UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Título del Proyecto:

Proyecto de Registro de Citas Médicas

Autor:

Cáceres Zurita, Ishmael Santiago

Director:

Girones, Jesús Tomás

TESINA PARA LA OBTENCIÓN DEL TÍTULO DE:

Máster en Desarrollo de Aplicaciones sobre Dispositivos Móviles

Septiembre del 2015

Contenido

Introducción

Descripción del problema

Partiendo de la problemática que genera la obtención de citas médicas y la poca organización de la información, siendo esta ultima la más importante y sin dejar de lado el costo que implica la movilización y el tiempo invertido para la obtención de una cita médica, surge la idea de desarrollar una aplicación que minimice estos inconvenientes.

Al no contar los médicos y los pacientes con una comunicación adicional que no sea la de reservar citas vía telefónica, con el problema que conlleva este, como puede ser la perdida de información, o al no contar los pacientes con una lista de médicos e instituciones de salud que puedan darle sus servicios en lugares cercanos a su domicilio, genera malestar en los usuarios y perdidas económicas para las instituciones y médicos, ya que no se dan a conocer o su publicidad está limitada.

Gracias a que las soluciones desarrolladas en aplicaciones para dispositivos móviles dentro del Ecuador han ido en aumento en los últimos años, y a la creciente demanda de desarrollos sobre estas tecnologías, se ha tomado la decisión, por parte de VlipCode, en realizar una aplicación que cubra una parte del mercado en cuanto al sector de salud respecta.

Objetivos

Construir una aplicación para dispositivos móviles con sistema operativo Android que permita a los potenciales usuarios la obtención de una cita médica de forma fácil e inmediata.

Llevar a cabo la integración de la aplicación móvil con el sistema web, para facilitar la gestión de las citas registradas, de esta manera poder obtener la información en tiempo real, mediante una conexión a internet.

Facilitar a los médicos o instituciones de salud, su registro en el sistema para que por medio de la aplicación puedan llegar a más usuarios, siendo una nueva forma de interacción entre el médico y el paciente.

Motivación

Vlipcode es una empresa nueva, donde uno de sus principales objetivos es la venta de software como servicio, de esta forma brindar la posibilidad al cliente de obtener un software que maneje la lógica de su negocio con un costo menor al que conllevaría el desarrollo completo de la solución.

Mi relación con la empresa es de socio, y me encuentro realizando las funciones de desarrollo y mantenimiento de aplicaciones web empresariales bajo la plataforma JEE, últimamente se está incursionando en el desarrollo de aplicaciones móviles, para complementar la funcionalidad de las aplicaciones web, donde se toma el proyecto Registro de Citas Médicas como un comienzo.

Tecnologías utilizadas

Dentro de las tecnologías utilizadas para complementar el desarrollo en Android, se fundamenta principalmente en la plataforma JEE en su versión 7, utilizando las especificaciones:

JPA 2.1 para la capa de datos,

EJB 3.2 para la lógica del negocio,

JAX-RS 2.0 para la construcción de servicios web REST

La librería de google, gson-2.3.1 para parsear los POJOs (de *Plain Old Java Object*) a JSON y viceversa.

Maven para la construcción automática del artefacto WAR (de *Web Application Archive*), el cual establece la comunicación entre los dispositivos móviles y el servidor de aplicaciones.

Glassfish 4.1 como servidor de aplicaciones.

Adicionalmente cabe indicar que se utiliza POSTGRESQL 9.4 como motor de base de datos.

JEE7 es una poderosa y completa plataforma de desarrollo de aplicaciones para múltiples sistemas operativos. Que nos permite desarrollar todo tipo de sistemas informáticos tanto móviles, desktop, Web, SOAP, entre otros. Entre las características de JEE7 tenemos:

Casi todo se puede hacer con anotaciones y desaparece la necesidad de usar descriptores.

