ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERÍA INFORMÁTICA (2º CICLO)

Título del proyecto:

APLICACIÓN ANDROID BUSCADOR DE CENTROS EDUCATIVOS DE NAVARRA

Autor: Mikel San Martín Huarte

Tutor: Óscar Ardaiz Villanueva

Pamplona, a 10 de noviembre de 2014

ÍNDICE

Contenido

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

1 – Introducción

1.1 - Problema a resolver y motivaciones

El Gobierno de Navarra, sacó en el boletín oficial nº 96[1], correspondiente al 20 de Mayo de 2014, la convocatoria para el I Premio de Diseño de Aplicaciones para Teléfonos Inteligentes (SmartPhones) con datos abiertos del Departamento de Educación.

El objetivo de este concurso era hacer una aplicación para el sistema operativo Android para facilitar la búsqueda y ofrecer información acerca de todos los centros educativos que tenemos en la Comunidad Foral de Navarra. Más en concreto, ofrecer la información que el Departamento de Educación del Gobierno de Navarra ha ofrecido últimamente en su Directorio de Centros Educativos web[2] en una aplicación móvil compatible para el máximo número posible de versiones Android.

El Gobierno de Navarra ofrece estos datos sobre los centros educativos de la comunidad a través del Portal Open Data Navarra, portal de internet para hacer accesibles los datos para la redistribución, reutilización y aprovechamiento por parte de terceros.

Otros de los requisitos para el proyecto eran que la aplicación, al menos, debía estar en bilingüe tanto en castellano como en euskera y no haber sido difundidos con anterioridad.

Por tanto, el objetivo era hacer una aplicación Android amigable para el ciudadano, con el propósito de acercarle la información sobre los centros educativos navarros de una manera atractiva y sencilla con la aplicación móvil bilingüe.

[1] http://www.navarra.es/home_es/Actualidad/BON/Boletines/2014/96/Anuncio-2/
[2] http://www.educacion.navarra.es/web/dpto/centros-educativos

Los criterios de valoración de los proyectos serían los siguientes:

a) Funcionalidad otorgada: hasta un máximo de 50 puntos.

b) Usabilidad y amigabilidad de la interfaz de usuario: hasta un máximo de 25 puntos.

c) Diseño gráfico: hasta un máximo de 15 puntos.

d) Tecnología y rendimiento: hasta un máximo de 10 puntos.

Para poder optar al premio, había que obtener al menos el 33% de los puntos repartidos en cada uno de los cuatro apartados anteriores.

Los premios ofrecidos eran como primer premio un ordenador portátil valorado en 1.500€ o su equivalente en metálico, y como accésit una Tablet valorada en 600€ o su equivalente en metálico, y las aplicaciones debían ser presentadas con fecha límite 1 de septiembre de 2014.

Esta aplicación que yo he desarrollado con el nombre “Buscador de Centros Educativos de Navarra” y que detallaré en este documento se ha proclamado ganadora del primer premio del concurso.

1.2 – Estado del arte

Como antecedentes de aplicaciones con las que tomar referencia para la realización de este proyecto se han tomado principalmente los proyectos Open Data Navarra 2011[3], para ver la estructura y consecución de un tipo de proyecto parecido al que este concurso se trataba, y por otro lado y más importante, el Directorio de Centros Educativos web para ver todas las opciones y funcionalidades básicas que habían de implementarse en la aplicación móvil.

Por ello, se ha tomado como referencia el Directorio de Centros Educativos web, que tiene la siguiente estructura:

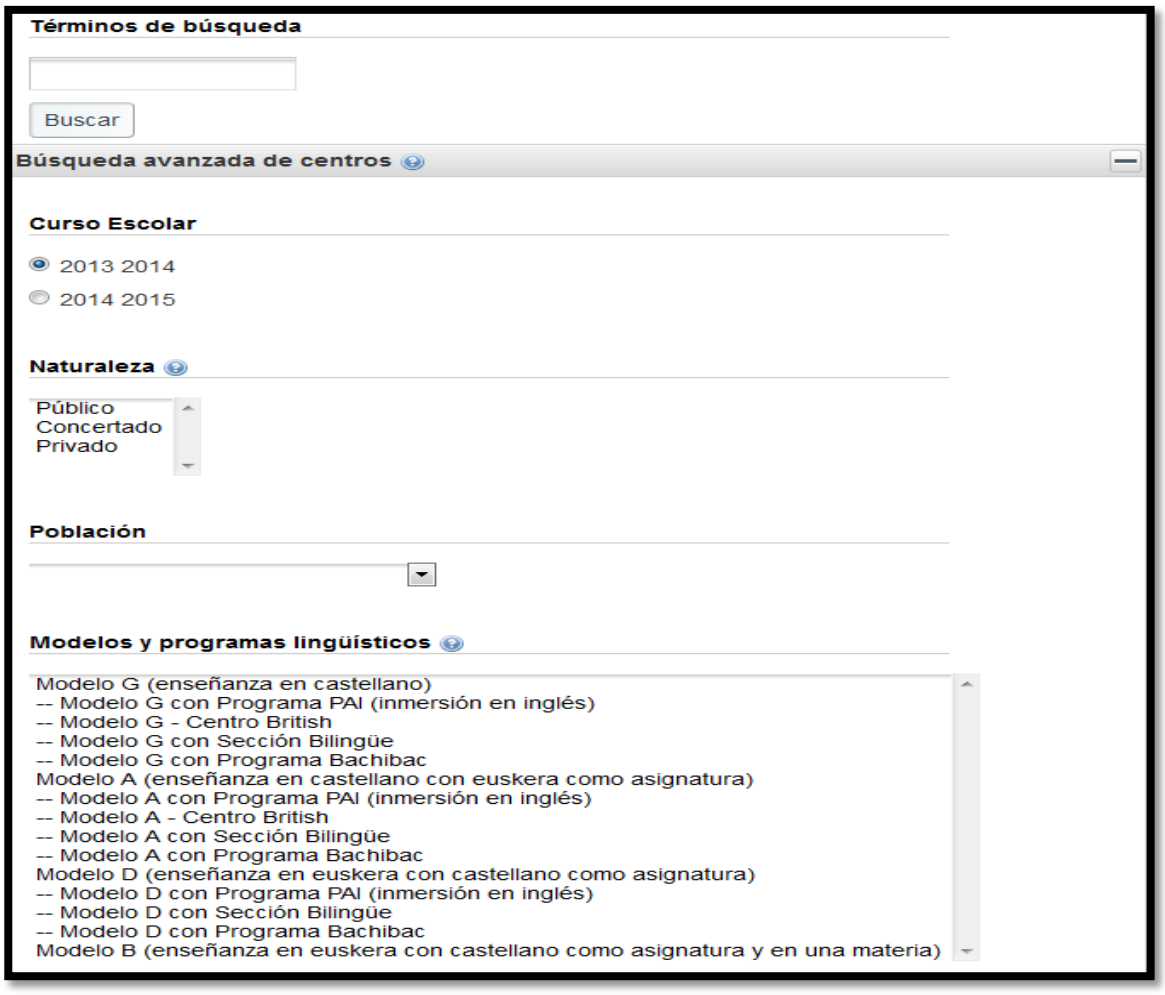


Figura 1: Buscador Directorio de Centros Educativos web.

3 http://www.navarra.es/home_es/Congresos/PremiosOpenData/2ProyectosPresentados.htm

Como se puede observar, en el Directorio de Centros Educativos web, podemos hacer búsquedas de centros educativos especificando diferentes parámetros, como en su caso son unas palabras de búsqueda, el curso escolar, la naturaleza del centro, la población o los modelos y programas lingüísticos.

Al realizar una de estas búsquedas, se logrará un listado con los centros que cumplan las características especificadas.

Nombre	Localidad	Sitio web	
Esc. Mús. Púb. Doneztebe AGORRETA	Doneztebe/Santesteban		Más datos
CPEIP San Adrian	San Adrián		Más datos
IES Corella Alhama	Corella		Más datos
Esc. Mús. Púb. Altsasu	Altsasu/Alsasua		Más datos
Esc. Ed. Inf. Lerín Amado Alonso	Lerín		Más datos
Esc. Ed. Inf. Villava Amalur	Villava		Más datos
Esc. Ed. Inf. Garínoain Amatxi	Garínoain		Más datos
C.Con. Bur. Amor de Dios	Burlada		Más datos
C.Con. Bur. Amor - Regina	Burlada		Más datos
IK Etxarri-Ar. Andra Mari	Etxarri Aranatz		Más datos

Figura 2: Resultado búsqueda.

Si pulsamos sobre el botón más datos, se mostrará información detallada sobre el centro en cuestión, dividiéndose la información en datos básicos, plan de estudios y localización.

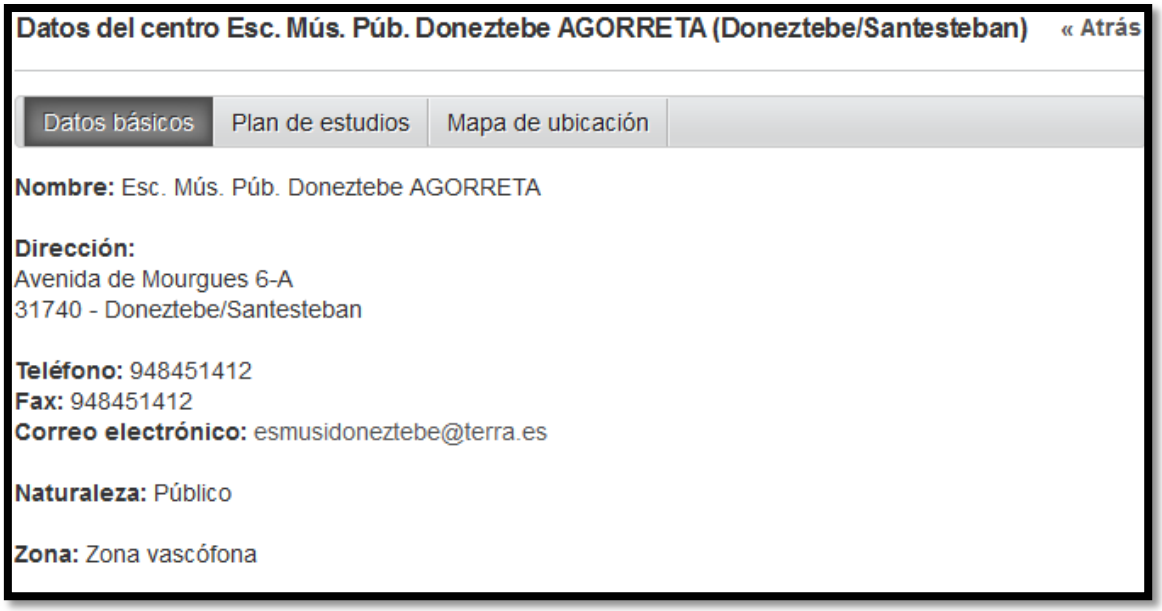


Figura 3: Información detallada del centro.

Tras analizar los pormenores de este servicio, nos disponemos a diferenciar los objetivos que deberá cumplir nuestra aplicación.

1.3 – Objetivos

Por tanto, los objetivos a cumplir en el desarrollo de la aplicación serán los siguientes:

- Elaborar una aplicación amigable y fácil de usar para acercar la información de los centros educativos navarros a todos los ciudadanos de la comunidad.
- Bilingüidad, castellano y euskera.
- Hacerla compatible al máximo posible de versiones de Android.
- La aplicación debe permitir hacer una búsqueda al ciudadano entre todos los centros ya sea por su nombre o por la localidad en la que se encuentra.
- El usuario tendrá una opción de búsqueda avanzada en la que podrá especificar el curso educativo, la naturaleza del centro, el modelo lingüístico, servicios complementarios o nivel educativo.
- Habrá una tercera opción para especificar el camino desde la ubicación del usuario hasta el centro educativo especificado.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

- Tras hacer una búsqueda, se podrá seleccionar un centro para lograr información adicional acerca de él. Esta información estará dividida en tres grupos: información básica, plan de estudios y otros datos y localización.

1.4 – Solución propuesta

La aplicación Android Buscador de Centros Educativos de Navarra – Nafarroako Ikastetxeen Bilatzailea se ha desarrollado con el conjunto de datos OpenData que el Gobierno de Navarra proporciona, y su objetivo es hacer accesible de una forma interactiva, amigable y atractiva al ciudadano navarro a los datos de los centros educativos de nuestra comunidad foral.

Los objetivos principales de la aplicación son que el ciudadano pueda buscar fácilmente información sobre cualquiera de los centros educativos de nuestra comunidad, ya sea buscando su nombre o localidad, o especificando unos parámetros como el modelo lingüístico, servicios adicionales, nivel de enseñanza… así como también poderle ofrecer cómo llegar a cualquiera de estos centros educativos ofreciéndole la mejor ruta tanto en vehículo como andando.

La aplicación es compatible con los sistemas Android entre el SDK 7 (Android 2.1) y el SDK 19 (Android 4.4), que a fecha de hoy es la última versión disponible de Android. La aplicación ha sido testeada en un teléfono móvil Sony Xperia U con la versión de Android 2.3.7 y una tablet BQ Edison 2 con la versión de Android 4.2.2 y funciona a la perfección en ambos dispositivos.

La aplicación es completamente bilingüe en castellano y euskera. Además si tuviera que adaptarse a más idiomas este paso podría hacerse con muy poco esfuerzo ya que el código está adaptado al multilingüismo (carpetas values values-eu, layout… en Android). La aplicación se arrancará en el idioma en el que tengamos configurado nuestro terminal Android (castellano o euskera) y si lo deseamos como veremos más adelante podremos cambiar en cualquier momento la aplicación desde un idioma al otro desde la pantalla principal de la aplicación.

Para recibir la información acerca de todos los centros, los datos abiertos son recibidos al terminal Android haciendo un intercambio de información con el Web Service que pone en servicio el Gobierno de Navarra[4]. Para este intercambio de información se utiliza el protocolo SOAP.

[4] https://educages.navarra.es/xwiki/bin/view/Main/Directorio+de+centros

En la siguiente imagen se esquematiza el funcionamiento de la aplicación

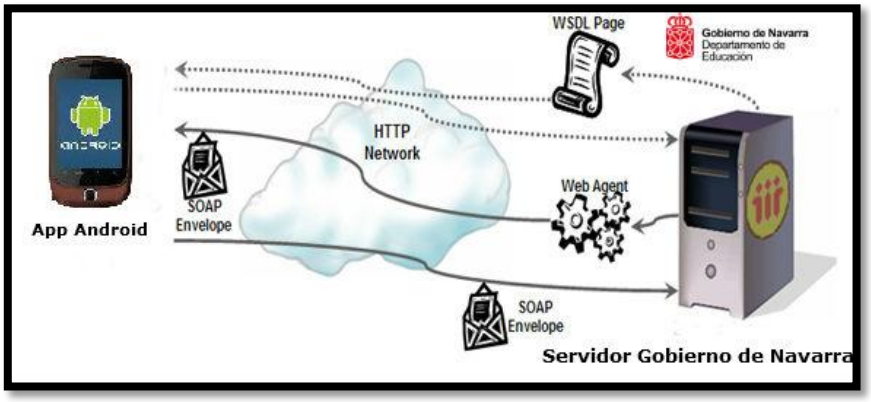


Figura 4: Esquema del funcionamiento de la aplicación

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

2 – Desarrollo del trabajo

En esta sección haré el análisis el desarrollo del trabajo para la creación de la aplicación, detallando como se ha llevado a cabo cada iteración de la metodología ágil SCRUM.

2.1 – Metodología de trabajo

Para el desarrollo de la aplicación se ha seguido el modelo de la metodología ágil SCRUM. De esta manera se haría una primera iteración con las funcionalidades más básicas de la aplicación, para seguidamente añadir otras características en una segunda iteración.

En la primera etapa se hizo el diseño de la aplicación y un primer prototipo con las funcionalidades más básicas.

En la segunda iteración se añadieron más opciones a la aplicación como son el buscador avanzado y la ruta hacia el centro educativo.

Finalmente hubo que añadir una tercera iteración para adaptar la aplicación a unas mejoras sugeridas por el personal del Gobierno de Navarra.

Se ha llevado un control de las versiones de la aplicación móvil en el sistema de almacenamiento en la nube gratuito de Google Drive, donde se subía una versión cada vez que había un progreso importante en la aplicación.

Como se puede observar en las figuras 5 y 6, el desarrollo de la aplicación en su primera versión empezó el día 25 de Junio, obteniendo la versión para la entrega del concurso el 28 de Agosto, y finalmente la versión con mejoras sugeridas por el Gobierno de Navarra el 1 de Octubre, todas las fechas del año 2014 .

Figura 5: Control de cambios 1

Figura 6: Control de cambios 2

Para cada una de las tres iteraciones, se ha desarrollado un análisis y diseño, implementación y plan de pruebas. Tras las dos primeras iteraciones se hizo una documentación inicial con el funcionamiento de la aplicación, actualizándola y completándola al finalizar la tercera iteración.

2.2 – Análisis de requisitos

El objeto del proyecto es la creación de una aplicación Android utilizando los datos de educación que ofrece el Gobierno de Navarra. Con la aplicación lograremos el objetivo de mostrar los datos de los centros de educación que el mencionado portal ofrece de una forma atractiva para el usuario.

Este proyecto se basa en la integración del sistema WSDL que ofrece el Gobierno de Navarra con datos abiertos sobre el departamento de educación con una

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

aplicación móvil con tecnología Android donde se presentarán los datos que le son de utilidad al usuario.

2.2.1 – Requisitos funcionales

- La aplicación mostrará los datos de los centros que ofrece el servidor mediante conexión SOAP con el servicio WSDL proporcionado para la obtención de información, de manera dinámica.
- Las tres opciones más importantes en la pantalla principal serán mostrar todos los centros, hacer una búsqueda avanzada o lograr el camino hacia un centro educativo.
- Cuando se ofrezca un listado de centros, se podrá hacer una búsqueda por nombre del centro o por localidad.
- Los centros buscados podrán verse en modo lista o en modo mapa.
- Los parámetros para hacer una búsqueda avanzada serán curso, naturaleza, modelos y programas lingüísticos, servicios complementarios y nivel educativo.
- La opción ruta hacia el centro educativo mostrará la mejor ruta en coche y la mejor ruta caminando.
- Cuando pulsemos sobre un centro educativo en el listado o en el modo mapa veremos información adicional sobre él, dividida en tres pestañas: información básica, plan de estudios y otros datos y mapa de localización.
- En la pestaña información básica veremos nombre, dirección, código postal, localidad, teléfono, fax, e-mail, web y naturaleza del centro.
- En la pestaña plan de estudios y otros datos se mostrará información referente a modelos lingüísticos, tipo de enseñanza, servicios complementarios, atención nee, programas, plazas 3 años, plan de estudios, enlace a Sitna, número de matrículas, objetivos educativos, valores del centro, distinciones, jornada laboral, proyectos, reconocimientos y zona lingüística a la que pertenece.
- En la pestaña mapa de localización veremos el interfaz de Google Maps con el punto marcado del centro educativo en cuestión.
- Los datos referentes a los centros como enlaces, teléfono… sean clickables.
- La aplicación debe ser totalmente intuitiva y sencilla para adaptarse a usuarios de todo tipo.

2.2.2 – Requisitos no funcionales

- La aplicación es compatible con los sistemas Android entre el SDK 7 (Android 2.1) y el SDK 19 (Android 4.4).
- La aplicación se adapta a dispositivos de diferentes tamaños.
- Para el intercambio de datos entre el dispositivo móvil y el servidor se usa una conexión segura y mediante el protocolo SOAP para interactuar con el WSDL ofrecido por el Gobierno de Navarra.
- Para el uso de SOAP en Android, se usará la librería externa ksoap2. La información XML recibida se parseará utilizando la clase XMLPullParser de Android.
- La aplicación tendrá que ser estable y manejar que no se den errores en su uso.
- Se usan tareas asíncronas de Android para un funcionamiento más rápido y fluido de la aplicación.
- Las clases están organizadas según su función por medio de paquetes en la estructura del proyecto.
- Para mostrar la localización de los centros educativos como las rutas se usará el API V2 de Google Maps.
- La aplicación deberá ser lo más ligera posible para ocupar el mínimo espacio en la memoria del teléfono.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

2.3 – Diagrama de casos de uso

La Figura 7 muestra el diagrama de casos de uso general de la aplicación.

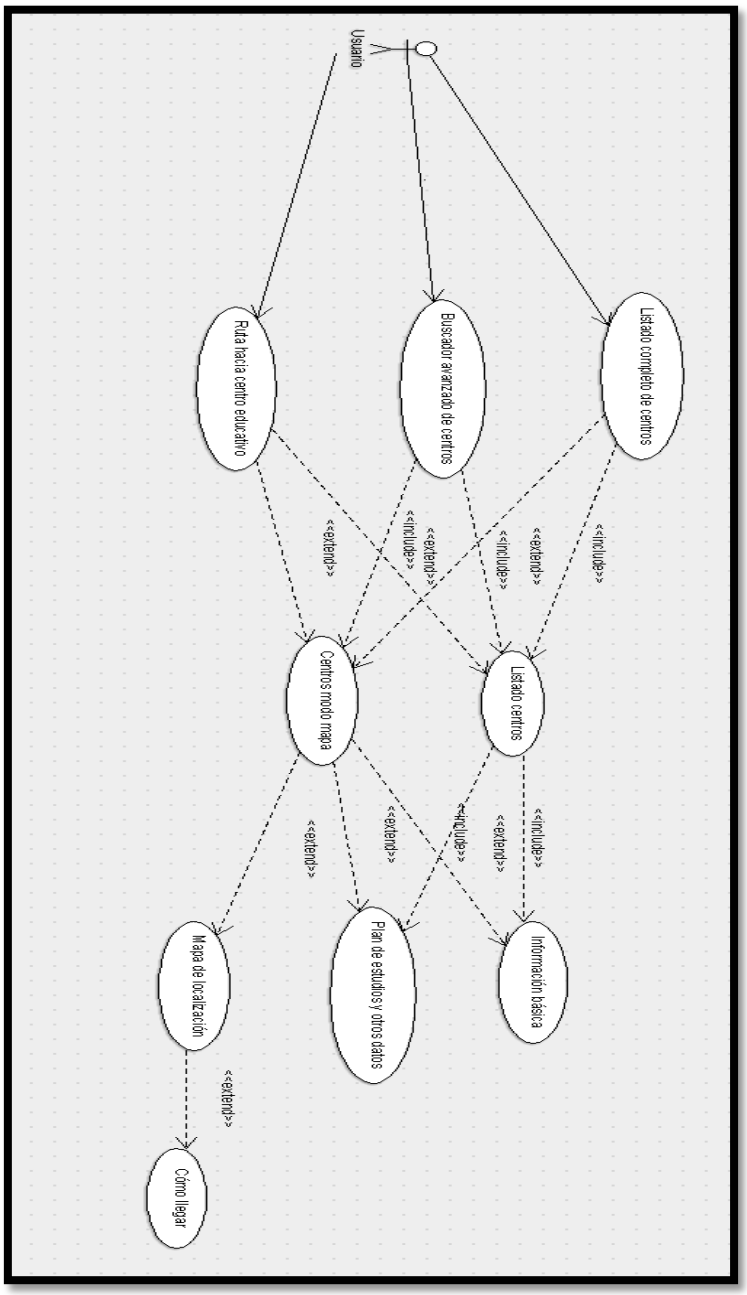


Figura 7: Diagrama de casos de uso

En él podemos observar las acciones que puede realizar un usuario de la aplicación, que como ya hemos apuntado serán las tres principales: ver el listado completo de centros, ir al buscador avanzado de centros o a ver la ruta hacia el centro educativo. Desde cada uno de ellos veremos los centros solicitados en modo lista,

opcionalmente en modo mapa, y desde ellos podremos ver su información básica, plan de estudios y otros datos o mapa de localización. Además, desde dicho mapa podremos ver cómo llegar hasta esa localización si lo queremos.

2.4 – Diagrama de clases

En la figura 8 podemos ver el diagrama de clases general de la aplicación.

En esta parte del diagrama de clases podemos ver la interacción de las clases del proyecto desde la actividad que es la pantalla principal hacia las 3 opciones básicas que tenemos, así como las clases auxiliares que usa cada una de las opciones en cuestión. El flujo de información desde CentrosActivity hacia el resto de actividades y clases, se ha hecho en la segunda parte del diagrama, para que los dibujos queden de una manera más clara e inteligible, pero nótese que el diagrama es como si las dos partes estuvieran juntas, en un solo dibujo.

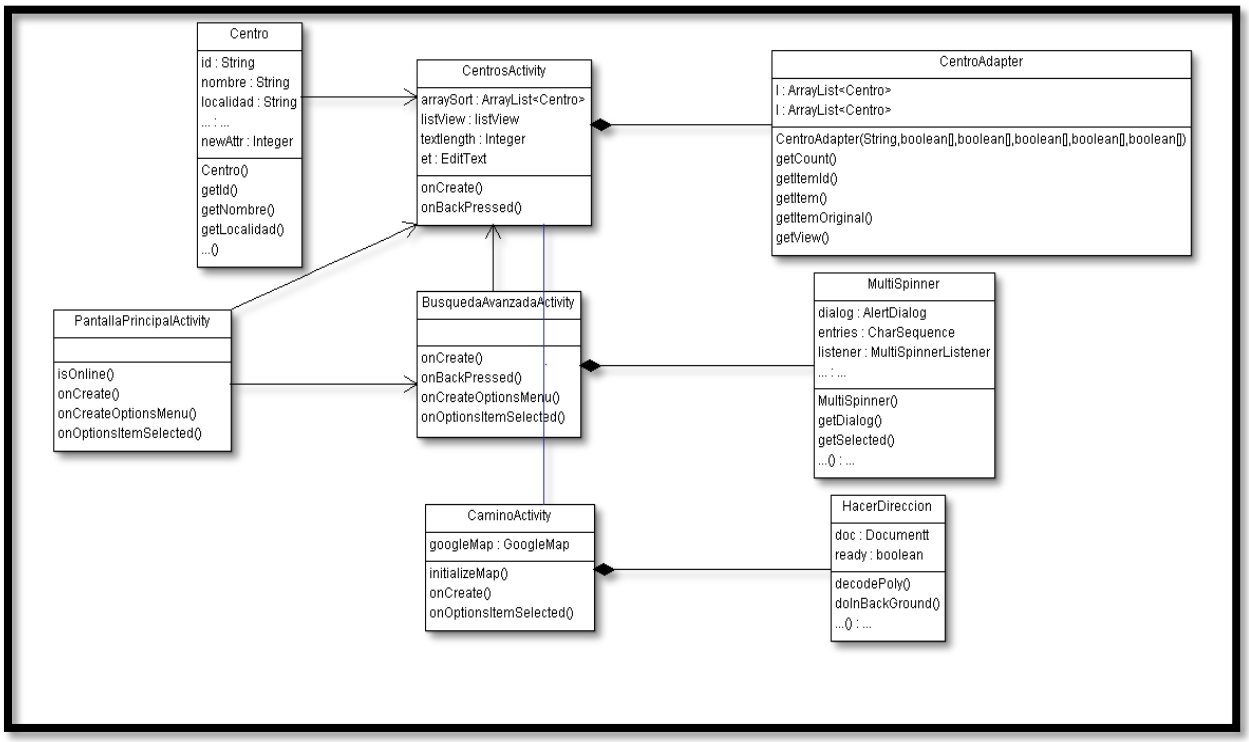


Figura 8: Diagrama de clases primera parte

Por tanto, en las siguiente figura 9 vemos la segunda parte del diagrama de clases.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

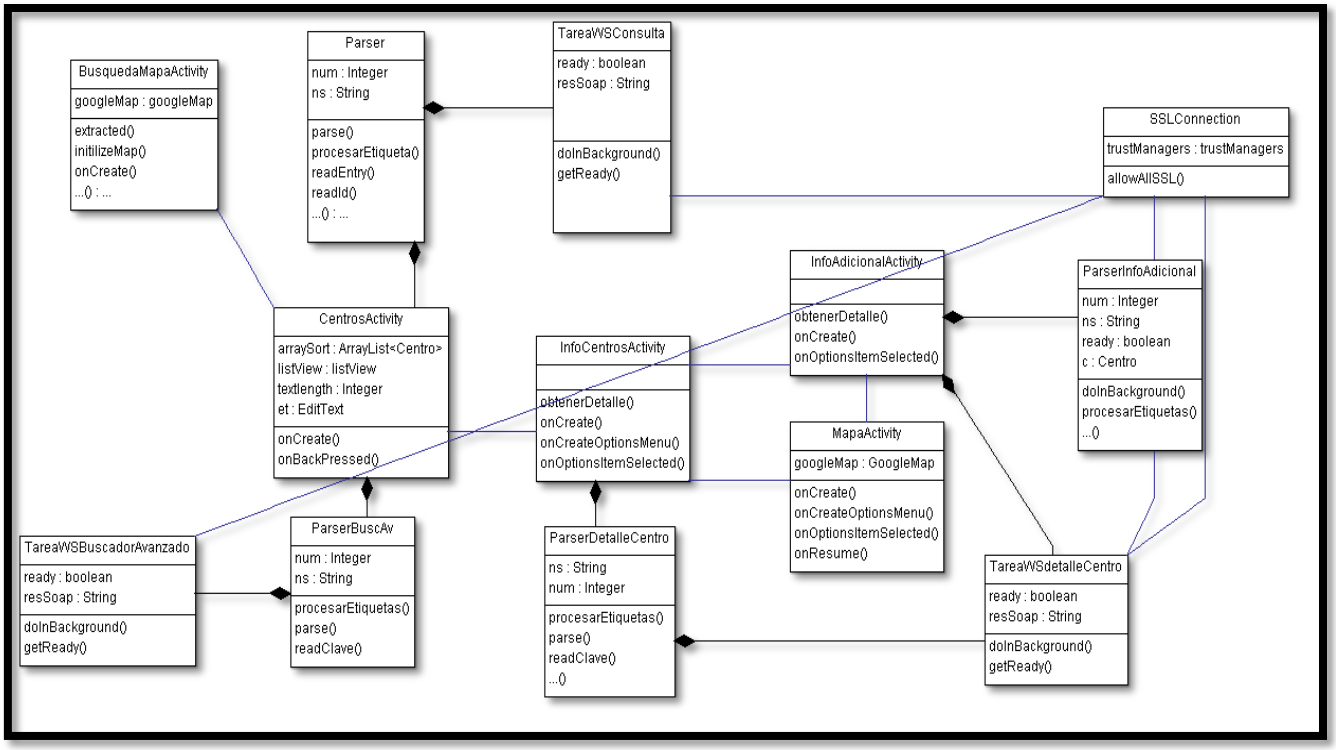


Figura 9: Diagrama de clases segunda parte

En esta segunda parte del diagrama vemos la interacción entre CentrosActivity y las actividades que ofrecen los detalles de los centros, su información y su localización, que a su vez cada una usa su Tarea para hacer la petición al WSDL y un parser para tratar la información recibida.

