UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Aplicación Android para gestión remota de datos policiales

Trabajo Fin de Máster

Máster Universitario en Ingeniería Informática

Autor: Pablo Adrián Moreno Sierra

Tutor: Carlos Miguel Tavares Calafate

Curso: 2016/2017

Resumen

En este proyecto se propone la realización de una aplicación para Tablet Android, que permita a los agentes de la Policía Local de Mislata consultar los datos tanto de la base de datos alojada en sus servidores, donde se pueden verificar datos de ciudadanos, vehículos o locales comerciales entre otros, como otras bases de datos y servicios web como puede ser la Dirección General de Tráfico (DGT), y tener el acceso a todos estos servicios unificados en la misma app, además, mantener localizados en tiempo real estos dispositivos a través de un servicio web con un mapa. Y todo esto sin dejar atrás la seguridad de la red y los servidores a causa de estos dispositivos.

Palabras clave: Android, Policía, Base de datos, Acceso remoto, Gestión.

Abstract

This project proposes developing an Android Tablet Application that allows the agents of the Local Police of Mislata to perform queries to the database hosted in their servers. Through this application they can check information about citizens, vehicles or shops. In addition, this app must perform queries to other databases and web services like the ones from DGT (National Traffic Agency). The access to all these services is unified in the same app, besides of keeping a full-time location of those devices through a web service with a map. And always taking in consideration the security on the network and even, the hosted servers because of those devices.

Keywords: Android, Police, Database, Remote Access, Management.

Tabla de contenidos

1. Introducción 9
   1.1. Motivación 9
   1.2. Objetivos 10
   1.3. Estructura de la memoria 10
2. Estudio de soluciones existentes 12
3. Propuesta del proyecto 15
4. Detalles de desarrollo del servidor 16
   4.1. Instalación y configuración de las herramientas 16
   4.2. Detalles del desarrollo 26
5. Detalles de desarrollo del cliente Android 33
   5.1. Desarrollo del reconocimiento de voz 43
   5.2. Desarrollo del sistema de localización GPS 46
6. Validaciones y pruebas 52
7. Conclusiones 60
8. Bibliografía 61

Tabla de ilustraciones

Ilustración 1: Gescar 12
Ilustración 2: Opciones Gescar 12
Ilustración 3: Menú Europol 13
Ilustración 4: Denuncias Europol 13
Ilustración 5: SQL Server Management Studio 20
Ilustración 6: SQL Developer 23
Ilustración 7: Interfaz principal Aplicación Android 34
Ilustración 8: Registro del dispositivo 36
Ilustración 9: Menú opciones 37
Ilustración 10: Historia de consultas 38
Ilustración 11: Menú Gespol 39
Ilustración 12: Consulta Ciudadanos 40
Ilustración 13: Confirmación de búsqueda por voz 41
Ilustración 14: Detalles del ciudadano 41
Ilustración 15: Creación de un informe de novedades 42
Ilustración 16: Aplicación, fallo de conexión 52
Ilustración 17: Aplicación, error de servidor 53
Ilustración 18: Aplicación, sin resultados 53
Ilustración 19: Aplicación, inyección SQL 54
Ilustración 20: Aplicación, sin valor de búsqueda 54
Ilustración 21: Aplicación, buscando 55
Ilustración 22: Aplicación, resultado ciudadano 55
Ilustración 23: Aplicación, resultado vehículo 56
Ilustración 24: Aplicación, detalles de ciudadano 56
Ilustración 25: Aplicación, varios resultados 57
Ilustración 26: Aplicación, edición de informes 57
Ilustración 27: Aplicación, identificador ya utilizado 58
Ilustración 28: Aplicación, opciones 58
Ilustración 29: Aplicación, historial 59
Ilustración 30: Mapa con dispositivos 59

1. Introducción

Desde hace unos cuantos años, las TIC están facilitando el acceso a recursos a través de Internet, utilizando para ello los ordenadores, pero, más recientemente se está viendo un auge en los dispositivos móviles que permiten acceder a la mayoría de recursos a través de Internet. Para ello contamos con dispositivos como smartphones, tablets, e incluso *wearables*, como pueden ser los relojes o pulseras inteligentes.

Fue en 2010 cuando Apple presentó el iPad, y a partir de ese momento empezó a conocerse el concepto que tenemos de Tablet y empezó su amplia difusión, hoy en día compartida entre bastantes fabricantes como Apple, Google, Samsung, Sony…[1].

Estos dispositivos están muy extendidos, pero en lugares como en las administraciones públicas como la policía o los hospitales no están tan extendidos, y es ahora cuando están empezando a aparecer estas "nuevas" tecnologías ya comunes para el groso de la población. Gracias a estas tecnologías, la administración pública tiene la oportunidad de ganar en eficiencia e incluso poder ofrecer nuevos servicios a los ciudadanos.

1.1. Motivación

En base a una experiencia laboral como informático en prácticas la Policía Local de Mislata se ha realizado un análisis del funcionamiento del sistema informático de dicha entidad, y se han detectado algunas carencias, entre las cuales destaca la utilización de Windows XP, a pesar de estar sin mantenimiento por parte de Microsoft, entre otras compañías.

Igualmente se ha constatado que los agentes que estaban en la calle tenían que llamar a central para verificar, por ejemplo, los datos de algún ciudadano o vehículo. En base a ese análisis se consideró que esa acción podía ser mucho más ágil y cómoda sin entorpecer las labores de la central por estos menesteres. Para este propósito, se podría utilizar una Tablet que conectara con los servidores de la central, permitiendo a los agentes en el terrero poder realizar cualquiera consulta desde su ubicación, incluso mediante comandos de voz para hacerlo de manera más ágil, cómoda y sin distracciones en caso de situación comprometida.

Se han buscado también proyectos parecidos en Google Scholar y en el polibuscador de la UPV, donde solo se ha encontrado referencia al proyecto "Unixpol" [2] y "Aplicación para la gestión policial de la violencia de género del municipio de Valencia y pedanías" [3]. Dichas aplicaciones no contemplan el acceso mediante dispositivo móvil, por lo que se puede observar una carencia de

proyectos innovadores en el campo de la gestión policial mediante dispositivos móviles.

En base al análisis realizado, se ha propuesto este proyecto al jefe de la Policía Local de Mislata, lo cual ha viabilizado su realización.

1.2. Objetivos

El objetivo principal de este proyecto consiste en realizar una aplicación Android para Tablet, que permita a los agentes de la Policía Local de Mislata realizar consultas a la base de datos de la propia policía, el padrón municipal y otros servicios web como puede ser el de la DGT o la policía nacional, y sin tener que llamar a central para ello, además, incorporar a estas Tablets un sistema de localización, de manera, que puedan ser localizadas en cualquier momento y seguidas en tiempo real. Y todo esto sin renunciar a la seguridad, que es un factor clave en este caso.

1.3. Estructura de la memoria

Esta memoria está organizada en el siguiente conjunto de capítulos o secciones, los cuales, contienen información acerca de cada uno de los aspectos que componen el proyecto realizado. A continuación, se describe cada una de las secciones.

- **Capítulo 1 – Introducción:** Sección que pretende situar al lector en el contexto en el que se desarrolla el proyecto, exponiendo así la idea general del mismo, los objetivos que este persigue y la motivación que ha impulsado su realización.

- **Capítulo 2 – Estudio de las soluciones existentes:** En este apartado se presentan de manera general las soluciones existentes para este tipo de servicios.

- **Capítulo 3 – Propuesta del proyecto:** Este capítulo describe con precisión la arquitectura de este proyecto y los distintos elementos que lo componen.

- **Capítulo 4 – Detalles de desarrollo del servidor:** En esta sección se muestran los detalles de implementación del lado del servidor.

- **Capítulo 5 – Detalles de desarrollo del cliente Android:** Este apartado presentan los detalles de la implementación del lado del cliente.

- **Capítulo 6 – Validaciones y pruebas:** En este epígrafe se describen las pruebas y validaciones realizadas en el sistema desarrollado.

- **Capítulo 7 – Conclusiones:** En este capítulo se analizan los resultados logrados frente a los objetivos marcados al inicio del proyecto y se establecen futuras líneas de trabajo.

- **Capítulo 8 – Bibliografía:** En esta última sección se enumeran los recursos bibliográficos a los que se ha recurrido para la elaboración del proyecto.

2. Estudio de soluciones existentes

Tenemos varias soluciones existentes para la gestión policial portátil, entre ellas destacan las siguientes:

- **Gescar (Gespol Car):** Es la solución para vehículos patrulla de Gespol y conecta con su ERP. El ERP de Gespol es el que posee actualmente la Policía Local de Mislata en sus ordenadores. Su plataforma móvil tiene las siguientes características.
    - Consulta ciudadanos, vehículos, locales comerciales, vados a la base de datos de Gespol instalada en los servidores de la policía. Además, permite dar de alta incidencias o multas, además de realizar consultas a la DGT y consultar cámaras de vigilancia. Seguimiento GPS.
    - El manual de dicha *app* puede ser consultado en: https://es.scribd.com/document/6566406/Manual-Gespol-V
    - A continuación, se muestran unas imágenes de la aplicación:

Ilustración 1: Gescar

Ilustración 2: Opciones Gescar

- **VinfoPOL Mobile:** Es la solución móvil que da VinfoPOL para conectar con su ERP. Posee las siguientes características:
    - Consulta de ciudadanos, vehículos y locales comerciales, realizar consultas a la DGT, insertar novedades o crear cuadros de control como cosas más destacadas. Se puede consultar más información sobre esta herramienta en: http://www.vinfoval.es/programas/software-de-gestion-policial-vinfopol/

- **EuroCop Mobile:** Solución móvil de EuroCop, y que permite conectar con su ERP. Destacan las siguientes características:
    - Consulta a los datos proporcionados por su ERP, la DGT, turnos de agentes, registro Catastral, registro de vehículos, e integración con cámaras de seguridad y GPS.
    - Para más información sobre esta aplicación se puede consultar: http://www.eurocop.com/sistemas-de-eurocop/sistemas-de-movilidad/eurocop-mobile/
    - A continuación, se muestran unas capturas de dicha aplicación:

Ilustración 3: Menú Europol

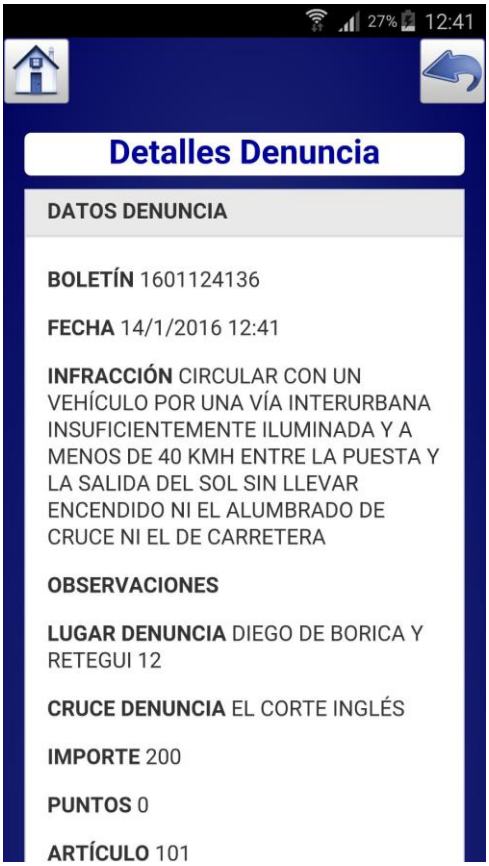

Ilustración 4: Denuncias Europol

Estas soluciones móviles tienen algo importante en común, y es que se centran fundamentalmente en lo mismo, ofreciendo soluciones similares. Exigen conexión con su propio ERP, por lo que se necesita un ERP de la misma compañía para poder operar, y si sumamos el coste del ERP con el alto coste también de su aplicación móvil nos encontramos con un sistema muy caro a la par que limitado.

3. Propuesta del proyecto

El proyecto consiste en proporcionar una aplicación Android a unas tabletas que llevarán los coches de la Policía Local, las cuales se conectarían de forma segura con el retén de la Policía Local.

La aplicación tendrá una interfaz que permitirá acceder a las diferentes secciones de esta aplicación; Base de datos de Gespol (base de datos de la Policía Local de Mislata), Dirección General de Tráfico (DGT), Cuerpo Nacional de Policía (CNP), padrón municipal, y página web interna de la Policía Local de Mislata.

- Desde estas tabletas se podrá hacer consultas a la base de datos de Gespol, la cual permitiría elaborar *in situ* los informes directamente desde la propia tableta proporcionando, datos más concretos sobre los hechos acaecidos, como asimismo consultar datos sobre atestados anteriores, además de consultar otros datos como información acerca de los ciudadanos, vehículos, locales de Mislata y otros datos recogidos en la base de datos de Gespol, y todo esto sin tener que llamar al retén como se hace actualmente.

- También se podrá utilizar el servicio web, Atex, de tráfico, para obtener datos de las matrículas tal que el propietario o la posible carencia de seguro o ITV.

- Asimismo, se podrán consultar los antecedentes en el servicio web, Escorial, de la Policía Nacional, pudiendo así identificar en caso de que sea necesario a alguna persona, en lugar de tener que llamar al Retén de la Policía Local para realizar dicha consulta, ahorrando así una gran cantidad de tiempo.

- Se proporcionará una conexión a la base de datos del Padrón Municipal para poder obtener los datos de éste, ya sea para una identificación de un ciudadano o ver la localización de su domicilio.

- Historial de consultas realizadas a las diferentes bases de datos, de manera que se puedan ver las consultas que han sido realizadas y copiarlas para volverlas a buscar si así se desea.

- La aplicación también mantendrá los dispositivos localizados en tiempo real y visibles en todo momento mediante un servicio web con un mapa donde estos aparecerán. Así pudiendo localizarlos por si hubiera cualquier problema o incluso robo del vehículo o la Tablet.

4. Detalles de desarrollo del servidor

Para mayor seguridad, las consultas a la base de datos se realizan mediante un servidor web como intermediario, de manera que el cliente Android envía una serie de parámetros y el servidor web, mediante PHP, ejecuta una serie consultas SQL predefinidas, relacionados a dichos comandos, controlando además la inyección SQL. De esta manera, no es posible ejecutar ninguna consulta fuera de las ya establecidas.

Como ya sabemos, Windows Server 2003, junto con Windows XP, terminaron su periodo de soporte ampliado el 14 de julio de 2015 y 8 de abril de 2014, respectivamente.

Cuando se inició el proyecto los servidores del ayuntamiento de Mislata estaban en la versión de Windows Server 2003, y se trataba de servidores físicos, además de bastantes ordenadores utilizando Windows XP. Actualmente se está haciendo una migración a servidores virtuales con Windows Server 2012.

Esta migración ha provocado un cambio en el proyecto, que inicialmente se inició para un servidor Windows Server 2003, y ha tenido que ser modificado a Windows Server 2012, por lo que se va a detallar las variantes en ambos sistemas operativos con el fin de observar sus diferencias.

4.1. Instalación y configuración de las herramientas

4.1.1. Versión Windows Server 2003

Como ya se ha comentado, inicialmente se trataba de un servidor con Windows Server 2003 alojado en el propio retén de la Policía Local de Mislata, y conectado a la red local del ayuntamiento.

Hay que destacar que al tratarse de una versión sin soporte por parte de Microsoft y muchos otros elementos software, como pueden ser las aplicaciones de Google o incluso el propio PHP y Apache, terminó en muchas horas de investigación para buscar la manera de instalar el software con las versiones pertinentes e incluso encontrarlas por la red, debido a que en ocasiones no estaban en las páginas oficiales por estar obsoletos actualmente.

Inicialmente se planteó que servidor web utilizaría entre XAMPP o IIS. Procedamos a analizarlos:

- XAMPP:
    - **Descripción:** Entorno muy popular de desarrollo con PHP. XAMPP es una distribución de Apache completamente gratuita y fácil de instalar que contiene MariaDB, PHP y Perl. Con un paquete de instalación muy fácil de instalar y utilizar.
    - **Multiplataforma:** Versiones para Windows, Linux y OS X.
    - **Versiones actuales:** 7.1.7 y 5.6.31 (Windows XP y Server 2003 no soportadas). Mismas versiones que PHP.
    - **Última versión compatible:** La última versión compatible con Windows Server 2003 es la 1.7.2. Contiene Apache 2.2.12 y PHP 5.3 de 32 bits (Última versión compatible de PHP).
    - **Instalación:** Descargable desde la página oficial y el propio ejecutable te instala y configura las herramientas requeridas.

