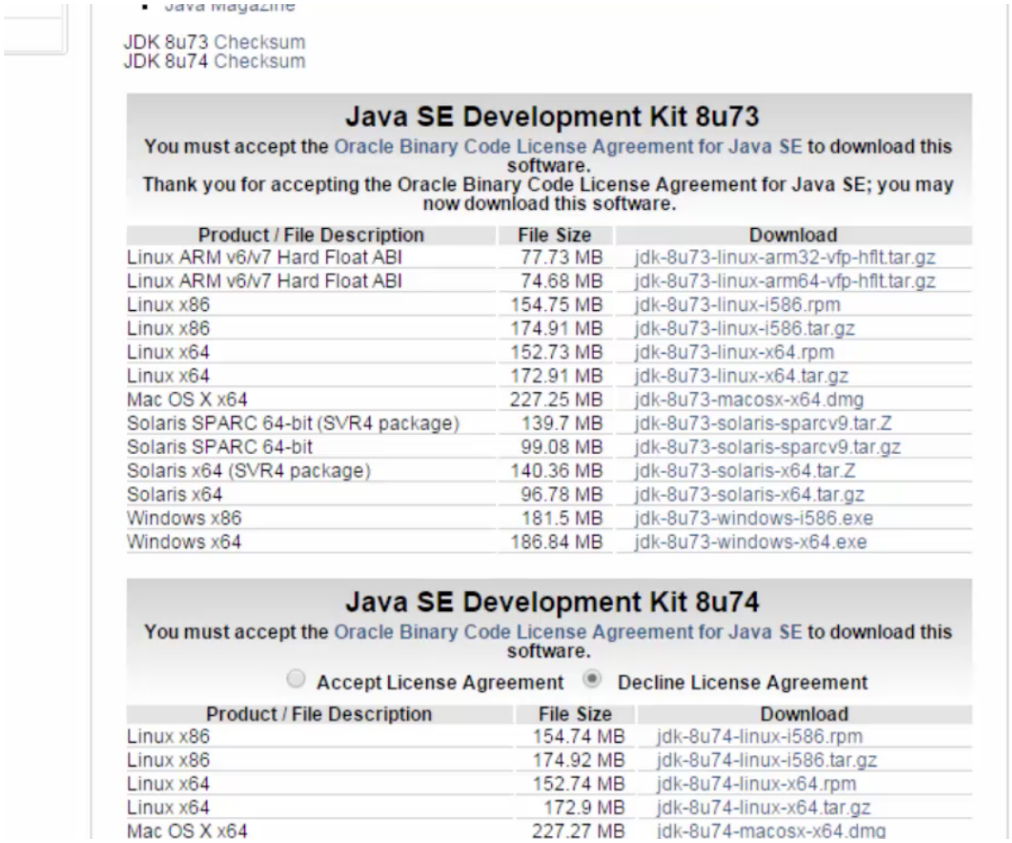


Figura 25. Opciones de descarga para el JDK

De igual manera en caso de usar Java Netbeans procedemos de manera similar o bien podemos descargar algún editor de código como por ejemplo sublimtext (Figuras 26 y 27) .

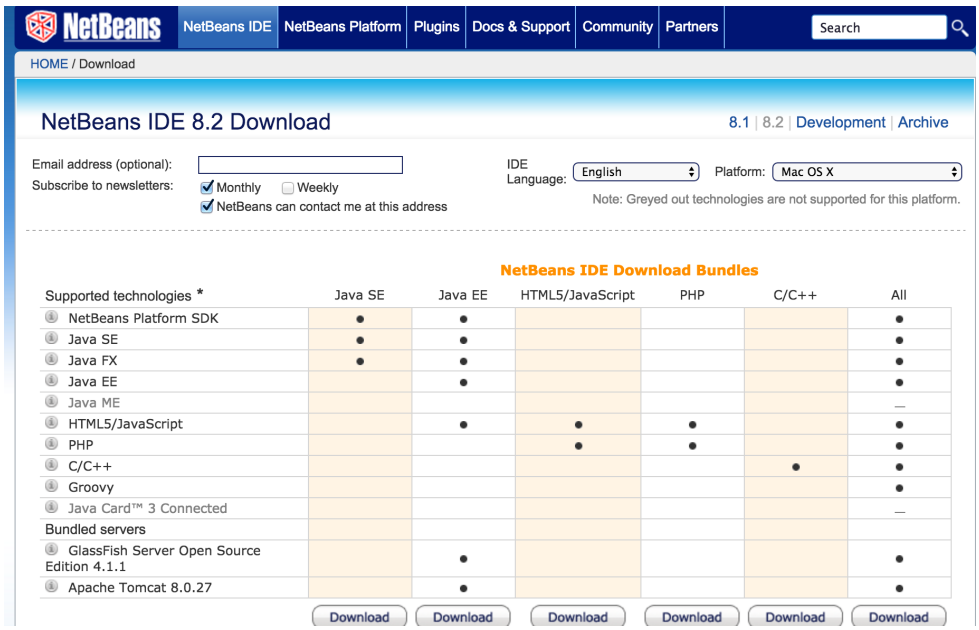


*Figura 26. D*escarga de Netbeans

Figura 27. Descarga de Sublime Text

Se procedió a descargar Android Studio , automáticamente se selecciona la versión correspondiente a nuestro sistema operativo (Figura 28).

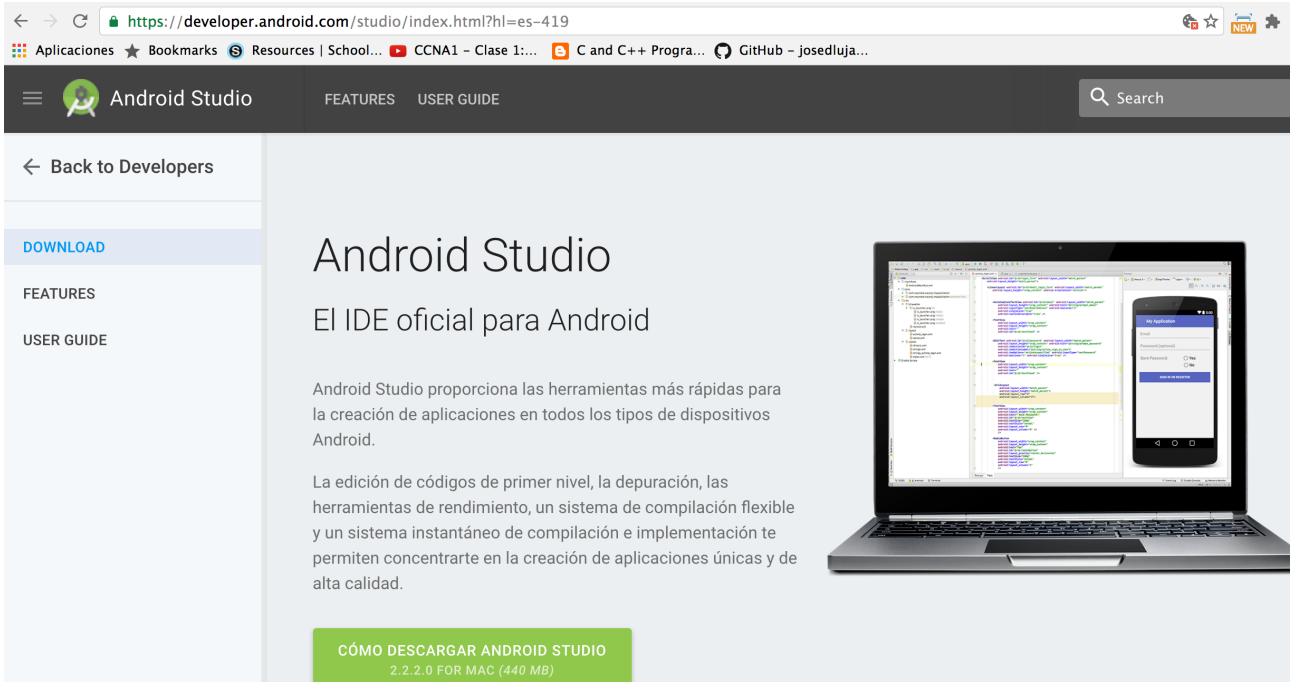


Figura 28 Descarga de Android Studio

Se Necesita un servidor local o un servidor en la nube, para un servidor local podemos usar WampServer o MampServer según nuestro sistema operativo (figura 29).

