

Programación Web Avanzada

-Trabajo Final-

Jesús Rodríguez Martínez

Enunciado

1. Ejercicio Obligatorio

En este ejercicio se propone la creación de la oficina virtual del Banco de Alcalá. El Banco de Alcalá a través de su aplicación Android permitirá al menos las siguientes operaciones:

1. **Alta de Cuenta:** El usuario solicitará el alta de una cuenta. El sistema creará dicha cuenta y le asignará un número de cuenta compuesto por 4 dígitos, un PIN también compuesto por 4 dígitos (aleatorios) y se lo notificará al usuario. El sistema tendrá que tener en cuenta que no pueden existir dos números de cuenta iguales.
2. **Acceso de Cuenta:** El usuario accederá a su cuenta a través de su número de cuenta y su PIN, en caso de ser correctos le permitirá realizar la siguientes operaciones:
 - 2.1. **Ingresos de Efectivo:** El sistema le pedirá una cantidad positiva de dinero y procederá a ingresarla en su cuenta.
 - 2.2. **Retirada de Efectivo:** El sistema le pedirá una cantidad positiva de dinero y procederá a retirarlo de su cuenta siempre y cuando disponga de suficiente efectivo.
 - 2.3. **Transferencia:** El sistema le pedirá una cantidad positiva de dinero, un número de cuenta destino y siempre y cuando disponga de suficiente efectivo y la cuenta de destino exista, le realizará la transferencia.
 - 2.4. **Recarga Telefónica:** El sistema le pedirá una cantidad positiva de dinero y un número de teléfono y siempre y cuando el usuario disponga de efectivo suficiente procederá a realizar el cargo en su cuenta.
 - 2.5. **Movimiento.** El sistema mostrará los detalles de los
 - 2.6. **Saldo.** El sistema mostrará el saldo contable de la cuenta.

El programa almacenará todos los datos de los usuarios en una externa al terminal, que será atendida por uno o varios servicios Web RESTFul.

Además el terminal analizará a través de su sensor de proximidad la distancia que objeto más cercano. Cuando esta distancia sea inferior a 3 cm. sacará por pantalla una ventana con el nombre y los datos del autor de la práctica. Esta ventana desaparecerá a los 10 sg.

2.1. Ejercicio Opcional

Se propone almacenar además la geolocalización del terminal cuando se realice cualquier tipo de trámite con el Banco de Alcalá.

Solución

Para la resolución del ejercicio se van a crear 2 aplicaciones:

1. **BancoAlcala**: aplicación encargada de gestionar las peticiones RESTful y acceder a la base de datos a realizar las acciones solicitadas.
2. **BancoAlcalaCliente**: aplicación Android encarga de la interacción entre el usuario y BancoAlcala.

Para almacenar la información se va a utilizar una base de datos en MySQL.

A continuación se detalla la implementación tanto de la base de datos como de cada una de las aplicaciones.

Base de datos

Este es el diagrama de tablas de la base de datos:

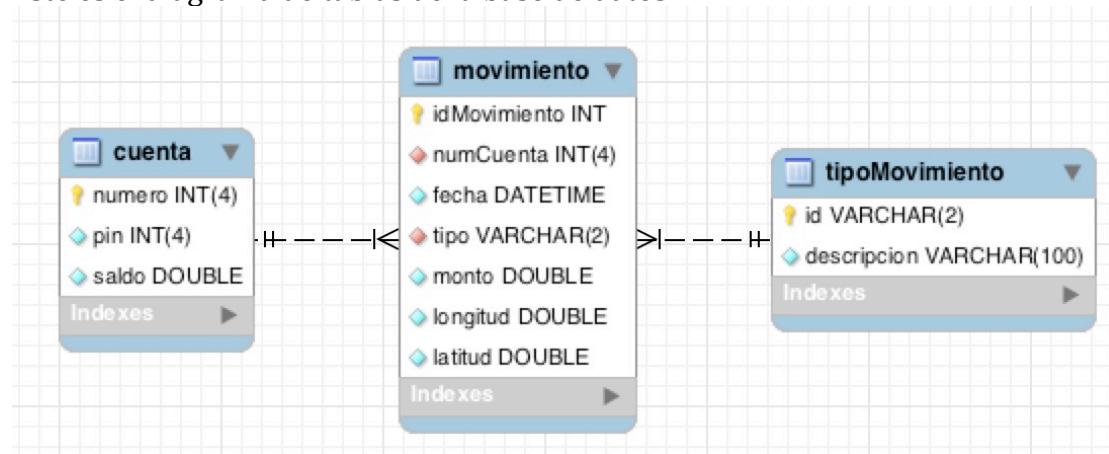


Imagen 1 Diseño base de datos

- “cuenta”: tabla donde almacenar la información de las cuentas. La columna “numero” es su clave, autogenerada, y de 4 dígitos. El pin también es de 4 dígitos.
- “movimiento”: tabla donde almacenar la información de los movimientos.
- “tipoMovimiento”: tabla catalogo tipos de movimiento.

