_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

APLICACIÓN MÓVIL DE EVENTOS CULTURALES GEOPOSICIONADOS EMPLEANDO FIWARE

Agenda cultural de Valencia

Rafael Vaño Garcia

Tutor: Carlos Enrique Palau Salvador

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación.

Curso 2015-16

Valencia, 5 de septiembre de 2016

A mis padres por su inestimable apoyo en todo momento.

A Carlos Palau por darme la oportunidad de realizar este proyecto.

A Eneko por su gran ayuda.

Resumen

Este proyecto trata sobre el geoposicionamiento de noticias, extraídas de feeds RSS, de la agenda cultural de la ciudad de Valencia. Estas noticias, se relacionarán con los sitios (teatros, monumentos, museos, …) ya geolocalizados de la web de datos abiertos del Ayuntamiento de Valencia, mediante el procesado del texto de las noticias. Además, en este proyecto también se desarrolla una aplicación para dispositivos *Android*, llamada *Agenda Cultural de Valencia*, que se nutrirá de una base de datos con los eventos geoposicionados anteriormente.

Las funciones principales de la aplicación son dos: por una parte, mostrar todas las noticias que aparecen en los *feed RSS* que se han conseguido geolocalizar y mostrar los detalles de cada una, y, por otra parte, mostrar en un mapa los eventos que tiene el usuario a su alrededor en un radio de búsqueda seleccionable, pudiendo el usuario seleccionar un evento para ver los detalles.

La aplicación ha sido desarrollada en *Android* nativo y es compatible con versiones de *Android* superiores a la 4.1. Además, requiere permisos por parte del usuario de conexión a Internet y ubicación del dispositivo. *Agenda Cultural de Valencia* está orientada a todo tipo de usuario.

Resum

Aquest projecte tracta sobre el geoposicionament de notícies, obtingudes de feeds RSS, de l'agenda cultural de la ciutat de València. Aquestes notícies, es relacionaran amb els llocs (teatres, monuments, museus, …) ja geolocalitzats de la web de dades obertes de l'Ajuntament de València, mitjançant el processament del text de les notícies. A més, en este projecte també es desenvolupa una aplicació per a dispositius *Android*, nomenada *Agenda Cultural de Valencia*, que s'alimentarà d'una base de dades amb els esdeveniments geoposicionats anteriorment.

Les funcions principals de l'aplicació son dos: per una part, mostrar totes les notícies que apareixen en els *feed RSS* que s'han aconseguit geolocalitzar i mostrar els detalls de cada una, i, per altra part, mostrar en el mapa els events que té l'usuari al seu voltant en un radi de recerca sel·leccionable, podent l'usuari seleccionar un esdeveniment per a veure els detalls.

L'aplicació ha sigut programada en Android natiu i és compatible amb versions d'*Android* superiors a la 4.1. A més, requereix permisos per part de l'usuari de connexió a Internet i ubicació del dispositiu. *Agenda Cultural de Valencia* està orientada a tot tipus d'usuari.

Abstract

This project is about news geolocation, from news that are extracted from *RSS feeds* about Valencia's cultural events. This news, will be related with places (like theatres, monuments, museums, …) from the Valencia Open Data website by processing the text of the news. Furthermore, in this project, is also being developed an application for *Android* devices called *Agenda Cultural de Valencia*, which will obtain the geopositioned events from a database.

Agenda Cultural de Valencia has two main functions: on the one hand, the app has to show all the news from the *feeds* that have already been correctly geopositioned and show the details of each one. On the other hand, the app has to show in a map the events that are around the user in a selectable search radius. Furthermore, the user could select an event in order to see more details about this event.

This application has been developed in native *Android* and supports Android versions above 4.1. Moreover, it requires user's permissions like internet connection and device location. *Agenda Cultural de Valencia* is aimed at all types of users.

Índice

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

1.1 Motivación

Los teléfonos móviles han evolucionado mucho desde que Martin Cooper desarrollara el primer teléfono celular para Motorola en 1973. Actualmente los móviles se utilizan para muchas más cosas que su cometido original, que consistía en recibir y realizar llamadas en terminales inalámbricos.

En 2016 nos encontramos de lleno en la era tecnológica, donde internet, los Smartphone y las aplicaciones para estos se han convertido en una parte imprescindible de nuestro modo de vida. Los llamados Smartphone (teléfonos inteligentes) han conseguido unos niveles de penetración casi impensables hace unos años. Como señala un estudio realizado por *We are social*, hay 3790 usuarios únicos de móviles, de los cuales 1968 billones son usuarios sociales activos [1].

Figura 1. Cifras globales de usuarios tecnológicos

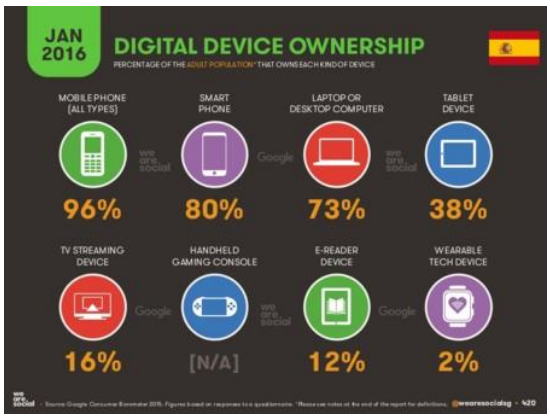


Figura 2. Datos de posesión de dispositivos tecnológicos en España

En España, un 80% de la población adulta posee un Smartphone, una cifra que parece aún más interesante si la comparamos con la de la posesión de ordenadores, cuya posesión alcanza un 73% de la población adulta. La conclusión es clara: en enero de 2016, los Smartphones ya han superado a los ordenadores en nuestro país.

Por otra parte, la popularización de internet y el avance de la tecnología, también ha permitido que avance y se expanda el concepto de *Smart City* (ciudad inteligente). Una *Smart City* es una ciudad que se ha adaptado tecnológicamente para responder a las demandas de sus ciudadanos, proporcionando datos y servicios en tiempo real (por ejemplo, proporcionar el estado del tráfico o de los sensores que miden la contaminación).

La ciudad de Valencia ha sido una de las ciudades pioneras en el ámbito de las ciudades inteligentes, implementando la plataforma *VLCi* (Valencia Ciudad Inteligente) con el Estándar abierto recomendado por la Comisión Europea para las *Smart City* llamado *FIWARE* [2]. Además, se ha habilitado una web de datos abiertos, disponibles para todos los usuarios (http://gobiernoabierto.valencia.es/es/data/) y una aplicación para smartphones en Android y IOS llamada *AppValencia*.

