Universidad Autónoma Metropolitana

Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Sistema para facilitar asesorías académicas con aplicación móvil

Proyecto Tecnológico

Alumno: Adamo Jordán Figueroa Pérez

2113001882

Jordan.figu@gmail.com

Asesor: Dr. José Alejandro Reyes Ortiz

jaro@correo.azc.uam.mx

Trimestre: 2016 Otoño

Fecha: 6-ene-2017

Declaratoria

Yo, José Alejandro Reyes Ortiz, 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.

Yo, Adamo Jordán Figueroa Pérez, 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.

I. RESUMEN

El objetivo de este proyecto es crear una aplicación móvil que ayude a mejorar la comunicación entre estudiante y profesor,especialmente a la hora de buscar un profesor para pedir una asesoría, se logra gracias a la programación de citas con anticipación y a la comunicación mediante notificaciones de una manera rápida y ordenada.

El producto final del proyecto es una aplicación móvil; sin embargo es más que eso, implica todo el sistema de información detrás, ya que cuenta con distintos servicios web que ayudan a identificar al usuario, reservar, aceptar o cancelar citas.

Los servicios web se encuentran funcionando en un servidor Apache Tomcat en la dirección

http://aisii.azc.uam.mx:8080//citasUAM

A continuación se muestra la documentación completa del proyecto terminal “Sistema para facilitar asesorías académicas con aplicación móvil”

II. TABLA DE CONTENIDO

III. ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

Tanto en la UAM como en otras universidades los profesores cuentan con horas de asesorías dedicas a resolver dudas e inquietudes de los alumnos, sin embargo, existe un problema, cada día es diferente, y un docente nunca sabe el número de estudiantes que lo buscarán ni el horario en que cada alumno llegará para solicitar ayuda académica. Al mismo tiempo un estudiante no puede asegurarse que el profesor estará libre en el momento en que se asista a su cubículo, tal vez tenga una cola de alumnos esperando, o puede que el profesor ni siquiera se encuentre en su cubículo. El objetivo de este proyecto de integración, es solucionar estos problemas, se logrará con un sistema informático para agendar citas de asesorías, donde a través de una aplicación móvil los profesores y alumnos acordarán la fecha y hora de la cita. En la aplicación, los docentes podrán publicar sus horarios de asesoría, los estudiantes tendrán acceso a estos horarios, podrán verificar la disponibilidad del profesor y agendar una cita en el horario que les sea conveniente, además el profesor podrá confirmar la cita o rechazarla en caso de algún contratiempo y proponer una nueva cita.

2. JUSTIFICACIÓN

En la mayoría de los casos, un alumno busca una asesoría cuando tiene problemas académicos, por ejemplo, no dominar o comprender un tema, que exista algún problema que no pueda resolver, no contar con los métodos de estudio adecuados etc. Cuando se acude a una asesoría sin previo aviso se pueden encontrar inconvenientes, el alumno debe tener la seguridad de ser atendido por su profesor, siendo la única preocupación llegar a tiempo a su cita, y no, si el profesor estará presente, o cuánto tiempo tendrá que esperar para ser atendido. Los celulares se han hecho indispensables para nuestra vida diaria, es un hecho que la mayoría de los estudiantes y docentes cuentan con un teléfono móvil inteligente, por esta razón una aplicación móvil es factible para resolver este problema, tanto estudiantes como docentes recibirán las notificaciones tales como petición, aceptación, rechazo o cancelación de citas, directamente en sus celulares, este es el punto más atractivo, dado que la mayoría de los sistemas actuales son aplicaciones web, y para visualizar un mensaje es necesario autentificarse en el sistema y buscar en la bandeja de mensajes por alguna notificación, en cambio, con la aplicación móvil las notificaciones llegarán en tiempo real sin la necesidad de 2 tener que conectarse a alguna página web, con esto se logrará que la comunicación entre alumno y profesor sea más cómoda y eficiente. Además, aunque no es parte del proyecto existirá la posibilidad de obtener reportes y estadísticas con la información que quedará guardada en la base de datos, se podrá responder preguntas como: ¿Qué profesores son más solicitados? ¿Cuáles es el principal motivo de un alumno para buscar una asesoría? ¿En qué fechas y horarios se solicitan más tutorías? ¿Cuál es la división que requiere más asesorías? Al responder estas preguntas se encontrarán las debilidades y necesidades de los estudiantes, y atacar estos problemas ayudará a mejorar la calidad de la educación.

3. ANTECEDENTES

Proyectos de integración internos

Aplicación móvil para la recomendación de productos en el comercio electrónico [1]

El proyecto propone una aplicación móvil que predice el grado de preferencia sobre un artículo, explotando la información que conoce del usuario. Es similar al proyecto propuesto en cuanto a la arquitectura, ya que para dar sustento a la aplicación utiliza una capa de servicios web, donde se hacen altas, bajas, modificaciones y consultas de los datos, difiere en el enfoque, ya que este proyecto recomienda productos y va orientado al comercio.

Prototipo de una aplicación móvil para la gestión de síntomas de un paciente. [2]

Este proyecto es un gestor de síntomas que intenta dar una sugerencia del tipo de enfermedad que se padece, utiliza una aplicación móvil que registra síntomas de enfermedades, se piden además datos adicionales como la fecha, hora, intensidad del síntoma y comentarios, todo con el fin de dar un diagnóstico más preciso.

Es similar al proyecto propuesto en cuanto a la arquitectura ya que cuenta con una aplicación móvil que se comunica con un servidor de servicios web, y desde estos servicios se hacen consultas a la base de datos, difiere en cuanto al enfoque, dado que este proyecto está orientado al campo de la medicina.

Tesis

Desarrollo de un sistema de gestión de tutorías a través de una aplicación móvil [3]

El proyecto propone un sistema de gestión de tutorías para la universidad Politécnica de Madrid, a través de una aplicación móvil, en la cual los alumnos pueden reservar una tutoría y agregar un motivo, además podrán conocer la disponibilidad del profesor.

Este proyecto es bastante similar al proyecto propuesto, difiere en el público, ya que el proyecto propuesto está dirigido a los alumnos de la Universidad Autónoma Metropolitana, también en la arquitectura del sistema y las tecnologías a usar para el desarrollo.

Análisis, diseño e implementación del sistema de seguimiento, evaluación y control de las tutorías de tesis para las direcciones de carrera de la facultad de ingeniería ciencias físicas y matemática [4]

El objetivo del proyecto es desarrollar un sistema para la gestión de tutorías de tesis, el sistema asigna profesores a los alumnos para que sean tutores de sus tesis, además de equilibrar la carga de trabajo entre profesores, de esta forma cada docente no tendrá exceso de trabajo o muy poco, el sistema también genera reportes y estadísticas con la información de las tesis, alumnos egresados y tesis encargadas cada profesor.

La principal diferencia es la orientación del sistema, este está enfocado especialmente a tutorías para tesis, y en la asignación de tutores a las tesis convenientes, además no es un sistema a base de servicios web.

Artículos

Desarrollo de aplicación web para tutorías académicas, incorporando reingeniería de procesos, programación concurrente y sistemas de gestión de bases de datos distribuidas [5]

El artículo propone un sistema de información que involucra a tutores y tutorados, el objetivo es que los tutores tengan un mejor control de sus tutorados, cada tutor podrá ver los datos de los alumnos tutorados, información como diagnósticos académicos, rendimiento etc. Así los tutores pueden ayudar a sus tutorados detectando debilidades y necesidades. Los tutorados podrán comunicarse directamente con su asesor para pedir ayuda.

La principal diferencia es que el sistema está dirigido a alumnos y profesores que sean oficialmente tutores y tutorados, es decir un alumno tutorado solo puede pedir ayuda de su tutor, y de igual manera un tutor sólo puede ver la información de sus tutorados.

4. OBJETIVO GENERAL

Diseñar e implementar una aplicación móvil que gestione citas para asesorías académicas.

5. OBJETIVOS ESPECÍFICOS

• Analizar y proponer las reglas de consenso entre el solicitante y el profesor y condiciones para hacer posible el funcionamiento del sistema.

• Diseñar y crear una base de datos utilizando las reglas encontradas.

• Diseñar e implementar una capa de servicios web para el almacenamiento y consultas a la base de datos.

• Diseñar la interfaz gráfica de la aplicación móvil.

• Implementar la aplicación móvil basada en los servicios web para la gestión de citas de asesoría.

6. MARCO TEÓRICO

Android.

Es el sistema Operativo más popular para smartphones hoy en día, las solicitudes se han escrito utilizando el lenguaje de programación Java y se ejecutan en una máquina virtual personalizada que se ejecuta en la parte superior de un núcleo de Linux, actualmente está en la versión 7.1.

Servicios web REST.

Es una arquitectura para implementar servicios web, su acrónimo significa REpresentational State Transfer su traducción sería transferencia de representación de estado, lo realmente significa es, un servicio que no tiene estado, es decir no guarda sesión o algún tipo de información en cada llamada, simplemente entrega la información al solicitante, además utiliza los métodos HTTP de manera explícita, algunas ventajas son, optimización de recursos del sistema, gran escalabilidad y facilidad de implementación.

Apache Cordova

Cordova es un framework de desarrollo de aplicaciones móviles, que permite usar tecnologías web y es multiplataforma, tiene muchas APIs para acceder a componentes nativos del sistema operativo móvil, por medio de funciones JavaScript, como el acelerómetro, cámara geolocalización y otros sensores y funciones nativas del sistema.

Angular

Angular es un framework de desarrollo de aplicaciones web y móviles desarrollado por Google, el lenguaje de programación utilizado es TypeScript.

Angular propone una arquitectura modular a base de web components, donde cada componente tiene una plantilla que es la interfaz gráfica que verá el usuario y su código de funcionamiento, estos pueden estar en archivos diferentes, y están ligados con un “doble bingding” significa que cambiar una variable en la vista hará que se vea reflejado automáticamente en el componente, y de forma viceversa también.

Se muestra la arquitectura general del funcionamiento de angular en la figura 1.

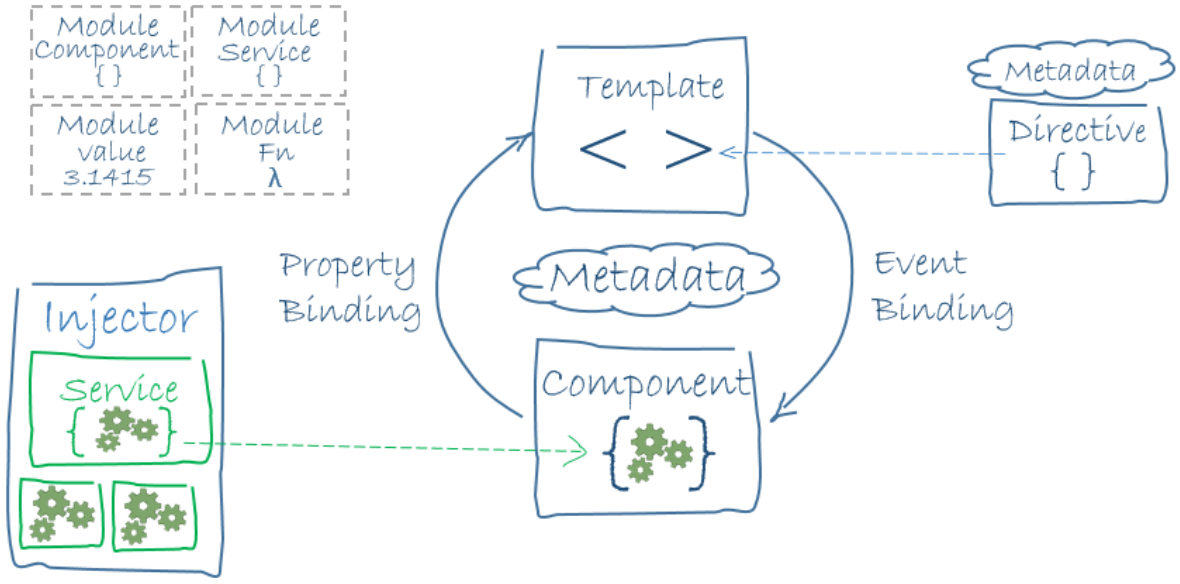

Figura 1. Arquitectura general de funcionamiento de Angular.

Ionic

Ionic es un framework para desarrollar aplicaciones móviles híbridas, esto significa que usa tecnologías web como HTML CSS y JS las cuales pueden comunicarse con funciones nativas del Sistema Operativo, con Ionic es posible desarrollar y generar la misma aplicación en distintas plataformas como son Android IOS y Windows Phone.

El funcionamiento de Ionic se basa en la utilización de 2 frameworks muy poderosos Angular y Apache Cordova, básicamente toma completamente la arquitectura de Angular y las Apis de Cordova para compilar las aplicaciones en los distintos S.O. Además agrega sus propias etiquetas HTML para facilitar el desarrollo front end de cada interfaz gráfica, cabe mencionar que el lenguaje de programación es TypeScript.[6]

TypeScript

Es un superconjunto de JavaScript que esencialmente agrega tipado estático y objetos basados en clases al lenguaje, gracias a esto se tiene un lenguaje orientado a objetos y por lo tanto es posible aplicar patrones de diseño.

HTTP (HiperText Transfer Protocol)

Es un protocolo para la comunicación de un cliente y un servidor a través de Internet.

Existen además diferentes métodos utilizados para enviar o pedir información los más importantes y sus convenciones de uso son los siguientes.
GET : para solicitar un recurso.
POST : para enviar información que comúnmente será guardada en una base de datos.
PUT: para solicitar una actualización en la información ya guardada.
DELETE : para eliminar contenido o información.

JSON (JavaScript Object Notation**)**

Es un formato ligero de intercambio de datos, es el formato comúnmente utilizado para enviar y recibir información en servicios web REST.

Push notification

Las notificaciones push son mensajes que se reciben en el dispositivo y que han sido emitidos desde cualquier punto de un sistema.

Las notificaciones push en Android son emitidas mediante FireBase Cloud Messaging (FCM).

FireBase Cloud Messaging

Antes llamado Google Cloud Messaging (GSM), es el servicio de Google habilitado para el envío de Notificaciones Push a dispositivos Android.

Ionic Cloud

Es plataforma de ionic de la nube que ayuda a facilitar la integración funcionalidades y servicios como notificaciones push.

Para esto se cuenta con una API la cual es intermediaria entre el sistema que genera la notificación y el sistema que envía la notificación FCM o GCM para android o APNs para IOS.

Se muestra en la figura 2 la arquitectura para el envío de notificaciones.

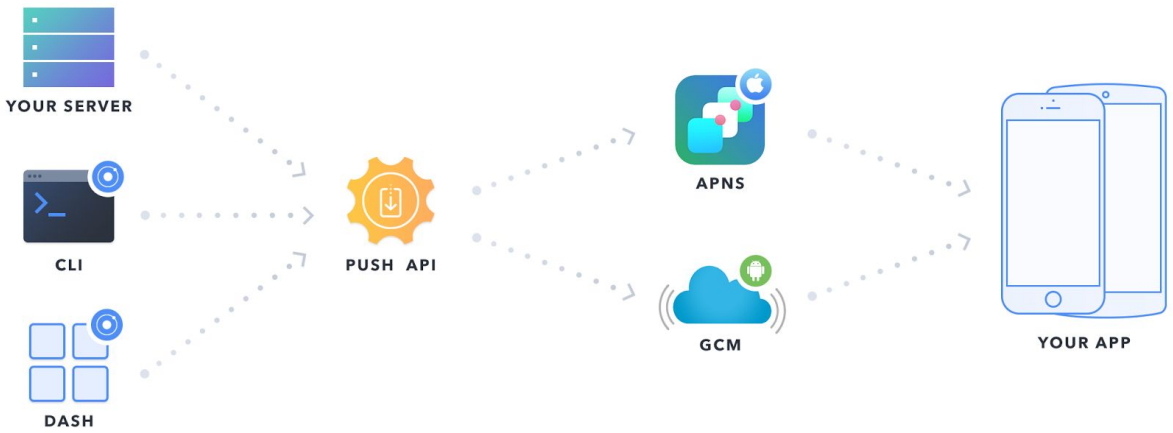

Figura 2. Arquitectura Notificaciones Ionic Cloud.

7. DESARROLLO DEL PROYECTO

7.1 Arquitectura general

Se muestra a continuación en la figura 3 una arquitectura general del sistema desarrollado, la cual consta de una aplicación móvil, un servidor de servicios web, y una base de datos.

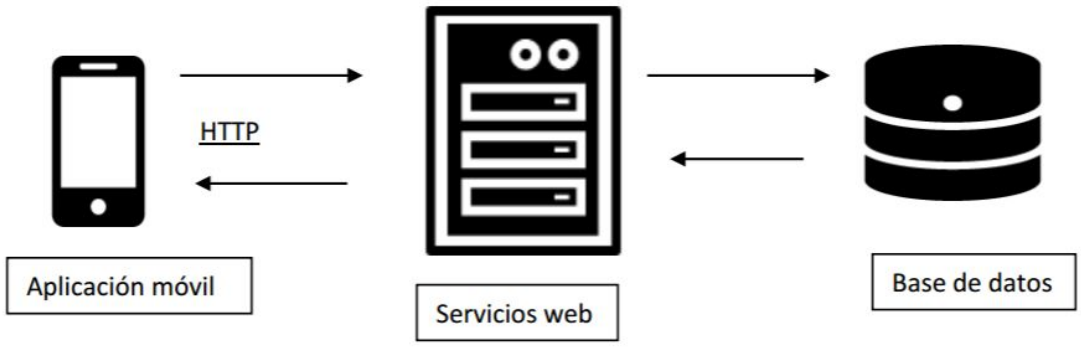

Figura 3. Arquitectura general del sistema.

Para desarrollar el sistema se utilizó una arquitectura basada en 3 capas que se describen a continuación.

7.2 Aplicación móvil.

Se crearon un total de 14 páginas, con las siguientes funcionalidades

- Introducción
- Inicio
- registro general
- registro de estudiante
- registro de profesor

- sesión principal de estudiante
- programación de citas
- calendario de estudiante
- opciones de estudiante

- sesión principal de profesor
- aceptar / cancelar citas
- horario de profesor
- registro de horarios profesor
- opciones de profesor

Se muestra el mapa de navegación en la figura 4.

Figura 4. Mapa de navegación de la aplicación.

Interfaces

Antes de programar las interfaces se diseñaron los mockups con ayuda de adobe illustrator, se muestra el primer boceto en la figura 5.

Figura 5. Interfaces primera versión.

Conforme se fueron programando, se modificó un poco el diseño inicial para hacerlo compatible con las funcionalidades de Ionic, se muestra el diseño final en la figura 6.

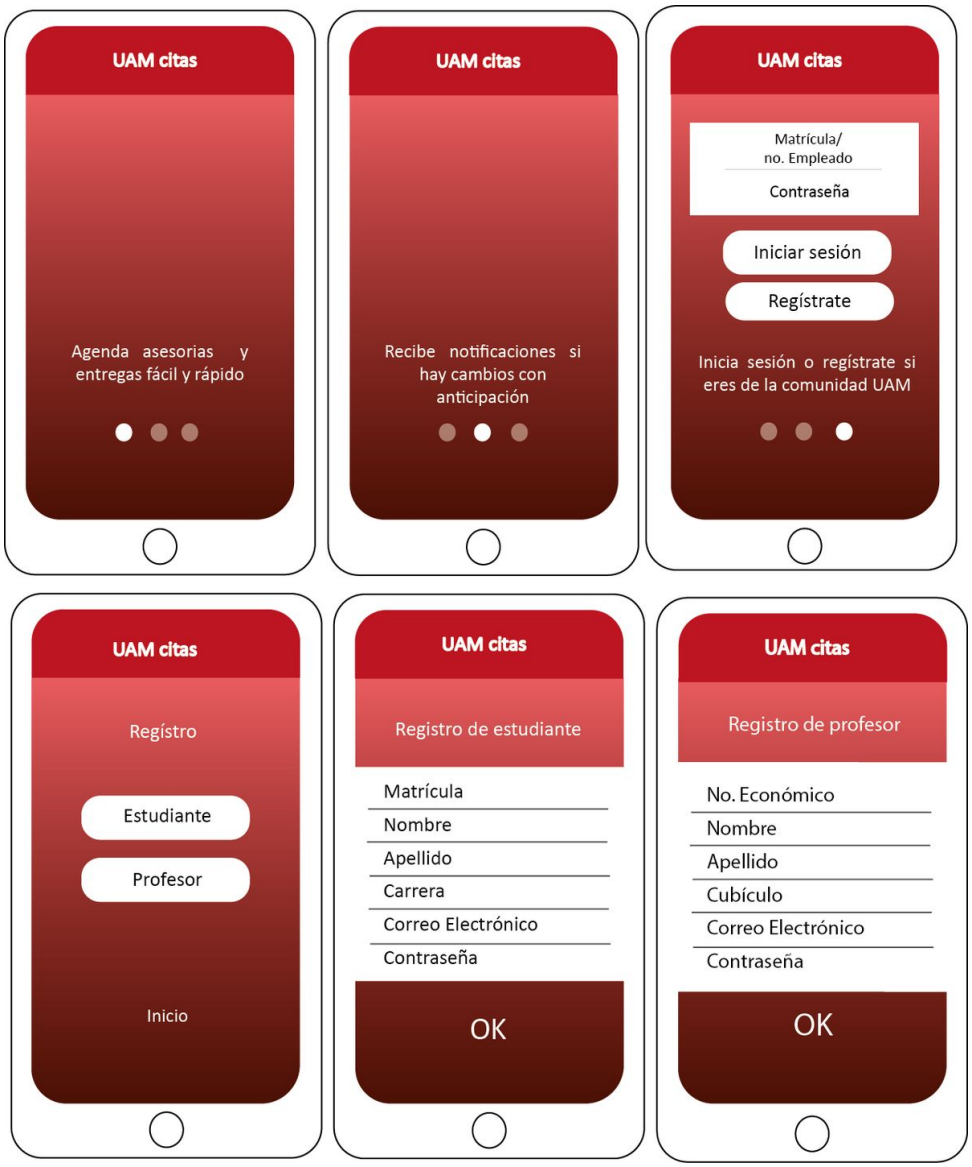

Figura 6. Interfaces segunda versión.

A continuación se explica el código de la página Inicio, se muestra la interfaz en la figura 7.

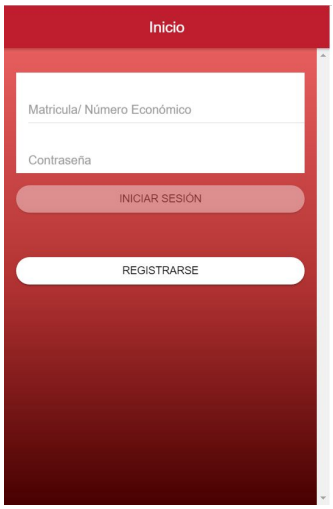

Figura 7. Interfaz de página inicio.

Vista

Header

Con el código mostrado a continuación se crea el header de la aplicación, se muestra el header en la figura 8, las etiquetas ion-header, ion-navbar, ion-title no pertenecen a html , éstas y algunas otras son agregadas por Ionic, el color rojo-uam está definido en una hoja de estilo general en el archivo variables.scss.

```
<ion-header>
  <ion-navbar color="rojo-uam">
    <ion-title text-center >Inicio </ion-title>
  </ion-navbar>
</ion-header>
```

Figura 8. header de la aplicación.