- IIS:
    - **Descripción:** Internet Information Server, más conocido como IIS es un servidor web y un conjunto de servicios para Microsoft Windows.
    - **Sistema operativo:** Solo para las diferentes versiones de Windows desde Windows Server 2000.
    - Versión actual: IIS v10.0
    - **Última versión compatible:** IIS v6.0, Versión PHP 5.3.28 (32 bits).
    - **Instalación:** Es instalable desde el propio Windows Server en la aplicación “Administre su servidor” y seleccionando “Agregar o quitar función” se puede instalar “Servicios de Internet Information Server”.

Especialmente debido a la mayor estabilidad, en Windows se ha optado por crear el servidor en el IIS.

La instalación del IIS se ha realizado desde la aplicación "Administre su servidor" y seleccionando "Agregar o quitar función", y a continuación seleccionando "Servicios de Internet Information Server".

Una vez instalado el IIS, la forma más recomendable para instalar el PHP es mediante el WPI (Instalador de plataforma web) de Microsoft, versión 5.0. Este es el instalador de Microsoft para herramientas web y se puede descargar desde las fuentes de Microsoft.

Como Windows Server 2003 está sin soportar por parte de Microsoft el WPI no funciona correctamente y hay que editar el registro de Windows para arreglarlo. Para ello hay que abrir la aplicación "regedit.exe" que permite editar en el registro de Windows. Ahí, hay que navegar a "HKLM\Software\Microsoft\webplatforminstaller". Aquí hay que crear una clave como *String* y llamarla "ProductXMLLocation", y como valor añadir "http://www.microsoft.com/web/webpi/5.0/webproductlist.xml". Esto hará que el WPI obtenga correctamente la lista de productos descargables desde el mismo.

Una vez solucionado este inconveniente se puede proceder a la instalación de PHP, donde hay que seleccionar la única versión compatible para Windows Server 2003, PHP 5.3.19, y también es necesario FastCGI para poder utilizar el PHP desde el IIS.

Para el correcto funcionamiento de PHP desde el IIS hay que configurar una serie de cosas:

- Editar el fichero php.ini en el directorio donde este se ha instalado:
    - Añadir la línea: "cgi.force_redirect = 0"
    - Añadir la línea: "fastcgi.impersonate = 1"
    - Buscar la clave "extension_dir" y poner como *path* el directorio "ext" de la carpeta PHP.
    - Buscar la clave "date.timezone" y poner "Europe/Madrid" para España.
    - Comprobar que el fichero "C:\WINDOWS\system32\inetsrv\fcgiext.ini" contiene el *path* de PHP indicado correctamente.
- En Herramientas administrativas -> Administrador de IIS -> "Servidor local" -> Extensiones de servicio web. Comprobar que "FastCGI Handler" esté permitido.

- También en Herramientas administrativas -> Administrador de IIS -> "Servidor local" hay que ir a "Sitios web" e ir a propiedades y la pestaña "Directorio particular" y configuración, lugar donde hay que agregar una nueva extensión de aplicación. Esta será ".php" y se tiene que indicar como librería ejecutable la del FastCGI, "C:\WINDOWS\system32\inetsrv\fcgiext.dll".

Una vez realizados estos pasos es necesario reiniciar el IIS o el servidor y ya debería funcionar. Se puede comprobar creando un sitio web en Herramientas administrativas -> Administrador de IIS -> "Servidor local" hay que ir a "Sitios web" y añadiendo un fichero *.php* con el siguiente contenido:

```
<?php phpinfo(); ?>
```

Si en la web donde esté colocada esa línea nos muestra una tabla informativa de PHP todo estará correcto, por ejemplo, en "http://localhost/phpinfo.php".

Con el PHP ya funcionando, y para realizar pruebas con la base de datos que hay instalada en la Policía Local de Mislata, Microsoft SQL Server 2008.

Para la instalación de Microsoft SQL Server 2008, basta con dejar seleccionados los valores por defecto y únicamente añadir la opción de instalar "SQL Server Management Studio" para poder gestionar bases de datos; crearlas, crear sus tablas, inserciones, etc. Posteriormente hay crear una ID de servicio de base de datos, que también se puede dejar por defecto, y únicamente añadir una contraseña y más usuarios si se desea.

Ya con estos servicios instalados hay que crear una base de datos para lo que se utiliza el Microsoft SQL Server 2008. Este paso es muy intuitivo, basta con seleccionar el servicio de base de datos y en la carpeta "bases de datos", con botón derecho se selecciona "Nueva base de datos..." y se establece un nombre para esta. Con esta base de datos creada, se crean las tablas necesarias donde encontraríamos los ciudadanos, vehículos, nacionalidades, locales comerciales, etc. La estructura de estas tablas ha sido proporcionada por la Policía Local de Mislata de cara a poder crear datos de prueba con los que realizar las pruebas de la aplicación.

Finalmente, para poder utilizar las funciones PHP para consultar la base de datos Microsoft SQL Server 2008 desde Microsoft Windows Server 2003 hay que configurar el driver necesario. En este caso es "SQLSRV" versión 2.0. Ya que es la última versión compatible para este sistema operativo y base de datos. Este driver se puede descargar desde la página web de Microsoft. Y una vez descargado, hay que poner los ficheros *.dll* en la carpeta "ext" de PHP que se

configuró previamente en el fichero “php.ini”. También hay que volver a editar este fichero y añadir al final las líneas:

```
extension=php_pdo_sqlsrv_53_nts_vc9.dll
extension=php_sqlsrv_53_nts_vc9.dll
```

Estas líneas coinciden con el nombre de la librería requerida. Si analizamos el nombre de esta librería podemos ver que es para la versión 5.3 de PHP. También es requerido utilizar la librería “nts” o “non-thread-blocking” ya que es la única compatible con el IIS. Y por último vemos que es está compilada con Microsoft Visual Studio 2009.

Una vez reiniciado el servidor ya debería estar funcionando esta librería correctamente y se deberían poder realizar consultas a la base de datos desde PHP utilizando para ello los comandos necesarios.

4.1.2. Versión Windows Server 2012

Actualmente, el Ayuntamiento de Mislata está migrando sus servidores a Windows Server 2012 alojadas en máquinas virtuales, por lo que el desarrollo ha tenido que ser trasladado a este sistema operativo, lo cual implica realizar una instalación del mismo para realizar las pruebas. Destacar que esta vez todas las instalaciones han sido mucho más sencillas, guiadas e intuitivas.

Con el sistema operativo instalado se ha instalado Microsoft SQL Server 2012 como base de datos, donde con éste, como en el caso anterior, se ha instalado “SQL Server Management Studio” para la gestión de las bases de datos.

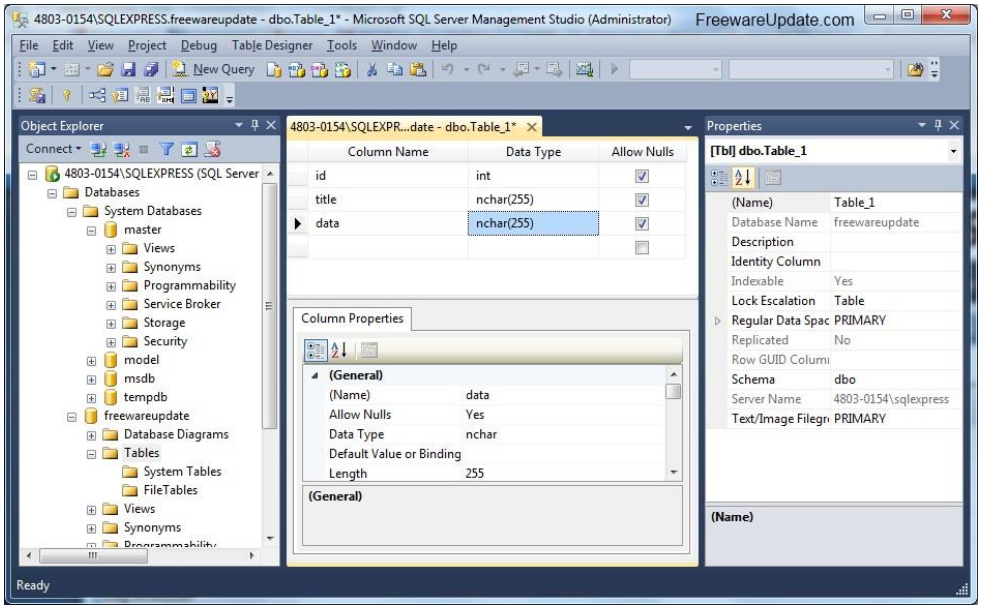

Ilustración 5: SQL Server Management Studio

Con la base de datos ya creada, desde el gestor de bases de datos se importa la base de datos que teníamos en Windows Server 2003.

De cara a la instalación del IIS hay que ir a Inicio -> Administrador del servidor. Y en este, seleccionar "Agregar roles y características", posteriormente "Roles de servidor", a continuación, "Servidor web (IIS)" y tras esto, seleccionar características, en este caso, "CGI", lo cual nos instalará FastCGI para que después podamos instalar PHP.

Con esta instalación terminada, procedemos a ejecutar el "Administrador de Internet Information Services (IIS)", y en este seleccionamos el servidor local, para posteriormente ir al Instalador de plataforma web, que esta vez está preinstalado en Windows, y en este hay que instalar el software "PHP 7.1.7" de 64 bits y el "Microsoft Drivers 4.3 for PHP 7.1 for SQL Server in IIS" de 64 bits, que contiene el driver para SQL Server en su última versión. Con esto ya se instala PHP con los Drivers correspondientes para SQL Server 2012.

Una vez hecho esto ya se puede crear el sitio web desde "Administrador de Internet Information Services (IIS)" e importar los ficheros web que ya teníamos en Windows Server 2003.

En este momento ya tenemos lo mismo que en la versión Windows 2003.

Para acceder a la base de datos utilizando estos drivers desde PHP ahora ya solo queda utilizar los comandos pertinentes. En primer lugar, para conectar con la base de datos se hace mediante las siguientes líneas:

```
$options = array("UID" => $user,  "PWD" => $pass,  "Database" =>
$database, "CharacterSet"=>"UTF-8");

$conn = sqlsrv_connect($host, $options); //Error de conexion
if ($conn === false) die(print_r(sqlsrv_errors(), true));
```

En las líneas anteriores utilizamos la función "sqlsrv_connect" para conectar con la base de datos Microsoft SQL Server. Esta función consta de dos parámetros, la dirección de host de la base de datos, y un vector con los parámetros de conexión, que son: usuario, contraseña, nombre de base de datos y codificación de los datos que se van a transmitir entre DB y PHP, que se va a utilizar *UTF-8*[10]. Finalmente se comprueba mediante un *if* si se ha establecido correctamente la conexión.

Para cerrar la conexión con la base de datos únicamente hay que utilizar el comando siguiente:

```
sqlsrv_close($conn);
```

Como se ve en la línea anterior, se utiliza la función “sqlsrv_close” para cerrar la conexión. Como parámetro a esta función se le pasa la conexión a cerrar.

Por último, para realizar consultas:

```
$result = sqlsrv_query($conn, $consulta_sql);

if($result === false ) {
         echo "Error ejecutando la consulta: $consulta_sql";
         die( print_r( sqlsrv_errors(), true));
}

$json = array();

while ($row = sqlsrv_fetch_array($result, SQLSRV_FETCH_ASSOC)) {
        $json[] = $row;
}

sqlsrv_free_stmt($result);

$salida = json_encode($json);
```

El código anterior consta de una función “sqlsrv_query” que es la que hace la consulta a la base de datos. Esta función toma como parámetros la conexión establecida a la BBDD y un *String* con la consulta en lenguaje SQL. Tras esto, con un *if* se comprueba si ha habido algún error realizando la consulta. Si todo ha ido bien, se recorre con un bucle las filas resultantes, utilizando la función “sqlsrv_fetch_array” para crear un array con las filas resultantes, para esto, se pasa el parámetro resultado de la consulta y el comando “SQLSRV_FETCH_ASSOC”, que hace que resulte un array asociativo, esto es, que se pueda buscar en él por nombres de columna. Además, en este caso, como estos resultados se desean enviar a un cliente Android, se va a estructurar el array como un array de JSON, por lo que se guarda cada fila resultado como un elemento del array JSON. Una vez finalizado el bucle estructurando los resultados, se utiliza la función “sqlsrv_free_stmt” para liberar la memoria referente a esta consulta, y finalmente se utiliza “json_encode” para pasar el array JSON a *String* formateado como JSON.

Una vez ya tenemos todo lo relacionado con la instalación y configuración de Microsoft SQL Server en PHP hay que instalar y configurar la base de datos del padrón de habitantes de Mislata, que es Oracle Database 11g.

En primer lugar, desde la página web de Oracle se puede descargar el instalador de este motor de base de datos, en cuya instalación, basta con utilizar los valores por defecto de instalación, además de, crear una clave para el usuario de administración "sysdba". Posteriormente, para iniciar la base de datos cada vez que se inicie el sistema, hay que abrir el ejecutable "Start Database". También se puede programar el inicio de esta base de datos de varias maneras, como puede ser, añadiendo un acceso directo del ejecutable a la carpeta "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp", a través del programador de tareas de Windows, o en el registro de Windows, navegando hasta: "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run", y agregando ahí la dirección al ejecutable como un *String*. También es posible detener la base de datos iniciando el ejecutable "Stop Database".

Para gestionar la base de datos, el cliente más recomendado es el "SQL Developer". Este cliente de bases de datos es el recomendado y ofrecido por Oracle, pero también funciona con cualquier otro motor de base de datos agregando las extensiones correspondientes. Esta herramienta utiliza la máquina virtual Java por lo que hay que tenerlo instalado o utilizar la versión de SQL Developer que trae incluida la última versión de Java.

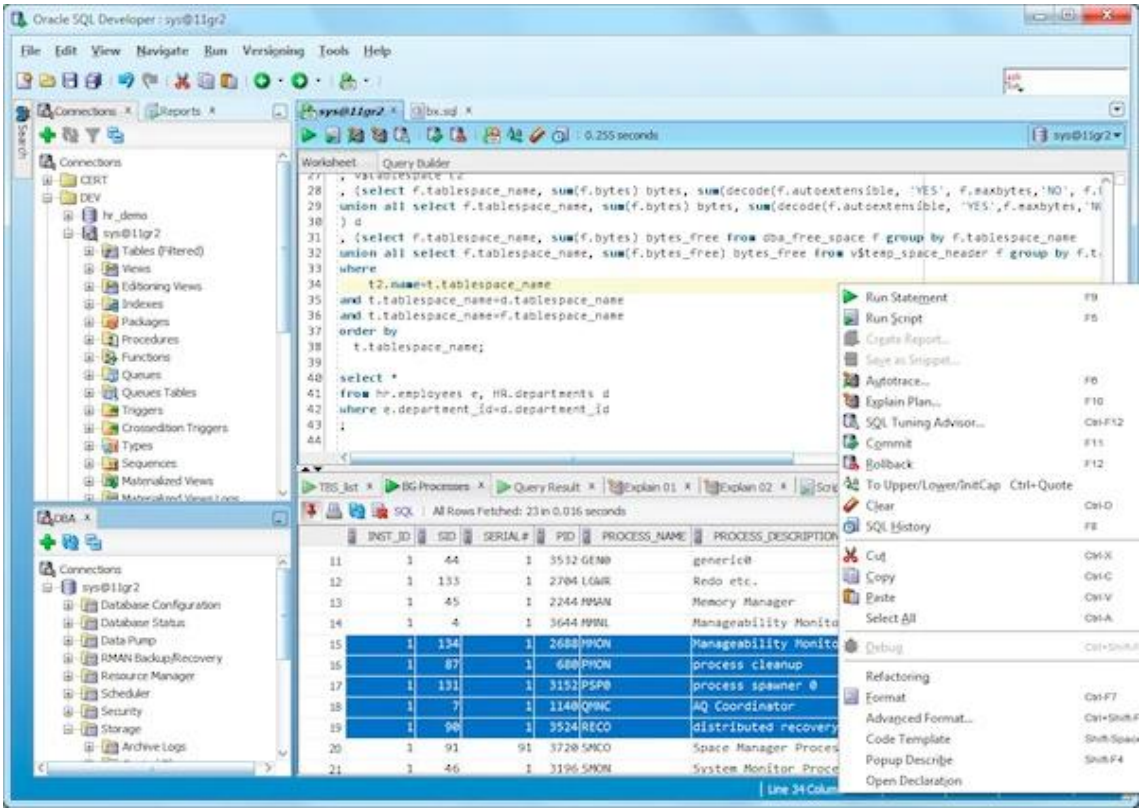

Ilustración 6: SQL Developer

En primer lugar, con el SQL Developer ya instalado e iniciado, hay que configurar la conexión a la base de datos, para lo cual basta con seleccionar "Conexiones" y seleccionar "Nueva conexión...", lo que abrirá una ventana

donde se deberá indicar el tipo de base de datos (en caso de tener extensiones para otras bases de datos), el nombre de usuario, contraseña, y host, puerto y SID, en caso de no ser los por defecto.