Se incluyen especificaciones para normalizar los nombres Java Native Directory Interface - JNDI de los EJBs (lo que era siempre un problema entre servidores) y se incluye el concepto de EmbeddedContainer para poder probar unitariamente los EJBs.

Se incluyen nuevas especificaciones como por ejemplo RESTFul Webservices (JAX-RS).
(Oracle)

En cuanto a las herramientas de desarrollo se utilizó Android Studio para el componente móvil y NetBeans IDE para el desarrollo de los servicios web.

Arquitectura de la aplicación

Esquema del diseño

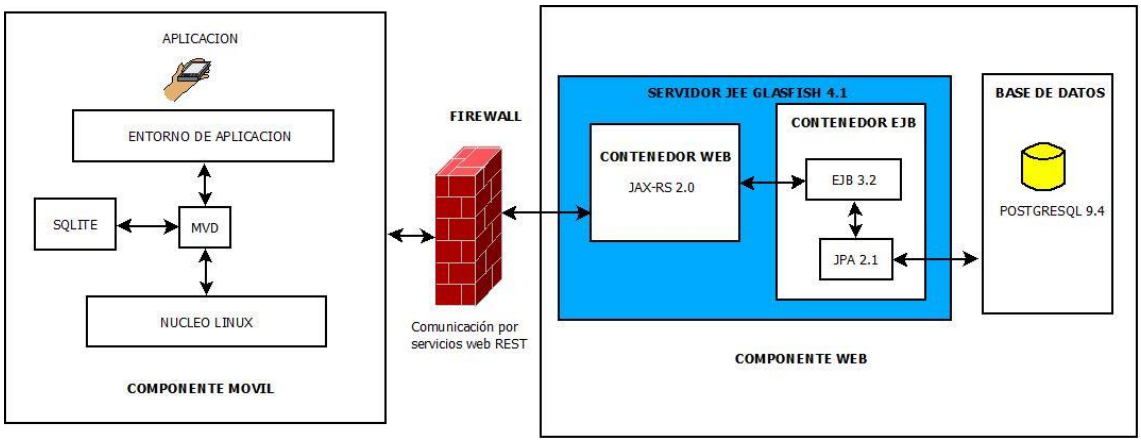
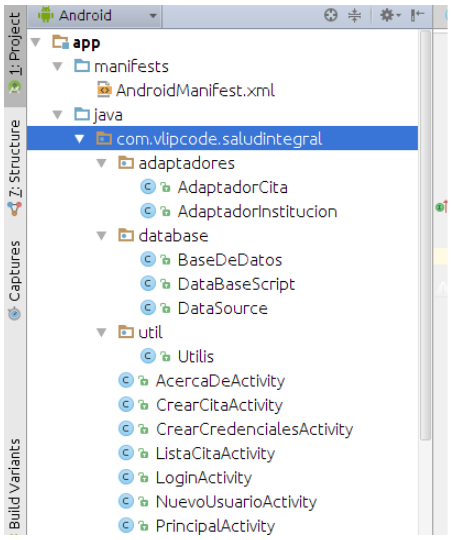


Diagrama general de la arquitectura del proyecto

Distribución de paquetes.

Para el componente móvil se ha utilizado la siguiente distribución de paquetes:



Paquetes componente móvil

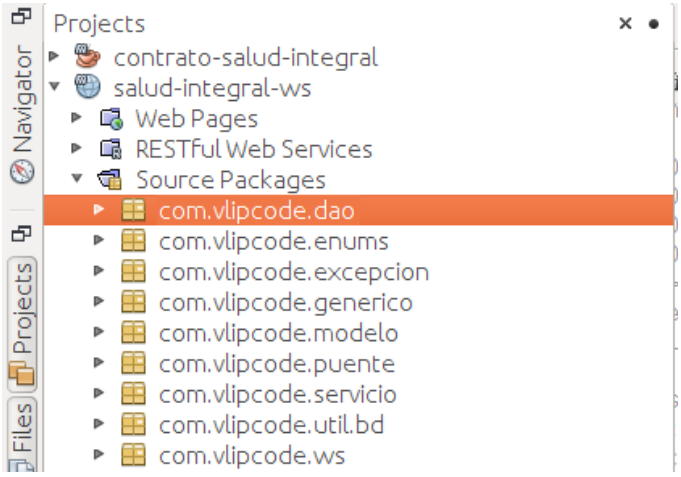
com.vlipcode.saludintegral: Este paquete contiene todas las actividades de la aplicación.