Contenido

```
<ion-content class="degrade-principal" padding >
<br>

<form [formGroup]= "valoresFormulario" (ngSubmit)="iniciarSesion()">
    <ion-list>
    <ion-item>
    <ion-label floating>Matricula/ Número Económico</ion-label>
      <ion-input formControlName="id" type="text" value=""></ion-input>
    </ion-item>
```

```
    <ion-item>
      <ion-label floating>Contraseña</ion-label>
      <ion-input formControlName="contrasena" type="password"></ion-input>
    </ion-item>
  </ion-list>
  <button ion-button type="submit" [disabled]="!valoresFormulario.valid" full round color="blanco"
style="color:black" > Iniciar sesión </button>
</form>
```

Todo el contenido general va dentro de la etiqueta <ion-content>, se tiene un formulario en los atributos de la etiqueta form, (ngSubmit) especifica la función a llamar cuando se presiona el botón "inicia sesión" ésta función está definida en el componente de la página. formControlName es un atributo de angular que liga el contenido del input con en el componente en las variables llamadas id y contrasena.
Se observa también que la etiqueta <button> tiene el siguiente atributo de angular [disabled]="!valoresFormulario.valid" indica que el botón estará deshabilitado hasta que se inserten en los campos valores válidos, se muestra el formulario en la figura 9.

Figura 9. Formulario de inicio de sesión.

```
  <br><br><br>
  <button ion-button full round color="blanco" style="color:black"  (click)="irARegistro()"> Registrarse </button>
</ion-content>
```

Por último se tiene un botón el cual al presionar se activará la función irARegistro() la cual únicamente cambia a la página de registro, se muestra el botón de registro en la figura 10.

Figura 10. Botón de registro.

El código del componente es el siguiente.

```
import { Component } from '@angular/core';
import { NavController, MenuController } from 'ionic-angular';
import { SesionEstudiantePage } from '../sesion-estudiante/sesion-estudiante';
import { SesionProfesorPage} from  "../sesion-profesor/sesion-profesor";
import { RegistroPage} from "../registro/registro";
import {ConexionesREST} from "../../providers/conexiones-rest";
import {StorageProvider} from "../../providers/storage-provider";
import { FormBuilder, Validators } from '@angular/forms';
import {Push,PushToken} from '@ionic/cloud-angular';
```

En el código de arriba se importan los servicios y otras paginas a las que se es posible navegar, las cuales se explican a continuación.
Component es la base de angular convierte la clase en un componente agregando distintos patrones de diseño lo que le permite tener ligado las funcionalidades del código con la vista y otras funcionalidades.
NavController agrega las funciones para navegar entre páginas de la aplicaciones.
MenuController agrega las funciones para activar el menú de la aplicación.
SesionProfesor, SesionEstudiante, y ResgitroPage son páginas de la aplicación, se agregan para indicar que se es posible navegar hacia ellas con alguna función.
ConexionesRest es un clase especial llamada servicio en angular, la cual contiene todos los métodos para consultar los servicios web del sistema.
StorageProvider es un servicio que contiene las funciones para almacenar datos en memoria caché del teléfono.
FormBuilder y Validator son las clases proporcionadas por Angular para crear y ligar el formulario con la vista además de validar que los datos sean correctos.
por último Push,PushToken son clases de ionic cloud para usar su API con la cual es posible recibir notificaciones push en el Móvil.

```
@Component({
  templateUrl: 'inicio.html',
})
export class InicioPage {
  alumnoPrueba: any;
  valoresFormulario: any;//para agrupar los valores de los inputs
  token: string;
```

Se crea la clase InicioPage especificando que es un componente y que su plantilla (la vista) se encuentra en el archivo inicio.html.

```
  constructor(public push: Push,public menuCtrl: MenuController, public navCtrl: NavController, public formBuilder :
FormBuilder, private conexion:  ConexionesREST ,public storageProvider : StorageProvider   ) {
```

Angular está lleno de patrones de diseño de software, usa el patrón Singleton para inyectar una instancia de algunas de clases importadas para utilizar sus métodos, esto se hace en el constructor.

```
   //deshabilita los menus
   this.menuCtrl.enable(false, 'menuProfesor');
   this.menuCtrl.enable(false, 'menuEstudiante');

    //registra el teleforno para las push
     this.push.register().then((t: PushToken) => {
      return this.push.saveToken(t);
    }).then((t: PushToken) => {
      console.log('Token saved:', t.token);
      //se guarda el token guardar en memoria también
      this.token= t.token;
    });
```

```
this.push.rx.notification()
  .subscribe((msg) => {
    alert(msg.title + ': ' + msg.text);
});
```

Para que sea posible recibir una notificación push es necesario obtener un token único de cada móvil, con el código anterior se obtiene el token y se guarda en una variable para después enviar el token al servidor, también se habilita la recepción de notificaciones.

```
//liga el formulario al código
 this.valoresFormulario = this.formBuilder.group({
   id: ["",[Validators.maxLength(10),Validators.required]],
   contrasena : ["",Validators.required],

 })
```

Se crea el formulario, id y contrasena son variables que guardan la información de los campos del formulario en la vista,ambos son necesarios para la validar el formulario además id tiene una restricción a que permite máximo 10 caracteres.

```
  //elimina los datos en cache si había
   storageProvider.clearData("matricula");
   storageProvider.clearData("numeroEconomico");


}


 //conecta con el servidor envia json con id y contraseña recupera json con datos de la sesión
```

La siguiente función se llama al presionar el botón “iniciar sesión” desde la vista, primero crea un objeto json con la matrícula contraseña y token, para después enviarlo al servicio web correspondiente con el método login de conexión, éste está programado en la clase ConexionesRest importadas al inicio del código y básicamente hace una solicitud post a un servicio web para hacer el inicio de sesión.

```
iniciarSesion(){
  var tokenAndroid;
  if(this.token!= null){
    tokenAndroid= this.token;
  }
  var datosSesion = {
    "id": this.valoresFormulario.value.id,
    "contrasena" : this.valoresFormulario.value.contrasena,
    "token" : tokenAndroid",
  };
```

El servicio web puede responder con los datos de un estudiante o un profesor, las sentencias if siguientes verifican que tipo de usuario és ,se guardan los datos en la

memoria caché del móvil, y por último se cambia a la página de la sesión correspondiente, estudiante o profesor.

```
  this.conexion.login(datosSesion).subscribe(data => {
    if(data.matricula!=null){ //guarda datos de alumno
      this.storageProvider.saveData("matricula",data.matricula);
      this.storageProvider.saveData("nombre",data.nombre);
      this.storageProvider.saveData("apellido",data.apellido);
      this.storageProvider.saveData("carrera",data.carrera);
      this.storageProvider.saveData("correo",data.correo);
      this.storageProvider.saveData("urlFoto",data.urlFoto);
      // envia a sesion alumno
      this.navCtrl.setRoot(SesionEstudiantePage,data);
    }
    if(data.numeroEconomico!=null)
      { // guarda datos de profesor
      this.storageProvider.saveData("numeroEconomico",data.numeroEconomico);
      this.storageProvider.saveData("nombre",data.nombre);
      this.storageProvider.saveData("apellido",data.apellido);
      this.storageProvider.saveData("cubiculo",data.cubiculo);
      this.storageProvider.saveData("correo",data.correo);
      this.storageProvider.saveData("urlFoto",data.urlFoto);

          //cambia de pagina y envia datos
      this.navCtrl.setRoot(SesionProfesorPage,data);

    }
```

Los servicios web siempre responden con el mismo formato, si se envía un json con el atributo message significa que hay un error y éste es mostrado a continuación.

```
    if(data.message!=null){
      alert(data.message);
    }
            },
            err => {
                alert("problemas en el servidor");
            });

 }
```

En caso de presionar registrarse, se llamará a la siguiente función la cual usa el sistema de navegación para cambiar de página.

```
 irARegistro(){
   this.navCtrl.push(RegistroPage);
 }


}
```

Para darle originalidad a la aplicación se diseñó el logo el cual se muestra en la figura 11.

Figura 11. Logo de la aplicación.

El cogido de todas las paginas es encontrado en el anexo A.

7.3 Servicios web

Para crear los servicios web REST se hizo uso de un framework llamado Jersey el cual utiliza la API JAX-RS de Java que facilita las conexiones http.

Se crearon 7 servicios con sus diferentes métodos de comunicación, que ayudan identificar al usuario, buscar profesores, buscar estudiantes, programar citas etc. Se muestra a continuación el catálogo de servicios.

Los siguientes servicios web se encuentran en la url
http://aisii.azc.uam.mx:8080/citasUAM/
todos los servicios web reciben y devuelven información en objetos JSON, el formato es especificado a continuación en el ejemplo de cada servicio.

7.3.1 Login

Éste servicio web permite identificar a un usuario registrado, recibe un json con los datos matrícula en caso de ser estudiante o número económico para profesor y contraseña. La respuesta son los datos del estudiante o profesor, que son nombre apellido, correo, urlFoto, se muestra un ejemplo a continuación.

Ejemplo.
POST - /rest/login
Recibe.
{"id":"2113001882",
"contrasena":"miContraseña"}

Devuelve.
{"matricula":2113001882,
"nombre":"adamo Jordan",

"apellido":"figueroa Pérez",
"carrera":"Ingenieria en Computación",
"correo":"jordan.figu@gmail.com",
"urlFoto":"https://scontent-dft4-2.xx.fbcdn.net/v/t1.0-9/13151628_1138629776188795_8733678262060846505_n.jpg?oh\u003d35631ac2b82402daaa0fa1d6025b07af\u0026oe\u003d58C4B542",
"contrasena":"miContraseña"}

7.3.2 Profesores

El servicio profesores busca un profesor registrado con el número económico indicado, si éste existe devuelve los datos del profesor, que son numero economico, nombre, apellido, cubículo , correo, contraseña, url de la foto, a continuación se muestra un ejemplo.

GET- /rest/profesores/:numeroEconomico
Ejemplo.
Petición get : /rest/profesores/11111.
Devuelve
{"numeroEconomico":"11111","nombre":"José Alejandro","apellido":"Reyes Ortiz","cubiculo":"h-293","correo":"jose@gmail.com"}

El método post registra un nuevo profesor con la información enviada en un json, que es numero economico , nombre, apellido, cubículo, correo, contraseña y la url de la foto, se muestra un ejemplo a continuación.

POST - /rest/profesores/
Ejemplo.
Recibe
```
{
  "numeroEconomico": "3333",
  "nombre": "Armando",
  "apellido": "Prado",
  "cubiculo": "h-100",
  "correo": "armando.prado@gmail.com",
  "contrasena": "123",
  "urlFoto": "http://...."
}
```
Si se guardó con éxito devuelve
{"code":200, "message":"registro exitoso, ahora inicia sesión " }
Si hubo algún error devuelve
{"code":400, "message":"error al registrar probablemente ya existe " }
Si los datos enviados no son correctos
{"code":400, "message":"error datos incorrectos " }

7.3.3 Estudiantes

El método get del servicio estudiantes busca y devuelve la información registrada de un estudiante por matrícula, la cual es matricula, nombre, apellido, carreara, correo, url de foto, se muestra un ejemplo a continuación.

GET - /rest/estudiantes/:matricula

Ejemplo.
Petición get : /rest/estudiantes/2113001882
Devuelve
{"matricula":2113001882,
"nombre":"adamo Jordan",
"apellido":"figueroa Pérez",
"carrera":"Ingenieria en Computación",
"correo":"jordan.figu@gmail.com",
"urlFoto":"https://scontent-dft4-2.xx.fbcdn.net/v/t1.0-9/13151628_1138629776188795_8733678262060846505_n.jpg?oh\u003d35631ac2b82402daaa0fa1d6025b07af\u0026oe\u003d58C4B542",
"contrasena":"miContraseña"}
Si no está registrada la matrícula devuelve
{"code":400" ,"message" :"No Existe el estudiante " }

El método post de estudiantes registra un nuevo estudiante con los datos enviados, que deben ser matricula, nombre, apellido, carrera, correo, contraseña, url de foto, se muestra un ejemplo a continuación.

POST- /rest/estudiantes
Ejemplo.
{
"matricula": "211300",
"nombre": "Jordán",
"apellido": "Figueroa",
"carrera": "Ing. en Computación",
"correo": "jordan.figu@gmail.com",
"contrasena": "123",
"urlFoto": "http://...."
}
Si el estudiante es registrado, devuelve
{"code":200, "message":"registro exitoso, ahora inicia sesión " }
Si se hubo algún error, devuelve
{"code": "400", "message":"error al registrar probablemente ya existe " }

7.3.4 Citas

La petición get al servicio citas busca las horas ya reservadas de un profesor en un día en particular, para esto recibe la fecha y número económico del profesor con el formato mostrado en el ejemplo siguiente , y devuelve todas las citas reservadas.

get - rest/citas/fecha-numeroEconomico

Ejemplo
petición get : **rest/citas/2016-11-02-25051**
devuelve
{"dia":"miercoles","horaInicio":"08:00","horaFinal":"10:00","horasReservadas":[{"horaInicioR":"8:40","horaFinalR":"9:20"},{"horaInicioR":"9:45","horaFinalR":"10:00"}]}

Se tiene otra petición get la cual busca todas las citas pertenecientes al profesor especificado que aún no se han aceptado, varía de la primera en la respuesta, ya que devuelve todas las citas pendientes además solo recibe el número económico, se muestra un ejemplo a continuación.

get - rest/citas/numeroEconomico
Ejemplo.
Petición get : /citasUAM/rest/citas/11111
Devuelve
{"listaCitas":[{"numeroEconomico":"11111","idCita":"4","horaInicio":"10:00","horaFinal":"11:00","asunto":"necesito ayuda con mi tarea de patrones
","estatus":"pendiente","fecha":"2016-12-14","horariodias_idHorario":"2","estudiante_matricula":"2113001882","nombreEstudiante":"adamo figueroa"}]}

El método post crea una nueva solicitud de cita con estado “pendiente” y la guarda en la base de datos, para esto recibe la hora de inicio y hora final de la cita, asunto, fecha, la matrícula del estudiante que solicita, el numero economico del profesor y el dia, se muestra un ejemplo a continuación.

POST - rest/citas/
Ejemplo.
Recibe
{
  "horaInicio": "08:00",
  "horaFinal": "10:00",
  "asunto": "ayuda con el pt",
  "fecha": "2016-12-20",
  "matricula": "2113001882",
  "numeroEconomico": "11111",
  "dia": "martes"
}
Devuelve
{"code":200, "message":"petición de reservación enviada, solo espera la confirmación del profesor " }

La petición put cambia el estado de una cita “pendiente” a “aceptada” o “cancelada”, para esto es necesario enviar el id de cita y el estatus deseado, se muestra el ejemplo a continuación.

PUT - rest/citas/

Ejemplo.
Recibe.
{
  "idCita": "4",
  "estatus": "aceptada"
}
Si el cambio fue exitoso, devuelve el mismo json.
En caso de error devuelve
{"code":400, "message":"Error al modificar el estatus" }

7.3.5 Horario Profesores

El método get de este servicio busca el último horario registrado por el profesor especificado, y devuelve los datos que son el trimestre actual y los días de asesorías registrados, se muestra un ejemplo a continuación.

get - /rest/horarioProfesores/:numeroEconomico

Ejemplo.
Petición GET - /rest/horarioProfesores/11111
devuelve
{"trimestreActual":"17-i","dias":[{"dia":"martes","horaInicio":"07:00","horaFinal":"10:00"},{"dia":"miercoles","horaInicio":"10:00",
"horaFinal":"12:00"}]}
si no existe el profesor devuelve
{\"code\":400, \"message\":\"No existe el profesor \" }
si el profesor no tiene registrado horarios devuelve
{\"code\":400, \"message\":\"No hay horarios registrados \" }

El método post crea un nuevo horario, el cual recibe la información necesaria, una lista con los días disponibles y sus respectivos horarios de asesoría, para después guardarlo en la base de datos, se muestra un ejemplo a continuación.

post - /rest/horarioProfesores/

Ejemplo.
Recibe
{
"numeroEconomico": "11111",
"trimestre": "17-O",
"horarioDias": [
{
"dia": "lunes",
"horaInicio": "08:00",
"horaFinal": "09:00"
},
{
"dia": "jueves",
"horaInicio": "10:00",
"horaFinal": "13:00"
}
]
}
En caso de éxito devuelve.
{\"code\":200, \"message\":\"se registraron los horarios \" }
Si hubo algún error devuelve.
{\"code\":400, \"message\":\"error al registrar horario \" }

7.3.6 citas próximas profesor

Mandar una solicitud get a este servicio busca todas las citas del profesor que ya están programadas en los próximos 3 días, es necesario enviar el numero economico del profesor, ejemplo si hoy es 01/01/2017 regresará las citas en las fechas 01/01/2017, 02/01/2017 y 03/01/2017, se muestra un ejemplo a continuación.

Get /rest/citasProximasProfesor/:numeroEconomico

Ejemplo.
Petición get : /rest/citasProximasProfesor/:11111

Recibe.
[{"horaInicio":"11:00","horaFinal":"15:00","asunto":"Cita","fecha":"2016-12-16","dia":"viernes","matricula":"2113001882","nombre":"adamo ","apellido":"figueroa","urlFoto":""}]
en caso de no tener citas
{"code":400, "message":"Sin Citas Próximas " }

7.3.7 citas próximas estudiante

El método get de este servicio busca todas las citas del estudiante ya programadas en los próximos 3 días, es necesario enviar la matrícula del estudiante y devuelve una lista con las citas aceptadas de sus diferentes profesores, se muestra un ejemplo a continuación.

Get /rest/citasProximasEstudiante/:matricula
Ejemplo.
Petición get : /rest/citasProximasEstudiante/2113001882
[{"horaInicio":"11:00","horaFinal":"15:00","fecha":"2016-12-16","asunto":"Cita","dia":"viernes","nombreProfesor":"alberto","apellidoProfesor":"santander","cubiculo":"h-200","urlFoto":""}]
Si no hay citas regresa.
{"code":400, "message":"Sin Citas Próximas " }

7.3.8 Organización de servicios web.

Los servicios web se conectan con la base de datos a través de un framework llamado hibernate, el cual facilita las conexiones y mapea las tablas de la base de datos a clases en java llamados POJOS, se organizó el código en 4 paquetes como se muestra en la figura 5.

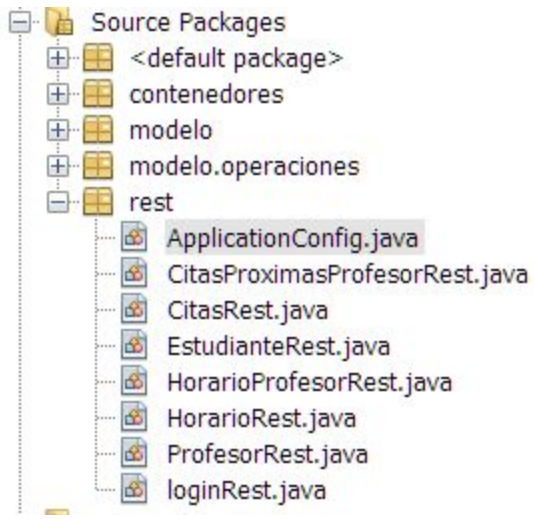

Figura 12. Organización de paquetes.

contenedores: Tiene clases para mapear los objetos que son enviados y recibidos por la aplicación móvil.

modelo: Tiene las clases mapeadas con la base de datos.

modelo.operaciones: Contiene la clase encargada de hacer todas las operaciones finales con la base de datos utilizando las APIs de Hibernate.

rest: Contiene las clases de los servicios web.

7.4 Base de datos

Para guardar la información del sistema se creó una base de datos con MySQL, la cual consta de 5 tablas, las cuales son, profesor para guardar la información necesaria de cada profesor, estudiante para guardar la información necesaria de cada estudiante, horariotrimestre la cual guarda los horarios del profesor trimestre por trimestre, horariodias guarda el día en que el profesor está disponible para dar asesoría y sus respectivas horas de inicio y fin, por último se tiene la tabla citas, la cual guarda la cita que el alumno programe con el profesor, con datos como hora de inicio y fin, asunto de la asesoría y la fecha, para complementar se muestra el modelo entidad relación en la figura 12.

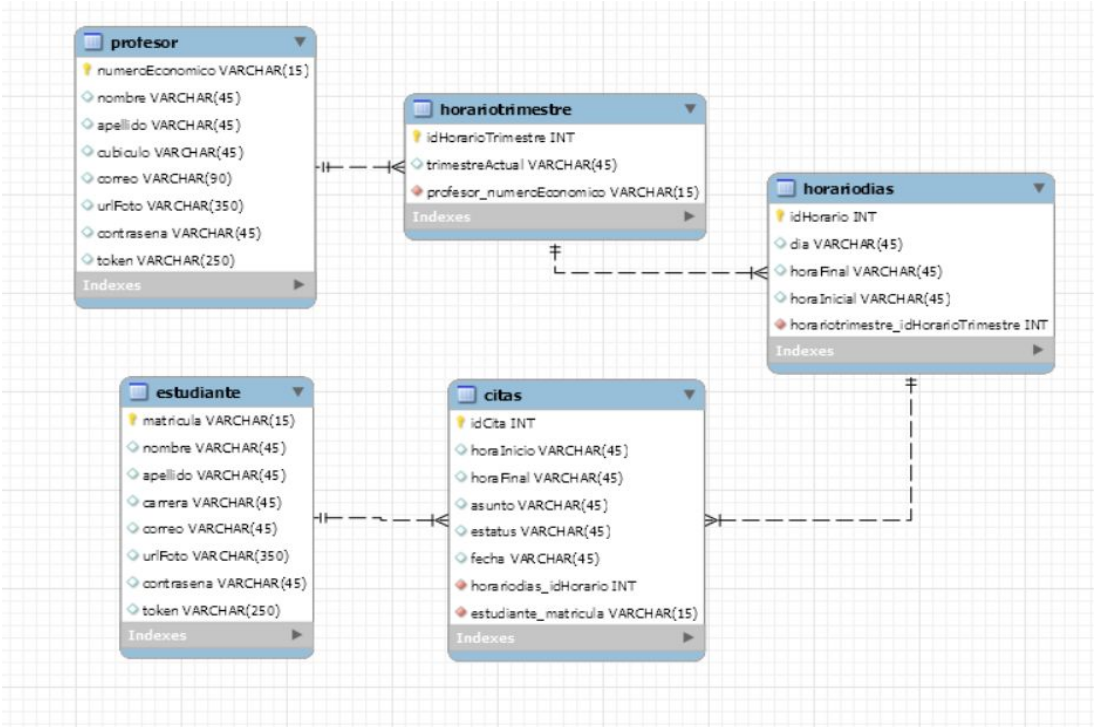

Figura 6. Modelo entidad relación BD.

El script para generar la base de datos se muestra en el anexo C.

7.5 Conexiones

El proceso para hacer una conexión comienza con una la solicitud desde la aplicación, está reacciona a cierto evento puede ser automáticamente al entrar a alguna página como la sesión principal o al presionar un botón como el de registro.