La primera vez, habrá que conectarse con el usuario de administración, "sysdba", para así crear un usuario para utilizar desde PHP y para crear la estructura de la base de datos. Para crear dicho usuario, utilizando el usuario de administración, hay que ir a "Otros Usuarios" y con clic derecho, seleccionar "Crear Usuario…". Esto abrirá una ventana donde habrá que indicar el nombre de usuario y su clave, y además, seleccionar los permisos deseados.

Una vez creado ese usuario ya se puede establecer una conexión con él y empezar a importar la estructura de la base de datos del padrón, la cual ha sido proporcionada por el ayuntamiento de Mislata y, asimismo, poder crear habitantes de prueba y otros elementos necesarios para realizar las pruebas.

Una vez está la base de datos con los datos pertinentes para las pruebas hay que proceder a instalar el driver para las bases de datos de Oracle, de PHP de 32 o 64 bits, según corresponda, para lo cual, hay que descargarse el driver "OCI8" desde el PELC (The PHP Extension Community Library), un repositorio de extensiones de PHP [10].

Ya con este driver descargado, hay que añadir la *dll* "php_oci8_11g.dll" a la carpeta "ext" dentro del directorio de PHP y editar el fichero "php.ini" del directorio PHP añadiendo la siguiente línea:

```
extension=php_oci8_11g.dll
```

Con estos pasos realizados, es necesario reiniciar el servidor web para que se apliquen los cambios.

Llegado este momento ya se pueden utilizar una serie de funciones para interactuar con la base de datos Oracle desde PHP. En primer lugar, para conectar con la base de datos se utilizan las líneas siguientes:

```
$conn = oci_connect($user, $pass, "localhost/XE", 'AL32UTF8');

if ($conn === false) {
        $e = oci_error();
        trigger_error(htmlentities($e['message'], ENT_QUOTES),
                                    E_USER_ERROR);
}
```

Como se puede observar en el código anterior, para conectar con la base de datos se utiliza la función “oci_connect” [15]. Los parámetros requeridos son: el usuario, la contraseña, el host y el SID de la DB en formato “host/SID”, y por último la codificación, que como en el caso anterior será la UTF-8 para que las palabras con acento o letra “Ñ” se vean correctamente. Luego con un if se verifica que la conexión haya sido exitosa y, en caso contrario, se lanza un error.

Para realizar la desconexión a la base de datos se utiliza la línea siguiente:

```
oci_close($conn);
```

Como se ve en la línea anterior, se utiliza la función “oci_close” y se le pasa el parámetro obtenido por el establecimiento de conexión.

Finalmente, para realizar las consultas a la base de datos de Oracle se utilizan las siguientes líneas:

```
$result = oci_parse($conn, $consulta_sql);
oci_execute($result);
if ($result === false ) {
         echo "Error ejecutando la consulta: $consulta_sql";
         die(print_r( sqlsrv_errors(), true));
}

$json = array();

while ($row = oci_fetch_array($result, OCI_ASSOC+OCI_RETURN_NULLS)) {
       $json[] = $row;
}

oci_free_statement($result);

$salida = json_encode($json);
```

En este código se utiliza la función “oci_parse” para preparar la sentencia SQL introducida para su ejecución. Sus parámetros son, la conexión a la base de datos y un *String* con la consulta. Posteriormente se utiliza la función “oci_execute” con el resultado de la función anterior para ejecutar esta consulta preparada. Se comprueba con un *if* si la consulta ha devuelto algún error, en caso de que todo haya ido bien, con la función “oci_fetch_array” se obtiene un array de filas resultado, pasando como parámetros el resultado de la consulta y los comandos “OCI_ASSOC+OCI_RETURN_NULLS” para que devuelva un array asociativo y para que devuelva los valores *null* de la base de datos como valores *null* de PHP. Como en el caso de la base de datos Microsoft SQL Server,

se asocia cada fila resultado como un elemento de array JSON para enviarlo al cliente en este formato. Cuando se han terminado de utilizar los resultados se liberan utilizando la función “oci_free_statement”. Finalmente, con la función “json_encode” como en el caso anterior, se transforma el JSON a *String*.

4.2. Detalles del desarrollo

4.2.1. Desarrollo del programa principal

En este apartado se va a detallar concretamente la manera en que el programa servidor lee las órdenes del cliente y las responde.

El programa principal consta de un fichero PHP que trata de leer una serie de variables que deben ser recibidas por POST para cada consulta y, según se reciban unas variables u otras se procederá a consultar un dato u otro. A continuación, se explica con más detalle.

En primer lugar, se utiliza las funciones “isset” y “empty” para comprobar si las variables existen y contienen datos. En el siguiente código se puede ver un ejemplo:

```
if(!(isset($_POST[$KEY_TIPO]) && !empty($_POST[$KEY_TIPO]))){
     die("Error de comando"); //No se envía clave TIPO gespol/padrón
}
```

Como se ha dicho antes, en el código anterior se comprueba que la variable “$KEY_TIPO” existe y tenga un valor y, en caso de no ser así, se cerrará la conexión. Este valor ha de contener un parámetro específico para diferenciar si la consulta es a la base de datos de Gespol o a la del padrón municipal, ya que se comprobará posteriormente en otro *if*. También se utiliza un código parecido al anterior para comprobar si existen las variables de Comando y de Parámetro. La variable Comando indica, por ejemplo, si se va a crear/actualizar una novedad o se va a buscar un ciudadano, vehículo, comercio o novedad. Por otro lado, la variable Parámetro, indica si se va a realizar la consulta mediante un DNI, apellido, matrícula o algún otro parámetro.

Si se comprueba que una clave es para la base de datos de Gespol, se comprobará si el Comando introducido está entre los listados, para posteriormente comprobar si el Parámetro introducido está dentro de las opciones preestablecidas. Por último, se enviaría una consulta establecida anteriormente, donde en la clausula WHERE del código SQL, se buscaría el valor pedido por el cliente como Parámetro X. En el siguiente ejemplo se puede ver más claro.

```
if ($_POST[$KEY_TIPO] === $KEY_GESPOL) {
    if ($_POST[$KEY_COMANDO] === $GET_CIUDADANOS) {
        $parametro = sqlsrv_escape_string($_POST[$KEY_PARAMETRO]);
        if (!($parametro == $DNI || $parametro == $APELLIDOS)){

        }
        $parametro_valor=sqlsrv_escape_string($_POST[$parametro]);
        $sql = "SELECT Nombre, Apellidos FROM $DB_CIUDADANOS";
        $resultado = buscar_en_DB_to_JSON($sql);
        echo $resultado;
    }
}
```

Como se puede ver en este sencillo ejemplo, después de comprobar la base de datos a la que se consulta, el Comando y el Parámetro, se utiliza la función "sqlsrv_escape_string" creada manualmente para evitar la inyección de código SQL en la base de datos y, así controlar que las variables tengan valores correctos para la búsqueda. A continuación, se introducen las variables del parámetro y del valor del parámetro en la consulta SQL. El valor de estas variables podría ser, por ejemplo, DNI como parámetro y 12345678 como valor de parámetro. Esta consulta se le pasa a una función que contiene el código necesario para ejecutarla y devolver un array JSON como se vio anteriormente. Finalmente, este resultado es enviado con un *echo*.

En caso de tratarse de un comando de creación o actualización de una novedad, los datos vendrían como un objeto JSON, y se verificaría cada uno de los valores para comprobar que no haya inyección SQL y, tratarlos en caso de estar vacíos.

4.2.2. Desarrollo de la aplicación de localización por GPS

Esta aplicación consta de 3 ficheros PHP bien diferenciados:

1- El primero de ellos gestiona la recepción de datos GPS y su inserción en la base de datos. Se trata de una base de datos Microsoft SQL Server 2012, pero es una base de datos específica para los datos GPS, por lo que no está vinculada a la base de datos de Gespol.

   Esta base de datos contiene 2 tablas. La primera de estas tablas está dedicada al registro de los dispositivos y consta de dos columnas: una para el identificador del dispositivo, el cual va a ser único y, por tanto, se va a utilizar el IMEI (International Mobile Station Equipment Identity), eso es un código que identifica el dispositivo a nivel global [16]. La segunda columna de la primera tabla contiene el nombre dado a ese

dispositivo. Este nombre es introducido por el cliente desde el dispositivo y también se comprueba que sea único en el momento de su inserción en base de datos. En el momento de instalación de la aplicación cliente se introducirán sus datos en esta base de datos, aunque posteriormente es posible su modificación.

La segunda tabla es la que se va a ocupar de mantener actualizada la ubicación del dispositivo, por lo que contiene 4 campos. Estos son, la ID del dispositivo, que va relacionada con la ID de la tabla anterior. La fecha exacta de actualización de la ubicación y, por último, las coordenadas, dadas en latitud y longitud.

Con la estructura de la base de datos ya aclarada se puede explicar paso por paso como funciona este fichero PHP.

En primer lugar, se conecta a la base de datos con los comandos que vimos anteriormente para la base de datos Microsoft SQL Server.

Después tenemos una serie de funciones. La primera, para comprobar si un dispositivo está registrado en la base de datos. Esta función consta de una consulta SQL que comprueba si una determinada ID y Nombre, están en la tabla donde se registran las IDs. En caso de encontrarse estos valores en base de datos se devolverá un OK.

La segunda función registra a un dispositivo mediante su ID y Nombre. Si la tabla no contiene dicho identificador se creará una fila nueva con éste y el nombre. En caso de existir dicho identificador, se actualizará su nombre.

La última función consta de un código SQL sobre la tabla que almacena las ubicaciones de los dispositivos. Este código inserta en caso de un dispositivo nuevo o actualiza en caso de ya tener datos del mismo.

Con estas funciones vistas se puede ver el funcionamiento del programa. En primer lugar, se espera por POST una variable que indique la ID del dispositivo. En caso de esta no existir esta variable, se comprobará mediante otra variable si el cliente desea comprobar si una determinada ID está registrada y en tal caso se llamará a la función pertinente para realizar la comprobación. En caso de no existir tampoco esta variable se comprobará si el cliente desea registrar una ID, en cuyo caso se llamará a la función para registrar un dispositivo. En caso de no cumplirse ninguna de estas condiciones, se cerrará la conexión dando un error. En caso de que el cliente indique correctamente la ID del dispositivo, se comprobará que se ha enviado las coordenadas de longitud y latitud, para proceder a

llamar a la función que inserta la localización del dispositivo, por lo que se le pasarán los parámetros ID, las coordenadas y el resultado de la función “date()” de PHP para almacenar la fecha actual del servidor.

2- El segundo fichero se encarga de conectarse a la base de datos para obtener los últimos datos de ubicación, fecha y nombre de cada dispositivo para después crear una estructura XML con ellos de manera que para cada dispositivo se obtenga el código XML un marcador para Google Maps [17][18]. A continuación, se detalla mejor con un ejemplo de código.

```
header("Content-type: text/xml");
$dom = new DOMDocument("1.0");
$node = $dom->createElement("markers");
$parnode = $dom->appendChild($node);

while ($row = sqlsrv_fetch_array($result, SQLSRV_FETCH_ASSOC)) {
        $nombre = trim($row['NOMBRE']);
        $node = $dom->createElement("marker");
        $newnode = $parnode->appendChild($node);
        $newnode->setAttribute("name",$nombre);
        $newnode->setAttribute("fecha",trim($row['Fecha']));
        $newnode->setAttribute("lat",trim($row['Latitud']));
        $newnode->setAttribute("lng",trim($row['Longitud']));
        $newnode->setAttribute("ID", trim($row['DeviceId']));
}

echo $dom->saveXML();
```

En el Código anterior están las 3 primeras líneas donde organizamos el XML, se introduce la versión del documento, se crean el elemento “markers” para englobar a los marcadores, por lo que dentro de este están los elementos “marker”. Con un bucle se recorren las filas resultado donde para cada elemento o dispositivo se genera una serie de atributos, que son, el nombre del dispositivo, la ID y la última fecha de actualización y coordenadas del mismo.

A continuación, se puede ver un código de ejemplo resultante.

```
<?xml version="1.0"?>
<markers>
        <marker name="Tablet2" fecha="08 Aug 2017 12:53:45"
                lat="39.47490716953975" lng="-0.4202941551996423"
                ID="353346056362122"/>
        <marker name="Tablet1" fecha="08 Aug 2017 12:54:39"
                lat="39.46085325137945" lng="-0.38071886599929855"
                ID="355188060259953"/>
```

```
</markers>
```

3- El ultimo fichero es el que va a mostrar la interfaz web, concretamente un mapa proporcionado por Google Maps, en el cual se podrán observar los diferentes marcadores situados en la localización pertinente en su última fecha de localización.

El primer paso para tener una web que muestre los mapas de Google es obtener una clave de API. Para lo cual hay que tener una cuenta Google y acceder a https://console.developers.google.com. En esta web es necesario crear un proyecto y, al momento te saldrá una clave de API para utilizarla en el proyecto deseado [19]. Una vez conseguida la clave API hay que habilitar una serie de servicios, por lo que en la misma web hay que navegar hasta Biblioteca, lugar donde hay que ir a "APIs de Google Maps" y clicar en "Más" para que aparezcan todas las opciones disponibles. De esta lista habrá que habilitar los servicios "Google Maps JavaScript API" y "Google Maps Geocoding API". El primero de estos servicios es para realizar consultas básicas a la API desde JavaScript y, la segunda se utiliza para a partir de unas determinadas coordenadas poder obtener una dirección exacta, ya que, es más entendible para una persona.

En el siguiente código se va a mostrar la función principal de este fichero, que es "initMap()". Esta es la función que se ejecuta inicialmente y genera el mapa y sus elementos.

```
function initMap() {
   map = new google.maps.Map(document.getElementById("map"), {
         center: new google.maps.LatLng(39.474565, -0.420401),
         zoom: 16,
         mapTypeId: 'roadmap'
   });
   geocoder = new google.maps.Geocoder();
   infoWindow = new google.maps.InfoWindow;

   if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(function(position) {
           var pos = {
                lat: position.coords.latitude,
                lng: position.coords.longitude
           };
          var marker = new google.maps.Marker({
                position: pos,
                map: map,
                icon: new
google.maps.MarkerImage('//maps.gstatic.com/mapfiles/mobile/mobileimgs
2.png',
             new google.maps.Size(22,22),
             new google.maps.Point(0,18),
             new google.maps.Point(11,11))
           });
```

```
            map.setCenter(pos);
        }, function() {
            handleLocationError(true, infoWindow, map.getCenter());
        });
    }else {
        //Si el buscador no soporta la Geolocation
        handleLocationError(false, infoWindow, map.getCenter());
    }
    //llamada a los marcadores
    downloadUrl("marcadores.php", processXML);
}
```

Como se puede observar en el código anterior, lo primero que se hace en la función "InitMap()" es generar el mapa, en el cual, se establece el centro del mapa con unas coordenadas establecidas, un grado de zoom en el que se pueda ver de cerca el pueblo y, finalmente se indica que sea un mapa de carreteras. También se inicia el "geocoder" y las ventanas de información, para que cuando se seleccione un marcador, se abra una ventanita con más información sobre el mismo. Después, se trata de leer la ubicación del usuario para colocar un marcador simbolizando "tu ubicación actual". Finalmente, se utiliza la función "downloadUrl" para cargar los marcadores generados por el fichero PHP anterior y va a llamar a una función que va a procesar el XML leído, donde introducirá los marcadores, al igual que en el código de arriba, al introducir el marcador de la "ubicación actual".

Esta función de cargar los marcadores de la base de datos se realizará periódicamente utilizando el siguiente código dentro de la función "processXML":

```
setTimeout(function() {
      downloadUrl("marcadores.php", processXML);
}, timeout);
```

La última parte del fichero contiene como parte del código HTML, una llamada asíncrona con la clave de API obtenida anteriormente que llamará a la función "InitMap" si Google lo autoriza, a continuación, se puede ver el código para ello:

```
<script async defer
    src="https://maps.googleapis.com/maps/api/js?key=CLAVE_API
        &callback=initMap">
</script>
```

Con esto se ha visto toda la parte de desarrollo del servidor.