Se ha precargado en una cuenta, y la información de los tipos de movimientos:

numero	pin	saldo
0000	0000	15000

Imagen 2 Precarga cuentas

id	descripcion
IE	Ingreso Efectivo
RE	Retirada Efectivo
TR	Transferencia
RT	Recarga Telefonica

Imagen 3 Precarga tipos de movimiento

Se incluye en el zip el fichero creacionBBDD.sql que incluye todas las sentencias para crear la base de datos, las tablas y las inserciones iniciales.

Una vez ejecutado el script creacionBBDD.sql (yo lo he hecho desde el propio Netbeans) se crea la base de datos (se elimina primero en caso de existir).

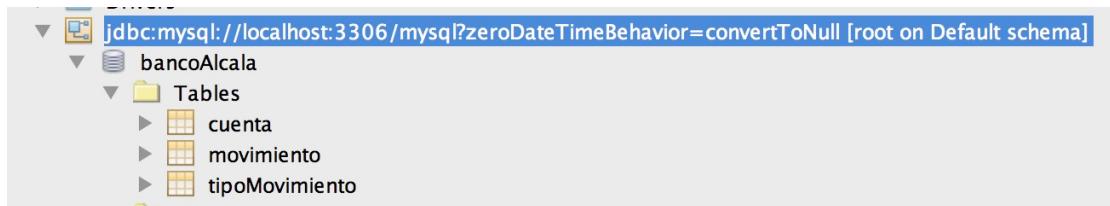
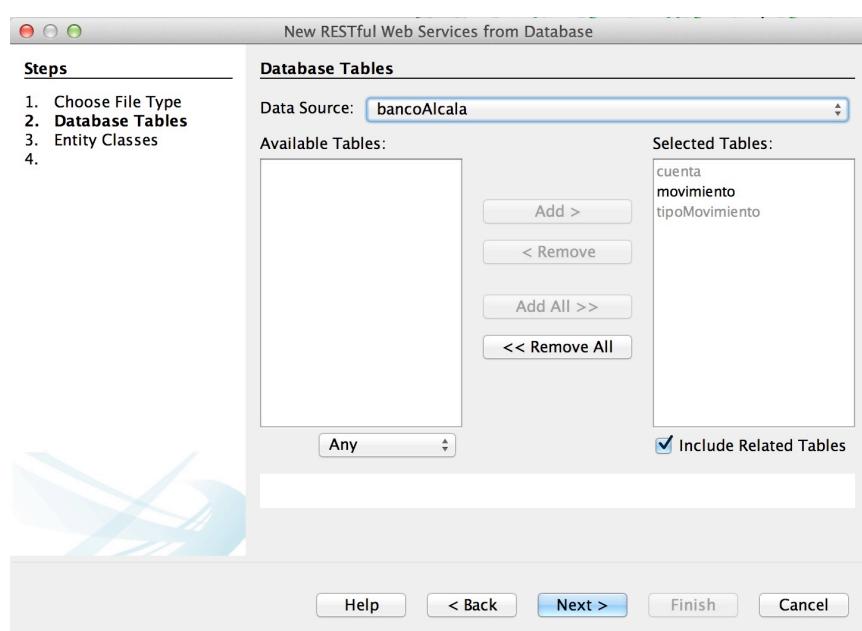
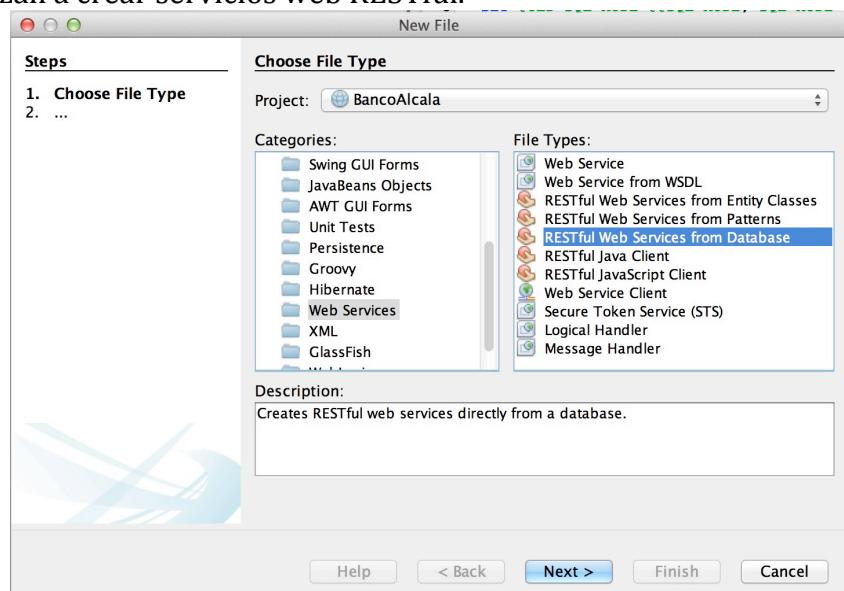