Existe una solo clase llamada ConexionesREST programada en el archivo conexiones-rest.ts, mostrado en el anexo A, esta clase se encarga de hacer todas conexiones con el servidor mediante una petición http, a continuación se explica el funcionamiento de la clase y algunos métodos.

```
export class ConexionesREST {


import { Injectable } from '@angular/core';

import { Http,Headers } from '@angular/http';

import 'rxjs/add/operator/map';

import { Observable } from 'rxjs/Observable';
```

Ionic provee la clase Http las cual facilita los métodos para hacer una solicitud http, ésta clase es inyectada en el constructor, para ser usada en los métodos también se especifica la dirección url donde se encuentran los servicios en la variable urlServer.

```
 constructor(public http: Http) {

 urlServer: string = 'http://aisii.azc.uam.mx:8080//citasUAM/rest/';


   console.log('Hello ConexionesREST Provider');

 }
```

Cada método en esta clase consulta por un servicio diferente, por ejemplo en el método login mostrado a continuación hace una solicitud POST en la dirección http://aisii.azc.uam.mx:8080//citasUAM/rest/login, ademas envia datos los cuales son el id y contraseña estos estan guardados en la variable datosSesión, se destaca que el método post de http recibe 2 parametros, la dirección url del servicio a consultar y los datos a enviar, el método map sirve para mapear el resultado a una variable json, y continuación retorna el resultado al método que lo llamo, en este caso se encuentra en inicio.ts

```
   login(datosSesion){
```

```
    var response = this.http.post(this.urlServer+"login",datosSesion).map(res => res.json());
                    return response;
  }
```

El siguiente método hace una solicitud POST para registrar un nuevo estudiante, este se manda a llamar en la página registro estudiante programada en el archivo registro-estudiante.ts se muestra el código completo en el Anexo A, el resultado se mapea en un objeto json con el método map().

```
  registroEstudiante(datosRegistro){
    var response = this.http.post(this.urlServer+"estudiantes",datosRegistro).map(res => res.json());
    return response;
  }
```

El método getProfesor() mostrado a continuación se usa para buscar profesores, por ejemplo al momento de programar una cita, ya que es un método get la información es enviada en la url, por eso se concatena la direción con numeroEconomico y esta es una variable que contiene el id del profesor a buscar, una solicitud de ejemplo es http://aisii.azc.uam.mx:8080//citasUAM/rest/profesores/12345 devuelde los datos del profesor con numero economico 12345.

```
  getProfesor(numeroEconomico){

    var response = this.http.get(this.urlServer+"profesores/"+ numeroEconomico).map(res => res.json());
    return response;
  }
```

El método siguiente registrarHorarios() es usado para que el profesor guarde su horario actual en el sistema, lo hace con una petición post a http://aisii.azc.uam.mx:8080/citasUAM/rest/horarioProfesores/ enviando como parametro un objeto json con los datos del horario a registrar, el servidor responde con un mensaje con codigo 200 al ser exitoso.

```
  registrarHorario(datosHorario){
    var response = this.http.post(this.urlServer+"horarioProfesores",datosHorario).map(res => res.json());
    return response;
  }
```

Todos los servicios web se describen en la sección 7.3 de este documento.

7.5 Caso de uso programar una cita.

Se muestra en la figura 14 el caso de uso principal del sistema, el cual consiste en programar una cita.

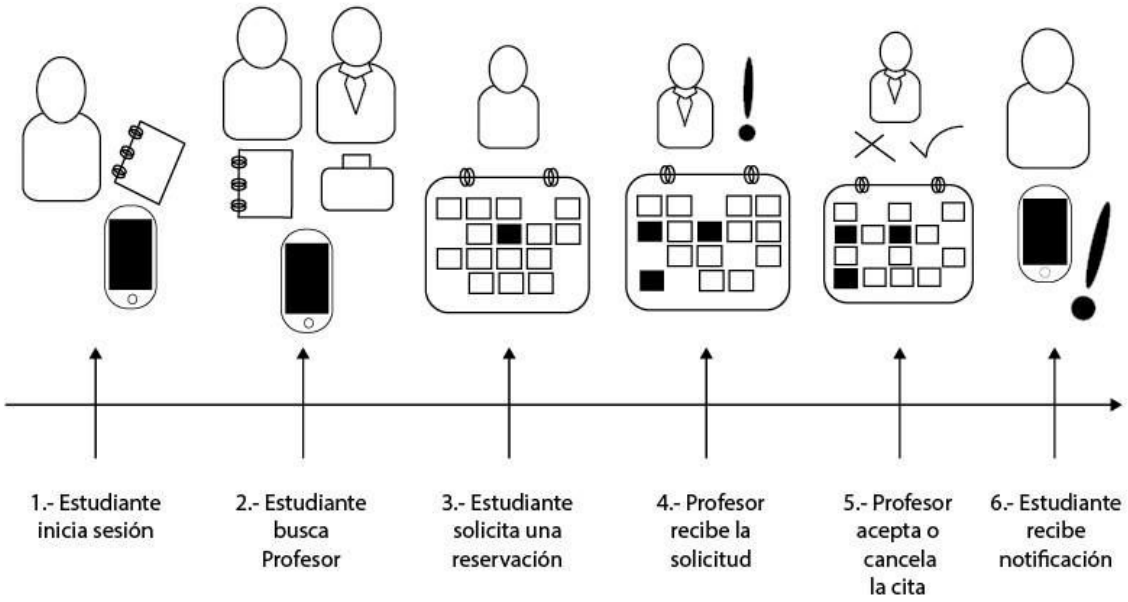

Figura 14. Caso de uso programar cita

El caso de uso programar una cita, es básicamente la funcionalidad principal del sistema, a continuación se describe detalladamente el código ejecutado para cumplir este caso de uso.

1.El Estudiante inicia sesión

Desde la página inicio se escriben los datos en el formulario al presionar el botón iniciar sesión llama al siguiente método.

```
iniciarSesion(){
  var tokenAndroid;
  if(this.token!= null){
    tokenAndroid= this.token;
  }
  var datosSesion = {
    "id": this.valoresFormulario.value.id,
    "contrasena" : this.valoresFormulario.value.contrasena,
    "token" : tokenAndroid,
  };
```

```
  this.conexion.login(datosSesion).subscribe(data => {
    if(data.matricula!=null){ //guarda datos de alumno
      this.storageProvider.saveData("matricula",data.matricula);
      this.storageProvider.saveData("nombre",data.nombre);
      this.storageProvider.saveData("apellido",data.apellido);
      this.storageProvider.saveData("carrera",data.carrera);
      this.storageProvider.saveData("correo",data.correo);
      this.storageProvider.saveData("urlFoto",data.urlFoto);
      // envia a sesion alumno
      this.navCtrl.setRoot(SesionEstudiantePage,data);
    }
    if(data.numeroEconomico!=null)
      { // guarda datos de profesor
      this.storageProvider.saveData("numeroEconomico",data.numeroEconomico);
      this.storageProvider.saveData("nombre",data.nombre);
      this.storageProvider.saveData("apellido",data.apellido);
      this.storageProvider.saveData("cubiculo",data.cubiculo);
      this.storageProvider.saveData("correo",data.correo);
      this.storageProvider.saveData("urlFoto",data.urlFoto);

          //cambia de pagina y envia datos
      this.navCtrl.setRoot(SesionProfesorPage,data);

    }
    if(data.message!=null){
      alert(data.message);
    }
            },
            err => {
                alert("problemas en el servidor");
            });
  }
```

El método que hace la conexión con el servicio web es this.conexion.login(datosSesion) se debe recordar que conexion es un objeto de ConexionesREST, la clase encargada de hacer solicitudes al servidor.

Por otro lado en el servidor la clase Login contiene el servicio web encargado de responder a esta solicitud, programado en el archivo LoginRest.java mostrado en el anexo A, dado que es una solicitud post, és el siguiente método quien se encarga de manejar la solicitud, se explica el código a continuación.

```
@POST
   public String post(String data){
       //crea un objeto login con los datos enviados y luego comprueba el login
       System.out.println(data);
```

Para facilitar el manejo de datos en Java se crea un objeto login de clase Login, que es un mapeo del string en formato json recibido.

```
       Login login = gson.fromJson(data, Login.class);
```

Con el código mostrado a continuación se busca el estudiante o profesor en la base de datos, con el método buscarEstudiante() el cual se explica en breve, éste regresa los datos registrados del estudiante, al tenerlos se compara la contraseña enviada con la

registrada en la base de datos, luego se dará una respuesta a la aplicación dependiendo del resultado, se ocupa el mismo proceso para iniciar sesión con un profesor.

```
Estudiante est = operDB.buscarEstudiante(login.getId());
if(est != null){
     if(est.getContrasena().equals(login.getContrasena())){
        //inicio de sesión agregar token
        if(login.getToken()!=null){
           operDB.tokenEstudiante(login);
        }
       est.setCitases(null); // para que no exista error en el parser gson
         return gson.toJson(est);
       }else{
         return "{\"code\":400, \"message\":\"Contraseña incorrecta \" }";
        }
}else{
    Profesor profe = operDB.buscarProfesor(login.getId());
    if(profe!=null){
       System.out.println(profe.getContrasena()+ " "+ login.getContrasena());
       if(profe.getContrasena().equals(login.getContrasena())){
         if(login.getToken()!=null){
            System.out.println("se va a regitrar el token ");
           operDB.tokenProfesor(login);
        }
         profe.setHorariotrimestres(null);// genera error en  el parser profesor tiene horarios
         return gson.toJson(profe);
       }else{
         return "{\"code\":400, \"message\":\"Contraseña incorrecta \" }";
        }

    }else{
       return "{\"code\":400, \"message\":\"Error usuario o contraseña inválidos \" }";
    }
}
}
```

La clase encargada de hacer todas las conexiones a la base de datos es OperacionesDB, en el código anterior operDB es un objeto de esta clase la cual contienen los métodos buscarProfesor() y buscarEstudiante() utilizados anteriormente, El codigo completo del archivo OperacionesDB.java se encuentra en el anexo B.

A continuación se explica el código del método buscarEstudiante().

```
public Estudiante buscarEstudiante(String matricula){
```

Se recibe como parámetro la matrícula.

```
Estudiante estudianteBuscado;
```

En hibernate es necesario crear una sesión con la base de datos, session es el objeto encargado de hacer las consultas y modificaciones.

```
Transaction tx = session.beginTransaction();
```

El siguiente método de session busca el estudiante por id, en este caso la matricula y es mapeado a una clase de tipo Estudiante.

```
estudianteBuscado = (Estudiante) session.get(Estudiante.class, matricula);
```

Se hace commit y devuelve el estudiante buscado.

```
tx.commit();
return estudianteBuscado;
```

```
}
```

En resumen el resultado de hacer una solicitud post al servicio web login, es saber si el estudiante está registrado en el sistema, y si su contraseña es correcta, si todo sale bien el servicio devuelve los datos del estudiante.

Regresando a la aplicación en el código de inicio.ts

La respuesta exitosa del servidor es regresar los datos del estudiante, por lo tanto en el código siguiente se compara por el atributo matrícula, si este existe significa que no hubo errores y se procede a guardar los datos en memoria local del teléfono después se redirecciona a la página de la sesión del estudiante.

```
this.conexion.login(datosSesion).subscribe(data => {
  if(data.matricula!=null){ //guarda datos de alumno
    this.storageProvider.saveData("matricula",data.matricula);
    this.storageProvider.saveData("nombre",data.nombre);
    this.storageProvider.saveData("apellido",data.apellido);
    this.storageProvider.saveData("carrera",data.carrera);
    this.storageProvider.saveData("correo",data.correo);
    this.storageProvider.saveData("urlFoto",data.urlFoto);
    // envía a sesion alumno
    this.navCtrl.setRoot(SesionEstudiantePage,data);
  }
```

2. El estudiante busca un profesor

Antes de poder programar una cita es necesario conocer al profesor, debe estar registrado en la aplicación y tener registrado un horario, desde una cuenta de estudiante en el menú se presiona programar cita, este botón redirecciona a la página programar cita, mostrada en el archivo programar-cita.ts y programar-cita.html en el anexo A.

El estudiante ingresa un profesor en el campo del formulario mostrado, y presiona el botón buscar.

Se ejecuta el siguiente código dentro de la clase ProgramarCita, conexion.gethorarioActual() es el método encargado de consultar al servicio web por el horario del profesor.

```
buscarProfesor(){
  this.conexion.getHorarioActual(this.numeroEconomico).subscribe(data=>{
    if(data.dias!=null){
       console.log(data);
    // this.horarioDias=data;
    this.horarioDia= data.dias;
    this.mostrarFechas= true;
    }else if(data.message!=null){
      alert(data.message);
    }
  });
}
```

La petición se hace al servicio web horarioProfesores el cual primero busca al profesor solicitado en la base de datos, en caso de no encontrarlo devuelve un mensaje notificando que el profesor no está registrado.

```
Profesor profesor= operDB.buscarProfesor(id);
```

Crea un objeto de tipo HorarioReturnJson el cual sólo servirá como contenedor para guardar los datos necesarios que son el horario actual sus días y horas de asesoría.

```
HorarioReturnJson horarioReturn = new HorarioReturnJson();//objeto a retornar en json

        Set<Horariodias> horarioDias= horarioActual.getHorariodiases();
        for( Horariodias horarioDia : horarioDias){
           System.out.println(" dia : " + horarioDia.getDia());
           HorarioDiasJson horarioj = new
HorarioDiasJson(horarioDia.getDia(),horarioDia.getHoraInicial(),horarioDia.getHoraFinal());
           listaDias.add(horarioj);
        }
        horarioReturn.setDias(listaDias);
```

Devuelve un json con los datos del horario del profesor.

```
        return gson.toJson(horarioReturn);
```

La aplicación al obtener respuesta muestra los datos consultados, se utiliza el siguiente fragmento de código para esto se crea el objeto horarioDia el cual muestra los datos en la vista.

```
    if(data.dias!=null){
       console.log(data);
    // this.horarioDias=data;
    this.horarioDia= data.dias;
    this.mostrarFechas= true;
    }
```

Se muestra todo el código del servicio HorarioProfesor en el archivo HorarioProfesorREST.java del Anexo B.

3. El estudiante solicita una reservación

Una vez consultado el horario actual es necesario, seleccionar la fecha en la que se desea programar una cita y presionar el botón ver disponibilidad.

El cual ejecuta el el método buscarDisponibilidad() y este llama a

```
this.conexion.getDisponibilidad(this.myDate+"-"+this.numeroEconomico)
```

Recordar que el código está en el archivo programar-cita.ts mostrado en el anexo A.

getDisponibilidad() hace una petición get al servicio web citas, programado en el archivo CitasREST.java mostrado en el Anexo B.

En la clase CitasREST se usaron las clases Calendar para obtener la fecha del día actual que registra el servidor y después modificarla con la fecha especificada,se hace con la finalidad de obtener el nombre del dia en esa fecha.

```
    Calendar now = Calendar.getInstance();
```

```
now.set(Integer.parseInt(year), Integer.parseInt(month)-1, Integer.parseInt(day));
```

Se compara en un ciclo, si el profesor tiene asesorias en ese dia.

```
for( Horariodias horarioDia : horarioDias){

        if(horarioDia.getDia().equals(diaCita)){

           System.out.println("existe el dia¡¡¡");
```

Por ejemplo si el profesor sólo registra asesorías los días jueves, pero la fecha enviada es un dia martes entonces devuelve un mensaje especificando que ese dia no hay asesorías.

```
return "{\"code\":400, \"message\":\" no hay asesoría el "+diaCita+" \" }";
```

Si el día es válido se busca por las citas ya aceptadas y se agregan al objeto horasReservadas, el cual será enviado a la aplicación para que el estudiante sepa en qué hora no puede solicitar asesoría.

```
for(Citas cita : citas)
            {
               if(cita.getEstatus().equals("aceptada")){
                  horas = new HorasReservadas();
                  horas.setHoraInicioR(cita.getHoraInicio());
                  horas.setHoraFinalR(cita.getHoraFinal());
                  horasReservadas.add(horas);

               }

            }
```

Con esto se muestra en la página un formulario para elegir el horario y el motivo de la cita, se debe presionar en reservar cita para enviar la solicitud.

Al presionar el botón reservar cita se llama al método reservarCita() el cual crea el objeto json en la variable cita y conexion.reservarCita() hace una petición post a al servicio web citas, como se muestra en el siguiente código.

```
var cita = {horaInicio: this.horario.horaInicio,

          horaFinal:this.horario.horaFinal,
          asunto:this.asunto,
          fecha:this.myDate,
          matricula:this.matricula,
          numeroEconomico:this.numeroEconomico,
          dia: this.horasDisponibles.dia,
        };
    this.conexion.reservarCita(cita).subscribe(data=>{
```

El servicio web citas crea el objeto miCita el cual guardará los datos en la base de datos cabe resaltar que el estatus se configura en pendiente.

```
      Citas miCita =  new Citas();
        miCita.setEstudiante(estudiante);
        miCita.setHorariodias(horaDia);
```

```
miCita.setAsunto(cita.getAsunto());
miCita.setEstatus("pendiente");
miCita.setFecha(cita.getFecha());
miCita.setHoraInicio(cita.getHoraInicio());
miCita.setHoraFinal(cita.getHoraFinal());

operDB.reservarCita(miCita);
```

Después de esto se notifica al estudiante que la petición de cita ha sido enviada.

4. El profesor recibe la solicitud.

Antes de terminar con la clase programarCita el método reservarCita() también envía una notificación push al profesor, de la siguiente manera.

Primero hace una solicitud get al servicio profesores con la finalidad de obtener el token del teléfono que se registró como teléfono del profesor, después se envía una solicitud a la API de Ionic la cual se encarga de enviar la notificación al teléfono especificado gracias al token, de esta manera el profesor será notificado al instante.

```
this.conexion.getProfesor(cita.numeroEconomico).subscribe(data=>{
            alert(JSON.stringify(data));
            if(data.token!=null){
              var mensaje = "Nueva solicitud - " + cita.matricula + " : " + cita.asunto;
              this.conexion.push(data.token,mensaje).subscribe(data=>{
                alert(JSON.stringify(data));
              });
            }
```

5. El profesor acepta o cancela la cita

El profesor inicia sesión de la misma manera que lo hace un estudiante, dentro del menú selecciona aceptación/cancelación de citas, esto redirecciona a la página citas pendientes, programada en el archivo cancelar-citas.ts mostrado en el Anexo A.

En el constructor de la clase se tiene el método getDisponibilidad() el cual hace una solicitud get al servicio web citas para encontrar las citas pendientes,como este código está en el constructor el proceso es automático y se hace cada vez que se redirecciona a esta página, se muestra el código a continuación.

```
this.conexion.getDisponibilidad(this.numeroEconomico).subscribe(data=>{
  // this.horarioDias=data;
  if(data.message==null){
    console.log(data);
    this.listaCitas= data.listaCitas;
    if(this.listaCitas.length==0){
      this.mensaje="sin citas pendientes";
    }else{
      this.mensaje="";
    }
    console.log(this.listaCitas);

  }else{
```

```
      alert(data.message);
    }
  });
```

El resultado de la consulta se guardan en la variable de arreglo listaCitas y los datos que tenga listaCitas se ven reflejado en la interfaz gracias al siguiente código html mostrado a continuación,

```
<ion-list *ngFor="let citas of listaCitas">
  <ion-item>
    <h2>{{citas.nombreEstudiante}} - {{citas.estudiante_matricula}}</h2>
    <p>{{citas.asunto}}</p>
    <p>Fecha: {{citas.fecha}}  Hora: {{citas.horaInicio}} - {{citas.horaFinal}}</p>
    <p>id cita : {{citas.idCita}}</p>
    <button ion-button color="rojo-uam" round (click)="reservarCita(citas.idCita,
citas.estudiante_matricula)">Aceptar</button>
    <button ion-button color="danger" round (click)="cancelarCita(citas.idCita,
citas.estudiante_matricula)">Cancelar</button>
  </ion-item>
</ion-list>
```

Se utiliza la directiva *ngFor hace un ciclo que itera un objeto del arreglo listaCitas, donde cada objeto es una cita pendiente.

Se tienen 2 botones para cada cita pendiente, que son las opciones de aceptar o cancelar, al presionar los botones se llama a los métodos reservarCita() y cancelarCita() respectivamente.
éstos métodos envían una solicitud put al servicio web citas, con el estatus correspondiente.

```
var cambioCita={
     "idCita": idCita,
     "estatus": "aceptada"
   }
   alert(matricula);
   this.conexion.AceptarCita(cambioCita).subscribe(data=>{
     console.log(data);
```

El servicio web recibe los datos id de cita y estatus, únicamente cambia el estatus de cita.

```
ModificacionCita cambioCita = gson.fromJson(data, ModificacionCita.class);
System.out.println("Solicitud put");
if(operDB.cambioCita(cambioCita)){
   return gson.toJson(cambioCita) ;
}else{
    return "{\"code\":400, \"message\":\"Error al modificar el estatus  \" }";

}
```

El método cambioCita() ejecuta un update a la base de datos.

```
"UPDATE citas SET estatus = '"+cambio.getEstatus()+"' WHERE citas.idCita = '"+cambio.getIdCita()+"'
```

De esta manera cita queda aceptada o cancelada.

6. Estudiante recibe la notificación.

Al final del método cambioCita() mencionado anteriormente, cuando la respuesta es exitosa se ejecuta el siguiente código.

```
this.conexion.getEstudiante(matricula).subscribe(data=>{
   alert(JSON.stringify(data));
   if(data.token!=null){
     var mensaje = "Cita aceptada : " + this.numeroEconomico  ;
     this.conexion.push(data.token,mensaje).subscribe(data=>{
       alert(data);
     });
   }
});
```

El cual busca el estudiante a notificar, consultando con el servicio web estudiantes, para así obtener el token del celular y a continuación envía la solicitud con el metodo conexion.push(), a la API de Ionic la cual se encarga de enviar la notificación.

Es así como se cumple el objetivo principal de la aplicación.

8. RESULTADOS

Para probar la aplicación se registraron 5 estudiantes y 5 profesores desde la aplicación, después se programaron 3 citas por cada profesor y se aceptaron citas y cancelaron algunas,las notificaciones siempre llegaron.

Se muestran los datos usados para las pruebas.

Primero se registraron los profesores exitosamente.

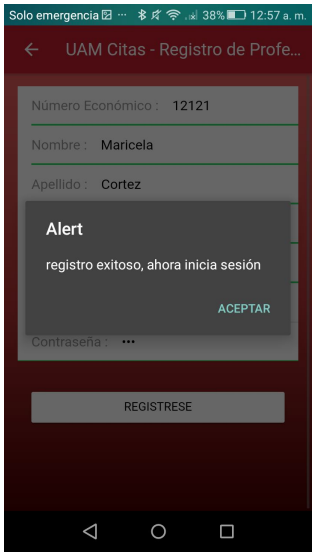

Figura 15. registro exitoso de profesor

Se muestran los datos de todos los profesores registrados en la figura 16.

numeroEconomico	nombre	apellido	cubiculo	correo	urlFoto	contrasena
11111	Armando Rodrigo	Jiménez Fuentes	h-200	jose@correo.com	*NULL*	123
12121	Maricela	Cortez	h-202	Mari.cortez@gmail.com	https://lh6.googleusercontent.com/-0SpfhUloR0U/AAA...	123
12345	José Alejandro	Reyes Ortiz	H-200	jose.reyes@gmail.com	http://aisii.azc.uam.mx/investigadores/Alejandro/i...	123
22222	alberto	santander	h-200	albert@gmail.com	http://www.lavozlibre.com/userfiles/2a_decada/imag...	123
33332	Francisco	trujillo flores	h-102	Fran.tru@gmail.com		123

Figura 16. Datos de profesores registrados.

Se registraron 5 estudiantes exitosamente, se muestra la captura de pantalla en la figura 17.

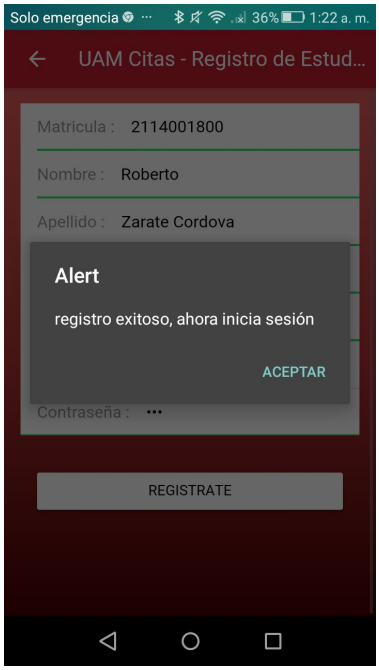

Figura 17. registro exitoso de estudiante.