com.vlipcode.saludintegral.adaptadores: Esta paquete contiene los adaptadores para personalizar los elementos de las listas, así como también los elementos de los spinners.

com.vlipcode.saludintegral.database: Este paquete contiene las clases de scripts de base de datos (DataBaseScript), manejo de sentencias CRUD (DataSource) y la clase BaseDeDatos que gestiona la conexión con la base de datos SQLite.

com.vlipcode.saludintegral.util: Paquete que contiene la clase Utilis la que es utilizada para validaciones y manejo de fechas.

Para el componente web de servicios web se tiene la siguiente distribución de paquetes:



Paquetes componente web

com.vlipcode.modelo: Dentro de este paquete están todas las clases mapeadas a la base de datos, el nombre de estas clases coinciden con el de las tablas de bases de datos.

com.vlipcode.generico: Dentro de este paquete se encuentra la clase encargada de manejar las transacciones de Create Update Delete

(Crud) la misma que recibe como parámetro cualquier objeto de tipo modelo, esta clase heredada a todos los Ejbs Daos.

com.vlipcode.dao: Dentro de este paquete están todas las clases de acceso a la base de, se nombran especificando el nombre de la clase modelo y con la terminación Dao, por ejemplo si vamos a procesar la tabla rol de una base de datos, el nombre de la clase seria RolDao. Estas clases son EJBs del tipo Stateless.

com.vlipcode.puente: Dentro de este paquete están todas las clases de tipo puente las mismas que heredan de su correspondiente clase Dao.

com.vlipcode.servicio: Dentro de este paquete se encuentran todas las clases de servicio las mismas que serán accedidas por los servicios web, estas clases son EJBs de tipo Stateless, y heredan las clases puente correspondiente, su nombre debe terminar con la palabra Servicio, por ejemplo si vamos a hacer un servicio de la tabla rol sería RolServicio.

com.vlipcode.ws: Dentro de este paquete se encuentran todas las clases de servicios web, estas clases deben usar anotaciones de tipo @Singleton indicando que es un EJB de tipo singleton, @Path el path para acceder al recurso, además de consumir y producir objetos JSON en cada uno de sus métodos, su nombre debe terminar con la palabra **Ws.**

com.vlipcode.enums: En este paquete se encuentran los enums que contienen la definición de constantes, que usa la aplicación.

com.vlipcode.excepcion: En este paquete se encuentran las clases de tipo excepción personalizada.

com.vlipcode.util: En este paquete están todas las clases de utilitarios de lógica de negocio como son ayudas de fechas, conexiones a otras bases de datos, etc.

Descripción de librerías

Dentro de las librerías utilizadas se encuentra las siguientes:

gson-2.3.1: Librería de google, utilizada para parsear los POJOs que se transmiten por los servicios web en formato JSON y viceversa.

contrato-salud-integral-1.0: Librería que contiene los POJOs utilizados para la comunicación entre el componente web y el móvil, para transmitir la información, así como los mensajes en caso de éxito o error, esta librería es la misma que utiliza el componente web para garantizar el contrato en los servicios web.

Patrones de diseño utilizados

En el componente web se han utilizado los siguientes patrones de diseño:

Data Access Object (DAO): Son los encargados de almacenar y obtener objetos de cualquier repositorio persistente de datos, como por ejemplo una base de datos relacional.

En la arquitectura planteada estos DAOs son implementados utilizando el API de JPA, dentro de un Session Bean de tipo **Stateless**.