5. Detalles de desarrollo del cliente Android

De cara al desarrollo se ha utilizado la herramienta de Google, Android Studio, en su última versión (2.3.3). Esta herramienta es actualmente la más recomendada de cara al desarrollo Android. El lenguaje de programación utilizado ha sido el de Java para la lógica y XML para la interfaz de usuario. En cuanto al Java, se ha utilizado el JDK (Java Development Kit) versión 8 *update* 131 de 64 bits, es decir, una de las últimas versiones disponibles.

En cuanto a las versiones de Android, se ha focalizado el desarrollo en la versión de SDK 25, esto es, Android 7.1 y, como versión mínima de SDK se ha utilizado la 16, que equivale a la versión 4.1 de Android. Se ha utilizado esta versión mínima de SDK debido a la incompatibilidad en versiones inferiores de algunos recursos de programación no compatibles con versiones anteriores a la 16.

Para la realización de la conexión entre el dispositivo Android y el servidor se debatió entre dos opciones, la primera de ellas, la librería nativa de Android, "HttpUrlConnection" y, por otro lado, una librería de Google muy extendida para la realización de peticiones HTTP, "Volley".

Si comparamos las dos librerías, "HttpUrlConnection" es una librería que trabaja a un nivel más bajo que "Volley", lo que hace que esta última sea más simple de utilizar, especialmente en modo asíncrono. Aquí abajo se puede ver un ejemplo de código.

```
RequestQueue request = Volley.newRequestQueue(context);
StringRequest stringRequest = new StringRequest(Request.Method.POST,
    URL_STRING,
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {

        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {

        }
    }) {
    @Override
    public Map<String, String> getParams() {
        Map<String, String> params = new HashMap<String, String>();
        params.put(PARAMETRO_POST_1, valor1);
        params.put(PARAMETRO_POST_2, valor2);
        return params;
```

```
    }
};
request.add(stringRequest);
```

Si se analiza el código del recuadro superior, se puede ver que primero se inicializa una cola de peticiones, que es donde después se enviará la petición. Para ello, se crea la petición. Se utiliza el comando “StringRequest” y se elige el método POST de HTTP y la URL de destino. Dentro de esta petición hay un método “Response.Listener” que está escuchando a la espera de la respuesta correcta por parte del servidor, en este método se introduce el código para tratar la respuesta del servidor. El método “Response.ErrorListener” se ejecutará si hay un error con el servidor como puede ser un fallo de conexión. Por último, la función “Map” se utiliza para introducir los parámetros que se enviarán al servidor, para lo que se utiliza el comando “HashMap”, lo que relaciona pares clave-valor. Finalmente, la petición de envío se realiza en la última línea con el “request.add”, pasando como parámetro la “StringRequest”, lo que introduce la petición en la cola de peticiones.

A partir de aquí se empieza con los detalles del desarrollo. En primer lugar, hay que resaltar que esto va a ser una aplicación Android que va a tratar de unificar diversos servicios y bases de datos de la Policía Local de Mislata, por esta razón, la aplicación consta de una interfaz principal con un botón para cada uno de estos servicios bien diferenciados. A continuación, se muestra una imagen de dicha interfaz para una mejor explicación:

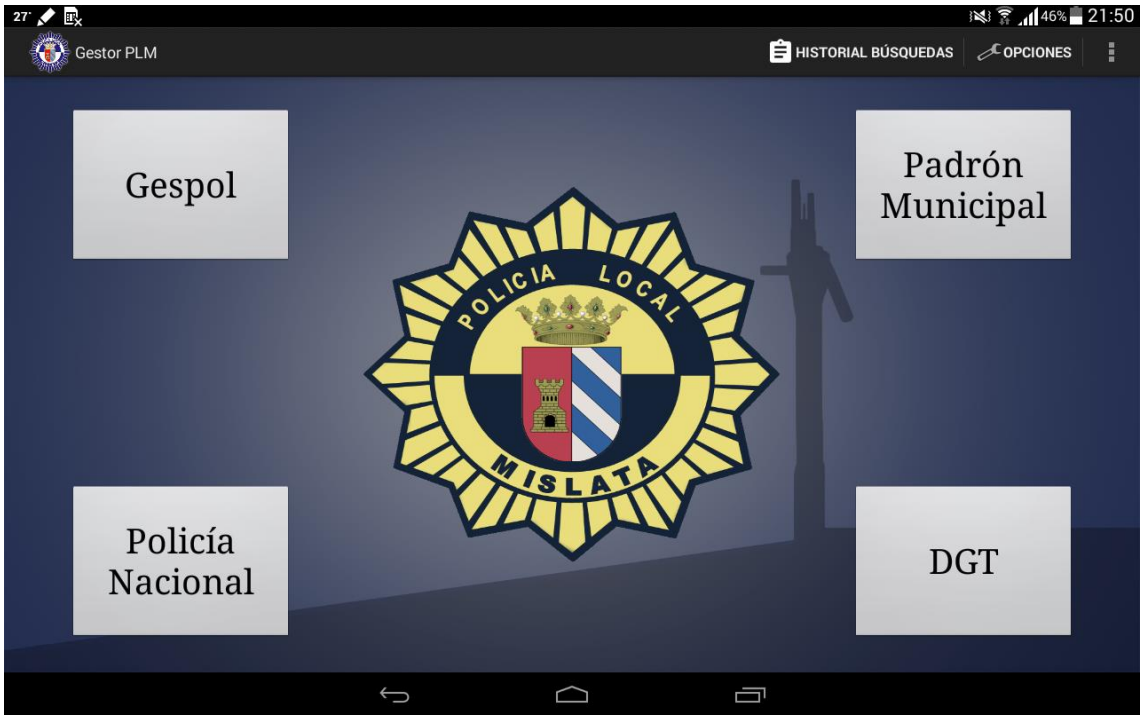

Ilustración 7: Interfaz principal Aplicación Android

En dicha imagen se pueden ver como imagen de fondo el escudo de la Policía Local de Mislata y al lado de éste, la sombra de una estatua muy característica de la localidad de Mislata. Con esto se espera dar un toque más personalizado para esta localidad (dicha medida ha sido muy bien acogida por parte del cuerpo policial).

Dejando de lado un poco la parte estética, en la imagen anterior llama la atención los 4 botones que rellenan las 4 esquinas de la pantalla. Estos botones reflejan los 4 diferenciados servicios que presta esta aplicación a parte del servicio de localización GPS. Estos servicios son: Gespol, el padrón municipal, policía nacional y la DGT.

- Gespol: Este botón da acceso a las diversas utilidades requeridas que conectan con la base de datos que almacena los datos del ERP de Gespol. Esta es la principal funcionalidad requerida por la Policía Local de Mislata.
- Padrón Municipal: Este apartado permite utilizar el servicio del padrón municipal y realizar consultas a su base de datos.
- Policía Nacional: Este botón conecta con el servicio web de la policía nacional.
- DGT: Este último botón enlaza con el servicio web de la Dirección General de Tráfico.

Además de estos botones principales, el menú posee un botón para visualizar un histórico de las consultas realizadas a las bases de datos de Gespol y del Padrón. También hay otro botón para cambiar los ajustes de la aplicación.

En el caso de que sea la primera vez que se utiliza la aplicación en un determinado dispositivo o dicho dispositivo no se encuentra registrado en el servidor, se abrirá una ventana de alerta pidiendo que el usuario introduzca un nombre para identificar fácilmente dicho dispositivo. Dicha ventana no puede ser cerrada hasta que se confirme en el servidor un registro correcto y sin duplicidades.

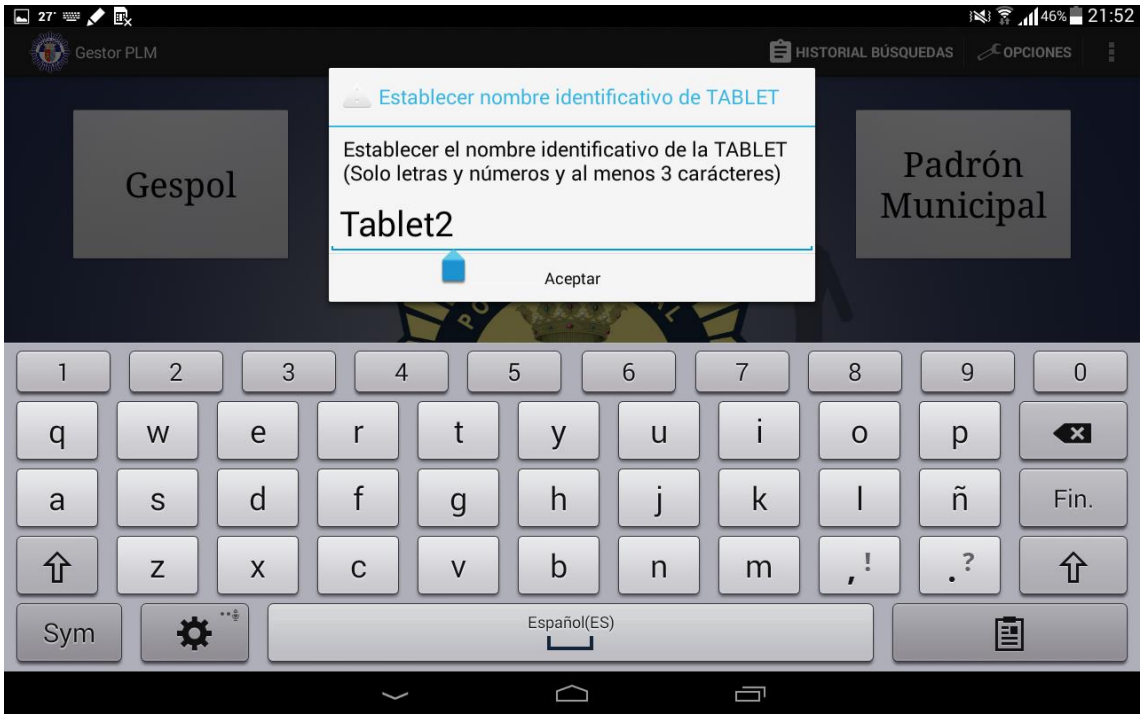

Ilustración 8: Registro del dispositivo

Como se menciona en el texto de la ventanita emergente, solo se pueden utilizar letras y números con al menos 3 caracteres, además, el máximo es de 20 dígitos y en mayúscula.

Para realizar esto a nivel de código, se ha utilizado el comando "DigitsKeyListener", el cual, hace que solo los caracteres especificados sean introducidos en el campo de texto. Para forzar que todos los caracteres estén en mayúscula se ha utilizado la función "InputType.TYPE_TEXT_FLAG_CAP_SENTENCES". Con el comando "InputFilter.LengthFilter(20)" se establece el máximo número de caracteres del campo de texto. Finalmente, para comprobar que el campo contenga el mínimo de 3 dígitos se utiliza un *if* comprobando la longitud del *String*.

El menú de opciones consta de 3 *CheckBox*:

- El primero de ellos es la auto búsqueda por voz. Esto es, que cuando el usuario realice una consulta por voz, automáticamente esta se enviará al servidor pasados un establecido número de segundos, durante el cual, el usuario podrá cancelar la consulta en caso de fallo de reconocimiento de voz.
- La segunda opción es para activar o desactivar la respuesta por voz generada por el dispositivo en respuesta a las consultas realizadas por voz.
- La última opción es para activar o desactivar las respuestas por voz cuando el usuario realiza una consulta manual, es decir, la ha introducido en modo texto.

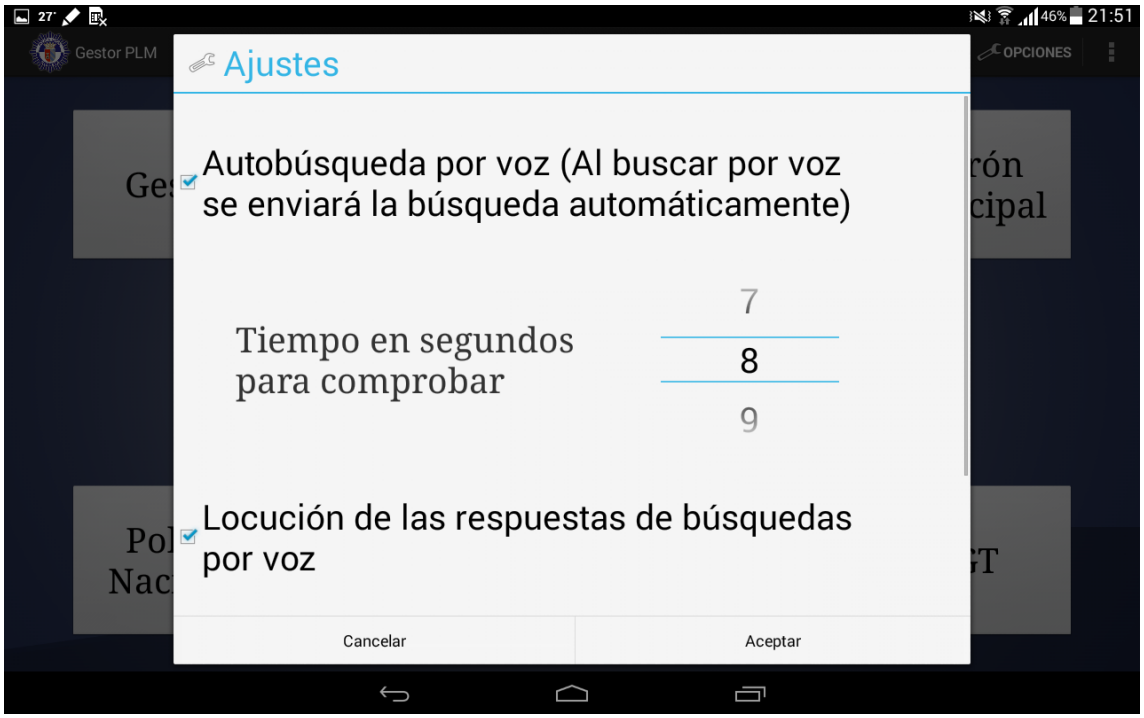

Ilustración 9: Menú opciones

Como se puede comprobar en la imagen anterior, el usuario dispone de un *NumberPicker* para seleccionar el número de segundos de comprobación deseado en caso de activar la auto búsqueda por voz. Una vez los datos se han aceptado, estos se guardarán en la base de datos de la aplicación, para lo cual se utiliza SQLite, que es la base de datos que se utiliza tradicionalmente en las aplicaciones de Android.

Si se desea ver el historial de consultas realizadas, basta con tocar sobre el botón que lo indica y aparecerá una pantalla con una tablita como se muestra en la siguiente ilustración:

	Consulta en:	Sección:	Búsqueda por:	Fecha:	Consulta:
1	GESPOL	CIUDADANOS	DNI	16/08/2017 13:07:50	53750292
2	PADRON		NOMBRE	16/08/2017 04:01:11	PABLO MORENO SIERI
3	GESPOL	CIUDADANOS	Apellidos	16/08/2017 02:46:56	MORENO SIERRA
4	PADRON		DNI	16/08/2017 02:14:35	53
5	GESPOL	CIUDADANOS	HABNUMIDE	16/08/2017 02:14:27	53
6	GESPOL	CIUDADANOS	Apellidos	16/08/2017 02:13:46	MORENO SIERRA
7	GESPOL	CIUDADANOS	APELLIDOS	16/08/2017 02:13:12	MORENO SIERRA
8	GESPOL	CIUDADANOS	APELLIDOS	16/08/2017 02:10:15	MORENO SIERRA
9	GESPOL	CIUDADANOS	HABNUMIDE	14/08/2017 01:18:42	53
10	GESPOL	CIUDADANOS	HABNUMIDE	14/08/2017 01:04:28	53
11	GESPOL	CIUDADANOS	HABNUMIDE	14/08/2017 01:04:25	53
12	GESPOL	CIUDADANOS	HABNUMIDE	14/08/2017 01:04:07	53
13	GESPOL	CIUDADANOS	HABNUMIDE	14/08/2017 01:04:05	53
14	GESPOL	CIUDADANOS	HABNUMIDE	14/08/2017 01:03:11	53
15	GESPOL	CIUDADANOS	HABNUMIDE	14/08/2017 00:44:54	53
16	GESPOL	CIUDADANOS	HABNUMIDE	14/08/2017 00:43:01	53
17	GESPOL	CIUDADANOS	HABNUMIDE	14/08/2017 00:42:45	53
18	GESPOL	CIUDADANOS	HABNUMIDE	14/08/2017 00:42:08	53
19	GESPOL	CIUDADANOS	HABNUMIDE	14/08/2017 00:42:02	53

Ilustración 10: Historia de consultas

Como se puede ver en la ilustración, se ha generado una tablita donde sale el historial de consultas, en concreto, se pueden visualizar las 100 últimas consultas. En esta tablita se puede ver a que base de datos se ha realizado la consulta, que es lo que se ha buscado en ella, el parámetro por el cual se ha realizado, la fecha exacta y, por supuesta el campo que introdujo el usuario en la consulta. Además, si el usuario desea volver a realizar una consulta anterior, bastaría con mantener presionado sobre el elemento deseado y el valor de la consulta se copiará al portapapeles del dispositivo. Para realizar la copia a nivel de código se ha realizado lo siguiente:

```
TableRow fila = new TableRow(getApplicationContext());
fila.setLayoutParams(layoutFila);
fila.setLongClickable(true);
fila.setOnLongClickListener(new View.OnLongClickListener() {
   @Override
   public boolean onLongClick(View v) {
       TextView textView =
               (TextView)((TableRow)v).getChildAt(5);
       String texto = textView.getText().toString();
       ClipboardManager clipboard = (ClipboardManager)
                getSystemService(Context.CLIPBOARD_SERVICE);
       clipboard.setPrimaryClip(
                ClipData.newPlainText("Dato búsqueda",texto));

  }
```

Si se analiza el código anterior, lo que se hace es utilizar la fila que es de tipo "TableRow" y darle la propiedad para que pueda realizarse sobre ella un clic largo, para posteriormente, utilizar un *listener* donde se introduce el código a ejecutar cuando se mantenga presionado el elemento. En este caso, se selecciona al elemento hijo con índice 5, esto es, la celda o columna número 5 de esta fila. Con esta celda seleccionada se obtiene el texto que contiene y se utiliza la función del "ClipboardManager" para guardar este texto en el portapapeles.