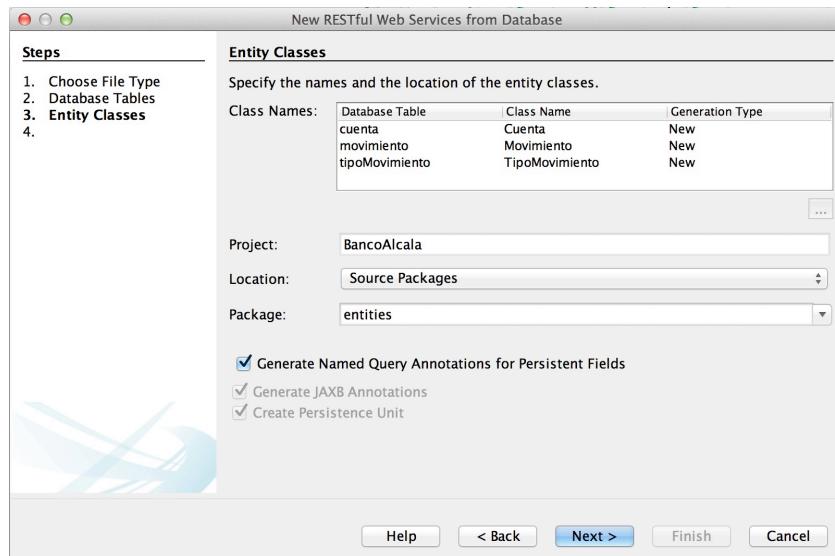
Imagen 4 Base de Datos creada desde Netbeans

BancoAlcala

Aplicación encarga de gestionar a través de peticiones RESTful la información del banco. Se ha implementado utilizando Netbeans.

Una vez creada la base de datos, se crean sus clases de entidad, y a partir de estas se empiezan a crear servicios web RESTful.





En las clases ya generadas, sólo añado un método para obtener los últimos n movimientos de una cuenta en MovimientoFacadeREST:

```
@GET
@Path("ultimos/{numCuenta}/{numVistos}")
@Produces({"application/xml", "application/json"})
public List<Movimiento> ultimos(@PathParam("numCuenta") int numCuenta, @PathParam("numVistos") int numVistos) {
    Query query = em.createNamedQuery("Movimiento.ultimos");
    Cuenta cuenta = cuentaFacadeREST.find(numCuenta);
    query.setParameter("numCuenta", cuenta);
    query.setMaxResults(numVistos);
    return query.getResultList();
}
```

Imagen 5 Método "ultimos"

Y en la clase Movimiento incluyo una nueva NamedQuery:

```
@NamedQuery(name = "Movimiento.ultimos", query = "SELECT m FROM Movimiento m WHERE m.numCuenta = :numCuenta order by m.fecha desc")})
```

Imagen 6 NamedQuery para "ultimos"

Esta aplicación requiere de la base de datos en marcha, y el esquema “bancoAlcala” creado.

Se incluye en el zip el proyecto de Netbeans correspondiente a esta aplicación en la carpeta “/BancoAlcala”.