Figura 29 Descarga de servidor local.

Instalación del software

Una vez que tenemos la descarga completa de los programas necesarios procedemos a su respectiva instalación.

Instalar el JDK 8 es muy sencillo. Luego de bajar el software correspondiente, ejecutar el archivo *jdk-8u65-windows-x64.exe* y seguir las instrucciones. Opcionalmente, se puede elegir cambiar el directorio por defecto en donde instalar el JDK (Figura 30).

Figura 30. Descarga del JDK

Instalación de Android Studio

De la misma manera damos doble click al icono de Android Studio y se procede a ejecutar.

Seguir el asistente de instalación de Android Studio e instalar las SDK tools necesarias. En algunos sistemas Windows, puede ocurrir que el asistente no encuentre la ruta de instalación de Java. Si ocurriera esto, debemos ir a las Variables de entorno y establecer cuál es la ruta en la que se localiza la instalación de la JDK.Para ello ir a Menú Inicio > Equipo > Propiedades > Propiedades Avanzadas (Figura 31).

De igual manera si deseamos instalar Java Netbeans para el desarrollo de los servicios web solo se da click en el icono y seguimos los pasos, vemos los componentes y entre ellos agregamos Apache Tompac. (Figura 32)

Cabe señalar que podemos instalar un editor de código como sublime Text o bien Coda o algún otro editor de código.

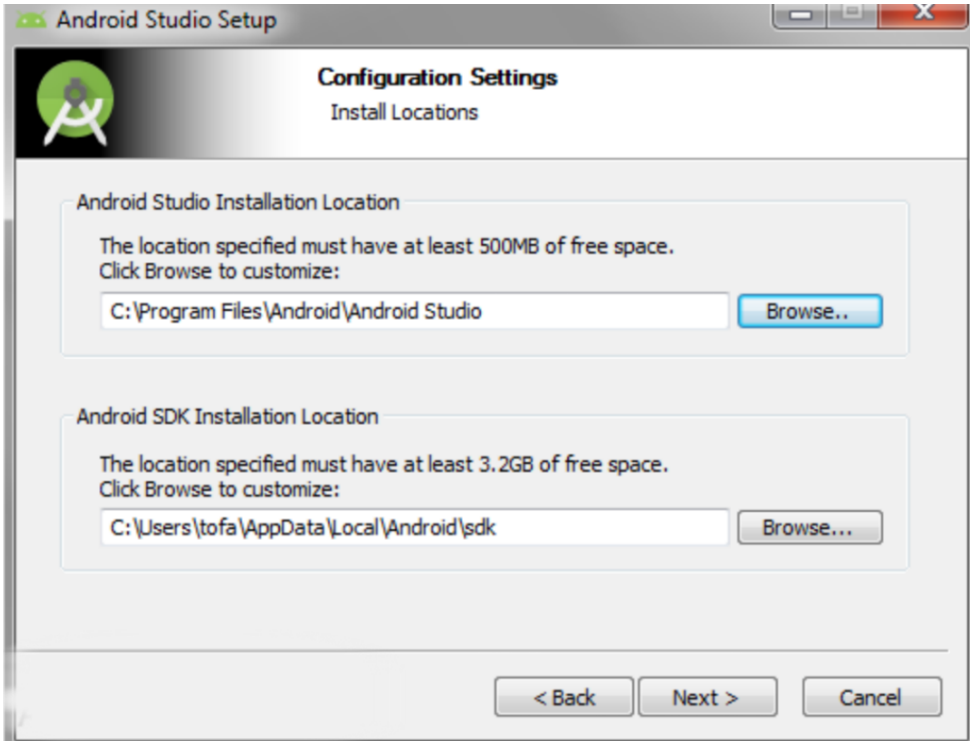


Figura 31. Instalación de Android Studio

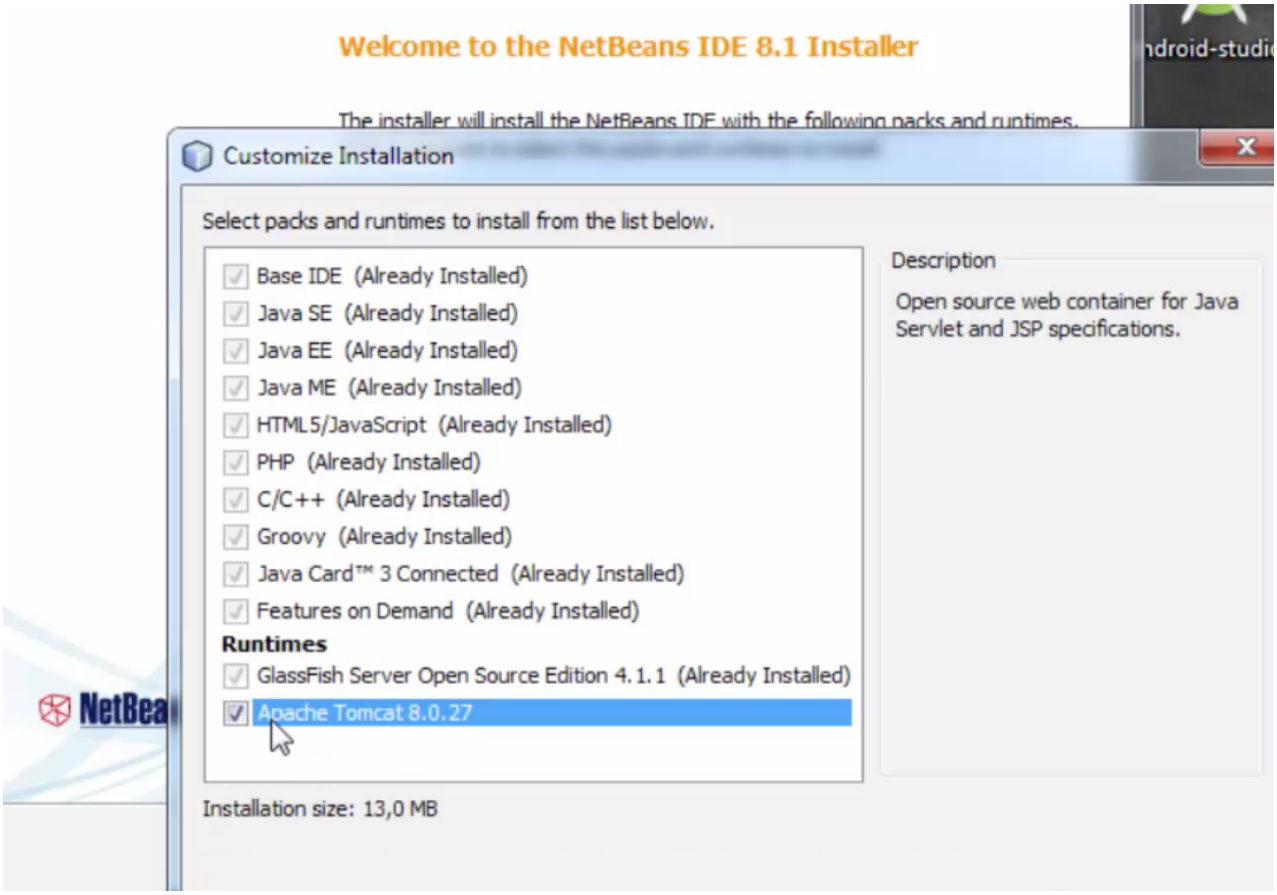


Figura 32. Instalación de Java Netbeans

Servidor Local

Se procede a la instalación del servidor local, por ejemplo WampServer dandole click en el icono de la aplicación y siguiendo las instrucciones (Figura 33).