2.5 – Diagramas de secuencia

Estos son los diagramas de secuencia que muestran la interacción del conjunto de objetos de la aplicación a través del tiempo.

2.5.1 – Ver listado completo de centros

En la figura 10 vemos el diagrama de secuencia cuando pulsamos el botón de ver el listado completo de centros.

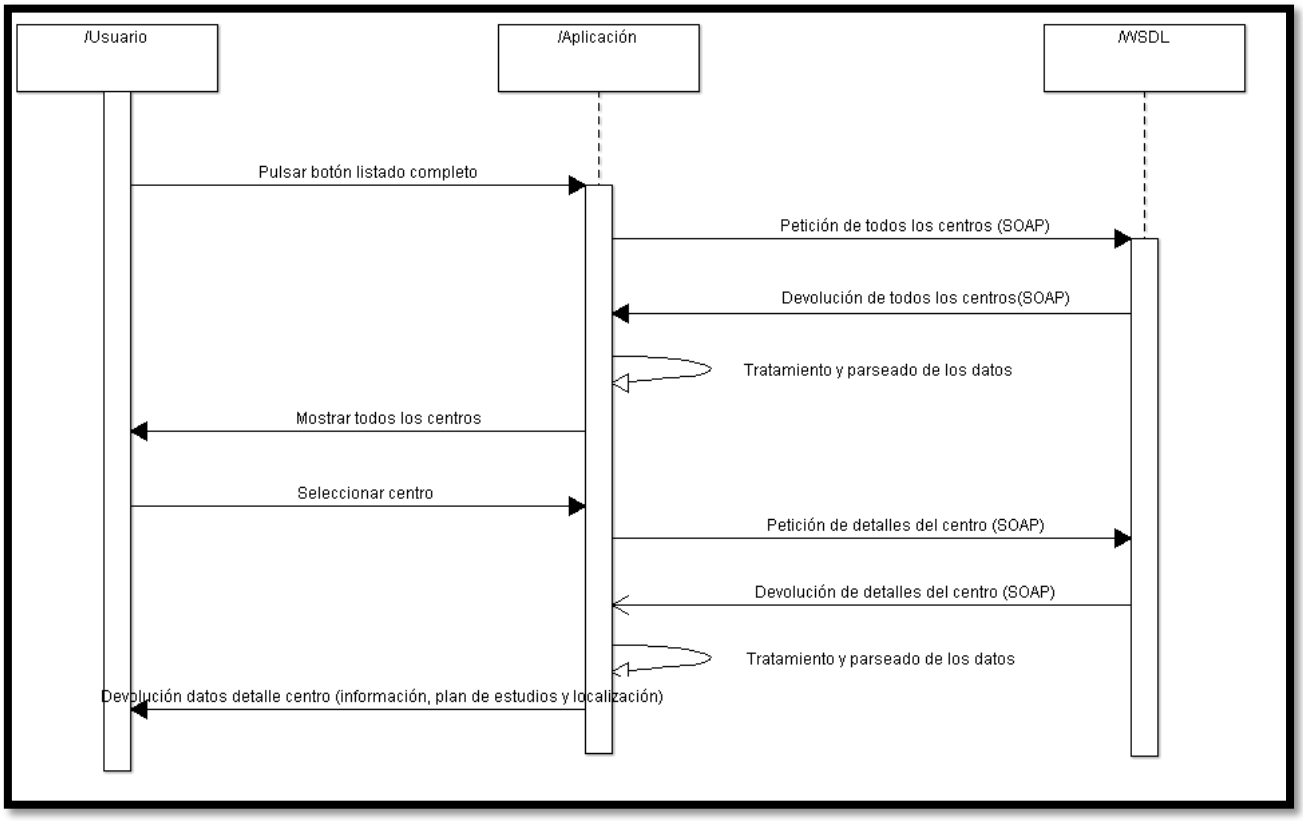


Figura 10: Diagrama secuencia ver listado completo de centros

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

2.5.2 – Búsqueda avanzada de centros

En la figura 11 vemos el diagrama de secuencia que es el que se produce cuando hacemos una búsqueda avanzada especificando características concretas de los centros (curso, nivel educativo, modelo lingüístico, servicios complementarios y naturaleza), para ver solamente los centros que cumplen las características especificadas.

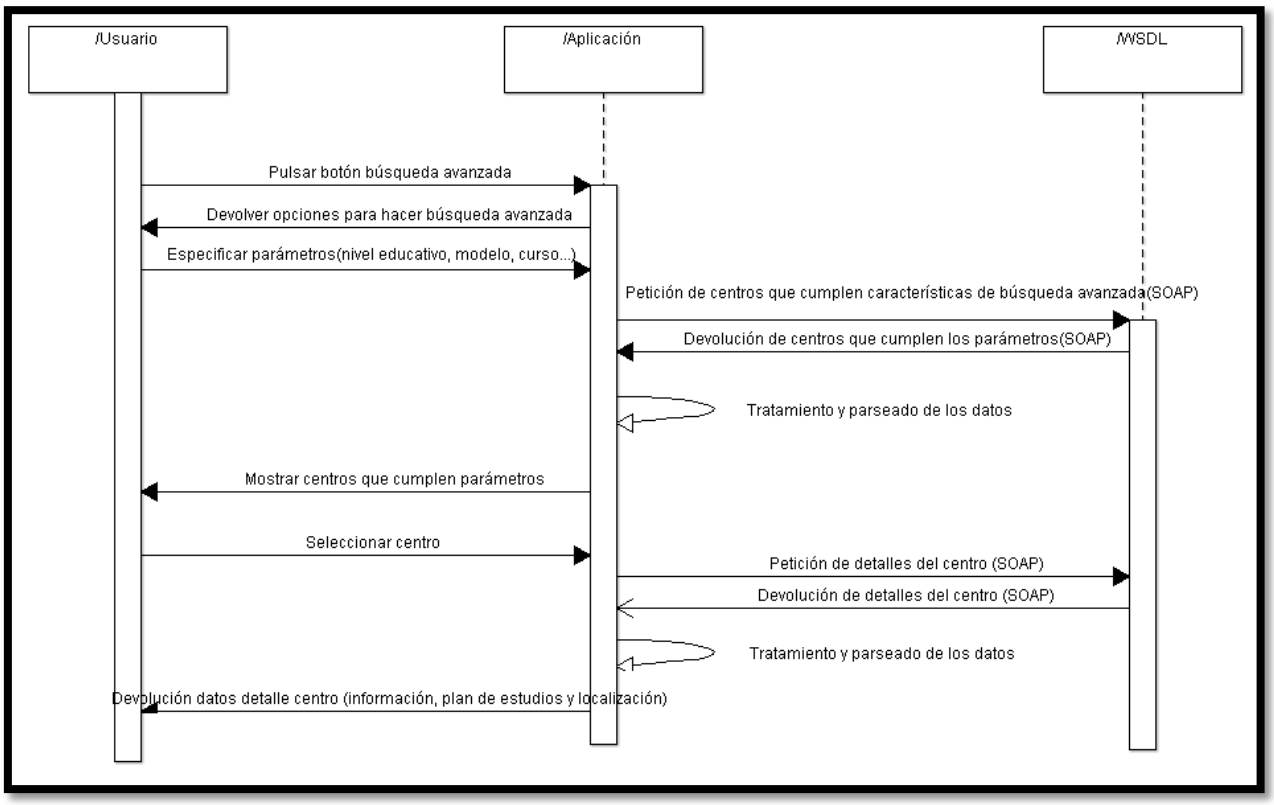


Figura 11: Diagrama de secuencia lista de la compra online

2.5.3 –Ruta hacia el centro educativo

En la figura 12 vemos el diagrama de secuencia que se da cuando solicitamos la opción de ruta hacia un centro educativo.

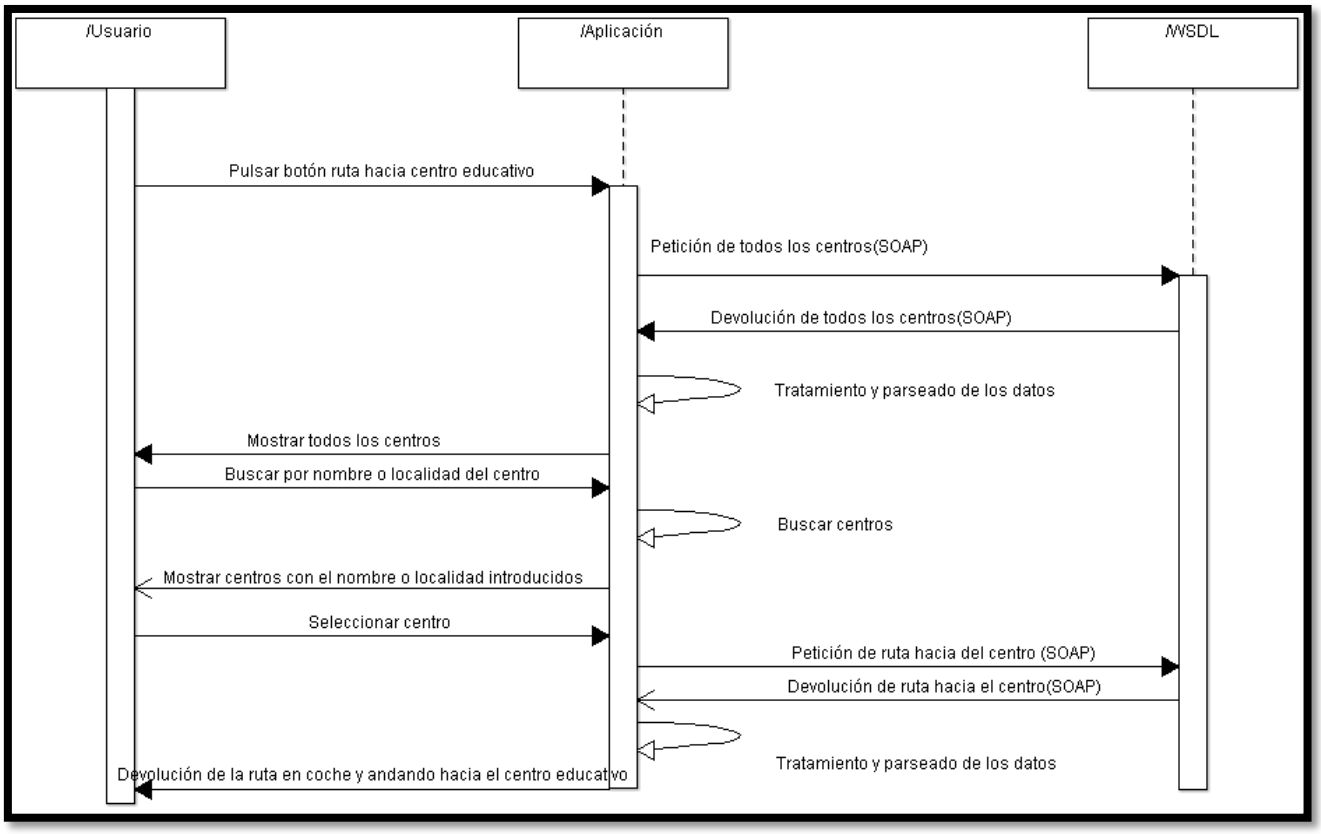


Figura 12: Diagrama de secuencia ruta hacia centro educativo

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

2.6 – Tablas de importaciones

A continuación, se detallan unas tablas con todas las importaciones que tiene cada una de las clases que forman parte del proyecto.

BusquedaAvanzadaActivity	BusquedaMapaActivity
import java.util.Locale;	import java.util.List;
import android.app.Dialog;	import java.util.Locale;
import android.content.Context;	import android.content.Context;
import android.content.Intent;	import android.content.Intent;
import android.content.SharedPreferences;	import android.content.SharedPreferences;
import android.content.pm.ActivityInfo;	import android.content.pm.ActivityInfo;
import android.content.res.Configuration;	import android.content.res.Configuration;
import android.os.Bundle;	import android.content.res.Resources;
import android.support.v4.app.Fragment;	import android.os.Bundle;
import android.support.v4.app.NavUtils;	import android.support.v4.app.Fragment;
import android.support.v7.app.ActionBarActivity;	import android.support.v7.app.ActionBarActivity;
import android.view.LayoutInflater;	import android.util.Log;
import android.view.Menu;	import android.view.LayoutInflater;
import android.view.MenuItem;	import android.view.Menu;
import android.view.View;	import android.view.MenuItem;
import android.view.View.OnClickListener;	import android.view.View;
import android.view.ViewGroup;	import android.view.ViewGroup;
import android.view.WindowManager;	import android.widget.TabHost;
import android.view.WindowManager.LayoutParams;	import android.widget.TabHost.OnTabChangeListener;
	import android.widget.TextView;
	import android.widget.Toast;
	import com.google.android.gms.maps.CameraUpdateFactory;
	import com.google.android.gms.maps.GoogleMap;
	import com.google.android.gms.maps.GoogleMap.InfoWindowAdapter;
	import com.google.android.gms.maps.GoogleMap.OnInfoWindowClickListener;
	import com.google.android.gms.maps.SupportMapFragment;
	import com.google.android.gms.maps.model.BitmapDescriptorFactory;
	import com.google.android.gms.maps.model.CameraPosition;
	import com.google.android.gms.maps.model.LatLng;

Tabla de importaciones 1: BusquedaAvanzadaActivity y BusquedaMapaActivity

CaminoActivity	Centros
import java.util.ArrayList;	import java.io.ByteArrayInputStream;
import java.util.List;	import java.io.InputStream;
import java.util.Locale;	import java.io.Serializable;
import org.w3c.dom.Document;	import java.util.List;
import android.content.Context;	import Proceso_XML.Parser;
import android.content.Intent;	import Proceso_XML.ParserBuscAv;
import android.content.SharedPreferences;	import SOAP.TareaWSBuscadorAvanzado;
import android.content.pm.ActivityInfo;	import SOAP.TareaWSConsulta;
import android.content.res.Configuration;	import android.app.ProgressDialog;
import android.graphics.Color;	import android.util.Log;
import android.graphics.PorterDuff;	
import android.location.Location;	
import android.location.LocationManager;	
import android.net.Uri;	
import android.os.Bundle;	
import android.support.v7.app.ActionBarActivity;	
import android.util.Log;	
import android.view.MenuItem;	
import android.view.View;	
import android.view.View.OnClickListener;	
import android.widget.ImageView;	
import android.widget.Toast;	
import com.google.android.gms.maps.CameraUpdateFactory;	
import com.google.android.gms.maps.GoogleMap;	
import com.google.android.gms.maps.SupportMapFragment;	
import com.google.android.gms.maps.model.CameraPosition;	
import com.google.android.gms.maps.model.LatLng;	
import com.google.android.gms.maps.model.PolylineOptions;	

Tabla de importaciones 2: CaminoActivity y Centros

CentroAdapter	CentrosActivity
import java.io.Serializable;	import java.io.Serializable;
import java.util.List;	import java.util.ArrayList;
import android.util.Log;	import android.app.ProgressDialog;
import android.view.LayoutInflater;	import android.content.Context;
import android.view.View;	import android.content.Intent;
import android.view.ViewGroup;	import android.content.pm.ActivityInfo;
import android.widget.BaseAdapter;	import android.content.res.Resources;
import android.widget.TextView;	import android.graphics.PorterDuff;
	import android.os.Bundle;
	import android.support.v4.app.Fragment;
	import android.support.v4.app.NavUtils;
	import android.support.v7.app.ActionBarActivity;
	import android.text.Editable;
	import android.text.TextWatcher;
	import android.util.Log;
	import android.view.LayoutInflater;
	import android.view.View;
	import android.view.View.OnClickListener;
	import android.view.ViewGroup;
	import android.widget.AdapterView;
	import android.widget.AdapterView.OnItemClickListener;
	import android.widget.TabHost.OnTabChangeListener;
	import android.widget.EditText;
	import android.widget.ImageView;
	import android.widget.ListView;
	import android.widget.TabHost;

Tabla de importaciones 3: CentroAdapter y CentrosActivity

HacerDireccion	InfoAdicionalActivity
import java.io.InputStream;	import java.io.ByteArrayInputStream;
import java.util.ArrayList;	import java.io.InputStream;
import javax.xml.parsers.DocumentBuilder;	import java.util.Locale;
import javax.xml.parsers.DocumentBuilderFactory;	import Proceso_XML.ParserInfoAdicional;
import org.apache.http.HttpResponse;	import SOAP.TareaWSdetalleCentro;
import org.apache.http.client.HttpClient;	import android.content.Context;
import org.apache.http.client.methods.HttpPost;	import android.content.Intent;
import org.apache.http.impl.client.DefaultHttpClient;	import android.content.SharedPreferences;
import org.apache.http.protocol.BasicHttpContext;	import android.content.pm.ActivityInfo;
import org.apache.http.protocol.HttpContext;	import android.content.res.Configuration;
import org.w3c.dom.Document;	import android.content.res.Resources;
import org.w3c.dom.Node;	import android.net.Uri;
import org.w3c.dom.NodeList;	import android.os.Bundle;
import android.os.AsyncTask;	import android.support.v7.app.ActionBarActivity;
import android.util.Log;	import android.text.Html;
import com.google.android.gms.maps.model.LatLng;	import android.util.Log;
	import android.view.MenuItem;
	import android.view.View;
	import android.widget.TabHost;
	import android.widget.TabHost.OnTabChangeListener;
	import android.widget.TextView;

Tabla de importaciones 4: HacerDireccion y InfoAdicionalActivity

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

InfoCentrosActivity	MapaActivity
import java.io.ByteArrayInputStream;	import android.content.SharedPreferences;
import java.io.InputStream;	import android.content.pm.ActivityInfo;
import java.util.Locale;	import android.content.res.Configuration;
import Proceso_XML.ParserDetalleCentro;	import android.content.res.Resources;
import SOAP.TareaWSdetalleCentro;	import android.graphics.PorterDuff;
import android.app.AlertDialog;	import android.location.Location;
import android.content.Context;	import android.location.LocationManager;
import android.content.DialogInterface;	import android.net.Uri;
import android.content.Intent;	import android.os.Bundle;
import android.content.SharedPreferences;	import android.support.v4.app.Fragment;
import android.content.pm.ActivityInfo;	import android.support.v7.app.ActionBarActivity;
import android.content.res.Configuration;	import android.util.Log;
import android.content.res.Resources;	import android.view.LayoutInflater;
import android.net.Uri;	import android.view.Menu;
import android.os.Bundle;	import android.view.MenuItem;
import android.support.v7.app.ActionBarActivity;	import android.view.View;
import android.text.Html;	import android.view.View.OnClickListener;
import android.util.Log;	import android.view.ViewGroup;
import android.view.Menu;	import android.widget.ImageView;
import android.view.MenuItem;	import android.widget.TabHost;
import android.view.View;	import android.widget.TabHost.OnTabChangeListener;
import android.widget.TabHost;	import android.widget.Toast;
import android.widget.TabHost.OnTabChangeListener	import com.google.android.gms.maps.CameraUpdateFactory;
import android.widget.TextView;	import com.google.android.gms.maps.GoogleMap;
	import com.google.android.gms.maps.SupportMapFragment;
	import com.google.android.gms.maps.model.CameraPosition;
	import com.google.android.gms.maps.model.LatLng;
	import com.google.android.gms.maps.model.MarkerOptions;

Tabla de importaciones 5: InfoCentrosActivity y MapaActivity

MultiSpinner	PantallaPrincipalActivity
import android.app.AlertDialog;	import java.util.Locale;
import android.content.Context;	import android.app.AlertDialog;
import android.content.DialogInterface;	import android.content.Context;
import android.content.DialogInterface.OnMultiChoic	import android.content.DialogInterface;
import android.content.res.TypedArray;	import android.content.Intent;
import android.util.AttributeSet;	import android.content.SharedPreferences;
import android.widget.ArrayAdapter;	import android.content.pm.ActivityInfo;
import android.widget.Spinner;	import android.content.res.Configuration;
	import android.graphics.PorterDuff;
	import android.net.ConnectivityManager;
	import android.os.Bundle;
	import android.support.v4.app.Fragment;
	import android.support.v7.app.ActionBarActivity;
	import android.util.Log;
	import android.view.LayoutInflater;
	import android.view.Menu;
	import android.view.MenuItem;
	import android.view.View;
	import android.view.ViewGroup;
	import android.view.View.OnClickListener;
	import android.widget.ImageView;

Tabla de importaciones 6: MultiSpinner y PantallaPrincipalActivity

Parser	ParserBuscAv
import java.io.IOException;	import java.io.IOException;
import java.io.InputStream;	import java.io.InputStream;
import java.util.ArrayList;	import java.util.ArrayList;
import java.util.List;	import java.util.List;
import org.xmlpull.v1.XmlPullParser;	import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;	import org.xmlpull.v1.XmlPullParserException;
import android.util.Xml;	import android.util.Xml;
import com.example.pruebasoap.Centro;	import com.example.pruebasoap.Centro;
ParserDetalleCentro	**ParserInfoAdicional**
import java.io.IOException;	import java.io.IOException;
import java.io.InputStream;	import java.io.InputStream;
import java.util.ArrayList;	import java.util.ArrayList;
import java.util.List;	import java.util.List;
import org.xmlpull.v1.XmlPullParser;	import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;	import org.xmlpull.v1.XmlPullParserException;
import android.util.Xml;	import android.util.Xml;
import com.example.pruebasoap.Centro;	import com.example.pruebasoap.Centro;

Tabla de importaciones 7: Parser, ParserBuscAv, ParserDetalleCentro y ParserInfoAdicional

SSLConnection	TareaWSBuscadorAvanzado
import android.util.Log;	import java.util.Locale;
import javax.net.ssl.HostnameVerifier;	import org.ksoap2.SoapEnvelope;
import javax.net.ssl.SSLSession;	import org.ksoap2.serialization.SoapObject;
import javax.net.ssl.TrustManager;	import org.ksoap2.serialization.SoapSerializationEnvelope;
import java.security.KeyManagementException;	import org.ksoap2.transport.HttpTransportSE;
import java.security.NoSuchAlgorithmException;	import android.os.AsyncTask;
import java.security.SecureRandom;	import android.util.Log;
import java.security.cert.CertificateException;	
import java.security.cert.X509Certificate	
TareaWSConsulta	**TareaWSDetalleCentro**
import org.ksoap2.SoapEnvelope;	import org.ksoap2.SoapEnvelope;
import org.ksoap2.serialization.SoapObject;	import org.ksoap2.serialization.SoapObject;
import org.ksoap2.serialization.SoapSerializationEnve	import org.ksoap2.serialization.SoapSerializationEnvelope;
import org.ksoap2.transport.HttpTransportSE;	import org.ksoap2.transport.HttpTransportSE;
import android.os.AsyncTask;	import android.os.AsyncTask;
import android.util.Log;	import android.util.Log;

Tabla de importaciones 8: SSLConnection, TareaWSBuscadorAvanzado, TareaWSConsulta y TareaWSDetalleCentro

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

2.7 – Arquitectura del sistema

El sistema, estará basado en la comunicación de la aplicación Android del dispositivo móvil del usuario con el servidor de datos WSDL del Gobierno de Navarra, con datos abiertos sobre el departamento de educación. Este servidor tiene unas funciones públicas para consultar la información de los centros educativos.

WSDL son las siglas de Web Services Description Language, un formato XML que se utiliza para describir servicios Web . La versión 1.0 fue la primera recomendación por parte del W3C y la versión 1.1 no alcanzó nunca tal estatus. La versión 2.0 se convirtió en la recomendación actual por parte de dicha entidad.

WSDL describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje.

Así, WSDL se usa a menudo en combinación con SOAP y XML Schema. Un programa cliente que se conecta a un servicio web puede leer el WSDL para determinar qué funciones están disponibles en el servidor. Los tipos de datos especiales se incluyen en el archivo WSDL en forma de XML Schema. El cliente puede usar SOAP para hacer la llamada a una de las funciones listadas en el WSDL.

El WSDL nos permite tener una descripción de un servicio web. Especifica la interfaz abstracta a través de la cual un cliente puede acceder al servicio y los detalles de cómo se debe utilizar.

Para hacer la conexión SOAP desde el cliente, en este caso la aplicación Android, se ha usado la librería ksoap2, ya que Android no trae por defecto opción de conexión directa mediante SOAP.

Figura 13: Arquitectura del proyecto

La arquitectura que se ha explicado anteriormente queda reflejada en la figura 13, para mostrar claramente la interacción entre la aplicación y el servidor WSDL con los datos abiertos del departamento de educación mediante el protocolo SOAP.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

3 – Implementación del proyecto

3.1 – Aplicación móvil Android

En este apartado explicaré cómo se ha llevado a cabo la implementación de la aplicación móvil para Android Buscador de Centros Educativos de Navarra.

3.1.1 – Instalación de Eclipse y SDK de Android

Como primera acción, debemos tener instalado el programa Eclipse, o en su defecto tener un ejecutable del mismo. Esto se puede lograr en la siguiente URL, para lograr la versión Eclipse Juno, que es la que se ha utilizado para el desarrollo de este proyecto.

http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops4/R-4.2.1-201209141800/eclipse-SDK-4.2.1-win32.zip

Eclipse es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado *Java Development Toolkit* (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). En la Figura 14 vemos la imagen que nos muestra al lanzar Eclipse.

