UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

APLICACIÓN ANDROID: GUÍA INTERACTIVA PARA EDIFICIOS USANDO CÓDIGOS QR, BRÚJULA Y WIFI.

Proyecto Final de Carrera

ITIS

DISCA-213

Autor: Luis Muñoz López

Director: Manuel Agustí Melchor

Septiembre 2013

Resumen

En este proyecto voy a intentar dar una solución de navegación en edificios de gran tamaño, pensado inicialmente como un directorio y asistente para movernos dentro de edificios como universidades, hospitales, ferias.

Lugares grandes y poco conocidos para los visitantes puntuales que junto con las prisas podrían beneficiarse de las capacidades de esta solución.

Para ello utilizaré un teléfono Android y sus sensores como son la cámara, conectividad Wifi, brújula y mostraremos la guía interactiva en la pantalla táctil del teléfono coordinándolo con un servicio de información del edificio que nos proporciona el mapa y directorio al cual accedemos a través de la red Wifi para invitados.

Palabras clave: Android, navegación, edificios, guía.

Tabla de Contenidos

Tabla de contenido

Tabla de Ilustraciones

1. Introducción

La cuestión se plantea normalmente una vez en la vida- ¿Qué quiero hacer como Proyecto de Fin de Carrera? Siempre hay ofertas de PFC`s más o menos interesantes pero siempre he pensado que lo mejor es hacer algo que sea idea tuya, que quieras hacer y que vaya a hacerte crecer como profesional en el mundo de la informática.

Inicialmente tenía muy claro que quería hacer algún tipo de desarrollo en Android, es una plataforma en crecimiento y una tecnología que desconozco en el punto en el que redacto estas líneas. Esto tiene beneficios e inconvenientes, voy a aprender una tecnología nueva que me da Currículum pero aumenta el coste del PFC por esa curva de aprendizaje.

Una vez decidida la tecnología en la que quería trabajar, necesitaba un problema a resolver con Android, en particular con las características disponibles en los teléfonos Android. Uno de los usos cotidianos de los teléfonos inteligentes (Smartphone) suele ser como GPS para facilitar los trayectos en coche. Esta misma tecnología no se puede utilizar por las limitaciones de intensidad de señal GPS dentro de edificios, este es el problema para el que pretendo dar un acercamiento de solución.

Tengo un problema al que enfrentarme y una tecnología para resolverlo, con estos elementos definidos pasamos a trabajar en el proyecto.

1.1. Presentación Informal

El mundo de la telefonía móvil ha evolucionado en direcciones insospechadas para la mayoría de la gente y más en dirección a los gadgets de ciencia ficción de los 70-90, convirtiéndose en herramientas multiuso cotidianas, con potencia de procesamiento y multitud de sensores e interfaces.

El punto vital de esta evolución ocurrió con el lanzamiento del primer iPhone, no siendo este el primer teléfono inteligente del mercado consiguió marcar un antes y un después en la telefonía por varios motivos:

- Interfaz multi-táctil sin teclado con una pantalla de 3.5".
- Pantalla todo color de gran calidad.
- Capacidades multimedia.
    - Cámara de calidad
    - Reproducción de video y Audio.
- Almacenamiento interno de gran densidad.

En general un dispositivo innovador y que superaba de manera aplastante a la competencia, este teléfono se convirtió en el objeto de deseo de todo usuario de un teléfono anterior.

Tras este golpe maestro por parte de Apple la competencia comenzó una pugna por ponerse a la altura, aumentando las capacidades de sus dispositivos y forzando a Apple a volver a innovar.

Sin profundizar demasiado en la historia del Smartphone tal y como lo conocemos hoy en día voy a hablar sobre uno de los sensores que era más obvio que se integraría en los teléfonos de este tipo, el GPS.

El GPS (del inglés Global Positioning System) es un sistema de navegación basado en satélites que provee de información Horaria y de localización en cualquier punto de la tierra que tenga en "línea visible" al menos 4 satélites GPS. (Wikipedia: http://en.wikipedia.org/wiki/Global_Positioning_System)

Estos satélites orbitan la tierra en una configuración orbital en la cual en todo momento es posible tener entre 8 y 9 de estos satélites a la vista. Estos satélites emiten una señal de radio en la que anuncian la hora de emisión del mensaje y donde se encuentra el satélite al emitir dicha señal, entonces nuestro dispositivo en la tierra calcula, según estos datos proporcionados por los satélites, donde nos encontramos, en qué región horaria, a qué velocidad nos movemos, a que altura estamos.

Se trata de una tecnología muy ingeniosa, funcionando desde hace más de 40 años y que ha llegado al nivel de usuario común ya sea en el GPS o en un Smartphone con GPS.

A pesar de lo ingenioso del sistema tiene ciertas limitaciones, alto consumo de batería pero la más seria es que hay que recibir señal externa de al menos 4 satélites, que se encuentran a más de 20000 Km sobre la tierra, esta señal no es lo bastante potente por lo general como para ser captada dentro de un edificio, haciendo este sistema inútil dentro de un edificio donde se bloquea la señal por las paredes.

Debido a esta limitación física no es posible usar la seña GPS para guiarnos en el interior de un edificio ya que la señal es insuficiente. Por ello se han realizado algunos esfuerzos a favor de conseguir una alternativa al GPS para la navegación en edificios, yo intentaré dar una solución basada en posición inicial y navegación con brújula y ayudas de posicionamiento fijas en un Smartphone Android junto al servicio que nos provee de la información del edificio en el que nos encontramos.

Existen otras soluciones actualmente en boga, como posicionamiento por Wifi/3G que presentaré a continuación en el apartado **Situación General y Estado del arte**.

1.2. Objetivos

El objetivo del proyecto es proporcionar posicionamiento dentro de grandes edificios, gubernamentales, supermercados, grandes almacenes, hospitales sin usar el GPS ya que no está disponible por la falta de señal.

Para ello usaré todo el elenco de sensores disponibles en los Smartphone Android de una manera particular aprendiendo a interactuar con estos sensores y a comunicarme con entidades externas al propio teléfono para consumir recursos.

Existen en la actualidad soluciones para posicionamiento en interiores (Indoor Positioning system), el producto más conocido y que ha saltado a la fama hace poco es WifiSLAM: https://angel.co/wifislam ya que ha sido Adquirido por Apple por una ingente suma de dinero.

Esta compra por parte de Apple indica que es un campo que no tiene solución dominante de momento con lo cual voy a intentar dar una alternativa a estas propuestas que trabajan sobre intensidad de señal y otras tecnologías, algunas de ellas mu interesantes.

2. Situación General y estado del arte

Hace un año se formó la alianza "In-Location: http://goo.gl/5sBDkD", promovida por Nokia en Agosto de 2012 se trata de una alianza estratégica entre 22 grandes compañías para el desarrollo de tecnologías de posicionamiento en interiores.

Las tecnologías más comunes dentro del posicionamiento dentro de edificios están basadas en las siguientes tecnologías:

- Señales EM: Estas soluciones trabajan de manera muy similar al GPS, se emiten señales de radio ya sea, bluetooth, Wifi, 3G para triangular la posición del dispositivo que las revibe.

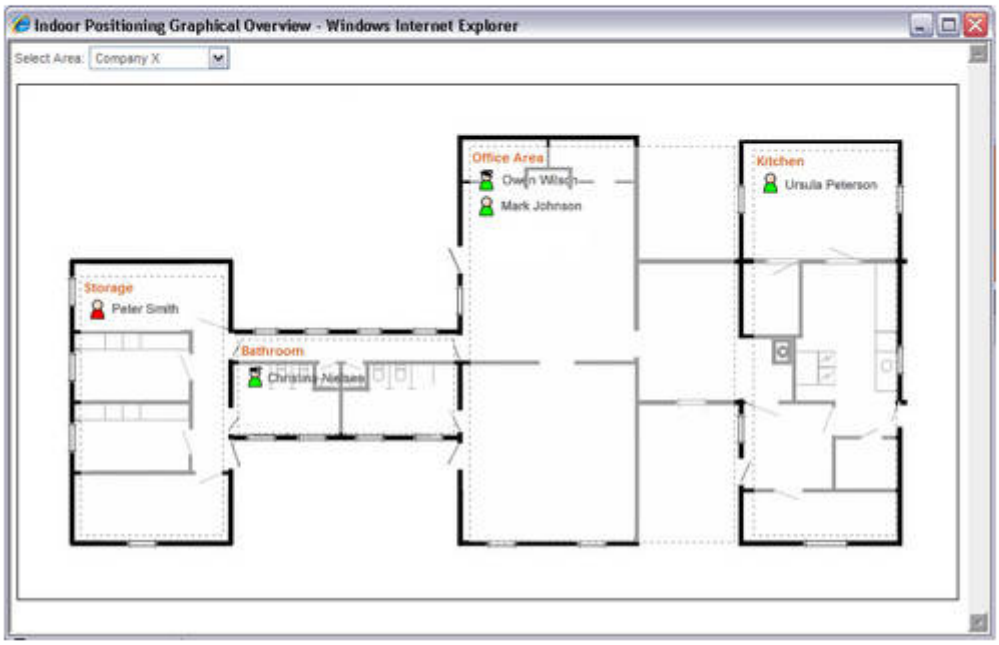
Ilustración 1 Navizon

- Luz: Montando emisores de luz Led en las zonas a controlar y controlando las frecuencias de emisión es posible con la cámara del teléfono identificar en que área nos encontramos, en términos generales (Identifica la habitación pero no te posiciona en ella por ejemplo).

ZONITH Indoor Positioning System (IPS)

Locating People and Assets with Bluetooth

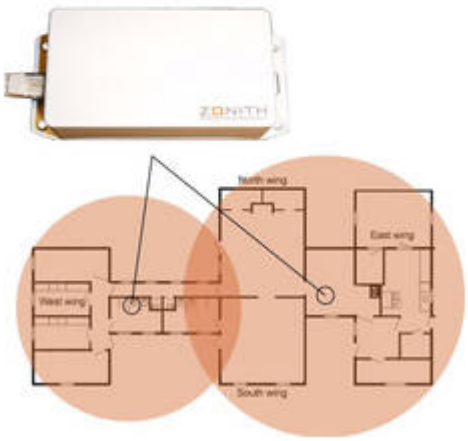
Keep track of your employees for safety purposes and of your assets for heightened efficiency.The ZONITH Indoor Positioning Module can locate and track people and assets indoors. People in distress can press their panic button, and a text message with location information will reach the colleagues or rescue central instantly. This is the perfect lone worker protection tool.



The graphical interface, can be shown in a browser. Each icon (left) symbolizes an employee with a Bluetooth unit. If the icon is green there is no trouble. If it is red the person has activated a personal alarm. The Icons with caps are guardsmen.

Zones Created by Beacons

Each of the small, discretely designed ZONITH Bluetooth Positioning Beacons (shown right) create a detection zone. The Beacons are connected to the ZONITH Alarm Control System software through a standart LAN network.



People Detected by Beacons

People are detected by the ZONITH Bluetooth Positioning Beacons by wearing *any* kind of Bluetooth unit (including radios, mobile phones, head sets and tags). Every Bluetooth unit has its own unique identity, and it is easy to register it in the system.

- Campo Magnético: Se mapea el campo magnético del edificio y después se usa la brújula del teléfono móvil para posicionarnos dentro de dicho mapa.
- RFID: Se trata de etiquetas de radiofrecuencia que se excitan al pasar por determinados puntos de excitación, se trata de una solución a base de Checkpoints.

Diverse Tag Line-up for Assets, Patients and Staff

T-100 Series, Disposable Patient Bracelet Tag

The Radianse T-100 Series active-RFID tag is built into a lightweight, flexible and attractive patient bracelet, which is both durable and extremely functional. The single-use tag is engineered with inherent high-end location tracking capabilities at an attractively low cost-per-use.

The T-100 tag can be used hospital-wide for monitoring all patients, from the point of entry into a healthcare facility to the time they are discharged. Configuration and data entry has been simplified and only takes seconds per patient for the RFID bracelet tag to be enabled.

The tag has a battery life of 30 days, which will handle the majority of all typical patient hospital stays. The tag is designed for single patient use and is disposable. The band is of a similar design and material found in most patient bracelets current used in healthcare today, with the addition of ultra-thin active-RFID micro-circuitry and battery embedded right in the ultra-thin wrist band.

- Low cost per tag, 30 day battery life and slim-line wristband design
- Perfect for patient tracking and throughput analysis
- Single use application, each patient receives bracelet at time of admission

---

T-400 Series, Patient and Asset Tag

Radianse T-400 Series ID-Tags are highly durable, tamper-resistant and programmable. A remarkably long battery life greatly reduces maintenance and the total cost of ownership.

Authorized users can find any tagged device from a simple easy-to-use web-based interface. Two programmable buttons on the tag give you the flexibility to design customized alert notifications. For example, pressing a button may indicate that a device needs servicing or is ready for return. A button can alert an asset management application that preventive maintenance is complete.

- 2 programmable buttons for customized alert notifications
- Locates using both InfraRed (IR) and 433 MHz Radio Frequency (RF)
- Long 2-year battery life, batteries are field replaceable

---

1. Problema

Muchos de estos sistemas no permiten obtener la orientación o dirección, necesitan complementarse en otras tecnologías.

2. Orientación y futuro de estas tecnologías

El crecimiento de estas tecnologías se está viendo promocionado por grandes actores dentro de la alianza In-Location y presionando desde los gobiernos para poder aplicar estas tecnologías para situaciones de emergencia donde mejorarían la atención a los ciudadanos en casos de catástrofes naturales.

Información obtenida Directions Magazine: http://goo.gl/qeEvZ5

3. Soluciones “Indoor Positioning”

Elenco de enlaces a soluciones de Indoor Positioning para hacerme una idea del estado del arte en esta modalidad de posicionamiento.

1. **WIFI**
    1. http://www.skyhookwireless.com/
    2. https://angel.co/wifislam
    3. http://www.wifarer.com/deployment
    4. http://www.navizon.com/

2. **BLUETOOTH**
    1. http://www.senionlab.com/
    2. http://www.zonith.com/products/ips/

3. **RFID/WIFI**
    1. http://www.radianse.com/press-himss-020104.html

4. **LUZ**
    1. http://lighthousesignal.com/our-technology/

5. **HÍBRIDO**
    1. http://www.polestar.eu/en/nao-campus/indoor-positioning.html
    2. http://locata.com/

6. **RADIO BALIZA**
    1. http://www.nomadicsolutions.biz/prod.php?lg=uk&id=51

7. **INDETERMINADO**
    1. http://www.trueposition.com/
    2. https://www.guardly.com/technology/indoor-positioning-system

3. Requerimientos

Para la realización del proyecto necesito dos elementos principale:

- Teléfono Android.
- Servicio web donde obtendremos la información.

Con lo cual estos son los requerimientos Hardware, Software:

Hardware:

Teléfono Android con los sensores:

- Cámara.
- Brújula.
- Wifi.

En el caso del teléfono de pruebas para la aplicación que desarrollaré durante el proyecto se trata de un Nexus 4 de LG y Google.