BancoAlcalaCliente

Aplicación Android desarrollada como cliente de “BancoAlcala”.

Requisitos

Aplicación BancoAlcala

Para que la aplicación pueda operar es necesario tener corriendo en un servidor web la aplicación “BancoAlcala”.



Banco de Alcalá - Servicios RESTful

Esta aplicación debe estar corriendo para ser accedida desde los clientes. Aplicación desarrollada por Jesús Rodríguez como parte del trabajo final de la asignatura Programación Web Avanzada.

Imagen 7 Banco Alcalá en marcha

Y que los servicios responden:

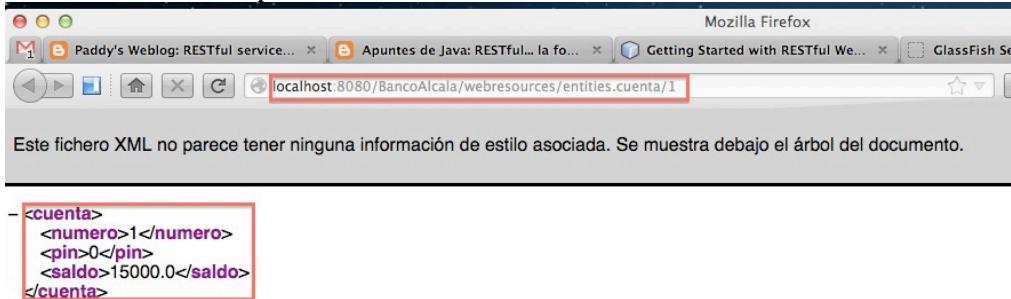


Imagen 8 Servicio REST en BancoAlcala

Es necesario configurar el cliente con la dirección y el puerto donde está corriendo BancoAlcala, para ello basta con cambiar el valor de la siguiente constante en la clase 'MiAplicacion':

```
public class MiAplicacion extends Application {
    final private String direccionPuerto = "192.168.1.102:8080/";
```

Imagen 9 Configuración conexión con BancaAlcala

Si no se configura correctamente este parámetro, la aplicación no podrá hacer ninguna petición.

Simulador Sensores

Debido a la implementación del simulador de los sensores para el emulador de Android, una vez incluído el código que hace uso de él, hace obligatorio que la aplicación que simula los sensores esté en marcha, y correctamente configurada en el emulador de Android (más detalles a continuación).

Extendiendo “Application”

Creo la clase “MiAplicacion” que extiende la clase “Application” para que las clases que implementan mis actividades tengan disponible información (gracias a “getApplication()”) y sin tener que preocuparme por tener que se recreen Singlettons, o perder información por el ciclo de vida de la propia aplicación y sus procesos. Para esto debo crear la clase, y modificar el manifiesto de la aplicación.

```
public class MiAplicacion extends Application {
```

Imagen 10 Clase MiAplicacion

```
        <activity android:name="com.jrm.bancoAlcalaCliente.modelo.utiles.MiAplicacion">
```

Imagen 11 Incluyo MiAplicacion en el manifiesto

Modelo de Datos

He copiado las clases Cuenta, Movimiento y TipoMovimiento del proyecto BancoAlcala, y las he dejado como POJOs.

Para facilitar la conversión entre Clases y JSON he utilizado la librería GSON.