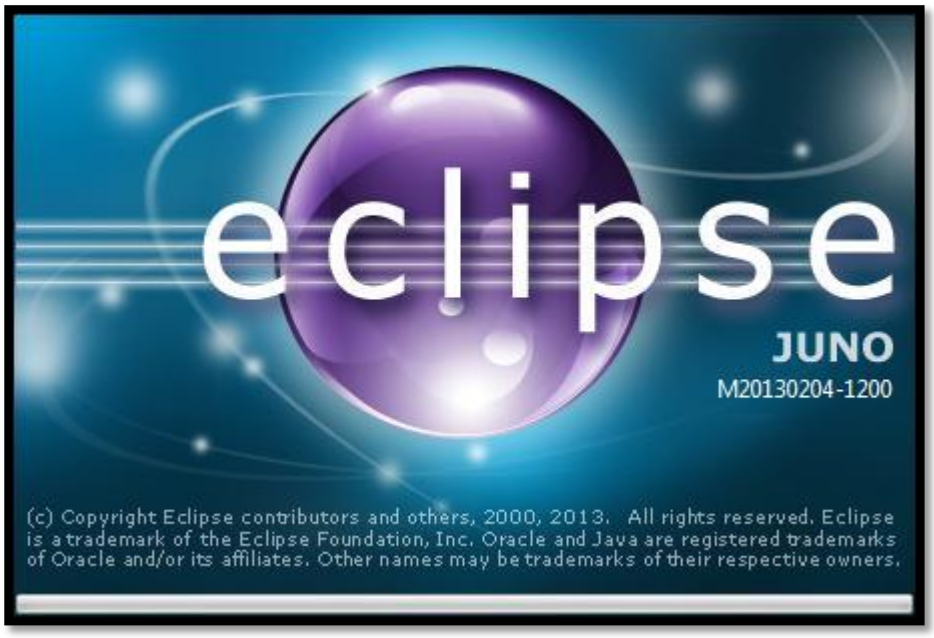


Figura 14: Eclipse

Para poder desarrollar aplicaciones java, tendremos que instalar y agregar a Eclipse el SDK que proporciona Android para el desarrollo de aplicaciones en su sistema operativo.

Para ello, en la página de Android Developers http://developer.android.com/sdk/installing/installing-adt.html podemos seguir un sencillo tutorial y de esa manera lograr acoplar el SDK de Android y Eclipse.

Cuando tengamos todo instalado, en la Figura 15 vemos el espacio de trabajo que veremos en nuestra pantalla.

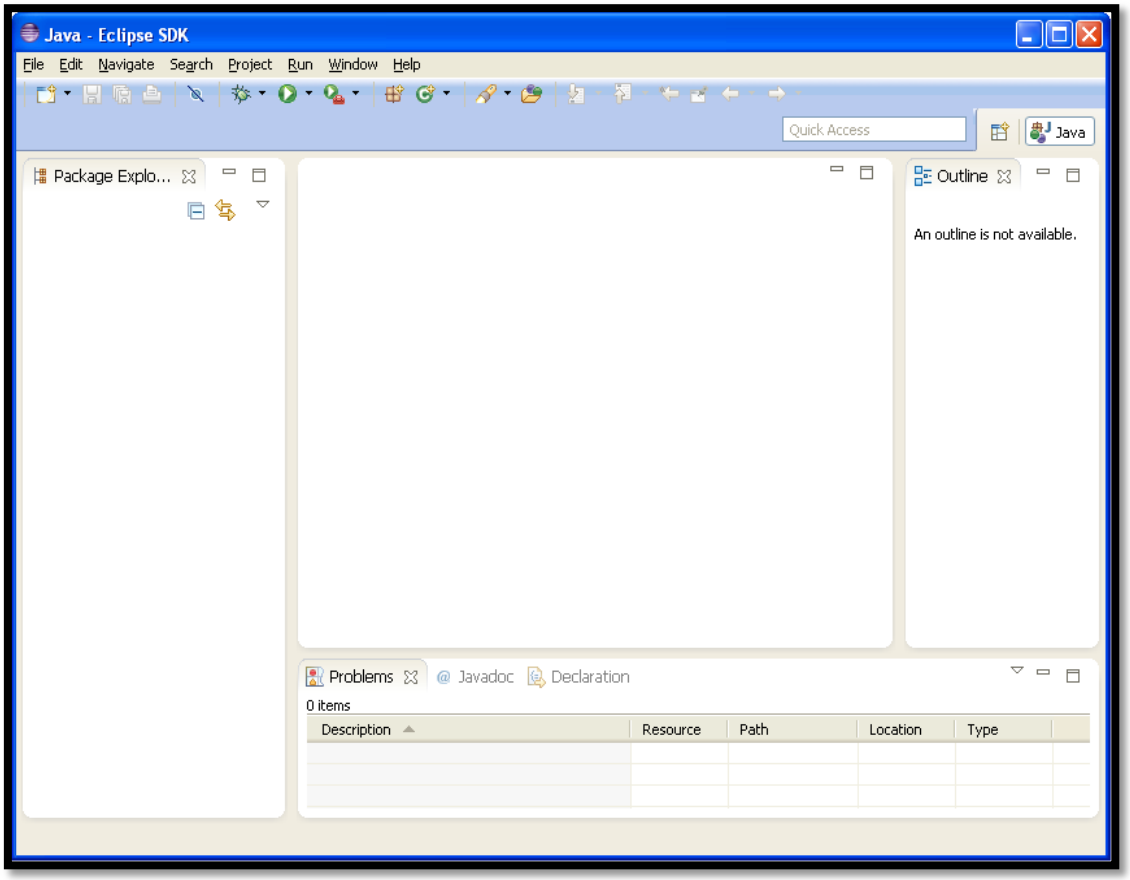


Figura 15: Espacio de trabajo Eclipse

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Para observar que hemos instalado correctamente el SDK de Android, nos deberán aparecer estas dos imágenes que vemos en la Figura 16 pertenecientes al Android SDK Manager y Android Virtual Device Manager.

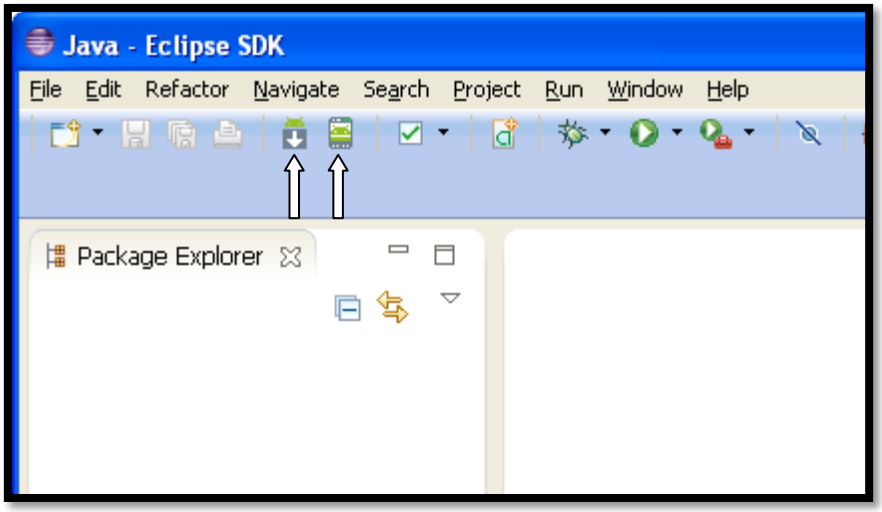


Figura 16: Eclipse con SDK de Android

Ahora ya tenemos las aplicaciones necesarias para empezar a desarrollar el proyecto Android.

3.1.2 –Creación del proyecto Android

Para crear el proyecto Android tendremos que hacer click en File – New – Other…

Nos saldrá la siguiente pantalla que vemos en la Figura 17, y en ella crearemos un Android Application Project de la siguiente manera:

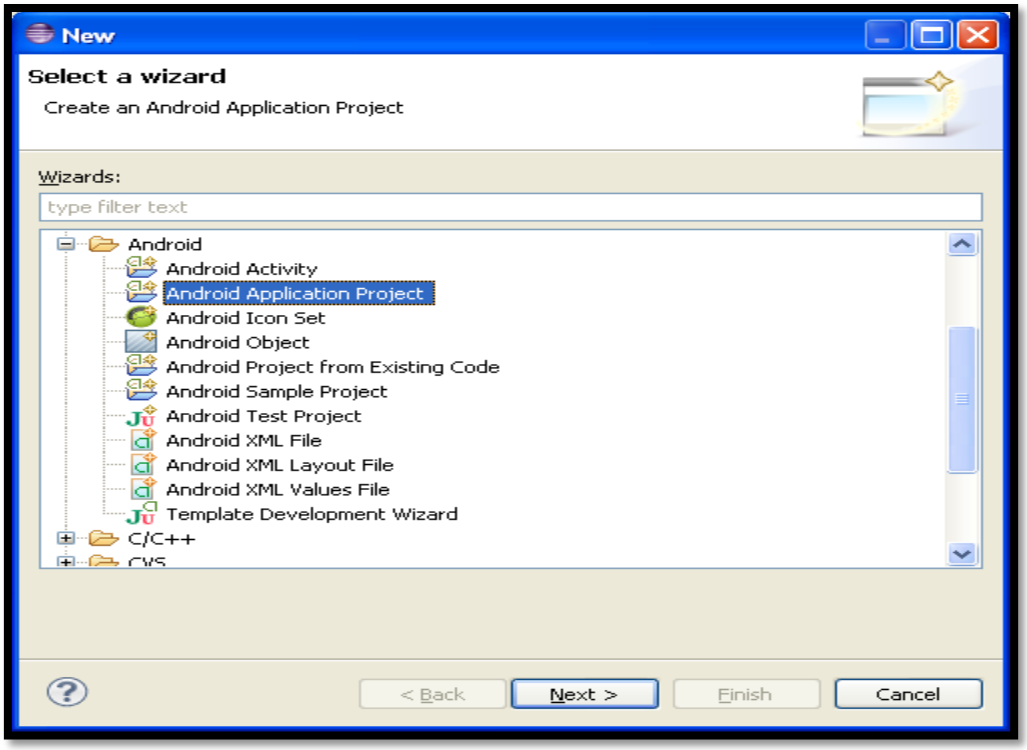


Figura 17: Creación del proyecto Android

Seguiremos rellenando los campos que se nos requieran como en nombre del proyecto, mínima API para la cual queremos que funcione el proyecto… y ya lo tendremos creado y estará listo para empezar a programar.

Fijándonos en la parte izquierda, tendremos todas las carpetas que vemos en la Figura 18:

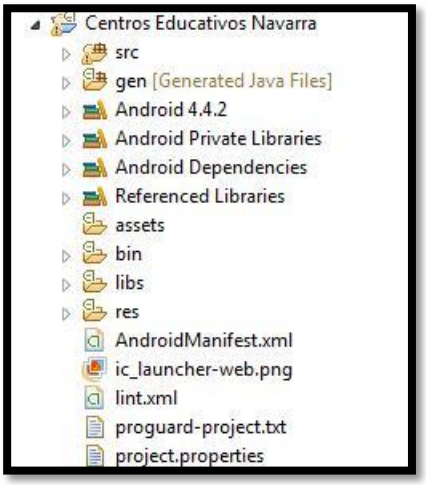


Figura 18: Carpetas aplicación Android

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

src: Carpeta que contiene el código fuente de la aplicación. Como se puede observar los ficheros Java se almacenan en un espacio de nombres.

gen: Carpeta que contiene el código generado de forma automática por el SDK. Nunca hay que modificar de forma manual estos ficheros. Dentro encontraremos:

BuildConfig.java: Define la constante DEBUG para que desde Java puedas saber si tu aplicación está en fase de desarrollo.

R.java: Define una clase que asocia los recursos de la aplicación con identificadores. De esta forma los recursos podrán ser accedidos desde Java.

Android 4.4.2: Código JAR, el API de Android según la versión seleccionada.

Android Dependencies: Librerías asociadas al proyecto.

assets: Carpeta que puede contener una serie arbitraria de ficheros o carpetas que podrán ser utilizados por la aplicación (ficheros de datos, fuentes,…). A diferencia de la carpeta *res,* nunca se modifica el contenido de los ficheros de esta carpeta ni se les asociará un identificador.

bin: En esta carpeta se compila el código y se genera el .apk, fichero comprimido que contine la aplicación final lista para instalar.

libs: Código JAR con librerías que quieras usar en tu proyecto. Se ha añadido una librería android-support cuyo objetivo es añadir nuevas funcionalidades que no aparecían en el nivel de API 4 (recuerda que es el nivel de API mínimo que hemos indicado) y que aparecieron en versiones más recientes del SDK.

res: Carpeta que contiene los recursos usados por la aplicación. Como veremos en el siguiente capítulo las subcarpetas pueden tener un sufijo si queremos que el recurso solo se cargue al cumplirse una condición. Por ejemplo –hdpi significa que solo ha de cargar los recursos contenidos en esta carpeta cuando el dispositivo donde se instala la aplicación tiene una densidad gráfica alta (>180dpi); -v11significa que el recurso solo ha de cargarse en un dispositivo con nivel de API 11 (v3.0).

drawable: En esta carpeta se almacenan los ficheros de imágenes (JPG o PNG) y descriptores de imágenes en XML.

layout: Contiene ficheros XML con vistas de la aplicación. Las vistas nos permitirán configurar las diferentes pantallas que compondrán la interfaz de usuario de la aplicación. Se utiliza un formato similar al HTML usado para diseñar páginas Web. Serán tratadas en el siguiente capítulo.

menu:Ficheros XML con los menús de cada actividad.

values: También utilizaremos ficheros XML para indicar valores del tipo string, color o estilo. De esta manera podremos cambiar los valores sin

necesidad de ir al código fuente. Por ejemplo, nos permitiría traducir una aplicación a otro idioma.

anim: Contiene ficheros XML con animaciones Tween

animator: Contiene ficheros XML con animaciones de propiedades.

xml: Otros ficheros XML requeridos por la aplicación.

raw: Ficheros adicionales que no se encuentran en formato XML.

AndroidManifest.xml: Este fichero describe la aplicación Android. En él se indican las *actividades*, *intenciones*, *servicios* y *proveedores de contenido* de la aplicación. También se declaran los permisos que requerirá la aplicación. Se indica la versión mínima de Android para poder ejecutarla, el paquete Java, la versión de la aplicación, etc.

ic_launcher-web.png: Icono de la aplicación de gran tamaño para ser usado en páginas Web. El nombre puede variar si se indicó uno diferente en el proceso de creación del proyecto. Ha de tener una resolución de 512x512 (con alfa).

proguard-project.txt: Fichero de configuración de la herramienta *ProGuard*, que te permite optimizar y ofuscar el código generado. Es decir, se obtiene un .apk más pequeño y donde resulta más difícil hacer ingeniería inversa.

default.properties: Fichero generado automáticamente por el SDK. Nunca hay que modificarlo. Se utiliza para comprobar la versión del API y otras características cuando se instala la aplicación en el terminal.

3.1.3 –Ciclo de vida de las actividades

Una aplicación en Android va a estar formada por un conjunto de elementos básicos de visualización, conocidos como actividades. Además de varias actividades una aplicación también puede contener servicios. Son las actividades las que realmente controlan el ciclo de vida de las aplicaciones, dado que el usuario no cambia de aplicación, sino de actividad. El sistema va a mantener una pila con las actividades previamente visualizadas, de forma que el usuario va a poder regresar a la actividad anterior pulsando la tecla "atrás".

Para entender muchas de las funciones que tienen las Activitys de Android, hay que conocer cuál es el ciclo de vida de una actividad, que vemos en la Figura 19.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

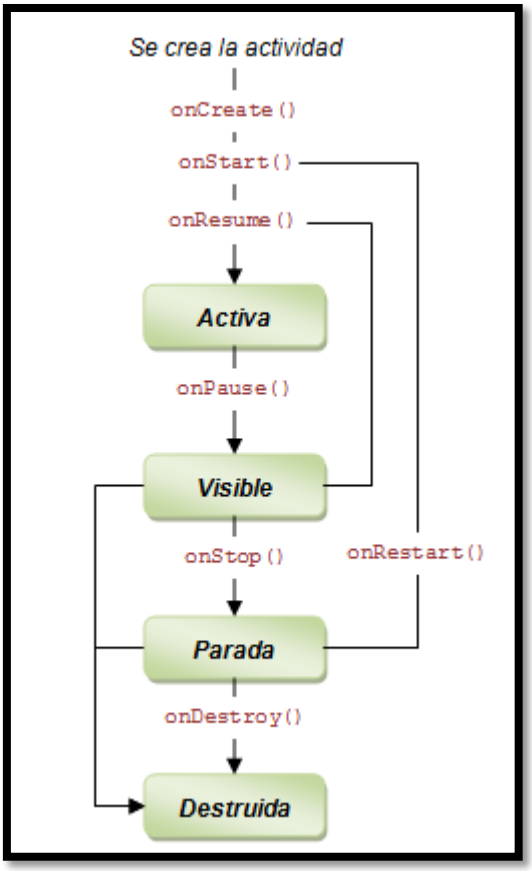


Figura 19: Ciclo de vida Android

Descripción de esos métodos:

onCreate(Bundle): Se llama en la creación de la actividad. Se utiliza para realizar todo tipo de inicializaciones, como la creación de la interfaz de usuario o la inicialización de estructuras de datos. Puede recibir información de estado de instancia (en una instancia de la clase Bundle), por si se reanuda desde una actividad que ha sido destruida y vuelta a crear.

onStart(): Nos indica que la actividad está a punto de ser mostrada al usuario.

onResume(): Se llama cuando la actividad va a comenzar a interactuar con el usuario. Es un buen lugar para lanzar las animaciones y la música.

onPause(): Indica que la actividad está a punto de ser lanzada a segundo plano, normalmente porque otra aplicación es lanzada. Es el lugar adecuado para detener animaciones, música o almacenar los datos que estaban en edición.

onStop(): La actividad ya no va a ser visible para el usuario. Ojo si hay muy poca memoria, es posible que la actividad se destruya sin llamar a este método.

onRestart(): Indica que la actividad va a volver a ser representada después de haber pasado por *onStop()*.

onDestroy(): Se llama antes de que la actividad sea totalmente destruida. Por ejemplo, cuando el usuario pulsa el botón <volver> o cuando se llama al método *finish()*. Ojo si hay muy poca memoria, es posible que la actividad se destruya sin llamar a este método.

3.1.4 – Descripción de las clases

En este apartado se describirá el objetivo de cada una de las clases del proyecto, así como las partes más importantes de la programación de las mismas (sólo las más importantes). En este apartado se mostrarán las pantallas de la aplicación por funcionalidad, ya que todas ellas estarán mostradas en su totalidad con el flujo de funcionamiento de la misma en uno de los siguientes apartados de esta memoria que será el manual de usuario, para mostrar más claramente el funcionamiento de la misma.

PantallaPrincipalActivity.java y especificaciones del idioma

Esta actividad es la actividad principal, que se carga en cuanto abrimos la aplicación, y tiene la apariencia que se muestra en la siguiente figura número 20.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 20: Pantalla principal de la aplicación

Lo primero que se hace en la aplicación es ver qué idioma es nuestro preferido para la aplicación. Por defecto será el idioma en el que tengamos nuestro terminal móvil, castellano o euskera. Por si cambiamos el idioma de la aplicación, lo almaceno como preferencia compartida (como si fuera un campo de una base de datos tradicional), y compruebo para ofrecer cada pantalla de la aplicación en el idioma preferido, de la forma que muestro en la siguiente figura 21.

```
SharedPreferences prefs =
        getSharedPreferences("MisPreferencias",Context.MODE_PRIVATE);
if(prefs.getString("idioma", "es").equalsIgnoreCase("es")){
    Locale locale = new Locale("es");
Locale.setDefault(locale);
Configuration config = new Configuration();
config.locale = locale;
Context context =getApplicationContext() ;
context.getApplicationContext().getResources().updateConfiguration(config, null);
}
else if (prefs.getString("idioma", "es").equalsIgnoreCase("eu")){
    Locale locale = new Locale("eu");
Locale.setDefault(locale);
Configuration config = new Configuration();
config.locale = locale;
Context context =getApplicationContext() ;
context.getApplicationContext().getResources().updateConfiguration(config, null);
}
```

Figura 21: Preferencias compartidas para el idioma

Ahora ya sabemos cuál es el idioma preferido por el usuario, por lo que ofrecemos la aplicación en dicho idioma, castellano o euskera. Respecto al idioma, la aplicación hace la separación de todos los textos en las carpetas y layouts para que el código escrito en la aplicación sea completamente independiente al idioma. De esta manera, también sería muy sencillo extender esta aplicación a cualquier otro idioma, ya que este paso podría hacerse sin hacer mucho esfuerzo extra.

En nuestro caso, tenemos las carpetas values y layout para el idioma castellano y values-eu y layout-eu para ofrecer la aplicación en euskera, como vemos en la siguiente figura 22.

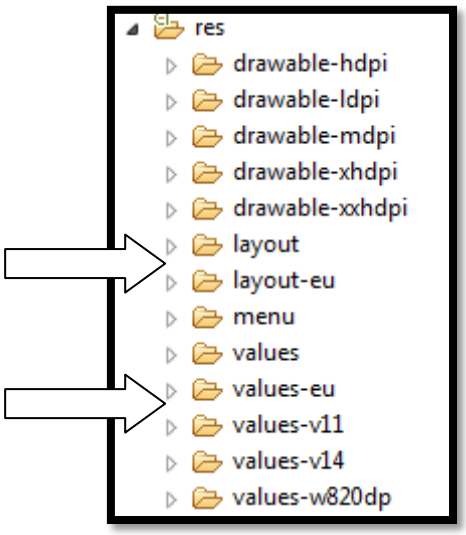


Figura 22: Carpetas values y layout multi idioma

Dentro las carpetas values y values-eu, tenemos todos los textos en cada uno de los idiomas disponibles, como podemos observar en las siguientes imágenes 23 y 24, perteneciente a los archivos string en cada uno de los idiomas.

```
<string name="nombre">Nombre del centro: </string>
<string name="direccion">Dirección: </string>
<string name="cp">Código postal: </string>
<string name="localidad">Localidad: </string>
<string name="telefono">Teléfono: </string>
<string name="fax">Fax: </string>
<string name="email">E-mail: </string>
<string name="web">Web: </string>
<string name="naturaleza">Naturaleza: </string>
<string name="info_basica">Información básica</string>
<string name="info_adicional">Plan de estudios \ny otros datos</string>
<string name="localizacion">Mapa de localización</string>
<string name="modelo">Modelos y programas lingüísticos:</string>
<string name="tipo">Tipo de enseñanza:</string>
<string name="programas">Programas:</string>
<string name="objetivos">Objetivos educativos:</string>
<string name="sitna">Enlace Sitna:</string>
<string name="zona">Zona lingüística:</string>
<string name="matriculas">Número de matrículas:</string>
<string name="servicios">Servicios complementarios:</string>
<string name="atencion">Atención NEE:</string>
<string name="plan">Plan de estudios:</string>
```

Figura 23: Parte del archivo strings en español

```
<string name="nombre">Ikastetxearen izena: </string>
<string name="direccion">Helbidea: </string>
<string name="cp">Posta kodea: </string>
<string name="localidad">Herria: </string>
<string name="telefono">Telefonoa: </string>
<string name="fax">Faxa: </string>
<string name="email">E-maila: </string>
<string name="web">Webgunea: </string>
<string name="naturaleza">Mota: </string>
<string name="info_basica">Oinarrizko datuak</string>
<string name="info_adicional">Ikasketa plangintza \neta datu gehiago</string>
<string name="localizacion">Kokapen mapa</string>
<string name="modelo">Hizkuntz ereduak eta programak:</string>
<string name="tipo">Irakaskuntzak:</string>
<string name="programas">Programak:</string>
<string name="objetivos">Ikasketa helburuak:</string>
<string name="sitna">Sitna webgunea:</string>
<string name="zona">Eremu linguistikoa:</string>
<string name="matriculas">Matrikula kopurua:</string>
<string name="servicios">Zerbitzu osagarriak:</string>
<string name="atencion">Hezkuntza premia bereziak:</string>
<string name="plan">Ikasketa plana:</string>
```

Figura 24: Parte del archivo strings en euskera

Si queremos cambiar el idioma de la aplicación, lo haremos pulsando sobre el botón de alguno de los idiomas en la parte superior derecha de la pantalla principal. Si pulsamos sobre el botón ES (español), ponemos la aplicación en español guardando el idioma como preferencia compartida para guardarlo para futuras ejecuciones de la misma, y vamos a la actividad correspondiente.

```
SharedPreferences prefs =
     getSharedPreferences("MisPreferencias",Context.MODE_PRIVATE);

SharedPreferences.Editor editor = prefs.edit();
editor.putString("idioma", "es");
editor.commit();
image4.setColorFilter(0xFF6E6E6E, PorterDuff.Mode.MULTIPLY);
Locale locale = new Locale("es");
Locale.setDefault(locale);
Configuration config = new Configuration();
config.locale = locale;
Context context =getApplicationContext() ;
context.getApplicationContext().getResources().updateConfiguration(config, null);
Intent intent = new Intent(PantallaPrincipalActivity.this, PantallaPrincipalActivity.class);
startActivity(intent);
finish();
```

Figura 25: Cambio de idioma

Respecto a los layouts en otro idioma, la actuación es similar, cargando en cada uno la imagen con los textos en el idioma correspondiente.

Siguiendo ya con la descripción de la clase, se comprueba que el dispositivo en el que está instalada la aplicación tenga conexión a internet. Es imprescindible para su uso estar conectado a la red de redes, ya que para lograr toda la información acerca de los centros educativos se hace haciendo peticiones al servidor del gobierno de Navarra que tiene alojada toda esta información, vías SOAP como ya se ha comentado.

De no contar con conexión a internet, se muestra una ventana y se cierra la aplicación, en caso de cómo es normal hoy en día en todos los smartphones cuentan con conexión a internet la aplicación se inicia correctamente.

```
public boolean isOnline() {
    ConnectivityManager cm =
        (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);

    return cm.getActiveNetworkInfo() != null &&
       cm.getActiveNetworkInfo().isConnectedOrConnecting();
}
```

Figura 26: Comprobación de la conexión a internet

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

```
if(!isOnline()){
    // Instantiate an AlertDialog.Builder with its constructor
    AlertDialog.Builder builder = new AlertDialog.Builder(this);


    // Chain together various setter methods to set the dialog characteristics
    builder.setMessage(R.string.dialog_message)
           .setTitle(R.string.dialog_title);

    // Add the buttons
    builder.setPositiveButton(R.string.ok, new DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int id) {
                   // User clicked OK button
                   finish();
               }
           });
    AlertDialog dialog = builder.create();
    dialog.setCancelable(false);
      dialog.show();

}
```

Figura 27: Ventana error en conexión a internet

Si pulsamos sobre alguno de los 3 botones principales de la pantalla, que como hemos podido observar son “Listado completo de centros”, “Buscador avanzado de centros” o “Ruta hacia centro educativo” vamos a cada una de las actividades correspondientes.

CentrosActivity.java y CentroAdapter.java

Estas dos clases, en asociación con algunas otras se encargan de mostrarnos los listados de centros en "modo lista". A esta pantalla llegamos pulsando sobre el primero de los botones para que se nos muestren todos los centros, o cuando hacemos una búsqueda avanzada, así como también pulsando sobre el tercero de los botones de la pantalla principal.

El listado de centros se mostrará como vemos en la siguiente imagen:

Figura 28: Listado de centros "modo lista"

Para ver desde qué actividad hemos llegado a esta, se hará una comprobación, ya que el comportamiento de la aplicación variará dependiendo de cuál sea la actividad predecesora a esta. Lo hacemos como se muestra en la figura 29.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

```
final CentroAdapter adaptador;

if (bundle.getString("modo").equalsIgnoreCase("buscadorSimple") ){
    adaptador = new CentroAdapter(bundle.getString("modo"), null, null,null,null,null);
}
else if (bundle.getString("modo").equalsIgnoreCase("camino") ){
    adaptador = new CentroAdapter(bundle.getString("modo"), null, null,null,null,null);
    setTitle(getResources().getString(R.string.destino));
}
else {
    adaptador = new CentroAdapter(bundle.getString("modo"), (boolean[])getIntent().getExtras()
}
```

Figura 29: Modo de procedencia a CentrosActivity

Como observamos, según vengamos de una u otra actividad el constructor del adaptador tendrá diferentes características.

Antes de explicar cómo se muestran los datos de los centros mediante el adaptador, la otra característica importante de esta clase es el buscador por nombre de centro o localidad que podemos ver en la parte superior de la pantalla, debajo de las pestañas “Lista” y “Mapa”.