Ya con toda la interfaz principal explicada se puede proceder a la principal herramienta de esta aplicación, el acceso a Gespol.

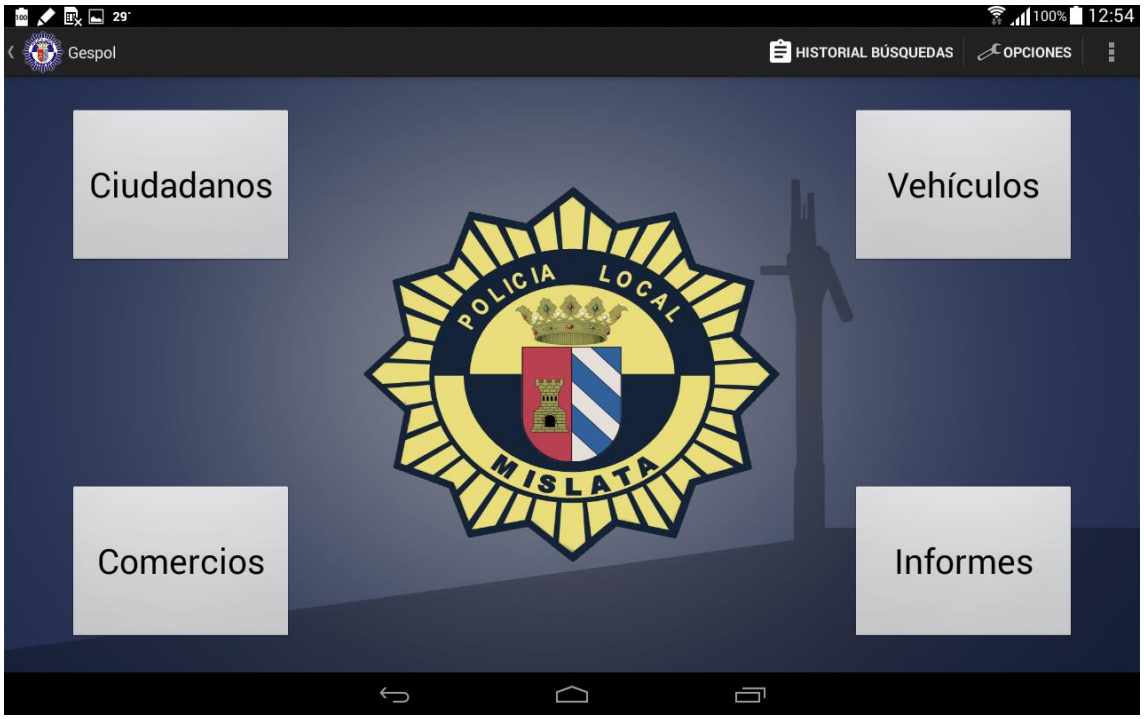

Ilustración 11: Menú Gespol

Esta interfaz como se puede observar en la imagen anterior es muy parecida a la del menú principal, ya que, consta del mismo menú y 4 botones, pero con una diferente funcionalidad. En este caso, los 4 botones principales son:

- Ciudadanos: Permite realizar consultas a la base de datos de Gespol para obtener datos de los ciudadanos.
- Vehículos: Para realizar consultas acerca de los vehículos.
- Comercios: Permite consultar los locales comerciales de la base de datos.
- Informes: Acceder a una pantalla que permite al agente consultar, editar y crear informes de novedades.

En la siguiente interfaz se muestra la pantalla donde se pueden acceder a los datos de los ciudadanos.

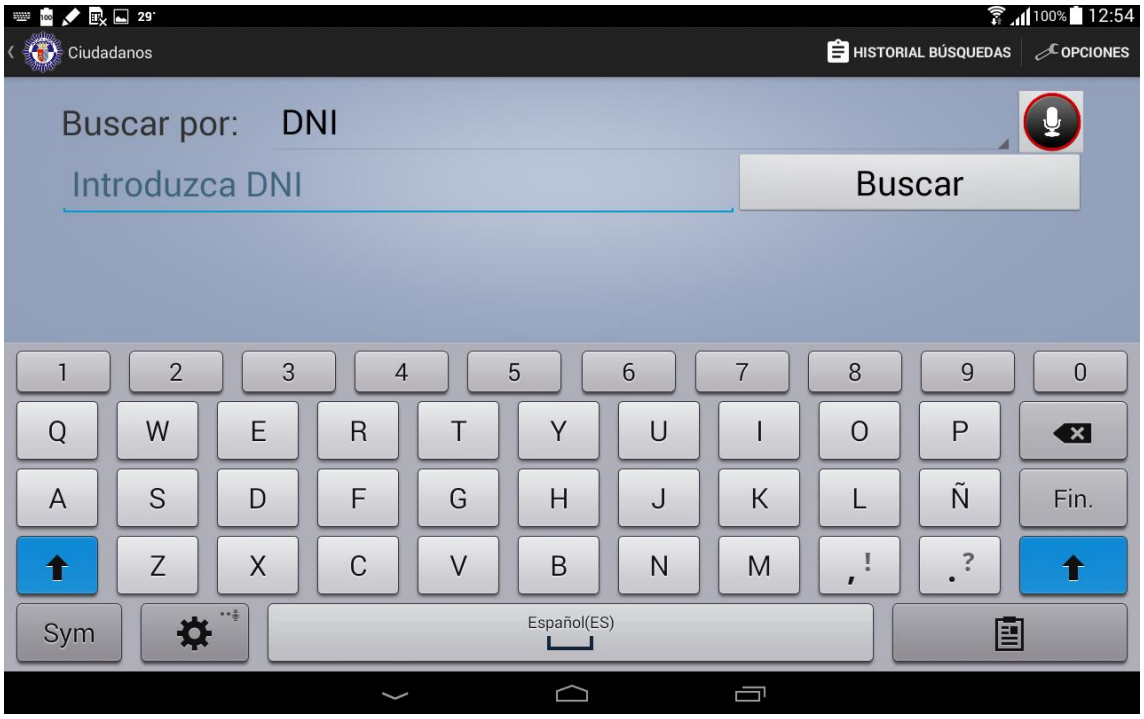

Ilustración 12: Consulta Ciudadanos

Como se puede ver en la ilustración anterior, la interfaz para buscar ciudadanos es sencilla. Esta contiene un desplegable en el cual se ha de marcar el parámetro por el cual se va a buscar el ciudadano, el cual puede ser, DNI o Apellidos. Debajo de este desplegable se dispone de una caja de texto donde introducir dicho DNI o apellidos. Con estos campos rellenados se puede utilizar el botón "Buscar" para realizar la consulta en base de datos a través de Internet. La interfaz, además, posee un botón con el símbolo de un micrófono encima del botón de búsqueda. Este botón abre una ventanita emergente donde se capturará la voz del usuario. Cuando se obtenga un resultado de búsqueda saldrá el DNI y nombre y apellidos del ciudadano en una lista en el hueco que hay debajo de la caja de texto de búsqueda. Se utiliza una lista ya que, si hay varios elementos coincidentes como puede ser dos hermanos que se apelliden igual, saldrán ambos elementos.

En caso de realizar una búsqueda por voz aparecerá una ventanita de confirmación de la consulta durante un tiempo determinado. Se puede apreciar en la siguiente ilustración:

Ilustración 13: Confirmación de búsqueda por voz

Una vez seleccionado un resultado se procederá a una nueva ventana donde saldrán todos los datos de éste en una lista.

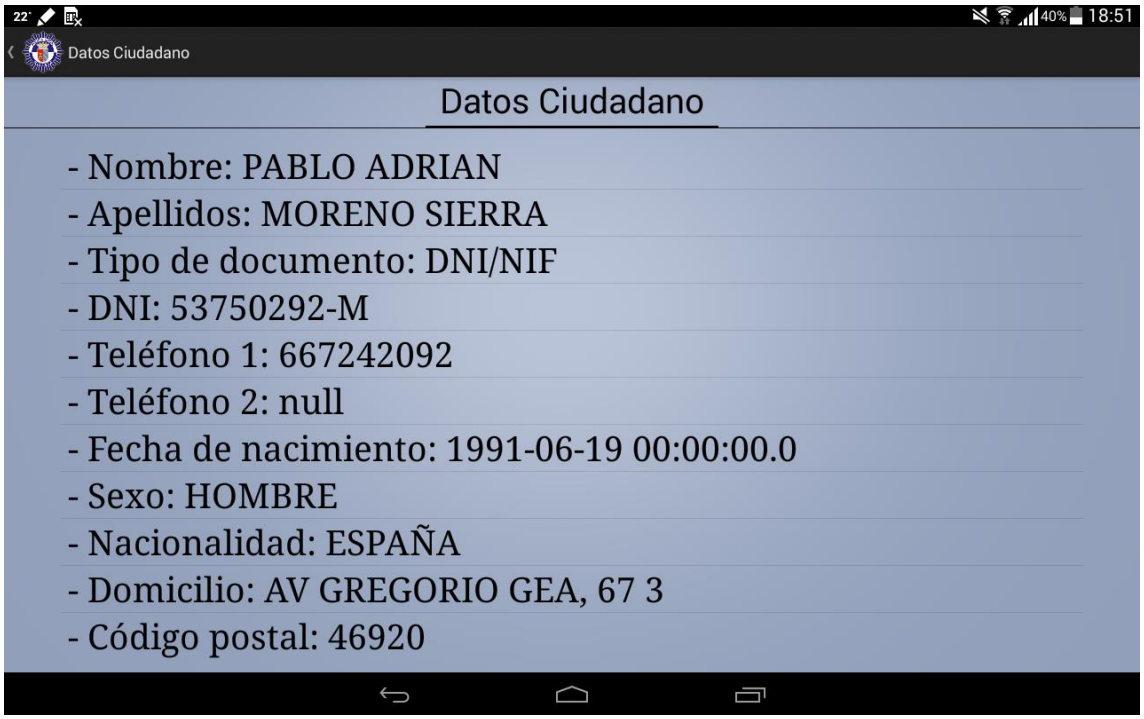

Ilustración 14: Detalles del ciudadano

En los apartados “Vehículos” y “Comercios” se dispone de una interfaz muy similar a la vista ahora, donde los únicos cambios son los diferentes parámetros de búsqueda, donde se pueden consultar vehículos por matrícula o por DNI de

propietario. En el caso de los comercios la búsqueda se podría realizar por CIF o nombre del local.

El apartado para consultar y crear informes consta de una ventana con dos pestañas, una de ellas, la de consulta, que sería similar a las vistas anteriormente, donde el usuario buscaría un informe por su código numérico. Por otro lado, si se desea generar un informe hay que rellenar los campos deseados del formulario que se muestra a continuación.

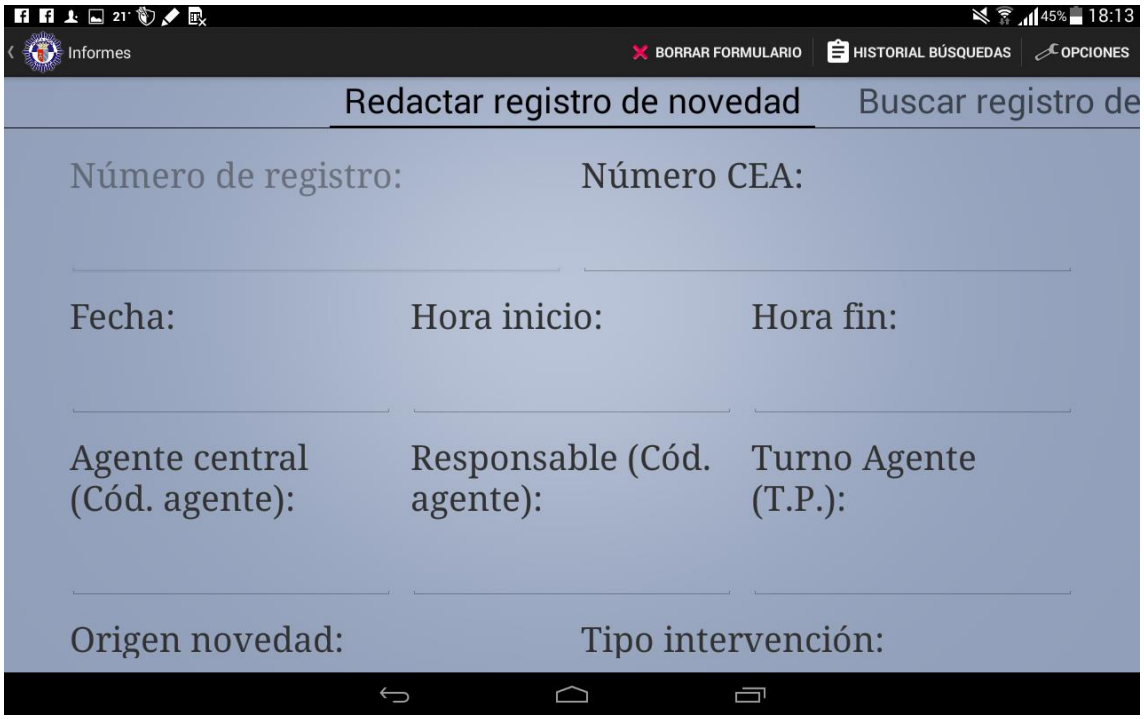

Ilustración 15: Creación de un informe de novedades

En la ilustración se puede, ver diferentes campos que se pueden rellenar para la creación de dicho informe. Estos campos coinciden con los que tiene el ERP de Gespol para generar las novedades. Cuando se realice una consulta de un informe se mostrará este formulario relleno con los datos pertinentes, además, se podrán modificar y actualizar en la base de datos los cambios realizados. En caso de querer borrar el formulario, hay un botón para ello en el menú superior.

En cuanto al apartador de la aplicación que conecta con la base de datos del padrón municipal se ha realizado de la misma manera que la de Gespol a nivel gráfico, por lo que es innecesario mostrar capturas de pantalla mostrándola. En cuanto a las opciones dentro de su interfaz, únicamente se posibilita la realización de consultas de los habitantes al igual que se hacía en Gespol.

Los apartados de DGT y Policía Nacional enlazan a través de una URL con el servicio web que poseen en sus respectivos servidores.

5.1. Desarrollo del reconocimiento de voz

El desarrollo del reconocimiento de voz ha sido algo complejo de realizar ya que hay que detectar palabras clave y después obtener el valor por ejemplo del DNI que se busca, de una forma correcta. Además, hay que tratar los diferentes reconocimientos que se producen, ya que, a partir de una frase oral, por si hay fallos, se devuelve un grupo de posibles frases interpretadas.