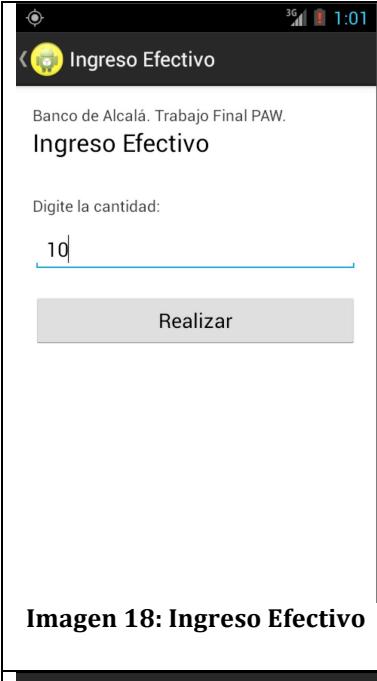
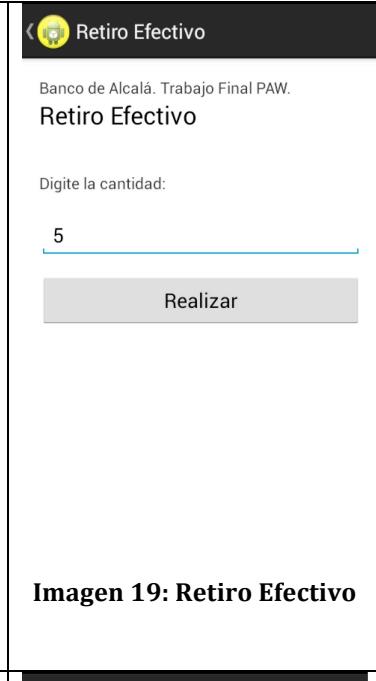
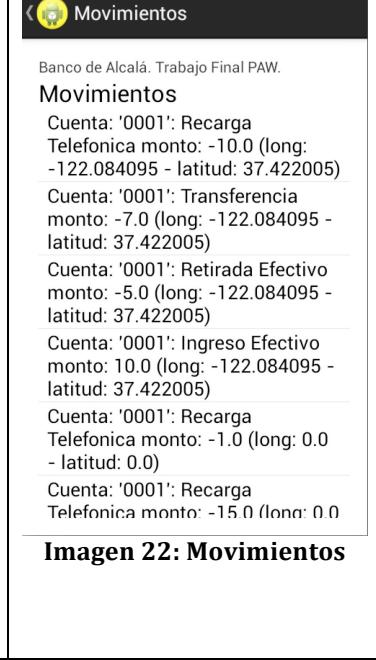
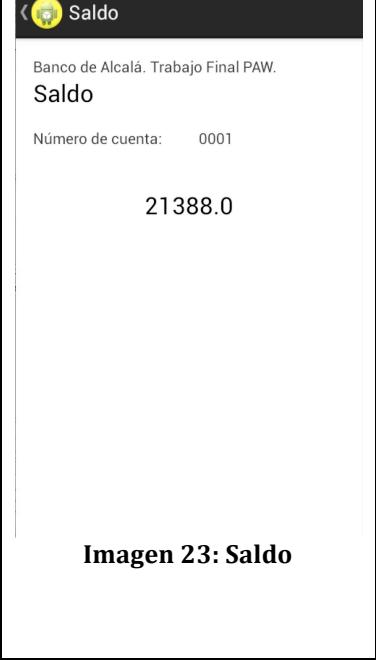
Servicios

He creado un paquete llamado “servicios” que contiene la lógica de cada una de las operaciones. Cada operación es un servicio (en este caso se podría haber dejado todas las operaciones relacionadas con cada entidad en una única clase).

Capturas Pantallas

 <p>Banco Alcalá JRM</p> <p>Banco de Alcalá. Trabajo Final PAW. INICIO</p> <p>Seleccione una opción:</p> <p>Alta de Cuenta</p> <p>Acceso Cuenta</p> <p>Ver Cuentas</p>	 <p>AltaCuenta</p> <p>Banco de Alcalá. Trabajo Final PAW. Alta de Cuenta</p> <p>¿Está seguro de que desea dar de alta una nueva cuenta?</p> <p>Si No</p> <p>Cuenta sin crear.</p>	 <p>AltaCuenta</p> <p>Banco de Alcalá. Trabajo Final PAW. Alta de Cuenta</p> <p>Cuenta Creada: Cuenta '0003' (PIN: '5547')</p>
<p>Imagen 12: Inicio</p>	<p>Imagen 13: Alta Cuenta</p>	<p>Imagen 14: Confirmación Alta Cuenta</p>

 <p>Ver Cuentas</p> <p>Banco de Alcalá. Trabajo Final PAW. Cuentas</p> <p>Cuenta '0001' (PIN: '0000')</p> <p>Cuenta '0002' (PIN: '4464')</p> <p>Cuenta '0003' (PIN: '5547')</p>	 <p>AccesoCuenta</p> <p>Banco de Alcalá. Trabajo Final PAW. Acceso a Cuenta</p> <p>Número de cuenta:</p> <input type="text"/> <p>PIN:</p> <input type="text"/> <p>Entrar</p>	 <p>Opciones Cuenta</p> <p>Banco de Alcalá. Trabajo Final PAW. Opciones</p> <p>Ingreso Efectivo</p> <p>Retiro Efectivo</p> <p>Transferencia</p> <p>Recarga Telefónica</p> <p>Movimientos</p> <p>Saldo</p> <p>Salir</p>
<p>Imagen 15: Listado Cuentas</p>	<p>Imagen 16: Acceso Cuenta</p>	<p>Imagen 17: Opciones Cuenta</p>