Usaremos el TextChangeListener para mostrar los centros que concuerdan con el nombre o localidad especificados cada vez que cambiemos una letra en el campo de texto, y lo hago de la forma que se muestra en la siguiente figura 30.

```
for (int i = 0; i < adaptador.ll.size(); i++) {
    if (textlength <= adaptador.getItemOriginal(i).getNombre().length()) {
        if ((adaptador.getItemOriginal(i).getNombre().toLowerCase().contains(et.getText().toString().toLowerCase()))) {
            array_sort.add(adaptador.getItemOriginal(i));
            //Log.d("lat",(String.valueOf(adaptador.getItemOriginal(i).getLatitud())));
        }

        else {
            if (textlength <= adaptador.getItemOriginal(i).getLocalidad().length()){

                if ((adaptador.getItemOriginal(i).getLocalidad().toLowerCase().contains(et.getText().toString().toLowerCase()))) {
                array_sort.add(adaptador.getItemOriginal(i));
            }
        }}
    }
```

Figura 30: Buscador por nombre de centro o localidad

Tendremos un listener en cada uno de los elementos de la lista y pasaremos su identificador a la actividad que muestra los detalles del centro para saber de cuál mostraré la información.

En la clase CentroAdapter lo que hacemos es pedir la información de los centros que queramos, ya sea una búsquea avanzada o simplemente mostrar todos los centros. Desde el adaptador, llamaremos a los métodos getInfoCentos() o getBuscadorAvanzadoCentros() dependiendo de qué actividad vengamos, como se muestra en la siguiente imagen.

```
public CentroAdapter(String modo, boolean[] ano, boolean[] naturaleza,boolean[] modelo,boolean[] servicios,boolean[] nivel ){
    //this.modo=modo;
    Centro c= new Centro("","","","",0,0);
    if (modo.equalsIgnoreCase("buscadorSimple") || modo.equalsIgnoreCase("camino")){
        l = c.getInfoCentros();
    }
    else if (modo.equalsIgnoreCase("buscadorAvanzado")){
        l = c.getBuscadorAvanzadoCentros(ano, naturaleza, modelo, servicios, nivel);
    }
```

Figura 31: Llamada a métodos para mostrar los centros

En el método getView del adaptador ponemos todos los nombres de los centros y las localidades en las que se encuentran de la siguiente manera:

```
TextView nombre = (TextView)convertView.findViewById(R.id.textView1);
TextView localidad = (TextView)convertView.findViewById(R.id.textView2);

Centro centro = l.get(position);

nombre.setText(centro.getNombre());
localidad.setText(centro.getLocalidad());

return convertView;
```

Figura 32: Código para rellenar el listado de centros y localidades

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Centro.java

Esta clase es la que define qué es un centro educativo y que información se guarda de cada uno de los centros. Por tanto, no tendrá una vista asociada, sino que es utilizada por otras actividades para lograr la información requerida.

Para los centros tengo 2 constructores, uno en la que solamente guardo información de 5 atributos de la clase (id, nombre, localidad, teléfono, latitud y longitud) y otro para guardar toda la información completa que pueden tener los centros, que consta de 27 atributos.

De la misma manera, tengo los métodos get y set de todos estos atributos para garantizar el acceso seguro a cada uno de ellos.

Para acabar de describir la clase, tenemos los métodos getInfoCentros y getBuscadorAvanzadoCentros que llaman a la clase que hace la petición SOAP y al parser para tratar la información recibida mediante el protocolo SOAP.

```
TareaWSConsulta llam_soap = (TareaWSConsulta) new TareaWSConsulta().execute();
Boolean ready = llam_soap.getReady();
while(!ready){
    Thread.sleep(1000);
    ready = llam_soap.getReady();
}

Parser p = new Parser();
InputStream in = new ByteArrayInputStream(llam_soap.resSoap.toString().getBytes("UTF-8"));
l = p.parse(in);
```

Figura 33: Método getInfoCentros

Tras recibir y parsear la información, se devuelve un array de centros con la información de los centros al adapter que hemos visto anteriormente y que se pueda mostrar la información.

El método getBuscadorAvanzadoCentros es análogo a éste, con las diferencias de que se llama a otro método SOAP al que se le pasan los parámetros de la búsqueda avanzada y luego se llama a su parser específico para los datos recibidos.

TareaWSConsulta.java, TareaWSBuscadorAvanzado.java, TareaWSdetalleCentro.java y SSLConnection.java

Estas 4 clases son las que se encargan de hacer posibles las peticiones por medio del protocolo SOAP al Web Service del Gobierno de Navarra para lograr la información de los centros educativos.

Para las nuevas versiones de Android es necesario habilitar los certificados SSL para hacer peticiones web, como en nuestro caso. Para llevar a cabo esta tarea se ha desarrollado la clase SSLConnection.java, que será llamada al principio de cada una de las otras tres tareas descritas en este apartado.

Para esta tarea, se ha desarrollado la función allowAllSSL(), como se muestra en la siguiente figura.

```
public static void allowAllSSL() {

    javax.net.ssl.HttpsURLConnection.setDefaultHostnameVerifier(new HostnameVerifier() {
        @Override
        public boolean verify(String hostname, SSLSession session) {
            return true;
        }
    });

    javax.net.ssl.SSLContext context;

    if (trustManagers == null) {
        trustManagers = new TrustManager[]{new _FakeX509TrustManager()};
    }

    try {
        context = javax.net.ssl.SSLContext.getInstance("TLS");
        context.init(null, trustManagers, new SecureRandom());
        javax.net.ssl.HttpsURLConnection.setDefaultSSLSocketFactory(context.getSocketFactory());
    } catch (NoSuchAlgorithmException e) {
        Log.e("allowAllSSL", e.toString());
    } catch (KeyManagementException e) {
        Log.e("allowAllSSL", e.toString());
    }
}
```

Figura 34: Método allowAllSSL

En este caso no era algo vitalmente importante hacer las peticiones por un medio seguro, ya que los datos que circulan en estos intercambios son accesibles a todo usuario que quiera mediante el portal de datos abiertos del Gobierno de Navarra, pero como las últimas versiones de Android lo requieren, se ha hecho usando las conexiones SSL.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Ahora explicaré como hago las conexiones SOAP al Web Service del Gobierno de Navarra. Para ello lo haré explicando la clase TareaWSBuscadorAvanzado. Las clases TareaWSBuscadorSimple y TareaWSdetalleCentro son muy similares a la del buscador avanzado, sólo que usando la definición SOAP correspondiente a cada clase.

Para hacer las peticiones SOAP, Android todavía no cuenta con una librería propia, por lo que hay que hacerlo mediante librerías externas. En mi caso, he utilizado la librería ksoap2-android, que es una ligera y eficiente librería cliente para Android.

Ya tenemos todas las herramientas para poder usar el servicio web del Gobieno de Navarra, por lo que ahora tendremos que fijarnos en la definición del cliente SOAP. Como ya he dicho, para usar la función de buscadorAvanzado, es la siguiente que se muestra en las próximas figuras.

Parámetros de entrada

buscadorAvanzado(int naturaleza[], int ubicacion[], int modeloLinguistico[], int servicios[], int atencionNEE[], int ensenanzas[], int ciclos[], int orden, int numPagina, int numElementos, String idioma)

```
<ctr:buscadorAvanzado>
    <!--Zero or more repetitions:-->
    <naturaleza>?</naturaleza>
    <!--Zero or more repetitions:-->
    <ubicacion>?</ubicacion>
    <!--Zero or more repetitions:-->
    <modeloLinguistico>?</modeloLinguistico>
    <!--Zero or more repetitions:-->
    <servicios>?</servicios>
    <!--Zero or more repetitions:-->
    <atencionNEE>?</atencionNEE>
    <!--Zero or more repetitions:-->
    <ensenanzas>?</ensenanzas>
    <!--Zero or more repetitions:-->
    <ciclos>?</ciclos>
    <orden>?</orden>
    <numPagina>?</numPagina>
    <numElementos>?</numElementos>
    <!--Optional:-->
    <idioma>?</idioma>
</ctr:buscadorAvanzado>
```

Figura 35: Definición SOAP Buscador Avanzado de centros

Como se puede observar, para hacer la petición SOAP, tendremos que especificar varios parámetros, unos opcionales y otros obligatorios. Cabe destacar también que otros parámetros pueden aparecer más de una vez, es decir, por ejemplo, podremos tener dos valores distintos de naturaleza, por si queremos pedir centros de naturaleza pública o naturaleza concertada. Un ejemplo simple de una petición SOAP como el arriba descrito, podría ser la siguiente:

```
<ctr:buscadorAvanzado>
    <!--Zero or more repetitions:-->
    <naturaleza>1</naturaleza>
    <ubicacion>310970001</ubicacion>
    <orden>1</orden>
    <numPagina>1</numPagina>
    <numElementos>2</numElementos>
    <!--Optional:-->
    <idioma>es</idioma>
</ctr:buscadorAvanzado>
```

Figura 36: Petición SOAP buscadorAvanzado simple

Las peticiones SOAP, como dependen de un número importante de factores, como por ejemplo, el tiempo de respuesta del servidor, se han hecho en tareas asíncronas, para de esta manera no ralentizar el funcionamiento de la aplicación. Por tanto, se hacen en el método doInBackground(), después de que la clase extienda de AsyncTask, como vemos en las siguientes imágenes.

```
public class TareaWSBuscadorAvanzado extends AsyncTask<String,Integer,Boolean> {
```

Figura 37: Definición de la clase

```
public Boolean doInBackground(String... params) {
```

Figura 38: Implementación método doInBackground()

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Ya dentro del método doInBackground, definimos la URL, espacio de nombres, nombre del método y acción SOAP a realizar, que en el caso que estoy explicando queda de la siguiente manera.

```
final String URL = " https://educa.educacion.navarra.es/Educa/services/ConsultaCentros?wsdl";
final String NAMESPACE="http://ctr.ws.educa.cein.com/";
final String METHOD_NAME = "buscadorAvanzado";
final String SOAP_ACTION = "";
```

Figura 39: Definición direcciones SOAP

Seguidamente, llamamos al método allowAllSSL antes descrito y creamos un nuevo objeto SOAP, especificando el espacio de nombres y el nombre del método. A continuación vamos añadiendo propiedades, como el curso escolar, naturaleza… dependiendo de los parámetros que hayamos recibido. En el ejemplo se muestran 4 de los 39 parámetros que tiene este método en particular.

```
    SSLConnection.allowAllSSL();
SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
if (params[0].equalsIgnoreCase("1")){
    request.addProperty("cursoEscolar",126);
}
if (params[1].equalsIgnoreCase("1")){
    request.addProperty("cursoEscolar",125);
}
if (params[2].equalsIgnoreCase("1")){
    request.addProperty("naturaleza",1);
}
if (params[3].equalsIgnoreCase("1")){
    request.addProperty("naturaleza",2);
```

Figura 40: Objeto SOAP y especificación de propiedades

Una vez especificadas todas las propiedades del cuerpo SOAP, se "envuelve" el objeto, se hace la petición a la web y se recibe la respuesta SOAP del Web Service en cuestión.

```
SoapSerializationEnvelope envelope =new SoapSerializationEnvelope(SoapEnvelope.VER11);


envelope.setOutputSoapObject(request);

HttpTransportSE transporte = new HttpTransportSE(URL);
transporte.debug = true;

    transporte.call(SOAP_ACTION, envelope);
    resSoap = transporte.responseDump;
```

Figura 41: Petición y respuesta SOAP

Ahora que ya tenemos la respuesta SOAP, se procederá a "parsear" o tratar esa información para recoger lo que nos es interesante para cada caso.

Parser.java, ParserBuscAv.java, ParserDetalleCentro.java y ParserInfoAdicional.java

La función de estas cuatro clases es "parsear" o tratar la información recibida en las peticiones SOAP que hemos hecho para consultar datos sobre los centros educativos.

Estas cuatro clases son similares entre sí, con la diferencia de que cada una de ellas espera las etiquetas correspondientes a la petición que se haya hecho.

La clase Parser.java se usa para recibir la información de nombre del centro y localidad cuando se solicitan todos los centros, ParserBuscAv.java extrae la misma información cuando se ha hecho una búsqueda avanzada. Por su parte, ParserDetalleCentro.java trata información básica sobre un centro educativo en concreto, y ParserInfoAdicional.java parsea la información adicional de un centro educativo en concreto.

Para explicarlas, nos centraremos en ParserInfoAdicional.java, ya que todas ellas son muy similares entre sí, y explicando una nos hacemos una clara idea de cómo funciona cada una de ellas.

Cuando seleccionemos un centro educativo para ver información a cerca de él, recibiremos un documento como ya se ha comentado que tratar para mostrar la información que le interese al usuario.

En este caso, si seleccionamos el centro Navarro Villoslada, recibiremos el siguiente documento, que se muestra en las siguientes figuras.

```
<return>
    <clave>1</clave>
    <valor>Navarro Villoslada</valor>
</return>
<return>
    <clave>2</clave>
    <valor>ARCADIO Mª LARRAONA, 3</valor>
</return>
```

Figura 42: Primera parte documento recibido

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Como podemos observar, en la primera parte recibimos la información codificada mediante unos números de clave y los valores correspondientes.

```
<return>
    <clave>10</clave>
    <lista>Modelo G (castellano)</lista>
    <lista>Modelo A (castellano con asignatura vascuence)</lista>
    <lista>Aprendizaje en inglés y/o francés en Secundaria</lista>
</return>
<return>
    <clave>11</clave>
    <lista>Educación Secundaria Obligatoria</lista>
    <lista>Bachillerato</lista>
</return>
```

Figura 42: Segunda parte documento recibido

En la segunda parte, tras recibir 11 atributos diferentes, recibimos una clave y una lista en la que se ofrece la información solicitada, para otros 3 atributos.

Y para finalizar, vemos en la figura 43 la tercera parte del documento, que es más complicada, con diferentes etiquetas anidadas para ofrecer la información.

```
<return>
    <clave>18</clave>
    <pe>
        <id>1</id>
        <nombre>PLAN_ESTUDIOS</nombre>
        <tipo>1</tipo>
        <etapaGrupo>
            <id>403</id>
            <nombre>Educación Secundaria Obligatoria</nombre>
            <tipo>102</tipo>
            <etapa>
                <id>4</id>
                <modeloLinguistico>G</modeloLinguistico>
                <modeloLinguistico>A</modeloLinguistico>
                <modeloLinguistico>S.Bil.-G</modeloLinguistico>
                <modeloLinguistico>S.Bil.-A</modeloLinguistico>
                <nombre>Educación Secundaria Obligatoria</nombre>
                <tipo>2</tipo>
                <url>http://www.educacion.navarra.es/portal/etapa/4</url>
                <curso>
                    <id>20</id>
                    <modeloLinguistico>G</modeloLinguistico>
                    <modeloLinguistico>A</modeloLinguistico>
                    <modeloLinguistico>S.Bil.-G</modeloLinguistico>
                    <modeloLinguistico>S.Bil.-A</modeloLinguistico>
                    <nombre>1º ESO</nombre>
                    <tipo>5</tipo>
                </curso>
                <curso>
                    <id>21</id>
                    <modeloLinguistico>G</modeloLinguistico>
                    <modeloLinguistico>A</modeloLinguistico>
                    <modeloLinguistico>S.Bil.-G</modeloLinguistico>
                    <modeloLinguistico>S.Bil.-A</modeloLinguistico>
                    <nombre>2º ESO</nombre>
                    <tipo>5</tipo>
                </curso>
```

Figura 43: Tercera parte del documento recibido

Para tratar esta información lo haremos mediante un XMLPullParser. Para ello, hacemos un método llamado readEntry() que devuelve un centro con su información completa.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

```
private Centro readEntry(XmlPullParser parser) throws XmlPullParserException, IOException {
    parser.require(XmlPullParser.START_TAG, ns, "ns2:detalleCentroResponse");
    String clave = null;
    String id = null;
    String nombre = null;
    String localidad = null;
    String telefono = null;
    double latitud = 0.00;
    double longitud = 0.00;
    String direccion = null;
    String cp = null;
    String fax= null;
    String email= null;
    String web= null;
    String naturaleza= null;
    String modelo= null;
    String tipo= null;
    String servicios= null;
    String atencionnee= null;
    String programas= null;
    String tanos= null;
    String sitna= null;
    String plan= null;
    String matriculas= null;
    String objetivos= null;
    String valores= null;
    String distinciones= null;
    String jornada= null;
    String proyectos= null;
    String reconocimientos= null;
    String zona = null;
```

Figura 44: Definición método readEntry() para parsear los detalles de un centro

Como vemos, devolveremos un centro con todos sus 29 atributos. Al principio inicializamos todos los atributos con valores nulos, y conforme vamos leyendo el documento les asignaremos los valores obtenidos.

```
while (parser.next() != XmlPullParser.END_TAG) {
    try{
    if (parser.getEventType() != XmlPullParser.START_TAG) {
        continue;
    }
    String name = parser.getName();
    if (name.equals("return")) {
        parser.nextTag();
        String tag_clave = parser.getName();
        if (tag_clave.equals("clave")){
        clave = readClave(parser);
        if (clave.equalsIgnoreCase("1")){;
            nombre = readString(parser);
        }
        if (clave.equalsIgnoreCase("2")){
            direccion = readString(parser);
        }
```

Figura 45: Parseo primera parte del documento

Para leer la primera parte del documento antes mostrado, primero esperamos la etiqueta return, seguida de clave y el número identificador, como se observa en este caso, el número 1 para el nombre y el 2 para la dirección del centro.

```
else if (clave.equalsIgnoreCase("10")) {
    modelo = readStringLista(parser);
}
else if (clave.equalsIgnoreCase("11")) {
    tipo = readStringLista(parser);
}
else if (clave.equalsIgnoreCase("12")) {
    servicios = readStringLista(parser);
}
```

Figura 46: Parseo segunda parte del documento

Para leer la segunda parte del documento, llamamos al método readStringLista(), ya que esperamos la clave, y una lista de elementos, siendo una lista de la que no sabemos su longitud, que tendremos que mostrar. El método queda definido de la siguiente manera.

```
private String readStringLista(XmlPullParser parser) throws IOException, XmlPullParserException {
    String string = "";
    while (parser.getName().equals("lista")) {
        parser.require(XmlPullParser.START_TAG, ns, "lista");
        string = string + "<br />" + readText(parser);
        parser.require(XmlPullParser.END_TAG, ns, "lista");
        parser.nextTag();
    }
    return string;
}
```

Figura 47: Método readStringLista()

Finalmente, para leer el atributo con clave 18, que es el plan de estudios del centro, la tercera parte del documento, he desarrollado la siguiente función read18().

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

```
private String read18(XmlPullParser parser) throws IOException, XmlPullParserException {
    String string = "";
    try{
        parser.require(XmlPullParser.START_TAG, ns, "pe");
        readText(parser);
        while(!parser.getName().equals("pe")){
            try{
            if (parser.getName().equals("nombre")){
                string = string + "<br />" + readText(parser);
            }
            else{
                readText(parser);
            }
            }catch(Exception e){
            }
        }
        parser.require(XmlPullParser.END_TAG, ns, "pe");
        parser.nextTag();
    }catch(Exception e){
        Log.d("excpar", e.toString());
    }
    string = string.replace("<br />PLAN_ESTUDIOS<br />", "");
    return string;

}
```

Figura 48: Función read18()

En esta ocasión buscamos la etiqueta “pe” para encontrar la información adecuada que tendremos que mostrar.

BusquedaAvanzadaActivity.java y MultiSpinner.java

Estas dos clases son las encargadas de que se puedan hacer las búsquedas avanzadas en la aplicación. La clase MultiSpinner es la auxiliar para mostrar los listados de opciones que permiten la selección múltiple al elegir las opciones en cada uno de los campos de la mencionada búsqueda avanzada.

La pantalla correspondiente a BusquedaAvanzadaActivity, será la siguiente:

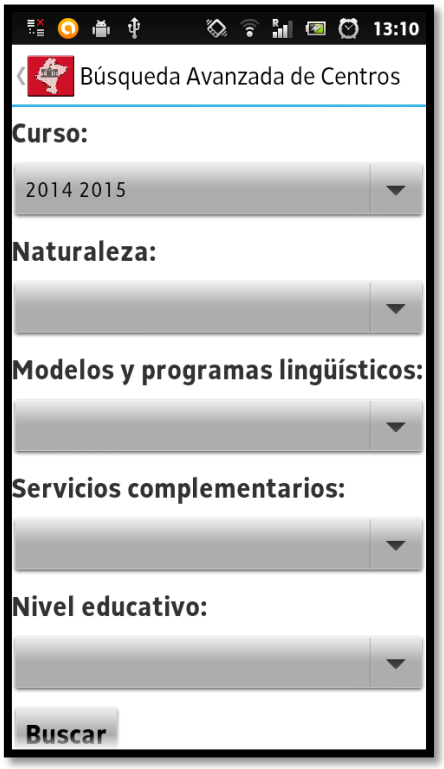


Figura 49: Pantalla para la búsqueda avanzada

Y la clase MultiSpinner es una auxiliar para lograr lo que vemos en esta figura número 50.

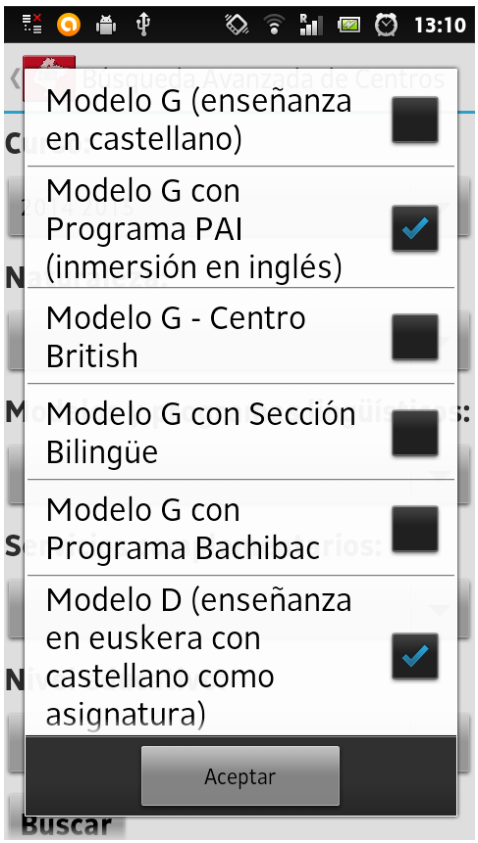


Figura 50: Opciones selección múltiple

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Empezando a explicar la clase principal de las búsquedas avanzadas, empezamos poniendo 5 spinners múltiples, que permiten la selección múltiple de campos.

```
final MultiSpinner multiSpinner1 = (MultiSpinner) findViewById(R.id.multispinner1);
final MultiSpinner multiSpinner2 = (MultiSpinner) findViewById(R.id.multispinner2);
final MultiSpinner multiSpinner3 = (MultiSpinner) findViewById(R.id.multispinner3);
final MultiSpinner multiSpinner4 = (MultiSpinner) findViewById(R.id.multispinner4);
final MultiSpinner multiSpinner5 = (MultiSpinner) findViewById(R.id.multispinner5);
```

Figura 51: MultiSpinners

A continuación rellenamos las casillas a chequeado o vacío dependiendo de la última búsqueda que haya hecho el usuario, usando las preferencias compartidas de Android. Es decir, los campos que estén marcados serán los que ha rellenado el usuario al hacer su última búsqueda, o en blanco en caso de que sea la primera vez que utilice esta opción.

```
prefs =
        getSharedPreferences("MisPreferencias",Context.MODE_PRIVATE);

if (prefs.getBoolean("ano1", true)){
    multiSpinner1.setSelected(0);
}
if (prefs.getBoolean("ano2", false)){
    multiSpinner1.setSelected(1);
}
if (prefs.getBoolean("natu1", false)){
    multiSpinner2.setSelected(0);
}
```

Figura 52: Rellenado de campos mediante preferencias compartidas

Cuando pulsamos el botón de buscar, recogemos cuáles han sido las opciones seleccionadas para buscar utilizando estas posibilidades y guardamos las preferencias compartidas para guardar la forma de mostrar la pantalla en el futuro.

```
boolean[] prueba1 = multiSpinner1.getSelected();
boolean[] prueba2 = multiSpinner2.getSelected();
boolean[] prueba3 = multiSpinner3.getSelected();
boolean[] prueba4 = multiSpinner4.getSelected();
boolean[] prueba5 = multiSpinner5.getSelected();
Log.d("valores", "hola"+ prueba1[0]);
SharedPreferences prefs =
        getSharedPreferences("MisPreferencias",Context.MODE_PRIVATE);

    SharedPreferences.Editor editor = prefs.edit();
    editor.putBoolean("ano1", multiSpinner1.getSelectedI(0));
    editor.putBoolean("ano2", multiSpinner1.getSelectedI(1));
    editor.putBoolean("natu1", multiSpinner2.getSelectedI(0));
```

Figura 53: Recogida de opciones y almacenado de preferencias

Le pasaremos a la actividad CentrosActivity la información de la búsqueda avanzada para desde ahí gestionar la llamada a los métodos SOAP y después parseado de la información como ya hemos visto.

De la clase MultiSpinner, cabe destacar el método que hace escribir el texto de todos los campos seleccionados separados por comas en la pantalla encima del selector.

```
private DialogInterface.OnClickListener mOnClickListener = new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        // build new spinner text & delimiter management
        StringBuffer spinnerBuffer = new StringBuffer();
        for (int i = 0; i < entries.length; i++) {
            if (selected[i]) {
                spinnerBuffer.append(entries[i+1]);
                spinnerBuffer.append(", ");
            }
        }

        // Remove trailing comma
        if (spinnerBuffer.length() > 2) {
            spinnerBuffer.setLength(spinnerBuffer.length() - 2);
        }

        // display new text
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(getContext(),
                android.R.layout.simple_spinner_item,
                new String[] { spinnerBuffer.toString() });
        //spinnerBuffer.
        //adapter.remove(getItem(0));
        setAdapter(adapter);

        if (listener != null) {
            listener.onItemsSelected(selected);
        }

        // hide dialog
        dialog.dismiss();
    }
};
```

Figura 54: Método mOnClickListener de MultiSpinner

La información de cada uno de los campos, está guardada en el archivo strings, como string-array.

```
<string-array name="curso">
    <item></item>
    <item>2014 2015</item>
    <item>2013 2014</item>
</string-array>
```

Figura 55: Ejemplo string-array

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

BusquedaMapaActivity.java

Esta clase se encarga de mostrar un conjunto de centros educativos en modo "mapa" como se puede ver en la siguiente imagen, mostrando en este caso los centros que hay en Aoiz.

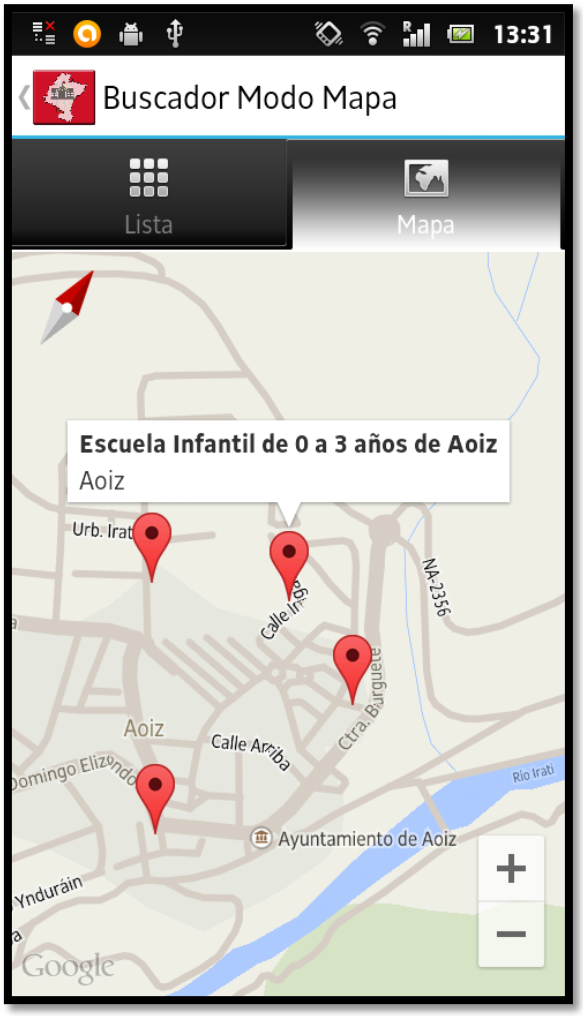


Figura 56: Búsqueda modo mapa centros en Aoiz

En primer lugar, inicializamos el mapa. Para hacer esta actividad, se ha utilizado el API versión 2 de Goole Maps.

```
private void initilizeMap() {
    if (googleMap == null) {
        googleMap = ((SupportMapFragment) getSupportFragmentManager().findFragmentById(
                R.id.map)).getMap();

        // check if map is created successfully or not
        if (googleMap == null) {
            Toast.makeText(getApplicationContext(),
                    "Sorry! unable to create maps", Toast.LENGTH_SHORT)
                    .show();
        }
    }
}
```

Figura 57: Inicialización del mapa

Para mostrarlos, hacemos un bucle para poner en el mapa todos los marcadores o "picas" del listado de centros seleccionados en ese momento.

```
for (int i = 0; i < l.size(); i++) {
// create marker
    final MarkerOptions marker;
    marker = new MarkerOptions().position(new LatLng(l.get(i).getLatitud(), l.get(i).getLongitud())).
marker.snippet(String.valueOf(i));
// adding marker
googleMap.addMarker(marker);
}

 googleMap.setInfoWindowAdapter(new InfoWindowAdapter(){

     public View getInfoWindow(Marker arg0){
         return null;
     }

     public View getInfoContents(Marker arg0){
         View v = getLayoutInflater().inflate(R.layout.marker, null);
         TextView centro = (TextView)v.findViewById(R.id.loc);
         centro.setText(arg0.getTitle());
         TextView localidad = (TextView)v.findViewById(R.id.loc1);
         localidad.setText(l.get(Integer.parseInt(arg0.getSnippet())).getLocalidad());
         return v;
     }
```

Figura 58: Rellenado del mapa con los marcadores

También reseñar que he desarrollado un adaptador para personalizar la información que se ofrece al pulsar sobre un marcador, para ofrecer el nombre del centro y la localidad. Al hacer click sobre el texto del marcador, iremos a la información detallada del centro educativo, igual que cuando pulsamos sobre un centro en el otro modo "lista".