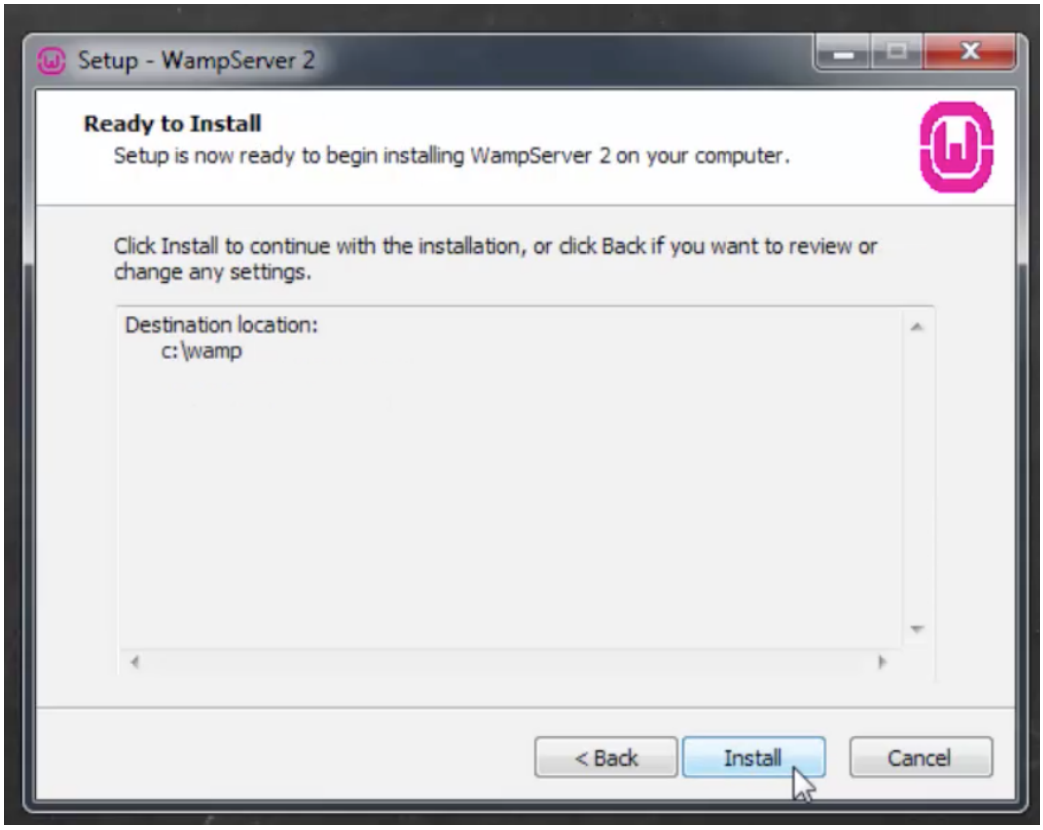


Figura 33. Instalación del servidor local

En ocasiones debemos verificar si el puerto 80 de Wampserver esta ocupado esto lo podemos ver verificando el archivo *httdp.conf* si estamos trabajando con windows, entonces debemos cambiar al puerto 82 (Figura 34).

Cerramos y guardamos los cambios y se reincida.

```
#
# If your host doesn't have a re
#
ServerName localhost:80

#
```

Figura 34. Segmento del archivo *httdp.conf para cambiar de puerro en Windows*

En Mac se utilizó el servidor sin hacer alguna modificación (Figura 35).

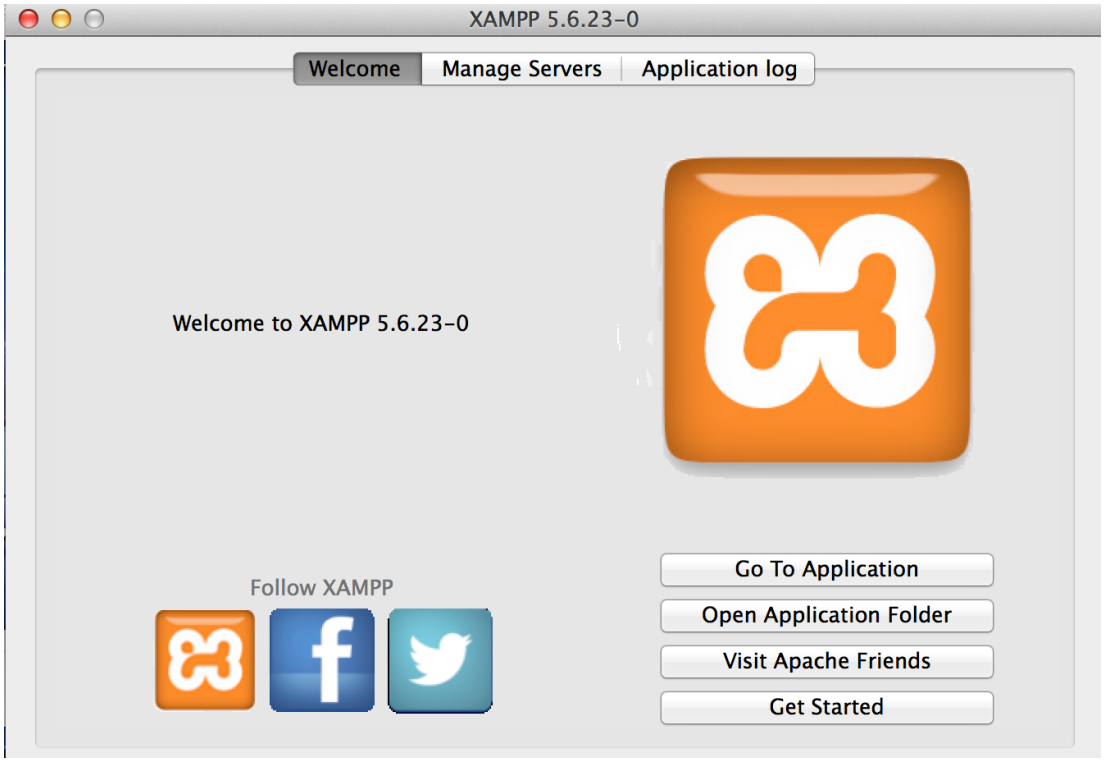


Figura 35, servidor local XAMPP para Mac

CREACION DE LOS SERVICIOS WEB EN JAVA NETBEANS O EN ALGUN EDITOR DE CÓDIGO

Seleccionamos crear un nuevo proyecto, le damos click en php, php aplication y en la parte superior el nombre del proyecto (Figura 36).

Así mismo escogemos la carpeta y la ubicación donde se guardará el proyecto, en el caso del servidor local los archivos php se guardarán en la carpeta www de la carpeta Wamp (Figura 37).

Para Mac los archivos se guardan en la carpeta htdocs de MAMP o XAMP (Figura 38).

En su caso si se trata de un servidor en la nube, los archivos se guardaran en una carpeta especificada por el programador.

Cabe señalar que si se ha cambiado el puerto al 82 debemos especificar la ruta agregando 82 a la URL de *localhost* (figura 39).

En nuestro caso el proyecto fue realizado en MAC con el servidor XAMP y no fue necesario cambio alguno.