Se muestran los datos de todos los estudiantes registrados en la figura 18.

matricula	nombre	apellido	carrera	correo	urlFoto	contrasena
2113001881	Maria	Hernández	Ing. en Computación	maria@gmail.com	https://scontent.cdninstagram.com/t51.2885-15/e15/...	123
2113001882	Adamo Jordán	Figueroa Pérez	Ing. En Computación	jordan@gmail.com	https://scontent-dft4-1.xx.fbcdn.net/v/t1.0-9/1272...	123
2113001883	Georgina	Martínez	Ing. Industrial	geor.Marti@gmail.com	https://scontent-dft4-1.xx.fbcdn.net/v/t1.0-9/1493...	123
2123003000	Juan	Gomez	Ing. Electronica	juan.gomez@gmail.com		123
2133008000	Estefania	Mendoza Báez	Ing Ambiental	fany.men@gmail.com	https://d1bvpoagx8hqbg.cloudfront.net/259/01db3687...	123

Se registraron horarios para cada profesor exitosamente, se muestra una captura en la Figura 19.

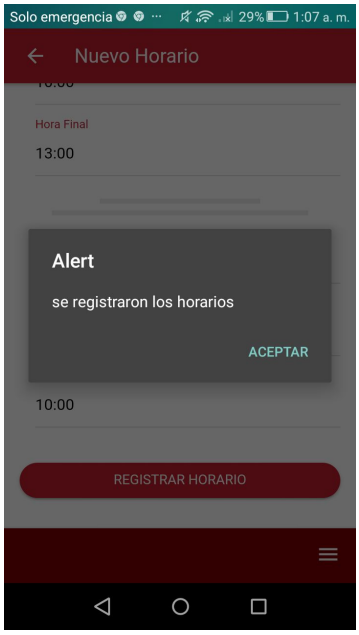

Figura 19. Registro exitoso de horarios

Con las cuentas de estudiantes se registraron diferentes citas con distintos profesores, en todas se envió la petición de reservación como notificación al profesor, se muestra una captura en la figura 20.

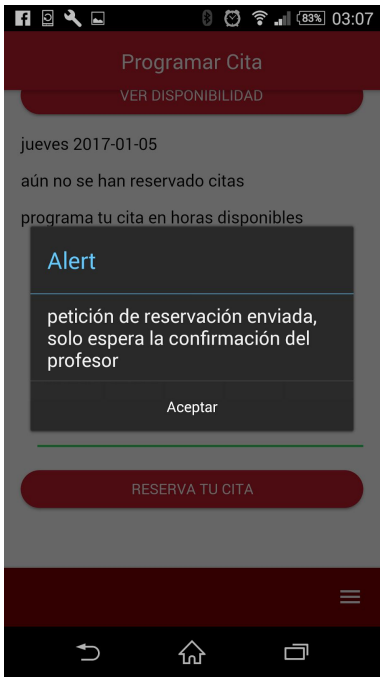

Figura 20. Petición de reservación

Las notificaciones llegan a la cuenta del profesor avisando que tiene citas pendientes por aceptar, se muestra una captura en la figura 21.

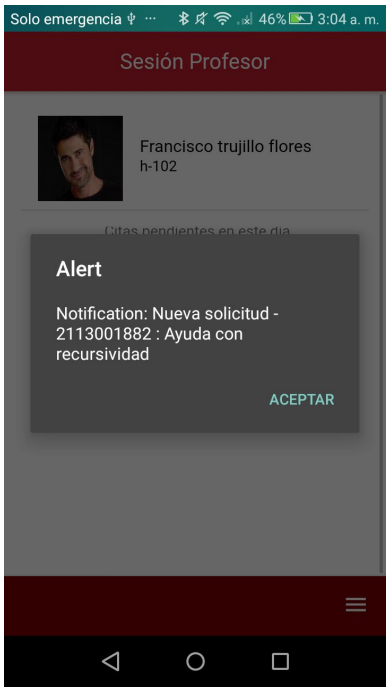

Figura 21. Notificación de nueva solicitud.

Se muestran al profesor las solicitudes de citas pendientes, este puede aceptar o cancelar las citas, se muestra una captura en la figura 22.

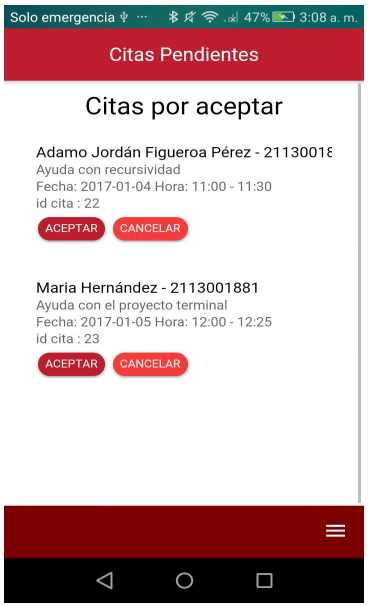

Figura 22. Citas pendientes.

Las notificaciones son enviadas correctamente al estudiante en el momento en que el profesor acepta o cancela una cita, si la aplicación está minimizada o en segundo plano la notificación la recibe el sistema de notificaciones push de android, se muestra una captura en la figura 23.

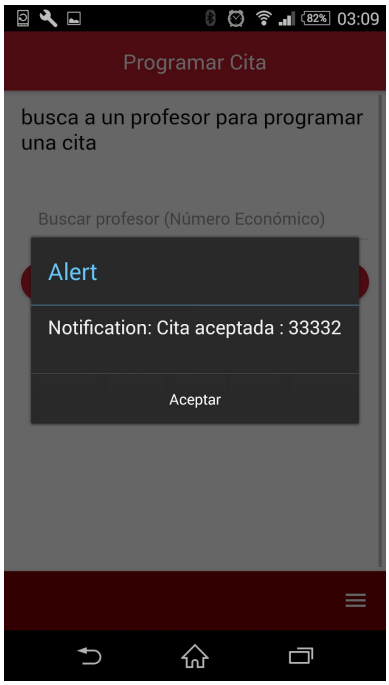

Figura 23. notificación de aceptación.

El estudiante visualiza correctamente las citas próximas que ya fueron aceptadas en la página principal de su sesión, se muestra una captura en la figura 24.

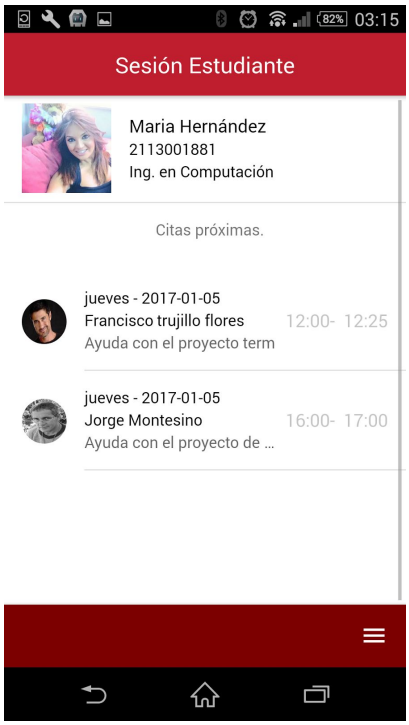

Figura 24. citas próximas estudiante.

El profesor también visualiza correctamente las citas próximas que ya fueron aceptadas. se muestra una captura en la figura 25.

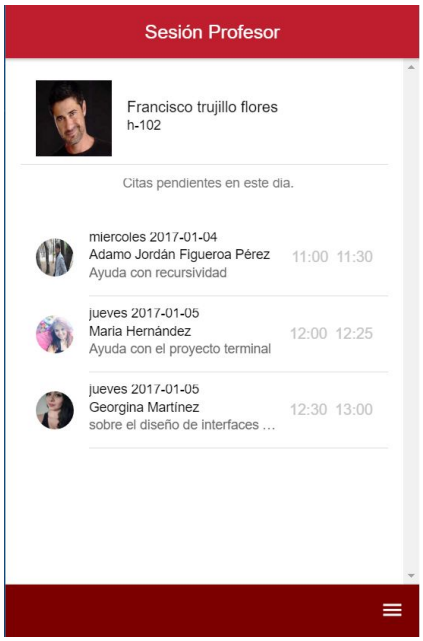

Figura 25. citas próximas profesor

9. ANÁLISIS Y DISCUSIÓN DE RESULTADOS.

Todos los registros fueron exitosos, la aplicación notifica en caso de intentar registrar un usuario ya existente.

El inicio de sesión siempre funciona,se distingue entre la sesión del estudiante y profesor.

La aplicación notifica en caso de iniciar sesión con un usuario no valido o contraseña incorrecta.

Las reservaciones de citas funcionan perfectamente, pero el sistema no verifica el traslape de horarios, las horas ya reservadas se le muestran al estudiante, es responsabilidad del estudiante programar la cita en horarios pertinentes.

Siempre se notifica en caso de aceptación o cancelación de cita,

Las notificaciones solo llegan cuando la aplicación está abierta o en segundo plano, cerrar la app hará que estas no lleguen.

La aplicación solo muestra las citas aceptadas en los proximos 3 dias.

EL sistema es completamente funcional sin embargo hacen falta más funcionalidades para estar completo, como opciones para para configurar la cuenta del usuario y un calendario interactivo.

10. CONCLUSIONES

La arquitectura utilizada para este sistema se ha vuelto muy popular, hoy en día los servicios web son indispensables para cada sistema o plataforma , REST permite la comunicación constante entre servidores y clientes la arquitectura tiene la ventaja de tener una gran escalabilidad y rendimiento, además estos servicios pueden ser consultados por todo tipo de aplicaciones, para este proyecto se implementó una aplicación móvil pero pudo haber sido una aplicación de escritorio o web.

Una de las dificultades es que se trabajo con tecnologías completamente nuevas, ionic al momento de terminar el proyecto Ionic se encuentra en una versión estable pero no final, en lo personal fue muy costoso desarrollar la aplicación debido la gran curva de aprendizaje que tiene Ionic, ya que aunque está basado en tecnologías web, tiene otro lenguaje de programación y etiquetas nuevas no pertenecientes a html.

Sin embargo hay una ventaja enorme de trabajar con aplicaciones híbridas, y es que es posible generar una versión para diferentes S.O. con el mismo código, lo que ahorra tiempo y esfuerzo en el desarrollo.

El sistema de notificaciones utiliza otros servicios web primero los de Ionic Cloud, los cuales conectan con los servicios de Firebase, que es el servicio Google que se encarga finalmente de enviar la notificación, estas notificaciones solo llegan cuando la aplicación se encuentra abierta o en segundo plano, cerrar la aplicación causará que no pueda recibir notificaciones, para hacer persistente el sistema como lo hace whatsapp es necesario una instalar algunos plugins de Cordova que permitan hacerlo, cabe hacer énfasis que esto hace que la aplicación se mantenga abierta todo el tiempo , y por lo tanto tiene consumo de continuo de memoria.

La aplicación es completamente funcional y puede ayudar a organizar y agilizar las citas académicas, básicamente permite el registro, inicio de sesión, reservación, aceptación y cancelación de citas, todo por medio de notificaciones.

El proyecto concluyó con la aplicación y el sistema funcionando; sin embargo es posible continuar con él y mejorarlo, esto gracias a la arquitectura utilizada, es

posible generar más servicios y obtener información útil, además gracias a los datos guardados en la base de datos es posible hacer análisis de datos, generar estadísticas y reportes, responder preguntas como ¿Quién es el profesor más solicitado? ¿Sobre qué temas existen más dudas? ¿En que horario hay más demanda de asesorías ? las respuestas a estas preguntas podrían ser de gran utilidad para ayudar a mejorar la educación.

11. REFERENCIAS BIBLIOGRÁFICAS

[1] k. Guzmán Villanueva, "Aplicación móvil para la recomendación de productos en el comercio electrónico", Proyecto terminal, Universidad Autónoma Metropolitana Azcapotzalco, División de Ciencias Básicas e Ingeniería, México, 2015.

[2] F. García Maldonado, "Prototipo de una aplicación móvil para la gestión de síntomas de un paciente", proyecto terminal, Universidad Autónoma Metropolitana Azcapotzalco, División de Ciencias Básicas e Ingeniería, México, 2014.

[3] A. Guerrero Martín, "Desarrollo de un sistema de gestión de tutorías a través de una aplicación móvil", Proyecto de fin de grado, Universidad Politécnica de Madrid, Escuela Técnica Superior de Ingeniería de Sistemas Informáticos, España, 2015.

[4] M. V. Vargas Ibarra, "Análisis, diseño e implementación del sistema de seguimiento, evaluación y control de las tutorías de tesis para las direcciones de carrera de la facultad de ingeniería ciencias físicas y matemática", Proyecto terminal, Universidad Central del Ecuador, Facultad de Ingeniería Ciencias Físicas y Matemática, Ecuador, 2013.

[5] D. Ramírez Peralta, E. Alcuida Fuentes and A. López Jiménez, "Desarrollo de aplicación web para tutorías académicas, incorporando reingeniería de procesos, programación concurrente y sistemas de gestión de bases de datos distribuidas", Revista Iberoamericana de Producción Académica y Gestión Educativa, no. 02, p. 10, 2016.

[6] Ionicframework.com, "Ionic: Advanced HTML5 Hybrid Mobile App Framework", 2016. [Online]. Available: http://ionicframework.com/. [Accessed: 02- ENE- 2016].

12. ANEXOS

A. Código de la aplicación.

Inicio.html

```
<ion-header>
  <ion-navbar color="rojo-uam">
    <ion-title text-center >Inicio </ion-title>
  </ion-navbar>
</ion-header>

<ion-content class="degrade-principal" padding >
<br>

<form [formGroup]= "valoresFormulario" (ngSubmit)="iniciarSesion()">
    <ion-list>
    <ion-item>
    <ion-label floating>Matricula/ Número Económico</ion-label>
      <ion-input formControlName="id" type="text" value=""></ion-input>
    </ion-item>

    <ion-item>
      <ion-label floating>Contraseña</ion-label>
      <ion-input formControlName="contrasena" type="password"></ion-input>
    </ion-item>
  </ion-list>
  <button ion-button type="submit" [disabled]="!valoresFormulario.valid" full round color="blanco" style="color:black" > Iniciar
sesión </button>
</form>

  <br><br><br>
  <button ion-button full round color="blanco" style="color:black"  (click)="irARegistro()"> Registrarse </button>

</ion-content>
```

inicio.ts

```
import { Component } from '@angular/core';
import { NavController, MenuController } from 'ionic-angular';
import { SesionEstudiantePage } from '../sesion-estudiante/sesion-estudiante';
import { SesionProfesorPage} from  "../sesion-profesor/sesion-profesor";
import { RegistroPage} from "../registro/registro";
import {ConexionesREST} from "../../providers/conexiones-rest";
import {StorageProvider} from "../../providers/storage-provider";


//para form de html
import { FormBuilder, Validators } from '@angular/forms';
import {Push,PushToken} from '@ionic/cloud-angular';

/*
  Generated class for the InicioPage page.

  See http://ionicframework.com/docs/v2/components/#navigation for more info on
  Ionic pages and navigation.
*/
@Component({
  templateUrl: 'inicio.html',
```

```
})
export class InicioPage {
  alumnoPrueba: any;
  valoresFormulario: any;//para agrupar los valores de los inputs
  token: string;


  constructor(public push: Push,public menuCtrl: MenuController, public navCtrl: NavController, public formBuilder :
FormBuilder, private conexion:  ConexionesREST ,public storageProvider : StorageProvider   ) {
    //deshabilita los menus
    this.menuCtrl.enable(false, 'menuProfesor');
    this.menuCtrl.enable(false, 'menuEstudiante');

     //registra el teleforno para las push
      this.push.register().then((t: PushToken) => {
        return this.push.saveToken(t);
      }).then((t: PushToken) => {
        console.log('Token saved:', t.token);
        //se guarda el token guardar en memoria también
        this.token= t.token;
        alert(t.token);
      });

  this.push.rx.notification()
    .subscribe((msg) => {
      alert(msg.title + ': ' + msg.text);
  });


  //liga el formularo al código
   this.valoresFormulario = this.formBuilder.group({
     id: ["",[Validators.maxLength(10),Validators.required]],
     contrasena : ["",Validators.required],

   });

   //elimina los datos en cache si había
    storageProvider.clearData("matricula");
    storageProvider.clearData("numeroEconomico");
    storageProvider.clearData("nombre");
    storageProvider.clearData("apellido");

  }


  //conecta con el servidor envia json con id y contraseña recupera json con datos de la sesión
  iniciarSesion(){
    var tokenAndroid;
    if(this.token!= null){
      tokenAndroid= this.token;
    }
    var datosSesion = {
      "id": this.valoresFormulario.value.id,
      "contrasena" : this.valoresFormulario.value.contrasena,
      "token" : tokenAndroid,
    };

    this.conexion.login(datosSesion).subscribe(data => {
      if(data.matricula!=null){ //guarda datos de alumno
        this.storageProvider.saveData("matricula",data.matricula);
        this.storageProvider.saveData("nombre",data.nombre);
        this.storageProvider.saveData("apellido",data.apellido);
        this.storageProvider.saveData("carrera",data.carrera);
        this.storageProvider.saveData("correo",data.correo);
        this.storageProvider.saveData("urlFoto",data.urlFoto);
        // envia a sesion alumno
```

```
      this.navCtrl.setRoot(SesionEstudiantePage,data);
    }
    if(data.numeroEconomico!=null)
     { // guarda datos de profesor
      this.storageProvider.saveData("numeroEconomico",data.numeroEconomico);
      this.storageProvider.saveData("nombre",data.nombre);
      this.storageProvider.saveData("apellido",data.apellido);
      this.storageProvider.saveData("cubiculo",data.cubiculo);
      this.storageProvider.saveData("correo",data.correo);
      this.storageProvider.saveData("urlFoto",data.urlFoto);

         //cambia de pagina y envia datos
      this.navCtrl.setRoot(SesionProfesorPage,data);

    }
    if(data.message!=null){
     alert(data.message);
    }
            },
            err => {
                alert("problemas en el servidor");
            });
 }

 irARegistro(){
  this.navCtrl.push(RegistroPage);
 }

}
```

intro.html

```
<ion-slides class="estatica" class="degrade-principal" >

 <ion-slide >
  <h2>  Bienvenido </h2>
  <div class="padding-icon" >
       <img style="width:100px;" src="assets/img/logoCitas.png" alt="">
  </div>
  <h3>Programa asesorías y entregas, fácil y rápido.</h3>
  <div class="slide-selector">
   <div class="acomodador">
    <div class="slide-selecionado"> </div>
    <div class="slide-siguiente"></div>
    <div class="slide-siguiente"> </div>
   </div>
  </div>
 </ion-slide>

 <ion-slide>
    <div class="padding-icon">
       <img  src="assets/img/school-bell.png" alt="">
    </div>
    <h3>Recibe notificaciones ante cambios con anticipación. </h3>
  <div class="slide-selector">
   <div class="acomodador">
    <div class="slide-siguiente"> </div>
    <div class="slide-selecionado"></div>
    <div class="slide-siguiente"> </div>
   </div>
  </div>
 </ion-slide>

 <ion-slide>
  <div >
```

```
        <img width ="300" src="assets/img/smartphone.png" alt="">
    </div>
    <h3>Registrate y comienza a programar tus citas</h3>
    <ion-row>
       <ion-col>
          <button ion-button color="blanco" light (click)="irAInicio()">Iniciar</button>
       </ion-col>
    </ion-row>
    <div class="slide-selector">
      <div class="acomodador">
       <div class="slide-siguiente"> </div>
       <div class="slide-siguiente"></div>
       <div class="slide-selecionado"> </div>
      </div>
    </div>
  </ion-slide>

</ion-slides>
```

intro.ts

```
import { Component } from '@angular/core';
import { NavController, MenuController } from 'ionic-angular';
import {InicioPage} from "../inicio/inicio";
import { Storage } from '@ionic/storage';


@Component({
  selector: 'page-intro',
  templateUrl: 'intro.html',

})
export class IntroPage {

  constructor(public navCtrl: NavController, public menuCtrl: MenuController, public storage: Storage) {
   this.menuCtrl.enable(false, 'menuProfesor');
   this.menuCtrl.enable(false, 'menuEstudiante');
   this.storage.set("intro","true");
  }
  irAInicio(){
    this.navCtrl.setRoot(InicioPage);
  }

}
```

cancelar-citas.html

```
<ion-header text-center>
  <ion-navbar color="rojo-uam">
    <ion-title>Citas Pendientes</ion-title>
  </ion-navbar>
</ion-header>


<ion-content padding>
  <h1 text-center>Citas por aceptar</h1>
  <p>{{mensaje}}</p>

<ion-list *ngFor="let citas of listaCitas">
  <ion-item>
    <h2>{{citas.nombreEstudiante}} - {{citas.estudiante_matricula}}</h2>
```

```
    <p>{{citas.asunto}}</p>
    <p>Fecha: {{citas.fecha}}  Hora: {{citas.horaInicio}} - {{citas.horaFinal}}</p>
    <p>id cita : {{citas.idCita}}</p>

    <button ion-button color="rojo-uam" round (click)="reservarCita(citas.idCita, citas.estudiante_matricula)">Aceptar</button>
    <button ion-button color="danger" round (click)="cancelarCita(citas.idCita, citas.estudiante_matricula)">Cancelar</button>


  </ion-item>
</ion-list>

</ion-content>

<ion-footer >
  <ion-toolbar color="footer-uam">
    <ion-icon menuToggle class="btn-menu-footer" color="blanco" name="menu"></ion-icon>
  </ion-toolbar>
</ion-footer>
```

cancelar-citas.ts

```
import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';
import {ConexionesREST} from "../../providers/conexiones-rest";
import { Storage } from '@ionic/storage';


/*
  Generated class for the CancelarCitasPage page.

  See http://ionicframework.com/docs/v2/components/#navigation for more info on
  Ionic pages and navigation.
*/
export class CitasPendientes{
public asunto :string;
public estatus: string;
public estudiante_matricula:string;
public fecha:string;
public horaFinal:string;
public horaInicio: string;
public horariodias_idHorario:string;
public idCita:string;
public numeroEconomico:string;
public nombreEstudiante:string;
}

@Component({
  templateUrl: 'cancelar-citas.html',
})
export class CancelarCitasPage {
listaCitas:Array <CitasPendientes>;
numeroEconomico: string;
mensaje: string = "buscando...";

  constructor(public storage:Storage,public navCtrl: NavController,public conexion:ConexionesREST) {
          this.storage.get("numeroEconomico").then((name) => {
        this.numeroEconomico=name;
        this.conexion.getDisponibilidad(this.numeroEconomico).subscribe(data=>{
          // this.horarioDias=data;
          if(data.message==null){
            console.log(data);
            this.listaCitas= data.listaCitas;
            if(this.listaCitas.length==0){
              this.mensaje="sin citas pendientes";
```

```
        }else{
          this.mensaje="";
        }
        console.log(this.listaCitas);

      }else{
        alert(data.message);
      }
    });

  });

}

reservarCita( idCita:string, matricula: string ){
  var cambioCita={
    "idCita": idCita,
    "estatus": "aceptada"
  }
  alert(matricula);
  this.conexion.AceptarCita(cambioCita).subscribe(data=>{
    console.log(data);
    if(data.message==null){
      alert("Cita aceptada");

          this.conexion.getEstudiante(matricula).subscribe(data=>{
            alert(JSON.stringify(data));
            if(data.token!=null){
              var mensaje = "Cita aceptada : " + this.numeroEconomico  ;
              this.conexion.push(data.token,mensaje).subscribe(data=>{
                alert(data);
              });
            }
          });

    this.navCtrl.setRoot(CancelarCitasPage);
    }else{
      alert( data.message);
    }
  });
}


cancelarCita(idCita:string, matricula: string){
      var cambioCita={
    "idCita": idCita,
    "estatus": "cancelada"
  }
  this.conexion.AceptarCita(cambioCita).subscribe(data=>{
    console.log(data);
    if(data.message==null){
      alert("Cita cancelada");
          this.conexion.getEstudiante(matricula).subscribe(data=>{
            alert(JSON.stringify(data));
            if(data.token!=null){
              var mensaje = "Cita rechazada : " + this.numeroEconomico  ;
              this.conexion.push(data.token,mensaje).subscribe(data=>{
                alert(data);
              });
            }
          });
      this.navCtrl.setRoot(CancelarCitasPage);

    }else{
      alert( data.message);
    }
```

```
    });
  }

}
```