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

InfoCentrosActivity.java

Esta clase es la encargada de mostrar la información básica de cada uno de los centros educativos. En ella podremos ver la información sobre el nombre del centro, dirección, código postal, localidad, teléfono, fax, email, web y naturaleza del mismo, como vemos en la siguiente imagen.

Figura 59: Información básica del centro educativo

Si de alguno de los centros educativos algunos de los campos no están disponibles, no serán mostrados en la pantalla.

Para conseguir esta información, se ha desarrollado una función llamada obtenerDetalle(), en la que se llama al método SOAP para pedir la información y al parser para tratarla, como se ve en la siguiente imagen.

```
Centro c = null;
TareaWSdetalleCentro llam_soap = (TareaWSdetalleCentro) new TareaWSdetalleCentro().execute(bundle.getString("id"));
ParserDetalleCentro p = new ParserDetalleCentro();
InputStream in = new ByteArrayInputStream(llam_soap.resSoap.toString().getBytes("UTF-8"));
c = p.parse(in);
```

Figura 60: Llamada a TareaWSdetalleCentro y ParserDetalleCentro

Tras hacer este par de operaciones, ya tendremos en el Centro c la información básica a ofrecer en la pantalla sobre el centro.

Ahora se procederá a rellenar la información de los textViews, en caso de que la información sea no nula, es decir, en caso de que esté disponible. Si no está disponible, el textView no será mostrado.

```
if(c.getNombre()==null){
    nombre = getResources().getString(R.string.nodisponible);
}
else {
    nombre = c.getNombre();
    text1.setText(Html.fromHtml("<font COLOR=#B41111>"+ getResources().getString(R.string.nombre)+"</font> " + nombre));
}
if(c.getDireccion()==null){
    //direccion = getResources().getString(R.string.nodisponible);
    text2.setVisibility(View.GONE);
}
else {
    direccion = c.getDireccion();
    text2.setText(Html.fromHtml("<font COLOR=#B41111>"+ getResources().getString(R.string.direccion)+"</font> " + direccion));
}
```

Figura 61: Rellenado de los 2 primeros TextViews

Algunos de los textos, los que se encuentran subrayados, serán clickables (teléfono, email y web), y en caso de ser pulsados, se lanzará la acción correspondiente (llamar, mandar email o visitar web).

```
email = c.getEmail();
text7.setText(Html.fromHtml("<font COLOR=#B41111>"+ getResources().getString(R.string.email)+"</font> <u> " + email + "</u>"));
text7.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try{
            Intent emailIntent = new Intent(Intent.ACTION_SENDTO, Uri.fromParts(
                    "mailto",email, null));
        emailIntent.putExtra(Intent.EXTRA_SUBJECT, "EXTRA_SUBJECT");
        startActivity(Intent.createChooser(emailIntent, "Send email..."));
        }catch(Exception e){

        }
    }
});
```

Figura 62: Ejemplo mandar email

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

InfoAdicionalActivity.java

Esta clase es muy similar a la anteriormente mostrada, con la diferencia que los datos mostrados son diferentes, siendo información adicional referente al plan de estudios y modelos lingüísticos ofrecidos por cada centro. Más concretamente modelos lingüísticos, tipo de enseñanza, servicios complementarios, atención nee, programas, plazas 3 años, plan de estudios, enlace a Sitna, número de matrículas, objetivos educativos, valores del centro, distinciones, jornada laboral, proyectos, reconocimientos y zona lingüística a la que pertenece.

Al igual que en la anterior, si alguno de los datos no está disponible no es mostrado. La pantalla tiene el diseño que se puede observar en al siguiente figura 63.

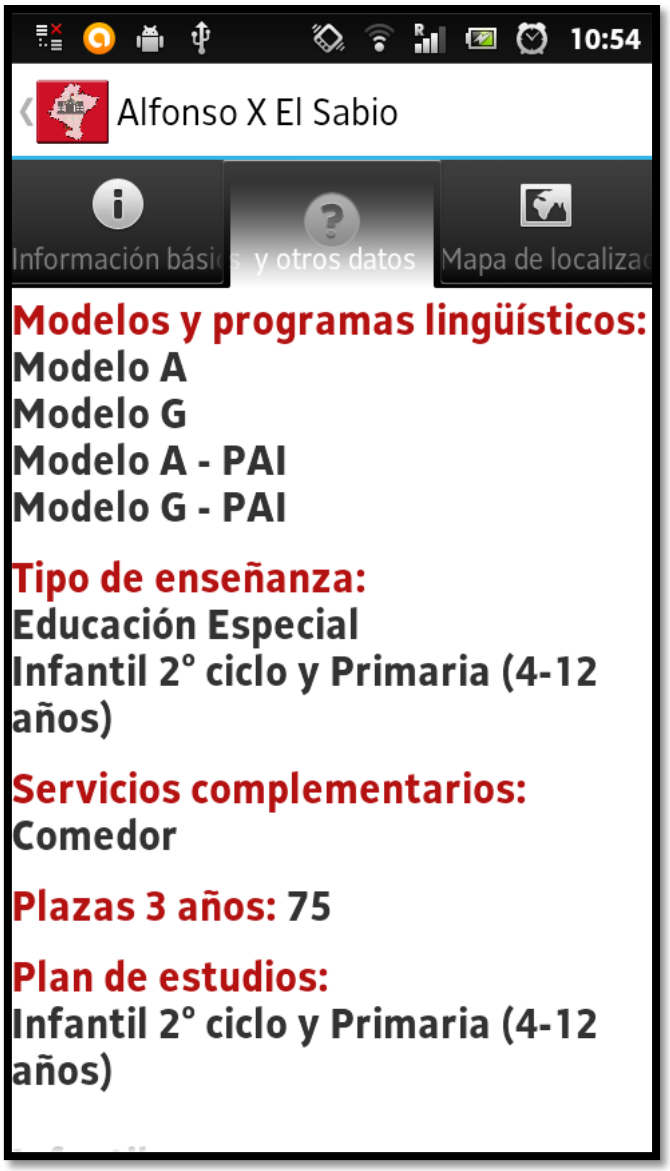


Figura 63: Información adicional del centro educativo

Como se ha hecho también en la clase anterior, se llama al método SOAP para recabar la información y luego se extrae parseándola, en este caso utilizando el parser ParserInfoAdicional, ya que la información que hay que leer es diferente.

```
TareaWSdetalleCentro llam_soap = (TareaWSdetalleCentro) new TareaWSdetalleCentro().execute(bundle.getString("id"));
ParserInfoAdicional p = new ParserInfoAdicional();
InputStream in = new ByteArrayInputStream(llam_soap.resSoap.toString().getBytes("UTF-8"));
c = p.parse(in);
```

Figura 64: Llamada a TareaWSdetalleCentro y ParserInfoAdicional

Y de nuevo de forma análoga a la clase anterior, ya tenemos la información adicional del centro y se procede a rellenar los TextViews.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

```
if(c.getModelo()==null){
    //modelo = getResources().getString(R.string.nodisponible);
    text1.setVisibility(View.GONE);
}
else {
    modelo = c.getModelo();
    text1.setText(Html.fromHtml("<font COLOR=#B41111>"+ getResources().getString(R.string.modelo)+"</font> " + modelo));
}
if(c.getTipo()==null){
    //tipo = getResources().getString(R.string.nodisponible);
    text2.setVisibility(View.GONE);
}
else {
    tipo = c.getTipo();
    text2.setText(Html.fromHtml("<font COLOR=#B41111>"+ getResources().getString(R.string.tipo)+"</font> " + tipo));
}
```

Figura 65: Rellenado de los 2 primeros TextViews

MapaActivity.java

Esta clase es para mostrar la localización de un centro educativo dentro de un mapa. Para desarrollar la clase, se ha usado la versión 2 del API de Google Maps.

El resultado de esta clase, sería esta pantalla.

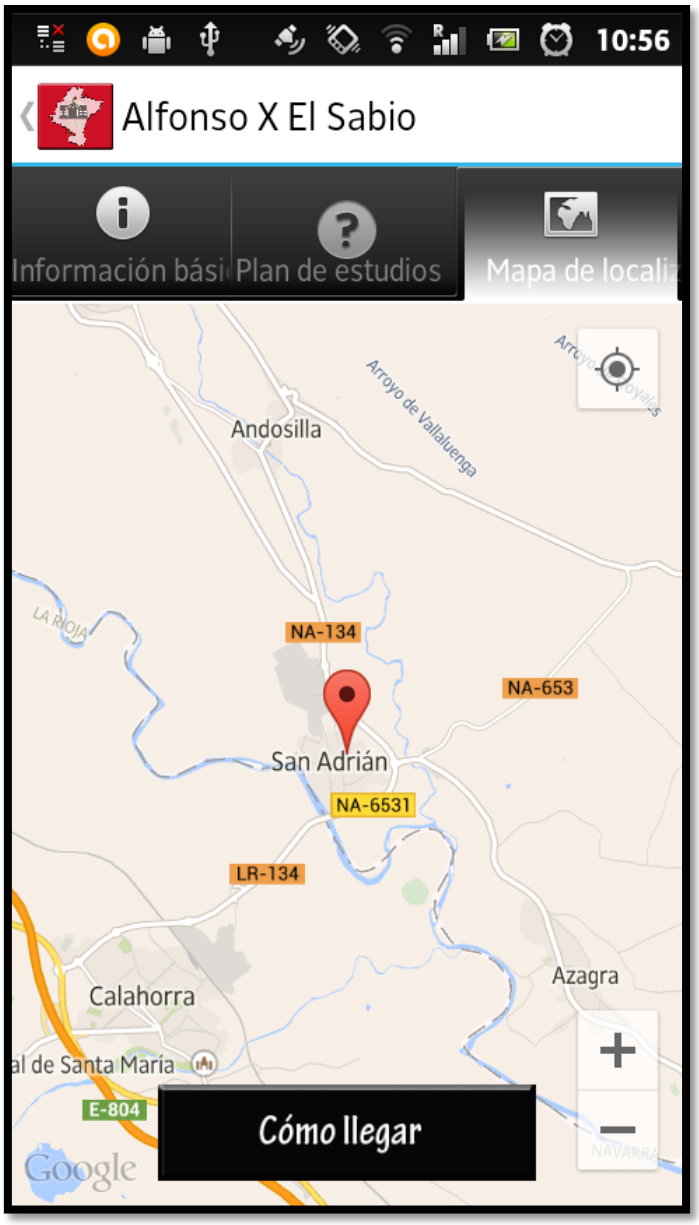


Figura 66: Localización en el mapa de centro educativo

Para empezar, inicializamos el mapa de la siguiente manera.

```
private void initilizeMap() {
    if (googleMap == null) {
        googleMap = ((SupportMapFragment) getSupportFragmentManager().findFragmentById(
                R.id.map)).getMap();

        // check if map is created successfully or not
        if (googleMap == null) {
            Toast.makeText(getApplicationContext(),
                    "Sorry! unable to create maps", Toast.LENGTH_SHORT)
                    .show();
        }
    }
}
```


upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 67: Inicialización del mapa

Y posteriormente, añadimos el marcador, ya que conocemos la latitud y la longitud del centro educativo. También ajustamos la posición de la cámara para mostrar la localización correctamente.

```
MarkerOptions marker = new MarkerOptions().position(new LatLng(latitude, longitude)).title(bundle.getString("nombre"));

// adding marker
googleMap.addMarker(marker);
CameraPosition cameraPosition = new CameraPosition.Builder().target(
        new LatLng(latitude, longitude)).zoom(12).build();
googleMap.animateCamera(CameraUpdateFactory.newCameraPosition(cameraPosition));
```

Figura 68: Añadir marcador y posicionar cámara

CaminoActivity.java y HacerDireccion.java

Estas dos clases se encargan de llevar a cabo la tercera y última de las opciones del menú principal, que es dibujar el camino desde nuestra ubicación actual hasta el centro educativo que especifiquemos. Nos mostrará con una línea verde la mejor ruta para ir andando y con una línea roja la mejor ruta para ir en coche.

En la figura vemos la pantalla en cuestión, que ofrece la ruta entre mi ubicación actual que es Aoiz y el centro Alfonso X el Sabio de San Adrián.

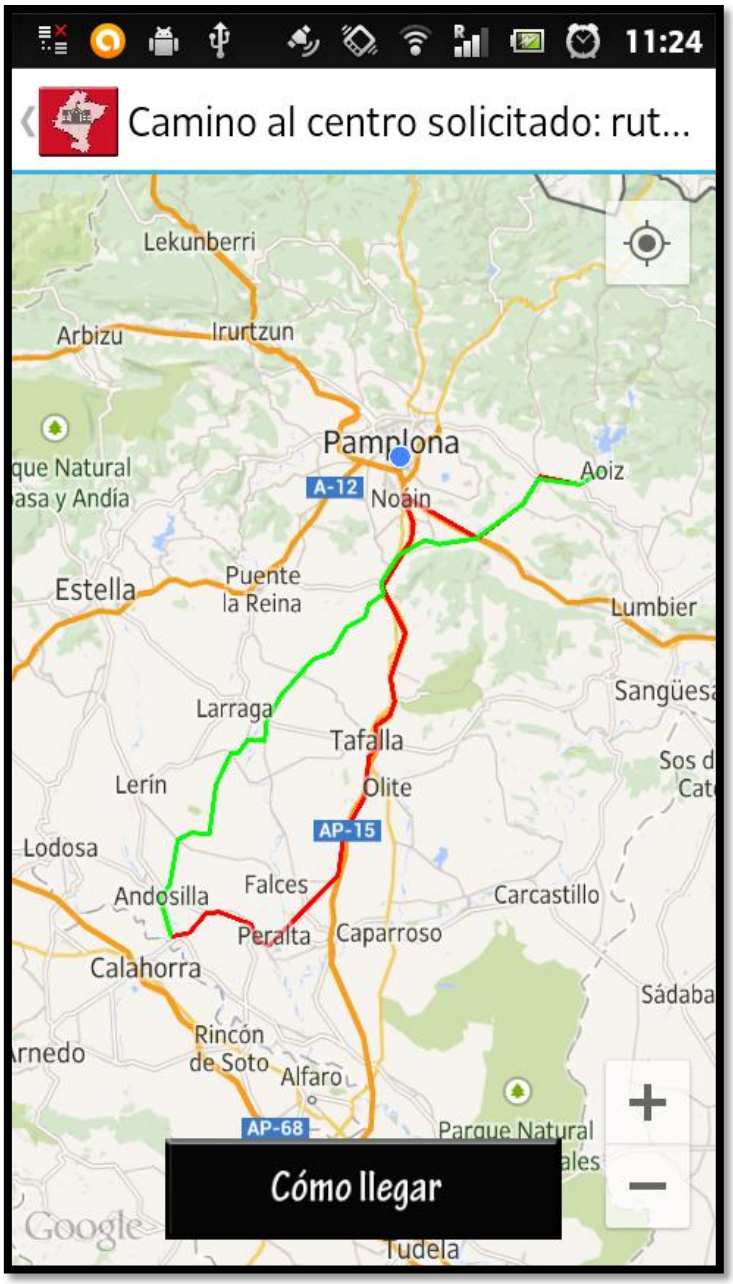


Figura 69: Ruta hacia el centro educativo

En primer lugar se inicializa el mapa al igual que hemos visto en la clase anterior MapaActivity.java. Lo siguiente que hacemos, es recoger la localización en la que se encuentre el teléfono mediante GPS.

```
googleMap.setMyLocationEnabled(true);
final Bundle bundle=getIntent().getExtras();
final double latitud = bundle.getDouble("latitud");
final double longitud = bundle.getDouble("longitud");
final double latitudini, longitudini;
if (googleMap.getMyLocation() != null) {
    latitudini = googleMap.getMyLocation().getLatitude();
    longitudini =googleMap.getMyLocation().getLongitude();
}
```

Figura 70: Localización actual del teléfono

Si nos encontramos en algún lugar cerrado, no podremos obtener la localización GPS actual, por lo que tendremos que consultar la última que tiene almacenada el teléfono, como vemos a continuación:

```
LocationManager lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
List<String> providers = lm.getProviders(true);

/* Loop over the array backwards, and if you get an accurate location, then break out the loop*/
Location l = null;

for (int i=providers.size()-1; i>=0; i--) {
        l = lm.getLastKnownLocation(providers.get(i));
        if (l != null) break;
}

if (l != null) {
        latitudini = l.getLatitude();
        longitudini = l.getLongitude();
}
```

Figura 71: Última localización disponible

Ahora que ya tenemos las posiciones de origen y destino, llamamos a la clase HacerDireccion.java, que es la que se encarga de dibujar el camino.

```
HacerDireccion md = (HacerDireccion) new HacerDireccion().execute(Double.toString(fromPosition.latitude), Double.toString(fromPosition.longitude),

Double.toString(toPosition.latitude), Double.toString(toPosition.longitude), HacerDireccion.MODE_DRIVING);
```

Figura 72: Llamada al método HacerDireccion

Dentro de la clase HacerDireccion.java, al margen de unas funciones auxiliares, la más importante es la que se encarga de hacer la petición de la ruta a GoogleMaps para poder dibujarla. Se hace mediante una tarea asíncrona para ganar en rapidez.

```
protected String doInBackground(String... params) {
    ready = false;

        String url = "http://maps.googleapis.com/maps/api/directions/xml?"
                + "origin=" + params[0] + "," + params[1]
                + "&destination=" + params[2] + "," + params[3]
                + "&sensor=false&units=metric&mode="+params[4];

        try {
            HttpClient httpClient = new DefaultHttpClient();
            HttpContext localContext = new BasicHttpContext();
            HttpPost httpPost = new HttpPost(url);
            HttpResponse response = httpClient.execute(httpPost, localContext);
            InputStream in = response.getEntity().getContent();
            DocumentBuilder builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
            doc = builder.parse(in);
            ready = true;
        } catch (Exception e) {
            Log.d("getdocument", e.toString());
        }
      return null;
    }
```

Figura 73: Petición de la ruta a GoogleMaps

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

3.3 – Pruebas unitarias

3.3.1 – Logs

La mayor parte de las pruebas unitarias se realizan en la programación, a través de trazas, que en Android son Logs. Es algo similar a hacer System.out en java. Eclipse nos proporciona una ventana de Logcat donde se nos muestran todos los logs conforme el usuario va pasando por diferentes lugares de nuestra aplicación.

En Android, todos los mensajes de log llevarán asociada la siguiente información:

- Fecha/Hora del mensaje.
- Criticidad. Nivel de gravedad del mensaje.
- PID. Código interno del proceso que ha introducido el mensaje.
- Tag. Etiqueta identificativa del mensaje.
- Mensaje. El texto completo del mensaje.

De forma similar a como ocurre con otros frameworks de logging, en Android los mensajes de log se van a clasificar por su criticidad, existiendo así varias categorias (ordenadas de mayor a menor criticidad):

1. Error (Color rojo)
2. Warning (Color amarillo)
3. Info (Color verde)
4. Debug (Color azul)
5. Verbose (Color negro)

En la siguiente Figura 74 vemos un ejemplo de los logs.

Figura 74: Ejemplo de Logs

En nuestra aplicación, he usado los logs en numerosas ubicaciones, por ejemplo, en los parsers, como se ve en la siguiente figura.

```
try{
String name = parser.getName();
Log.d("entra -2", parser.getName());
if (name.equals("return")) {
    //parser.nextToken();
    parser.nextTag();
    Log.d("entra -1", parser.getName());
    //Log.d("entra -0.5", parser.getName());
    String tag_clave = parser.getName();
    if (tag_clave.equals("clave")){
    Log.d("entra0",parser.getName());
```

Figura 75: Logs en ParserInfoAdicional

Alguna de las salidas que se obtienen son las siguientes.

```
D   10-31 17:49:4...   4985   4985                    entra -2        return
D   10-31 17:49:4...   4985   4985                    entra -1        clave
D   10-31 17:49:4...   4985   4985                    entra0          allo
D   10-31 17:49:4...   4985   4985                    6               61
```

Figura 76: Salida de algunos Logs.

3.4.2 – Junit

También es importante hacer pruebas de las funciones en nuestra aplicación, así comprobaremos que cada uno de los módulos que tenemos por separado funciona correctamente.

Para ello, utilizaremos el Android Test Project, que es la plataforma que proporciona Android para pruebas. Este test está basado en el que hace Junit pero adaptado a Android.

Creamos un proyecto de pruebas, y lo asociamos al de la aplicación móvil y haremos las operaciones sobre las funciones que queremos comprobar y ver los resultados.

Creamos los objetos y mediante las aserciones nos cercioramos que los resultados obtenidos son los esperados.

JUnit nos da el visto bueno de que las tres funciones funcionan correctamente.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

4 – Costes y planificación

4.1 – Planificación

En la siguiente tabla se detalla la planificación que se ha llevado en el desarrollo de la aplicación.

Para el desarrollo de la aplicación se ha seguido el modelo de la metodología ágil SCRUM, como ya se ha comentado.

En la primera etapa se hizo el diseño de la aplicación y un primer prototipo con las funcionalidades más básicas.

En la segunda iteración se añadieron más opciones a la aplicación como son el buscador avanzado y la ruta hacia el centro educativo.

Finalmente hubo que añadir una tercera iteración para adaptar la aplicación a unas mejoras sugeridas por el personal del Gobierno de Navarra.

Al final, se ha necesitado un periodo de aproximadamente un mes para escribir toda esta memoria del proyecto.

PLANIFICACIÓN		
Etapa	**Fecha comienzo**	**Fecha fin**
Diseño de la aplicación	26/6/2014	10/7/2014
Primera iteración	11/7/2014	1/8/2014
Segunda iteración	2/8/2014	29/8/2014
Pruebas	30/8/2014	31/8/2014
Tercera iteración	1/9/2014	1/10/2014
Documentación	1/10/2014	31/10/2014

4.2 - Costes

Se han estimado los costes con las fechas arriba descritas, estimando que un analista junior cobre 18€/hora, y un programador junior 12€/hora, con, en este caso, jornadas de 6 horas al día.

	Horas	**€/hora**	**Total**
Analista junior	100	18	1800€
Programador junior	350	12	4200€
Total	450		6000€

5 – Manual de usuario

En este apartado se procederá a mostrar la funcionalidad completa de la aplicación con ayuda de unas capturas de pantallas de la misma realizadas en una Tablet BQ Edison 2.

5.1- Pantalla principal e idioma:

Como ya hemos dicho, la aplicación se instalará con el nombre de Buscador de Centros Educativos de Navarra si tenemos el terminal en castellano (como vemos en la captura de pantalla bajo estas líneas, con el segundo icono de la fila de abajo empezando por la izquierda) o como Nafarroako Ikastetxeen Bilatzailea si lo tenemos en euskera.

El icono que he creado para la aplicación consta de la imagen de un centro educativo dentro de un mapa de Navarra con el fondo de la bandera de la comunidad foral.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Figura 77: Icono aplicación

La aplicación necesita para su ejecución conexión a internet. Sin conexión, pulsaremos aceptar y la aplicación se cerrará.

Figura 78: Pantalla conexión

Si por el contrario, como es normal en todos los Smartphones disponemos de conexión a internet, la aplicación arrancará normalmente y nos mostrará la siguiente imagen, que es la pantalla principal de la aplicación en la que se ofrecen cuatro opciones, que iremos viendo en las siguientes páginas, y que serán:

- Ver el listado completo de centros
- Ir al Buscador avanzado de centros
- Establecer la ruta a un centro educativo
- Cambiar el idioma de la aplicación

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

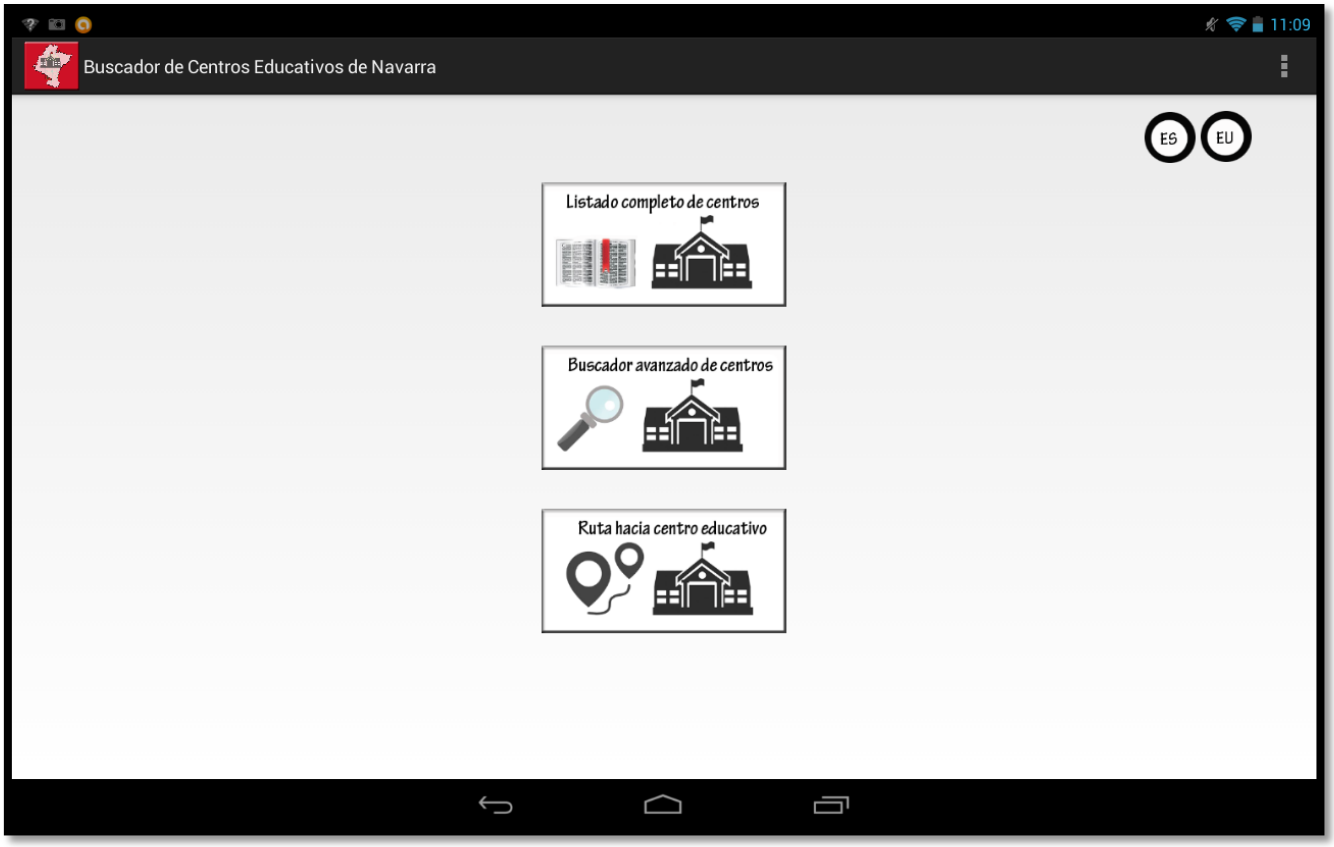


Figura 79: Pantalla principal

Empezando por la última de las opciones, la del cambio de idioma, cambiaremos desde los botones de la pantalla principal el idioma de la aplicación cuando lo deseemos. Como ya he comentado, actualmente la aplicación está en euskera y castellano pero fácilmente se podría adaptar a cualquier otro idioma que se estimara oportuno.