Bridge (Puente): Es una técnica usada en programación para desacoplar una abstracción de su implementación, de manera que ambas puedan ser modificadas independientemente sin necesidad de alterar por ello la otra.
En la arquitectura planteada estos puentes son utilizados para enlazar la capa Dao con la de Servicio.

Data Transfer Object: Es un patrón que se basa en la transferencia de datos de una base de datos o de un archivo a un POJO para ser procesado en una capa superior.

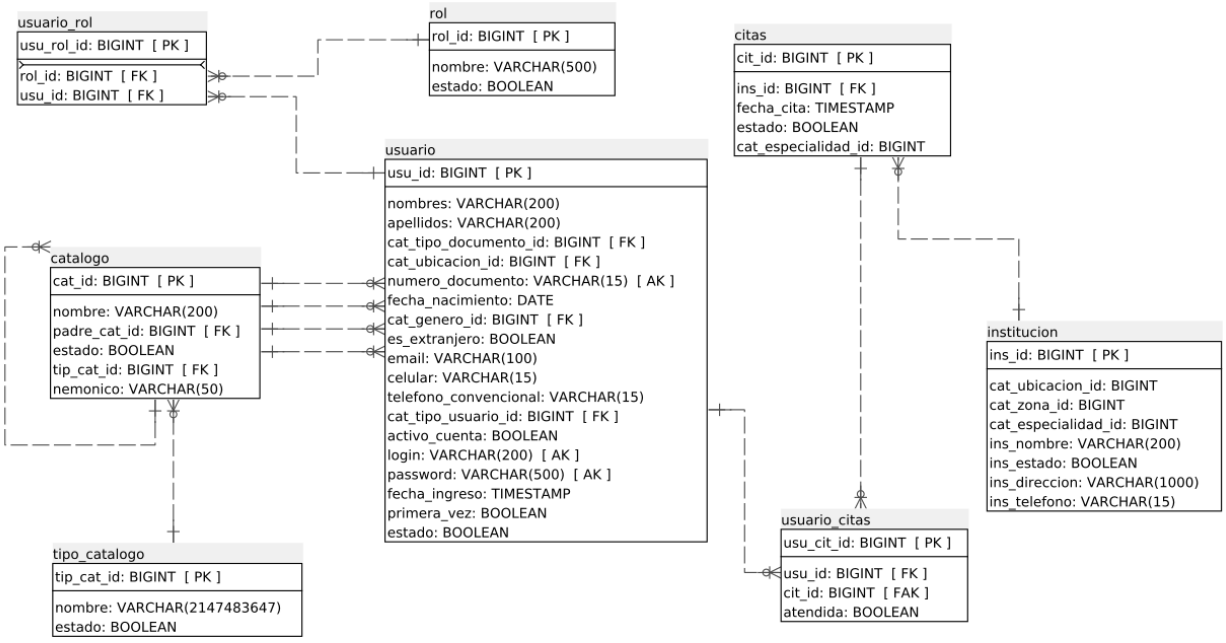
Facade (Fachada): Permite la creación de subsistemas, minimizando las comunicaciones y dependencias entre éstos**.** En la arquitectura planteada este se utiliza también en la capa de servicio cuando este debe utilizar varios DAOS.

Entre los estándares de codificación utilizados tanto en el componente web como en el móvil se destaca Java Code Conventions (http://java.sun.com/docs/codeconv/).

Estándares que SUN Microsystems recomienda para el desarrollo de aplicaciones en JAVA.

Siguiendo estos estándares en el desarrollo de aplicaciones, se puede ahorrar en un 80% de tiempo en el mantenimiento futuro de las aplicaciones, es decir, permite que cualquier programador pueda entender el código realizado por cualquier persona y pueda modificarlo de acuerdo a la necesidad futura sin atarse al programador inicial. Además si la aplicación respeta este estándar ganamos automáticamente 20% de eficiencia de la misma.

Modelo de datos



Esquema de la base de datos.