Para situarse, hay que decir que se está utilizando el sistema de reconocimiento de voz de Google ya que este funciona bastante bien y además viene incluido en el SDK de Android.

De cara al desarrollo, se ha declarado una función llamada "displaySpeechRecognizer", esta función lo que hace es llamar al servicio de reconocimiento de voz. En el siguiente cuadro se puede ver un ejemplo de esta función:

```
Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
                RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
startActivityForResult(intent, SPEECH_REQUEST_CODE);
```

Como se ve en el cuadro, se utiliza un "intent" hacer un "intento" de reconocimiento de voz. Se establecen también dos parámetros extra, que son: "EXTRA_LANGUAGE_MODEL, para poder especificar un modo de lenguaje. Con el parámetro "LANGUAGE_MODEL_FREE_FORM" elegimos el modo de lenguaje libre, ya que la única alternativa a este sería el modo para búsqueda en web y, no es lo que se busca. Finalmente, con la función "startActivityForResult" se realiza el "intento" y se espera un resultado que debe llegar en la función "onActivityResult". En dicha función es donde se debe tratar el resultado del reconocimiento de voz.

La función "onActivityResult" es una función predefinida por lo que hay que utilizar un "@Override" para ampliar el código de esta función de manera que podamos tratar el resultado obtenido por el reconocimiento de voz. A continuación, se muestra cómo quedaría la función reformada.

```
if (requestCode == SPEECH_REQUEST_CODE && resultCode == RESULT_OK) {
     List<String> results =
         data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
     tratarReconocimientoVoz(results);
}
super.onActivityResult(requestCode, resultCode, data);
```

Lo primero que se realiza es un *if* donde se comprueba el código de la petición, donde se verifica que esta sea la del reconocimiento de voz y, además, que esta sea correcta. En segundo lugar, se utiliza una lista de *Strings* que será donde se guardarán los resultados obtenidos y se enviarán a otra función donde se analizarán y tratarán los datos obtenidos.

En este momento hay que analizar los datos obtenidos, primero hay que matizar que esta lista de *Strings* que se ha obtenido contiene 5 interpretaciones posibles de una escucha de voz. Por lo que según los datos que se estén esperando hay que buscar unos datos u otros. Además, a la hora de comparar las cadenas obtenidas con los valores clave buscados, hay que eliminar las tildes y unificar mayúsculas y minúsculas. En este caso, aprovechando que la base de datos contiene todos los datos en mayúscula, se va a aprovechar para pasar los resultados obtenidos a mayúscula y tratarlo todo en mayúscula desde este momento. Para convertir los datos a mayúscula basta con utilizar la función "toUpperCase()" del *String* seleccionado, y dicho *String* convertirá todas las letras a mayúscula. Tras el siguiente código se explica cómo se soluciona la comparación con las tildes.

```
Collator instance = Collator.getInstance();
instance.setStrength(Collator.NO_DECOMPOSITION);
if(instance.equals(dato1,dato2)){}
```

En Java se dispone de la clase "Collator" que en este caso se utiliza para cambiar la "fuerza" de la comparación, de manera que se ignoren las tildes en dicha comparación. En la última línea del código se puede ver como se realiza una comparación entre dos cadenas de este modo. Otra opción que puede ser, conveniente según el caso, es reemplazar los caracteres con tilde por los propios caracteres sin la tilde. Para esto se podría utilizar el siguiente comando:

```
Normalizer.normalize(texto,
                Normalizer.Form.NFD).replaceAll("[^\\p{ASCII}]", "");
```

Una vez se tiene esto en cuenta, hay que analizar los resultados obtenidos en busca de las palabras clave. Por ejemplo, se esperan frases de entrada como: "Buscar el DNI 12345678X", "Encuéntrame datos del CIF X12345678Y" o "Buscar los apellidos Moreno Sierra". Si uno se fija en estas frases, hay algo en común, resulta que la palabra clave está justo a la izquierda de valor buscado. Por lo que esto es un buen punto de partida para el análisis de las órdenes de voz.

Para encontrar el parámetro en una orden de voz, por el cual se desea buscar, es tan fácil como utilizar la función “indexOf()” del *String* de manera que se introduzca como parámetro de dicha función el valor que se desea buscar, véase, “DNI”, “Apellido” o cualquier otro. Dicha función devuelve el índice donde comienza el parámetro buscado. Por lo que si se utiliza esta función y se le suma la longitud del parámetro y posteriormente se utiliza la función “split()”, la cual, separa el *String* sobre el que se usa dicha función en un *array* de *Strings*, utilizando como parámetro de separación lo que se le dé por parámetro al “split”. Es decir, si realizamos un “split” sobre la cadena “DNI 12345678” y como parámetro se introduce un espacio en blanco, (“ “), separará dicha cadena en dos, tal que, “DNI” y “12345678”. De esta manera es como se va a realizar la búsqueda y separación del parámetro y valor interpretados por el reconocedor de voz.

Pero no es tan simple el asunto, porque toca recordar que el reconocedor de voz obtiene 5 resultados distintos (aunque varios de estos pueden ser válidos para esta solución). Lo que se va a realizar es, utilizar un bucle que recorra las 5 interpretaciones y se utilice la primera que sea válida en todos los aspectos, es decir, que obtenga correctamente el parámetro sobre el cual buscar y el valor del mismo. Además, cuando se encuentra un parámetro, como, por ejemplo, “DNI” o “Apellidos”, este se selecciona en el desplegable donde el usuario debe seleccionar el parámetro de búsqueda, por lo que, en caso de control de voz, dicho desplegable se selecciona automáticamente en caso de encontrar un parámetro, en caso de no encontrarlo, la búsqueda se realizaría sobre el parámetro que hubiera seleccionado previamente en el desplegable.

A la hora de buscar el valor correcto, primero hay que tener en cuenta que tipo valor buscamos, véase, si se está buscando un apellido o el nombre de un establecimiento basta con obtener el trozo de cadena siguiente al parámetro correspondiente. Pero, si por el contrario se trata de un DNI o una matrícula entre otras cosas, hay que asegurarse de varios factores, como es, que los valores obtenidos sean numéricos o parcialmente numéricos, por lo que, en primer lugar, ha de comprobarse que haya valores numéricos (el reconocedor de voz, en varias de las interpretaciones proporcionadas, obtiene dicho número, pero escrito en forma de palabra), pero tampoco hay que olvidar que dicho valor puede empezar y/o terminar por letras. Por ejemplo, según que matrícula puede empezar o acabar con letras, e igualmente para el DNI, que para un extranjero sería el NIF, el cual, comienza por una letra.

En casos como la búsqueda de un DNI o matrículas también se ha implementado una opción de un gran valor añadido, que es el permitir que dicha consulta pueda ser utilizando el alfabeto radiofónico de manera automática. Esto es, que cada letra es equivalente a una palabra en este alfabeto, tal que, la palabra “Alfa” equivale a la “A”, “Bravo” la “B”, y así con todas las letras a excepción de la “Ñ”, ya que esta letra es más exclusiva del alfabeto

castellano. Además, está desarrollado de manera que el usuario no ha de especificar la forma de entrada de los datos, aceptará automáticamente tanto el alfabeto radiofónico como el convencional. Esto es un gran valor añadido debido a que, este alfabeto es el que utiliza la policía generalmente para transmitir a la central los datos de cualquier consulta que deseen realizar.

Para realizar esto, en casos como el DNI que pueda haber letras y números entremezclados, automáticamente si se detecta una palabra como la del alfabeto radiofónico, ésta será reemplazada por su equivalente.

5.2. Desarrollo del sistema de localización GPS

El sistema de localización es una parte también algo costosa ya que hay que crear un servicio, analizar y comparar las ubicaciones para comparar su exactitud, y ver si es necesario actualizarla o no, además de auto-arrancar este servicio al iniciar el dispositivo, entre otras cosas.

En primer lugar, hay que crear un servicio, para esto hay que crear una clase java y extender de la clase “Service. Por otro lado, es necesario ir al manifiesto de Android y agregar dicha clase como un servicio. Para hacer esto, se agregaría el siguiente código al fichero “AndroidManifest.xml”.

```
<service
     android:name=".Localizacion"
     android:enabled="true"
     android:exported="true"
/>
```

También es necesario añadir al manifiesto una serie de permisos para poder acceder a la localización, estos son:

```
<uses-permission
       android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-feature android:name="android.hardware.location.gps" />
```

A continuación, hay que seguir desarrollando la clase Java que se ha creado como servicio. En la clase creada será necesario declarar un objeto “LocationManager” e iniciarlo a *null*, para posteriormente poder gestionar la localización. Dentro de esta clase, habrá que crear una clase privada llamada “LocationListener” que implemente “android.location.LocationListener”, ya esta será la clase que “escuche” los datos del GPS. Dentro de dicha clase se declarará un objeto de tipo “Location” donde se guardará la última ubicación para compararla con la nueva. Esta clase privada tendrá un constructor

"LocationListener" que como parámetro tendrá un *String* que deberá ser un proveedor de GPS, este puede ser, del propio GPS o de la red móvil. En el siguiente código se puede ver cómo quedaría lo mencionado anteriormente:

```
public class Localizacion extends Service {

    private LocationManager mLocationManager = null;

    private class LocationListener implements
                                    android.location.LocationListener {
        Location mLastLocation;

        private LocationListener(String provider) {
            mLastLocation = new Location(provider);
        }
    }
}
```

Dentro de la clase "LocationListener" hay que sobrescribir la función "onLocationChanged" que viene la función abstracta con el mismo nombre, por lo que requiere de un "@Override". En esta función es donde se va a comparar la nueva localización con la anterior, y en caso de la nueva ser mejor se actualizará. Más adelante se mostrará cómo se realiza dicha comparación. A continuación, se muestra como resultaría esta función.

```
@Override
public void onLocationChanged(Location location) {
    if (isBetterLocation(location,mLastLocation)) {
        mLastLocation.set(location);
    }
}
```

La función que va a comparar la localización anterior y la nueva recibe ambas localizaciones por parámetro.

- Lo primero es comprobar si hay una localización antigua, en caso contrario, se asigna la nueva.
- En caso de que hubiera una localización anterior, se calcula la diferencia de tiempo entre ambas. En caso de que el tiempo haya aumentado considerablemente, la ubicación se actualiza. En caso de que la nueva localización no sea realmente más nueva, la actualización no se realiza.
- El siguiente paso se realiza si los tiempos de ambas ubicaciones son similares. En este caso se trata de comparar la precisión de estas. Para realizar esto se utiliza la función "getAccuracy" del objeto "Location". Si

la precisión de la nueva localización es más precisa o la nueva es un poco menos precisa, pero es más nueva o la nueva es considerablemente menos precisa, pero es más nueva, y a la vez es del mismo proveedor (el proveedor se comprueba comparando si el resultado de la función "getProvider" del objeto "Location" es igual para ambas localizaciones)., la ubicación se actualizará.

- En cualquier otro caso, no se actualizará.

En el siguiente código se puede ver cómo quedaría la función para comparar las dos localizaciones.

```
private boolean isBetterLocation(Location location, Location
                                        currentBestLocation) {
    if (currentBestLocation == null) {
        return true;
    }

    long timeDelta = location.getTime()-currentBestLocation.getTime();
    boolean isSignificantlyNewer = timeDelta >
                               LOCATION_INTERVAL_WITHOUT_MOVEMENT;
    boolean isSignificantlyOlder = timeDelta < -
                               LOCATION_INTERVAL_WITHOUT_MOVEMENT;
    boolean isNewer = timeDelta > 0;

    if (isSignificantlyNewer) {
        return true;
    }else if (isSignificantlyOlder) {
        return false;
    }
    int accuracyDelta = (int) (location.getAccuracy() -
                               currentBestLocation.getAccuracy());
    boolean isLessAccurate = accuracyDelta > 0;
    boolean isMoreAccurate = accuracyDelta < 0;
    boolean isSignificantlyLessAccurate = accuracyDelta > 200;

    boolean isFromSameProvider =
                          isSameProvider(location.getProvider(),
                          currentBestLocation.getProvider());

    if (isMoreAccurate) {
        return true;
    } else if (isNewer && !isLessAccurate) {
        return true;
    } else if (isNewer && !isSignificantlyLessAccurate &&
                isFromSameProvider) {
        return true;
    }
    return false;
}
```

A parte de la clase privada implementada dentro de la clase de tipo servicio, esta clase servicio también necesita sobrescribir las funciones "onCreate()" y "onDestroy()". En el siguiente código se muestra la función "onCreate", en dicha

función se inicializa el sistema de gestión del GPS y, se asigna un "listener" para recibir las actualizaciones de ubicación. Además, se trata de un "listener" para cada proveedor de GPS, se utiliza "mLocationListeners[0])" el proveedor tipo GPS, y "mLocationListeners[1])" para el proveedor tipo red móvil.

```
@Override
public void onCreate(){

 if (mLocationManager == null) {
   mLocationManager = (LocationManager)
   getApplicationContext().getSystemService(Context.LOCATION_SERVICE);
 }
 try {
     mLocationManager.requestLocationUpdates(
        LocationManager.GPS_PROVIDER, LOCATION_INTERVAL,
        LOCATION_DISTANCE,
                mLocationListeners[0]);
 } catch (java.lang.SecurityException ex) {
            Log.i(TAG, "fallo de actualización, ignore", ex);
    } catch (IllegalArgumentException ex) {
       Log.d(TAG, "No existe el proveedor GPS " + ex.getMessage());
         try {
            mLocationManager.requestLocationUpdates(
                        LocationManager.NETWORK_PROVIDER,
                        LOCATION_INTERVAL, LOCATION_DISTANCE,
                        mLocationListeners[1]);
         }catch (java.lang.SecurityException e) {
            Log.i(TAG, "fallo de actualización", e);
         }catch (IllegalArgumentException e) {
            Log.d(TAG, "No existe el proveedor GPS "+ e.getMessage());
         }
  }
}
```

En el "onDestroy", se utiliza la función "removeUpdates" para todas las instancias de la clase "LocationManager" que estén iniciadas. Con dicha función se cancela la recepción de actualizaciones a esas instancias.

```
@Override
public void onDestroy() {
    super.onDestroy();
    if (mLocationManager != null) {
        for (int i = 0; i < mLocationListeners.length; i++) {
            try {
                mLocationManager.removeUpdates(mLocationListeners[i]);
            } catch (SecurityException ex) {
                Log.i(TAG, "fallo al eliminar actualizaciones", ex);
            }
        }
    }
}
```

Con esto ya estaría el sistema de localización funcional, pero, falta un detalle muy importante. Este servicio ha de ejecutarse siempre, no únicamente cuando la aplicación este en marcha. Para ello, habrá que configurarla aplicación para que este servicio se inicie automáticamente al iniciar el sistema operativo Android.

Para conseguir esto, lo primero que se ha de hacer es añadir al manifiesto de Android el permiso necesario para poder recibir una notificación de que el sistema Android ha arrancado, a partir de esa notificación del sistema operativo ya se puede arrancar el servicio. El código necesario para añadir el permiso al manifiesto sería:

```
<uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
```

Una vez se posee el permiso requerido, es necesario crear también en el propio manifiesto, un "receiver", que va a ser lo que va a manejar la recepción de la notificación del sistema operativo de que éste ha arrancado. El código necesario se muestra en el siguiente recuadro.

```
<receiver
      android:name=".BootLauncher"
      android:enabled="true"
      android:exported="true">
      <intent-filter>
           <action
                android:name="android.intent.action.BOOT_COMPLETED" />
      </intent-filter>
</receiver>
```

El Código anterior lo que hace exactamente es permitir a una clase llamada "BootLauncher", el recibir las notificaciones de arranque. Además también hará falta añadir al manifiesto un servicio que va a ser el que ejecute el "BootLauncher".

```
<service
    android:name=".StarterService"
    android:enabled="true"
    android:exported="true">
</service>
```

Con el código anterior ya se podría crear un servicio cuya clase tenga de nombre "StarterService".

Con esto ya se ha completado la parte necesaria en el manifiesto. En siguiente lugar hay que realizar la clase recibidora "BootLauncher". Este fichero debe extender de la clase "BroadcastReceiver" y sobrescribir la función "onReceive" donde se utilizará un "Intent" para iniciar el servicio "StarterService" mencionado anteriormente. A continuación, se muestra como resultaría dicha clase recibidora.

```
public class BootLauncher extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent i) {
        Intent intent = new Intent(context, StarterService.class);
        context.startService(intent);
    }
}
```

Finalmente, el servicio "StarterService" es el que se ocupa de iniciar las clases deseadas de la aplicación. En este caso, se mandará iniciar la clase "Localizacion.class" como una tarea, por lo que, desde dicho momento, el servicio de localización estará activo. Si también se deseara ejecutar la actividad principal de la aplicación se realizaría de la misma manera.

```
public class StarterService extends Service {

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Intent intents = new Intent(getBaseContext(),
                            Localizacion.class);
        intents.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        startService(intents);
        return START_STICKY;
    }
}
```

Con esto ya se tiene un sistema de localización en Android siempre activo desde que se inicia el propio Android.