Este proyecto toma como objetivo la geolocalización de las noticias obtenidas de los diferentes feeds RSS de la agenda cultural de la ciudad, relacionando las noticias con los sitios ya

geoposicionados del portal de datos abiertos del ayuntamiento de Valencia. Seguidamente, estas noticias servirán para desarrollar una aplicación en Android que muestre las noticias ya geoposicionadas en el mapa y que también muestre las noticias que estén cerca del usuario en un radio determinado. La aplicación será compatible con las versiones más recientes de Android, concretamente con los dispositivos compatibles con la versión 4.2 hacia adelante.

La aplicación *AppValencia* tiene una sección en la que se pueden visualizar las noticias de la agenda cultural de la ciudad, pero solo las que aparecen en su feed RSS y sin mostrar su ubicación en el mapa. Por eso, en este proyecto se pretende ampliar esta función de la *AppValencia* seleccionando noticias de diferentes feeds RSS y mostrando al usuario los eventos en el mapa con una interfaz amigable y totalmente intuitiva, que pueda ser utilizada por cualquier usuario de un Smartphone.

1.2 Estructura del documento

Para guiar al lector de este Trabajo de Fin de Grado, se explicará a continuación la estructura del documento:

- **Introducción**

En el apartado introductorio, se hace una breve explicación sobre la motivación del proyecto, el problema que se plantea y la solución que se pretende implementar. También incluye esta explicación de la estructura del documento.

- **Gestión y organización del proyecto**

En este apartado, se describirá como se ha gestionado y organizado el proyecto, incluyendo la metodología que se ha empleado y las etapas por las que ha pasado su desarrollo.

- **Estado de la cuestión**

Se hace un estudio y explicación de la cuestión sobre la que se va a tratar, como la tecnología y herramientas utilizadas para la realización del proyecto. Podría entenderse como una introducción teórica.

- **Objetivos**

En este apartado, como su propio nombre indica, se describirán los objetivos que tiene este proyecto.

- **Desarrollo del sistema**

Se explicará el desarrollo del sistema completo, desde el programa para geoposicionar noticias hasta el desarrollo completo de la aplicación móvil.

- **Uso de la aplicación**

Se pretende explicar la funcionalidad completa de la aplicación desarrollada.

- **Conclusiones y líneas futuras**

En el apartado final, se expondrán las conclusiones finales derivadas de la realización del proyecto y las posibles líneas futuras para seguir mejorando el proyecto.

Capítulo 2. Gestión y organización del proyecto

Los proyectos de desarrollo de software necesitan seguir una planificación y una metodología específicas, no se puede pasar a la fase de implementación o desarrollo de manera directa. La guía que edita el mismo *INTECO* (Instituto Nacional de Tecnologías de la comunicación), más concretamente el *Laboratorio Nacional de Calidad del Software*¸ recomienda seguir una metodología para que el software desarrollado sea de mayor calidad [3].

Para esto, es necesario conocer qué es el software, y una vez afianzado este concepto, conviene adentrarse en el concepto de *Ingeniería del software*, ya que aplicar un enfoque de ingeniería permite solucionar un problema de manera más eficiente. Además, el *INTECO* recomienda utilizar la metodología del *desarrollo ágil*, por lo que se explicará este concepto y el método que se seguirá para el desarrollo del software.

Este capítulo empezará con un preludio con una explicación sobre el nacimiento y las condiciones de este proyecto, para seguidamente adentrarse en los conceptos que ya se han introducido, así como la planificación para llevar a cabo este proyecto.

2.1 Preludio

Este Proyecto tiene surge a principios de 2016 por la aparición de una beca de colaboración con el Grupo de Sistemas y Aplicaciones de Tiempo Real Distribuido de la Universitat Politècnica de València (SATRD-UPV), a través de la Cátedra Telefónica y con la colaboración del Ayuntamiento de Valencia. Por lo tanto, la aplicación que se desarrollará en este proyecto tiene como finalidad un funcionamiento real de la misma, para mejorar la experiencia de los asistentes a eventos culturales en la ciudad de Valencia y la mejor promoción de estos eventos de la ciudad.

Por estas razones, se ha intentado adecuar el software y la aplicación desarrollada a las exigencias de los supervisores de la beca de colaboración, y sobre todo teniendo en cuenta que este proyecto no es solo de ámbito académico, sino un proyecto con una aplicación real.

Para la realización de la aplicación, se ha decido que el sistema se compondrá de 4 partes diferenciadas: una base de datos, un programa que geoposicione las noticias y alimente a la base de datos, un servicio web que actúe como puente de datos entre la aplicación y la base de datos; y finalmente, la propia aplicación. Esta arquitectura se detallará en el capítulo 5 de esta memoria.

2.2 Software e Ingeniería del software

2.2.1 Software

El *IEEE* define en su estándar 610 el software como "programas, procedimientos y documentación y datos asociados, relacionados con la operación de un sistema informático" [3]. Además, se puede definir el software como el conjunto de tres componentes:

- **Programas:** proporciona la funcionalidad y rendimiento que debe de proporcionar el programa cuando se procesa a su ejecución. Están escritos en lenguajes entendibles para los ordenadores, para que puedan leerlos y ejecutarlos.
- **Datos:** son los datos imprescindibles para poder ejecutar los programas y las estructuras requeridas para la manipulación de estos datos.
- **Documentos:** describe como se usa y trabaja el programa, es como un manual de instrucciones sobre el programa para los usuarios que van a manejarlo.

Cada producto de software que se desarrolla es diferente, porque el software está pensado para satisfacer los requisitos únicos de cada cliente. Por esto, el software se crea normalmente desde cero y hace necesario entender y analizar las necesidades del producto, vigilar para que cumpla los requisitos del cliente, implementar este diseño con un lenguaje de programación y, finalmente, comprobar que el programa final cumpla con los requisitos anteriormente especificados.

2.2.2 Ingeniería del software

El *Diccionario de la Real Academia Española* define a la ingeniería del software como el "conjunto de conocimientos y técnicas que permiten aplicar el saber científico a la utilización de la materia y de las fuentes de energía". Según el *IEEE*, la ingeniería del software "es como un enfoque sistemático cubriendo los aspectos del desarrollo, operación y mantenimiento. Este enfoque es disciplinado y cuantificable" [3].

El objetivo principal de la ingeniería del software es desarrollar un producto de una alta calidad de una manera oportuna, utilizando las técnicas propias de la ingeniería para la resolución eficiente de los problemas. La ventaja de la ingeniería del software es que es una metodología adaptativa, lo que permite adaptarla a los diferentes elementos del desarrollo del software con una alta flexibilidad.