Para cambiar el idioma usaremos los botones en la parte superior derecha de la pantalla, el botón ES para ponerla en español y el botón EU para ponerla en euskera.

Como ya hemos visto la pantalla principal de la aplicación en español, si ahora elegimos euskera la veremos de la siguiente manera:

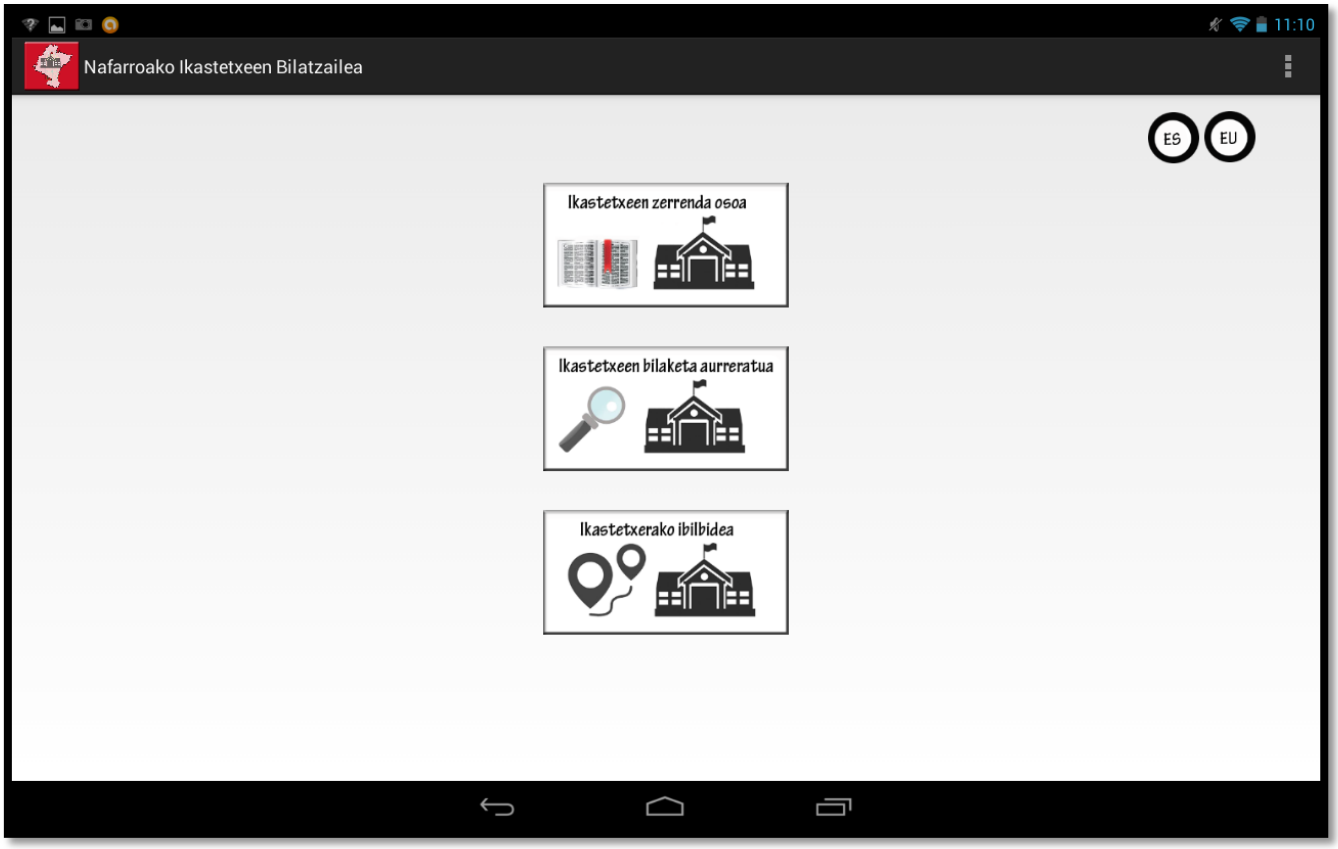


Figura 80: Pantalla principal euskera

Como podemos observar, la información y las opciones son exactamente iguales que en español.

5.2 - Listado completo de centros:

La primera opción del menú será la que veamos ahora, que es la del listado completo de centros. Como su propio nombre indica, se listarán todos los centros educativos almacenados en la información del Gobierno de Navarra de la siguiente manera, en modo lista:

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

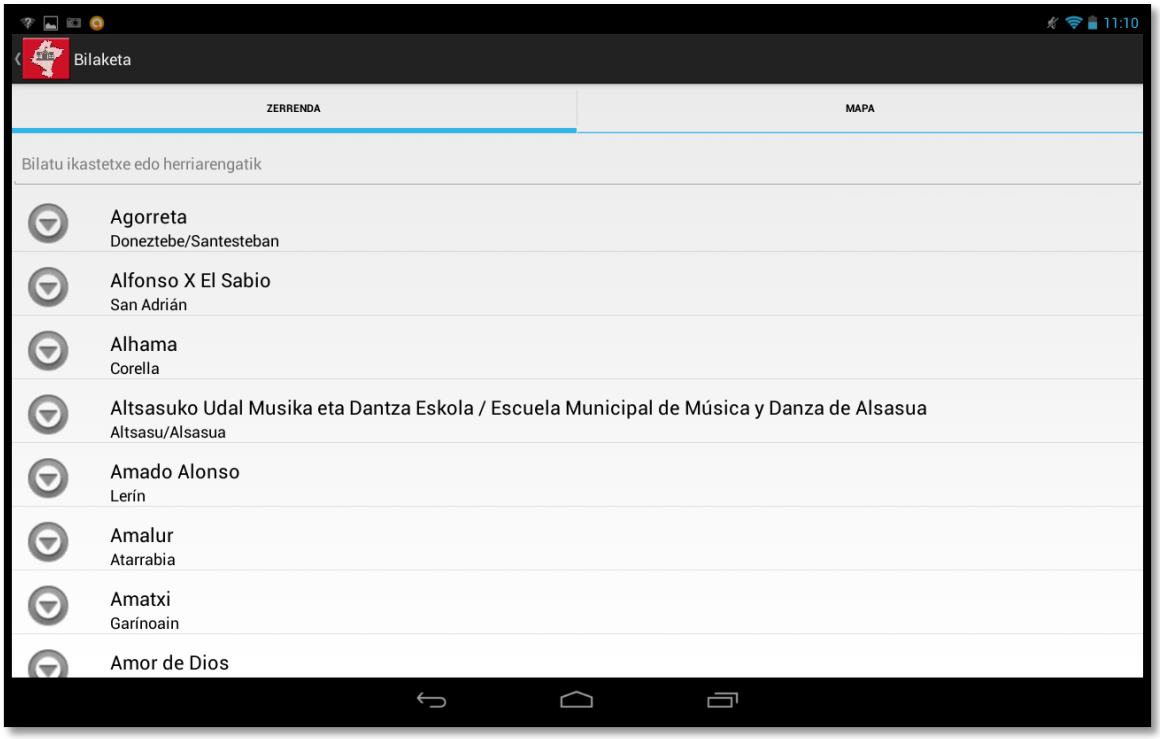


Figura 81: Listado de centros

Como se puede observar, vemos un listado completo de los centros, de los cuales se puede ver el nombre del centro y la localidad en la que están. En la parte superior de la pantalla, podemos escribir en la barra y hacer una búsqueda por nombre del centro o localidad. Al escribir Aoiz en la barra veremos lo siguiente:

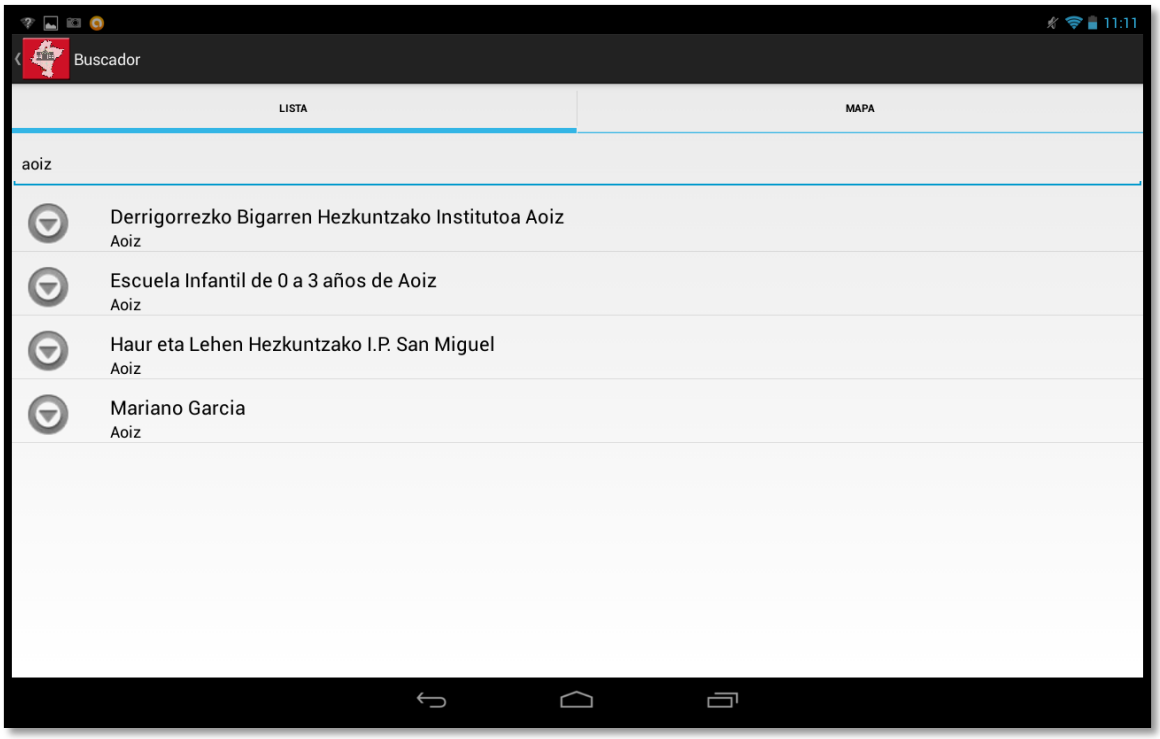


Figura 82: Listado de centros de Aoiz

Se nos mostrarán los centros educativos que coinciden con la búsqueda Aoiz.

Si pulsamos sobre el nombre de alguno de los centros, veremos su información detallada. También, podremos ver el resultado de la búsqueda en modo mapa haciendo click sobre la pestaña Mapa en la pantalla anterior, de la siguiente manera:

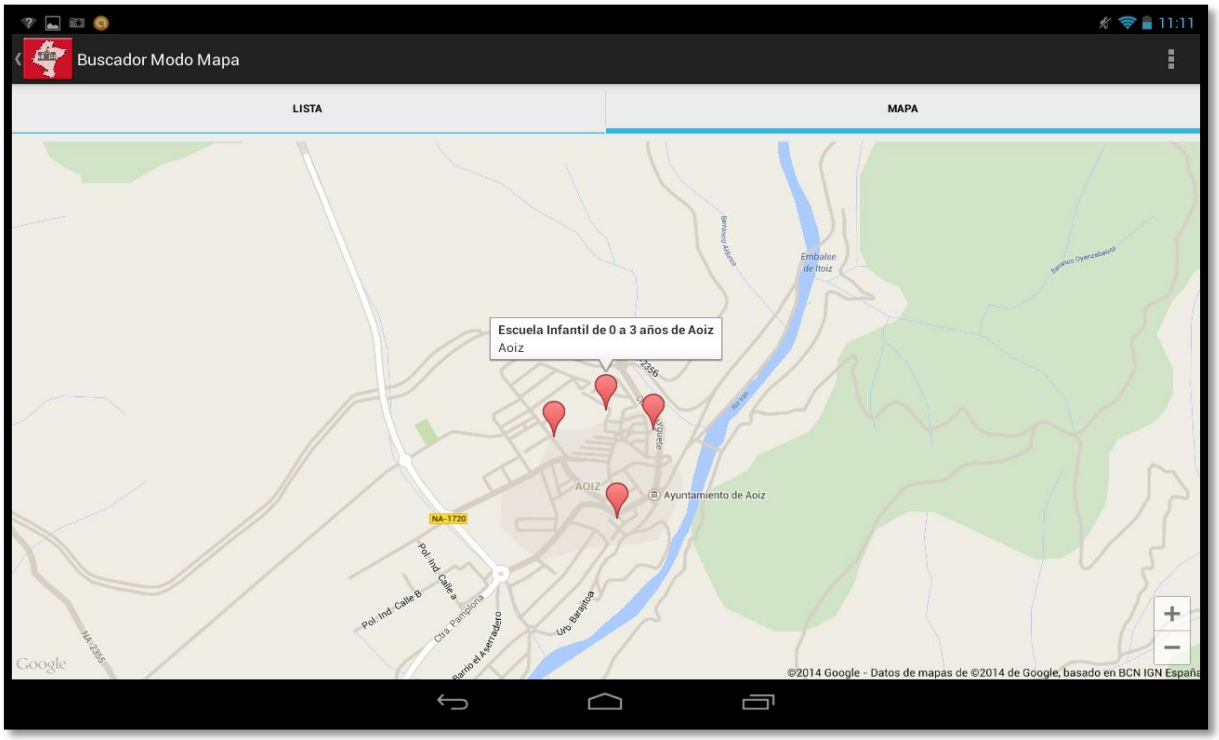


Figura 83: Listado de centros de Aoiz modo mapa

Al pulsar sobre la marca de color rojo de alguno de los centros, se nos mostrará el nombre del centro y su localidad. Al pulsar sobre el cuadro en el que aparece dicha información, veremos la información detallada del centro como en las próximas páginas se detalla.

Al tocar sobre el nombre de alguno de los centros educativos, podremos ver información ampliada sobre él, como se muestra a continuación con el centro Alfonso X el Sabio de San Adrián.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

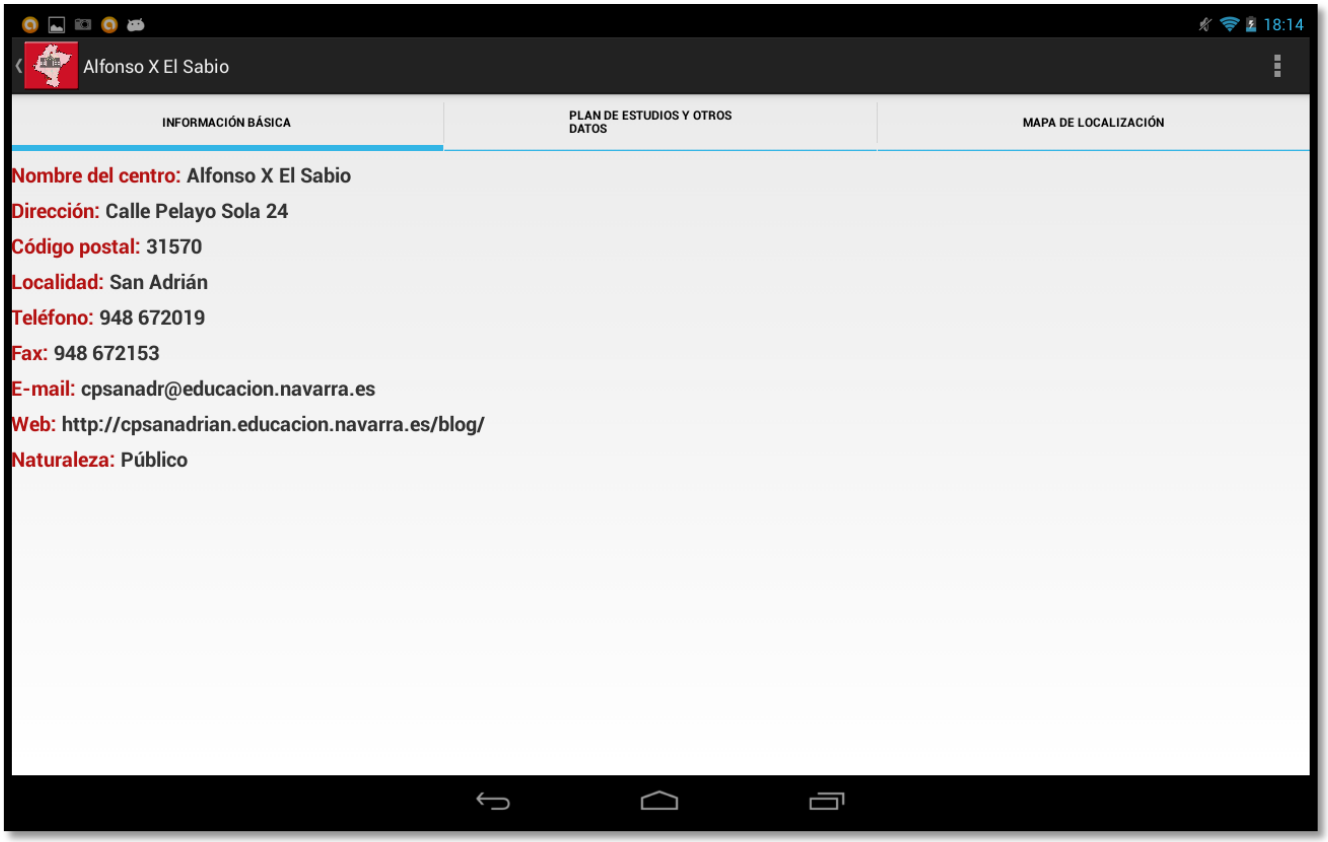


Figura 84: Información básica centro

En principio se desplegará la pestaña Información básica, en la que podremos consultar el nombre, dirección, código postal, localidad, teléfono, fax, e-mail, web y naturaleza del centro.

Al seleccionar la segunda pestaña Plan de estudios y otros datos, veremos la información referente a, como vemos en las dos siguientes capturas de pantalla: modelos lingüísticos, tipo de enseñanza, servicios complementarios, atención nee, programas, plazas 3 años, plan de estudios, enlace a Sitna, número de matrículas, objetivos educativos, valores del centro, distinciones, jornada laboral, proyectos, reconocimientos y zona lingüística a la que pertenece.

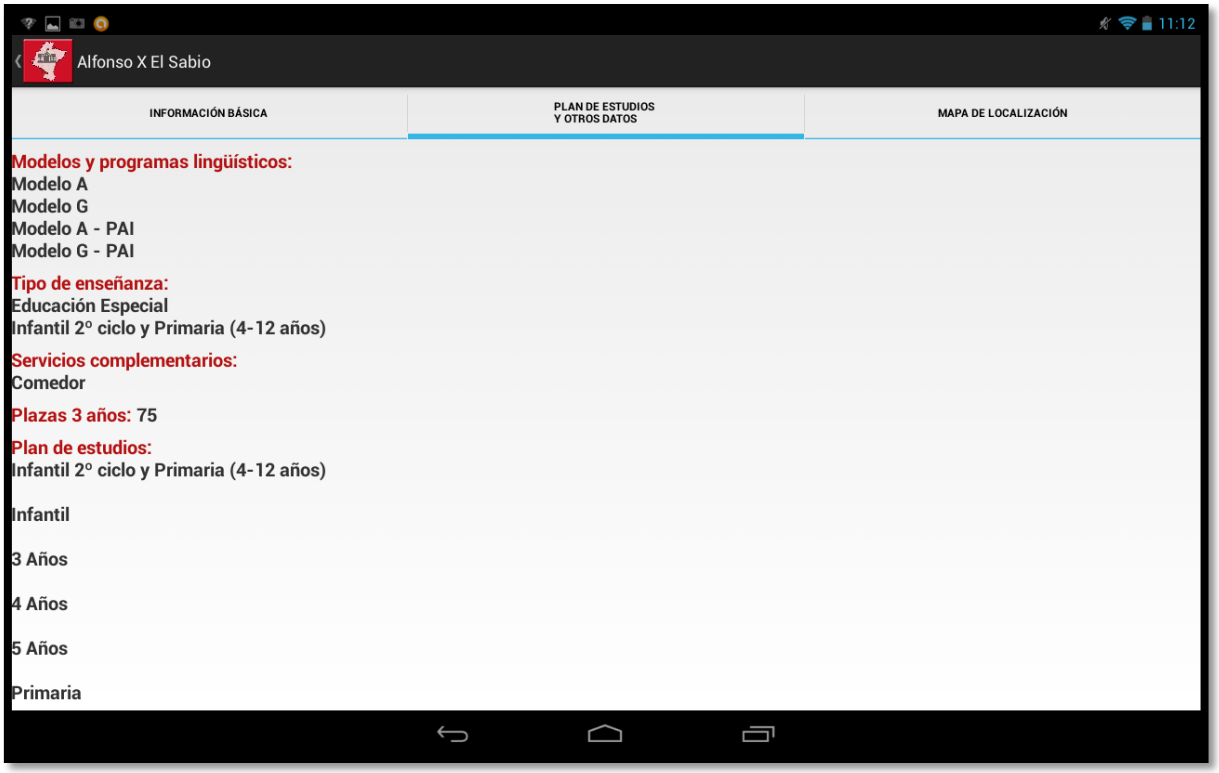


Figura 85: Información adicional centro

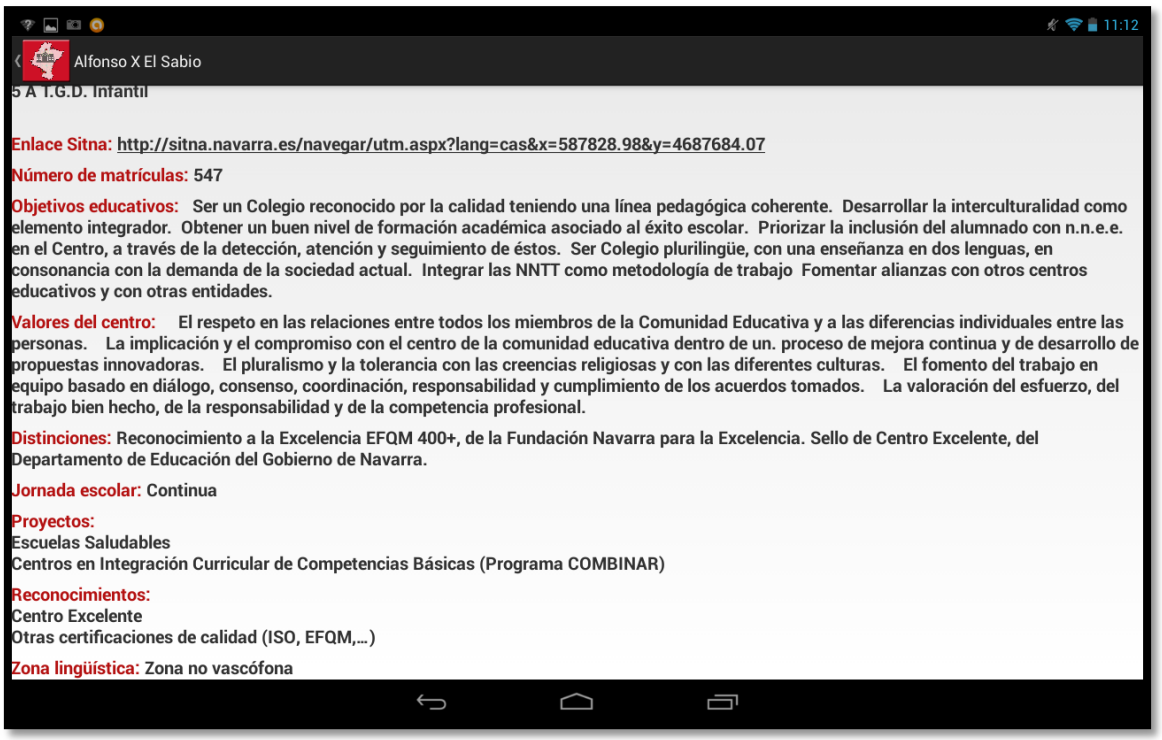


Figura 86: Información adicional centro

Si seleccionamos la tercera pestaña, veremos en un mapa con la localización del centro:

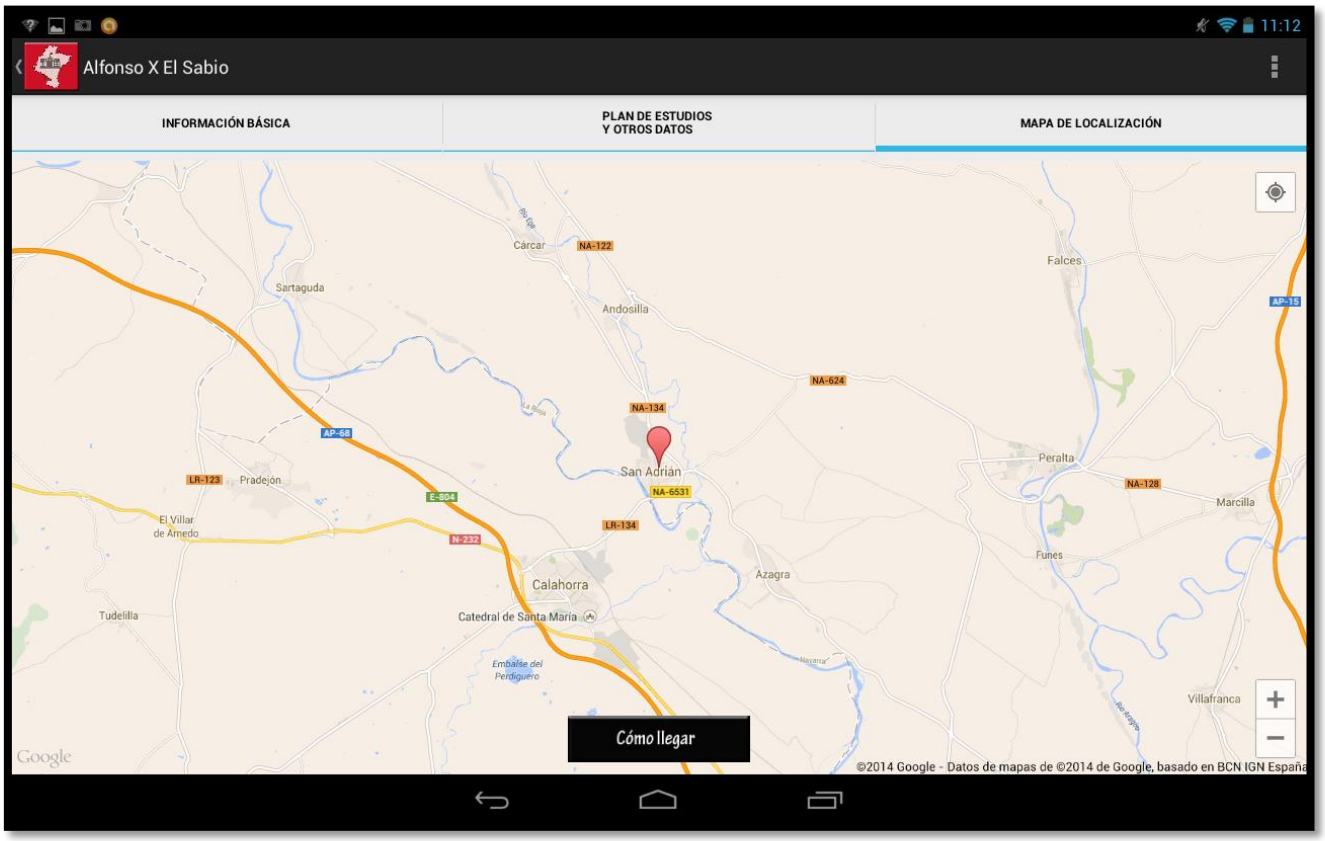


Figura 87: Localización centro

En la parte inferior de la imagen, tenemos el botón “Cómo llegar”, el cual, cuando sea pulsado nos mostrará información detallada de cómo llegar desde nuestra ubicación actual al centro educativo en cuestión. Al ser pulsado, se abrirá la aplicación Google Maps, que nos mostrará lo siguiente:

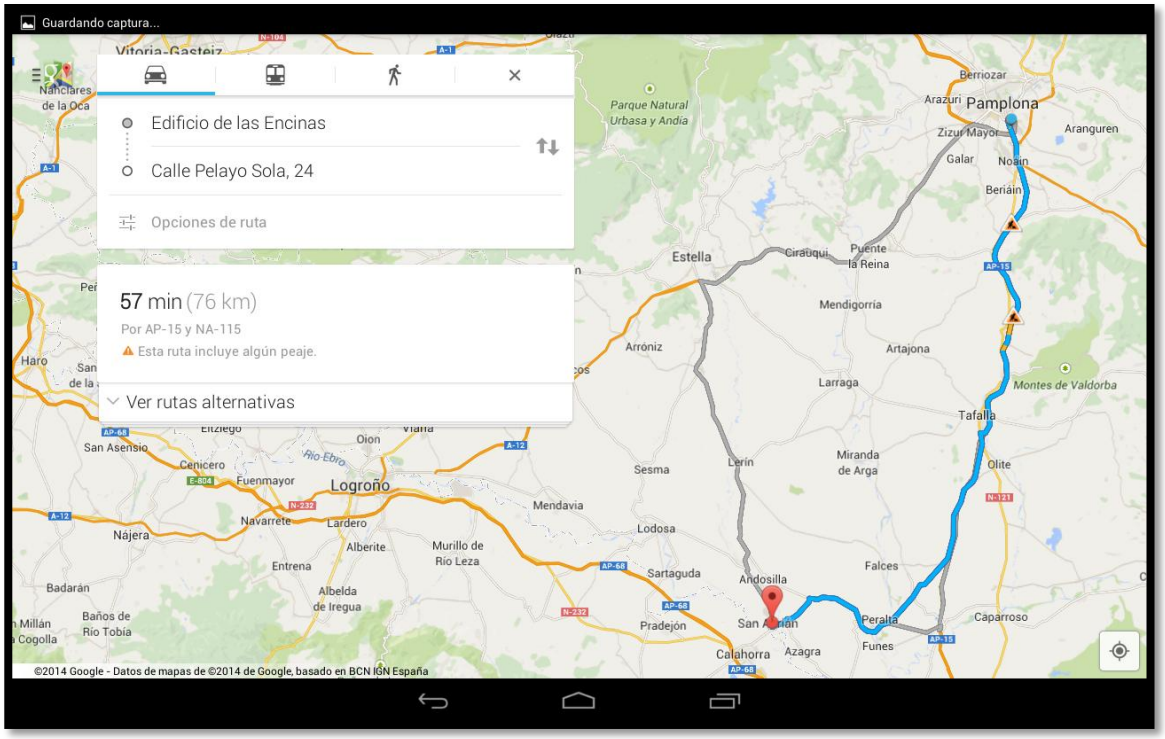


Figura 88: Cómo llegar al centro

Podremos elegir si queremos ir en coche, transporte urbano o caminando, y podremos abrir el modo navegación para que nos muestre la ruta hacia el centro educativo paso a paso, en este caso, entre la Universidad Pública de Navarra, situada en Pamplona y el centro Alfonso X el Sabio situado en San Adrián.