Enlace a las características técnicas completas del teléfono: Nexus 4: http://www.google.es/nexus/4/

Especificaciones técnicas

Ilustración 2: Nexus 4

Pantalla

- 4,7 pulgadas
- Resolución de 1280x768 píxeles (320ppi)
- Pantalla IPS WXGA
- Corning® Gorilla® Glass 2

Dimensiones

- 133,9 x 68,7 x 9,1 mm
- 139 g

Cámaras

- 8 MP (principal)
- 1,3 MP (frontal)

Conectividad

- Wi-Fi (802.11 a/b/g/n)
- Bluetooth
- NFC (Android Beam)
- GSM/UMTS/HSPA+ libre
- GSM/EDGE/GPRS (850, 900, 1800, 1900 MHz)
- 3G (850, 900, 1700, 1900, 2100 MHz)
- HSPA+ 42
- Carga inalámbrica
- SlimPort (TM)

Procesador y memoria

- 8 GB o 16 GB (la capacidad formateada real será inferior)
- 2 GB RAM
- CPU Qualcomm Snapdragon™ S4 Pro

Características

- Android 4.2 (Jelly Bean)
- Acelerómetro
- GPS
- Giroscopio
- Barómetro
- Micrófono
- Luz ambiental
- Brújula

Software:

En el caso de este teléfono tenemos una versión de Android 4.3 (Jeally Bean) de Android puro, es decir, sin personalización del fabricante si no una ROM base generada por Google.

Como software de desarrollo trabajaré con los IDEs:

- Eclipse ADT: IDE Eclipse con el plugin para desarrollo en Android proporcionado por Google. https://developers.google.com/
- Zend EclipsePDT: para el desarrollo del servicio de planos y directorio en tecnología PHP. http://www.zend.com/en/company/community/pdt/
- Sublime Text2: como editor de textos general. http://www.sublimetext.com/
- Android Studio: IDE de desarrollo Android sacado por Google http://developer.android.com/sdk/installing/studio.html
- Xampp: servidor de aplicaciones web, php. http://www.apachefriends.org/es/xampp.html
- Firefox: Navegador web con herramientas de desarrollo avanzada **Firebug** http://www.mozilla.org/es-ES/firefox/new/ y http://getfirebug.com/

Servicio de información para el APP:

Para hacer que una sola aplicación nos sirva para multitud de edificios hemos de abstraer la información del edificio del APP.

Para ello vamos a generar un servicio disponible en la red Wifi de invitados a la que se conecta la aplicación en la cual tras consultar dicho servicio se nos proporciona el Mapa de la planta (imagen) y el directorio de dicha planta.

Arquitectura:

Previo al desarrollo realizaré un análisis de requisitos y arquitectura donde estableceremos el flujo de navegación dentro de la aplicación y todos los factores necesarios para el desarrollo de la misma.

4. Enfoque (secuencia de trabajo)

Como futuro ingeniero y tras formarme en el entorno laboral, resulta difícil no proceder de forma ingenieril o cuanto menos técnica para evitar pérdidas de tiempo en el trabajo.

Si algo se queda marcado tras el trabajo profesional es que antes y durante el trabajo técnico informático hay que Analizar y planificar los siguientes pasos a tomar.

Por ellos vamos a seguir el siguiente orden de trabajo:

1. Análisis

Nunca te enfrentes a un problema sin antes haber realizado el análisis conveniente.

Esto ayuda a tomar el acercamiento más correcto posible inicialmente, que siempre sufre modificaciones ya que no hay plan que aguante el choque con la realidad con lo que concluyo, hay que ser analítico antes, durante y después de todo trabajo técnico.

Con estos principios claros paso a realizar el análisis inicial que luego repercutirá en la plantificación del trabajo técnico a realizar.

Análisis Servicio bNav

Primero hay que definir los requisitos para este servicio, son 2:

- Administrar la información a servir (Interfaz de administración)
- Proveer de información a la aplicación

Comenzando por la parte más sencilla teóricamente la parte de proveer información, el acercamiento es el siguiente, una llamada por URL a este servicio de forma correcta nos proporciona el directorio de la planta y una imagen que contiene el plano de la planta.

Este desarrollo nos va a permitir arrancar la arquitectura de la aplicación y definir algunas estructuras de datos para la Aplicación a posteriori.

Con respecto a la Administración, se trata de una parte más complicada ya que tiene múltiples vistas, control de usuario y contraseña de administración.

Las conclusiones de este análisis se muestran en la Descripción del servicio en el **Desarrollo de la solución (Servicio).**

Análisis App bNav

En el caso de la aplicación Android el objetivo es consumir los recursos proporcionados por el servicio y mostrarlos en una interfaz sencilla para que el usuario pueda moverse por el edificio.

Para ello proporcionaré una serie de transiciones en las cuales llegará hasta ver el mapa y su destino.

Las conclusiones de este análisis se muestran en la Descripción del servicio en el **Desarrollo de la solución (Aplicación).**

2. Planificación

Iniciaré el desarrollo desde la parte más fácil para mí ya que tengo ciertos conocimientos de desarrollo web aunque mi PHP está muy oxidado ya que hace más de dos años que no trabajo con esta tecnología, mis conocimientos de HTML5, CSS3 y JavaScript están al día por mi trabajo.

El orden de trabajo para todo el desarrollo queda plasmado en el apartado siguiente marcándolas con prioridad. Rojo 1 Amarillo 2 Verde 3

3. Desarrollo

a. Servicio bNav

Devolución de la información, es lo realmente necesario para que la aplicación pueda funcionar.

Herramienta de administración, como comienzo para trabajos futuros.

b. App bNav

Desarrollo integral de la misma que consuma los ficheros que provee el servicio.

Al finalizar el proyecto estableceremos los tiempos gastados para cada una de estas fases.

5. Desarrollo integral de la solución

Para poder consumir desde una aplicación los planos y el directorio desarrollaré primeramente el servicio que proporcionará los mismos. Lo denomino **bNavService**, se trata de un servicio web desarrollado en PHP principalmente junto con tecnologías web adyacentes.

5.1. Desarrollo de la solución (Servicio)

1. Descripción

Comenzamos el desarrollo de la solución con el servicio al que nos conectaremos para obtener el mapa y el directorio de la planta en la que nos encontramos.

Este servicio tiene 2 formas de funcionamiento:

1. Interfaz de administración
2. Consumo de servicio por Aplicación.

Para la implementación de este servicio nos vamos a basar en las siguientes tecnologías:

1. HTML5 Como lenguaje base de programación web.
   W3ORG(http://www.w3.org/html/logo/)
2. PHP Como lenguaje de programación dinámico en servidor.
   PHP.NET (http://php.net/)
3. JavaScript y CSS3 como lenguajes complementarios a los anteriores. **JS**(https://developer.mozilla.org/en-US/docs/Web/JavaScript) y **CSS3**(https://developer.mozilla.org/en-US/docs/Web/CSS/CSS3?redirectlocale=en-US&redirectslug=CSS%2FCSS3)
4. XML como estructura de datos donde almacenaremos el directorio y todas las configuraciones. **Extensible Markup Languaje** (http://www.w3.org/XML/)

Pasamos a una descripción más detallada de los **modos de funcionamiento** del servicio.

2. Interfaz de administración Servicio web

La interfaz de administración es un frontal web en la cual el administrador del servicio de posicionamiento gestiona los mapas, directorio, usuarios, contraseñas y generación de códigos QR para cada planta.

Primeramente el administrador ha de acceder a la herramienta desde un formulario de acceso.

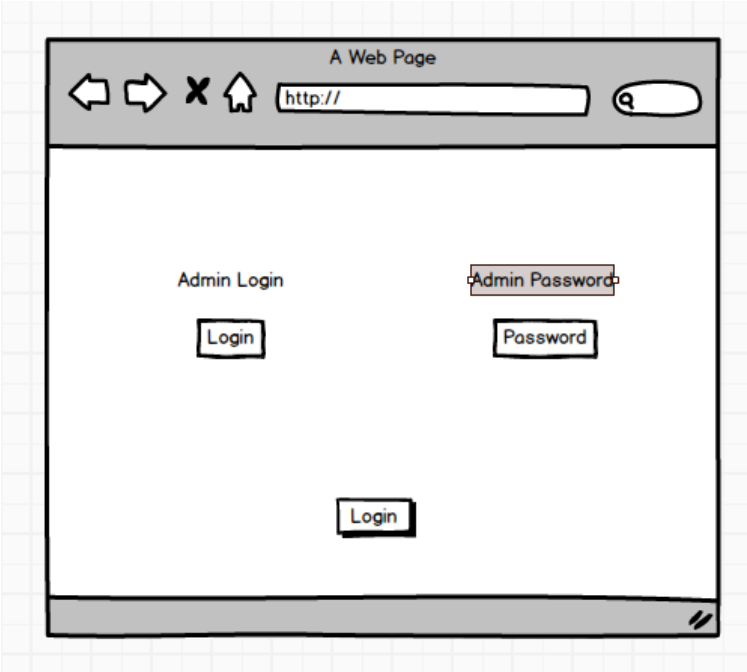


Ilustración 3 Login Mock Service

Una vez dentro dispone veremos un listado de las plantas y cada una con las siguientes opciones:

0. *Gestión planta.*
    1. Nueva planta
    2. Ver detalles planta.
    3. Modificar detalles planta.
    4. Eliminar planta.
    5. Generar QR.
    2. Salir de administración

1. *Nueva planta*

    En esta vista el administrador tendrá que introducir el mapa de la planta y luego rellenar los campos del XML del directorio.

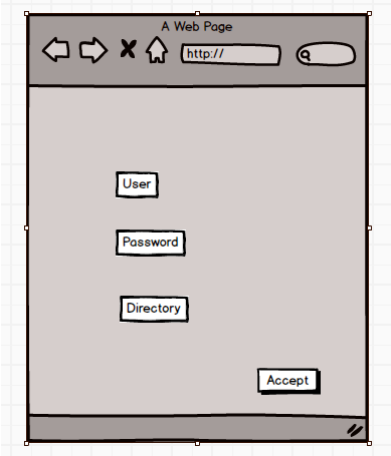


Ilustración 4 Añadiendo Mock

2. ***Detalles de la planta***

En esta vista el administrador tendrá una presentación de la imagen que constituye el mapa y los elementos del directorio sobrepuestos al mismo. Prácticamente lo mismo que en la aplicación Android pero sin ubicar a la persona.

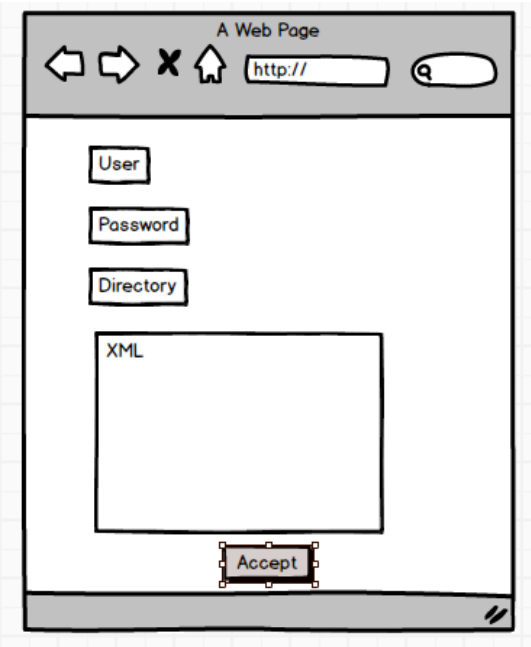


Ilustración 5 Detalle Mock

3. ***Modificar detalles de la planta***

Esta vista de modificación nos permite modificar datos del Directorio y cambiar la imagen, igual que la vista de añadido.

4. ***Eliminar planta***

Se trata de un botón en la vista principal que nos permite eliminar una planta tras una petición de confirmación. Se borran los archivos de directorio y la imagen del mapa.

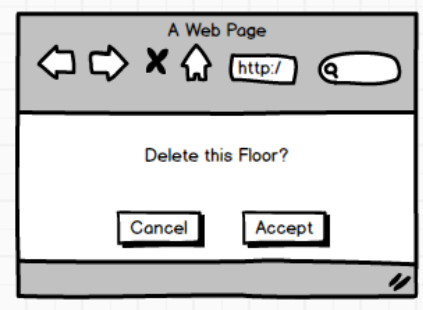


Ilustración 6 Borrar planta Mock

5. ***Generación de QR***

Esta vista nos proporciona una interfaz en la cual pinchando sobre la imagen del mapa introduciremos los datos necesarios para generar el QR para la aplicación.

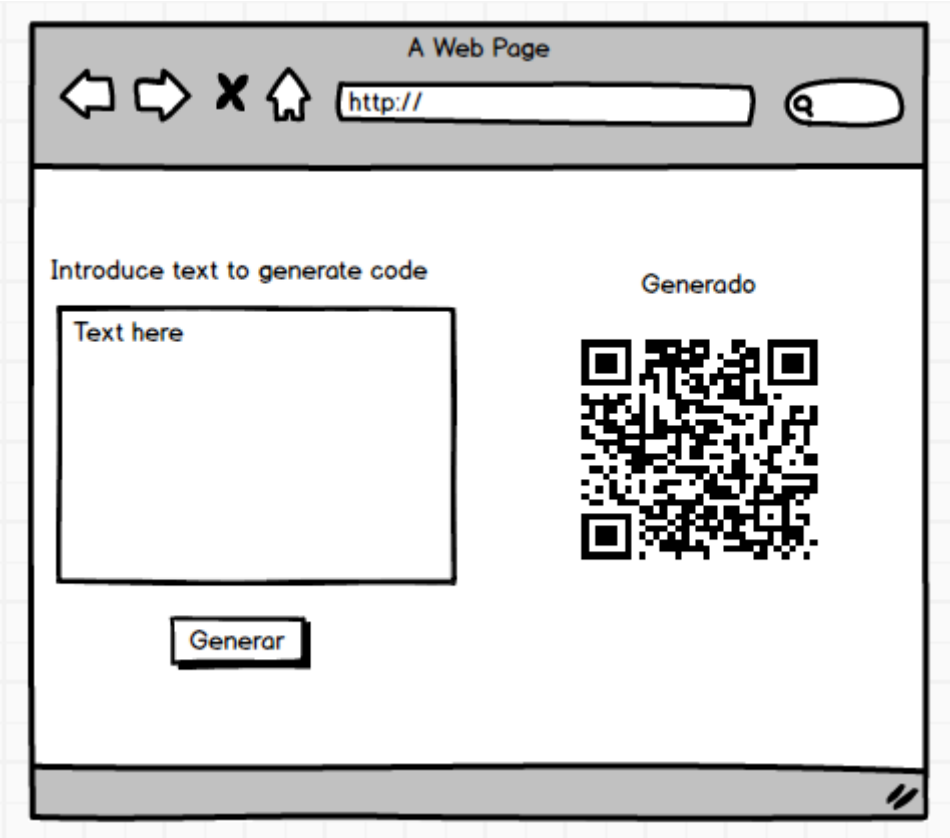


Ilustración 7 Generate QR Mock

3. Consumo de servicio

Esta parte no es nada visual, la aplicación, tras leer uno de los códigos QR generados, se conecta a la red de invitados del edificio y una vez dentro llama al servicio pasando como parámetros el usuario y contraseña leídos en el código QR.

El servicio devolverá el directorio y el mapa para que la aplicación monte el sistema de información para el usuario.

2. Diario de desarrollo

Comienzo el desarrollo buscando recursos para poder documentarnos durante el proceso, referencias sobre las tecnologías que me planteo utilizar.

A continuación viendo los requisitos de mi aplicación web, monto un apache con PHP como servidor web a través de un **XAMPP** (http://www.apachefriends.org/es/xampp.html).