Algunos de los principios más importantes de la ingeniería del software son:

- Realizar un diseño.
- Documentar el producto.
- Trabajar con la calidad del producto como objetivo.
- Realizar muchas pruebas.
- Compartir la visión del producto final con el cliente.
- Asunción de responsabilidades.

2.3 Metodología de diseño

La tarea de desarrollo del software es una tarea a menudo muy complicada, por lo que existen muchas metodologías a seguir para el desarrollo del software.

Una metodología consiste en "un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo. Es un proceso de software detallado y completo" [3].

Aplicando una metodología a la ingeniería del software, se optimiza todo el proceso y el software resultante de este proceso. Esta metodología está formada por unos métodos que definen qué y cómo hacer durante el proceso de desarrollo del software.

En el siguiente subapartado se detallarán las grandes ventajas que implica seguir una metodología de diseño.

2.3.1 Ventajas de uso de una metodología

Existen numerosas ventajas para seguir una metodología en el proceso de desarrollo del software en comparación a no seguir ninguna metodología concreta. Por esto conviene detallar algunas de estas ventajas desde diferentes puntos de vista.

Centrando el punto de vista desde la gestión del proyecto, seguir una metodología facilita la tarea de planificación, control y seguimiento del proyecto, mejora la relación beneficio/coste y optimiza el uso de los recursos de los que se dispone. Además, hace más fácil el análisis de los resultados y cumplimiento de los objetivos, y facilita una cosa tan imprescindible como la comunicación entre los usuarios y los desarrolladores.

Desde el punto de vista de los desarrolladores, algunas ventajas pueden ser la optimización de cada una de las fases del proceso de desarrollo y su conjunto, la ayuda a la correcta comprensión del problema a resolver, así como permitir la reutilización de partes del software.

Por último, entre las ventajas que puede aportar el uso de una metodología desde el punto de vista del cliente o usuario, destacan la obtención de una mayor seguridad en los plazos de tiempo fijados al comienzo del desarrollo, así como la garantía de una mayor calidad en el producto final.

2.3.2 Elección de una metodología

Actualmente, si clasificamos las metodologías según su filosofía de desarrollo, podemos diferenciar dos grupos: las metodologías clásicas y las metodologías ágiles.

Las **metodologías clásicas** o tradicionales se basan en una planificación potente durante todo el proceso de desarrollo, por lo que a menudo son denominadas como "pesadas". Dan mucha importancia a la fase inicial del proyecto, donde se define una planificación que se debe de cumplir a lo largo del proyecto, además de que pretenden que se lleve una documentación exhaustiva de todo el proyecto. Por lo tanto, son metodologías poco flexibles, ya que realizar cambios respecto a la planificación inicial en el proyecto implica costes muy altos.

Por otra parte, las **metodologías ágiles**, al contrario de las metodologías clásicas, están basadas en la adaptabilidad del proceso de desarrollo, dan más importancia a la flexibilidad y adaptabilidad que al seguimiento estricto de un plan desde el principio e intentan retrasar las decisiones. Defienden que esta metodología es más realista que definir todos los requisitos en el comienzo del proyecto.

Para la realización de este proyecto, se ha optado por seguir una metodología ágil, ya que se ha preferido un proceso de desarrollo flexible, para poder ir desarrollando el producto sin tener

desde el principio un plan encorsetado y claramente definido. Además, al no poder predecir los resultados de cada uno de los procesos, el software necesitará una cierta flexibilidad y adaptabilidad, dependiendo de los resultados que se van obteniendo de cada proceso. Se pretende partir de una planificación mínima, para ir alcanzando los objetivos con pequeños incrementos de la funcionalidad del software.

Dentro de los métodos de desarrollo ágil, se ha optado por seguir la metodología conocida como ***Programación Extrema (XP)***, la más destacada de las metodologías ágiles. Esta metodología fue formulada por Kent Beck en 1999, y defiende que los cambios de requisitos durante el proceso de desarrollo son un aspecto natural, inevitable e incluso deseable. La *Programación Extrema* intenta adaptar lo mejor de las metodologías de desarrollo a las características específicas del proyecto de manera dinámica durante la fase de desarrollo.

Otra característica que hace interesante a la *XP* para este trabajo de fin de grado, es que *XP* tiene como objetivo potenciar las relaciones interpersonales para llegar a conseguir el éxito en el desarrollo del software, ya que se interesa por el aprendizaje de los programadores e intenta propiciar un buen clima de trabajo. Además, la *Programación Extrema* se basa en el *feedback* entre el cliente (en este caso el tutor) y el equipo de desarrollo (en este caso el alumno) y en la apertura a los posibles cambios en el proyecto.

Los principios básicos de la *Programación Extrema* son los siguientes:

- **Simplicidad:** es una de las características de la *Programación Extrema*¸ ya que se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento. Para mantener la simplicidad del código a medida que va creciendo el proyecto, es necesario su refactorización continua.
- **Comunicación:** un código más simple es más fácil de leer y entender para los desarrolladores, por lo que se prefiere un código autodocumentado a los comentarios en el código, ya que este puede ir variando. La comunicación con el cliente también es muy importante, ya que éste decide las características con mayor prioridad y se pretende que forme parte del equipo de desarrollo.
- **Retroalimentación o *feedback*:** la realización de ciclos cortos y la obtención de resultados inmediatos como consecuencia, ayuda a los desarrolladores a centrarse en lo más importante. Las pruebas unitarias permiten saber el estado del proyecto en tiempo real.
- **Coraje:** permite a los programadores construir el código a su ritmo, con cierta comodidad, dándoles libertad para que examinen y evalúen el código. [3] [4]

Capítulo 3. Estado de la cuestión

En este capítulo, se hará una breve introducción teórica sobre la tecnología en la que está basada este proyecto, que servirá para asentar unos conocimientos básicos sobre estas tecnologías. También se detallarán las herramientas utilizadas para la realización del trabajo.

El programa que lee las noticias de los feeds RSS y las geoposiciona relacionándolas con los sitios donde ocurren está programado íntegramente en el lenguaje *Java*, por lo que se empezará explicando el lenguaje de programación *Java*, su historia, así como sus características y sus posibles usos. Además, estas noticias, sitios, feeds y *matches* de noticias y sitios se guardarán en una base de datos *MySQL*, por lo que se hará una breve exposición de las bases de datos, concretamente de *MySQL*, y el lenguaje *SQL*, incluyendo su uso en *Java*. También repasaremos brevemente el concepto de *RSS* y su formato, *XML;* además de explicar el formato *JSON*, importante porque es el formato en el que está la lista de sitios y el servidor web devolverá los datos en este formato.