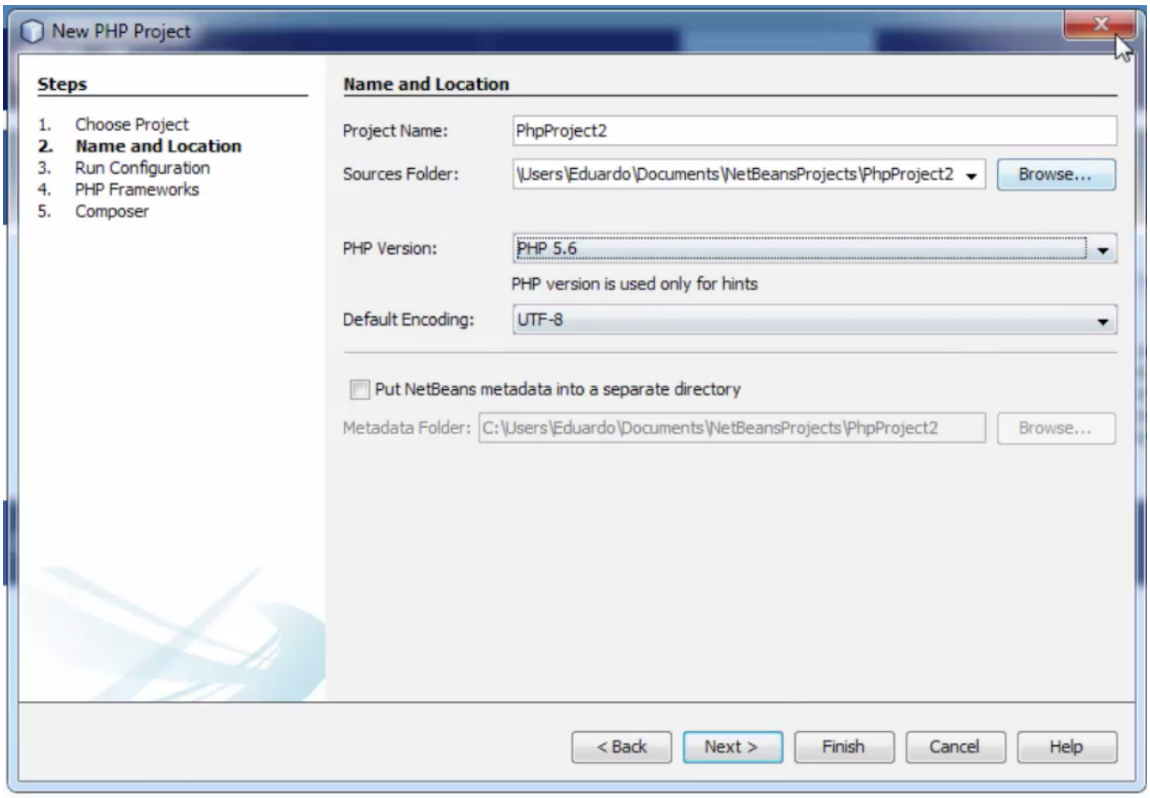


Figura 36. Creación del proyecto php para los servicios web en JavaNetbeans

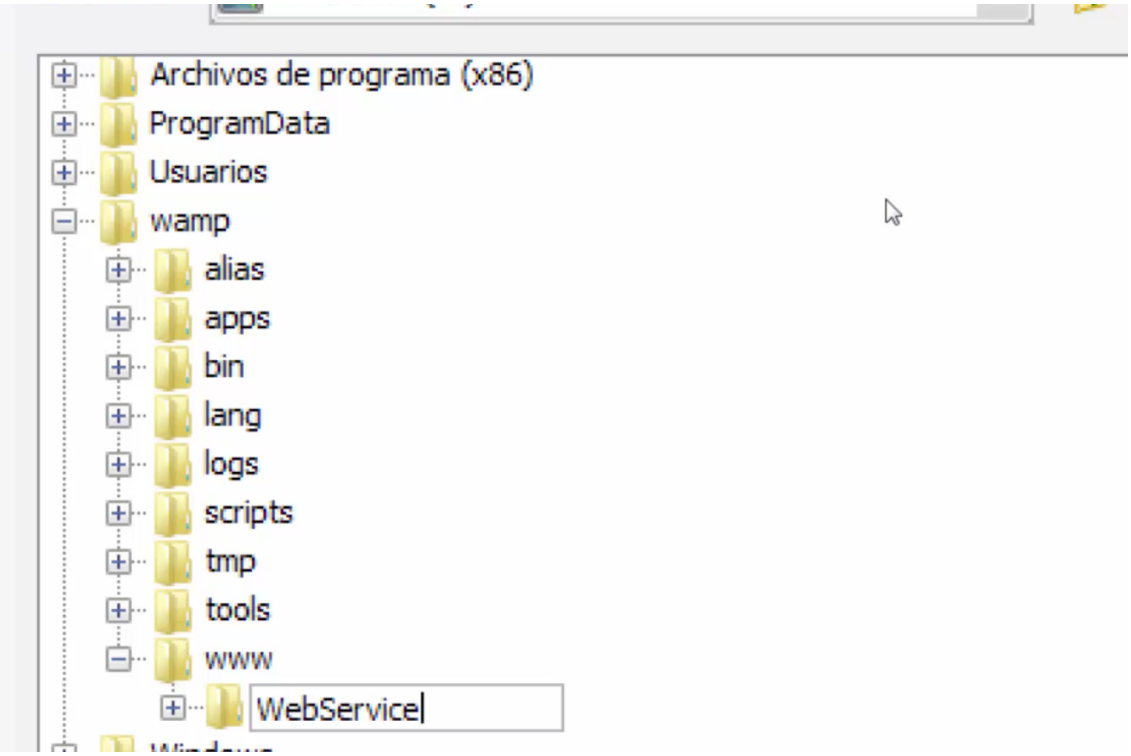


Figura 37. Archivos de los servicios Web y su ubicación en el servidor local

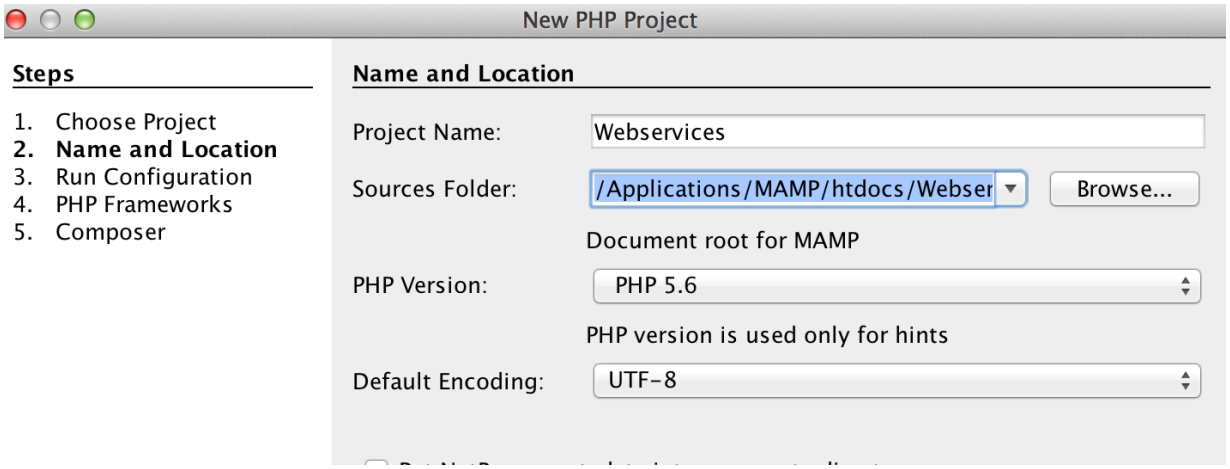


Figura 38. Ubicación de los archivos PHP de los servicios web en Mac

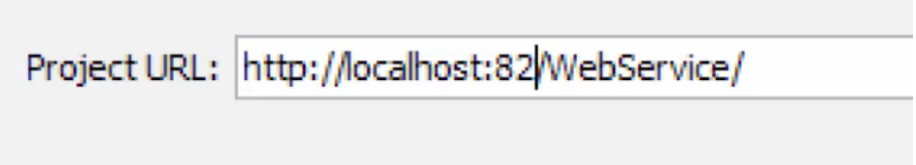


Figura 39. Puerto82 para servidor WAMP

8.- RESULTADOS Y ANALISIS

A continuación se muestran algunos resultados del proyecto, se hicieron pruebas en el emulador de Android Studio.