La base fue generada en Postgresql versión 9.4, la elección de este motor de base de datos se debe a que son proyectos de código abierto, además de soportar alta concurrencia, amplia variedad de tipos nativos, fácil de administrar, sintaxis SQL estándar, entre otras.

Para el componente móvil se realizó una réplica para SQLite de las siguientes tablas:

tipo_catalogo: Almacena los tipos de catálogos que se van a utilizar en la aplicación.

catalogo: Tabla que almacena los catálogos de agrupados por el tipo de catálogo, permitiendo una estructura de árbol para sus registros, aquí se encuentran los catálogos generales o globales de la aplicación.

usuario: Tabla que almacena los usuarios, permitiendo distinguirlos entre los usuarios de la aplicación móvil, que poseen el rol de pacientes, y de los usuarios de la aplicación web, que a su vez pueden ser; institución de salud o médicos particulares.

Estas tablas contienen información en común para el uso de la aplicación, de esta manera se evita el uso excesivo de servicios web.

Entre las tablas que manejan las citas están:

citas: Tabla que almacena las citas registradas, ya sea por los médicos particulares o las instituciones de salud.

usuario_citas: Tabla que almacena la relación entre la institución de salud/médicos particulares y el paciente que reservo la cita.

Listado de servicios webs.

Todos los servicios web creados son REST y utilizan el método POST para que los datos no sean mostrados en la URL, los objetos que se transmiten por los servicios son POJOs serializados en formato JSON y pertenecen a la librería **contrato-salud-integral-1.0**.

A continuación se listan los servicios web utilizados por la aplicación, así como la descripción de cada uno de ellos y su funcionalidad.

Registro de Usuario: Servicio web que permite el registro del usuario, recibe como parámetro un objeto de tipo RegistroUsuario y retorna un objeto RespuestaWs, su URL es http://server01.fehu.me/salud-integral-ws/webresources/usuario/registro

Recuperar Contraseña: Servicio web que permite la actualización de la contraseña al momento de realizar esta acción en la aplicación, recibe como parámetro el un objeto RegistroUsuario y retorna un objeto RespuestaWs, su URL es http://server01.fehu.me/salud-integral-ws/webresources/usuario/actualizar

Validar Credenciales: Servicio web que permite la validación del usuario al ingresar a la aplicación, recibe como parámetro un objeto de tipo LoginPassword y retorna como respuesta un objeto RespuestaWs, su URL es http://server01.fehu.me/salud-integral-ws/webresources/usuario/validarLoginPassword

Catálogo general: Servicio web que lista los catálogos de zonas y especialidades para la creación de una cita, no recibe parámetros y retorna un objeto de tipo CatalogoGeneral el cual contiene las dos listas descritas anteriormente, su URL es http://server01.fehu.me/salud-integral-ws/webresources/catalogo/zonasEspecialidades

Datos del médico: servicio web que provee un lista de médicos de acuerdo a los parámetros seleccionados en la interfaz de la aplicación, recibe como parámetro el objeto UbicacionZonaEspecialidadFecha y retorna una lista de objetos de tipo CatalogoInstituciones, su URL es http://server01.fehu.me/salud-integral-ws/webresources/institucion/institucionesUbicacion

Crear cita: Servicio web que realiza el registro de la cita seleccionada por el usuario, recibe como parámetro un objeto RegistroCita y retorna un objeto de tipo RespuestaWs, su URL es http://server01.fehu.me/salud-integral-ws/webresources/citas/citaPaciente

Listar citas: Servicio web que permite listar las citas creadas, recibe como parámetro un objeto RegistroCita y retorna una lista de objetos de tipo CatalogoCitasPaciente, su URL es http://server01.fehu.me/salud-integral-ws/webresources/citas/citaPorPaciente