5.3 - Buscador avanzado de centros:

Si en el menú principal seleccionamos la opción de buscador avanzado de centros, podremos hacer una búsqueda avanzada especificando diferentes características de los centros educativos.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

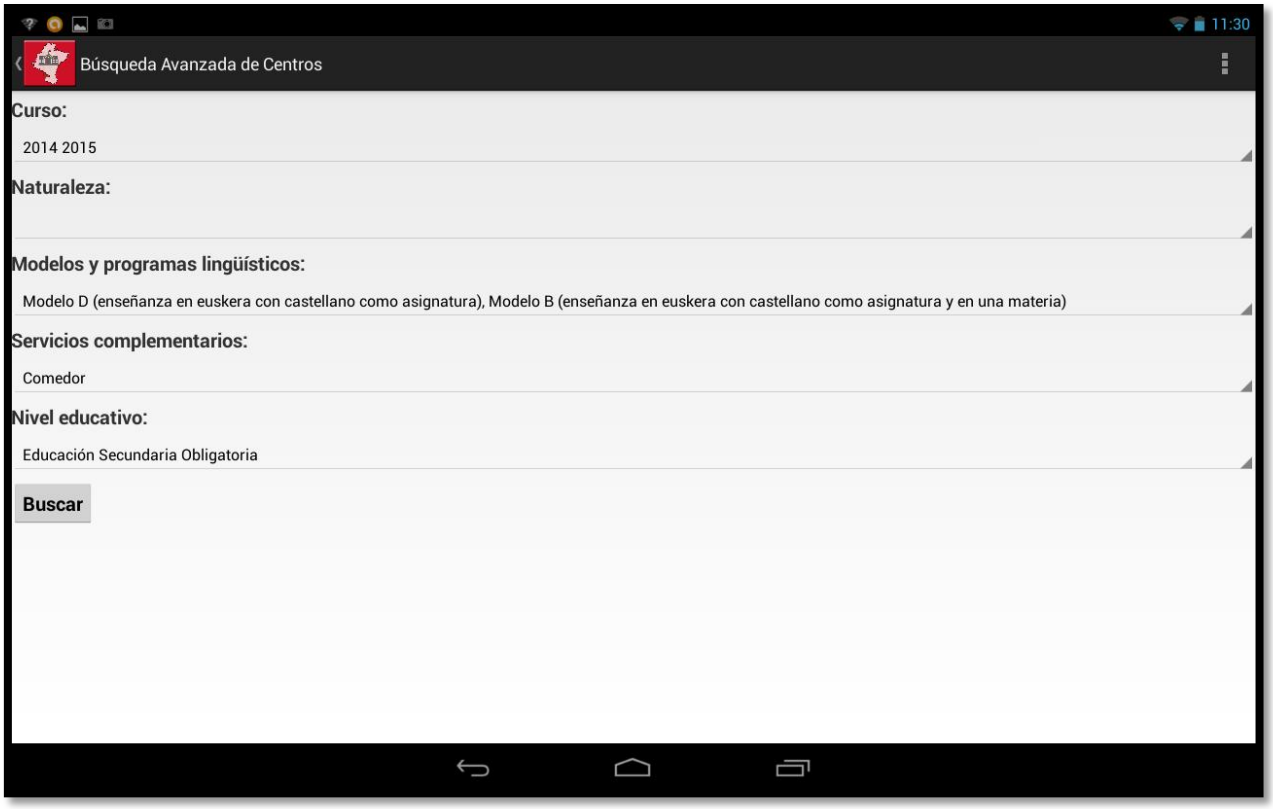


Figura 89: Búsqueda avanzada centro

Como vemos en la imagen, podremos especificar el curso del que queremos ver la información, la naturaleza del centro, los modelos lingüísticos, servicios complementarios y nivel educativo.

En cada una de las características tendremos numerosas opciones, por ejemplo, referentes al modelo lingüístico:

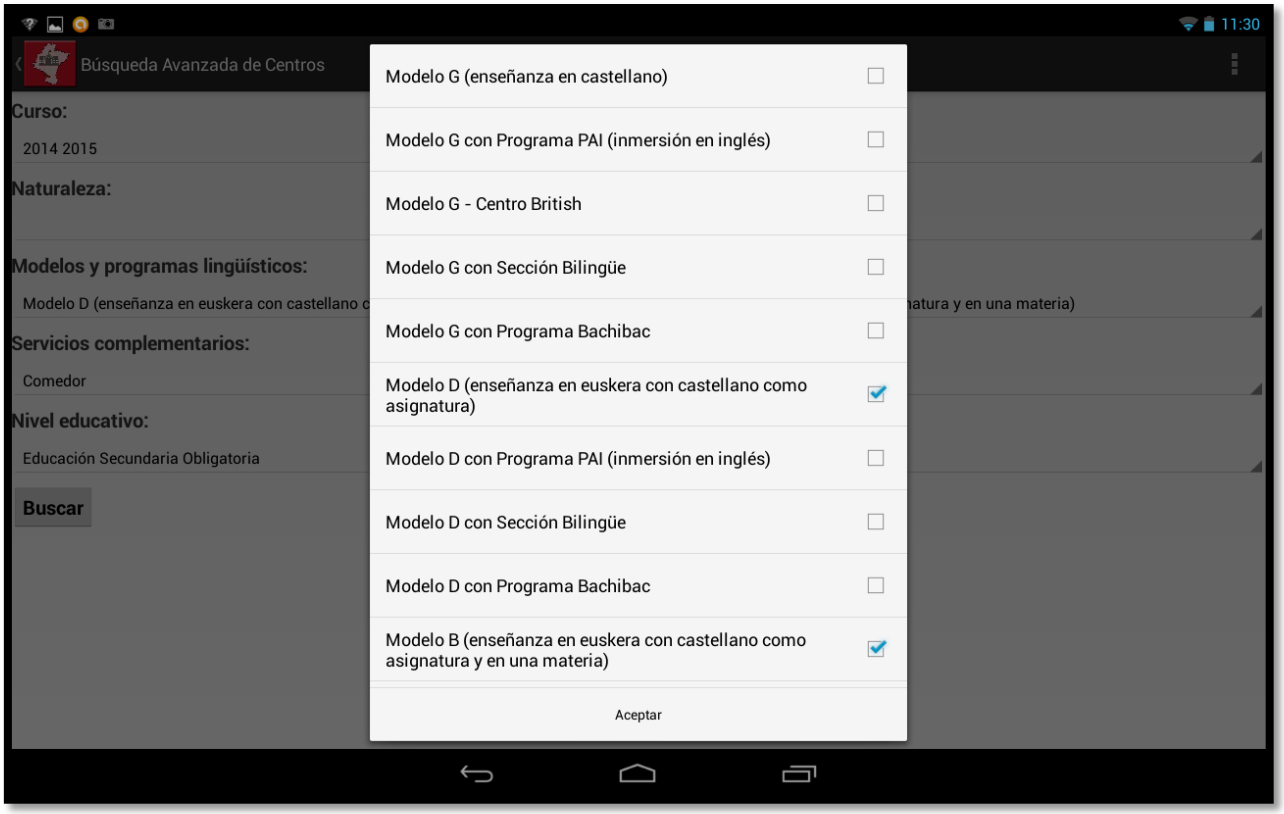


Figura 90: Modelos lingüísticos centros

Cabe reseñar que se puede hacer selección múltiple en todos los campos. Por ejemplo, especificando la búsqueda de arriba que es información del curso 2014-15 para modelos D o B, con servicio de comedor y en los cuales hay Educación Secundaria Obligatoria veremos los siguientes centros:

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

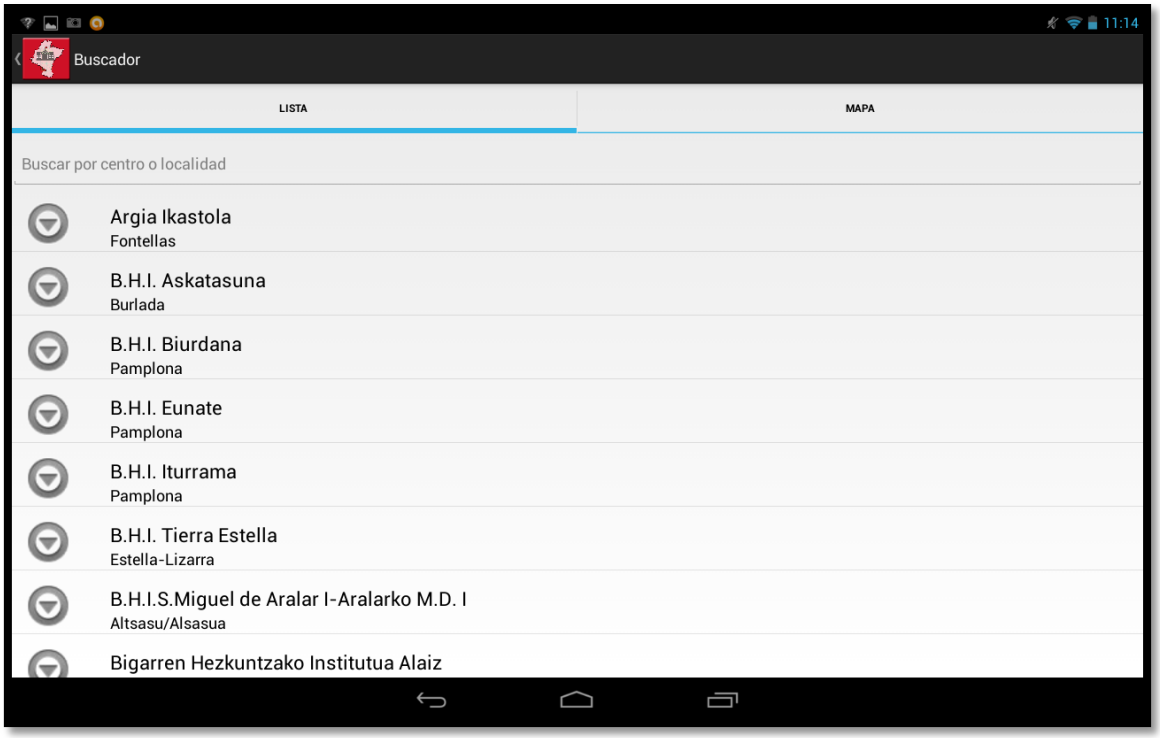


Figura 91: Resultado búsqueda avanzada

Seleccionando alguno de los centros podríamos ver su información ampliada como ya se ha enseñado anteriormente en este mismo documento. En esta pantalla se puede utilizar el buscador por palabra de centro o localidad que ya se ha comentado.

5.4 -Ruta hacia centro educativo:

La última opción que falta por detallar es la tercera del menú principal de la aplicación que es la de ruta hacia el centro educativo. Al acceder a ella se nos mostrarán el listado de todos los centros educativos que hemos visto anteriormente con el buscador por nombre de centro o localidad. En el momento en el que seleccionemos un centro, el dispositivo Android recogerá nuestra localización en ese momento o la última a la que pudo acceder en caso de tener el GPS apagado o encontrarnos dentro de un edificio y nos mostrará la mejor ruta en coche hacia el centro solicitado en color rojo y la mejor ruta andando en color verde.

Así es como veremos el camino que tendremos que hacer desde nuestra ubicación al centro en cuestión, en color rojo la mejor en coche, y en verde la mejor andando. En este caso, desde la Universidad Pública donde estoy ubicado yo hasta el centro Andrés Muños Garde de Pamplona.

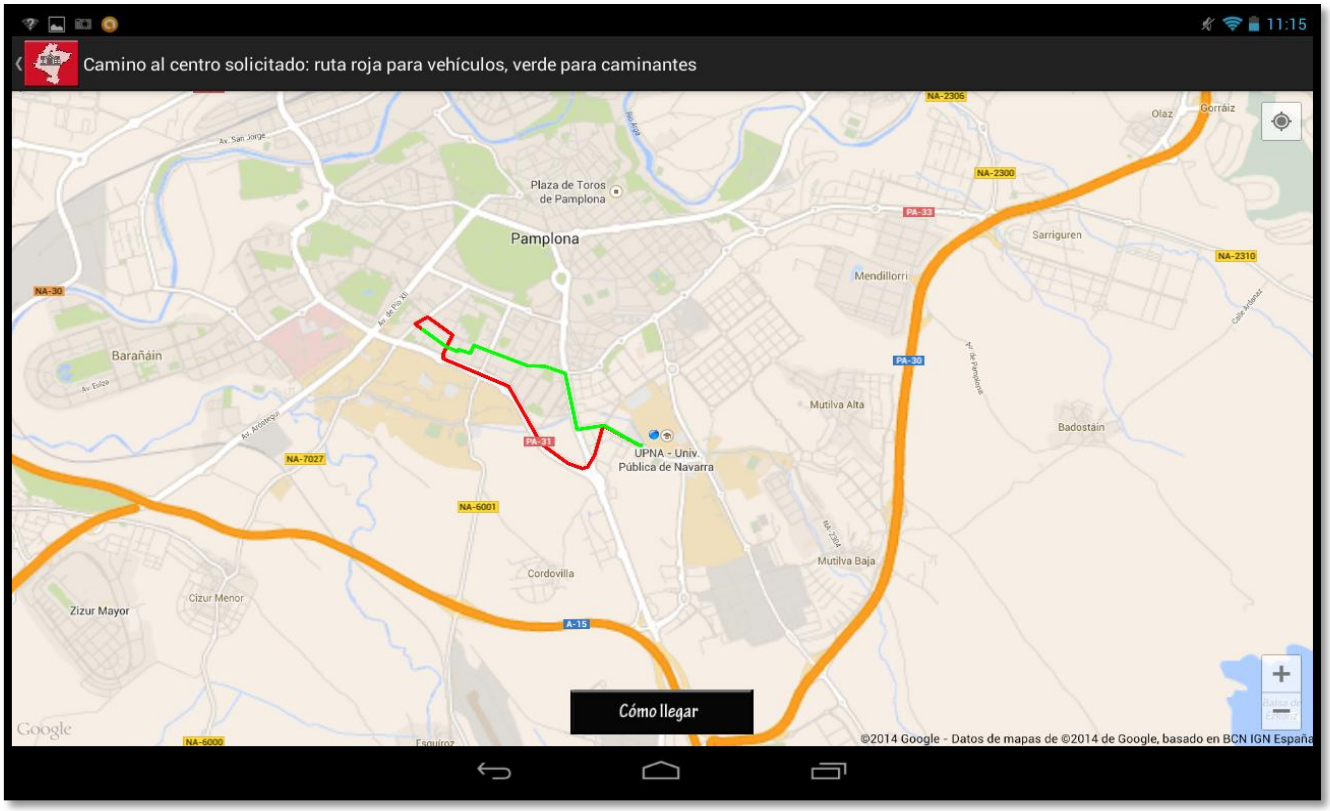


Figura 92: Ruta hacia centro educativo

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

6 – Conclusiones y líneas futuras

Como conclusión, se puede decir que se ha conseguido realizar una aplicación de gran utilidad para el usuario que quiera ser informado de forma bilingüe a cerca de detalles de los centros educativos de Navarra, tanto detalles sobre los centros (dirección, plan de estudios, localización, etc.) así como hacer búsqueda avanzadas sobre ellos y facilitar información para llegar a todos los centros.

El proyecto ha resultado todo un éxito al resultar ganador del I Premio de Diseño de Aplicaciones para Teléfonos Inteligentes (Smartphones) con datos abiertos del Departamento de Educación del Gobierno de Navarra.[5]

Asimismo, he aprendido también a desarrollar un proyecto informático, los imprevistos que pueden surgir en el desarrollo de ellos y resolver los problemas que van surgiendo. Por ejemplo, la declaración de las funciones que se establecía en el API no coincidía al 100% con la implementación real de ellas, y por ello hubo algunos imprevistos que se superaron con éxito.

Para la realización de este Proyecto Fin de Carrera, han sido necesarias muchas horas de trabajo, análisis, diseño, investigación, así como el apoyo de mi tutor en la resolución de algunos problemas.

Algunas de los aspectos en los que se podrían seguir trabajando como pasos futuros en este Proyecto Fin de Carrera serían los siguientes:

- Realizar una especie de red social por centros, eligiendo de qué centro eres y poder hacer un chat entre centros, con posibles juegos y ranking entre centros…
- Realizar búsquedas por proximidad a un centro educativo.
- Ofrecer opciones de administrador para poder cambiar los detalles de un centro desde la propia aplicación para una persona autorizada.
- Al hilo de la red social sacar si tus contactos están utilizando la aplicación, a qué centro pertenecen…

[5] http://www.navarra.es/home_es/Actualidad/BON/Boletines/2014/204/Anuncio-8/

7 – Bibliografía

[1] Directorio de Centros web: http://www.educacion.navarra.es/web/dpto/centros-educativos

[2] API Directorio de Centros: http://educages.navarra.es/xwiki/bin/view/Main/Directorio+de+centros

[3] Ejemplos ksoap: https://code.google.com/p/ksoap2-android/wiki/HowToUse?tm=2

[4] Anuncio convocatoria: http://www.navarra.es/home_es/Servicios/ficha/5356/I-Premio-de-diseno-de-aplicaciones-para-telefonos-inteligentes-(smartphones)#documentacion

[5] API Desarrollador de Android: http://developer.android.com

[6] IDE Eclipse: http://www.eclipse.org

[7] Foro de programadores: http://stackoverflow.com/

[8] Tutoriales Vogella: http://www.vogella.com/android.html

[9] Tutoriales Androideity: http://androideity.com/

[10] Tutoriales Android ya: http://www.javaya.com.ar/androidya/

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa
Todos los derechos reservados
Eskubide guztiak erresalbatu dira

Aplicación Android Buscador de Centros Educativos de Navarra

Autor: Mikel San Martín Huarte

Tutor: Óscar Ardaiz Villanueva

Pamplona, a 10 de noviembre de 2014

Introducción

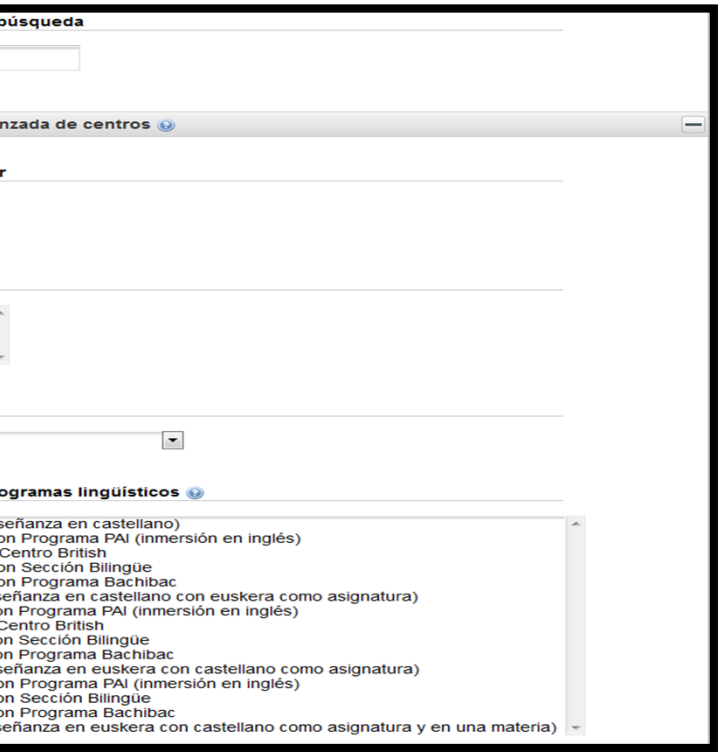
- Convocatoria Gobierno de Navarra boletín oficial nº 96
- I Premio de Diseño de Aplicaciones para Teléfonos Inteligentes (smartphones) con datos abiertos del Departamento de Educación.
- Aplicación Android bilingüe para acercar al ciudadano información sobre los centros educativos navarros de manera atractiva y sencilla.

Objetivos

- Elaborar una aplicación amigable y fácil de usar para acercar la información de los centros educativos navarros a todos los ciudadanos de la comunidad.
- Implementación en Android el Directorio de Centros Web del Gobierno de Navarra.
- La aplicación debe permitir hacer una búsqueda al ciudadano entre todos los centros ya sea por su nombre o por la localidad en la que se encuentra.
- El usuario tendrá una opción de búsqueda avanzada en la que podrá especificar el curso educativo, la naturaleza del centro, el modelo lingüístico, servicios complementarios o nivel educativo.

Sistema actual

- Directorio de Centros Web del Gobierno de Navarra



Nombre	Localidad	Sitio web	
Esc. Mús. Púb. Doneztebe AGORRETA	Doneztebe/Santesteban		Más datos
CPEIP San Adrian	San Adrián		Más datos
IES Corella Alhama	Corella		Más datos
Esc. Mús. Púb. Altsasu	Altsasu/Alsasua		Más datos
Esc. Ed. Inf. Lerín Amado Alonso	Lerín		Más datos
Esc. Ed. Inf. Villava Amalur	Villava		Más datos
Esc. Ed. Inf. Garínoain Amatxi	Garínoain		Más datos
C.Con. Bur. Amor de Dios	Burlada		Más datos
C.Con. Bur. Amor - Regina	Burlada		Más datos
IK Etxarri-Ar. Andra Mari	Etxarri Aranatz		Más datos

Metodología de trabajo

- Metodología ágil SCRUM – Sprints de 4 semanas
- Primera iteración: Diseño de la aplicación y un primer prototipo con las funcionalidades más básicas.
- Segunda iteración: se añadieron más opciones a la aplicación como son el buscador avanzado y la ruta hacia el centro educativo.
- Finalmente hubo que añadir una tercera iteración para adaptar la aplicación a unas mejoras sugeridas por el personal del Gobier de Navarra.

Análisis y diseño del software

- Requisitos funcionales

- Bilingüe: la aplicación estará en castellano y euskera.
- Las tres opciones más importantes en la pantalla principal serán mostrar todos los centros, hacer una búsqueda avanzada o lograr el camino hacia un centro educativo.
- La aplicación debe ser totalmente intuitiva y sencilla para adaptarse a usuarios de todo tipo.
- En la pestaña mapa de localización veremos el interfaz de Google Maps con el punto marcado del centro educativo en cuestión.

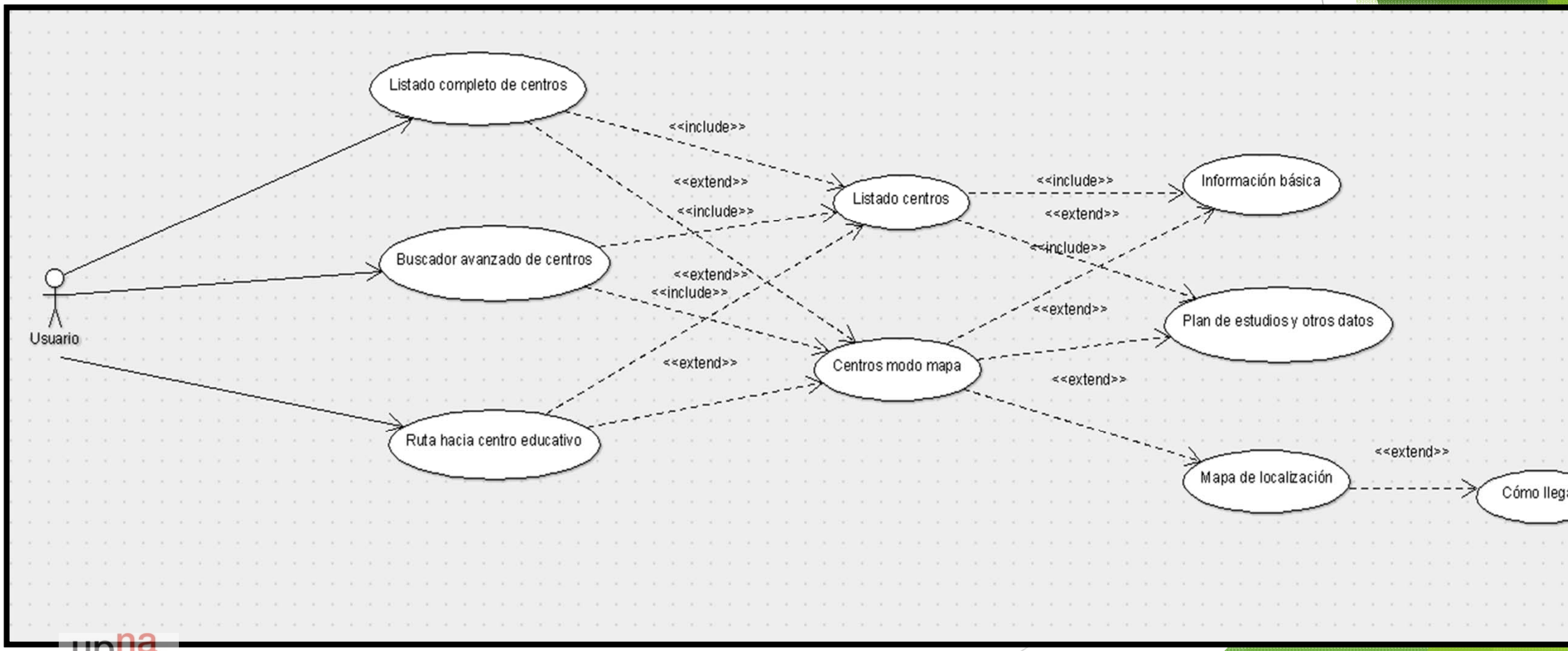
Análisis y diseño del software

- Requisitos no funcionales

- La aplicación es compatible con los sistemas Android entre el SDK 7 (Android 2.1) y el SDK 19 (Android 4.4) y se adaptará a dispositivos con diferentes tamaños.
- Para el intercambio de datos entre el dispositivo móvil y el servidor se usa una conexión segura y mediante el protocolo SOAP para interactuar con el WSDL ofrecido por el Gobierno de Navarra.
- Para el uso de SOAP en Android, se usará la librería externa ksoap2. La información XML recibida se parseará utilizando la clase XMLPullParser de Android.
- Se usan tareas asíncronas de Android para un funcionamiento más rápido y fluido de la aplicación.
- Para mostrar la localización de los centros educativos como las rutas se usará el API V2 de Google Maps.