8.1 Autenticación

Primeramente se agregaron datos a la base de datos directamente para hacer pruebas de autenticación .

Se muestran muestran algunos ejemplos (figura 40).

8.2 Nuevo Registro.

Se ingresa un nuevo usuario desde la pantalla principal en “Nuevo Registro “y verificamos que se registran sus datos en la base MySQL (figura 41).

8.3 Historial de Visitas

Verificamos el historial de visitas de un visitante registrado en el sistema, algunas visitas han sido registradas directamente en la base de datos con fines de prueba.

Ingresamos a la aplicación, ingresamos con el visitante registrado y verificamos que efectivamente las fechas de las citas registradas en la base de datos se muestran en la aplicación móvil (Figura 42).

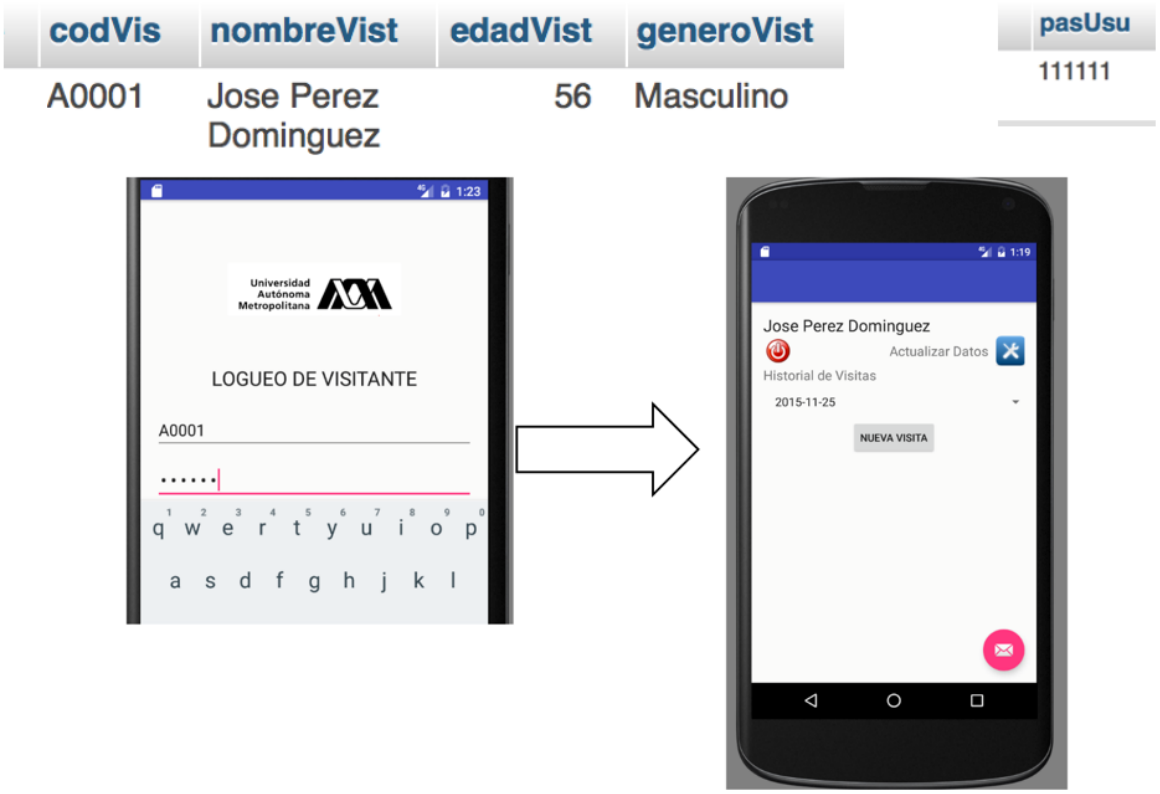


Figura 40. Ejemplo de visitante registrado en la base de datos y verificación en el sistema

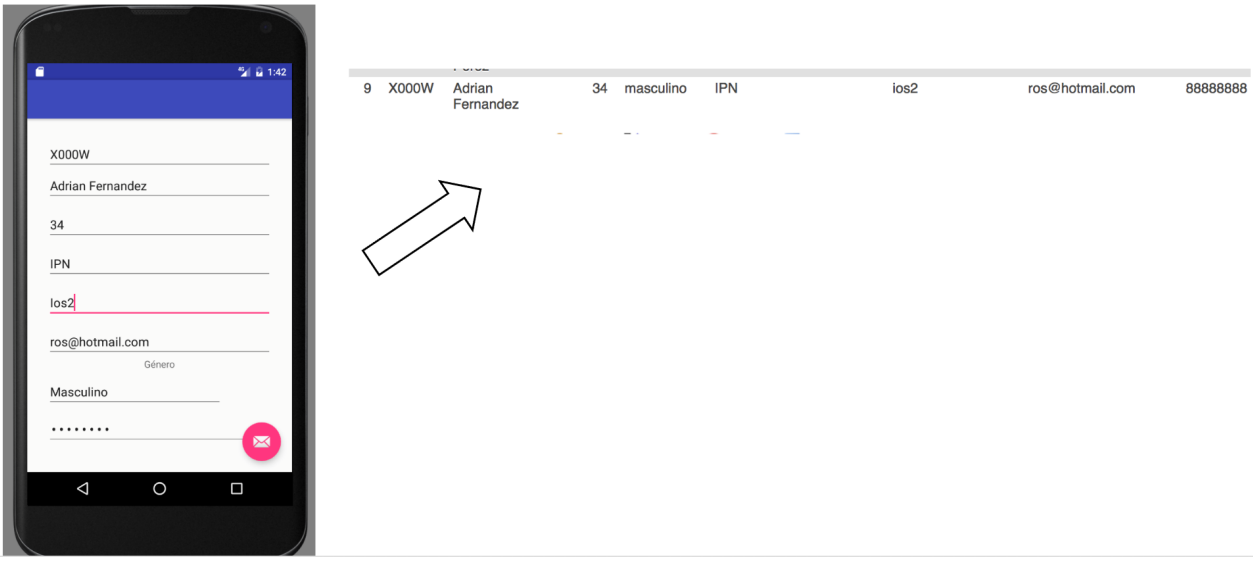


Figura 41. Registro de un nuevo usuario desde la aplicación móvil

	id_visita	codVis	fecha_visita	hora_entrada	hora_salida	puerta_entrada	motivo_visita
r Borrar	1	A0001	2015-11-25	09:00:00	09:10:00	3	
r Borrar	2	A0001	2017-01-02	13:00:00	14:00:00	3	ff
r Borrar	3	A0001	2015-11-25	09:00:00	09:10:00	3	Personal
r Borrar	4	A0001	2015-11-25	09:00:00	09:10:00	3	Conferencia UAM auditorio
r Borrar	5	A0002	2016-08-01	12:30:35	14:43:27	2	Preguntar Cursos de idiomas
r Borrar	6	A0001	2015-11-25	09:00:00	09:10:00	3	Libreria
r Borrar	7	A0005	2016-10-04	15:24:29	16:00:00	1	personal
r Borrar	8	A0001	2015-11-25	00:00:00	00:00:00	0	
r Borrar	9	A0001	2019-01-02	14:00:00	14:00:00	3	ff
r Borrar	13	A0001	2056-09-09	23:54:00	24:00:00	3	xxxxx