Una vez montado el servidor web, usando el editor de textos **Sublime Text 2** creamos un nuevo archivo denominado **index.php.**

En un primer momento comienzo con la parte más sencilla posible, una plantilla HTML5 vacía y un texto básico lanzado por PHP y probando que JavaScript y CSS funcionan correctamente.

```html
<!DOCTYPE HTML>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>bnav</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <section>
    <p id="css">Hello World CSS</p>
    <?php
     Echo "Hello, World!";
    ?>
    <script type="text/javascript">
    // Exercise the .helloWorld template
    prompt("Please enter your name","Harry Potter");
    </script>
    </section>
</body>
</html>
```

Ilustración 8 Hola Mundos Variados

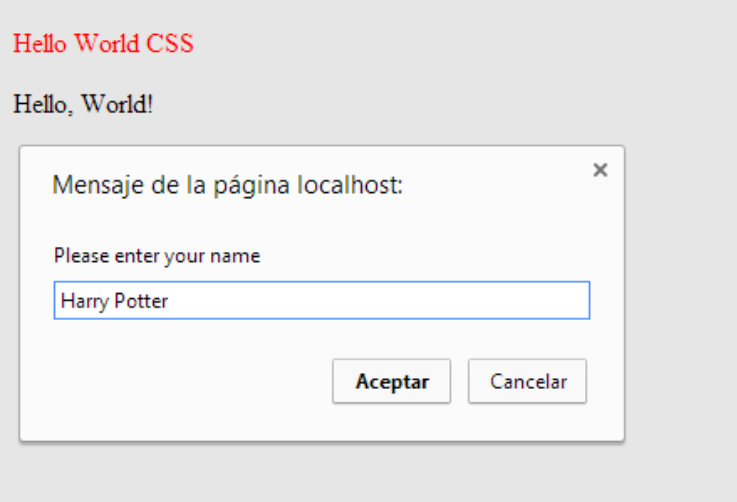


Ilustración 9 Resultado Hola Mundo

Comprobado el funcionamiento del entorno de desarrollo comienzo con el desarrollo como tal, en este caso arrancando con la parte teóricamente más sencilla que es el consumo de servicio por el app.

Tenemos una entrada a través de una conexión http al servicio, por ejemplo: http://bnav.upv.es/index.php?user=level1&pass=passlvl1

Este servicio estaría en un subdominio de la etsinf y nos conectamos a través de la red de invitados usando el QR y ese QR a su vez llamaría a esta URL con el usuario para el nivel en el que nos encontremos.

Necesitamos pues que nuestro fichero index.php pueda recibir parámetros y trabajar con ellos.

Esto podemos hacerlo muy fácilmente con PHP usando
$_GET["nombreParametro"]

Con este método podemos obtener los parámetros mediante una llamada a esa URL, a continuación verificaremos si los parámetros son correctos verificándolo dentro de nuestro archivo de contraseñas.

Primero vamos a capturar el usuario y la contraseña y los mostramos por pantalla, si no existen suficientes parámetros devolvemos un mensaje de error.

```
<section>
<?php
if(isset($_GET["user"]) && isset($_GET["pwd"])){
    echo "<h1>Hello " . $_GET["user"] . "</h1>";
    echo "<h1>Your password is:" . $_GET["pwd"] . "</h1>";
}else{
    echo "<h1>Not enought params</h1>";
}
?>
</section>
```

Ilustración 10 Captura de parámetros

Haciendo una llamada sin parámetros obtenemos el siguiente resultado:

Ilustración 11 Parámetros insuficientes o erróneos

Mientras que con los parámetros adecuados obtenemos:

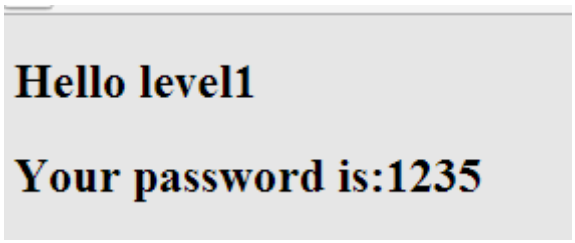


Ilustración 12 Parámetros correctos

Seguidamente paso a la validación de dichos parámetros primero mediante strings hardcoded en la aplicación:

User:level1

Pwd:test1

```php
<?php
if(isset($_GET["user"]) && isset($_GET["pwd"])){
    if($_GET["user"]=='level1' && $_GET["pwd"]=='test1'){
        echo "<h1>Hello Visitor your level is: " . $_GET["user"] . "</h1>";
        echo "<h1>Your password is:" . $_GET["pwd"] . "</h1>";
    }else{
        echo "<h1>Wrong user/pwd</h1>";
    }
}else{
    echo "<h1>Not enought params</h1>";
}
?>
```

Ilustración 13 Validación

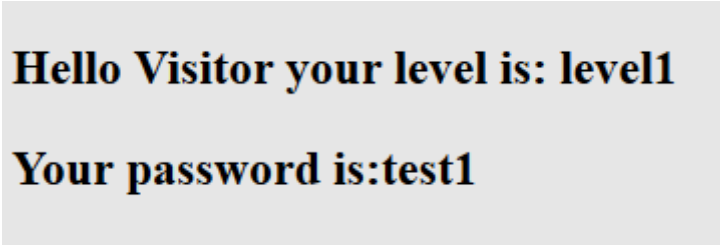


Ilustración 14 Validación correcta

Ilustración 15 Fallo en la validación

Tras esta validación hardcoded paso a la implementación de la lectura de XML para validar los usuarios y contraseñas, definimos un formato inicial de XML para esto, denominamos el archivo config.xml, se encuentra en la carpeta **resources**, todos los recursos de la aplicación a excepción de **index.php** están en dicha carpeta para no estar expuestos (seguridad de la aplicación).

Este es el aspecto inicial del xml de configuración.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<configs>
  <level>
    <user>level1</user>
    <pass>test1</pass>
  </level>
    <level>
    <user>level2</user>
    <pass>test2</pass>
  </level>
</configs>
```

Ilustración 16 Configuración Inicial

Con esta estructura definida pasamos a la parte de lectura de dicho XML mediante PHP, usando la clase simplexml (http://php.net/manual/es/book.simplexml.php) que me permite de forma fácil y rápida acceder a un XML en el sistema de ficheros y leer de sus elementos.

El método de validación de credenciales no se encuentra en el archivo **index.php** si no que está en un nuevo archivo PHP, llamado **checkCredentials.php** donde realizamos la lectura y comprobación de las credenciales.

Solo importamos este archivo si se han proporcionado credenciales en un formato valido de esta manera ahorramos llamadas no necesarias y por lo tanto recursos de servidor.

El código queda de la siguiente forma:

```php
<?php
$user = $_GET["user"];
$pwd = $_GET["pwd"];
if(isset($user) && isset($pwd)){
    include './resources/checkCredentials.php';
    $validation = check($user,$pwd);
    if($validation){
        echo "<h1>Hello Visitor your level is: " .$user . "</h1>";
        echo "<h1>Your password is:" . $pwd . "</h1>";
    }else{
        echo "<h1>Wrong user/pwd</h1>";
    }
}else{
    echo "<h1>Not enought params</h1>";
}
?>
```

Ilustración 17 Validación XML1

Como se puede ver realizamos una llamada a la función **check** que se encuentra en el fichero **checkCredentials.php**

Este es el código de lectura y validación:

checkCredentials.php | index.php | config.xml

```php
<?php
function check($user,$pwd)
{
    $configFile = './resources/config.xml';
    if (file_exists($configFile)) {
        $xml = simplexml_load_file($configFile);
        foreach ($xml->level as $level) {
            $xmlUser = $level->user;
            $xmlPwd = $level->pass;
            if($xmlUser==$user){
                if($xmlPwd==$pwd){
                 return true;
                }
            }
        }
        return false;
    } else {
        exit('Failed to open config.xml.');
    }
}
?>
```

Ilustración 18 checkCredentials.php

Comprobamos que exista el archivo de configuración antes de realizar cualquier tipo de operación sobre el mismo, una vez validado recorremos el XML en busca de los parámetros recibidos y en caso de existir devolvemos true, en caso negativo false.

Los resultados en los casos:

Usuario /contraseña correcto

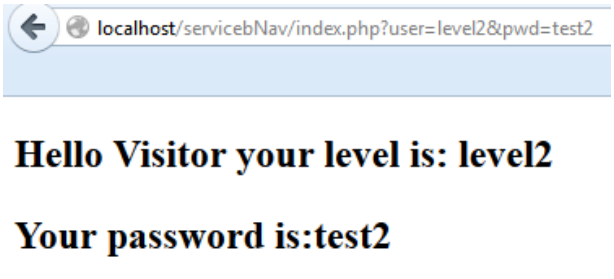


Ilustración 19 Ok

Usuario /contraseña incorrecto

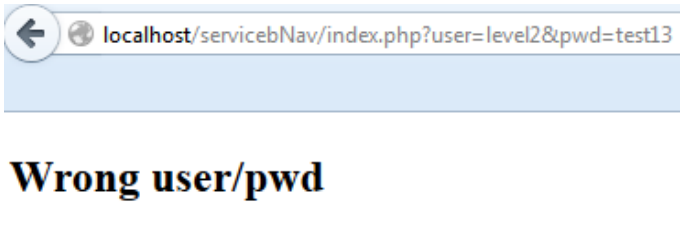


Ilustración 20 Nok

Comprobado el servicio de validación pasamos a la parte de devolución de la información solicitada, el directorio y el mapa.

Trabajando con el mismo esquema donde trabajamos con un fichero aparte que solo incluimos en nuestro **index.php** si es necesario creamos el fichero **returnElement.php**

En el mismo desarrollo la función **returnLevelFiles** a la cual pasándole como parámetro el level se nos devuelve el mapa y el xml correspondiente para dicho piso.

Defino la estructura de la siguiente manera, cada level tiene su directorio en el cual se almacenan los 2 ficheros, **map.jpg y bDirectory.xml.**

Así pues en **./resources/levels/level1/** tendremos disponibles estos dos archivos para servirlos al usuario.

Por el momento voy a usar un XML y un mapa placeholder ya que solo me interesa devolver el elemento solicitado, no lo que contenga el mismo.

Me he visto bloqueado en el desarrollo de esta parte porque solo consigo devolver un archivo queriendo hacerlo 2 veces espero solucionarlo consumiendo el archivo desde la aplicación con una URL.

Una vez completada la parte de devolución de los archivos paso a la parte de la interfaz de administrador que requiere algo de gusto con el diseño y unos retos de programación diferentes.

Arranco el desarrollo de esta parte con la pantalla de login para el administrador, disponible en el mismo **index.php.**

Vamos a usar un formulario HTML5 obtenido en la web de tutoriales sobre html5 http://www.hongkiat.com/blog/html5-loginpage/ al cual le realizo las modificaciones necesarias para que se ajuste a las necesidades de nuestra aplicación.

Montamos el formulario en nuestro **index.php**

```
<h1>bNav Admin Login</h1>
<form name="login" action="index.php" method="get" accept-charset="utf-8">
    <ul>
        <li>
            <label for="admin">User</label>
            <input type="text" name="admin" placeholder="User" required>
        </li>
        <li>
            <label for="password">Password</label>
            <input type="password" name="pwd" placeholder="Password" required></li>
        <li>
            <input type="submit" value="Login">
        </li>
    </ul>
</form>
```

Ilustración 21 Código LoginForm

El aspecto es bastante desolador sin CSS así que preparo una hoja de estilos CSS que será en principio la misma para toda la web denominada **style.css**.

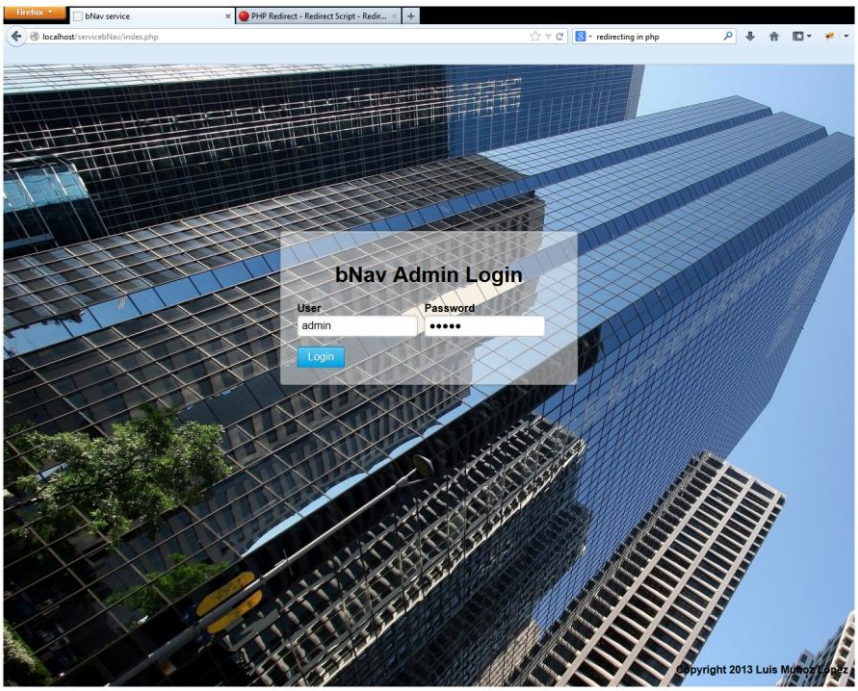


Ilustración 22 Login Form

Trataremos el login como un auto submit a nuestra página **index.php** y trato en el mismo la lógica para el login, comprobando el usuario y contraseña del administrador de manera similar a check Cedentials . Tenemos un nuevo archivo de configuración llamado **admin.xml**

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <configs>
3    <adminData>
4      <admin>admin</admin>
5      <pass>adminpwd</pass>
6    </adminData>
7  </configs>
```

Ilustración 23 admin.xml

Para leer dicho archive de configuración uso una nueva función denominada checkAdmin del archivo **checkAdminCredentials.php**

```php
<?php
function checkAdmin($admin,$pwd)
{
    $configFile = './resources/admin.xml';
    if (file_exists($configFile)) {
        $xml = simplexml_load_file($configFile);
        foreach ($xml->configs as $configs {
            $xmlUser = $configs->admin;
            $xmlPwd = $configs->pass;
            if($xmlUser==$admin){
                if($xmlPwd==$pwd){
                 return true;
                }
            }
        }
        return false;
    } else {
        exit('Failed to open admin.xml.');
    }
}
?>
```

Ilustración 24 checkAdminCredentials

Con esta función creada implemento el login de administración que n caso satisfactorio lleva a la página de administración.

```php
if(isset($_GET["admin"])){
    if(isset($_GET["pwd"])){
        $user = $_GET["admin"];
        $pwd = $_GET["pwd"];
        include './resources/checkAdminCredentials.php';
        $validationAdmin = checkAdmin($user,$pwd);
        if($validationAdmin){
            header( 'Location: ./resources/admin.php');
        }else{
            echo '<header>';
            echo    '<h2>Result Admin params</h2>';
            echo '</header>';
            echo "<p>Wrong user/pwd</p>";
        }
    }
}
?>
```

Ilustración 25 Validación Admin

Esta forma de pasar a la página de administración resulta realmente endeble ya que no hay seguridad real aparte de la redirección con lo cual añado una sesión en PHP para reforzar este tema.

Para ello, en caso de que la validación de los parámetros de entrada sea correcta creamos una sesión con el usuario del administrador que será buscada por todas las páginas secundarias en las cuales se muestran y crean los contenidos para la aplicación.

```php
$validationAdmin = checkAdmin($user,$pwd);
if($validationAdmin){
    session_start();
    $_SESSION['admin']  = $user;
    header( 'Location: ./resources/admin.php');
```

Ilustración 26 Inicio de sesión

```php
<?php
    session_start();
    if(!isset($_SESSION['admin'])){
        echo "<h1>Sesión no valida</h1>";
        die();
    }
?>
```

Ilustración 27 Validamos sesión

En esta vista de administración es donde mostramos el listado de plantas y las opciones que nos proporciona la herramienta para gestionar las plantas, para ello vamos a leer el archivo config.xml para generar dicho listado.