horario-profesor.html

```
<ion-header>
  <ion-navbar color="rojo-uam">
    <ion-title> Horario </ion-title>
  </ion-navbar>
</ion-header>


<ion-content padding>
    <h4 text-center> horario actual</h4>
    <hr color="rojo-uam" class="separador-1">
    <hr class="separador-2">
    <h4 text-center>{{horarioProfesor.trimestreActual}}</h4>
        <div *ngFor="let horario of horarioProfesor.dias">
          <h5 text-center>{{horario.dia}} : {{horario.horaInicio}} - {{horario.horaFinal}}</h5>
        </div>
    <hr class="separador-2">

    <br>
    <button  ion-button color="rojo-uam" full round (click)="irANuevoHorario()">Nuevo Horario</button>
</ion-content>

<ion-footer >
  <ion-toolbar color="footer-uam">
    <ion-icon menuToggle class="btn-menu-footer" color="blanco" name="menu"></ion-icon>
  </ion-toolbar>
</ion-footer>
```

horario-profesor.ts

```
@Component({
  selector: 'page-horario-profesor',
  templateUrl: 'horario-profesor.html'
})
export class HorarioProfesorPage {
  public numeroEconomico: string;
  public horarioProfesor: any;

  constructor(public navCtrl: NavController, public storage:Storage, public conexion :ConexionesREST ) {

     this.horarioProfesor = {
      trimestreActual:"",
      dias:[{dia:"",horaInicio:"", horaFinal:""}]

    }

    this.storage.get("numeroEconomico").then((name) => {
      this.numeroEconomico=name;
      this.conexion.getHorarioActual(this.numeroEconomico).subscribe(data => {
      if(data!=null){
        if(data.trimestreActual!=undefined){
           this.horarioProfesor = data;
        }else{
          this.horarioProfesor.trimestreActual="No se han registrado horarios";
        }
      }
```

```
  })
 });
 }

 irANuevoHorario(){
  this.navCtrl.push(HorarioProfesorNuevo);
 }
}
```

horario-profesor-nuevo.html

```
<ion-header>
 <ion-navbar color="rojo-uam">
  <button  menuToggle>
   <ion-icon  name="menu"></ion-icon>
  </button>
  <ion-title> Nuevo Horario </ion-title>
 </ion-navbar>
</ion-header>


<ion-content padding>


 <ion-list>
  <ion-item>
   <ion-label color="rojo-uam" stacked>Trimestre</ion-label>
   <ion-input [(ngModel)]="trimestre" type="text" placeholder="ejemplo (17-I)"></ion-input>
  </ion-item>

  <ion-item>
   <ion-label color="rojo-uam" stacked>Días de asesoría</ion-label>
   <ion-input type="number" placeholder="ejemplo (1)" [(ngModel)]="numDiasAsesoria"></ion-input>
  </ion-item>

   <br>
   <button ion-button full color="rojo-uam" (click)="agregarDias()"> Agregar dias</button>


  <div *ngFor="let horario of horarioDia">
   <hr class="separador-1">
   <hr class="separador-2">
   <ion-item>
    <ion-label stacked color="rojo-uam"> Día</ion-label>
    <ion-select [(ngModel)]="horario.dia">
     <ion-option value="lunes">lunes</ion-option>
     <ion-option value="martes">martes</ion-option>
     <ion-option value="miercoles">miercoles</ion-option>
     <ion-option value="jueves">jueves</ion-option>
     <ion-option value="viernes">viernes</ion-option>
    </ion-select>
   </ion-item>

   <ion-item>
    <ion-label stacked color="rojo-uam" >Hora Inicio</ion-label>
    <ion-datetime [(ngModel)]="horario.horaInicio" hourValues="{{horasPermitidas}}" minuteValues="{{minutosPermitidos}}"
cancelText="cancelar" doneText="ok" displayFormat="HH:mm"  ></ion-datetime>
   </ion-item>

   <ion-item>
    <ion-label stacked color="rojo-uam" >Hora Final</ion-label>
    <ion-datetime [(ngModel)]="horario.horaFinal" hourValues="{{horasPermitidas}}" minuteValues="{{minutosPermitidos}}"
cancelText="cancelar" doneText="ok" displayFormat="HH:mm"  ></ion-datetime>
   </ion-item>
```

```
    <br>
   </div>
    <br>
    <div *ngIf="registrar">
      <button ion-button (click)="registrarHorario()" color="rojo-uam"  full round >Registrar Horario</button>
    </div>
  </ion-list>



</ion-content>

<ion-footer >
  <ion-toolbar color="footer-uam">
    <ion-icon menuToggle class="btn-menu-footer" color="blanco" name="menu"></ion-icon>
  </ion-toolbar>
</ion-footer>
```

horario-profesor-nuevo.ts

```
import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';
import {ConexionesREST} from "../../providers/conexiones-rest";
import { Storage } from '@ionic/storage';
import {SesionProfesorPage} from "../sesion-profesor/sesion-profesor";



export class HorarioDia{
  public dia: string;
  public horaInicio: string;
  public horaFinal: string;
}
/*
  Generated class for the HorarioProfesorNuevo page.

  See http://ionicframework.com/docs/v2/components/#navigation for more info on
  Ionic pages and navigation.
*/
@Component({
  selector: 'page-horario-profesor-nuevo',
  templateUrl: 'horario-profesor-nuevo.html'
})
export class HorarioProfesorNuevo {
  trimestre: string;
  minutosPermitidos :String ="0,5,10,15,20,25,30,25,40,45,50,55";
  horasPermitidas : String ="07,08,09,10,11,12,13,14,15,16,17,18,19,20,21";
  numDiasAsesoria: number=0;
  horarioDia :Array<HorarioDia>=[];
  registrar: boolean;



  numeroEconomico: string;


  constructor(public navCtrl: NavController,  private conexion : ConexionesREST, public storage : Storage) {

        this.storage.get("numeroEconomico").then((name) => {
         this.numeroEconomico=name;
        });
```

```
}


 registrarHorario(){

   var trimestreHorarioDias = {
     numeroEconomico: this.numeroEconomico,
     trimestre: this.trimestre,
     horarioDias: this.horarioDia,
   }

     console.log(trimestreHorarioDias);
     this.conexion.registrarHorario(trimestreHorarioDias).subscribe(data => {
     alert(data.message);
     this.navCtrl.setRoot(SesionProfesorPage);
     },
     err => {
         alert("problemas en el servidor");
     });

 }


 agregarDias(){
   this.horarioDia=[];
   for(var i =0;i< this.numDiasAsesoria; i++)
   this.horarioDia.push({dia:"",horaInicio:"",horaFinal:""});
   this.registrar= true;
 }
```

programar-cita.html

```
<ion-header text-center>
  <ion-navbar color="rojo-uam">
    <ion-title>Programar Cita</ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>
  <h4>busca a un profesor para programar una cita</h4>
  <ion-item>
    <ion-label primary floating>Buscar profesor (Número Económico) </ion-label>
    <ion-input [(ngModel)]="numeroEconomico" ></ion-input>
  </ion-item>
<br>
  <button ion-button color="rojo-uam" full round (click)="buscarProfesor()"> Buscar </button>
<br>
 <div *ngIf="horarioDia">
  <ion-grid>
  <ion-row>
    <ion-col width-33><h5 text-center>dia</h5></ion-col>
    <ion-col width-33><h5 text-center>hora Inicio</h5></ion-col>
    <ion-col width-33><h5 text-center>hora Final</h5></ion-col>
  </ion-row>

  <div *ngFor="let horario of horarioDia">
      <ion-row>
    <ion-col width-33><h5 text-center>{{horario.dia}}</h5></ion-col>
    <ion-col width-33><h5 text-center>{{horario.horaInicio}}</h5></ion-col>
    <ion-col width-33><h5 text-center>{{horario.horaFinal}}</h5></ion-col>
  </ion-row>
    </div>
  </ion-grid>
  </div>
<div *ngIf="mostrarFechas">
```

```
  <ion-item >
  <ion-label>Fecha</ion-label>
  <ion-datetime displayFormat="DD/MM/YYYY" min="2016" [(ngModel)]="myDate"></ion-datetime>
</ion-item>

  <button ion-button full round color="rojo-uam"  (click)="buscarDisponibilidad()"> ver disponibilidad</button>
  <br>
   <div *ngIf="mostrarHoras">
    <h6>{{horasDisponibles.dia}} {{myDate}}</h6>
    <h6> {{mensaje}} </h6>
   <div *ngFor="let hora of horasDisponibles.horasReservadas">
      <h6> {{hora.horaInicioR}} - {{hora.horaFinalR}}</h6>
    </div>
    <h6> programa tu cita en horas disponibles </h6>
    <ion-grid>
      <ion-row>
        <ion-col width-33>
        </ion-col>
      </ion-row>
    </ion-grid>
      <ion-grid>
         <ion-row>
           <ion-col width-50>
             <ion-item>
               <ion-label stacked color="rojo-uam" >Hora Inicio</ion-label>
               <ion-datetime [(ngModel)]="horario.horaInicio" hourValues="{{horasPermitidas}}"
minuteValues="{{minutosPermitidos}}" cancelText="cancelar" doneText="ok" displayFormat="HH:mm"  ></ion-datetime>
             </ion-item>
           </ion-col>

            <ion-col width-50>
              <ion-item>
               <ion-label stacked color="rojo-uam" >Hora Final</ion-label>
               <ion-datetime [(ngModel)]="horario.horaFinal" hourValues="{{horasPermitidas}}"
minuteValues="{{minutosPermitidos}}" cancelText="cancelar" doneText="ok" displayFormat="HH:mm"  ></ion-datetime>
              </ion-item>
           </ion-col>
         </ion-row>
       </ion-grid>

<ion-item>
   <ion-label stacked>Asunto de la asesoría</ion-label>
   <ion-textarea [(ngModel)]="asunto"></ion-textarea>
 </ion-item>
 <br>
    <button ion-button color="rojo-uam" (click)="reservarCita()" full round>reserva tu cita</button>
   </div>
</div>
  <br><br>
</ion-content>

    <ion-footer >
      <ion-toolbar color="footer-uam">
         <ion-icon menuToggle class="btn-menu-footer" color="blanco" name="menu"></ion-icon>
      </ion-toolbar>
    </ion-footer>
```

programar-cita.ts

```
import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';
import {ConexionesREST} from "../../providers/conexiones-rest";
import { Storage } from '@ionic/storage';
```

```
/*
  Generated class for the ProgramarCitaPage page.

  See http://ionicframework.com/docs/v2/components/#navigation for more info on
  Ionic pages and navigation.
*/
export class HorarioDia{
  public dia: string;
  public horaInicio: string;
  public horaFinal: string;
}
export class HoraReservada{
  horaInicioR: string;
  horaFinalR: string;
}
export class HorasDisponibles{
  public dia: string;
  public horaInicio: string;
  public horaFinal: string;
  public horasReservadas :Array<HoraReservada>;

}
@Component({
  templateUrl: 'programar-cita.html',
})
export class ProgramarCitaPage {
  myDate:string;
  dia:any;
  hora: any;
  matricula: string;
  numeroEconomico: string;
  // horarioDias: any;
  horarioDia :Array<HorarioDia>;
  mostrarFechas: boolean;
  mostrarHoras:boolean;
  minutosPermitidos :String ="0,5,10,15,20,25,30,25,40,45,50,55";
  horasPermitidas : String ="07,08,09,10,11,12,13,14,15,16,17,18,19,20,21";
  horario = {horaInicio:"", horaFinal:""};
  horasDisponibles : HorasDisponibles;
  asunto: string;
  mensaje : string;

  constructor(public navCtrl: NavController,public storage: Storage, public conexion:ConexionesREST ) {
      this.storage.get("matricula").then((name) => {
      this.matricula=name;});
  }

  buscarProfesor(){
    this.conexion.getHorarioActual(this.numeroEconomico).subscribe(data=>{
      if(data.dias!=null){
         console.log(data);
      // this.horarioDias=data;
      this.horarioDia= data.dias;
      this.mostrarFechas= true;
      }else if(data.message!=null){
        alert(data.message);
      }
    });
  }

  buscarDisponibilidad(){
      this.conexion.getDisponibilidad(this.myDate+"-"+this.numeroEconomico).subscribe(data=>{
      // this.horarioDias=data;
      if(data.message==null){
        this.horasDisponibles= data;
```

```
      console.log(this.horasDisponibles);
      this.horario.horaInicio= this.horasDisponibles.horaInicio;
      this.horario.horaFinal = this.horasDisponibles.horaFinal;
      this.mostrarHoras=true;
      console.log(this.horasDisponibles.horasReservadas);
      console.log(this.horasDisponibles.horasReservadas.length);
      if(this.horasDisponibles.horasReservadas.length==0)
      {this.mensaje="aún no se han reservado citas";
      }else{
        this.mensaje="horas reservadas: ";
      }
    }else{
      alert(data.message);
    }
  });
 }

 reservarCita(){
   var cita = {horaInicio: this.horario.horaInicio,
              horaFinal:this.horario.horaFinal,
              asunto:this.asunto,
              fecha:this.myDate,
              matricula:this.matricula,
              numeroEconomico:this.numeroEconomico,
              dia: this.horasDisponibles.dia,
             };
     this.conexion.reservarCita(cita).subscribe(data=>{
       if(data.message!=null){
         alert(data.message);
             this.conexion.getProfesor(cita.numeroEconomico).subscribe(data=>{
               alert(JSON.stringify(data));
               if(data.token!=null){
                 var mensaje = "Nueva solicitud - " + cita.matricula + " : " + cita.asunto;
                 this.conexion.push(data.token,mensaje).subscribe(data=>{
                   alert(JSON.stringify(data));
                 });
               }

             });

         this.navCtrl.setRoot(ProgramarCitaPage);
       }

     });

 }
}
```

registro.html

```
<ion-header text-center>
  <ion-navbar color="rojo-uam">
    <ion-title>UAM Citas - Registro</ion-title>
  </ion-navbar>
</ion-header>

<ion-content class="degrade-principal" padding>
  <button ion-button full round color="blanco"  style="color:black" (click)="abrirRegistroEstudiante()" > Estudiante</button>
  <br>
  <button ion-button full round color="blanco" style="color:black" (click)="abrirRegistroProfesor()"> profesor</button>
</ion-content>
```

registro.ts

```
import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';
import { RegistroEstudiantePage} from "../registro-estudiante/registro-estudiante";
import {RegistroProfesorPage} from "../registro-profesor/registro-profesor";

// import {HttpService} from "../../services/http-service.ts"

/*
  Generated class for the RegistroPage page.

  See http://ionicframework.com/docs/v2/components/#navigation for more info on
  Ionic pages and navigation.
*/
@Component({
  templateUrl: 'registro.html',
})
export class RegistroPage {

  constructor(public navCtrl: NavController) {
  }

  abrirRegistroEstudiante(){
    this.navCtrl.push(RegistroEstudiantePage);
  }

  abrirRegistroProfesor(){
    this.navCtrl.push(RegistroProfesorPage);
  }

}
```

registro-estudiante.html

```
<ion-header text-center>
  <ion-navbar color="rojo-uam">
    <button  menuToggle>
      <ion-icon  name="menu"></ion-icon>
    </button>
    <ion-title>UAM Citas - Registro de Estudiante </ion-title>
  </ion-navbar>
</ion-header>


<ion-content class="degrade-principal" padding>
<form [formGroup]="valoresForm" (ngSubmit)="registrarEstudiante()">
  <ion-list>
    <ion-item>
    <ion-label>Matricula : </ion-label>
    <ion-input formControlName="matricula" type="text"></ion-input>
  </ion-item>

  <ion-item>
    <ion-label>Nombre : </ion-label>
    <ion-input formControlName="nombre" type="text"></ion-input>
  </ion-item>

  <ion-item>
    <ion-label>Apellido : </ion-label>
```

```
  <ion-input formControlName="apellido" type="text"></ion-input>
 </ion-item>

 <ion-item>
  <ion-label>Carrera : </ion-label>
  <ion-input formControlName="carrera" type="text"></ion-input>
 </ion-item>

 <ion-item>
  <ion-label>Email : </ion-label>
  <ion-input formControlName="correo" type="text"></ion-input>
 </ion-item>

 <ion-item>
  <ion-label>url de foto : </ion-label>
  <ion-input formControlName="urlFoto" type="text" placeholder="(opcional) copia el link para tu foto"></ion-input>
 </ion-item>

 <ion-item>
  <ion-label>Contraseña : </ion-label>
  <ion-input formControlName="contrasena" type="password"></ion-input>
 </ion-item>

</ion-list>

<div padding>
 <button ion-button type="submit" [disabled]="!valoresForm.valid" color="blanco" style="color:black" block>
Registrate</button>
</div>
</form>

</ion-content>
```

resistro-estudiante.ts

```
@Component({
 templateUrl: 'registro-estudiante.html',
})
export class RegistroEstudiantePage {
 valoresForm : any;

 constructor(public formBuilder : FormBuilder, public navCtrl: NavController, private conexion : ConexionesREST) {

   this.valoresForm = this.formBuilder.group({
   matricula: ["",Validators.required],
   nombre : ["",Validators.required],
   apellido : ["",Validators.required],
   carrera : ["",Validators.required],
   correo : ["",Validators.required],
   contrasena : ["",Validators.required],
   urlFoto : [""],
  });
 }

 irAInicio(){
  this.navCtrl.setRoot(InicioPage);
 }

 registrarEstudiante(){
  var datosRegistro = this.valoresForm.value;
  console.log(datosRegistro);
    this.conexion.registroEstudiante(datosRegistro).subscribe(data => {
    alert(data.message);
```

```
      this.navCtrl.setRoot(InicioPage);
      },
      err => {
         alert("problemas en el servidor");
      });
  }

}
```

registro-profesor.ts

```
<ion-header text-center>
  <ion-navbar color="rojo-uam">
    <button  menuToggle>
      <ion-icon  name="menu"></ion-icon>
    </button>
    <ion-title>UAM Citas - Registro de Profesor </ion-title>
  </ion-navbar>
</ion-header>

<ion-content class="degrade-principal" padding>

<form [formGroup]="valoresForm" (ngSubmit)="registrarProfesor()">
  <ion-list>
    <ion-item>
    <ion-label>Número Económico : </ion-label>
    <ion-input formControlName="numeroEconomico" type="text"></ion-input>
  </ion-item>

  <ion-item>
    <ion-label>Nombre : </ion-label>
    <ion-input formControlName="nombre" type="text"></ion-input>
  </ion-item>

  <ion-item>
    <ion-label>Apellido : </ion-label>
    <ion-input formControlName="apellido" type="text"></ion-input>
  </ion-item>

  <ion-item>
    <ion-label>Cubículo : </ion-label>
    <ion-input formControlName="cubiculo" type="text"></ion-input>
  </ion-item>

  <ion-item>
    <ion-label>Email : </ion-label>
    <ion-input formControlName="correo" type="text"></ion-input>
  </ion-item>

  <ion-item>
    <ion-label>url de foto : </ion-label>
    <ion-input formControlName="urlFoto" type="text" placeholder="(opcional) copia el link para tu foto"></ion-input>
  </ion-item>

  <ion-item>
    <ion-label>Contraseña : </ion-label>
    <ion-input formControlName="contrasena" type="password"></ion-input>
  </ion-item>

</ion-list>

<div padding>
  <button color="blanco" ion-button type="submit" [disabled]="!valoresForm.valid" blanco style="color:black" block>
Registrese</button>
```

```
</div>
</form>

</ion-content>
```

registro-profesor.html

```
import { Component } from '@angular/core';
import { NavController, MenuController } from 'ionic-angular';
import { InicioPage} from "../inicio/inicio";
import {ConexionesREST} from "../../providers/conexiones-rest";
import {FormBuilder, Validators } from '@angular/forms';
/*
  Generated class for the RegistroProfesorPage page.

  See http://ionicframework.com/docs/v2/components/#navigation for more info on
  Ionic pages and navigation.
*/
@Component({
  templateUrl: 'registro-profesor.html',
})
export class RegistroProfesorPage {
  valoresForm : any;

  constructor(public formBuilder : FormBuilder, public navCtrl: NavController, public menuCtrl: MenuController, private
conexion : ConexionesREST) {
     this.menuCtrl.enable(true,"menuProfesor");
        // desabilitar este menu o borror linea
     this.valoresForm = this.formBuilder.group({
     numeroEconomico: ["",Validators.required],
     nombre : ["",Validators.required],
     apellido : ["",Validators.required],
     cubiculo : ["",Validators.required],
     correo : ["",Validators.required],
     contrasena : ["",Validators.required],
     urlFoto : [""],
   });
  }

    abrirInicio(){
    this.navCtrl.setRoot(InicioPage);
    }

    registrarProfesor(){
      this.conexion.registroProfesor(this.valoresForm.value).subscribe(data => {
      alert(data.message);
      this.navCtrl.setRoot(InicioPage);

      },
      err => {
          alert("problemas en el servidor");
      });

    }


}
```

sesion-estudiante.html

```
<!--
  Generated template for the SesionEstudiantePage page.
```

```
  See http://ionicframework.com/docs/v2/components/#navigation for more info on
  Ionic pages and navigation.
-->
<ion-header text-center>
  <ion-navbar color="rojo-uam">

    <ion-title>Sesión Estudiante</ion-title>
  </ion-navbar>
</ion-header>


<ion-content>
      <ion-list>
        <ion-item>
          <ion-thumbnail item-left>
           <div *ngIf="alumno.urlFoto">
              <img src="{{alumno.urlFoto}}">
          </div>
          </ion-thumbnail>
          <h2>{{alumno.nombre}} {{alumno.apellido}}</h2>
          <h4>{{alumno.matricula}}</h4>
          <h4>{{alumno.carrera}}</h4>
        </ion-item>
      </ion-list>


    <ion-list>
      <ion-list-header><h5  text-center> Citas próximas.  </h5></ion-list-header>
      <div *ngIf="mensaje">
        <h4> {{mensaje}} </h4>
      </div>
      <div *ngFor="let cita of citasProximas">
      <ion-item>
        <ion-avatar item-left>
          <img src="{{cita.urlFoto}}">
        </ion-avatar>
        <h4>{{cita.dia}} - {{cita.fecha}}</h4>
        <h4>{{cita.nombreProfesor}} {{cita.apellidoProfesor}}</h4>
        <p>{{cita.asunto}}</p>
        <ion-note item-right>{{cita.horaInicio}}-</ion-note>
        <ion-note item-right>{{cita.horaFinal}}</ion-note>
        <p></p>
      </ion-item>
      </div>
    </ion-list>

      <div [hidden]="menuAbierto" menuToggle>
      </div>
</ion-content>

<ion-footer >
    <ion-toolbar color="footer-uam">
      <ion-icon menuToggle class="btn-menu-footer" color="blanco" name="menu"></ion-icon>
    </ion-toolbar>
</ion-footer>
```