 <p>Imagen 18: Ingreso Efectivo</p>	 <p>Imagen 19: Retiro Efectivo</p>	 <p>Imagen 20: Transferencia</p>
 <p>Imagen 21: Recarga Telefónica</p>	 <p>Imagen 22: Movimientos</p>	 <p>Imagen 23: Saldo</p>

Como puede verse, cada movimiento incluye su longitud y latitud de acuerdo a la última posición conocida por el GPS en el momento en que se realiza.

Sensor de proximidad

Para realizar esta parte he utilizado el emulador openintents. El código presentado está realizado para funcionar con este emulador, no en dispositivos reales.

Pasos necesarios:

- Instalar la aplicación Sensor Simulator en el emulador
- Arrancar la aplicación SensorSimulator
- Conectar desde la aplicación Sensor Simulator el emulador a la aplicación SensorSimulator

- Añadir en las librerías del proyecto el .jar del simulador de sensores.
- Crear la clase que escuche los sensores.
- Enviar los parámetros desde SensorSimulator a la aplicación.

Como ya se ha requerido, para que la aplicación funcione es necesario tener el simulador de sensores en marcha correctamente configurado en el emulador de Android.

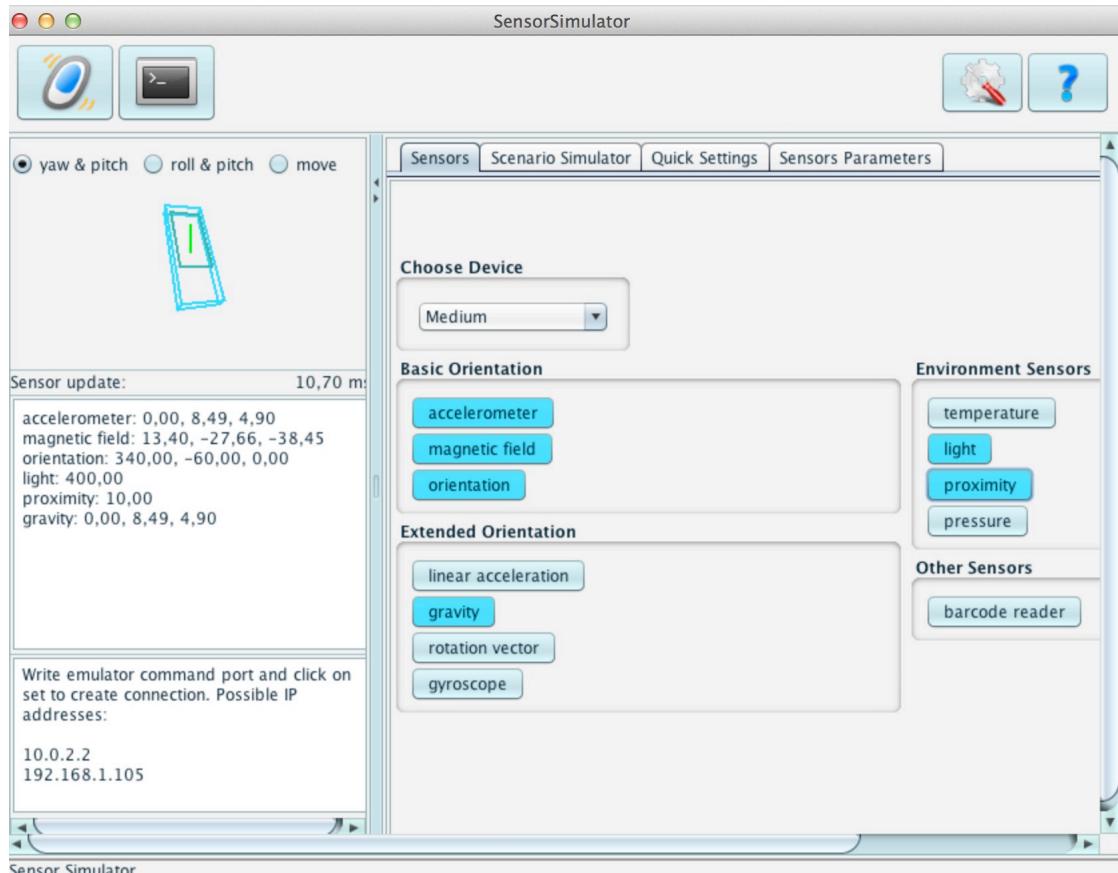
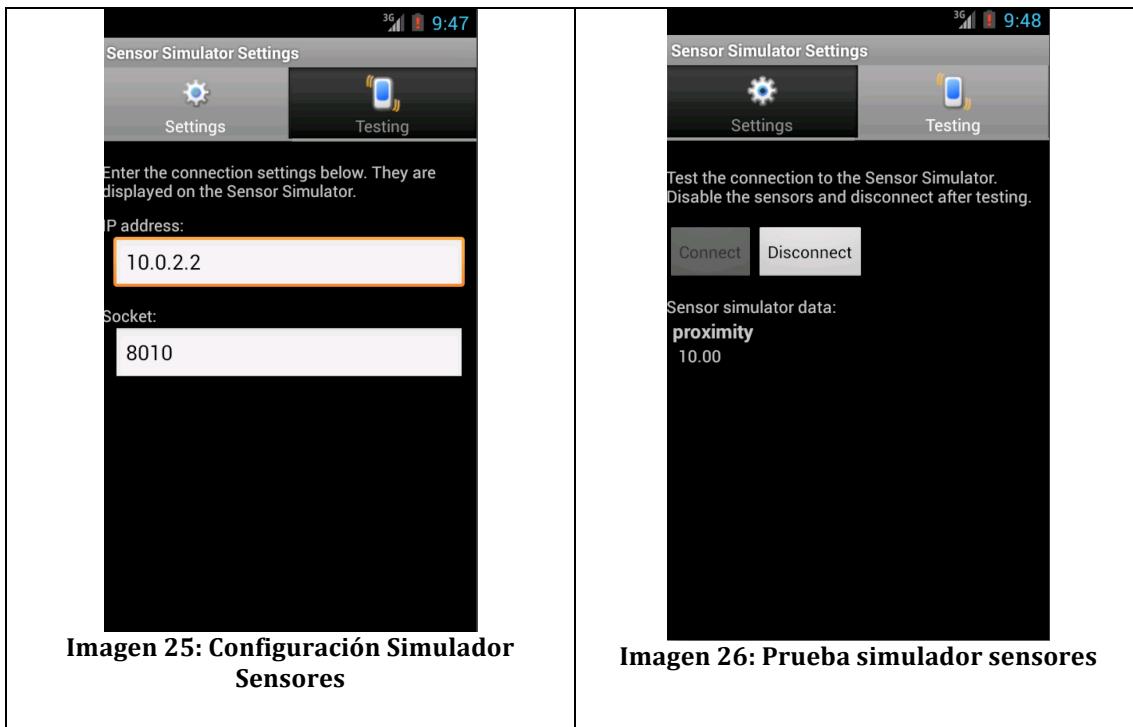