Vistas

Vista Login

Vista Crear Usuario

Vista Crear Credenciales

Vista Principal

Vista Crear Cita

Vista Mis Citas

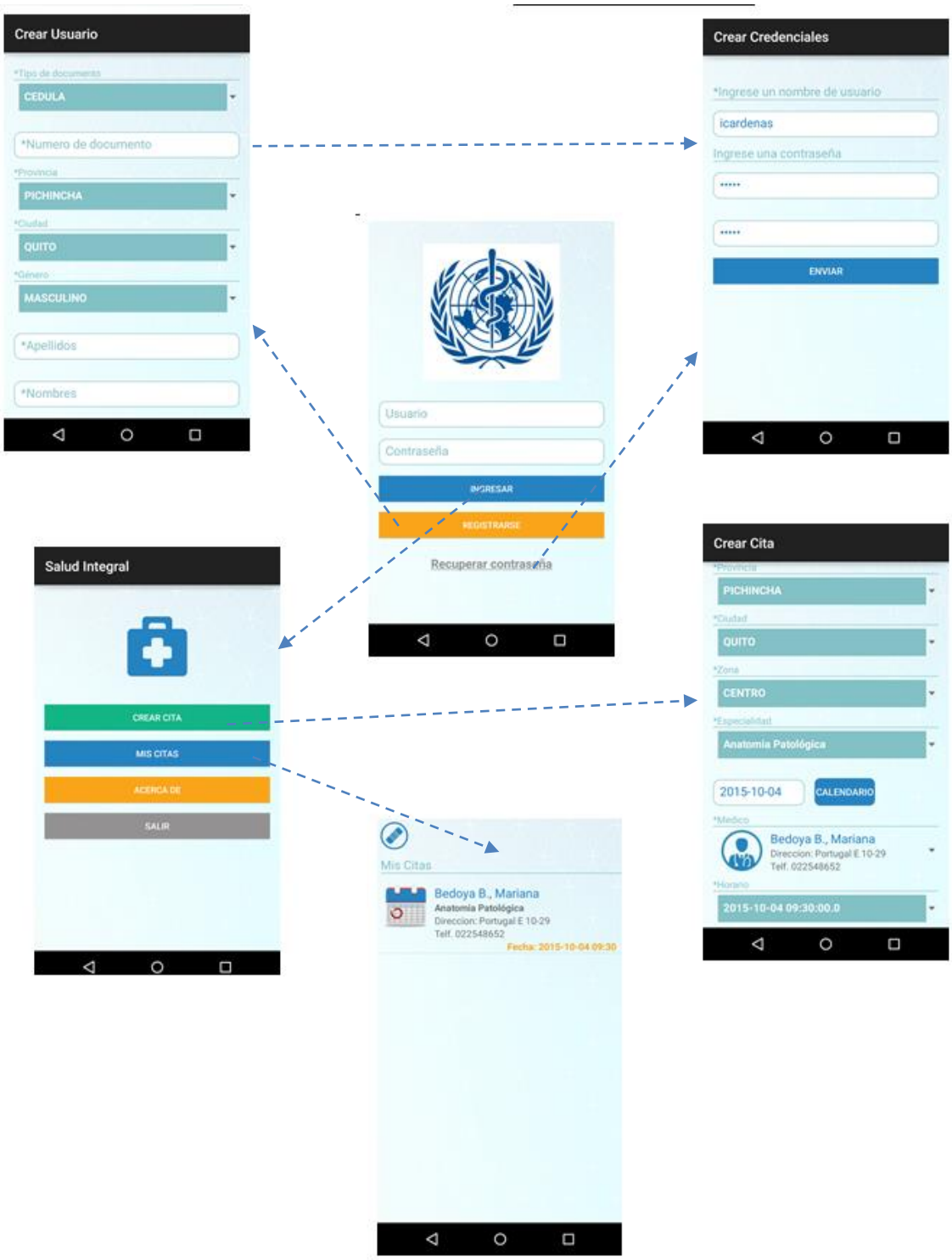


Diagrama de navegación

Alcance de la aplicación.