También se explicará el concepto de *NLP* en ciencias de la computación y el *chunking*, ya que se usarán estas técnicas para obtener los *matches* de las noticias con los sitios.

Para finalizar, conviene introducir el sistema operativo *Android*, ya que la aplicación se desarrollará para este sistema operativo. Por lo tanto, se detallará su historia, características, arquitectura y versiones más utilizadas.

3.1 Java

Como se define en la web oficial de *Java*, "*Java* es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems" [5]. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como *WORA*, o "*write once, run anywhere*"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para poderse ejecutar en otra.

El lenguaje de programación *Java* fue creado por James Gosling en 1995 para la compañía Sun Microsystems como un componente fundamental de la *plataforma Java*. Su sintaxis deriva en gran medida de los lenguajes de programación *C* y *C++*, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Además, no soporta extensiones de código ensamblador.

Figura 3. Icono de *Java*

Las características principales de *Java* son las siguientes:

- **Orientado a objetos:** la programación orientada a objetos es un paradigma de programación que utiliza entidades llamadas objetos como elementos fundamentales a la hora de construir una solución. El objetivo de este tipo de programación es la reutilización

de estos objetos genéricos en diferentes proyectos, una reutilización del código que es precisamente una premisa de la *Ingeniería del Software*, de este modo se pretende reducir el número de proyectos fallidos, así como obtener un menor tiempo de desarrollo. En *Java* están presentes todos los conceptos que componen esta técnica de programación como: herencia, polimorfismo, abstracción, …

- **Seguro:** *Java* fue diseñado en una estructura *Button Up* teniendo en cuenta la seguridad de la plataforma. Además, se implementaron medidas de seguridad en el lenguaje y en el sistema de ejecución en tiempo real.

- **Independencia de la plataforma:** el software escrito en *Java* está preparado para ejecutarse en cualquier tipo de hardware, es decir, el software se programa solamente una vez para que pueda ejecutarse en varios dispositivos, lo que se conoce como *WORA*, como ya hemos mencionado antes. El código *Java* se compila y se genera un código llamado *bytecode*, que posteriormente será ejecutado en la *Java Virtuale Machine* (*JVM*), que interpreta y ejecuta el código.

- **Multihilo:** el lenguaje de programación *Java* soporta la sincronización de múltiples hilos de ejecución, concepto llamado *multithreading*, por lo que tiene la capacidad de cumplir diferentes funciones al mismo tiempo.

- **Alto rendimiento:** al ser rápido ejecutando los programas y la capacidad de ahorrarse bastantes líneas de código, se puede considerar a *Java* de alto rendimiento.

- **Gestión automática de la memoria:** *Java* utiliza un recolector automático de basura para la gestión de la memoria en el ciclo de vida de los objetos. El programador decide cuando se crean y destruyen los objetos, sin preocuparse de la gestión de la memoria. El entorno en tiempo de ejecución de *Java*, llamado *Java runtime*¸ se encarga del ciclo de vida de los objetos con la ayuda del recolector de basura, que libera la memoria que ocupaban estos objetos. De esta forma se evitan posibles fugas de memoria y se libera al programador de tener que gestionar directamente la memoria.

La *plataforma Java* es un conjunto de programas que facilitan el desarrollo y la ejecución de programas escrito en el lenguaje de programación *Java*. Una plataforma *Java* está formada por un motor de ejecución (llamada máquina virtual), un compilador y un conjunto de librerías. Algunos ejemplos de plataformas son: *Java EE* (Enterprise Edition), *Java ME* (Micro Edition) y *Java SE* (Standard Edition).

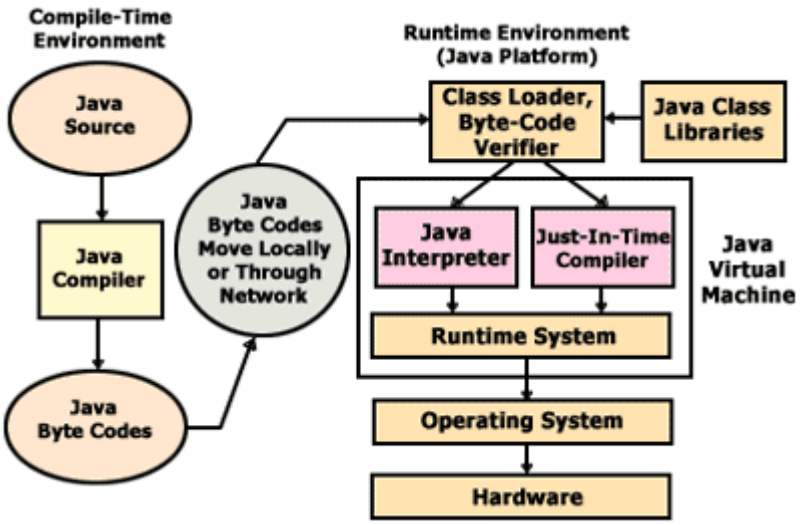


Figura 4. Estructura de ejecución de un programa escrito en *Java*

La figura 4 muestra en esquema de la ejecución de un programa escrito en *Java*, desde que de compila el código escrito en *Java* hasta que se ejecuta en el hardware del equipo. Algunos componentes importantes son:

- **Java Runtime Environment (JRE):** El *JRE* (en castellano Entorno en Tiempo de Ejecución de Java) es el software necesario para ejecutar cualquier aplicación desarrollada para la plataforma Java.

- **Java Virtual Machine (JVM):** es el centro de la *plataforma Java* y es donde se ejecutan los códigos bytecode *Java* generados anteriormente por el compilador. El uso de estos bytecode es lo que permite ejecutar los programas *Java* en cualquier *JVM* con independencia del dispositivo y sistema operativo que use.

- **Librerías de clases:** estas librerías tienen como objetivo facilitar la tarea del programador, aportando funciones comúnmente utilizadas. También aportan una interfaz abstracta para las tareas que normalmente dependen del hardware o sistema operativo utilizado. Algunos ejemplos de librerías son: *java.io* o *java.net*. [6]

3.2 MySQL y bases de datos

3.2.1 Bases de datos relacionales

Las bases de datos pueden definirse como un lugar donde se guarda un conjunto de información determinada, relacionada entre sí y que se busca ordenar y clasificar según unos criterios. Centrándonos en las bases de datos informáticas o las que se pueden utilizar en el contexto de este trabajo, estas bases de datos deben de ser dinámicas y de fácil acceso a la información que contienen.