Además defino el camino para el resto de opciones que deseamos tener accesibles: vista, añadido, modificación, eliminación, generación de QR's.

Generando el listado con listas no ordenadas **<ul>** ya que las tablas se están abandonando en html este es el código que produce el listado básico que más adelante enriqueceré con estilos e imágenes para los botones.

```php
<ul style="display: table;">
  <li style="display: table-row;">
    <ul class="fila title"><li>Floor</li><li>Administrator</li><li>Password</li><li>Options</li><li><a href="#">Add Floor</a></li></ul>
  </li>
  <?php
    $configFile = 'config.xml';
    if (file_exists($configFile)) {
        $contador = 0;
        $xml = simplexml_load_file($configFile);
        foreach ($xml->level as $level) {
            $xmlUser = $level->user;
            $xmlPwd = $level->pass;
            echo "<li style='display: table-row;'>";
            echo "<ul class='fila'><li>" . $contador . "</li><li>" . $xmlUser . "</li><li>" .$xmlPwd. "</li><li>Options</li><li><a href='./mod.php?level=". $xmlUser ."'>mod</a><a href='./view.php?level=". $
                xmlUser ."'>view</a><a href='./del.php?level=". $xmlUser ."'>del</a><a href='./qr.php?level=". $xmlUser ."'>qr</a></li></ul>";
            echo "</li>";
            $contador++;
        }
    } else {
        exit('Failed to open config.xml.');
    }
    ?>
    <li style="display: table-row;">
        <ul class="fila title"><li>Floor</li><li>Administrator</li><li>Password</li><li>Options</li><li><a href="../index.php?logOff=true">Log Out</a></li></ul>
    </li>
</ul>
```

Ilustración 28 Listado administración

El resultado con el contenido actual es:

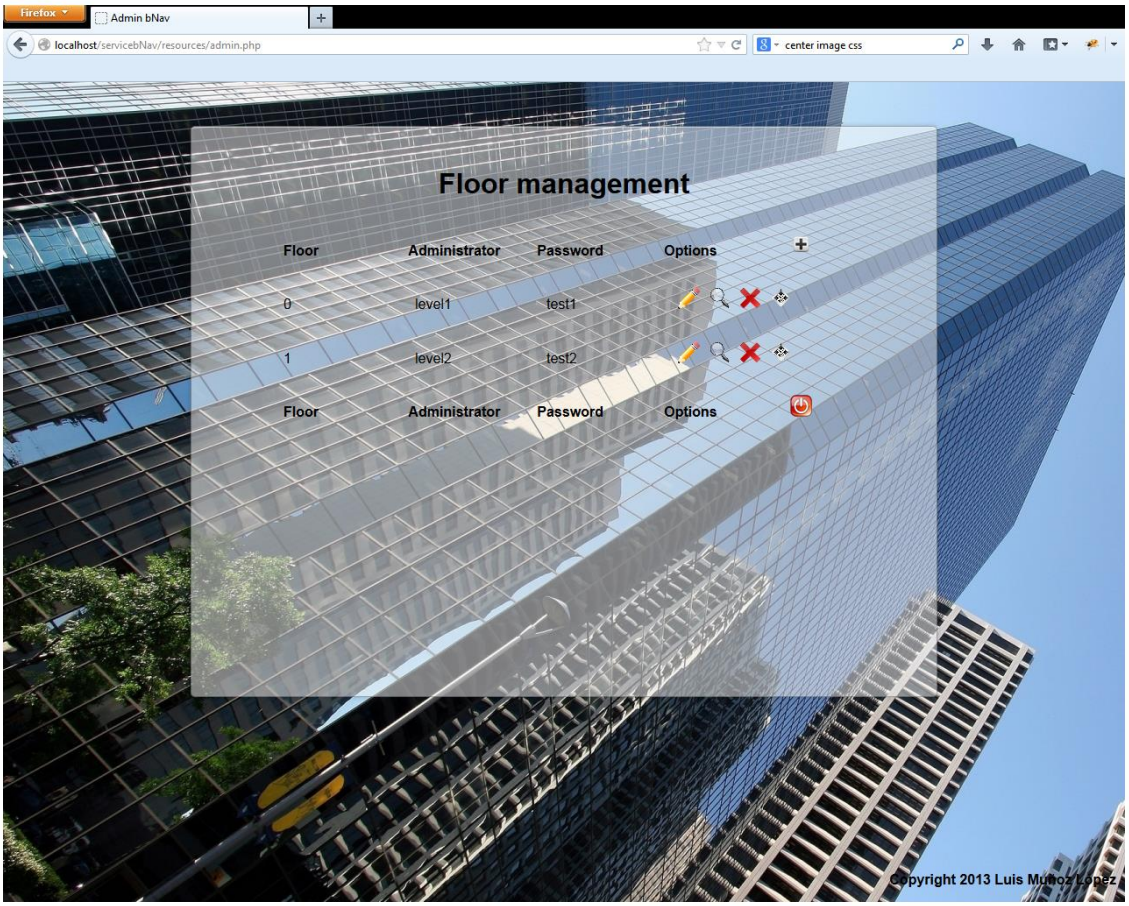


Ilustración 29 Vista de administración

Continúo enriqueciendo la vista con botones y mejorando el estilo de la vista en general para posteriormente centrarme en la vista de cada planta.

Primero creo cada página asociada a cada acción de estos menús aunque sea solo para poner un aviso de que está funcionando, con un botón para volver.

Creo los archivos **mod.php, view.php, add.php y qr.php**, con una plantilla similar a la de administración.

Les paso por parámetro que planta sobre la que interactuara dicha vista con una llamada del tipo **./view.php?level=level2**

A continuación el ejemplo de una llamada a la vista de modificación que se encuentra en construcción.

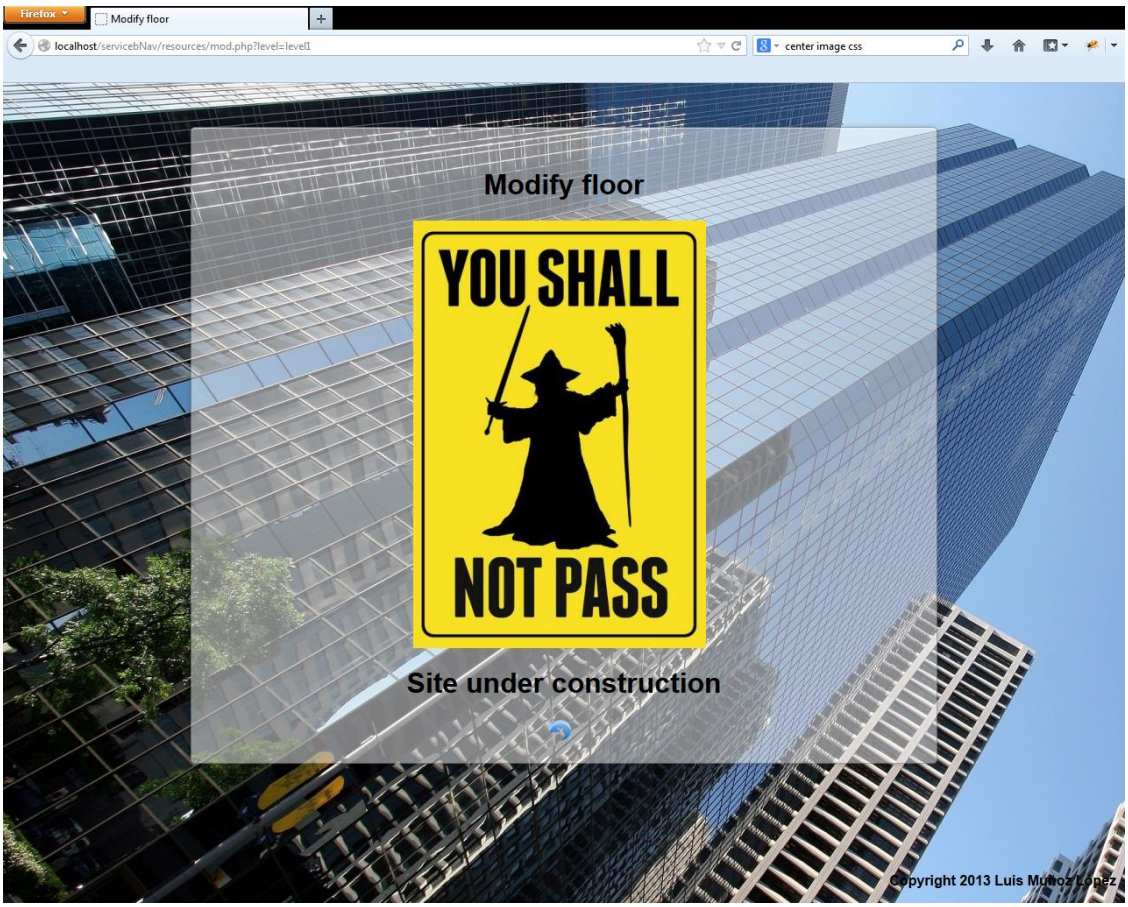


Ilustración 30 En construcción

El siguiente paso es la vista en detalle de una planta, en esta según lo planteado he de mostrar los datos de la planta, lo que significa que hay que definir como es el mapa y la forma de superponer en la imagen los puntos definidos en el directorio.

Para ello primero definir qué datos va a tener en el XML de directorio (**bDirectory.xml**).

- Coordenadas: X e Y sobre la imagen del mapa
- Identificador del punto
- Notas.

Mostraremos cada uno de los elementos de dicho directorio, usuario, contraseña y el mapa con las posiciones superpuestas que será sin duda la parte más complicada.

Partimos de la plantilla “en construcción y maquetando los elementos a mostrar, que luego daremos la dinamicidad necesaria.

Buscando la forma de mostrar el archivo bDirectory.xml a través de PHP usando la etiqueta **pre** para mostrar el código.

La definición del archivo bDirectory.xml queda de la siguiente manera:

index.php | admin.php | bDirectory.xml

```xml
<?xml version="1.0" encoding="utf-8" ?>
<configs>
    <level>
        <user>level1</user>
        <pass>test1</pass>
    </level>
    <element>
      <x>635</x>
      <y>630</y>
      <id>Despacho profesor Isaac Asimov</id>
      <note>Nota aparte</note>
    </element>
    <element>
      <x>642</x>
      <y>769</y>
      <id>Despacho profesor Carl Sagan</id>
      <note>Nota aparte2</note>
    </element>
</configs>
```

Ilustración 31 bDirectory definición

Con esta definición clara queda mostrar estos datos de una forma visual más clara que en un archivo XML que es poco informativo para un humano.

Presentamos los campos user y pass en la cabecera para después añadir el mapa de la planta y sobreponemos el id del directorio sobre la misma de forma informativa, similar a como se verá en la aplicación.

La presentación de los datos extraídos en la pantalla se hace con JS sacándolo de lo que ya está presentado en el PRE.

El código PHP para esta vista a continuación:

```php
<section class="mainTable cf">
    <h1>Floor Details</h1>
    <?php
        session_start();
        if(!isset($_SESSION['admin'])){
            echo "<h2>Invalid authentification</h2>";
            die();
        }
        if(!isset($_SESSION['level'])){
            echo "<h2>Level Details</h2>";
            echo "<p id='user'>Usuario: </p>";
            echo "<p id='pass'>Contraseña: </p>";
            echo "<h2>Level Map</h2>";
            $img = 'levels/'. $_GET['level'] .'/map.jpg';
            echo "<img class='flormap' alt='Level map' src=". $img . ">";
            echo "<h2>Level Directory</h2>";
            echo "<section class='background'>";
            echo "<pre class='prettyprint linenums'>";
            $level = 'levels/'. $_GET['level'] .'/bDirectory.xml';
            echo "<code class='language-xml'>" . htmlspecialchars(file_get_contents($level), ENT_QUOTES) ."</code>";
            echo "</pre>";
            echo "</section>";
        }else{
            echo "<h1>Invalid Level</h1>";
        }
    ?>
    <a href="./admin.php"><img id="volver" src="./css/back.png" alt="Return to admin"></a>
</section>
```

Ilustración 32 View.php

El código JavaScrip que muestra los datos sobreexpuestos en la imagen y el resto de datos:

```
<script type="text/javascript">
$(document).ready(function()
{
    var i=0;
    var xArray = new Array();
    var yArray = new Array();
    var idArray = new Array();
    var notesArray = new Array();
    setTimeout(function(){
        var x = $(".tag:contains(x)");
        var y = $(".tag:contains(y)");
        var id = $(".tag:contains(id)");
        var note = $(".tag:contains(note)");
        var user = $(".tag:contains(user)")[0];
        $("#user").append(user.nextSibling.textContent);
        var pass = $(".tag:contains(pass)")[0];
        $("#pass").append(pass.nextSibling.textContent);
        for (i;i<x.length;i++)
        {
            xArray.push(x[i].nextSibling.textContent);
            yArray.push(y[i].nextSibling.textContent);
            idArray.push(id[i].nextSibling.textContent);
            notesArray.push(note[i].nextSibling.textContent);
            $("body").append(
            $('<div>'+ idArray[i] +'</div>')
                .css('position', 'absolute')
                .css('top', yArray[i] + 'px')
                .css('left',  xArray[i] + 'px')
            );
            console.log(xArray[i] + "    " + yArray[i]);
        }

    },1000);
});
</script>
```

Ilustración 33 Muestra de datos JS

A continuación los resultados tal y como se muestran en la vista.

Ilustración 34 Usuario y contraseña

Mapa sobre expuesto con los elementos del directorio.

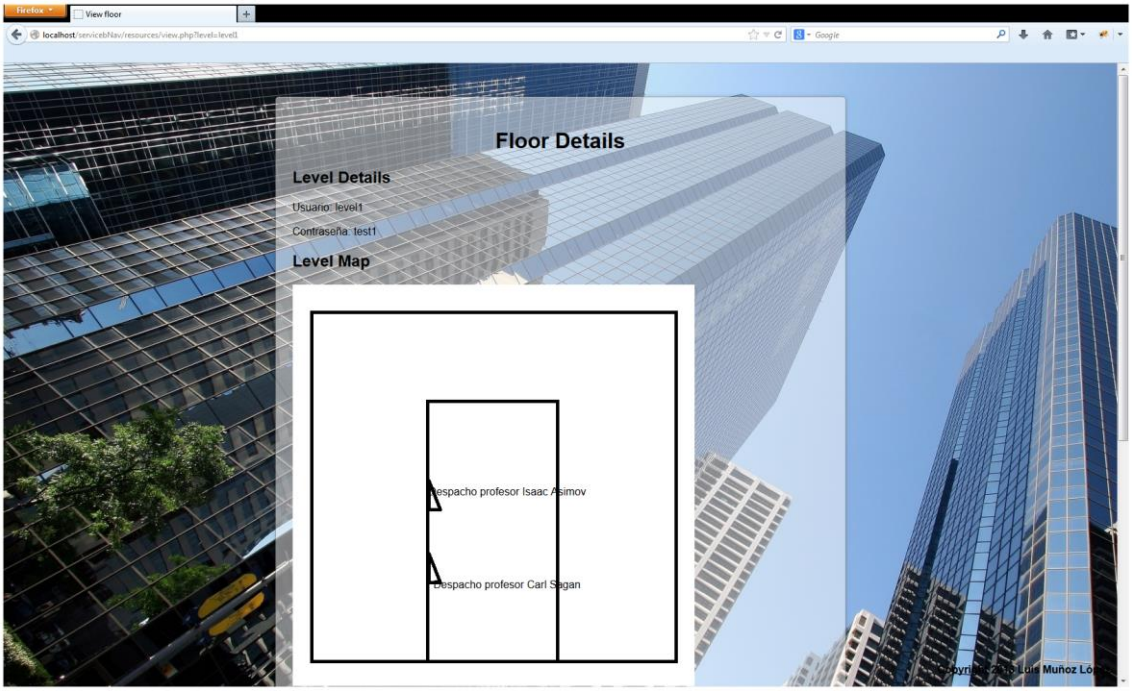


Ilustración 35 Mapa sobre expuesto

Mostrando el XML.