Para el desarrollo de la aplicación móvil en Android, se establece el siguiente alcance en su primera fase:

Funcionalidad de registro de usuarios: se debe permitir el registro de usuarios mediante una vista de registro de información, así como el registro de las credenciales de acceso a la aplicación, con las validaciones correspondientes de usuario único.

Funcionalidad de autenticación: se valida el ingreso a la aplicación directamente con el servidor a través de un servicio web que permita esta funcionalidad

Registro de citas: De acuerdo a los parámetros ingresados por el usuario (Ubicación, especialidad y la fecha en la que desea obtener la cita), la aplicación desplegara una lista de médicos y sus respectivos horarios de citas disponibles para esos criterios de búsqueda.

Consulta de citas: Se lista las citas registradas por el usuario que aún no sean atendidas, con la información que detalla dicha cita.

Además en esta fase se encuentra contemplado el desarrollo de los servicios web que interactúen con el servidor y cubran los requerimientos de la aplicación móvil en su primera fase.

Conclusiones

Se ha concluido el desarrollo de la aplicación móvil en Android en su primera Fase, permitiendo de esta manera cumplir con el alcance del proyecto propuesto en la primera fase.

Se realizó la integración con el servidor de aplicaciones Glassfish 4.1, mediante la creación de un artefacto war, el cual contiene los servicios web REST que interactúa entre la aplicación móvil y el servidor.

Se han construido los servicios web necesarios para la comunicación y funcionamiento de la aplicación móvil, utilizando REST obteniendo mayor rapidez y facilidad de consumo de la información.

Se debe considerar para las siguientes fases del proyecto, la gestión de las citas en la aplicación móvil, la creación de pacientes asociados al usuario principal, pudiendo ser estos familiares, la consulta de los resultados de su cita, como por ejemplo la receta y su historia médica. Se debe tener en cuenta estas funcionalidades en la aplicación web y su respectiva gestión.

Se debe plantear una estrategia de sociabilización del producto final, con el fin de llegar a cubrir la mayor cantidad de usuarios, para este caso, los médicos e instituciones de salud del Ecuador.

Anexos

Código servicio web

Código de ejemplo utilizado para el desarrollo de los servicios web en la plataforma JEE.

Clase que expone el servicio web para el consumo de la aplicación móvil:

```
@LocalBean
@Singleton
@Path("/catalogo")
@Consumes({MediaType.APPLICATION_JSON})
public class CatalogoWs {

    @EJB
    @Getter
    private CatalogoServicio catalogoServicio;

    @POST
    @Produces(MediaType.APPLICATION_JSON)
    @Path("/zonasEspecialidades")
    @Lock(LockType.READ)
    public CatalogoGeneral listarZonasEspecialidades() {
        CatalogoGeneral obj = new CatalogoGeneral();
      try {
      obj.setListaEspecialidades(getCatalogoServicio().listarPorN
      emonico(CatalogosEnum.ESPECIALIDADES.getNemonico()));

      obj.setListaZonas(getCatalogoServicio().listarPorNemonico(C
      atalogosEnum.ZONAS.getNemonico()));
         } catch (Exception e) {
                       Log.error(getClass().getName(), "Error al
            cargar especialidades " + e.getMessage());
         }
         return obj;
    }
}
```

Clase EJB que maneja la persistencia:

```
@LocalBean
@Stateless
public class CatalogoDao extends Generico<Catalogo> {

    @EJB
    @Getter
    private EjecutarSentenciasNativas ejecutarSentenciasNativas;

    public CatalogoDao() {
        super(Catalogo.class);
    }

    public List<CatalogoEspecialidades> listarPorNemonico(final
String nemonico) throws Exception {
        Map<String, Object> params = new HashMap<>();
        String sql = "SELECT cat_id,nombre FROM
sch_seguridades.catalogo WHERE nemonico = ? AND estado = true
ORDER BY nombre";
        params.put("1", nemonico);
        return
getEjecutarSentenciasNativas().listarPorSentenciaSql(sql,
CatalogoEspecialidades.class, params);
    }

}
```