Actualmente, hay diferentes modelos de bases de datos, como: *bases de datos jerárquicas* (un nodo principal de información con varios nodos secundarios), *bases de datos de red* (como las jerárquicas, pero un nodo secundario puede tener varios nodos principales), *bases de datos orientadas a objetos* (propio de los programas informáticos basados en objetos como *Java*, incorpora los aspectos principales del paradigma de objetos como herencia, polimorfismo, …) y las *bases de datos relacionales*, que se explicarán a continuación.

Fue el *Dr. Edgar F. Codd* en la compañía *IBM* en los años 70 quien creó el modelo de datos relacional en el que se basan estas bases de datos relacionales.

Las bases de datos relacionales cumplen con el modelo relacional, que es el modelo de bases de datos mejor para bases de datos con una estructura ya planificada con antelación. Estas bases de datos están formadas por un conjunto de tablas que son las que contienen la información, formadas por filas (campos) y columnas (registros), obteniéndose un dato cuando se cruzan una fila y una columna. [7]

Una vez introducidas las bases de datos relacionales, conviene detallar sus características:

- No puede haber diferentes tablas con el mismo nombre ni registro
- Cada tabla debe de tener al menos un registro o clave primaria y deben de cumplir la integridad de datos, es decir, la correcta complementación de datos de la base de datos para que no haya registros erróneos.
- Las relaciones entre tablas se establecen mediante el registro primario de una tabla y el registro de otra tabla, pudiendo ser este último registro primario o no. Estos registros contendrán el mismo valor en ambas tablas.
- Las relaciones entre tablas pueden ser *1 a 1*, *1 a n* y *n a n*.

3.2.2 MySQL

Actualmente, existen múltiples gestores de bases de datos para entornos de desarrollo web, como *MySQL*, *Oracle* o *Microsoft SQL Server*. Nos centraremos en *MySQL* que es el gestor de bases de datos que se utilizará en este proyecto.

MySQL es un sistema de gestión de bases de datos relacionales, multihilo y multiusuario, desarrollado por la compañía *Oracle Corporation.* Además, es de código abierto y según la misma *Oracle* es "la base de datos de código abierto más popular del mundo" [8]. *MySQL* tiene una alta disponibilidad, ya que se han creado bastantes marcos de trabajo o *frameworks* por parte de los usuarios de este sistema de bases de datos.

Por otra parte, *MySQL* también se presenta como la opción perfecta para el correcto desarrollo del concepto denominado como *IoT* (Internet of Things), el llamado internet de las cosas; ya que permite almacenar y tratar con eficiencia los datos que se manejan en estas aplicaciones. [9]

Figura 5. Logo de *MySQL*

3.2.3 El lenguaje SQL

El lenguaje *SQL* (Structured Query Language) es un lenguaje utilizado para la definición, manipulación y control de bases de datos relacionales [10]. Se ha convertido en el lenguaje estándar y por excelencia de las bases de datos relacionales, tal es así que los principales gestores de bases de datos utilizan el lenguaje *SQL*, como *MySQL*¸ que incluso ha adoptado su nombre.

SQL fue creado en los años 70 por un laboratorio de investigación de la empresa IBM, ya que estaban investigando las bases de datos relacionales. Este nuevo lenguaje se extendió y se fue popularizando, hasta que *SQL* fue nombrado estándar oficial del *ANSI* en 1986 y del *ISO* en 1987.

El lenguaje *SQL* es un lenguaje de programación **declarativo**, es decir, el programador especifica en la sintaxis un conjunto de preposiciones, afirmaciones, restricciones, … que directamente describen el problema y también la solución a este problema.

Otro aspecto importante de una base de datos, son los tipos de datos que ésta puede almacenar, por lo tanto, conviene describir los tipos de datos básicos de *SQL*, que son:

- **Cadenas de caracteres:** cadenas de caracteres que pueden contener letras, números o caracteres especiales, para almacenar datos del tipo texto. Algunos tipos son*: Char, Varchar*, …
- **Datos temporales:** información sobre fechas, horas o una combinación de estos. Son los tipos*: Date, Time, DateTime*.
- **Datos enteros:** permite almacenar números enteros. Dependiendo del tamaño hay distintos tipos: *Int, Bigint, TinyInt*, …
- **Datos flotantes:** para representar números de punto flotante, decimales. Son los tipos *Float, Double*, …

3.2.4 Tipos de sentencias SQL

El lenguaje *SQL* se utiliza mediante el uso de sentencias que describen las acciones que deben hacerse sobre la base de datos. Por lo tanto, conviene explicar los diferentes tipos de sentencias *SQL* que existen y las acciones que representan. Hay tres tipos de sentencias: [10]

- **Lenguaje de definición de datos (DDL)**

El lenguaje de definición de datos se encarga de hacer modificaciones en los objetos de la base de datos, como pueden ser su creación, eliminación o modificación. Estos objetos a los que se refiere pueden ser tablas, vistas, índices, …

Las instrucciones *SQL* pertenecientes a este grupo son: *CREATE, ALTER, DROP, TRUNCATE.*

- **Lenguaje de manipulación de datos (DML)**

Las instrucciones *SQL* incluidas en este grupo son las referidas a la consulta, gestión o manipulación de los datos almacenados en la base de datos. Estas operaciones pueden ser la inserción, eliminación o modificación de los datos.

Las instrucciones *SQL* pertenecientes a este grupo son: *INSERT*, *DELETE*, *UPDATE*, y las instrucciones de filtrado como *WHERE* y *HAVING*.

- **Lenguaje de control de datos (DCL)**

En este grupo se incluyen las instrucciones *SQL* relacionadas con la gestión del control del acceso de los usuarios y programas a objetos de la base de datos. Estas operaciones permiten configurar el tipo de acceso de estos usuarios y programas a la base de datos.

Las instrucciones *SQL* pertenecientes a este grupo son: *GRANT* o *REVOKE*.

```
SELECT title, timestamp, keywords, body FROM noticias WHERE feed=1
```

title	timestamp	keywords	body
Pinazo: Del ocaso de los grandes maestro...	1472556474000	,valencia, ayuntamiento, exposiciones, MU...	Pinazo: Del ocaso de los grandes maestro...
Notes Soltes	1472480002000	,valencia, ayuntamiento, música, CENTRO ...	Notes Soltes. FECHA: Viernes 9 de septiem...
Antropometrías. Del Yo al Nosotros.	1472473230000	,valencia, ayuntamiento, exposiciones, CE...	INAUGURACIÓN: 1 de septiembre de 2016...
Kapital 02 / no-sexism. Exposición colectiva.	1472472129000	,valencia, ayuntamiento, exposiciones, LAS...	FECHA: del 15 de septiembre al 1 de octu...
Jovens Solistes de Lliria	1472459910000	,valencia, ayuntamiento, música, CENTRO ...	Jovens Solistes de Lliria. FECHA: Sábado 1...
Historias de ratones	1472194684000	,valencia, ayuntamiento, agenda infantil, C...	Historias de ratones. FECHA: Domingo 4 d...