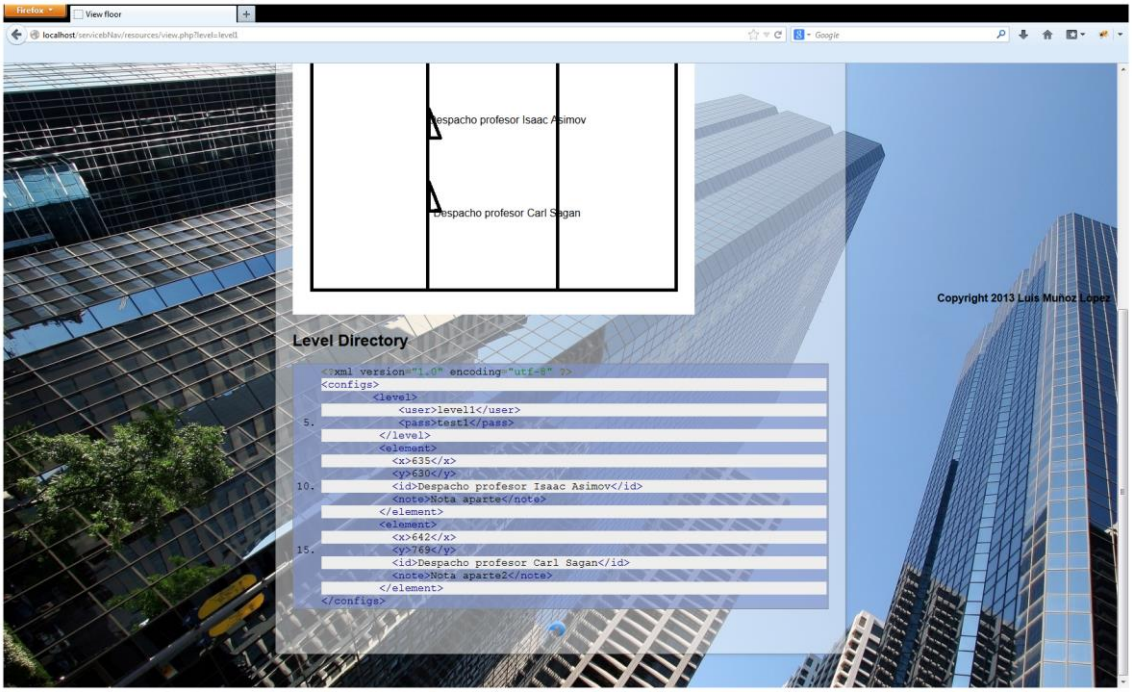


Ilustración 36 XML mostrado

TODO: Cesamos desarrollo del servicio aquí

5.2. Desarrollo de la solución (APP)

1. Descripción

La aplicación Android que he denominado **bNav** es la guía interactiva para edificios que consume los recursos del servicio web.

Consiste en una serie de Actividades con un flujo de navegación muy sencillo y directo que nos permitirá:

- Conectarnos a la red de invitados del edificio.
    a. A través de un código QR.
    b. Mediante un formulario de acceso.
- Seleccionar en que planta estamos.
- Elegir un destino en la planta.
- Obtener nuestra posición actual mediante un QR.

1. Aterrizaje

La primera actividad a crear será la de aterrizaje (Landing) en esta actividad podremos elegir nuestra manera de conectarnos a la red de invitados del edificio y ver el **AcercaDe** en las opciones de la aplicación.

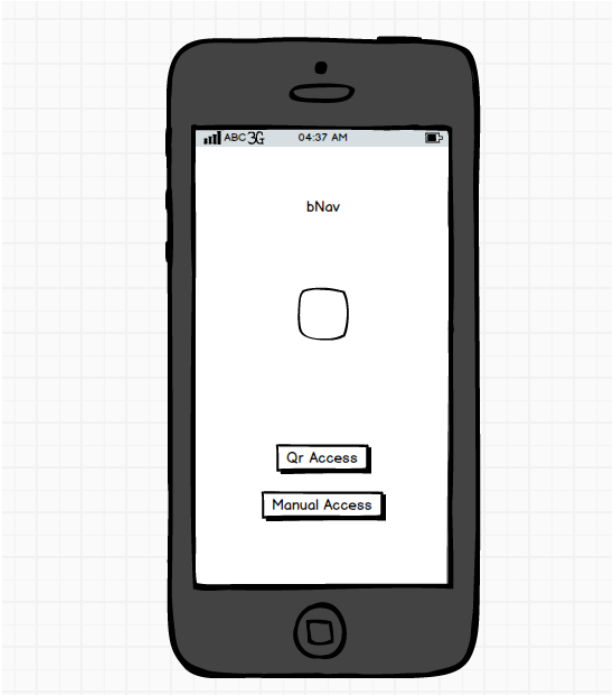


Ilustración 37 Landing Mock

2. Acceso por QR

Entrando en esta vista podremos acceder a la lectura de códigos QR, en caso de lectura correcta por parte de la herramienta externa para lectura de códigos que se integra con bNav (Zxing: https://code.google.com/p/zxing/) .

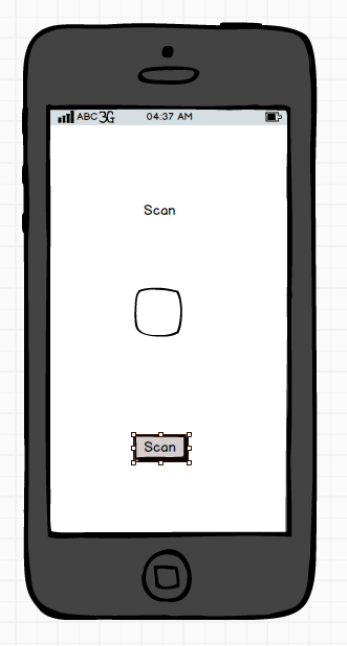


Ilustración 38 QR Mock

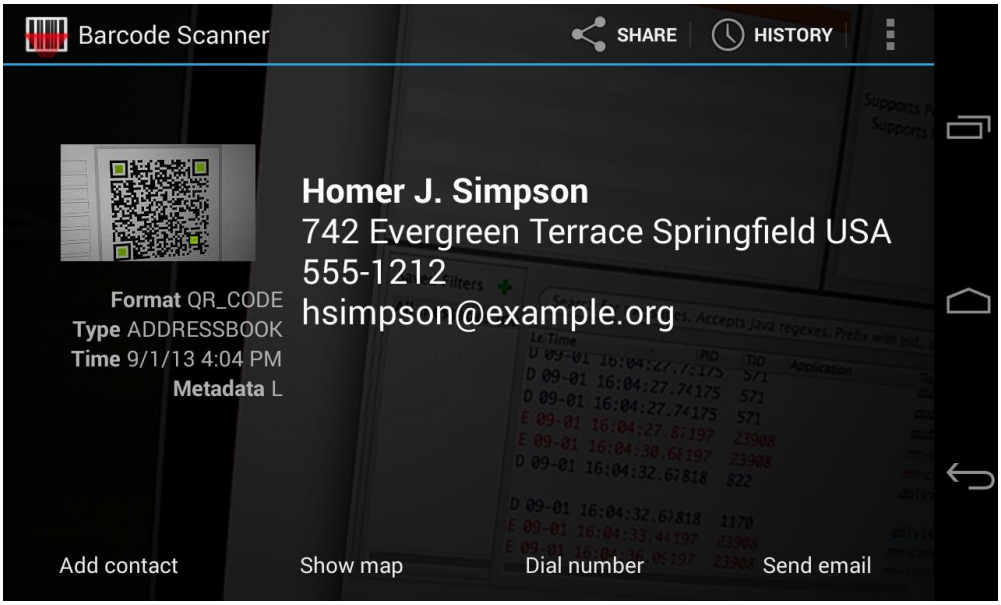


Ilustración 39 Aspecto de Barcode Scanner

3. Acceso por formulario

Se trata de una vista donde introduciríamos los datos de conexión en caso de no disponer de un Código QR que capturar o que ya conozcamos el SSID y la Contraseña compartida.

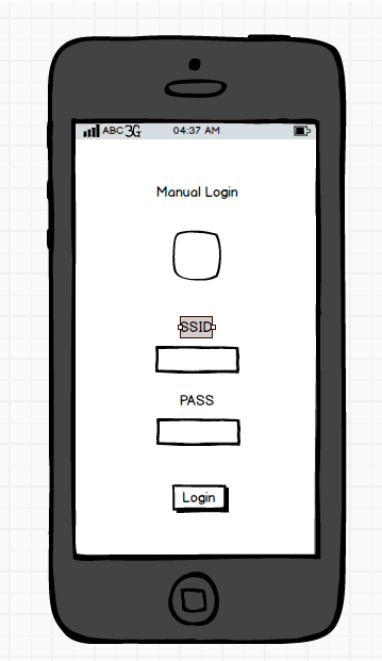


Ilustración 40 Entrada Manual Mock

4. Selección de planta y destino

En esta parte tendrá la mayor parte de la lógica a pesar de que en el frontal solo tenga un par de selectores, realizará el añadido de la red Wifi al teléfono, obtendremos los datos necesarios del servicio y mostrará las opciones disponibles al usuario.

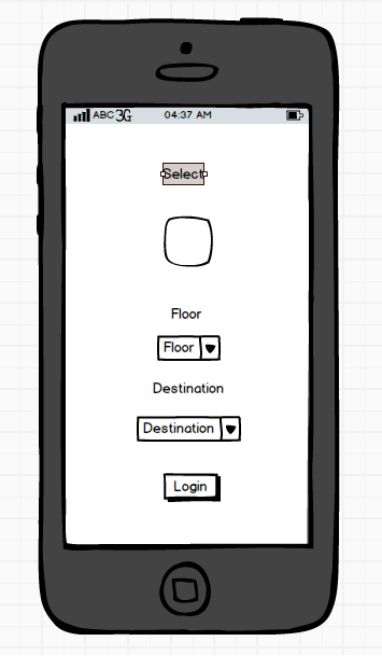


Ilustración 41 Select Mock

5. Vista de Mapa y posicionamiento

Esta es la vista final que el usuario necesita, se trata del mapa con las indicaciones para llegar al destino junto a una brújula y la posición actual en caso de haberse leído un QR, podremos reposicionarnos usando un botón de la interfaz y leyendo de nuevo un QR .

Ilustración 42 Vista Mock

6. Vista de Mapa y posicionamiento

Vista para mostrar la información sobre la información.

2. Diario de desarrollo

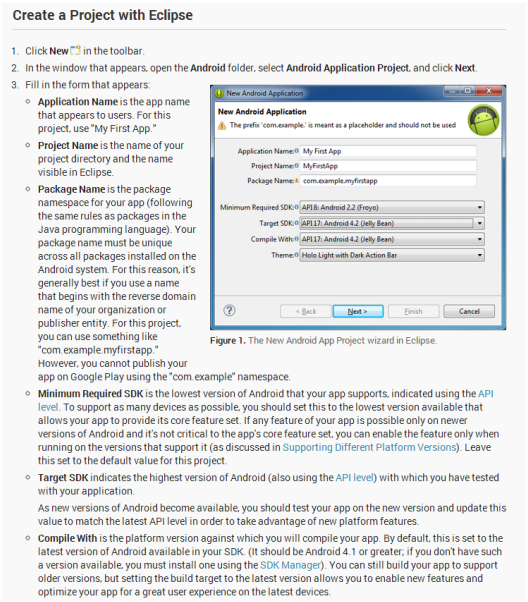
Este desarrollo de la aplicación Android sobre el último entorno de desarrollo promocionado por Google supone varios retos, aprender a programar aplicaciones Android y familiarizarme con un nuevo entorno como es Android estudio muy diferente a los que uso habitualmente. Además de refrescar y mejorar mis conocimientos de programación Java.

Por desgracia el nuevo entorno no está demasiado maduro y la documentación no abunda, aún que está en evolución resulta, por increíble que parezca, mucho más pesado que Eclipse ADT y debido a las restricciones de mi equipo me veo forzado a pasar de Android Studio a Eclipse ADT al ser más ligero, tener más documentación y serme más familiar el IDE.

Inicio el este desarrollo informándome de cómo funciona Android, viendo las relaciones entre los "layouts" y las "activitys", donde se guardan los recursos de la aplicación y otros detalles como el Manifest de la aplicación. Usando la documentación de Google: http://developer.android.com/tools/projects/index.html

Arranco el desarrollo instalando los elementos necesarios para hacer un "Hola Mundo" en Android http://developer.android.com/training/basics/firstapp/index.html El SDK y el Eclipse ADT.

Creo un nuevo proyecto siguiendo los pasos del tutorial.



Create a Project with Eclipse

1. Click **New** in the toolbar.
2. In the window that appears, open the **Android** folder, select **Android Application Project**, and click **Next**.
3. Fill in the form that appears:
   - **Application Name** is the app name that appears to users. For this project, use "My First App."
   - **Project Name** is the name of your project directory and the name visible in Eclipse.
   - **Package Name** is the package namespace for your app (following the same rules as packages in the Java programming language). Your package name must be unique across all packages installed on the Android system. For this reason, it's generally best if you use a name that begins with the reverse domain name of your organization or publisher entity. For this project, you can use something like "com.example.myfirstapp." However, you cannot publish your app on Google Play using the "com.example" namespace.

Figure 1. The New Android App Project wizard in Eclipse.

   - **Minimum Required SDK** is the lowest version of Android that your app supports, indicated using the API level. To support as many devices as possible, you should set this to the lowest version available that allows your app to provide its core feature set. If any feature of your app is possible only on newer versions of Android and it's not critical to the app's core feature set, you can enable the feature only when running on the versions that support it (as discussed in Supporting Different Platform Versions). Leave this set to the default value for this project.
   - **Target SDK** indicates the highest version of Android (also using the API level) with which you have tested with your application.

     As new versions of Android become available, you should test your app on the new version and update this value to match the latest API level in order to take advantage of new platform features.
   - **Compile With** is the platform version against which you will compile your app. By default, this is set to the latest version of Android available in your SDK. (It should be Android 4.1 or greater; if you don't have such a version available, you must install one using the SDK Manager). You can still build your app to support older versions, but setting the build target to the latest version allows you to enable new features and optimize your app for a great user experience on the latest devices.

Ilustración 43 Nuevo proyecto

Me tropiezo con un problema con el emulador Android, tras múltiples intentos consigo configurar un emulador que funcione ya que el emulador no llegaba a arrancar con la última versión de Android en un dispositivo Nexus 4 emulado.

La solución consistió en no utilizar un Nexus 4 emulado como plataforma de pruebas si no un dispositivo inferior ya que mi equipo de desarrollo no parecía lo bastante potente para mover dicho dispositivo emulado.

Con una configuración más básica el emulador arrancaba a tiempo y es posible comprobar el funcionamiento de la aplicación a excepción del uso de sensores.