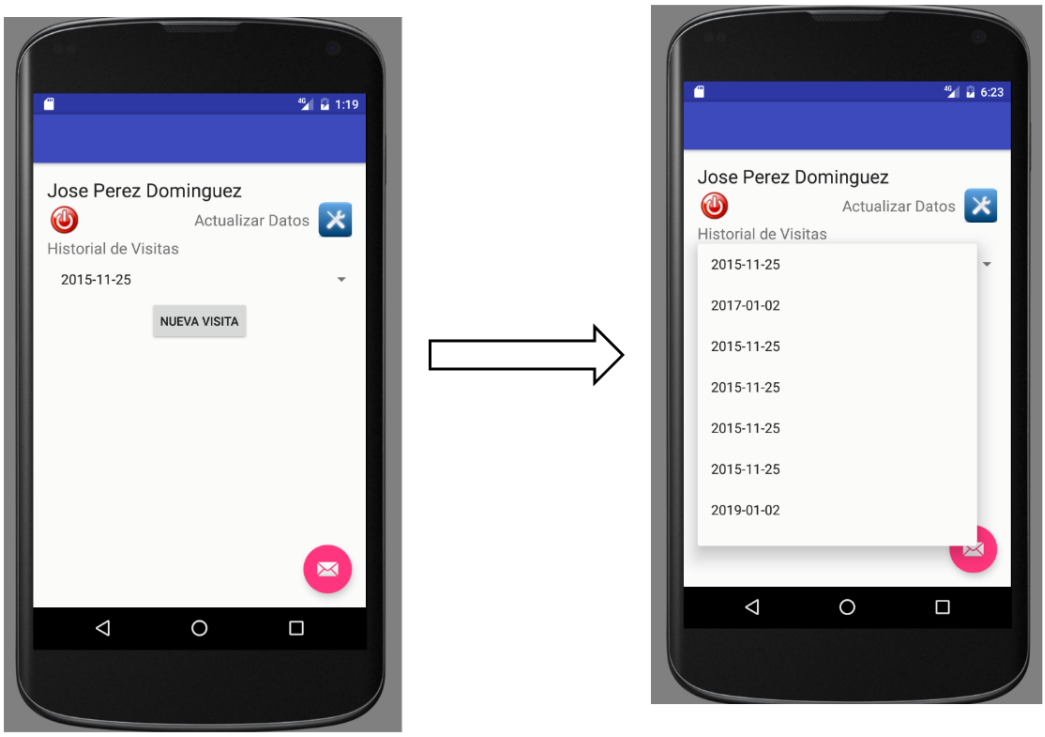


Figura 42. Resultado del historial de visitas de un usuario registrado

8.4 Registro de nueva visita

Una vez que el visitante ha ingresado, puede registrar una nueva visita al sistema.

Se ingresa al sistema y desde la plataforma se registra una nueva visita, se verifica que se ha registrado en la base de datos (Figura 43).

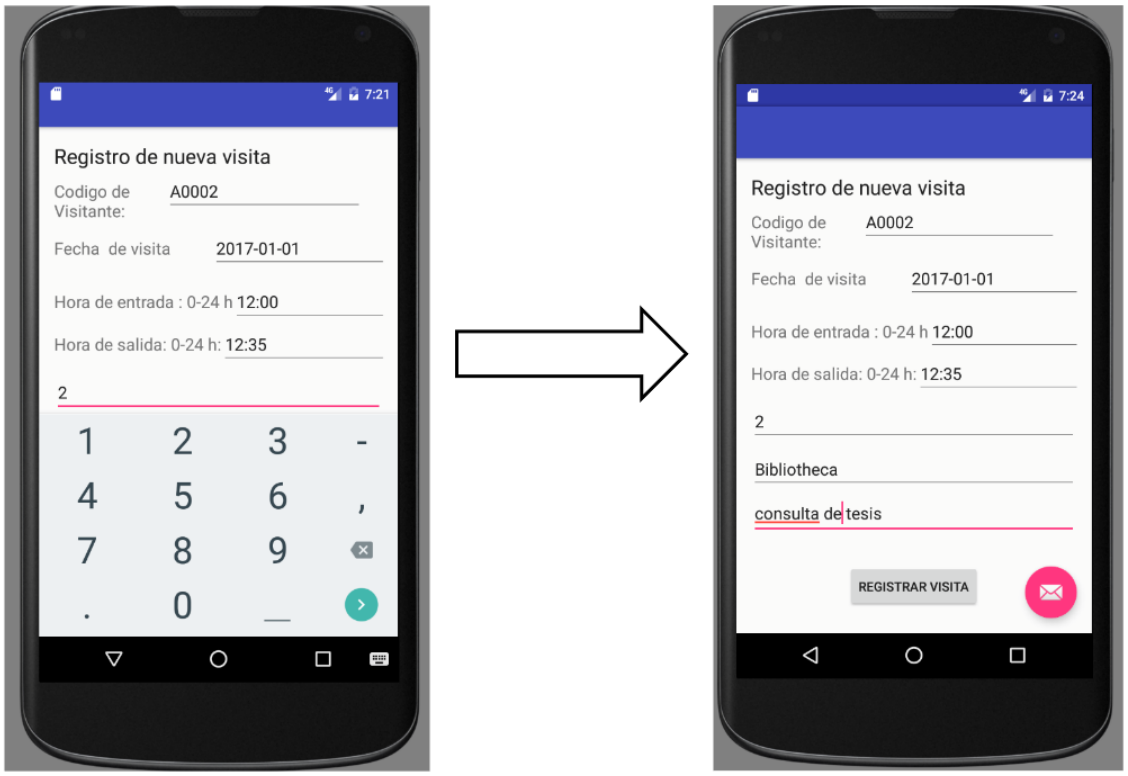


Figura 43. Registro de una nueva visita al sistema.

8.5 Configuración

El visitante puede configurar o actualizar sus datos desde la aplicación móvil. Como por ejemplo modificaremos un usuario registrado. Se verifica antes y después de hacer la modificación de datos a través de la aplicación en la base de datos (figura 44)