Figura 6. Ejemplo de una consulta *SQL* y el resultado obtenido

3.3 XML

XML (eXtensive Markup Language) no es un lenguaje de marcado como podría sugerir su nombre, sino que es un meta-lenguaje para la definición de lenguajes de marcado, desarrollado por el *World Wide Web Consortium (W3C)* ¸ y que permite definir etiquetas personalizadas para descripción y organización de datos. *XML* define la sintaxis y los requisitos que deben cumplir los lenguajes demarcado que especifica

En 1970, *IBM* ideó un estándar llamado *Generalized Markup Language (GML)* por la necesidad de la empresa de almacenar una gran cantidad de información y compartirla con otras plataformas. La ISO se interesó por este estándar y se puso como objetivo su normalización, por lo que en 1986 creó el *Standard Generalized Markup Languag (SGML)* ¸ del que deriva *XML*.

Entre las características más importantes de *XML*, podemos destacar:

- **Extensible**: se pueden definir marcas propias y personalizadas.
- **General:** es válido para cualquier clase de objetos.
- **Descriptivo:** describe los datos, pero no su uso.
- **Marcado:** señales añadidas a la información para su procesado.

Los documentos *XML* han de seguir una estructura concreta para ser sintácticamente correctos y que puedan ser analizados por un analizador sintáctico sin errores. Las partes de un documento *XML* son:

- **Prólogo:** no es obligatorio, pero un documento *XML* puede empezar con una descripción de sus características, como el número de versión o el tipo de documento.
- **Cuerpo:** el cuerpo es obligatorio y solo debe de tener un elemento raíz. En el cuerpo se encuentra la información del documento.
- **Elementos:** pueden tener contenidos o estar vacíos.
- **Atributos:** son las características que pueden tener los elementos. [11]

3.4 RSS

RSS (Really Simple Syndication) es un formato *XML* que se utiliza para compartir información de manera muy simple, y continuamente actualizada, por internet, a usuarios que previamente se han suscrito a un canal de noticias o *feed RSS* [12]. Actualmente los *feeds RSS* se utilizan para compartir noticias de periódicos (cada periódico digital suele tener su propio canal de noticias *RSS* separado por contenidos), compartir actualizaciones de blogs o información interesante de páginas web.

Cuando un usuario se suscribe a un *feed RSS* no es necesario que entre a la página web para ver si hay actualizaciones, el programa que utiliza para la lectura y la gestión de los *feeds*, llamado *agregador* o *lector de fuentes web*, se encarga de la gestión de la actualización de las noticias de los *feeds* a los que está suscrito el usuario. Estos *agregadores* pueden ser aplicaciones de escritorio o una aplicación web, a los que el usuario facilita la *url* del *feed* a suscribirse, pasando a gestionarlo todo el *agregador*.

Un documento *RSS* contiene texto (noticias o información) y metadatos, como el autor, la fecha de publicación, la categoría… Al ser documentos *XML*, esta información aparece como atributos de los elementos, que son las noticias de los *feeds*.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <rss version="2.0">
  - <channel>
      <title>Agenda Ajuntament de València</title>
      <link>http://www.valencia.es</link>
      <description>Agenda del Ajuntament de València</description>
      <language>es-es</language>
      <pubDate>Tue, 1 Sep 2016 16:39:30 +0100</pubDate>
      <lastBuildDate>Tue, 1 Sep 2016 16:39:30 +0100</lastBuildDate>
      <managingEditor>webmaster@valencia.es</managingEditor>
      <webMaster>webmaster@valencia.es</webMaster>
      <copyright>Ajuntament València @2005</copyright>
      <ttl>30</ttl>
    - <image>
        <title>Agenda :::::::::::: Ajuntament de València </title>
        <link> http://www.valencia.es/ayuntamiento2/ </link>
        <url> http://www.valencia.es/ayuntamiento2/ndportada.nsf/logo1.gif </url>
      </image>
    - <item>
      - <title>
        - <![CDATA[
            Noche Americana de Rock
          ]]>
        </title>
        <link>http://www.valencia.es/ayuntamiento/Agenda_accesible.nsf/Agenda/96C4FFA1DB8008C5C1258021002ED78D?
          openDocument&lang=1&bdOrigen=ayuntamiento/laciudad.nsf</link>
```

Figura 7. Estructura de un *feed RSS* en *XML*

En la figura 7 se puede observar la estructura *XML* de un *feed RSS*, mientras que en la figura 8 se puede observar el mismo *feed* utilizando un *agregador* para su visualización. Como se puede ver, el *agregador* da forma al canal de noticias y lo presenta con una interfaz más visual y amigable al usuario.

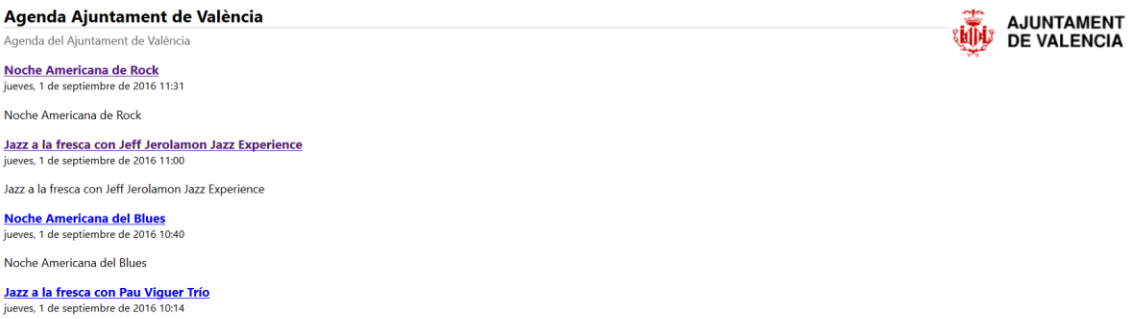


Figura 8. *Feed RSS* visto desde un *agregador*

3.5 JSON

JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos. Su escritura y su escritura es realmente fácil para los humanos, y su generación y parseo (lectura, análisis de la información) no cuesta trabajo a las computadoras [13]. Esta es una de sus ventajas, al ser fácil desarrollar un analizador de elementos *JSON*.

El formato de texto *JSON* es totalmente independiente del lenguaje, pero usa convenciones que les son familiares a los programadores de la familia de lenguajes como: *C*, *C++*, *Java*, *JavaScript*… Estas propiedades hacen de *JSON* un lenguaje excelente para el óptimo intercambio de datos [13].

JSON está construido por dos estructuras:

- Una colección de pares de nombres/valores, comúnmente denominados *objetos*.
- Una lista ordenada de valores.