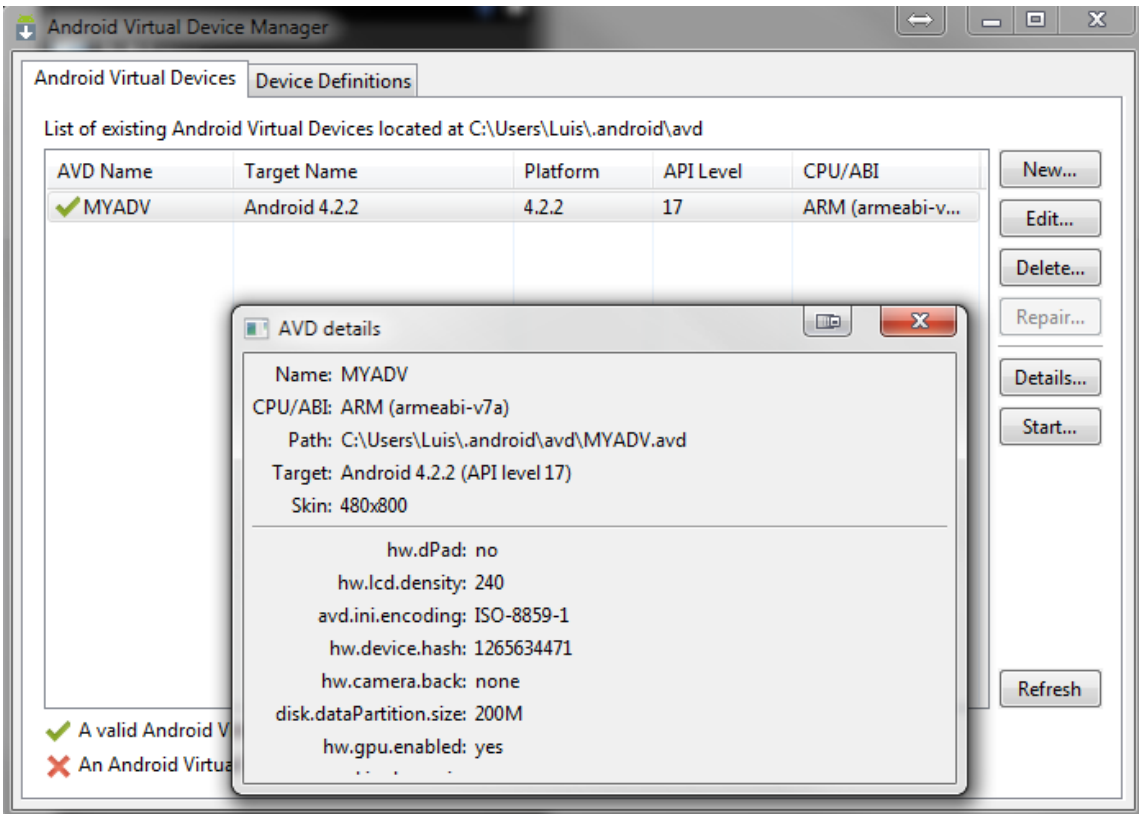


Ilustración 44 Configuración ADV

Obtenemos los resultados esperados.

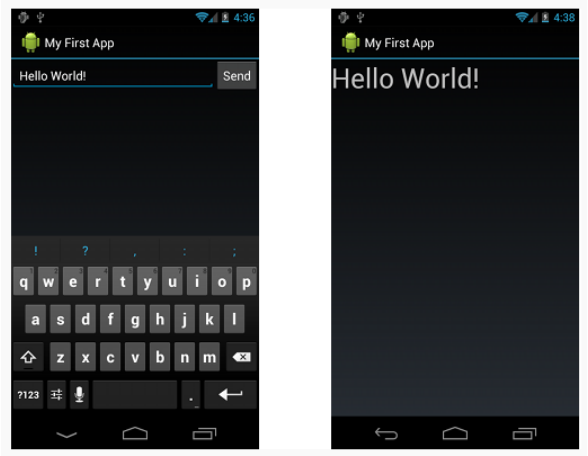


Ilustración 45 Hola Mundo

Con el entorno finalmente montado procedo a arrancar con la primera actividad denominada **Landing.java** y que está asociada a layout **main.xlm**

En Android se definen las interfaces visuales (layouts) a través de archivos XML con propiedades descritas en el API y que nos permiten posicionar y añadir elementos a este lienzo vacío.

A continuación adjunto el código del layout para la Actividad Landinj.java donde definimos los elementos que queremos dentro de nuestra vista.

Defino esta vista como un layout Relativo para poder posicionar en la interfaz gráfica solo arrastrando y moviendo con el ratón.

Ante mi falta de experiencia hasta obtener los resultado adecuados trabajé con varias pruebas y layouts hasta obtener un resultado aceptable.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/RelativeLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center_horizontal"
    android:background="@drawable/degradado"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="30dip"
    tools:context=".Landing" >
```

Esta es la definición del layout con atributos como gravedad el fondo y padding. Muchos de estos atributos se parecen a lo que veríamos en CSS para posicionar pero de manera peculiar como vemos.

`android:layout_width="fill_parent"`
Estamos configurando la vista para que ocupe todo el contenedor

`android:layout_gravity="center_horizontal"`
El concepto de gravedad en Android es muy curioso y algo que no había visto en otros modos de composición de interfaz gráfica. Básicamente es una especie de concepto de fuerza que estira de los elementos de la vista en una misma dirección. Esto nos sirve para unificar el diseño.

`android:orientation="vertical"`
Con este campo dictamos la orientación de la vista entre vertical y horizontal ya que los teléfonos y tablets pueden verse en estos dos modos.

`android:background="@drawable/degradado"`
Este campo nos permite poner el findo a la vista, en este caso hacemos uso de un recurso de sistema en la carpeta drawable, usando un XML para definir dicho degradado, un avance de cómo funciona el sistema de recursos de Android.

`android:id="@+id/RelativeLayout1"`
Este campo es principal para Android, es el identificador del elemento que luego lo identificará para el código Java en las Clases.

```
    <Button
        android:id="@+id/Button01"
        android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/Button02"
        android:onClick="lanzarManual"
        android:text="@string/Manual" />
```

Definición de un botón con atributos típicos, explicados a continuación.

`android:layout_below="@+id/Button02"`
Indicamos dentro del layout relativo, debajo de que elemento se encuentra este elemento.

`android:onClick="lanzarManual"`
Esta es la funcionalidad más directa de este botón nos permite lanzar una acción directamente desde el botón sin tener que declarar Listeners en el código Java.

`android:text="@string/Manual" />`
Con este campo se define el texto del botón en cuestión, se podría definir como un string directamente pero he desarrollado la aplicación para que se traté de la forma sugerida por las recomendaciones de desarrollo de Google, usando un archivo de Strings como recurso, haciendo así viable el uso de varios idiomas y con un mantenimiento mucho más sencillo.

```
    <TextView
        android:id="@+id/TextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/Button02"
        android:layout_marginBottom="20dp"
        android:gravity="top|center"
        android:text="@string/app_name"
        android:textColor="#000"
        android:textSize="35sp " />
```

Definición de un Texto con atributos típicos, los destacados.

`android:textColor="#000"`
Color del texto.

`android:textSize="35sp " />`
Tamaño del texto.

```
    <Button
        android:id="@+id/Button02"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/TextView1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="152dp"
        android:onClick="lanzarQR"
        android:text="@string/QR" />


    <ImageView
        android:id="@+id/icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_below="@+id/TextView1"
        android:layout_centerHorizontal="true"
        android:contentDescription="@string/app_name"
        android:layout_marginTop="40dp"
        android:src="@drawable/ic_launcher" />
```

Definición de una Imagen con atributos típicos

`android:contentDescription="@string/app_name"`
Los campos imagen necesitan una descripción en caso de que la carga de la imagen falle, como en HTML.

`android:layout_marginTop="40dp"`
Campo típico similar al de HTML.

`android:src="@drawable/ic_launcher" />`
Decimos de donde cargar la imagen en este caso desde el sistema de fichero local.

```
</RelativeLayout>
```

Cierre de la definición del Layout.

Este es el resultado tal y como se muestra en el diseñador gráfico.

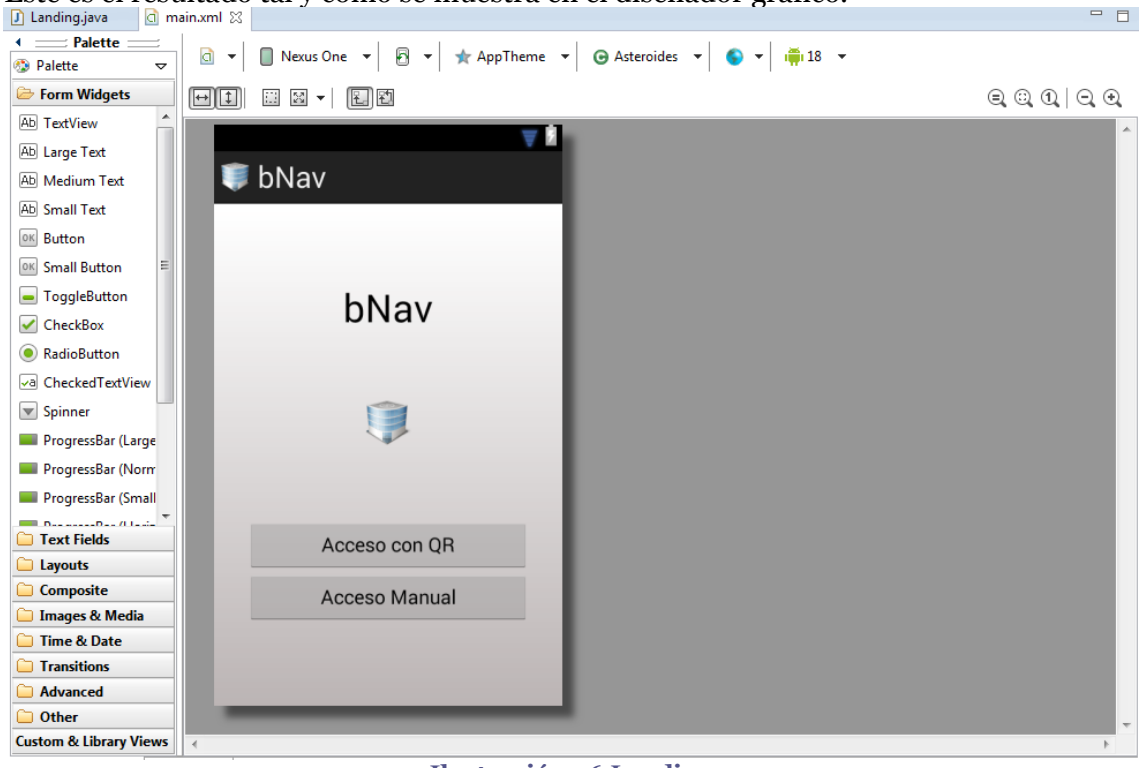


Ilustración 46 Landing

Todo esto puede hacerse ya sea a mano escribiendo el XML o usando la interfaz gráfica colocando botones y posicionándolos de manera sencilla que genera el código XML.

Detalle del fichero de **strings.xml** tanto en inglés como en castellano para hacer la aplicación multi idioma dependiendo el idioma del sistema.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">bNav</string>
    <string name="action_settings">Settings</string>
    <string name="Manual">Manual Access</string>
    <string name="Vista">Map View</string>
    <string name="LaunchQR">Scan QR</string>
    <string name="QR">QR Access</string>
    <string name="Acercade">About</string>
    <string name="Salir">Exit</string>
    <string name="User">SSID</string>
    <string name="Pass">Password</string>
    <string name="Login">Login</string>
    <string name="Scan">Scan</string>
    <string name="Error">Error scaning</string>
    <string name="Navigate">Navigate</string>
    <string name="Selector">Choose</string>
    <string name="Level">Floor</string>
    <string name="Destination">Destination:</string>
    <string name="Position">Position me</string>
    <string name="acercade">This program was developed as PFC by Luis Muñoz
López.</string>


    <string-array name="floors">
        <item>Floor1</item>
        <item>Floor2</item>
        <item>Floor3</item>
        <item>Floor4</item>
        <item>Floor5</item>
        <item>Floor6</item>
    </string-array>

    <string-array name="destinations">
        <item>Greece</item>
        <item>United Kingdom</item>
        <item>Italy</item>
        <item>France</item>
        <item>Germany</item>
        <item>Turkey</item>
        <item>Poland</item>
        <item>India</item>
    </string-array>


</resources>
```

Como puede verse se pueden definir strings y arrays de strings para los dropdowns.

Todo está interfaz necesita de un código que lo soporte para ejecutar acciones al pulsar un botón, llevarnos de una actividad a otra, describo el contenido del código.

```
package es.upv.buildingNavigator;
```

El paquete de la aplicación.

```
import es.upv.buildingNavigator.R;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
```

Librerías que vamos a usar en esta Actividad

```
public class Landing extends Activity {

	@Override
public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);
}
```

Creamos la Actividad usado el Layout main

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
		super.onCreateOptionsMenu(menu);
		MenuInflater inflater = getMenuInflater();
		inflater.inflate(R.menu.menu, menu);
		return true;
}
```

Creamos el menú para opciones.

```
	@Override
public boolean onOptionsItemSelected(MenuItem item) {
		switch (item.getItemId()) {
		case R.id.acercaDe:
			lanzarAcercaDe(null);
			break;
		}
		return true;
}
```

Añadimos los elementos para el menú de opciones.

```
public void lanzarQR(View view) {
		Intent i = new Intent(this, Qrlauncher.class);
		startActivity(i);
}
```

Declarado el método que crea y llama la actividad Qrlauncher a la que llamamos desde un botón de la interfaz.

```
public void lanzarManual(View view) {
		Intent i = new Intent(this, Manual.class);
		startActivity(i);
}
```

Declarado el método que crea y llama la actividad Manual a la que llamamos desde un botón de la interfaz.

```
public void lanzarAcercaDe(View view) {
            Intent i = new Intent(this, AcercaDe.class);
            startActivity(i);
}
```

Declarado el método que crea y llama la actividad AcercaDe a la que llamamos desde un botón de la interfaz.

}

De forma similar se define la Actividad QRlauncher.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/RelativeLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center_horizontal"
    android:background="@drawable/degradado"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="30dip"
    tools:context=".Landing" >

    <TextView
        android:id="@+id/TextView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignTop="@+id/icon"
        android:layout_marginRight="16dp"
        android:layout_marginTop="15dp"
        android:gravity="center"
        android:text="@string/LaunchQR"
        android:textColor="#000"
        android:textSize="35sp " />

    <ImageView
        android:id="@+id/icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/Scan"
        android:layout_alignParentTop="true"
        android:contentDescription="@string/app_name"
        android:src="@drawable/camera" />

    <Button
        android:id="@+id/Scan"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/icon"
        android:layout_alignLeft="@+id/TextView1"
        android:layout_marginBottom="16dp"
        android:onClick="lanzarScan"
        android:text="@string/Scan" />
</RelativeLayout>
```

Obteniendo una vista con este aspecto:

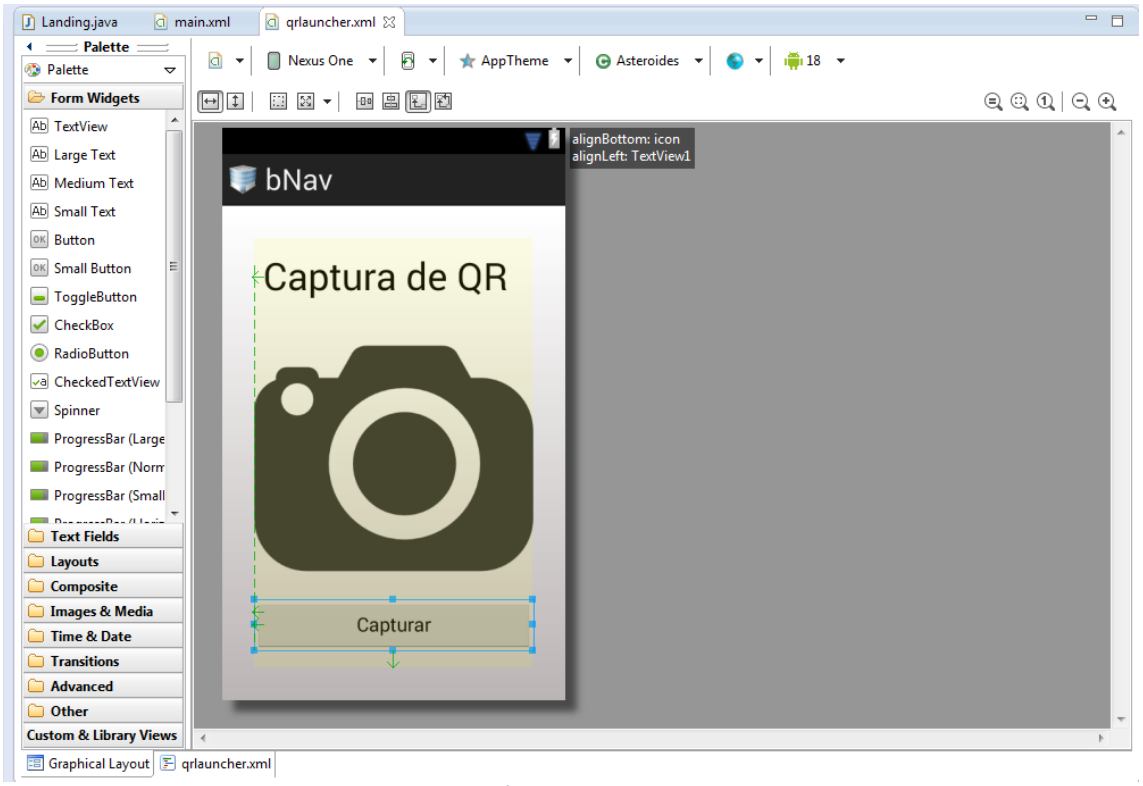


Ilustración 47 QRlauncher

El código sin embargo es algo diferente por la integración con el lector de códigos QR Zebra Crossing (Zxing). https://code.google.com/p/zxing/.