Imagen 24: Simulador Sensores



He tenido problemas para mostrar el mensaje durante 10 segundos (sólo se mostraba cuando se pasaba una de las constantes Toast.LENGTH_SHORT o Toast.LENGTH_LONG). La función mostrarToastDuracionVariable se encarga de alargar su visualización mostrándolo en un hilo propio y haciendo un sleep en dicho hilo.

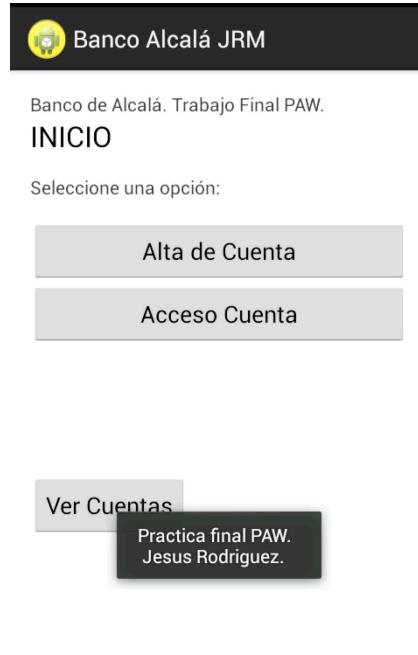


Imagen 27: Mensaje sensor proximidad

Se incluyen en el zip el proyecto de Eclipse correspondiente a esta aplicación.