```
{
"type": "FeatureCollection",
"crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:EPSG
  ::25830" } },

"features": [
{ "type": "Feature", "properties": { "nombre": "JIMMY GLASS JAZZ BAR",
  "identifica": "009200", "idclase": "57"}, "geometry": { "type":
  "Point", "coordinates": [ 725407.5, 4373053.9 ] } },
{ "type": "Feature", "properties": { "nombre": "SALA WAH WAH",
  "identifica": "009202", "idclase": "57"}, "geometry": { "type":
  "Point", "coordinates": [ 728215.5, 4372393.7 ] }}]}
```

Figura 9. Ejemplo de formato *JSON*

3.6 NLP

El *procesamiento de lenguajes naturales (NLP)* es un campo de estudio que tienen en común la inteligencia artificial, las ciencias de la computación y la lingüística que estudia la interacción entre las computadoras y el lenguaje humano. La investigación en este campo se centra en el desarrollo de algoritmos eficientes para el procesamiento de textos y hacer accesible la información de estos textos a las computadoras. Esto quiere decir, que el objetivo es desarrollar software que analice, "entienda" y consiga general lenguaje como el que los humanos usan para comunicarse [15] [16].

El *NLP* presenta un gran reto con muchas dificultades, ya que el lenguaje humano es muy difícil de poder interpretar para una máquina, ya que el único capaz de entenderlo y analizarlo con plenitud es el cerebro humano. Además, el lenguaje humano suele ser muy ambiguo y difícil de interpretar sin el contexto que rodea a la comunicación.

Algunas de las tareas más importantes propias del *NLP* son: [15]

- **Segmentación morfológica:** separación de las palabras en morfemas.
- **NER (Named Entity Recognition):** es una tarea importante, ya que determina las palabras de un texto que son nombres propios, para seguidamente intentar determinar el tipo al que pertenece este nombre propio (persona, lugar, empresa, …)
- **Comprensión del lenguaje natural:** convierte pedazos de texto a representaciones más fácilmente manipulables por los ordenadores, como la *lógica de primer orden*.
- **Etiquetado gramatical:** determina la categoría gramatical de cada palabra de una frase. Este es otro aspecto importante, ya que en muchos lenguajes como el inglés, hay palabras que dependiendo de la frase pueden ser un verbo o un sustantivo, como la palabra ring.
- **Desambiguación del significado de las palabras:** dependiendo del contexto de la frase, se debe escoger el significado correcto de una palabra con varios significados.

3.7 Chunking

En el contexto del *NLP*, la técnica del *chunking* (también llamada *shallow parsing*, o análisis superficial) consiste en la identificación de partes constituyentes de una frase (como nombres, verbos o adjetivos) [16]. En esta técnica no se analiza la estructura de la frase ni la funcionalidad dentro de la frase de la parte constituyente que se está analizando.

El *Chunking* se utiliza principalmente para buscar una o unas palabras determinadas dentro de una frase, sin importar el contexto ni su significado.

3.8 Android

Desde la popularización de los Smartphones, han surgido bastantes sistemas operativos específicos para estos dispositivos, como *Symbian* de *Nokia*, *Firefox OS* o *Blackberry OS*, pero los sistemas operativos que se han consolidado son *Android*, *IOS* y en menor medida *Windows Phone*. Android se ha convertido en el sistema operativo más usado en los Smartphones por excelencia, alcanzando en 2015 una cuota de mercado del 82.8%, mientras que IOS solo alcanzó un 13.9% de la cuota de mercado, según la *International Data Corporation* [17]. Este hecho puede deberse principalmente a que *Android* es un sistema operativo de código abierto, y no está limitado a unos dispositivos concretos, a diferencia de *IOS* que solo se comercializa en dispositivos exclusivos por parte de la marca *Apple*, que es también propietaria de *IOS*.

La aplicación que se tiene como objetivo en este proyecto se desarrollará exclusivamente para *Android*¸ por lo tanto, se explicará la historia, arquitectura, características y versiones de este sistema operativo.

Period	Android	iOS	Windows Phone	BlackBerry OS	Others
2015Q2	82.8%	13.9%	2.6%	0.3%	0.4%
2014Q2	84.8%	11.6%	2.5%	0.5%	0.7%
2013Q2	79.8%	12.9%	3.4%	2.8%	1.2%
2012Q2	69.3%	16.6%	3.1%	4.9%	6.1%

Figura 10. Cuotas de mercado de los sistemas operativos para Smartphones

3.8.1 Historia

Android empezó sus andaduras en 2003, cuando unos ingenieros llamados Rich Miner, Chris White y Nick Sears empiezan a desarrollar un sistema operativo para teléfonos móviles basada en Linux, y para ello crean una empresa llamada *Android Inc* que estaría ubicada en Plato Alto, California. El empujón definitivo para esta empresa y este sistema operativo que se estaba gestando, llega cuando la potente empresa *Google* compra *Android Inc.*, ya que estaba interesada en el cada vez más incipiente mercado de la telefonía móvil.