Hemos añadido a nuestro proyecto las librerias necesarias para hacer funcionar este lector de QRs, es necesario que esté instalado en el teléfono pero si no lo tenemos instalado no ofrece instalarlo, esta todo explicado en https://code.google.com/p/zxing/wiki/ScanningViaIntent .

He generado este código para probar en el terminal ya que el simulador no sirve para esto.

Ilustración 48 QR pruebas

```
package es.upv.buildingNavigator;

import es.upv.buildingNavigator.R;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class Qrlauncher extends Activity {

      IntentIntegrator integrator = new IntentIntegrator(this);
```

Declarado un nuevo integrador para interactuar con el lector de Qr.

```
      @Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.qrlauncher);
    }

@Override
public void onActivityResult(int requestCode, int resultCode, Intent intent)
{
    IntentResult result = IntentIntegrator.parseActivityResult(requestCode,
resultCode, intent);
          if (result != null) {
            String contents = result.getContents();
            if (contents != null) {
              showDialog(result.toString());
              lanzarSelector();
            } else {
             showDialog("Error reading");
            }
          }
    }
```

Esta es la manera que tiene Android de obtener información desde el cierre de otra actividad, cuando llamamos a una actividad y esta se cierra podemos recibir los datos que esta otra actividad ha generado, en este caso el lector de Qrs al cerrarse nos transmite los datos obtenidos al escanear el Qr . Una vez con estos datos podemos trabajar con ellos.

```
private void showDialog(String message) {
         AlertDialog.Builder builder = new AlertDialog.Builder(this);
         builder.setMessage(message);
         builder.setPositiveButton(R.string.Salir, null);
         builder.show();
}
```

Método para mostrar por pantalla un mensaje con los datos obtenidos.

```
      public void lanzarScan(View view) {
            integrator.initiateScan();
      }
```

Lanzando el lector de QRs desde nuestra actividad.

```
    public void lanzarSelector() {
          Intent i = new Intent(this, Selector.class);
          i.putExtra("NET", result.toString());
          i.putExtra("FROM","QR");
          startActivity(i);
    }
```

Con los datos obtenidos pasamos a la vista de selección.
}

Esta vista tiene como objetivo escanear un código QR que nos auto configurará el acceso a la red de invitados del edificio para conectarse al servicio de información del edificio.

A continuación el código de la vista manual para conectarnos a la Wifi.

La definición del Layout.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout2"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center_horizontal"
    android:background="@drawable/degradado"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="30dip"
    tools:context=".Landing" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dip"
        android:gravity="center"
        android:text="@string/Manual"
        android:textColor="#000"
        android:textSize="35sp " />

    <ImageView
        android:id="@+id/icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="30dp"
        android:layout_marginBottom="30dp"
        android:contentDescription="@string/app_name"
        android:src="@drawable/ic_launcher" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/User"
        android:textAppearance="?android:attr/textAppearanceMedium" />
```

```
<EditText
    android:id="@+id/user"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="textPersonName" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/Pass"
    android:textAppearance="?android:attr/textAppearanceMedium" />

<EditText
    android:id="@+id/pass"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="textPassword" />

<Button
    android:id="@+id/login"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:onClick="lanzarLogin"
    android:text="@string/Login" />

</LinearLayout>
```

Esta vista tiene como objetivo conectarnos a la red del edificio en caso de no disponer de un QR para escanear por ello tenemos un formulario de Login.

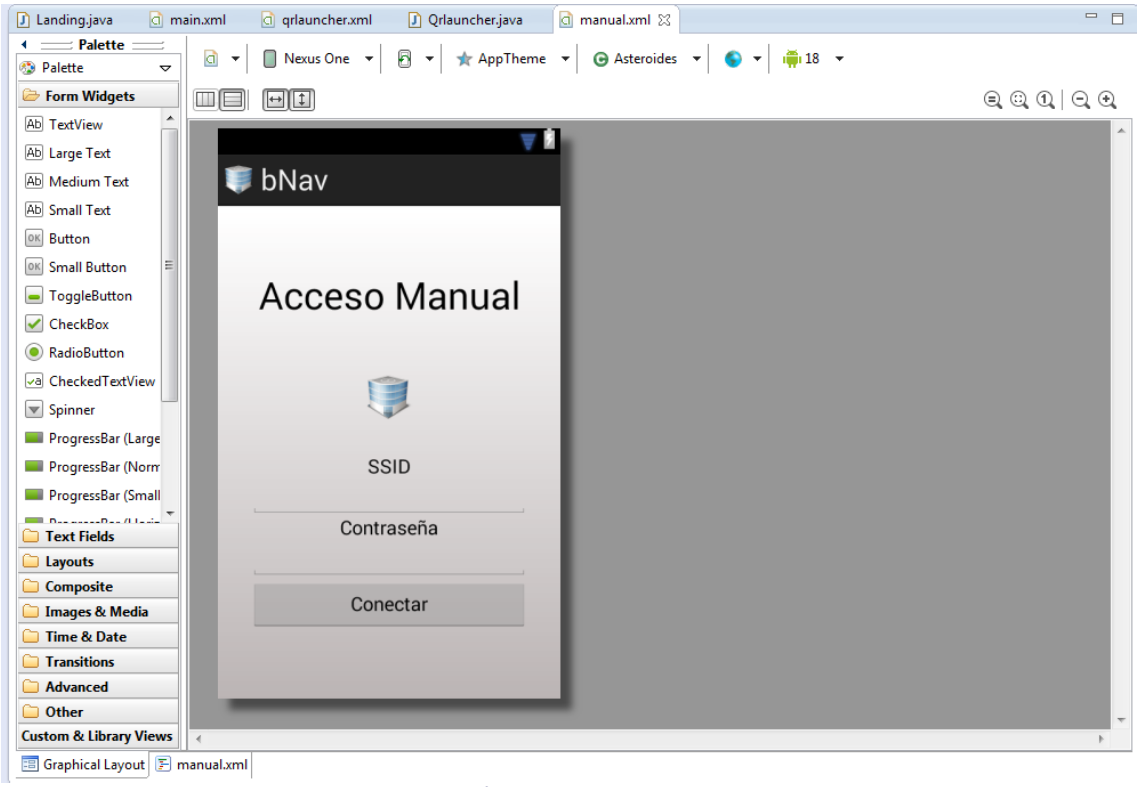


Ilustración 49 Acceso Manual

En la parte de código tenemos lo siguiente:

```
package es.upv.buildingNavigator;

import es.upv.buildingNavigator.R;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;


public class Manual extends Activity {
      Button   mButton;
      EditText user, pass;
```

Crear un botón y los campos para introducir texto.

```
@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.manual);

        mButton = (Button)findViewById(R.id.login);
        user    = (EditText)findViewById(R.id.user);
        pass    = (EditText)findViewById(R.id.pass);

    }
```

Los iniciamos asociándolos a los elementos de la vista.

```
public void lanzarLogin(View view) {
            if(!user.getText().toString().matches("") &&
!pass.getText().toString().matches("")){
            lanzarSelector();
        }else{
            showDialog("Error");
        }
      }
```

Cuando obtenemos los datos llamamos a la vista de selección pasándole los datos como NET.

```
public void lanzarSelector() {
            String login[] =
{user.getText().toString(),pass.getText().toString()};
            Intent i = new Intent(this, Selector.class);
            i.putExtra("NET", login);
            i.putExtra("FROM","Manual");
            startActivity(i);
      }

private void showDialog(String message) {
          AlertDialog.Builder builder = new AlertDialog.Builder(this);
          builder.setMessage(message);
          builder.setPositiveButton(R.string.Salir, null);
          builder.show();
      }
}
```

Una vez obtenidos los datos para la conexión pasamos a la vista de selección que además va a tener la mayor parte de la lógica y las partes más interesantes de la aplicación.

La vista tiene la siguiente definición con selectores en la misma.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout2"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center_horizontal"
    android:background="@drawable/degradado"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="30dip"
    tools:context=".Landing" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dip"
        android:gravity="center"
        android:text="@string/Selector"
        android:textColor="#000"
        android:textSize="35sp " />

    <ImageView
        android:id="@+id/icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="30dp"
        android:layout_marginBottom="30dp"
        android:contentDescription="@string/app_name"
        android:src="@drawable/ic_launcher" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/Level"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <Spinner
        android:id="@+id/spinner1"
        android:layout_width="match_parent"
        android:entries="@array/floors"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/Destination"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <Spinner
        android:id="@+id/spinner2"
        android:layout_width="match_parent"
```

```
        android:entries="@array/destinations"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/Navigate"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="lanzarVista"
        android:text="@string/Navigate" />

    </LinearLayout>
```

Obtenemos este aspecto:

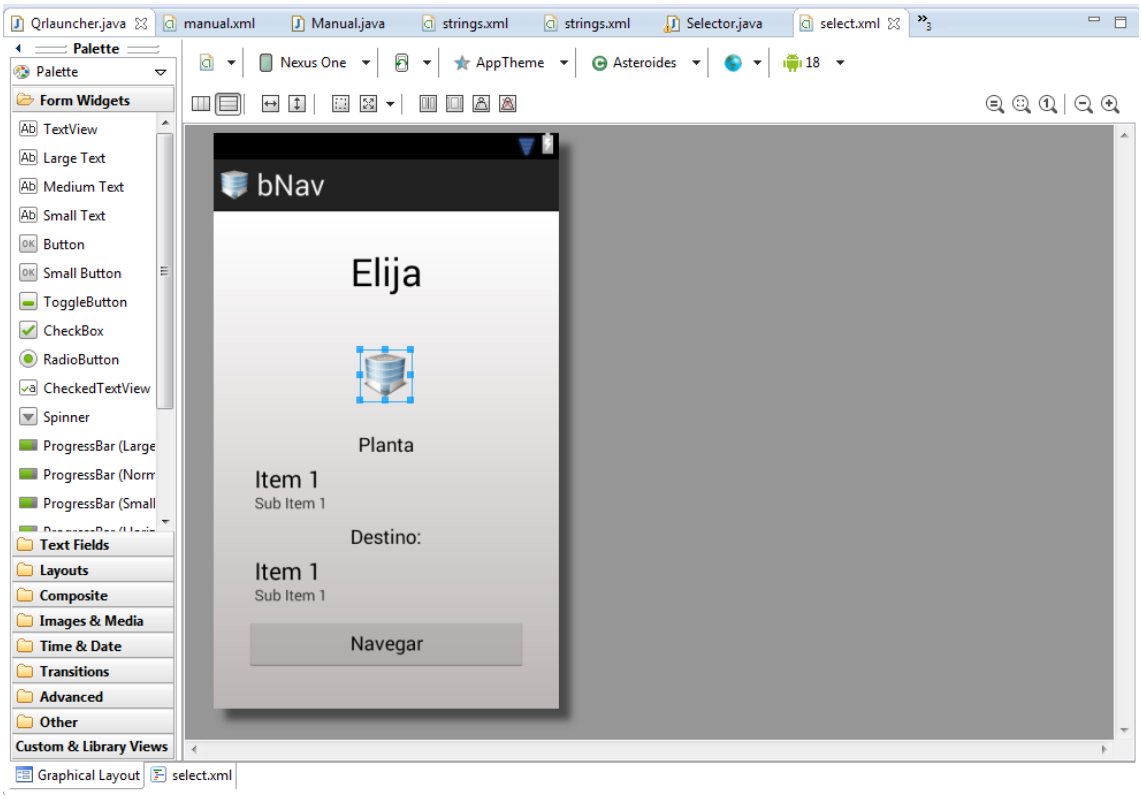


Ilustración 50 Selección del destino

El código de esta parte es el más complejo de la aplicación, ha de:

- Procesar los datos recibidos de las actividades para conectarse a la Wifi
- Conectarse al servicio y obtener el directorio.
- Mostrarlo en los selectores.
- Pasarnos a la vista final de navegación.

Para ello implemento el código siguiente y lo voy explicando.

```
package es.upv.buildingNavigator;

import java.net.MalformedURLException;
import java.net.URL;
```

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;
import es.upv.buildingNavigator.R;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.Intent;
import android.net.NetworkInfo;
import android.net.wifi.WifiConfiguration;
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.view.View;


public class Selector extends Activity {
```

En la creación de esta vista capturamos la información que se nos pasa para conectarnos a la red de invitados.

```
@Override
public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.select);
            Bundle extras = getIntent().getExtras();
            if (extras != null){
                  String value[] = extras.getStringArray("NET");
                  String activity = extras.getString("FROM");
                  processNet(value, activity);
            }
      }
```

Procesando los datos mediante una expresión regular en un caso y en otro a través de las variables directamente. Paso estos datos al método de conexión.