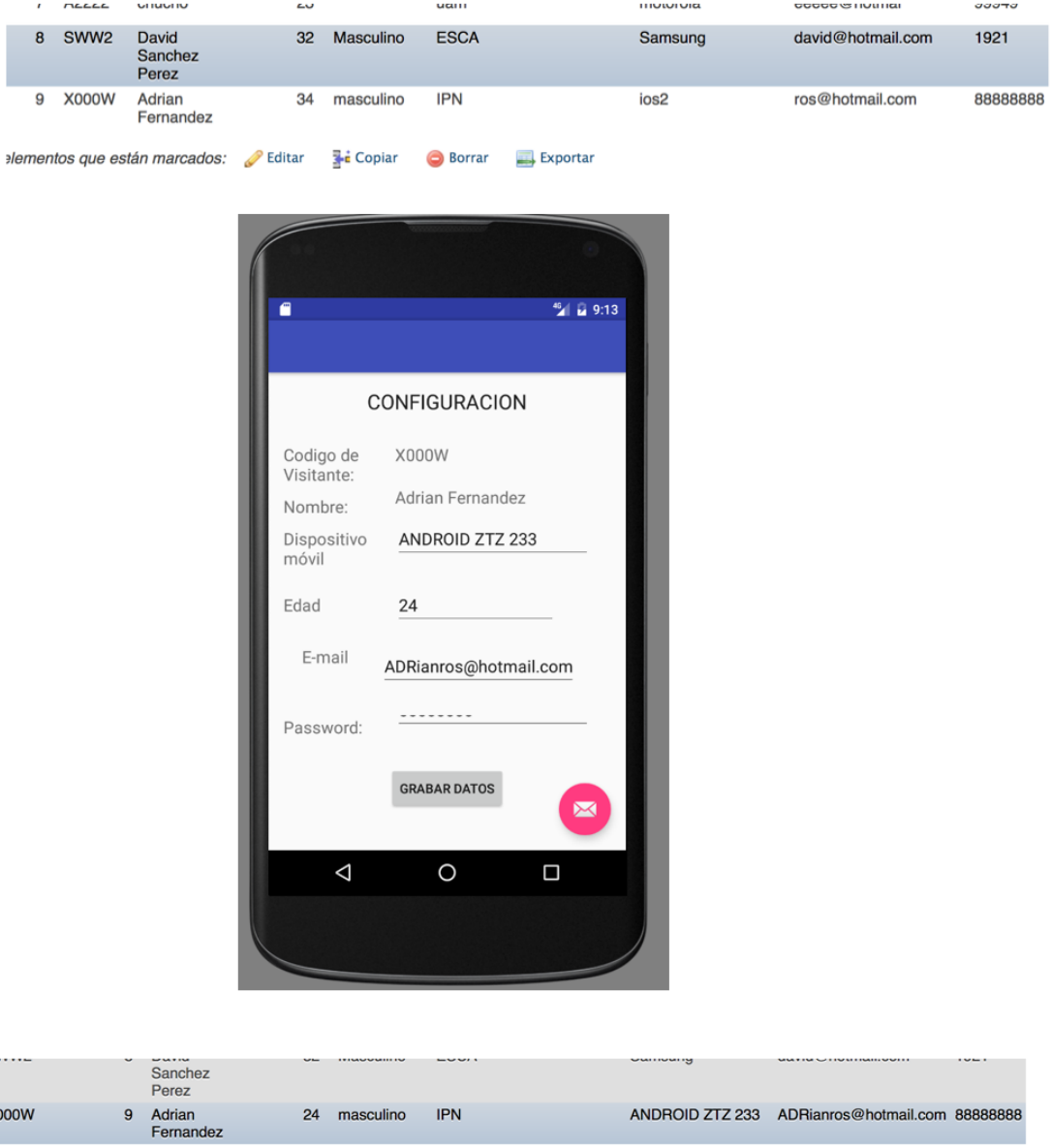


Figura 44. Configuración de datos vía la aplicación móvil.

De acuerdo a los visto anteriormente, vemos que se pueden registrar, modificar o actualizar los datos requeridos de los visitantes en cada uno de los casos señalados.

Con estos resultados obtenidos posteriormente podremos crear una gran base de datos de visitantes a la UAM Azcapotzalco.

De acuerdo a los resultados vemos que es posible consumir un servicio web tipo *RESTful* con una aplicación móvil vía el objeto *HTTPURLConnection*

De acuerdo a los resultados. Se puede decir que REST es un conjunto de principios, o maneras de hacer las cosas, que define la interacción entre distintos componentes, y se cumple en una arquitectura híbrida con diferentes tecnologías.

Con los servicios web creados en PHP vemos que podemos devolver un objeto JSON con los datos requeridos y como este se consumen mediante la aplicación móvil.

Se optó utilizar la clase HttpURLConnection y se aseveró que fue la mejor opción. Su sencilla API hacen que sea ideal para Android.

Hemos visto en cada uno de los caso que es posible utilizar este objeto para la manipulación de datos , su envío o consulta desde un servidor.

9.-CONCLUSIONES

Los logros alcanzados ah sido sustanciosos, se han tocado varios temas durante el desarrollo del proyecto , hemos visto la interacción del lenguajePHP en la creación de servicios web y como este nos devuelve un objeto JSON y como se consumen esos objetos desde Android Studio y como hacer la conexión a un servidor desde Android Studio.

Se hace una demostración de los métodos GET y POST para el envío de datos desde Android hacia una base de datos.

Cabe señalar que la arquitectura puede ser mejorada ya sea visualmente o agregando mas campos al registro de nueva visita, así el visitante podría ingresar su opinión sobre su visita.

Los datos recabados en la base de datos podrían ser utilizados para generar un portal o un sistema de información para crear una gestión y análisis de resultados, opiniones , elaborar un perfil de visitantes o bien ayudar a mejorar los servicios y expectativas de cada visitante.

10.- ANEXOS

A continuación de muestran algunos códigos de métodos del proyecto

10.1 Autenticación de usuario registrado

```
public String enviarDatosGET(String usu, String pas){//funcion que permite envia datos para la
coneccion a la bd
   URL url=null;
   String linea="";
   int respuesta=0;
   StringBuilder resul=null;//recibe la data base

   try{
      url=new URL("http://192.168.1.65/proyecto/validar.php?usu="+usu+"&pas="+pas);//
parametros del usiario que entra
```

```
      // url=new URL("http://192.168.1.65/WebService/valida.php?usu="+usu+"&pas="+pas);-->
aqui
      HttpURLConnection conection=(HttpURLConnection)url.openConnection();//
      respuesta=conection.getResponseCode();

      resul=new StringBuilder();

      if(respuesta==HttpURLConnection.HTTP_OK){//si hay respuesta se consume el Jason de la
respuesta
         InputStream in=new BufferedInputStream(conection.getInputStream());
         BufferedReader reader=new BufferedReader(new InputStreamReader(in));//se lee la
respuesta que se ha traido

         while((linea=reader.readLine())!=null){
            resul.append(linea);//se agregan cada una de las lineas
         }
      }

   }catch (Exception e){}

   return resul.toString();

}
```

Evento onClick

```
 @Override
   public void onClick(View v) {//Enviamos usuario y pasword

      Thread tr=new Thread(){//implementamos un hilo para poder trabajar con el web service
         @Override
         public void run() {
            final String resultado=enviarDatosGET(txtUsu.getText().toString(),
txtPas.getText().toString());
            runOnUiThread(new Runnable() {//permite trabajar con la intefaz grafica desde el hilo
               @Override
               public void run() {
                  int r = obtDatosJSON(resultado);
                   Toast.makeText(getApplicationContext(), r+"", Toast.LENGTH_LONG).show();

                   if (r > 0) {// existe usuario
                      Intent i = new Intent(getApplicationContext(), regitroNotas.class);
                      i.putExtra("cod", txtUsu.getText().toString());
                      startActivity(i);
                   } else {
                      Toast.makeText(getApplicationContext(), "Usuario o Pasword Incorrectos",
Toast.LENGTH_LONG).show();
                   }
```

```
                }
            });

        }
    };
    tr.start();
}
```

Método para saber si el objeto JSON tiene datos que devolver

```
public int obtDatosJSON(String response){// devuelve 0 o 1  recibe una cadena del método anterior
    int res=0;
        try{
            JSONArray json=new JSONArray(response);
            if(json.length()>0){//si la longitud >0 hay  datos
                res=1;
            }
        }catch(Exception e){}
    return res;
}
```

10.2 Métodos para registrar a un visitante al sistema

```
public void registrarvisitante( String c, String n, String e, String g, String p, String d, String m, String w) {
    String urlParameters="cod="+c+"&nom="+n+"&eda="+e+"&gen="+g+"&pro="+p+"&dis="+d+"&mal="+m+"&pas="+w;
    HttpURLConnection conection=null;
    try{
        URL url=new URL("http://192.168.1.65/proyecto/nuevo_registro.php");
        conection=(HttpURLConnection)url.openConnection();

        //estableciendo el metodo
        conection.setRequestMethod("POST");

        //longitud de datos que se envian por el método post
        conection.setRequestProperty("Content-Length", "" + Integer.toString(urlParameters.getBytes().length));

        //comando para la salida de datos
```

```
        conection.setDoOutput(true);

        DataOutputStream wr=new DataOutputStream(conection.getOutputStream());
        wr.writeBytes(urlParameters);
        wr.close();

        InputStream is=conection.getInputStream();


    }catch (Exception ex){}


}
```