6. Validaciones y pruebas

Debido a la LOPD (ley orgánica de protección de datos), estas pruebas son realizadas utilizando datos de prueba ficticios y tratando de asemejarse a los reales. Por lo que, en las bases de datos de prueba generadas, se han insertado una serie de datos para llevar a cabo dichas pruebas.

En primer lugar, se han realizado pruebas de uso tratando los típicos problemas que pueden ocurrir. Primero se ha realizado una búsqueda sin tener conexión a Internet. Esto ha dado un error de falta de conexión a la red.

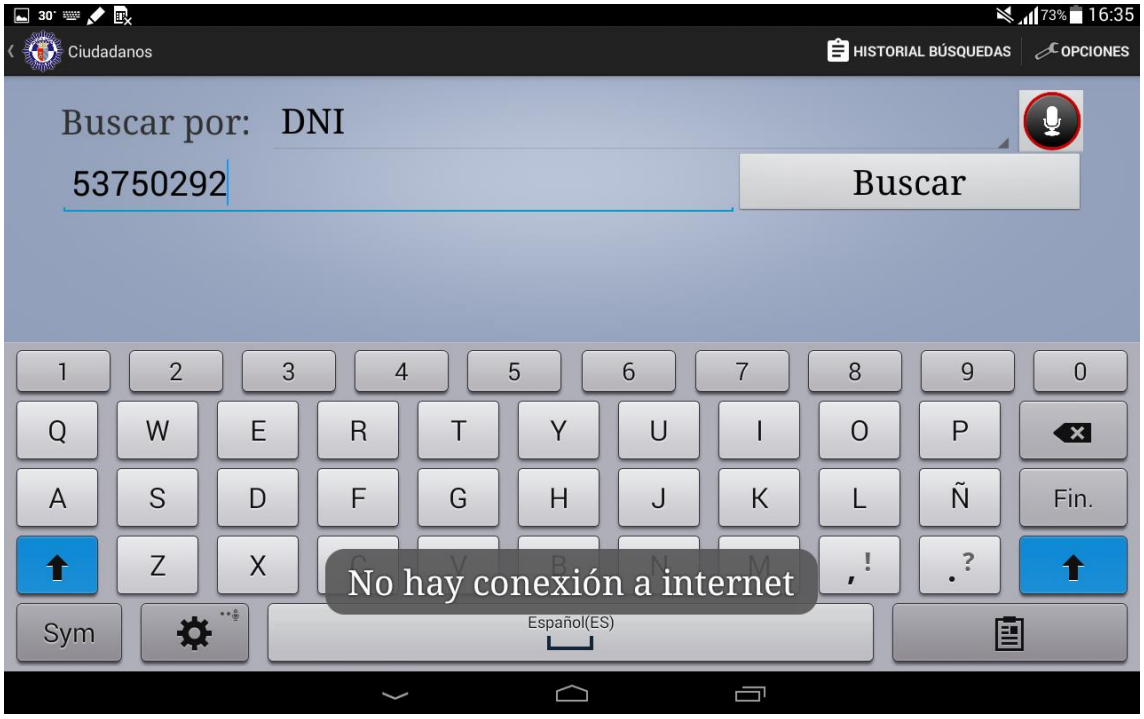

Ilustración 16: Aplicación, fallo de conexión

En segundo lugar, se ha tratado de acceder a la base de datos mientras el servidor estaba desconectado. Esto ha provocado un mensaje indicando que se ha producido un error con el servidor.

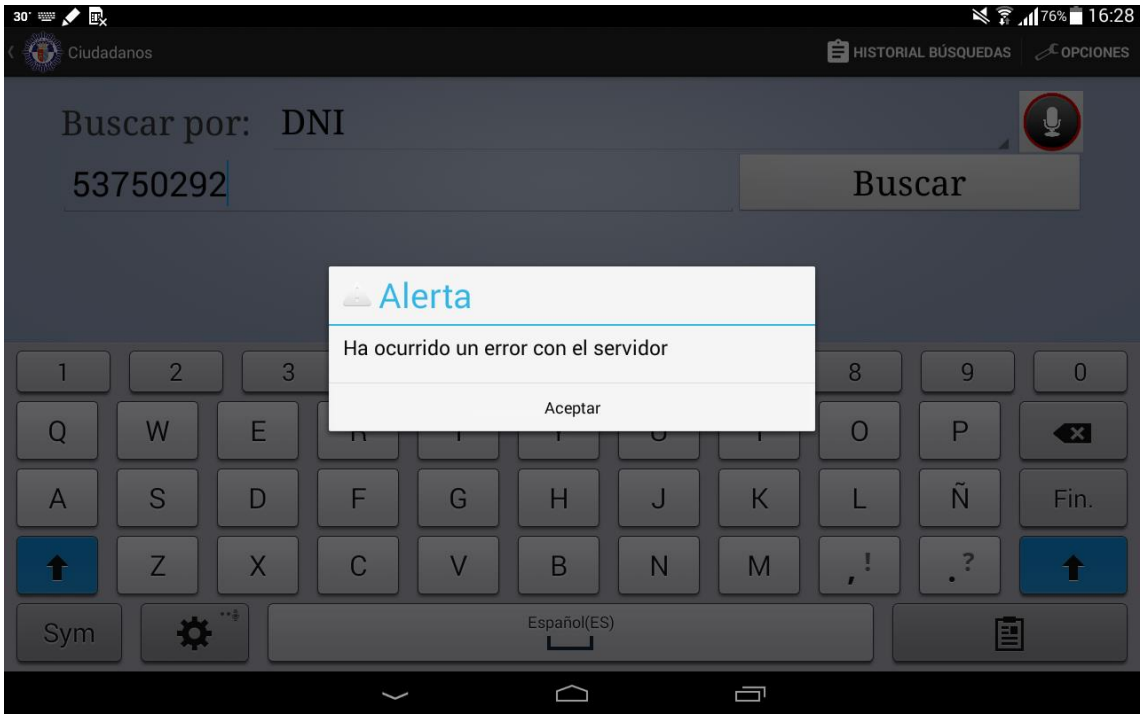

Ilustración 17: Aplicación, error de servidor

Lo siguiente, ha sido realizar una búsqueda con un dato no almacenado en la base de datos. Lo que ha resultado con un mensaje indicando que no se ha encontrado ningún resultado con esa búsqueda.

Ilustración 18: Aplicación, sin resultados

Como siguiente instancia, se ha tratado de introducir un código que provoque una inyección SQL. Como resultado no se ha provocado ningún cambio en la base de datos y se ha realizado una consulta sin encontrar resultados.

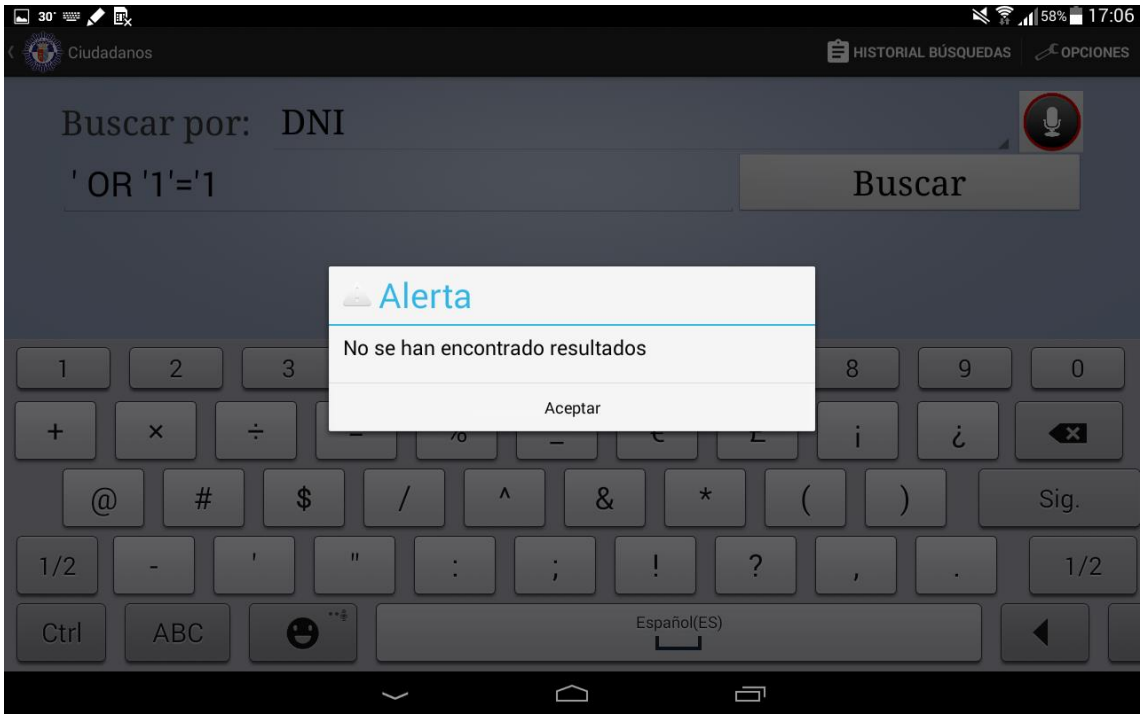

Ilustración 19: Aplicación, inyección SQL

La siguiente opción ha sido tratar de enviar una consulta vacía, ya que esto daría un trabajo innecesario a la base de datos y a la conexión de red. Al realizar esto se obtiene un mensaje advirtiendo de que no se ha introducido ningún valor de búsqueda y, por tanto, la consulta no se ha realizado.

Ilustración 20: Aplicación, sin valor de búsqueda

Finalmente, se ha introducido correctamente un valor de búsqueda que se puede encontrar en la base de datos. El resultado ha sido el esperado, mostrar una lista con dicho elemento.

Ilustración 21: Aplicación, buscando

Ilustración 22: Aplicación, resultado ciudadano

Ilustración 23: Aplicación, resultado vehículo

A continuación, se ha seleccionado dicho elemento para ver sus detalles, y estos se muestran adecuadamente.

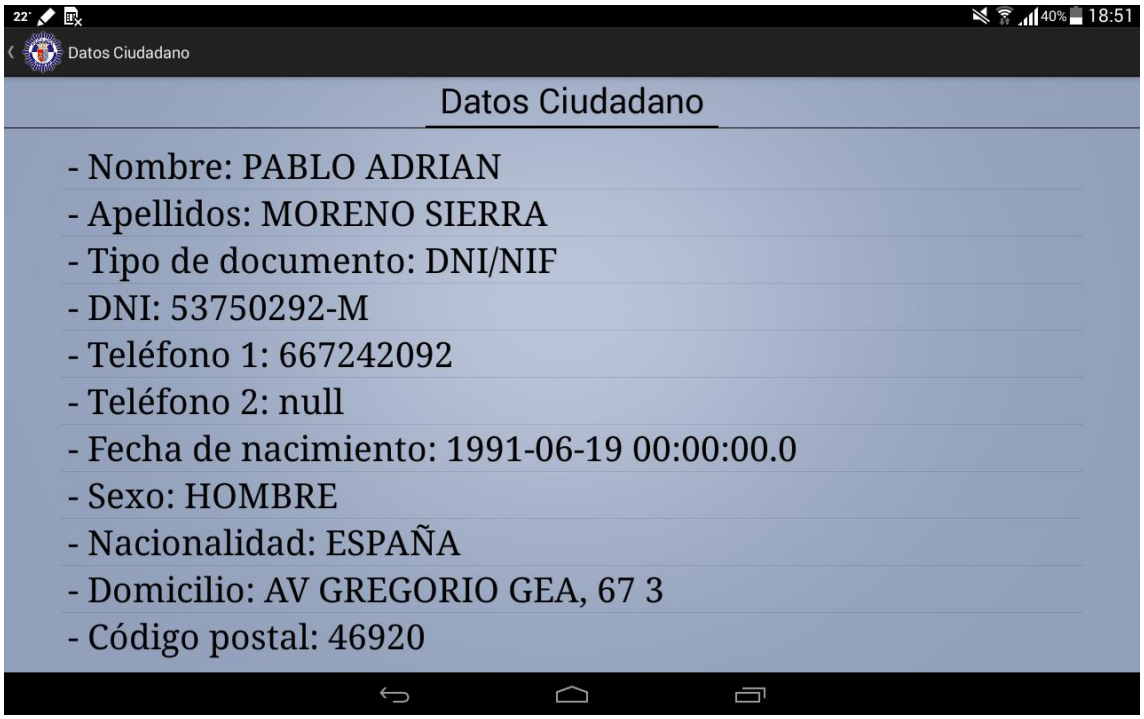

Ilustración 24: Aplicación, detalles de ciudadano

La siguiente prueba ha sido el utilizar un valor de búsqueda que pueda tener más de un elemento resultando. Como resultado se ha obtenido una lista con los elementos coincidentes, pudiendo pulsar sobre cualquiera de ellos para obtener sus datos detallados.

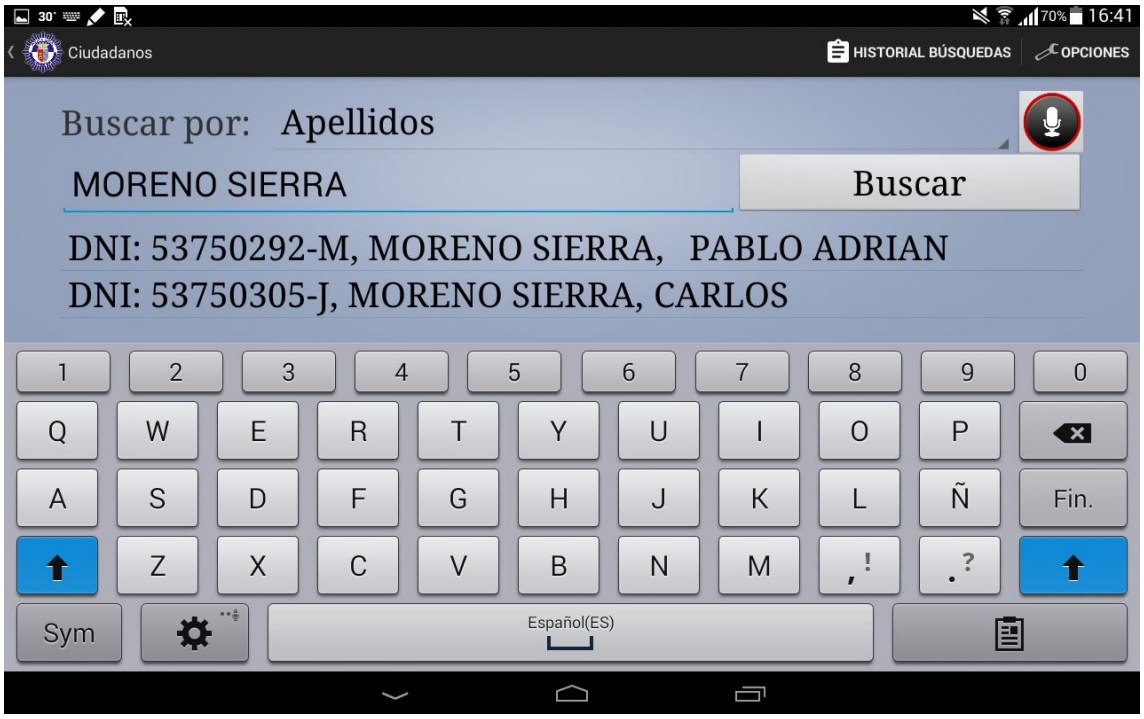

Ilustración 25: Aplicación, varios resultados

En cuanto a la creación de informes, estos se pueden generar correctamente a partir de los valores introducidos y, una vez generado, estos informes se pueden buscar y editar.