sesion-profesor.ts

```
import { Component } from '@angular/core';
import { NavController,MenuController,NavParams  } from 'ionic-angular';
```

```
import { Storage } from '@ionic/storage';
import {ConexionesREST} from "../../providers/conexiones-rest";


/*
  Generated class for the SesionEstudiantePage page.

  See http://ionicframework.com/docs/v2/components/#navigation for more info on
  Ionic pages and navigation.
*/
export class CitasProximas{
  apellidoProfesor: string;
  asunto: string;
  cubiculo: string;
  dia: string ;
  fecha: string;
  horaFinal: string;
  horaInicio: string;
  nombreProfesor: string;
  urlFoto: string;
}

@Component({
  templateUrl: 'sesion-estudiante.html',
})
export class SesionEstudiantePage {
  alumno:any;
  menuAbierto:boolean;
  citasProximas:Array<CitasProximas>=[];
  mensaje:string;
  pruebaString : any;

  constructor(public conexion: ConexionesREST, public navCtrl: NavController,public menuCtrl: MenuController,public
params: NavParams,public storage : Storage) {
   this.menuCtrl.enable(true,"menuEstudiante");

    // es necesario definir el objeto alumno
    this.alumno = {
      matricula: "",
      nombre : "",
      apellido: "",
      carrera: "",
      correo: "",
      urlFoto: null,
    }

    //compara si es la primera vez que entra, ya que los datos son enviados como parametros
    if(this.params.get("matricula")){
      this.alumno = {
      matricula: this.params.get("matricula"),
      nombre : this.params.get("nombre"),
      apellido: this.params.get("apellido"),
      carrera: this.params.get("carrera"),
      correo: this.params.get("correo"),
      urlFoto: this.params.get("urlFoto"),
      }
      this.getCitasProximas(this.alumno.matricula);
    }else{ // ya estaba en sesión y los datos estan en disco

      this.storage.get("matricula").then((name) => {
          this.alumno.matricula=name;
          this.getCitasProximas(this.alumno.matricula);
        });

        this.storage.get("nombre").then((name) => {
          this.alumno.nombre=name;
```

```
      });

      this.storage.get("apellido").then((name) => {
        this.alumno.apellido=name;
      });

      this.storage.get("carrera").then((name) => {
        this.alumno.carrera=name;
      });

      this.storage.get("correo").then((name) => {
        this.alumno.correo=name;
      });

      this.storage.get("urlFoto").then((name) => {
        this.alumno.urlFoto=name;
      });
    }



}

getCitasProximas(matricula){
    this.conexion.getCitasProximasEstudiante(matricula).subscribe(data=>{
    console.log(data);
    if(data.message==null){
      console.log("guardar citas");
      this.citasProximas = data;
      console.log(this.citasProximas);
      if(this.citasProximas.length>0){
        this.mensaje=null;
      }else{
        this.mensaje="No hay citas cercanas";
      }
    }else{
      console.log(data.message);
      this.mensaje=null;
    }
  });

}
}
```

sesión-profesor.html

```
<!--
  Generated template for the SesionProfesorPage page.

  See http://ionicframework.com/docs/v2/components/#navigation for more info on
  Ionic pages and navigation.
-->
<ion-header text-center>
  <ion-navbar color="rojo-uam">
    <ion-title>Sesión Profesor</ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>
  <ion-list>
    <ion-item>
      <ion-thumbnail item-left>
        <div *ngIf="profesor.urlfoto">
          <img src="{{profesor.urlfoto}}">
        </div>
      </ion-thumbnail>
```

```
    <h2>{{profesor.nombre}} {{profesor.apellido}} </h2>
    <h4>{{profesor.cubiculo}} </h4>
   </ion-item>
  </ion-list>
<ion-list>
  <ion-list-header><h5 blanco text-center> Citas pendientes en este dia.  </h5></ion-list-header>
  <div *ngFor="let cita of citasProximas">
  <ion-item>
   <ion-avatar item-left>
    <img src="{{cita.urlFoto}}">
   </ion-avatar>
   <h3>{{cita.dia}} {{cita.fecha}}</h3>
   <h4>{{cita.nombre}} {{cita.apellido}}</h4>
   <p>{{cita.asunto}}</p>
   <ion-note item-right> {{cita.horaInicio}} </ion-note>
   <ion-note item-right> {{cita.horaFinal}} </ion-note>
  </ion-item>
</div>
</ion-list>

</ion-content>

<ion-footer >
   <ion-toolbar color="footer-uam">
    <ion-icon menuToggle class="btn-menu-footer" color="blanco" name="menu"></ion-icon>
   </ion-toolbar>
</ion-footer>
```

sesion-profesor.ts

```
import { Component } from '@angular/core';
import { NavController, MenuController,NavParams } from 'ionic-angular';
import { Storage } from '@ionic/storage';
import {ConexionesREST} from "../../providers/conexiones-rest";


/*
  Generated class for the SesionProfesorPage page.

  See http://ionicframework.com/docs/v2/components/#navigation for more info on
  Ionic pages and navigation.
*/
export class CitasProximas{
  apellido: string;
  asunto: string;
  matricula: string;
  dia: string ;
  fecha: string;
  horaFinal: string;
  horaInicio: string;
  nombre: string;
  urlFoto: string;
}

@Component({
  templateUrl: 'sesion-profesor.html',
})
export class SesionProfesorPage {
  public profesor : any;
  citasProximas:Array<CitasProximas>=[];
  mensaje:string;

  constructor(public conexion: ConexionesREST, public params: NavParams,public navCtrl: NavController, public menuCtrl:
MenuController,public storage : Storage) {
    this.menuCtrl.enable(true,"menuProfesor");
```

```
  this.profesor = {
    numeroEconomico: "",
    nombre : "",
    apellido: "",
    cubiculo: "",
    correo: "",
    urlfoto: null,
  }
  console.log("DATOS RECIBIDOS");
  console.log(this.params.data);

   if(this.params.get("numeroEconomico")){
    this.profesor = {
    numeroEconomico: this.params.get("numeroEconomico"),
    nombre : this.params.get("nombre"),
    apellido: this.params.get("apellido"),
    cubiculo: this.params.get("cubiculo"),
    correo: this.params.get("correo"),
    urlfoto: this.params.get("urlFoto"),
    }
    this.getCitasProximas(this.profesor.numeroEconomico);

  }else{
    console.log("no se envió nada,  buscar en disco");

    this.storage.get("numeroEconomico").then((name) => {
        this.profesor.numeroEconomico=name;
        this.getCitasProximas(this.profesor.numeroEconomico);
      });

      this.storage.get("nombre").then((name) => {
        this.profesor.nombre=name;
      });

      this.storage.get("apellido").then((name) => {
        this.profesor.apellido=name;
      });

      this.storage.get("cubiculo").then((name) => {
        this.profesor.carrera=name;
      });

      this.storage.get("correo").then((name) => {
        this.profesor.correo=name;
      });

      this.storage.get("urlFoto").then((name) => {
        this.profesor.urlfoto=name;
        console.log(this.profesor.urlfoto);
      });
  }
}

getCitasProximas(numeroEconomico){
    this.conexion.getCitasProximasProfesor(numeroEconomico).subscribe(data=>{
    console.log(data);
    if(data.message==null){
      console.log("guardar citas");
      this.citasProximas = data;
      console.log(this.citasProximas);
      if(this.citasProximas.length>0){
        this.mensaje=null;
      }else{
        this.mensaje="No hay citas cercanas";
      }
    }else{
```

```
      console.log(data.message);
      this.mensaje=null;
    }
  });
}
}
```

conexiones-rest.ts

```
import { Injectable } from '@angular/core';
import { Http,Headers } from '@angular/http';
import 'rxjs/add/operator/map';
import { Observable } from 'rxjs/Observable';

/*
  Generated class for the ConexionesREST provider.

  See https://angular.io/docs/ts/latest/guide/dependency-injection.html
  for more info on providers and Angular 2 DI.
*/
@Injectable()
export class ConexionesREST {
  urlServer: string = 'http://192.168.0.2:8080/citasUAM/rest/';
  constructor(public http: Http) {

    console.log('Hello ConexionesREST Provider');
  }

    login(datosSesion){
       var response = this.http.post(this.urlServer+"login",datosSesion).map(res => res.json());
                        return response;
    }

    registroEstudiante(datosRegistro){
       var response = this.http.post(this.urlServer+"estudiantes",datosRegistro).map(res => res.json());
       return response;
    }
    getEstudiante(matricula){
       var response = this.http.get(this.urlServer+"estudiantes/"+ matricula).map(res => res.json());
       return response;
    }
    registroProfesor(datosRegistro){
       var response = this.http.post(this.urlServer+"profesores",datosRegistro).map(res => res.json());
       return response;
    }

    getProfesor(numeroEconomico){
       var response = this.http.get(this.urlServer+"profesores/"+ numeroEconomico).map(res => res.json());
       return response;
    }
    registrarHorario(datosHorario){
      var response = this.http.post(this.urlServer+"horarioProfesores",datosHorario).map(res => res.json());
      return response;
    }

    getHorarioActual(numeroEconomico){
       var response = this.http.get(this.urlServer+"horarioProfesores/"+numeroEconomico).map(res => res.json());
      return response;
    }

    getDisponibilidad(param){
      var response = this.http.get(this.urlServer+"citas/"+param).map(res => res.json());
      return response;
    }
```

```
  reservarCita(cita){
    var response = this.http.post(this.urlServer+"citas",cita).map(res => res.json());
    return response;
  }

  AceptarCita(idCita){
    var response = this.http.put(this.urlServer+"citas",idCita).map(res => res.json());
                        return response;
  }

    getCitasProximasEstudiante(matricula: string){
    var response = this.http.get(this.urlServer+"citasProximasEstudiante/"+matricula).map(res => res.json());
    return response;
  }

    getCitasProximasProfesor(numeroEconomico : string){
    var response = this.http.get(this.urlServer+"citasProximasProfesor/"+numeroEconomico).map(res => res.json());
    return response;
  }

  push(token, mensaje){
    var header = new Headers({"Content-Type" : "application/json" , "Authorization" : "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJqdGkiOiI5ZmZkMGYwOC01MjBiLTQxMzItOGI2Ni1iOTZmZWI4ZGM0ZTUifQ.
CjQ_w9OSMl9vEplWmDHFWhramUaYehAyXw06yFlCEx4"});
     var json = {
       "tokens": [token],
       "profile": "citas",
       "notification": {
          "message": mensaje
       }
     }

    var response = this.http.post("https://api.ionic.io/push/notifications",json,{ headers: header }).map(res => res.json());
    return response;
  }
}
```

storage-providers.ts

```
import { Injectable } from '@angular/core';
import { Storage } from '@ionic/storage';



@Injectable()
export class StorageProvider {
   local: any;
 constructor() {
    this.local= new Storage();
 }

 saveData(key : string, data : String){
    this.local.set(key,data);
 }

 getData(key : string){
   this.local.get(key).then((name) => {
   console.log('Your name is', name);
   });
 }

 clearData(key :string){
    this.local.remove(key).then( ()=>{ console.log("se eliminó " + key)} );
```

```
  }


}
```

Hoja de estilo general

variables.scss

```
@import "ionic.globals";

$colors: (
  primary:    #387ef5,
  secondary:  #32db64,
  danger:     #f53d3d,
  light:      #f4f4f4,
  dark:       #222,
  favorite:   #69BB7B,
  rojo-uam: #BE1E2D,
  blanco: #fff,
  footer-uam: #7D0000,
);


.menu-title{
    height: 15%;
    color: white;
}
.boton-menu{
    height:12%;
    color: white;
}
.estatica{
    position: static;
}
.padding-icon{
    position : static;
    top: -100px;
    padding-left: 60px;
    padding-right: 60px;

}

.degrade-principal{
    background: rgba(241,102,102,1);
background: -moz-linear-gradient(top, rgba(241,102,102,1) 0%, rgba(176,49,49,1) 57%, rgba(71,0,0,1) 100%);
background: -webkit-gradient(left top, left bottom, color-stop(0%, rgba(241,102,102,1)), color-stop(57%, rgba(176,49,49,1)),
color-stop(100%, rgba(71,0,0,1)));
background: -webkit-linear-gradient(top, rgba(241,102,102,1) 0%, rgba(176,49,49,1) 57%, rgba(71,0,0,1) 100%);
background: -o-linear-gradient(top, rgba(241,102,102,1) 0%, rgba(176,49,49,1) 57%, rgba(71,0,0,1) 100%);
background: -ms-linear-gradient(top, rgba(241,102,102,1) 0%, rgba(176,49,49,1) 57%, rgba(71,0,0,1) 100%);
background: linear-gradient(to bottom, rgba(241,102,102,1) 0%, rgba(176,49,49,1) 57%, rgba(71,0,0,1) 100%);
filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#f16666', endColorstr='#470000', GradientType=0 );
}

.btn-blanco{
    background: white;
    color: black;
}
.sin-margen{
    position: relative;
    right: 10px;
}

.btn-menu-footer{
```

```
    position: relative;
    right: 15px;
    font-size: 25px;
}

.alto-completo{
    height:100%;
}
.separador-1{
    width:50%;
    height:5px;
    background:#ff0000;
}

.separador-2{
    width:80%;
    height:5px;
    background-color: red;
}

@import "ionic.theme.default";


// Ionicons
// --------------------------------------------------
// The premium icon font for Ionic. For more info, please see:
// http://ionicframework.com/docs/v2/ionicons/

$ionicons-font-path: "../assets/fonts";
@import "ionicons";
```

B. Código de los servicios web

Se exponen los códigos utilizados organizados en paquetes

default package

hibernate.cgf.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
```

```
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
    <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/citasacademicasuam?zeroDateTimeBehavior=convertToNull</
property>
    <property name="hibernate.connection.username">root</property>
    <mapping resource="modelo/Horariotrimestre.hbm.xml"/>
    <mapping resource="modelo/Estudiante.hbm.xml"/>
    <mapping resource="modelo/Profesor.hbm.xml"/>
    <mapping resource="modelo/Citas.hbm.xml"/>
    <mapping resource="modelo/Horariodias.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

hibernate.reveng.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate Reverse Engineering DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-reverse-engineering-3.0.dtd">
<hibernate-reverse-engineering>
  <schema-selection match-catalog="citasacademicasuam"/>
  <table-filter match-name="profesor"/>
  <table-filter match-name="citas"/>
  <table-filter match-name="horariotrimestre"/>
  <table-filter match-name="horariodias"/>
  <table-filter match-name="estudiante"/>
</hibernate-reverse-engineering>
```

Contenedores

CitasPendientes.java

```
public class CitasPendientes {
    private String numeroEconomico;
    private String idCita;
    private String horaInicio;
    private String horaFinal;
    private String asunto;
    private String estatus;
    private String fecha;
    private String horariodias_idHorario;
    private String estudiante_matricula;
    private String nombreEstudiante;

    public String getNombreEstudiante() {
        return nombreEstudiante;
    }

    public void setNombreEstudiante(String nombreEstudiante) {
        this.nombreEstudiante = nombreEstudiante;
    }
    public String getEstatus() {
        return estatus;
    }

    public void setEstatus(String estatus) {
        this.estatus = estatus;
    }
```

```
    public String getNumeroEconomico() {
        return numeroEconomico;
    }

    public void setNumeroEconomico(String numeroEconomico) {
        this.numeroEconomico = numeroEconomico;
    }

    public String getIdCita() {
        return idCita;
    }

    public void setIdCita(String idCita) {
        this.idCita = idCita;
    }

    public String getHoraInicio() {
        return horaInicio;
    }

    public void setHoraInicio(String horaInicio) {
        this.horaInicio = horaInicio;
    }

    public String getHoraFinal() {
        return horaFinal;
    }

    public void setHoraFinal(String horaFinal) {
        this.horaFinal = horaFinal;
    }

    public String getAsunto() {
        return asunto;
    }

    public void setAsunto(String asunto) {
        this.asunto = asunto;
    }

    public String getFecha() {
        return fecha;
    }

    public void setFecha(String fecha) {
        this.fecha = fecha;
    }

    public String getHorariodias_idHorario() {
        return horariodias_idHorario;
    }

    public void setHorariodias_idHorario(String horariodias_idHorario) {
        this.horariodias_idHorario = horariodias_idHorario;
    }

    public String getEstudiante_matricula() {
        return estudiante_matricula;
    }

    public void setEstudiante_matricula(String estudiante_matricula) {
        this.estudiante_matricula = estudiante_matricula;
    }

}
```

CitasProximas.java

```
public class CitasProximasJson {
    private String horaInicio;
    private String horaFinal;
    private String fecha;
    private String asunto;
    private String dia;
    private String nombreProfesor;
    private String apellidoProfesor;
    private String cubiculo;
    private String urlFoto;

    public String getHoraInicio() {
        return horaInicio;
    }

    public void setHoraInicio(String horaInicio) {
        this.horaInicio = horaInicio;
    }

    public String getHoraFinal() {
        return horaFinal;
    }

    public void setHoraFinal(String horaFinal) {
        this.horaFinal = horaFinal;
    }

    public String getFecha() {
        return fecha;
    }

    public void setFecha(String fecha) {
        this.fecha = fecha;
    }

    public String getAsunto() {
        return asunto;
    }

    public void setAsunto(String asunto) {
        this.asunto = asunto;
    }

    public String getDia() {
        return dia;
    }

    public void setDia(String dia) {
        this.dia = dia;
    }

    public String getNombreProfesor() {
        return nombreProfesor;
    }

    public void setNombreProfesor(String nombreProfesor) {
        this.nombreProfesor = nombreProfesor;
    }

    public String getApellidoProfesor() {
        return apellidoProfesor;
    }

    public void setApellidoProfesor(String apellidoProfesor) {
        this.apellidoProfesor = apellidoProfesor;
```

```
    }

    public String getCubiculo() {
        return cubiculo;
    }

    public void setCubiculo(String cubiculo) {
        this.cubiculo = cubiculo;
    }

    public String getUrlFoto() {
        return urlFoto;
    }

    public void setUrlFoto(String urlFoto) {
        this.urlFoto = urlFoto;
    }
}
```

Modelo

Citas.java

```
public class Citas  implements java.io.Serializable {


     private Integer idCita;
     private Estudiante estudiante;
     private Horariodias horariodias;
     private String horaInicio;
     private String horaFinal;
     private String asunto;
     private String estatus;
     private String fecha;

    public Citas() {
    }


    public Citas(Estudiante estudiante, Horariodias horariodias) {
        this.estudiante = estudiante;
        this.horariodias = horariodias;
    }
    public Citas(Estudiante estudiante, Horariodias horariodias, String horaInicio, String horaFinal, String asunto, String
estatus, String fecha) {
       this.estudiante = estudiante;
       this.horariodias = horariodias;
       this.horaInicio = horaInicio;
       this.horaFinal = horaFinal;
       this.asunto = asunto;
       this.estatus = estatus;
       this.fecha = fecha;
    }

    public Integer getIdCita() {
        return this.idCita;
    }

    public void setIdCita(Integer idCita) {
        this.idCita = idCita;
    }
    public Estudiante getEstudiante() {
        return this.estudiante;
    }
```

```
    public void setEstudiante(Estudiante estudiante) {
        this.estudiante = estudiante;
    }
    public Horariodias getHorariodias() {
        return this.horariodias;
    }

    public void setHorariodias(Horariodias horariodias) {
        this.horariodias = horariodias;
    }
    public String getHoraInicio() {
        return this.horaInicio;
    }

    public void setHoraInicio(String horaInicio) {
        this.horaInicio = horaInicio;
    }
    public String getHoraFinal() {
        return this.horaFinal;
    }

    public void setHoraFinal(String horaFinal) {
        this.horaFinal = horaFinal;
    }
    public String getAsunto() {
        return this.asunto;
    }

    public void setAsunto(String asunto) {
        this.asunto = asunto;
    }
    public String getEstatus() {
        return this.estatus;
    }

    public void setEstatus(String estatus) {
        this.estatus = estatus;
    }
    public String getFecha() {
        return this.fecha;
    }

    public void setFecha(String fecha) {
        this.fecha = fecha;
    }

}
```

Modelo.operaciones

NewHibernateUtil.java

```
import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.SessionFactory;

/**
 * Hibernate Utility class with a convenient method to get Session Factory
 * object.
 *
 * @author jordanks
 */
public class NewHibernateUtil {
```

```
    private static final SessionFactory sessionFactory;

    static {
        try {
            // Create the SessionFactory from standard (hibernate.cfg.xml)
            // config file.
            sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            // Log the exception.
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