Método para el botón de registro de un visitante

```
 btnRegistrar.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
             Thread tr3=new Thread(){
                @Override
                public void run() {

registrarvisitante(txtCod.getText().toString(),txtNom.getText().toString(),txtedadvisitante.ge
tText().toString(),txtgen.getText().toString(),txtprocedencia.getText().toString(),
                        txtdispositivo.getText().toString(),txtmail.getText().toString(),
txtpassvisitante.getText().toString());
                    runOnUiThread(new Runnable() {
                       @Override
                       public void run() {
                          Toast.makeText(getApplicationContext(),"Datos Registrados",
                                Toast.LENGTH_SHORT).show();
                          Intent i=new Intent(getApplicationContext(),MainActivity.class);
                          i.putExtra("cod",recuperado1);
                          startActivity(i);
                       }
                    });
                }
             };
             tr3.start();
          }
       });
```

10.3 Método que permite mostrar el nombre de un visitante que ingresa al sistema

```
public void Mostrarvisitante(String response){
    try{
        JSONArray json=new JSONArray(response);
        for(int i=0;i<json.length();i++){
            txtVisitante.setText(json.getJSONObject(i).getString("nombreVist"));

        }
    }catch(Exception e){}
}
```

10.4 Métodos para cargar historial de visitas

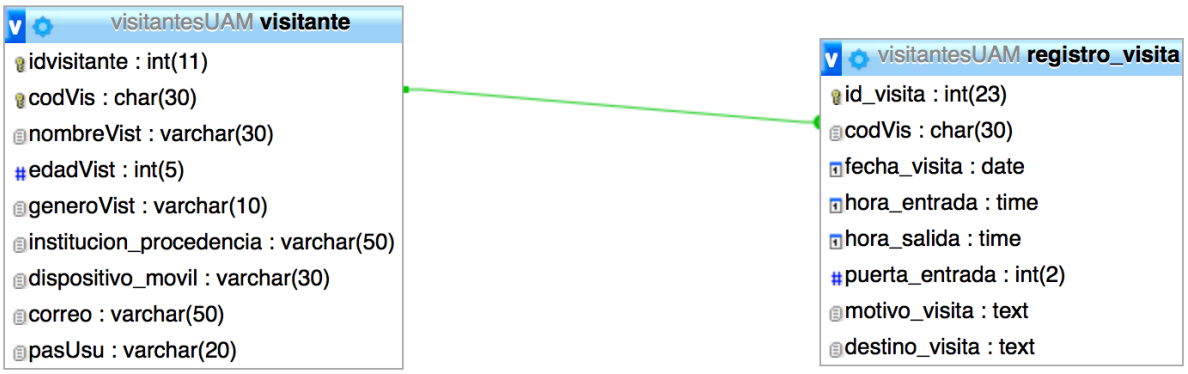
```
//METODO QUE PERMITE CREAR UN ARRAYLIST CON LAS VISITAS
public ArrayList<String> ArregloSpiner(String response){
    ArrayList<String> listado=new ArrayList<String>();
    try{
        JSONArray json=new JSONArray(response);
        String texto="";
        for(int i=0; i<json.length();i++){
            texto=json.getJSONObject(i).getString("fecha_visita");
            listado.add(texto);
        }
    }catch(Exception e){}
    return listado;
}

//METODO QUE PERMITE CREAR UN ARRAYLIST CON LA LISTA

//METODO QUE PERMITE CARGAR EL SPINNER
public void cargarSpiner(ArrayList<String> datos){
    ArrayAdapter<String> adaptador=new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,datos);
    fechas_visitas=(Spinner)findViewById(R.id.splista);

    fechas_visitas.setAdapter(adaptador);
}



    //METODO QUE PERMITE CARGAR LISTVIEW
  public void cargarListView(ArrayList<String> datos){
```

```
      ArrayAdapter<String> adaptador=new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,datos);
      lstVisitas=(ListView)findViewById(R.id.lista);
       lstVisitas.setAdapter(adaptador);
   }
```

10.5 Método para registrar una nueva visita al sistema

```
public void registrar_visita( String x, String f, String e, String s, String p, String d,
String m) {
       String urlParameters="cod="+x+"&fec="+f+"&etr="+e+"&sal="+s+"&pue="+p
+"&des="+d+"&mot="+m;
       HttpURLConnection conection=null;
       try{
          URL url=new URL("http://192.168.1.65/proyecto/nueva_visita.php");
          conection=(HttpURLConnection)url.openConnection();

          //estableciendo el metodo
          conection.setRequestMethod("POST");

          //longitud de datos que se envian
          conection.setRequestProperty("Content-Length", "" +
Integer.toString(urlParameters.getBytes().length));

          //comando para la salida de datos
          conection.setDoOutput(true);

          DataOutputStream wr=new DataOutputStream(conection.getOutputStream());
          wr.writeBytes(urlParameters);
          wr.close();

          InputStream is=conection.getInputStream();


       }catch (Exception ex){}

   }

}
```

10.6 Modelo relacional de la base de datos del sistema



11.- REFERENCIAS BIBLIOGRÁFICAS

[1] Castillo-Franco Nancy “Implementación de un sistema web para dar seguimiento a las visitas realizadas a entidades financieras por parte de la Comisión Nacional Bancaria y de Valores” . Proyecto Terminal, División de CBI, Universidad Autónoma Metropolitana Azcapotzalco. México, 2014.

[2] Luviano-Zaldivar Alejandro“Aplicación web para la visualización y análisis de datos extraídos del sistema ODBII para la gestión de una flota de camiones?. Proyecto Terminal, División de CBI, Universidad Autónoma Metropolitana Azcapotzalco. México, 2014.

[3] Sampere-Romero Edgar Rodrigo, Becerrir Rodriguez Edgar “Sistema de gestión escolar en Android” .Proyecto Terminal, División de CBI, Universidad Autónoma Metropolitana Azcapotzalco. México, 2013.

[4] “Research and implementation of Web Services in Android network communication framework Volley” publicado en Service Systems and Ser- vice Management (ICSSSM), 2014 11th International Conference on. http://ieeexplore.ieee.org/xpl/login.jsp?tp=arnumber=6943373url=http

[5] Narayana- Srirama,Satish, Paniagua Carlos “Mobile Web Service Provisioning and Discovery in Android Days” Mobile Cloud Lab, Institute of Computer Science, University of Tartu Tartu, Estonia 2014. http://kodu.ut.ee/ srirama/publications/ms13.pdf

[6] Leiva-Mundaca Ignacio, Villalobos-Abarca Marco “Método ágil híbrido para desarrollar software en dispositivos móviles” Área de Ingeniería en Computación e Informática, Escuela Universitaria de Ingeniería Industrial, Informática y de Sis- temas. Universidad de Tarapacá Chile 2015.

[7] Garcia-Zabala Jessica Xchel, Martinez-Vasquez Daniel, Rivera-Corona Dante “Desarrollo de un directorio usando un servicio web” Tesis para obtener el título de ingeniero en comunicaciones y electrónica . Escuela Superior de Ingeniería Mecánica y Eléctrica unidad Zacatenco. México 2009.

[8] Espinosa-Ureña Jorge “Sistema de gestión automatizada de personal para la comisión de vialidad y transporte urbano” Tesis para obtener el título de licenciatura en computación División de CBI, Universidad Autónoma Metropolitana Iztapalapa . México, 1992.

[9] Yevheniy Dzezhyts “Android Application Development” Thesis Business Information Technology Haaga-Helia University Finlandia 2013.

[10] http://es.ccm.net/contents/264-el-protocolo-http

[11] Rafael Navarro Marset. ELP-DSIC-UPV, Modelado, Diseño e Implementación de Servicios Web 2006-07 http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf

[12] https://developer.android.com/reference/java/net/HttpURLConnection.html

[13] http://php.net/manual/es/intro.pdo.php

[14] http://docs.oracle.com/javase/tutorial/java/data/buffers.html

[15] Craig Larman (2002). UML y patrones : introducción al análisis y diseño orientado a objetos y al proceso unificado. 2da. edición. Ed. Prentice Hall.