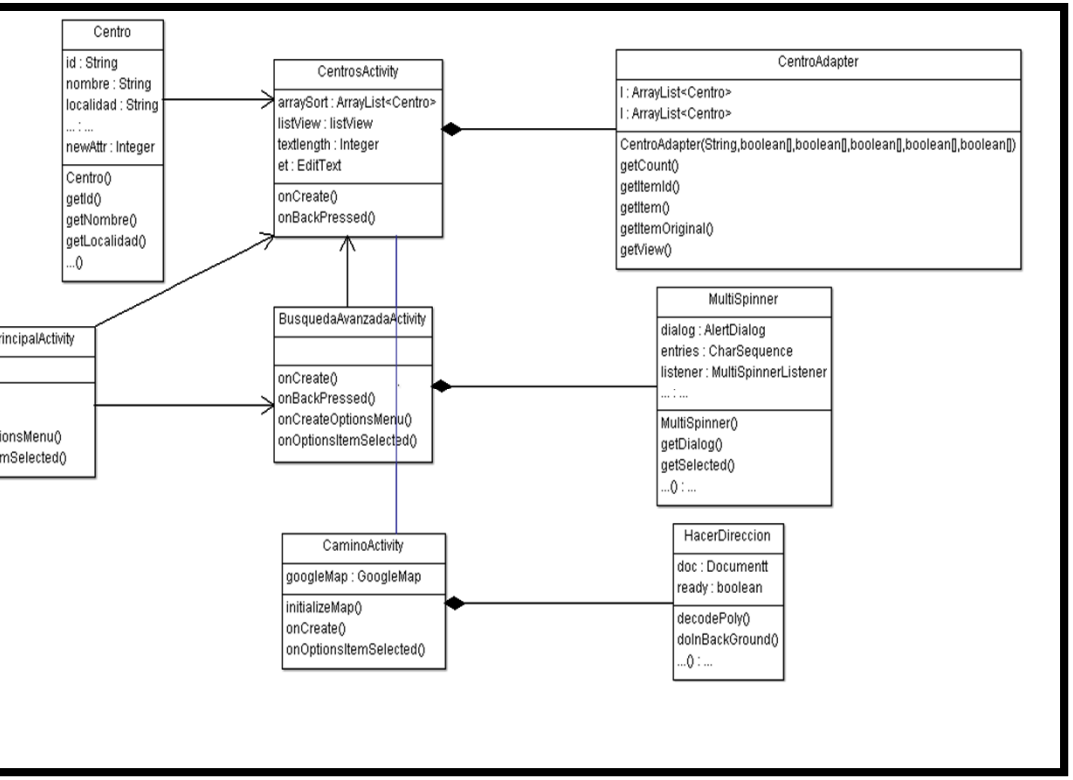
Análisis y diseño del software

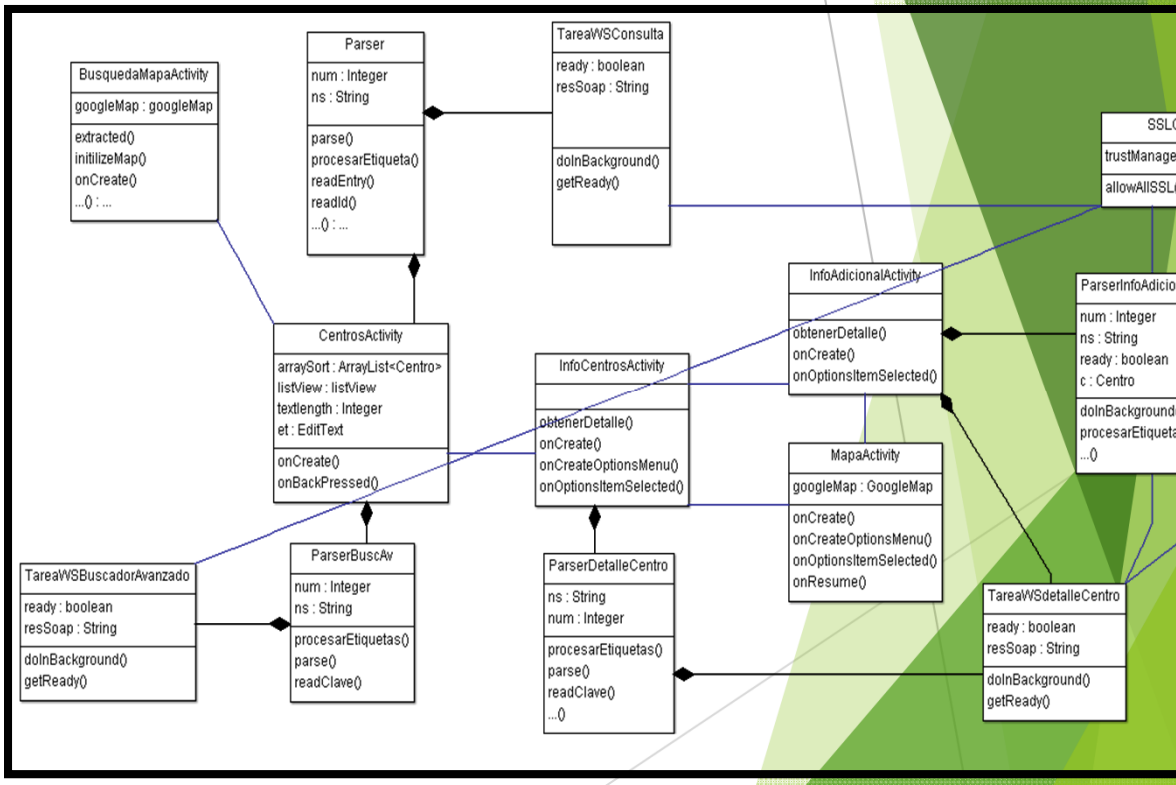
- Diagrama de casos de uso general



Análisis y diseño del software

- Diagrama de clases





Análisis y diseño del software

Imports

dapter	CentrosActivity
va.io.Serializable;	import java.io.Serializable;
va.util.List;	import java.util.ArrayList;
ndroid.util.Log;	import android.app.ProgressDialog;
ndroid.view.LayoutInflater;	import android.content.Context;
ndroid.view.View;	import android.content.Intent;
ndroid.view.ViewGroup;	import android.content.pm.ActivityInfo;
ndroid.widget.BaseAdapter;	import android.content.res.Resources;
ndroid.widget.TextView;	import android.graphics.PorterDuff;
	import android.os.Bundle;
	import android.support.v4.app.Fragment;
	import android.support.v4.app.NavUtils;
	import android.support.v7.app.ActionBarActivity;
	import android.text.Editable;
	import android.text.TextWatcher;
	import android.util.Log;
	import android.view.LayoutInflater;
	import android.view.View;
	import android.view.View.OnClickListener;

ctivity	Centros
va.util.ArrayList;	import java.io.ByteArrayInputStream;
va.util.List;	import java.io.InputStream;
va.util.Locale;	import java.io.Serializable;
rg.w3c.dom.Document;	import java.util.List;
ndroid.content.Context;	import Proceso_XML.Parser;
ndroid.content.Intent;	import Proceso_XML.ParserBuscAv;
ndroid.content.SharedPreferences;	import SOAP.TareaWSBuscadorAvanzado;
ndroid.content.pm.ActivityInfo;	import SOAP.TareaWSConsulta;
ndroid.content.res.Configuration;	import android.app.ProgressDialog;
ndroid.graphics.Color;	import android.util.Log;
ndroid.graphics.PorterDuff;	
ndroid.location.Location;	
ndroid.location.LocationManager;	
ndroid.net.Uri;	
ndroid.os.Bundle;	
ndroid.support.v7.app.ActionBarActivity;	
ndroid.util.Log;	
ndroid.view.MenuItem;	
ndroid.view.View;	
ndroid.view.View.OnClickListener;	
ndroid.widget.ImageView;	
ndroid.widget.Toast;	
om.google.android.gms.maps.CameraUpdateFactory;	
om.google.android.gms.maps.GoogleMap;	
om.google.android.gms.maps.SupportMapFragment;	
om.google.android.gms.maps.model.CameraPosition;	
om.google.android.gms.maps.model.LatLng;	
om.google.android.gms.maps.model.PolylineOptions;	

ccion	InfoAdicionalActivity
va.io.InputStream;	import java.io.ByteArrayInputStream;
va.util.ArrayList;	import java.io.InputStream;
vax.xml.parsers.DocumentBuilder;	import java.util.Locale;
vax.xml.parsers.DocumentBuilderFactory;	import Proceso_XML.ParserInfoAdicional;
g.apache.http.HttpResponse;	import SOAP.TareaWSdetalleCentro;
g.apache.http.client.HttpClient;	import android.content.Context;
g.apache.http.client.methods.HttpPost;	import android.content.Intent;
g.apache.http.impl.client.DefaultHttpClient;	import android.content.SharedPreferences;
g.apache.http.protocol.BasicHttpContext;	import android.content.pm.ActivityInfo;
g.apache.http.protocol.HttpContext;	import android.content.res.Configuration;
g.w3c.dom.Document;	import android.content.res.Resources;
g.w3c.dom.Node;	import android.net.Uri;
g.w3c.dom.NodeList;	import android.os.Bundle;
droid.os.AsyncTask;	import android.support.v7.app.ActionBarActivity;
droid.util.Log;	import android.text.Html;
m.google.android.gms.maps.model.LatLng;	import android.util.Log;
	import android.view.MenuItem;
	import android.view.View;
	import android.widget.TabHost;
	import android.widget.TabHost.OnTabChangeListener;
	import android.widget.TextView;

BusquedaAvanzadaActivity	BusquedaMapaActivity
import java.util.Locale;	import java.util.List;
import android.app.Dialog;	import java.util.Locale;
import android.content.Context;	import android.content.Context;
import android.content.Intent;	import android.content.Intent;
import android.content.SharedPreferences;	import android.content.SharedPreferences;
import android.content.pm.ActivityInfo;	import android.content.pm.ActivityInfo;
import android.content.res.Configuration;	import android.content.res.Configuration;
import android.os.Bundle;	import android.content.res.Resources;
import android.support.v4.app.Fragment;	import android.os.Bundle;
import android.support.v4.app.NavUtils;	import android.support.v4.app.Fragment;
import android.support.v7.app.ActionBarActivity;	import android.support.v7.app.ActionBarActivity;
import android.view.LayoutInflater;	import android.util.Log;
import android.view.Menu;	import android.view.LayoutInflater;
import android.view.MenuItem;	import android.view.Menu;
import android.view.View;	import android.view.MenuItem;
import android.view.View.OnClickListener;	import android.view.View;
import android.view.ViewGroup;	import android.view.ViewGroup;
import android.view.WindowManager;	import android.widget.TabHost;
import android.view.WindowManager.LayoutParams;	import android.widget.TabHost.OnTabChangeListener;
	import android.widget.TextView;
	import android.widget.Toast;
	import com.google.android.gms.maps.CameraUpdateFactory;
	import com.google.android.gms.maps.GoogleMap;
	import com.google.android.gms.maps.GoogleMap.InfoWindowAdapter;
	import com.google.android.gms.maps.GoogleMap.OnInfoWindowClickListene
	import com.google.android.gms.maps.SupportMapFragment;
	import com.google.android.gms.maps.model.BitmapDescriptorFactory;
	import com.google.android.gms.maps.model.CameraPosition;
	import com.google.android.gms.maps.model.LatLng;

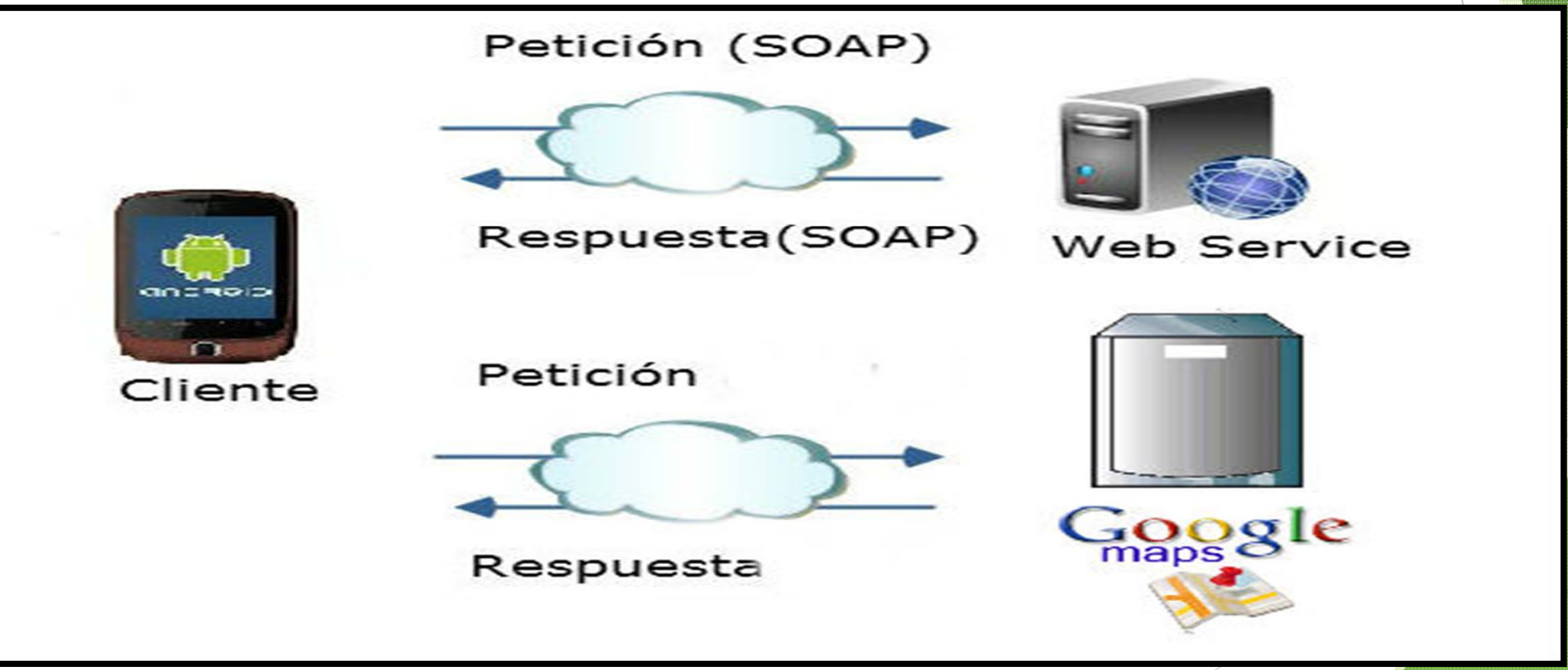
InfoCentrosActivity	MapaActivity
import java.io.ByteArrayInputStream;	import android.content.SharedPreferences;
import java.io.InputStream;	import android.content.pm.ActivityInfo;
import java.util.Locale;	import android.content.res.Configuration;
import Proceso_XML.ParserDetalleCentro;	import android.content.res.Resources;
import SOAP.TareaWSdetalleCentro;	import android.graphics.PorterDuff;
import android.app.AlertDialog;	import android.location.Location;
import android.content.Context;	import android.location.LocationManager;
import android.content.DialogInterface;	import android.net.Uri;
import android.content.Intent;	import android.os.Bundle;
import android.content.SharedPreferences;	import android.support.v4.app.Fragment;
import android.content.pm.ActivityInfo;	import android.support.v7.app.ActionBarActivity;
import android.content.res.Configuration;	import android.util.Log;
import android.content.res.Resources;	import android.view.LayoutInflater;
import android.net.Uri;	import android.view.Menu;
import android.os.Bundle;	import android.view.MenuItem;
import android.support.v7.app.ActionBarActivity;	import android.view.View;
import android.text.Html;	import android.view.View.OnClickListener;
import android.util.Log;	import android.view.ViewGroup;
import android.view.Menu;	import android.widget.ImageView;
import android.view.MenuItem;	import android.widget.TabHost;
import android.view.View;	import android.widget.TabHost.OnTabChangeListener;
import android.widget.TabHost;	import android.widget.Toast;
import android.widget.TabHost.OnTabChangeListener	import com.google.android.gms.maps.CameraUpdateFactory
import android.widget.TextView;	import com.google.android.gms.maps.GoogleMap;
	import com.google.android.gms.maps.SupportMapFragment;
	import com.google.android.gms.maps.model.CameraPositior
	import com.google.android.gms.maps.model.LatLng;
	import com.google.android.gms.maps.model.MarkerOptions;

MultiSpinner	PantallaPrincipalActivity
import android.app.AlertDialog;	import java.util.Locale;
import android.content.Context;	import android.app.AlertDialog;
import android.content.DialogInterface;	import android.content.Context;
import android.content.DialogInterface.OnMultiChoic	import android.content.DialogInte
import android.content.res.TypedArray;	import android.content.Intent;
import android.util.AttributeSet;	import android.content.SharedPre
import android.widget.ArrayAdapter;	import android.content.pm.Activit
import android.widget.Spinner;	import android.content.res.Config
	import android.graphics.PorterDuf
	import android.net.ConnectivityM
	import android.os.Bundle;
	import android.support.v4.app.Fra
	import android.support.v7.app.Act
	import android.util.Log;
	import android.view.LayoutInflate
	import android.view.Menu;
	import android.view.MenuItem;
	import android.view.View;
	import android.view.ViewGroup;
	import android.view.View.OnClick
	import android.widget.ImageView

Parser	ParserBuscAv
import java.io.IOException;	import java.io.IOException;
import java.io.InputStream;	import java.io.InputStream;
import java.util.ArrayList;	import java.util.ArrayList;
import java.util.List;	import java.util.List;
import org.xmlpull.v1.XmlPullParser;	import org.xmlpull.v1.XmlPullParser
import org.xmlpull.v1.XmlPullParserException;	import org.xmlpull.v1.XmlPullParser
import android.util.Xml;	import android.util.Xml;
import com.example.pruebasoap.Centro;	import com.example.pruebasoap.Ce
ParserDetalleCentro	**ParserInfoAdicional**
import java.io.IOException;	import java.io.IOException;
import java.io.InputStream;	import java.io.InputStream;
import java.util.ArrayList;	import java.util.ArrayList;
import java.util.List;	import java.util.List;
import org.xmlpull.v1.XmlPullParser;	import org.xmlpull.v1.XmlPullParser
import org.xmlpull.v1.XmlPullParserException;	import org.xmlpull.v1.XmlPullParser
import android.util.Xml;	import android.util.Xml;
import com.example.pruebasoap.Centro;	import com.example.pruebasoap.Ce

SSLConnection	TareaWSBuscadorAvanzado
import android.util.Log;	import java.util.Locale;
import javax.net.ssl.HostnameVerifier;	import org.ksoap2.SoapEnvelope;
import javax.net.ssl.SSLSession;	import org.ksoap2.serialization.SoapObjec
import javax.net.ssl.TrustManager;	import org.ksoap2.serialization.SoapSerial
import java.security.KeyManagementException;	import org.ksoap2.transport.HttpTranspor
import java.security.NoSuchAlgorithmException;	import android.os.AsyncTask;
import java.security.SecureRandom;	import android.util.Log;
import java.security.cert.CertificateException;	
import java.security.cert.X509Certificate	
TareaWSConsulta	**TareaWSDetalleCentro**
import org.ksoap2.SoapEnvelope;	import org.ksoap2.SoapEnvelope;
import org.ksoap2.serialization.SoapObject;	import org.ksoap2.serialization.SoapObjec
import org.ksoap2.serialization.SoapSerializationEnve	import org.ksoap2.serialization.SoapSerial
import org.ksoap2.transport.HttpTransportSE;	import org.ksoap2.transport.HttpTranspor
import android.os.AsyncTask;	import android.os.AsyncTask;
import android.util.Log;	import android.util.Log;

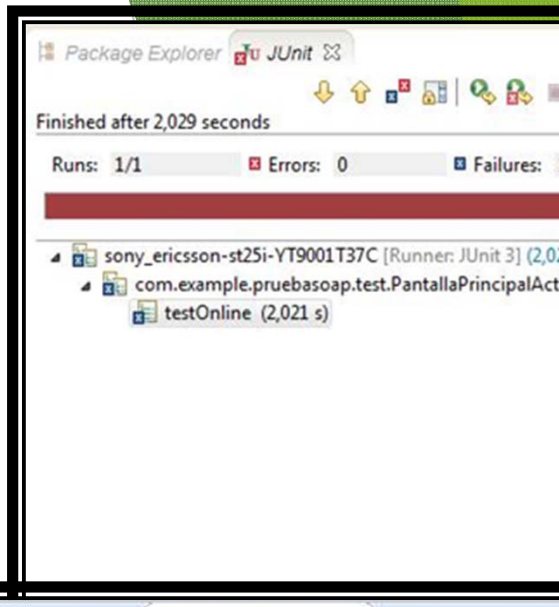
Arquitectura del sistema

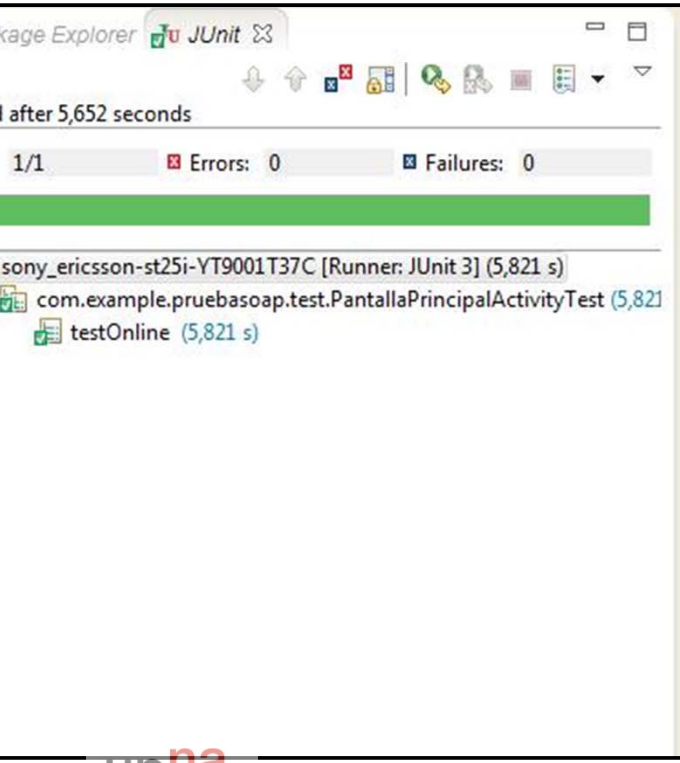


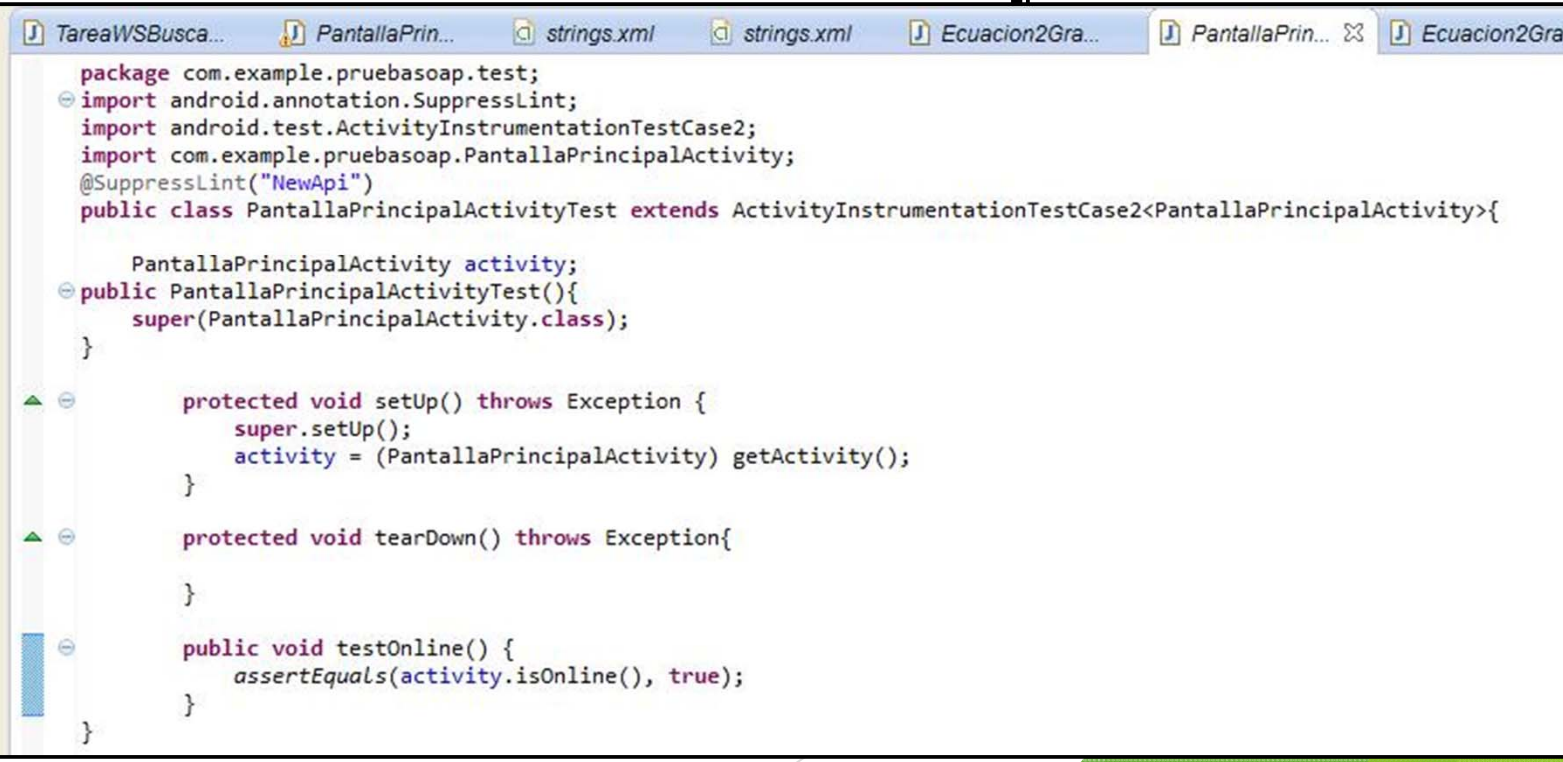
Demo

Pruebas

- Pruebas de caja negra y caja blanca en el desarrollo de la aplicación.
- Logs (trazas de ejecución)
- Android Test Unit







Conclusiones

- El proyecto ha resultado todo un éxito al resultar ganador del I Premio de Diseño de Aplicaciones para Teléfonos Inteligentes (Smartphones) con datos abiertos del Departamento de Educación del Gobierno de Navarra.

- Aprender en el desarrollo de un proyecto informático, así como resolver los imprevistos que surgen durante la elaboración del mismo.

- Se ha conseguido una aplicación de gran utilidad para informar al usuario de forma bilingüe a cerca de detalles de los centros educativos de Navarra.

Líneas futuras

- Realizar una especie de red social por centros, eligiendo de qué centro eres y poder hacer un chat entre centros, con posibles juegos y ranking entre centros…
- Realizar búsquedas por proximidad a un centro educativo.
- Ofrecer opciones de administrador para poder cambiar los detalles de un centro desde la propia aplicación para una persona autorizada.
- Al hilo de la red social sacar si tus contactos están utilizando la aplicación, a qué centro pertenecen…

Aplicación Android Buscador de Centros Educativos de Navarra

Autor: Mikel San Martín Huarte

Tutor: Óscar Ardaiz Villanueva

Pamplona, a 10 de noviembre de 2014