OperacionesDB.java

```
package modelo.operaciones;

import contenedores.CitasPendientes;
import contenedores.HorarioDiasJson;
import contenedores.Login;
import contenedores.ModificacionCita;
import contenedores.TrimestreHorarioDias;
import java.util.ArrayList;
import java.util.Set;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import modelo.*;
import java.util.Date;
import java.util.List;
import org.hibernate.SQLQuery;


/**
 *
 * @author jordanks
 */
public class OperacionesDB {

    Session session;
    SessionFactory sesion;

    public OperacionesDB(){
    sesion = NewHibernateUtil.getSessionFactory();
    // se crea una fabrica de sesiones, se necesita un objeto de la clase Session
    session = sesion.openSession();
    }


    public Profesor buscarProfesor(String numEconomico){
    Profesor profe;
    Transaction tx = session.beginTransaction();
    // necesitamos la trasacion  porque si algo falla entre la transacion y el commit,  no se hace el comit por lo que no se
suben cambios
    profe = (Profesor) session.get(Profesor.class, numEconomico);
    tx.commit();
    return profe;
    }
```

```
    public Horariotrimestre buscarHorariosProfesor(Horariotrimestre horario){
        return horario;
    }

    public boolean registrarProfesor(Profesor profesor){

        Transaction tx = null;
        try {
        tx = session.beginTransaction();
           System.out.println("antes de la excepcion ");
        session.save(profesor);
        tx.commit();
        System.out.println("se inserto el profesor");
        return true;
        }
        catch (Exception e) {
        if (tx!=null) tx.rollback();
        System.out.println(" no se hizo nada");
        return false;
        }
    }

    public void registraCita(Citas cita){

       Transaction tx = session.beginTransaction();
        session.save(cita);
        tx.commit();
        System.out.println("se inserto el prestamo");

    }
    //---------------------Estudiantes----------------------------

    public Estudiante buscarEstudiante(String matricula){
    Estudiante estudianteBuscado;
    Transaction tx = session.beginTransaction();
    estudianteBuscado = (Estudiante) session.get(Estudiante.class, matricula);
    tx.commit();
    return estudianteBuscado;
    }
    // cambia los datos del estudiante, excepto el id, ya que el metodo update utiliza el id.
    //problema creo una sesion de hibernate global, en constructor de esta clase, cuando actualizo un Estudante queda
guardado en sesion
    // no es posible actualizarlo 2 veces porque el id del Estudainte queda asociado en la sesion
    // posible solucion ,borrar de la sesion al estudiante al hacer update.

    public boolean actualizarEstudiante(Estudiante est){
        Transaction tx = null;
        try {
            System.out.println(est.getMatricula()+" "+ est.getNombre() +" "+ est.getCorreo());
            tx = session.beginTransaction();
            System.out.println("incia");
            session.update(est);
            System.out.println("upadte");
            tx.commit();
            return true;

        }catch (RuntimeException e) {
            System.out.println(e);
            if (tx != null) {
                tx.rollback();
            }
            return false;
        }
    }
```

```
   public boolean registrarEstudiante(Estudiante estudiante){
      Transaction tx = null;
      try {
      tx = session.beginTransaction();
      session.save(estudiante);
      tx.commit();
      return true;
      }
      catch (Exception e) {
        if (tx!=null) tx.rollback();
            return false;
      }
   }

   //Métodos para registrar horarios de profesor
      public Horariotrimestre registrarHorario(TrimestreHorarioDias nuevoTrimestre){
      Profesor profesor;
      Transaction tx = null;
      try {
      tx = session.beginTransaction();
      //obtiene json, encuentra el numero economico para buscar al profesor
      String trimestreActual = nuevoTrimestre.getTrimestre();
      session.clear();
      profesor = (Profesor) session.get(Profesor.class, nuevoTrimestre.getNumeroEconomico());
      System.out.println("profesor encontrado" + profesor.getNombre());

      //se crea el trimestre y se guarda el trimestre
      Horariotrimestre horarioTrimestre = new Horariotrimestre();
      horarioTrimestre.setProfesor(profesor);
      horarioTrimestre.setTrimestreActual(trimestreActual);
        //  session.save(horarioTrimestre) debería regresar el id del horarioTrimestre pero no lo hace y es necesario para
guardar los dias de asesoria
        //entonces antes de guardar se busca el ultimo id en la tabla horariodias con la siguiente query
        List result;
        result =session.createSQLQuery("SELECT horariodias.idHorario from horariodias ORDER BY horariodias.idHorario
DESC LIMIT 1").list();
        if(result.isEmpty()){
           System.out.println(" esta vacio");
           int id=1;
           horarioTrimestre.setIdHorarioTrimestre(id);
        }else{
        int id=(int)result.get(0);
        System.out.println("resultado de query " + id);
        horarioTrimestre.setIdHorarioTrimestre(id);
        }
        //al tener el id ya es posible asignarlo con metodo set y conservarlo en memoria
        System.out.println("id " + horarioTrimestre.getIdHorarioTrimestre());

        session.save(horarioTrimestre);
        tx.commit();

         System.out.println("Se guardo el horario");

         return horarioTrimestre;
      }
    catch (Exception e) {
      if (tx!=null) tx.rollback();
      throw e;
    }
   }

      public boolean registrarHorarioDias(Horariodias horarioDia){
       Transaction tx = null;
          try {
          tx = session.beginTransaction();
          session.save(horarioDia);
```

```
          //una vez guardado el trimestre se interan los dias de asesoria que se recibieron en el json
           tx.commit();
           session.clear();
           return true;
           }

          catch (Exception e) {
            if (tx!=null) tx.rollback();
                throw e;
          }
        }

    public boolean reservarCita(Citas cita){
       Transaction tx = null;
      try {
       tx = session.beginTransaction();
       session.save(cita);
       tx.commit();
       return true;
       }
    catch (Exception e) {
       if (tx!=null) tx.rollback();
           return false;
    }
   }

      public String[][] getCitasPendientes(String  numeroEconomico){
       Transaction tx = null;
      try {
       tx = session.beginTransaction();
       List result;
        String consulta = "select profesor.numeroEconomico,citas.* , estudiante.nombre, estudiante.apellido from profesor join
horariotrimestre on profesor.numeroEconomico= horariotrimestre.profesor_numeroEconomico "
       + "join horariodias on horariotrimestre.idHorarioTrimestre =horariodias.horariotrimestre_idHorarioTrimestre join citas on
horariodias.idHorario = citas.horariodias_idHorario join estudiante on estudiante.matricula = citas.estudiante_matricula "
              +  " WHERE estatus = 'pendiente' and profesor.numeroEconomico='"+numeroEconomico+"' ";

        // esta es la manera en que conseguí guardar los resultados de una consulta, se itera el objeto list resultado y luego
        //se guarda en un arreglo de [n][9] donde n es el numero de filas restultantes de la consulta
        result =session.createSQLQuery(consulta).list();
        String arreglo[][]= new String[result.size()][11];

          List<Object[]> entities = result;
          int i=0;
          int j=0;

    for (Object[] entity : entities) {
       j=0;
       for (Object entityCol : entity) {
           System.out.print(" " + entityCol);
           arreglo[i][j]=entityCol.toString();
           j++;
       }
       i++;
    }
       tx.commit();
          System.out.println("todo bien se encontraron las citas pendientes");
       return arreglo;
       }
    catch (Exception e) {
       if (tx!=null) tx.rollback();
           return null;
    }
   }

      public boolean cambioCita(ModificacionCita cambio){
```

```
        Transaction tx = null;
       try {
        tx = session.beginTransaction();
        String update = "UPDATE citas SET estatus = '"+cambio.getEstatus()+"' WHERE citas.idCita =
'"+cambio.getIdCita()+"'";
        SQLQuery sqlQuery=session.createSQLQuery(update);
        sqlQuery.executeUpdate();
        tx.commit();
        return true;
        }
     catch (Exception e) {
        if (tx!=null) tx.rollback();
            return false;
     }
    }

      public String[][] citasProximas(String matricula, String[] fechas){
        Transaction tx = null;
       try {
        tx = session.beginTransaction();
        String consulta = "select * from citas where citas.estudiante_matricula='2113001882' and citas.estatus='aceptada' and "
              + "(citas.fecha='"+fechas[0]+"' or citas.fecha='"+fechas[1]+"' or citas.fecha='"+fechas[2]+"')";

        String query ="select citas.horaInicio, citas.horaFinal, citas.fecha, citas.asunto, dia,profesor.nombre , profesor.apellido ,
profesor.cubiculo, profesor.urlFoto from citas join horariodias on citas.horariodias_idHorario= horariodias.idHorario "
              + "join horariotrimestre on horariodias.horariotrimestre_idHorarioTrimestre = horariotrimestre.idHorarioTrimestre
join profesor on horariotrimestre.profesor_numeroEconomico = profesor.numeroEconomico "
              + "where citas.estudiante_matricula='"+matricula+"' and citas.estatus='aceptada' and (citas.fecha='"+fechas[0]+"'
or citas.fecha='"+fechas[1]+"' or citas.fecha='"+fechas[2]+"')";
        List result;
        result=session.createSQLQuery(query).list();

       String arreglo[][]= new String[result.size()][10];

          List<Object[]> entities = result;
          int i=0;
          int j=0;

    for (Object[] entity : entities) {
       j=0;
       for (Object entityCol : entity) {
           System.out.print(" " + entityCol);
           if(entityCol!=null)
           arreglo[i][j]=entityCol.toString();

           System.out.println(arreglo[i][j]);
           j++;
       }
       i++;
    }
          System.out.println("todo bien hasta qui");
       tx.commit();
       return arreglo;
       }
     catch (Exception e) {
       if (tx!=null) tx.rollback();
           System.out.println("Fallo en consulta de citas proximas");
           return null;
     }
    }


      public String[][] citasProximasProfesor(String numeroEconomico, String[] fechas){
        Transaction tx = null;
       try {
        tx = session.beginTransaction();
```

```
        String query ="SELECT citas.horaInicio, citas.horaFinal, citas.asunto, citas.fecha, dia, estudiante.matricula,
estudiante.nombre, estudiante.apellido, estudiante.urlFoto FROM citas join estudiante on citas.estudiante_matricula =
estudiante.matricula "
                + "join horariodias on citas.horariodias_idHorario = horariodias.idHorario "
                + "join horariotrimestre on horariodias.horariotrimestre_idHorarioTrimestre= horariotrimestre.idHorarioTrimestre
where citas.estatus='aceptada' and "
                + "horariotrimestre.profesor_numeroEconomico='"+numeroEconomico+"' and (citas.fecha='"+fechas[0]+"' or
citas.fecha='"+fechas[1]+"' or citas.fecha='"+fechas[2]+"') ";

        List result;
        result=session.createSQLQuery(query).list();

       String arreglo[][]= new String[result.size()][10];

          List<Object[]> entities = result;
          int i=0;
          int j=0;

    for (Object[] entity : entities) {
        j=0;
        for (Object entityCol : entity) {
            System.out.print(" " + entityCol);
            if(entityCol!=null)
            arreglo[i][j]=entityCol.toString();

            System.out.println(arreglo[i][j]);
            j++;
        }
        i++;
    }
           System.out.println("todo bien hasta qui");
        tx.commit();
        return arreglo;
        }
     catch (Exception e) {
        if (tx!=null) tx.rollback();
            System.out.println("Fallo en consulta de citas proximas");
            return null;
     }
    }

       public boolean tokenEstudiante(Login login){
        Transaction tx = null;
       try {
        tx = session.beginTransaction();
        String update = "UPDATE estudiante SET token = '"+ login.getToken()+ "' WHERE estudiante.matricula =
'"+login.getId()+"'";
        SQLQuery sqlQuery=session.createSQLQuery(update);
        sqlQuery.executeUpdate();
        tx.commit();
        return true;
        }
     catch (Exception e) {
        if (tx!=null) tx.rollback();
            return false;
        }
       }

      public boolean tokenProfesor(Login login){
        Transaction tx = null;
       try {
        tx = session.beginTransaction();
        String update = "update profesor set token= '"+login.getToken()+"' where profesor.numeroEconomico =
'"+login.getId()+"'";
        SQLQuery sqlQuery=session.createSQLQuery(update);
```

```
        sqlQuery.executeUpdate();
        tx.commit();
        return true;
        }
     catch (Exception e) {
        if (tx!=null) tx.rollback();
            return false;
     }
    }

}
```

Rest

CitasProximasProfesorRest.java

```
package rest;

import com.google.gson.Gson;
import contenedores.CitasProximasJson;
import contenedores.CitasProximasProfesorJson;
import contenedores.HorarioDiasJson;
import contenedores.HorarioReturnJson;
import java.io.IOException;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.UriInfo;
import javax.ws.rs.Consumes;
import javax.ws.rs.Produces;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.PUT;
import javax.ws.rs.PathParam;
import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerResponseContext;
import javax.ws.rs.container.ContainerResponseFilter;
import javax.ws.rs.core.MediaType;
import contenedores.TrimestreHorarioDias;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Locale;
import java.util.Set;
import modelo.Horariodias;
import modelo.Horariotrimestre;
import modelo.Profesor;
import modelo.operaciones.OperacionesDB;

/**
 * REST Web Service
 *
 * @author jordanks
 */

//aqui van todos los métodos para la api rest
@Path("citasProximasProfesor")
public class CitasProximasProfesorRest implements ContainerResponseFilter{

    @Context
    private UriInfo context;

    private Gson gson;
    OperacionesDB operDB ;
    /**
```

```
 * Creates a new instance of GenericResource
 */
public CitasProximasProfesorRest() {
    operDB= new OperacionesDB();
    gson = new Gson();
}
//este metodo regresa el ultimo horario del profesor con sus dias de asesorias
//busca el id mas alto, luego convierte la info a un objeto HorarioReturnJson para que se posible enviarlo como un Json
@GET @Path("{id}")
@Produces({ MediaType.TEXT_PLAIN,MediaType.APPLICATION_JSON })
public String findById(@PathParam("id") String id) {

/*    String year;
    String month;
    String day;
    year = id.substring(0, 4);
    month = id.substring(5, 7);
    day= id.substring(8, 10);*/
    String numeroEconomico ;
    numeroEconomico= id;

    String[] fechas= new String[3];

    for(int i=0;i<3;i++){
        Calendar cal = Calendar.getInstance();
        cal.add(Calendar.DATE,i);
        SimpleDateFormat format1 = new SimpleDateFormat("yyyy-MM-dd");

        String formatted = format1.format(cal.getTime());
        fechas[i]= formatted;
        System.out.println(i+" " + fechas[i]);
    }
    String horario[][]=operDB.citasProximasProfesor(numeroEconomico,fechas);
    if(horario.length!=0){
    ArrayList <CitasProximasProfesorJson> citasProximas = new ArrayList();

    for(int i=0; i<horario.length; i++){
         System.out.println(i);
        CitasProximasProfesorJson citaProxima= new CitasProximasProfesorJson();

        citaProxima.setHoraInicio(horario[i][0]);
        citaProxima.setHoraFinal(horario[i][1]);
        citaProxima.setAsunto(horario[i][2]);
        citaProxima.setFecha(horario[i][3]);
        citaProxima.setDia(horario[i][4]);
        citaProxima.setMatricula(horario[i][5]);
        citaProxima.setNombre(horario[i][6]);
        citaProxima.setApellido(horario[i][7]);
        citaProxima.setUrlFoto(horario[i][8]);

        citasProximas.add(citaProxima);
        System.out.println(citaProxima.getAsunto() + " " + citaProxima.getFecha());
    }
    return gson.toJson(citasProximas);
    }else{
    return "{\"code\":400, \"message\":\"Sin Citas Proximas \" }";
    }
    }

/**
 * Retrieves representation of an instance of rest.GenericResource
 * @return an instance of java.lang.String
 */
@GET
@Produces(MediaType.TEXT_PLAIN)
public String get() {
```

```
        //TODO return proper representation object
        return "esto devuelve una cita programada ";
    }

        @POST
    public String post(String data){
            return "{\"code\":200, \"message\":\"error al registrar horario \" }";
    }

    @PUT
    @Consumes(MediaType.APPLICATION_XML)
    public void putXml(String content) {
    }

    @Override
    public void filter(ContainerRequestContext requestContext, ContainerResponseContext responseContext) throws
IOException {
        responseContext.getHeaders().putSingle("Access-Control-Allow-Origin", "*");
        responseContext.getHeaders().putSingle("Access-Control-Allow-Credentials", "true");
        responseContext.getHeaders().putSingle("Access-Control-Allow-Methods", "GET, POST, DELETE, PUT");
        responseContext.getHeaders().putSingle("Access-Control-Allow-Headers", "Content-Type, Accept");
    }
}
```

CitasRest.java

```
import com.google.gson.Gson;
import contenedores.CitasPendientes;
import contenedores.DisponibilidadCitasJson;
import contenedores.HorarioDiasJson;
import contenedores.HorarioReturnJson;
import contenedores.HorasReservadas;
import contenedores.ListaCitasPendientes;
import contenedores.ModificacionCita;
import contenedores.ReservacionCitaJson;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Set;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.UriInfo;
import javax.ws.rs.Consumes;
import javax.ws.rs.Produces;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.PUT;
import javax.ws.rs.PathParam;
import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerResponseContext;
import javax.ws.rs.container.ContainerResponseFilter;
import javax.ws.rs.core.MediaType;
import modelo.Citas;
import modelo.Estudiante;
import modelo.Horariodias;
import modelo.Horariotrimestre;
import modelo.Profesor;
import modelo.operaciones.OperacionesDB;

/**
 * REST Web Service
 *
 * @author jordanks
```

```
 */

//aqui van todos los métodos para la api rest
@Path("citas")
public class CitasRest implements ContainerResponseFilter{

    @Context
    private UriInfo context;

    private Gson gson;
    OperacionesDB operDB ;
    /**
     * Creates a new instance of GenericResource
     */
    public CitasRest() {
        operDB= new OperacionesDB();
        gson = new Gson();
    }

     @GET @Path("{id}")
    @Produces({ MediaType.TEXT_PLAIN,MediaType.APPLICATION_JSON })
    public String findById(@PathParam("id") String id) {

        int longitud=5;
        //envio de numero economico
        if(id.length()<=longitud){
            String numeroEconomico= id;
            System.out.println("numero E: "+ numeroEconomico);
            String datos[][]=operDB.getCitasPendientes(numeroEconomico);

            if(datos!=null){
                ListaCitasPendientes listaCitas = new ListaCitasPendientes();

                ArrayList <CitasPendientes> listaCita = new ArrayList();//contenedor para guardar citas
                CitasPendientes citas;
                for(int i=0; i< datos.length; i++){
                  citas = new CitasPendientes();
                  citas.setNumeroEconomico(datos[i][0]);
                  citas.setIdCita(datos[i][1]);
                  citas.setHoraInicio(datos[i][2]);
                  citas.setHoraFinal(datos[i][3]);
                  citas.setAsunto(datos[i][4]);
                  citas.setEstatus(datos[i][5]);
                  citas.setFecha(datos[i][6]);
                  citas.setHorariodias_idHorario(datos[i][7]);
                  citas.setEstudiante_matricula(datos[i][8]);
                  citas.setNombreEstudiante(datos[i][9]+" " +datos[i][10]);
                  listaCita.add(citas);
                }

                listaCitas.setListaCitas(listaCita);

                //System.out.println(citas.getNumeroEconomico() + " " + citas.getIdCita() + " " + citas.getHoraInicio() + " " +
citas.getHoraFinal());
                //System.out.println( citas.getAsunto()+ " " + citas.getEstatus() + " " + citas.getFecha()+ " "+
citas.getHorariodias_idHorario() ) ;
                //System.out.println("tamaño es" + datos.length);

                return gson.toJson(listaCitas);

            }
            //return "{\"code\":200, \"message\":\" numeroEconomico \" }";

        }else{
            System.out.println("buscar por disponibilidad en fecha");
            //return "{\"code\":200, \"message\":\" fecha \" }";
```

```
        }
        String year;
        String month;
        String day;
        year = id.substring(0, 4);
        month = id.substring(5, 7);
        day= id.substring(8, 10);
        String numeroEconomico ;
        numeroEconomico= id.substring(11);


        Calendar now = Calendar.getInstance();
        now.set(Integer.parseInt(year), Integer.parseInt(month)-1, Integer.parseInt(day));

    //System.out.println("Current date : " + (now.get(Calendar.MONTH) + 1) + "-"+ now.get(Calendar.DATE) + "-" +
now.get(Calendar.YEAR));

    String[] strDays = new String[] { "domingo", "lunes", "martes", "miercoles", "jueves",
        "viernes", "sabado" };
    // Day_OF_WEEK starts from 1 while array index starts from 0
    System.out.println("dia para buscar citas: " + strDays[now.get(Calendar.DAY_OF_WEEK) - 1]);
    String diaCita = strDays[now.get(Calendar.DAY_OF_WEEK) - 1];
    //------------
          Profesor profesor= operDB.buscarProfesor(numeroEconomico);

          System.out.println("profesor " + profesor.getApellido());
           //comprueba que el profesor tiene horarios
          if(!profesor.getHorariotrimestres().isEmpty()){
           Set <Horariotrimestre> horarioTrimestre= profesor.getHorariotrimestres();
          //regresar horario con id mas alto
           Horariotrimestre horarioActual= null;
           int idFor=0;
           for( Horariotrimestre  h : horarioTrimestre )
           {
              if(h.getIdHorarioTrimestre() > idFor){
                idFor=h.getIdHorarioTrimestre();
                horarioActual=h;
              }
           }
           System.out.println("Actual" + horarioActual.getTrimestreActual());

           //buscar si tiene el dia jueves
           Set<Horariodias> horarioDias= horarioActual.getHorariodiases();
           for( Horariodias horarioDia : horarioDias){
              if(horarioDia.getDia().equals(diaCita)){
                 System.out.println("existe el dia¡¡¡");
                 System.out.println(horarioDia.getHoraInicial()+" - "+ horarioDia.getHoraFinal());

                 DisponibilidadCitasJson disponible = new DisponibilidadCitasJson();
                 HorasReservadas horas ;
                 Set <Citas> citas =horarioDia.getCitases();

                 ArrayList <HorasReservadas> horasReservadas = new ArrayList();//contenedor para regresar las horas
reservadas

                 for(Citas cita : citas)
                 {
                    System.out.println("***"+ cita.getHoraInicio() + " " + cita.getHoraFinal() + " " + cita.getEstatus());
                    if(cita.getEstatus().equals("aceptada ")){
                       horas = new HorasReservadas();
                       horas.setHoraInicioR(cita.getHoraInicio());
                       horas.setHoraFinalR(cita.getHoraFinal());
                       horasReservadas.add(horas);
                       System.out.println(" se agregó al arrego");
                    }
```

```
            }
            disponible.setHoraInicio(horarioDia.getHoraInicial());
            disponible.setHoraFinal(horarioDia.getHoraFinal());

            disponible.setDia(diaCita);
            disponible.setHorasReservadas(horasReservadas);
            return gson.toJson(disponible);

            //return "{\"code\":200, \"message\":\" jueves \" }";

          }else{
            System.out.println("no existe");

          }
       }

     }
//--------
            return "{\"code\":400, \"message\":\" no hay asesoría el "+diaCita+" \" }";
   }

/**
 * Retrieves representation of an instance of rest.GenericResource
 * @return an instance of java.lang.String
 */
@GET
@Produces(MediaType.TEXT_PLAIN)
public String get() {
    //TODO return proper representation object
    return "esto devuelve una cita";
}

   @POST
public String post(String data){
       System.out.println(data  );
       ReservacionCitaJson cita = gson.fromJson(data, ReservacionCitaJson.class);
       Estudiante estudiante = operDB.buscarEstudiante(cita.getMatricula());
       Profesor profesor = operDB.buscarProfesor(cita.getNumeroEconomico());
       Horariodias horaDia = null ;//se guardará el dia en el que se agenda

       //----buscar horario
       if(!profesor.getHorariotrimestres().isEmpty()){
       Set <Horariotrimestre> horarioTrimestre= profesor.getHorariotrimestres();
       //regresar horario con id mas alto
       Horariotrimestre horarioActual= null;
       int idFor=0;
       for( Horariotrimestre  h : horarioTrimestre )
       {
          if(h.getIdHorarioTrimestre() > idFor){
            idFor=h.getIdHorarioTrimestre();
            horarioActual=h;
          }
       }
       System.out.println("Actual" + horarioActual.getTrimestreActual());
       //buscar si tiene el dia especificado
       Set<Horariodias> horarioDias= horarioActual.getHorariodiases();
       for( Horariodias horarioDia : horarioDias){
          System.out.println(" dia : " + horarioDia.getDia());
          if(horarioDia.getDia().equals(cita.getDia())){
             horaDia= horarioDia;
             System.out.println("existe el dia¡¡¡");
             break;
          }else{
             System.out.println("no existe");
             //return "{\"code\":400, \"message\":\"error: En el dia especificada no hay asesorías  \" }";
```

```
            }
        }

        Citas miCita =  new Citas();
        miCita.setEstudiante(estudiante);
        miCita.setHorariodias(horaDia);

        miCita.setAsunto(cita.getAsunto());
        miCita.setEstatus("pendiente");
        miCita.setFecha(cita.getFecha());
        miCita.setHoraInicio(cita.getHoraInicio());
        miCita.setHoraFinal(cita.getHoraFinal());

        operDB.reservarCita(miCita);

        return "{\"code\":200, \"message\":\"petición de reservación enviada, solo espera la confirmación del profesor \" }";

        }else{
            return "{\"code\":400, \"message\":\"error: el profesor no ha registrado horarios.  \" }";

        }
        //--.-----

   //    if ( operDB.registrarProfesor(nuevoProfe))
     //  else
   }

   /**
    * PUT method for updating or creating an instance of GenericResource
    * @param content representation for the resource
    */
   @PUT

   public String put(String data) {
       System.out.println(data);
       ModificacionCita cambioCita = gson.fromJson(data, ModificacionCita.class);
       System.out.println("Solicitud put");
       if(operDB.cambioCita(cambioCita)){
           return gson.toJson(cambioCita) ;
       }else{
            return "{\"code\":400, \"message\":\"Error al modificar el estatus  \" }";

       }

   }

   @Override
   public void filter(ContainerRequestContext requestContext, ContainerResponseContext responseContext) throws
IOException {
       responseContext.getHeaders().putSingle("Access-Control-Allow-Origin", "*");
       responseContext.getHeaders().putSingle("Access-Control-Allow-Credentials", "true");
       responseContext.getHeaders().putSingle("Access-Control-Allow-Methods", "GET, POST, DELETE, PUT");
       responseContext.getHeaders().putSingle("Access-Control-Allow-Headers", "Content-Type, Accept");
   }
}
```