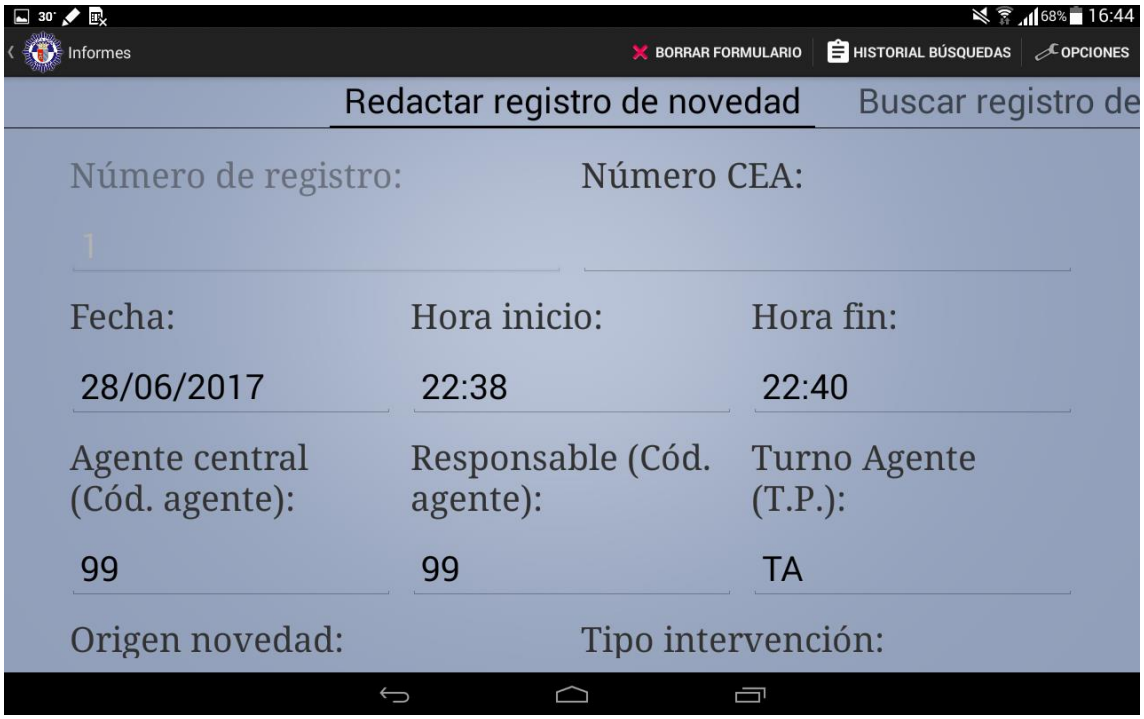

Ilustración 26: Aplicación, edición de informes

Todas estas pruebas se han realizado para todas las interfaces y posibilidades de consulta de la aplicación y han obtenido el mismo resultado.

La elección de un nombre de dispositivo se puede elegir correctamente, devolviendo un error y obligando al usuario a elegir un nombre único en caso de que dicho nombre ya esté registrado.

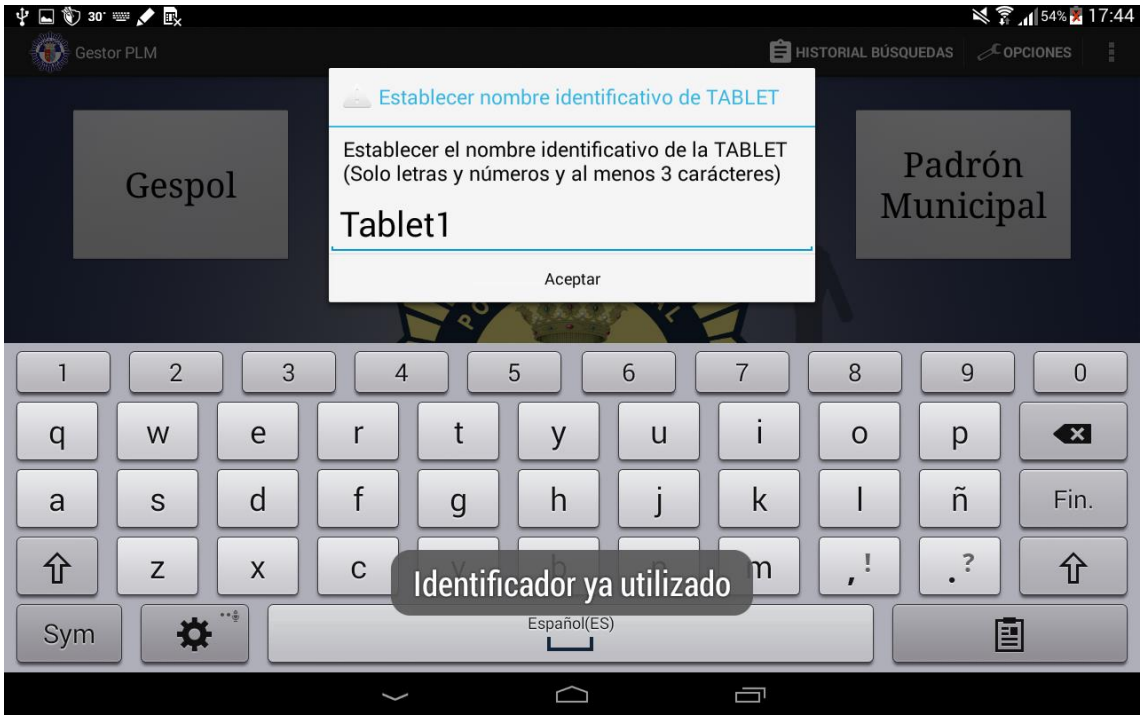

Ilustración 27: Aplicación, identificador ya utilizado

Los parámetros del menú de opciones se guardan correctamente.

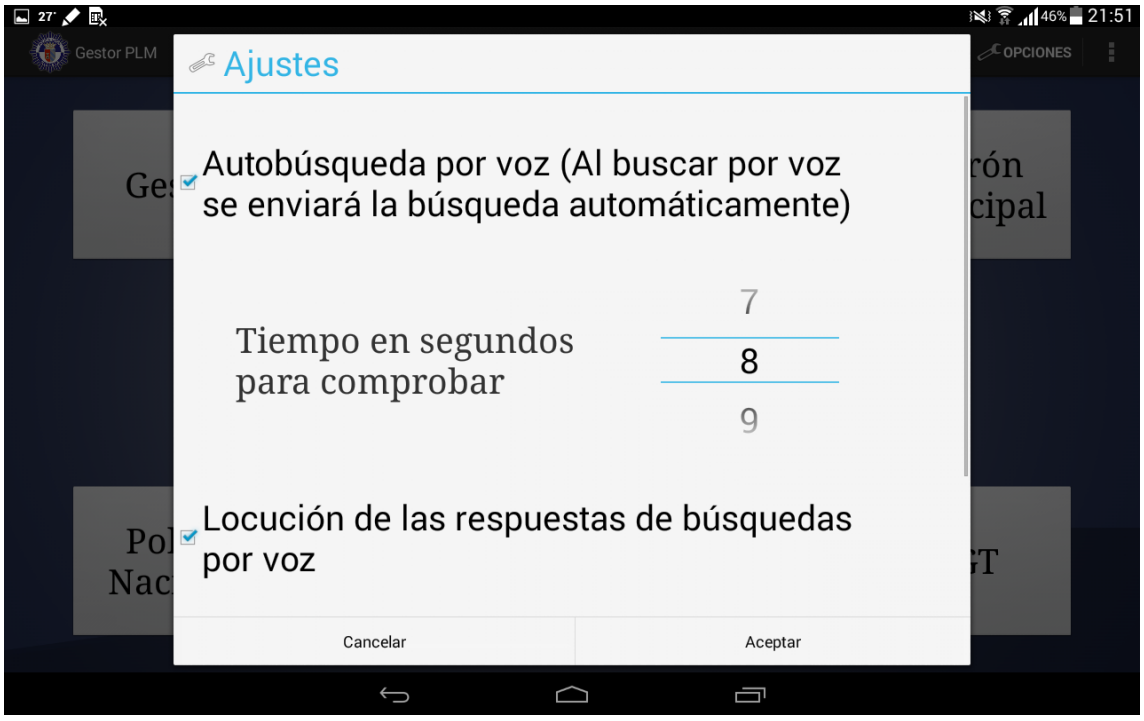

Ilustración 28: Aplicación, opciones

El historial de búsquedas muestra correctamente una tablita con los datos consultados.

	Consulta en:	Sección:	Búsqueda por:	Fecha:	Consulta:
1	GESPOL	CIUDADANOS	DNI	04/09/2017 17:06:23	' OR '1'='1
2	GESPOL	CIUDADANOS	DNI	04/09/2017 16:49:36	53750292
3	GESPOL	CIUDADANOS	DNI	04/09/2017 16:48:45	53750292
4	GESPOL	CIUDADANOS	DNI	04/09/2017 16:48:34	53750292
5	GESPOL	CIUDADANOS	DNI	04/09/2017 16:47:36	53750292
6	GESPOL	CIUDADANOS	DNI	04/09/2017 16:47:33	53750292
7	PADRON		DNI	04/09/2017 16:46:26	56678
8	GESPOL	CIUDADANOS	DNI	04/09/2017 16:46:01	344T5
9	PADRON		DNI	04/09/2017 16:45:50	53750292
10	PADRON		DNI	04/09/2017 16:45:47	53750292
11	PADRON		DNI	04/09/2017 16:45:45	53750292
12	PADRON		DNI	04/09/2017 16:45:40	53750292
13	GESPOL	BUSCAR NOVEDADES	CODIGO	04/09/2017 16:44:37	1
14	GESPOL	COMERCIOS	NOMBRE	04/09/2017 16:43:26	CHIPIPABLO
15	GESPOL	COMERCIOS	NOMBRE	04/09/2017 16:43:20	CHIPI
16	GESPOL	VEHICULOS	DNI	04/09/2017 16:43:07	53750292
17	GESPOL	VEHICULOS	MATRICULA	04/09/2017 16:42:29	V-8972-HF
18	GESPOL	CIUDADANOS	APELLIDO	04/09/2017 16:41:43	PEDRERO
19	GESPOL	CIUDADANOS	APELLIDO	04/09/2017 16:41:06	MORENO SIERRA

Ilustración 29: Aplicación, historial

En cuanto a los botones de acceso a DGT y a la policía Nacional enlazan correctamente con su servicio web correspondiente, pero no se muestra debido a la sencillez de la realización y a la privacidad de estos sitios web.

Finalmente, se han hecho pruebas de ubicación. Para ello se ha accedido a la interfaz web desarrollada para visualizar el mapa de Mislata donde se encuentra la ubicación de los dispositivos mediante unos marcadores. Además, si estos marcadores son pulsados, una ventanita muestra todos sus datos como la fecha exacta de ubicación, nombre del dispositivo, la dirección donde este se encuentra o el identificador único de este dispositivo.

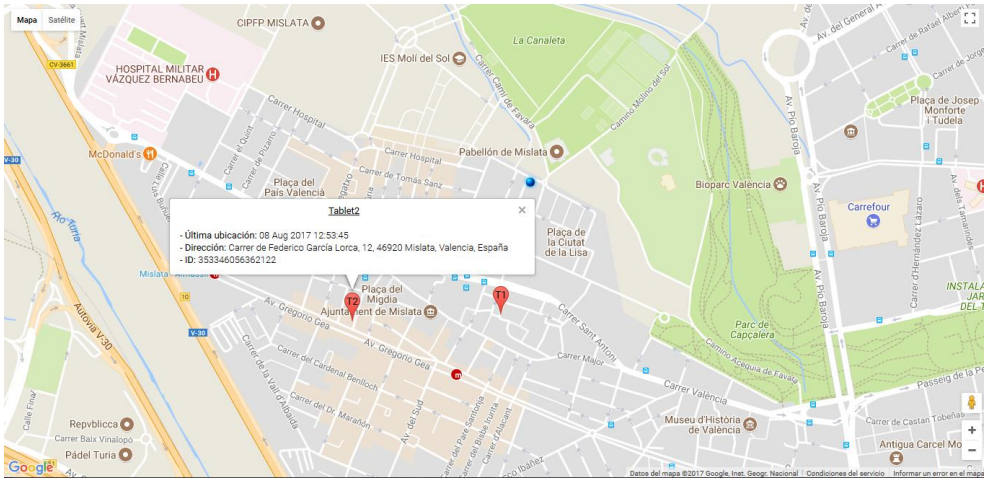

Ilustración 30: Mapa con dispositivos

7. Conclusiones

La idea de este proyecto era fundamentalmente dar un servicio remoto al cuerpo de la Policía Local de Mislata para poder acceder a su base de datos local y, a la vez, tratar de unificar diversos servicios a los que este cuerpo de policía tiene acceso. Finalmente, se puede considerar que se ha conseguido lo que se había estimado, por lo que, este proyecto ha sido un gran éxito.

En resumen, la aplicación para Tablet Android ideada en este proyecto, ha conseguido unificar una serie de servicios remotos, los cuales son: (i) acceso a la base de datos de Gespol de la Policía Local de Mislata, (ii) acceso a la base de datos del padrón de habitantes del Ayuntamiento de Mislata, (iii) enlazar a los servicios web de la DGT y la policía nacional y, por último, (iv) mantener un servicio web donde se puede ver sobre un mapa la localización en tiempo real de los dispositivos con los que irán equipados los vehículos policiales, que, además, mediante el control por voz de dicha aplicación, se facilita y se hace más cómoda la utilización de la misma. Asimismo, estos dispositivos constarán de un teclado inalámbrico para poder redactar de manera sencilla cuando sea necesario, como puede ser el momento de rellenar un informe de un suceso.

Durante los meses siguientes a la entrega de este proyecto se van a realizar pruebas reales para proceder a la implantación de dicha aplicación en la Policía Local de Mislata.

Una vez este proyecto esté totalmente instalado en la localidad de Mislata, se buscará ofrecer esta aplicación y adaptarla a las policías de otras localidades, ya que este proyecto puede ayudar a otros cuerpos de policía a dar un acceso remoto a sus bases de datos de manera que los agentes puedan consultar datos o elaborar informes *in situ* sin tener que llamar a la central de policía en cada ocasión.

8. Bibliografía

[1] Historia de las Tablets: https://es.wikipedia.org/wiki/Tableta_(computadora)

[2] Unixpol, software de gestón policial: https://polibuscador.upv.es/primo-explore/fulldisplay?docid=alma21388558400037 06&context=L&vid=bibupv&search_scope=ALL&tab=default_tab&lang=es_ES

[3] Aplicación para la gestión policial de la violencia de género del municipio de Valencia y pedanías: https://es.wikipedia.org/wiki/Tableta_(computadora)

[4] Definición de XAMPP: https://www.apachefriends.org/es/index.html

[5] Definición de IIS: https://es.wikipedia.org/wiki/Internet_Information_Services

[6] Instalación de IIS y PHP en WS 2003: https://msdn.microsoft.com/es-es/library/hh994592(v=ws.11).aspx

[7] Instalación de IIS y PHP: https://www.howtogeek.com/50455/how-to-install-php-on-iis-6-for-windows-server-2003/

[8] Solución de problemas WPI: https://community.rackspace.com/general/f/34/t/5739

[9] Instalación de IIS y PHP en WS 2012: https://msdn.microsoft.com/es-es/library/hh994592(v=ws.11).aspx

[10] Conexión PHP – SQL Server: http://php.net/manual/es/function.sqlsrv-connect.php

[11] Repositorio de extensiones de PHP: http://php.net/manual/es/install.pecl.intro.php

[12] Driver Oracle para PHP: https://pecl.php.net/package/oci8

[13] Configuración de BBDD Oracle en PHP: http://www.oracle.com/webfolder/technetwork/tutorials/obe/db/oow10/php_db/php_db.htm

[14] Configuración de BBDD Oracle en PHP: http://www.oracle.com/technetwork/articles/dsl/technote-php-instant-084410.html

[15] Conexión PHP – Oracle Database : http://php.net/manual/es/function.oci-pconnect.php

[16] Definición de IMEI: https://es.wikipedia.org/wiki/IMEI

[17] Google Maps y base de datos:
https://developers.google.com/maps/documentation/javascript/mysql-to-maps

[18] Marcadores en Google Maps:
https://developers.google.com/maps/documentation/javascript/markers

[19] Obtener clave API Google Maps:
https://developers.google.com/maps/documentation/javascript/get-api-key

[20] Visualización de datos en Google Maps:
https://developers.google.com/maps/documentation/javascript/earthquakes

[21] Geo-codificación:
https://developers.google.com/maps/documentation/javascript/geocoding

[22] Envío de datos en Android con Volley:
https://developer.android.com/training/volley/index.html

[23] Envío de datos por POST en Android con Volley:
https://www.simplifiedcoding.net/android-volley-post-request-tutorial/

[24] Obtención de los datos del GPS en Android:
https://developer.android.com/guide/topics/location/strategies.html