```
private void processNet(String value[], String activity) {
            String SSID= "",pass = "";
            if(activity=="QR"){
                  //String activity =
"<USER:TESTSSID><PASS:TESTPASS><LEVEL:LEVEL1><X:350><Y:400>";
                  Pattern userPattern = Pattern.compile("<USER\b[^>]*>");
                  Matcher userMatcher = userPattern.matcher(activity);
                  if(userMatcher.matches()) {
                        SSID =
userMatcher.group(1).substring(userMatcher.group(1).lastIndexOf(":"),
userMatcher.group(1).lastIndexOf(">"));
                  }
                  Pattern passPattern = Pattern.compile("<PASS\b[^>]*>");
                  Matcher passMatcher = passPattern.matcher(activity);
                  if(passMatcher.matches()) {
                        pass =
passMatcher.group(1).substring(passMatcher.group(1).lastIndexOf(":"),
passMatcher.group(1).lastIndexOf(">"));
                  }
```

```
            }
            if(activity=="Manual"){
                  SSID = value[0];
                  pass = value[1];
            }
            if(!checkConected(SSID)){
                  wifiAdd(SSID,pass);
                  if(checkConected(SSID)){
//                      try {
//                            getDirectory(SSID);
//                      } catch (MalformedURLException e) {
//                            // TODO Auto-generated catch block
//                            e.printStackTrace();
//                      }
                  }
            }
      }
```

Método para comprobar si está conectado a la red de invitados que toca para esta SSD

```
private boolean checkConected(String SSID) {
            WifiManager wifiManager = (WifiManager)
getSystemService(Context.WIFI_SERVICE);
            WifiInfo wifiInfo = wifiManager.getConnectionInfo();
            if (WifiInfo.getDetailedStateOf(wifiInfo.getSupplicantState())
== NetworkInfo.DetailedState.CONNECTED) {
                String ssid = wifiInfo.getSSID();
                if(ssid==SSID){
                  return true;
                }
            }
            return false;
      }
```

Este método ha sido por desgracia el bloqueante en esta Actividad, no he podido hacerlo funcionar correctamente, siempre produce excepción y no he encontrado una solución que funcione ahora mismo.

```
private void getDirectory(String SSID) throws MalformedURLException {
            try {

            URL url = new URL("http://www.sol-
radio.es/resources/stream.xml");
            DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            Document doc = db.parse(new InputSource(url.openStream()));
            doc.getDocumentElement().normalize();

            NodeList nodeList = doc.getElementsByTagName("stream");
            showDialog("Works");

        } catch (Exception e) {
            showDialog("XML Pasing Excpetion = " + e);
        }

      }
```

```
private void showDialog(String message) {
          AlertDialog.Builder builder = new AlertDialog.Builder(this);
          builder.setMessage(message);
          builder.setPositiveButton(R.string.Salir, null);
          builder.show();
      }
```

En este método añadimos la red wifi a las redes preferidas del sistema donde nos conectamos tras ese paso previo.

```
public void wifiAdd(String SSID,String pass){
      WifiManager wifiManager = (WifiManager)
getSystemService(Context.WIFI_SERVICE);
        // setup a wifi configuration
        WifiConfiguration wc = new WifiConfiguration();
        wc.SSID = "\""+SSID+"\"";
        wc.preSharedKey = "\""+pass+"\"";
        wc.status = WifiConfiguration.Status.ENABLED;
        wc.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.TKIP);
        wc.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.CCMP);
        wc.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.WPA_PSK);
        wc.allowedPairwiseCiphers.set(WifiConfiguration.PairwiseCipher.TKIP);
        wc.allowedPairwiseCiphers.set(WifiConfiguration.PairwiseCipher.CCMP);
        wc.allowedProtocols.set(WifiConfiguration.Protocol.RSN);

        // connect to and enable the connection
        int netId = wifiManager.addNetwork(wc);
        wifiManager.enableNetwork(netId, true);
        wifiManager.setWifiEnabled(true);
    }

public void lanzarVista(View view) {
             Intent i = new Intent(this, Vista.class);
             startActivity(i);
      }
}
```

Comprobamos que se ha establecido la conexión y entonces hemos de solicitar el directorio.

Evitando este paso por los problemas he mockeado el funcionamiento para que funcione directamente con unos datos fijos actualmente de forma que las opciones están prefijadas.

Tras completar todos estos pasos paso a la vista final del usuario para mostrar el mapa, el destino y su posición si es que ha usado un código QR.

La vista definida por el XML

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/degradado" >

    <TextView
        android:id="@+id/tvHeading"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="32dp"
        android:layout_marginTop="25dp"
        android:text="Heading: 0.0" />

    <ImageView
        android:id="@+id/map"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/tvHeading"
        android:layout_centerHorizontal="true"
        android:src="@drawable/map" />

    <ImageView
        android:id="@+id/imageViewCompass"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:layout_marginRight="14dp"
        android:adjustViewBounds="true"
        android:maxWidth="100dp"
        android:src="@drawable/img_compass" />

    <Button
        android:id="@+id/positioning"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="22dp"
        android:onClick="Position"
        android:text="@string/Position" />

</RelativeLayout>
```

Y su código Java

```
package es.upv.buildingNavigator;

import es.upv.buildingNavigator.R;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.Intent;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
```

```
import android.hardware.SensorManager;
import android.os.Bundle;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.RotateAnimation;
import android.widget.ImageView;
import android.widget.TextView;

public class Vista extends Activity implements SensorEventListener {

    // define the display assembly compass picture
    private ImageView image, dinamic;
    // record the compass picture angle turned
    private float currentDegree = 0f;
    // device sensor manager
    private SensorManager mSensorManager;

    TextView tvHeading;
    IntentIntegrator integrator = new IntentIntegrator(this);

    @Override
  protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.vista);

        // Compass
        image = (ImageView) findViewById(R.id.imageViewCompass);
        // TextView that will tell the user what degree is he heading
        tvHeading = (TextView) findViewById(R.id.tvHeading);
        // initialize your android device sensor capabilities
        mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

        /*
        dinamic.setImageBitmap(bmp);
         showDialog("Success");
         */
        //showDialog((new Image().execute("Test")).toString());


    }

@Override
protected void onResume() {
        super.onResume();
        // for the system's orientation sensor registered listeners
        mSensorManager.registerListener(this,
mSensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION),
                SensorManager.SENSOR_DELAY_GAME);
    }

    @Override
protected void onPause() {
        super.onPause();
        // to stop the listener and save battery
        mSensorManager.unregisterListener(this);
    }
```

Método para el uso de la brújula en la vista.

```
@Override
public void onSensorChanged(SensorEvent event) {
        // get the angle around the z-axis rotated
        float degree = Math.round(event.values[0]);
        tvHeading.setText("Heading: " + Float.toString(degree) + " degrees");

        // create a rotation animation (reverse turn degree degrees)
        RotateAnimation ra = new RotateAnimation(
                currentDegree,
                -degree,
                Animation.RELATIVE_TO_SELF, 0.5f,
                Animation.RELATIVE_TO_SELF,
                0.5f);

        // how long the animation will take place
        ra.setDuration(210);
        // set the animation after the end of the reservation status
        ra.setFillAfter(true);
        // Start the animation
        image.startAnimation(ra);
        currentDegree = -degree;
    }

@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
        // not in use
    }
```

Código para obtener nuestra posición en un momento determinado.

```
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent
intent) {
    IntentResult result = IntentIntegrator.parseActivityResult(requestCode,
resultCode, intent);
           if (result != null) {
             String contents = result.getContents();
             if (contents != null) {
               showDialog(result.toString());
             } else {
              showDialog("Error");
             }
           }
    }

      private void showDialog(String message) {
         AlertDialog.Builder builder = new AlertDialog.Builder(this);
         builder.setMessage(message);
         builder.setPositiveButton(R.string.Salir, null);
         builder.show();
      }

      public void Position(View view) {
            integrator.initiateScan();
      }
```

```
private String processPosition( String activity) {
            String x= "",y = "";
            //String activity =
"<USER:TESTSSID><PASS:TESTPASS><LEVEL:LEVEL1><X:350><Y:400>";
            Pattern xPosition = Pattern.compile("<X\b[^>]*>");
            Matcher xMatcher = xPosition.matcher(activity);
            if(xMatcher.matches()) {
                  x =
xMatcher.group(1).substring(xMatcher.group(1).lastIndexOf(":"),
xMatcher.group(1).lastIndexOf(">"));
            }
            Pattern yPosition = Pattern.compile("<Y\b[^>]*>");
            Matcher yMatcher = yPosition.matcher(activity);
            if(yMatcher.matches()) {
                  y =
yMatcher.group(1).substring(yMatcher.group(1).lastIndexOf(":"),
yMatcher.group(1).lastIndexOf(">"));
            }
            return x+':'+y;
      }
}
```

Finalmente obtenemos la vista de navegación.

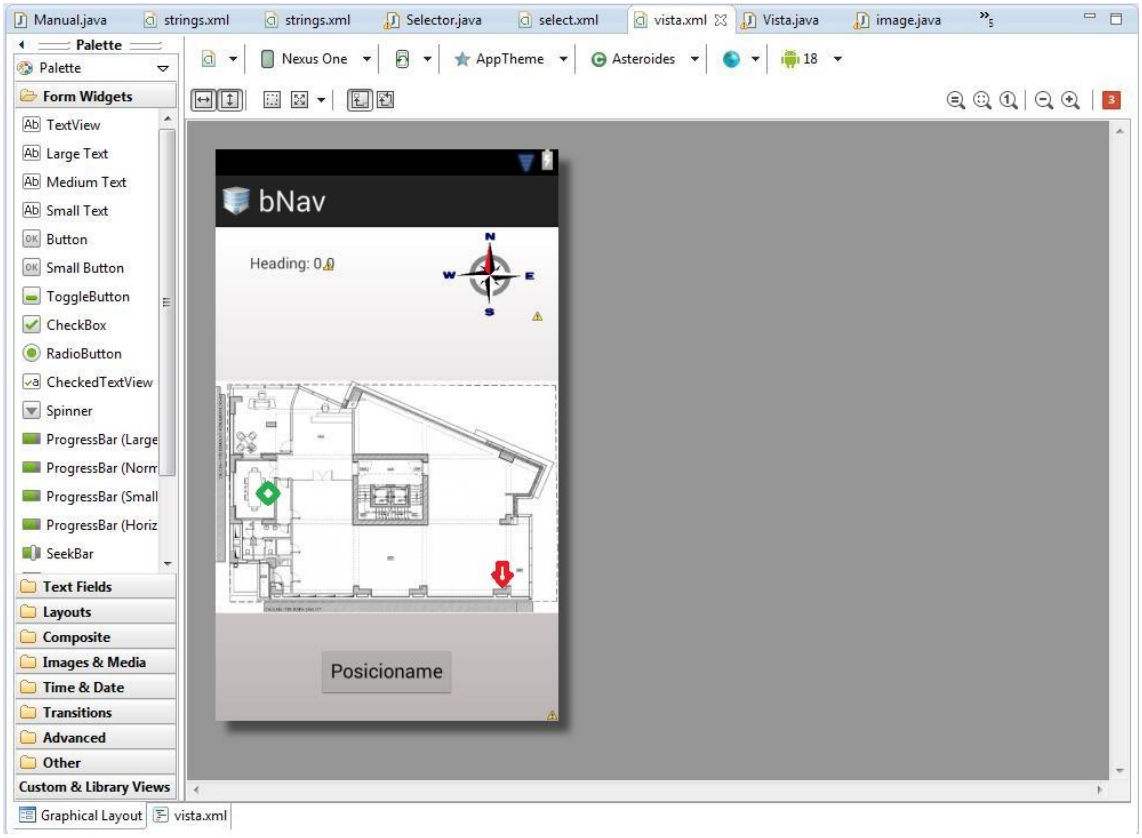


Ilustración 51 Vista final

Con esta vista y ayudándonos reposicionándonos en caso de perdernos el usuario debe ser capaz de llegar a su destino dentro del edificio.

5.3. Problemas y soluciones aplicadas

Durante el desarrollo he tropezado con múltiples problemas, algunos de ellos sin resolver por falta de recursos.

Enumeración de los mismos:

- Problema con la devolución de archivos.
- Montando Android Studio
- Rendimiento del Emulador Android
- Obtención de recursos desde internet en el App

Problema con la devolución de archivos.

Este problema ha quedado pendiente de solución, durante el desarrollo a la hora de devolver archivos xml y jpg directamente desde el servicio y luego desde la aplicación intenté solucionarlo mediante intentar consumir directamente a través de URL los recursos pero se nos ha planteado el problema de que no se permite bloquear la actividad principal con ejecuciones que puedan tardar.

Se ha intentado solucionar por medio de una llamada Asíncrona a otra clase para no bloquearlo pero no he conseguido solucionarlo.

Montando Android Studio

El nuevo entorno de desarrollo muestra potencial pero las limitaciones de mi equipo de trabajo junto a la falta de documentación tan abundante me han hecho decantarme por Eclipse como IDE, en próximos desarrollos a nivel profesional y personal pienso volver a intentar manejarme con Android Studio.

Rendimiento del Emulador Android

Este problema ha estado relacionado con el punto anterior, por falta de rendimiento no puedo emular los últimos dispositivos en el emulador de Android. Además el emulador tiene ciertas limitaciones a la hora de probar sensores e interactuar con otras aplicaciones como es el caso de mi aplicación.

Queda en manos de una supuesta evolución de este proyecto resolver estos problemas que no se han podido solventar por falta de tiempo.

6. Evaluación de los resultados

Los resultados del proyecto son en mi opinión positivos y muy beneficiosos, a pesar de no haber completado la solución se ha llegado a obtener conocimientos, experiencia y una base a desarrollar para un próximo proyecto.

Formación

En el tema de **Formación** he conseguido una cantidad considerable de conocimiento sobre cómo funciona un desarrollo Android

- Estructura del proyecto.
- Funcionamiento de las vistas.
- Recursos.
- Refresco sobre Java

Y con respecto a la parte del Servicio he refrescado tecnologías como PHP.

Tras este desarrollo una persona que retome el proyecto debería ser capaz de completar y poner en funcionamiento la solución en aproximadamente un 50% del tiempo gastado en este Proyecto de Fin de Carrera.

Tiempo empleado en el desarrollo

La distribución de tiempo de este proyecto ha sido aproximadamente:

- Memoria. **40 %**
- bNavService. **20 %**
- bNav Aplicación. **40 %**

7. Conclusiones y Trabajos Futuros

En conclusión se ha arrancado el proyecto pero queda mucho desarrollo por hacer muchas mejoras posibles enumero algunas de ellas a continuación.

Sobre la aplicación.

- Adaptar a Tablet
- Completar funcionalidad.
- Mejoras del aspecto.
- Multi planta.

Sobre el Servicio.

- Completar funcionalidad.
- Mejoras de Aspecto.
- Multi planta.

Sigue sin haber una solución para posicionamiento dentro de edificios y probablemente la solución ganadora del mercado trabajará con varios tecnologías y seguramente Wifi u otra tecnología electro magnética ya que permiten posicionarnos mejor que el resto, esto junto a brújula y otras tecnologías auxiliares nos proporcionará una buena solución.

8. Bibliografía

- **The Global Positioning System** (http://en.wikipedia.org/wiki/Global_Positioning_System).
- **Indoor Positioning System** (http://en.wikipedia.org/wiki/Indoor_positioning_system).
- **Google Nexus 4** (http://www.google.es/nexus/4/)
- **Directions Magazine** (http://www.directionsmag.com/articles/10-things-you-need-to-know-about-indoor-positioning/324602)
- **Stackoverflow Foro de programación** (http://stackoverflow.com)
- **W3ORG**(http://www.w3.org/html/logo/)
- **PHP.NET** (http://php.net/)
- **JS**(https://developer.mozilla.org/en-US/docs/Web/JavaScript)
- **CSS3**(https://developer.mozilla.org/en-US/docs/Web/CSS/CSS3?redirectlocale=en-US&redirectslug=CSS%2FCSS3)
- **Extensible Markup Languaje** (http://www.w3.org/XML/)
- **XAMPP** (http://www.apachefriends.org/es/xampp.html).
- **Sublime text2** (http://www.sublimetext.com/2)
- **HTML5 Login Form** (http://www.hongkiat.com/blog/html5-loginpage/)
- **Developers Google** (http://developer.android.com/develop/index.html)
- **31 Days Android** (http://chrisrisner.com/31-Days-of-Android)
- **Zebra Crossing** (https://code.google.com/p/zxing/)
- **Generador Codigos QR** (http://www.codigos-qr.com/generador-de-codigos-qr/)