EstudianteRest.java

```
import com.google.gson.Gson;
import java.io.IOException;
import javax.ws.rs.core.Context;
```

```
import javax.ws.rs.core.UriInfo;
import javax.ws.rs.Consumes;
import javax.ws.rs.Produces;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.PUT;
import javax.ws.rs.PathParam;
import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerResponseContext;
import javax.ws.rs.container.ContainerResponseFilter;
import javax.ws.rs.core.MediaType;
import modelo.Estudiante;
import modelo.Profesor;
import modelo.operaciones.OperacionesDB;

/**
 * REST Web Service
 *
 * @author jordanks
 */

//aqui van todos los métodos para la api rest
@Path("estudiantes")
public class EstudianteRest implements ContainerResponseFilter{

    @Context
    private UriInfo context;

    private Gson gson;
    OperacionesDB operDB ;
    public EstudianteRest() {
        operDB= new OperacionesDB();
        gson = new Gson();
    }

    /**
     * Retrieves representation of an instance of rest.GenericResource
     * @return an instance of java.lang.String
     */
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String get() {
        //TODO return proper representation object

        return "busca un estudiante";
    }
    //https://www.mkyong.com/webservices/jax-rs/jax-rs-pathparam-example/ sobre PathParam
    //el volor de {id} se toma de la uri www...estudiante/{id}
    @GET @Path("{id}")
    @Produces({ MediaType.TEXT_PLAIN,MediaType.APPLICATION_JSON })
    public String findById(@PathParam("id") String id) {
        Estudiante est=operDB.buscarEstudiante(id);
        if( est!=null){
            est.setCitases(null);
            return gson.toJson(est);
        }else
             return "{\"code\":200, \"message\":\"No Existe el estudiante \" }";
        }

        @POST
    public String post(String data){
        Estudiante nuevoEst = gson.fromJson(data, Estudiante.class);

            System.out.println(data);
        System.out.println("se va a registrar" + " mat: " +  nuevoEst.getMatricula() +" nombre:  "+ nuevoEst.getNombre());
```

```
        if ( operDB.registrarEstudiante(nuevoEst)){
            return "{\"code\":200, \"message\":\"registro exitoso, ahora inicia sesión \" }";
        }
        else
            return "{\"code\":400, \"message\":\"error al registrar probablemente ya existe \" }";

    }

    @PUT
    public String put(String data) {
        System.out.println("se recibió un put");
        Estudiante actEst = gson.fromJson(data, Estudiante.class);
        if ( operDB.actualizarEstudiante(actEst))
            return "se actualizó el estudiante";
        else
            return "error al actualizar";
    }

    @Override
    public void filter(ContainerRequestContext requestContext, ContainerResponseContext responseContext) throws
IOException {
        responseContext.getHeaders().putSingle("Access-Control-Allow-Origin", "*");
        responseContext.getHeaders().putSingle("Access-Control-Allow-Credentials", "true");
        responseContext.getHeaders().putSingle("Access-Control-Allow-Methods", "GET, POST, DELETE, PUT");
        responseContext.getHeaders().putSingle("Access-Control-Allow-Headers", "Content-Type, Accept");
    }
}
```

HorarioProfesorRest.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package rest;

import com.google.gson.Gson;
import contenedores.HorarioDiasJson;
import contenedores.HorarioReturnJson;
import java.io.IOException;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.UriInfo;
import javax.ws.rs.Consumes;
import javax.ws.rs.Produces;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.PUT;
import javax.ws.rs.PathParam;
import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerResponseContext;
import javax.ws.rs.container.ContainerResponseFilter;
import javax.ws.rs.core.MediaType;
import contenedores.TrimestreHorarioDias;
import java.util.ArrayList;
import java.util.Set;
import modelo.Horariodias;
import modelo.Horariotrimestre;
import modelo.Profesor;
import modelo.operaciones.OperacionesDB;
```

```
/**
 * REST Web Service
 *
 * @author jordanks
 */

@Path("horarioProfesores")
public class HorarioProfesorRest implements ContainerResponseFilter{

    @Context
    private UriInfo context;

    private Gson gson;
    OperacionesDB operDB ;
    public HorarioProfesorRest() {
        operDB= new OperacionesDB();
        gson = new Gson();
    }
    //este metodo regresa el ultimo horario del profesor con sus dias de asesorias
    //busca el id mas alto, luego convierte la info a un objeto HorarioReturnJson para que se posible enviarlo como un Json
    @GET @Path("{id}")
    @Produces({ MediaType.TEXT_PLAIN,MediaType.APPLICATION_JSON })
    public String findById(@PathParam("id") String id) {

        Profesor profesor= operDB.buscarProfesor(id);
        if(profesor!=null){
            HorarioReturnJson horarioReturn = new HorarioReturnJson();//objeto a retornar en json
            ArrayList <HorarioDiasJson> listaDias = new ArrayList();//contenedor para regresar los dias

            //comprueba que el profesor tiene horarios
           if(!profesor.getHorariotrimestres().isEmpty()){
            Set <Horariotrimestre> horarioTrimestre= profesor.getHorariotrimestres();
           //regresar horario con id mas alto
            Horariotrimestre horarioActual= null;
            int idFor=0;
            for( Horariotrimestre  h : horarioTrimestre )
            {
                System.out.println(h.getTrimestreActual()+ "      " + h.getIdHorarioTrimestre());
                if(h.getIdHorarioTrimestre() > idFor){
                    System.out.println("id: " + h.getIdHorarioTrimestre() + " idFor: " + idFor);
                   idFor=h.getIdHorarioTrimestre();
                   horarioActual=h;
                }
            }
            System.out.println("Actual" + horarioActual.getTrimestreActual());
            horarioReturn.setTrimestreActual( horarioActual.getTrimestreActual());

            Set<Horariodias> horarioDias= horarioActual.getHorariodiases();
            for( Horariodias horarioDia : horarioDias){
                System.out.println(" dia : " + horarioDia.getDia());
                HorarioDiasJson horarioj = new
HorarioDiasJson(horarioDia.getDia(),horarioDia.getHoraInicial(),horarioDia.getHoraFinal());
                listaDias.add(horarioj);
            }
            horarioReturn.setDias(listaDias);

            return gson.toJson(horarioReturn);
           }else{
            return "{\"code\":400, \"message\":\"No hay horarios registrados \" }";
           }
        }else
        return "{\"code\":400, \"message\":\"No existe el profesor \" }";
        }

    /**
     * Retrieves representation of an instance of rest.GenericResource
```

```
     * @return an instance of java.lang.String
     */
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String get() {
        //TODO return proper representation object
        return "esto devuelve un Horario de profesor";

    }

        @POST
    public String post(String data){
        TrimestreHorarioDias nuevoTrimestre = gson.fromJson(data, TrimestreHorarioDias.class);
            System.out.println(data);

            //primero ser recibe un json con toda la información para crera un horario, se usa la clase TrimestreHorarioDias
como contenedor para recibir el json
            System.out.println("numero economico "+ nuevoTrimestre.getNumeroEconomico()  +" trimestre "
+nuevoTrimestre.getTrimestre() +"  horario " + nuevoTrimestre.getHorariodia().get(0).getDia());
            Integer numeroEconomico= Integer.parseInt(nuevoTrimestre.getNumeroEconomico());
            //se crea el nuevo trimestre enviando el objeto creado
            Horariotrimestre horarioTrimestre = operDB.registrarHorario(nuevoTrimestre);
            //se uso un truquito especial para obtener el id de horarioTrimestre, ver comentarios en metodo registrarHorario de
OperacionesDB

            ArrayList <HorarioDiasJson> horarioDiasJson = nuevoTrimestre.getHorariodia();
            boolean resultadoExitoso= true;
        //horarioDiasJson es otra clase contenedora que ayuda a iterar los dias  de asesoria recibidos

                for(HorarioDiasJson horarioDiaJson : horarioDiasJson )
        {
        // horarioDia tiene un dia de asesoria con sus respectivas horas
        Horariodias horarioDia = new Horariodias();
        // este el el bueno el objeto que se guardará en la base de datos
        horarioDia.setDia(horarioDiaJson.getDia());
        horarioDia.setHoraInicial(horarioDiaJson.getHoraInicio());
        horarioDia.setHoraFinal(horarioDiaJson.getHoraFinal());
        horarioDia.setHorariotrimestre(horarioTrimestre);
        System.out.println(" el horario a guardar " + horarioDia.getDia() + " "+ horarioDia.getHoraFinal() + " "+
horarioDia.getHoraInicial());
        if(!operDB.registrarHorarioDias(horarioDia))
            resultadoExitoso= false;

        System.out.println("se guardo el horarioDia");
        //System.out.println(horarioDias.getDia());
        }


            System.out.println("id horario " + horarioTrimestre.getIdHorarioTrimestre());

        if ( resultadoExitoso)
            return "{\"code\":200, \"message\":\"se registraron los horarios \" }";
        else
            return "{\"code\":400, \"message\":\"error al registrar horario \" }";


    }


    /**
     * PUT method for updating or creating an instance of GenericResource
     * @param content representation for the resource
     */
    @PUT
    @Consumes(MediaType.APPLICATION_XML)
    public void putXml(String content) {
```

```
    }

    @Override
    public void filter(ContainerRequestContext requestContext, ContainerResponseContext responseContext) throws
IOException {
        responseContext.getHeaders().putSingle("Access-Control-Allow-Origin", "*");
        responseContext.getHeaders().putSingle("Access-Control-Allow-Credentials", "true");
        responseContext.getHeaders().putSingle("Access-Control-Allow-Methods", "GET, POST, DELETE, PUT");
        responseContext.getHeaders().putSingle("Access-Control-Allow-Headers", "Content-Type, Accept");
    }
}
```

HorarioRest.java

```
package rest;

import com.google.gson.Gson;
import contenedores.CitasProximasJson;
import contenedores.HorarioDiasJson;
import contenedores.HorarioReturnJson;
import java.io.IOException;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.UriInfo;
import javax.ws.rs.Consumes;
import javax.ws.rs.Produces;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.PUT;
import javax.ws.rs.PathParam;
import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerResponseContext;
import javax.ws.rs.container.ContainerResponseFilter;
import javax.ws.rs.core.MediaType;
import contenedores.TrimestreHorarioDias;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Locale;
import java.util.Set;
import modelo.Horariodias;
import modelo.Horariotrimestre;
import modelo.Profesor;
import modelo.operaciones.OperacionesDB;

/**
 * REST Web Service
 *
 * @author jordanks
 */

//aqui van todos los métodos para la api rest
@Path("citasProximasEstudiante")
public class HorarioRest implements ContainerResponseFilter{

    @Context
    private UriInfo context;

    private Gson gson;
    OperacionesDB operDB ;
    /**
     * Creates a new instance of GenericResource
     */
    public HorarioRest() {
```

```
    operDB= new OperacionesDB();
    gson = new Gson();
}
//este metodo regresa el ultimo horario del profesor con sus dias de asesorias
//busca el id mas alto, luego convierte la info a un objeto HorarioReturnJson para que se posible enviarlo como un Json
@GET @Path("{id}")
@Produces({ MediaType.TEXT_PLAIN,MediaType.APPLICATION_JSON })
public String findById(@PathParam("id") String id) {

    String matricula ;
    matricula= id;

    String[] fechas= new String[3];

    for(int i=0;i<3;i++){
        Calendar cal = Calendar.getInstance();
        cal.add(Calendar.DATE,i);
        SimpleDateFormat format1 = new SimpleDateFormat("yyyy-MM-dd");

        String formatted = format1.format(cal.getTime());
        fechas[i]= formatted;
        System.out.println(i+" " + fechas[i]);
    }
    String horario[][]=operDB.citasProximas(matricula,fechas);
    if(horario.length!=0){
    ArrayList <CitasProximasJson> citasProximas = new ArrayList();

    for(int i=0; i<horario.length; i++){
         System.out.println(i);
        CitasProximasJson citaProxima= new CitasProximasJson();

        citaProxima.setHoraInicio(horario[i][0]);
        citaProxima.setHoraFinal(horario[i][1]);
        citaProxima.setFecha(horario[i][2]);
        citaProxima.setAsunto(horario[i][3]);
        citaProxima.setDia(horario[i][4]);
        citaProxima.setNombreProfesor(horario[i][5]);
        citaProxima.setApellidoProfesor(horario[i][6]);
        citaProxima.setCubiculo(horario[i][7]);
        citaProxima.setUrlFoto(horario[i][8]);

        citasProximas.add(citaProxima);
        System.out.println(citaProxima.getAsunto() + " " + citaProxima.getFecha());
    }
    return gson.toJson(citasProximas);
    }else{
    return "{\"code\":400, \"message\":\"Sin Citas Próximas \" }";
    }


    }


@GET
@Produces(MediaType.TEXT_PLAIN)
public String get() {
    //TODO return proper representation object
    return "esto devuelve una cita programada ";
}

    @POST
public String post(String data){
        return "{\"code\":200, \"message\":\"Error al registrar horario \" }";

}
```

```
    /**
     * PUT method for updating or creating an instance of GenericResource
     * @param content representation for the resource
     */
    @PUT
    @Consumes(MediaType.APPLICATION_XML)
    public void putXml(String content) {
    }

    @Override
    public void filter(ContainerRequestContext requestContext, ContainerResponseContext responseContext) throws
IOException {
        responseContext.getHeaders().putSingle("Access-Control-Allow-Origin", "*");
        responseContext.getHeaders().putSingle("Access-Control-Allow-Credentials", "true");
        responseContext.getHeaders().putSingle("Access-Control-Allow-Methods", "GET, POST, DELETE, PUT");
        responseContext.getHeaders().putSingle("Access-Control-Allow-Headers", "Content-Type, Accept");
    }
}
```

ProfesorRest.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package rest;

import com.google.gson.Gson;
import java.io.IOException;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.UriInfo;
import javax.ws.rs.Consumes;
import javax.ws.rs.Produces;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.PUT;
import javax.ws.rs.PathParam;
import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerResponseContext;
import javax.ws.rs.container.ContainerResponseFilter;
import javax.ws.rs.core.MediaType;
import modelo.Profesor;
import modelo.operaciones.OperacionesDB;

/**
 * REST Web Service
 *
 * @author jordanks
 */

//aqui van todos los métodos para la api rest
@Path("profesores")
public class ProfesorRest implements ContainerResponseFilter{

    @Context
    private UriInfo context;

    private Gson gson;
    OperacionesDB operDB ;
    /**
     * Creates a new instance of GenericResource
     */
```

```
    public ProfesorRest() {
        operDB= new OperacionesDB();
        gson = new Gson();
    }

     @GET @Path("{id}")
    @Produces({ MediaType.TEXT_PLAIN,MediaType.APPLICATION_JSON })
    public String findById(@PathParam("id") String id) {
         System.out.println("id " + id);
        Profesor profesor= operDB.buscarProfesor(id);
        if( profesor!=null){
            profesor.setHorariotrimestres(null);
            return gson.toJson(profesor);
        }else
        return "{\"code\":200, \"message\":\"No existe el profesor \" }";

        }

    /**
     * Retrieves representation of an instance of rest.GenericResource
     * @return an instance of java.lang.String
     */
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String get(String men) {
        //TODO return proper representation object
        System.out.println("men" + men);
        return "{\"code\":200, \"message\":\"busca un profesor \" }";
    }

        @POST
    public String post(String data){
            System.out.println(data);
        Profesor nuevoProfe = gson.fromJson(data, Profesor.class);
        if( nuevoProfe.getNombre()!=null){

        if ( operDB.registrarProfesor(nuevoProfe))
            return "{\"code\":200, \"message\":\"registro exitoso, ahora inicia sesión \" }";
        else
            return "{\"code\":200, \"message\":\"error al registrar probablemente ya existe \" }";
        }else{
            return "{\"code\":200, \"message\":\"error datos incorrectos \" }";
        }


    }


    @PUT
    @Consumes(MediaType.APPLICATION_XML)
    public void putXml(String content) {
    }

    @Override
    public void filter(ContainerRequestContext requestContext, ContainerResponseContext responseContext) throws
IOException {
        responseContext.getHeaders().putSingle("Access-Control-Allow-Origin", "*");
        responseContext.getHeaders().putSingle("Access-Control-Allow-Credentials", "true");
        responseContext.getHeaders().putSingle("Access-Control-Allow-Methods", "GET, POST, DELETE, PUT");
        responseContext.getHeaders().putSingle("Access-Control-Allow-Headers", "Content-Type, Accept");
    }
}
```

LoginRest.java

```
package rest;
import com.google.gson.Gson;
import java.io.IOException;

import javax.ws.rs.core.Context;
import javax.ws.rs.core.UriInfo;
import javax.ws.rs.Consumes;
import javax.ws.rs.Produces;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.PUT;
import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerResponseContext;
import javax.ws.rs.core.MediaType;
import modelo.Estudiante;
import contenedores.Login;
import modelo.Profesor;
import modelo.operaciones.OperacionesDB;


/**
 * REST Web Service
 *
 * @author jordanks
 */
// codigo para hacer el login
@Path("login")
public class loginRest {

    @Context
    private UriInfo context;
        OperacionesDB operDB;
        Gson gson;
    /**
     * Creates a new instance of login
     */
    public loginRest() {
        operDB= new OperacionesDB();
        gson = new Gson();
    }

    /**
     * Retrieves representation of an instance of rest.login
     * @return an instance of java.lang.String
     */
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String getText() {
        //TODO return proper representation object
        return "login get";
    }
    //recibe un json y con id y contraseña
     @POST
    public String post(String data){
        //crea un objeto login con los datos enviados y luego comprueba el login
        System.out.println(data);
        Login login = gson.fromJson(data, Login.class);
        System.out.println("login " + login.getId() + login.getContrasena() + login.getToken());
        Estudiante est = operDB.buscarEstudiante(login.getId());
        if(est != null){
            if(est.getContrasena().equals(login.getContrasena())){
                //inicio de sesión agregar token
                if(login.getToken()!=null){
                    operDB.tokenEstudiante(login);
```

```
                }
                est.setCitases(null); // para que no exista error en el parser gson
                 System.out.println("se logeo un estudiante");
                   return gson.toJson(est);
                }else{
                   return "{\"code\":400, \"message\":\"Contraseña incorrecta \" }";
                 }
            }else{
                Profesor profe = operDB.buscarProfesor(login.getId());
                System.out.println("se busco profesor");
                if(profe!=null){
                    System.out.println(profe.getContrasena()+ " "+ login.getContrasena());
                    if(profe.getContrasena().equals(login.getContrasena())){
                       if(login.getToken()!=null){
                           System.out.println("se va a regitrar el token ");
                          operDB.tokenProfesor(login);
                        }
                        profe.setHorariotrimestres(null);// genera error en  el parser profesor tiene horarios
                        return gson.toJson(profe);
                    }else{
                       return "{\"code\":400, \"message\":\"Contraseña incorrecta \" }";
                     }

                }else{
                    return "{\"code\":400, \"message\":\"Error usuario o contraseña inválidos \" }";
                }
            }

    }

    /**
     * PUT method for updating or creating an instance of login
     * @param content representation for the resource
     */
    @PUT
    @Consumes(MediaType.TEXT_PLAIN)
    public void putText(String content) {
    }

    public void filter(ContainerRequestContext requestContext, ContainerResponseContext responseContext) throws
IOException {
        responseContext.getHeaders().putSingle("Access-Control-Allow-Origin", "*");
        responseContext.getHeaders().putSingle("Access-Control-Allow-Credentials", "true");
        responseContext.getHeaders().putSingle("Access-Control-Allow-Methods", "GET, POST, DELETE, PUT");
        responseContext.getHeaders().putSingle("Access-Control-Allow-Headers", "Content-Type, Accept");
    }
}
```

C. Script de la base de datos

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema citasacademicasuam
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema citasacademicasuam
-- -----------------------------------------------------
```

```
CREATE SCHEMA IF NOT EXISTS `citasacademicasuam` DEFAULT CHARACTER SET utf8 ;
USE `citasacademicasuam` ;

-- -----------------------------------------------------
-- Table `citasacademicasuam`.`profesor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `citasacademicasuam`.`profesor` (
  `numeroEconomico` VARCHAR(15) NOT NULL,
  `nombre` VARCHAR(45) NULL,
  `apellido` VARCHAR(45) NULL,
  `cubiculo` VARCHAR(45) NULL,
  `correo` VARCHAR(90) NULL,
  `urlFoto` VARCHAR(350) NULL,
  `contrasena` VARCHAR(45) NULL,
  `token` VARCHAR(250) NULL,
  PRIMARY KEY (`numeroEconomico`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `citasacademicasuam`.`horariotrimestre`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `citasacademicasuam`.`horariotrimestre` (
  `idHorarioTrimestre` INT NOT NULL AUTO_INCREMENT,
  `trimestreActual` VARCHAR(45) NULL,
  `profesor_numeroEconomico` VARCHAR(15) NOT NULL,
  PRIMARY KEY (`idHorarioTrimestre`),
  INDEX `fk_horariotrimestre_profesor1_idx` (`profesor_numeroEconomico` ASC),
  CONSTRAINT `fk_horariotrimestre_profesor1`
    FOREIGN KEY (`profesor_numeroEconomico`)
    REFERENCES `citasacademicasuam`.`profesor` (`numeroEconomico`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `citasacademicasuam`.`horariodias`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `citasacademicasuam`.`horariodias` (
  `idHorario` INT NOT NULL AUTO_INCREMENT,
  `dia` VARCHAR(45) NULL,
  `horaFinal` VARCHAR(45) NULL,
  `horaInicial` VARCHAR(45) NULL,
  `horariotrimestre_idHorarioTrimestre` INT NOT NULL,
  PRIMARY KEY (`idHorario`),
  INDEX `fk_horariodias_horariotrimestre1_idx` (`horariotrimestre_idHorarioTrimestre` ASC),
  CONSTRAINT `fk_horariodias_horariotrimestre1`
    FOREIGN KEY (`horariotrimestre_idHorarioTrimestre`)
    REFERENCES `citasacademicasuam`.`horariotrimestre` (`idHorarioTrimestre`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `citasacademicasuam`.`estudiante`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `citasacademicasuam`.`estudiante` (
  `matricula` VARCHAR(15) NOT NULL,
  `nombre` VARCHAR(45) NULL,
```

```
  `apellido` VARCHAR(45) NULL,
  `carrera` VARCHAR(45) NULL,
  `correo` VARCHAR(45) NULL,
  `urlFoto` VARCHAR(350) NULL,
  `contrasena` VARCHAR(45) NULL,
  `token` VARCHAR(250) NULL,
  PRIMARY KEY (`matricula`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `citasacademicasuam`.`citas`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `citasacademicasuam`.`citas` (
  `idCita` INT NOT NULL AUTO_INCREMENT,
  `horaInicio` VARCHAR(45) NULL,
  `horaFinal` VARCHAR(45) NULL,
  `asunto` VARCHAR(45) NULL,
  `estatus` VARCHAR(45) NULL,
  `fecha` VARCHAR(45) NULL,
  `horariodias_idHorario` INT NOT NULL,
  `estudiante_matricula` VARCHAR(15) NOT NULL,
  PRIMARY KEY (`idCita`),
  INDEX `fk_citas_horariodias1_idx` (`horariodias_idHorario` ASC),
  INDEX `fk_citas_estudiante1_idx` (`estudiante_matricula` ASC),
  CONSTRAINT `fk_citas_horariodias1`
    FOREIGN KEY (`horariodias_idHorario`)
    REFERENCES `citasacademicasuam`.`horariodias` (`idHorario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_citas_estudiante1`
    FOREIGN KEY (`estudiante_matricula`)
    REFERENCES `citasacademicasuam`.`estudiante` (`matricula`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```