A partir de este momento, *Google* intentó sumar a los diferentes fabricantes de hardware, desarrolladores de software y operadores de telefonía a su proyecto de *Android*. Estos acercamientos cristalizaron finalmente en noviembre de 2007, cuando se fundó la *Open Handset Alliance*, que es precisamente una alianza de fabricantes y desarrolladores de hardware y software, además de operadores de servicios de telefonía, siendo sus miembros más destacados *Google*, *HTC*, *Dell*, *T-Mobile*, *NVIDIA*, *Intel*, *Qualcomm* o *LG*.

Junto a la creación de esta alianza, llegó la primera versión del sistema operativo *Android*, llamada *Android 1.0 Apple Pie*. Pero no fue hasta un año más tarde, en 2008, cuando se lanzó al mercado el primer Smartphone *Android*¸ el *HTC Dream*, con un precio de 179 dólares y que ya contaba con la versión de *Android 1.5*. Además, en la cubierta del dispositivo se podían ver la marca *Google*, lo que significaba la confirmación de la *Open Handset Alliance* y el renacer de la empresa *HTC*. También se estrenó en 2008 la tienda de aplicaciones para *Android*, llamada en aquel momento *Android Market*, que luego derivaría en la actual *Google Play*. [18]

Figura 11. Logo de *Android*

Desde 2008, *Android* ha ido expandiendo su cuota de mercado hasta alcanzar un 82.8% en 2015, como ya hemos señalado antes. Además, esta gran popularización se ha correspondido con el desarrollo y actualización de más versiones de *Android*, que se describirán más adelante.

3.8.2 Características

Es conveniente explicar las características que hacen de *Android* un sistema operativo tan importante. La característica más importante de *Android*, y la que seguramente le hacen tener una ventaja competitiva respecto a sus principales competidores, es que *Android* es un sistema operativo de **código abierto**, aunque pertenezca a *Google*. Al ser de código abierto, cualquier usuario puede acceder a su código fuente y modificar este sistema operativo, lo que convierte a *Android* en altamente personalizable por sus usuarios, otra gran ventaja.

Al ser de código abierto y distribuible a una gran variedad de dispositivos, *Android* tiene capacidad para adaptarse al dispositivo en el que esté instalado y a todo tipo de pantallas y resoluciones. Para que las aplicaciones se visualicen correctamente en todo tipo de dispositivos, se recomienda al desarrollador que adapte su aplicación a distintos tamaños de pantallas y resoluciones, ya que el kit de desarrollo de *Android* da muchas facilidades en este aspecto.

Android ofrece otras muchas características importantes. Ofrece soporte de *Java*, aunque no haya una máquina virtual en la plataforma, ya que el bytecode de *Java* se ejecuta en la *Máquina Virtual Dalvik*. Además, soporta una gran cantidad de formatos multimedia como *H264*, *JPEG*, *MP3*, *GIF*, … y soporte para streaming como el streaming *RTP/RTSP*. Otro dato importante, es que ofrece soporte nativo a pantallas capacitivas multi-táctil. [18] [21]

3.8.3 Arquitectura

Como muestra la figura 12, la arquitectura de *Android* está compuesta por cuatro capas, cuya característica principal es que están basadas en software libre.

El núcleo de *Android* está formado por la versión 2.6 del sistema operativo *Linux*, y proporciona servicios como la seguridad, manejo de memoria, multiproceso y soporte para los *drivers* de los dispositivos. Esta capa actúa como capa de abstracción entre el *hardware* y el resto de la pila, por lo que es la única capa que depende del *hardware*.

La siguiente capa serían el conjunto de librerías en C/C++ que se usan en varios componentes de *Android*. Están compiladas en código nativo del procesador. Algunas de estas librerías son: *System C library*, *Media Framework* o *WebKit*.

El *Runtime* o entorno de ejecución de *Android* está basado en el concepto de máquina virtual que utiliza *Java*. Se creó una máquina virtual llamada *Dalvik*, basada en registros, que ejecuta las aplicaciones compiladas en el formato *.dex*, un formato que facilita la optimización de recursos. A partir de la versión *Android 5.0* se sustituyó la máquina virtual *Dalvik* por otra máquina llamada *ART*, con lo que se consiguió reducir el tiempo de ejecución de código *Java* hasta en un 33%.

El entorno de aplicación o *framework* de *Android* proporciona una plataforma de desarrollo libre para aplicaciones con múltiples opciones. Esta capa ha sido diseñada para simplificar la reutilización de componentes. Incluye los servicios como *Views, Resource Manager, Content Provider, Activity Manager, Notification Manager.* [21]

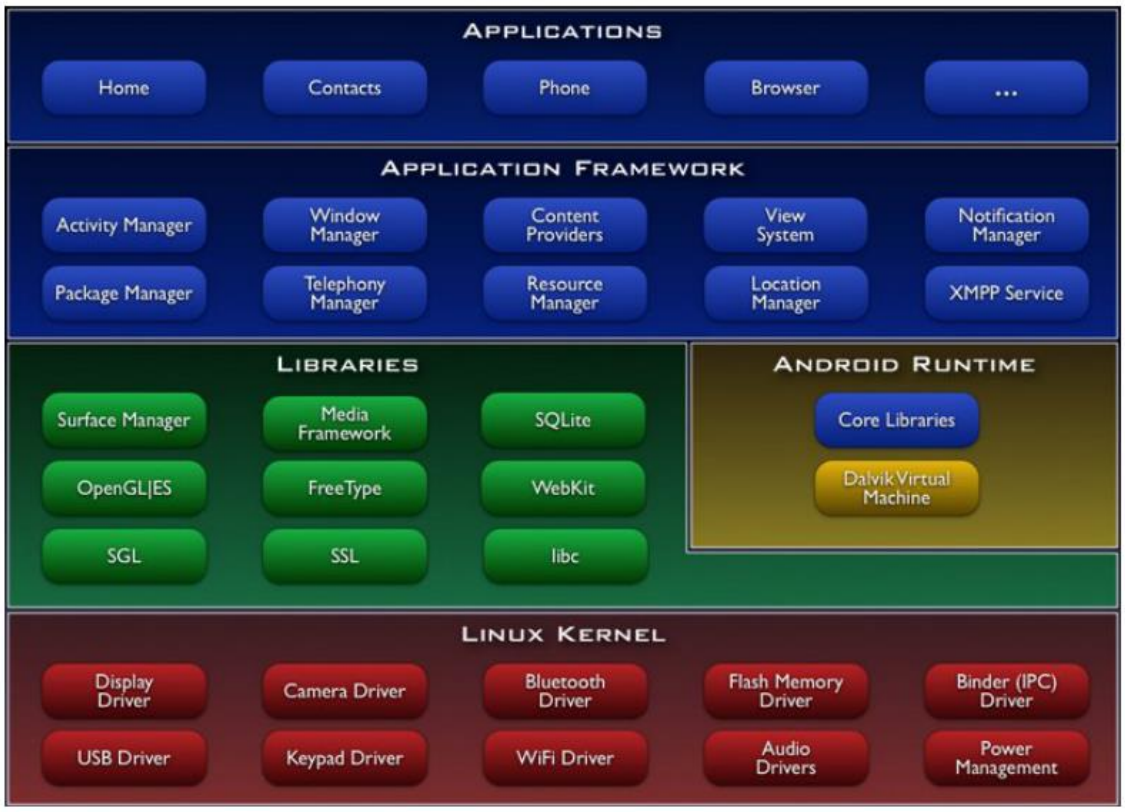


Figura 12. Arquitectura de *Android*

3.8.4 Versiones

Desde la primera versión *Android 1.0 Apple Pie*, hasta la versión que aún se encuentra en fase de prueba *Android 7.0 Nougat*, el sistema operativo se ha actualizado con numerosas versiones. *Google* siempre ha llamado con nombres de postres o dulces a las versiones de su sistema operativo, además de con el número de versión. A continuación, se pretende hacer un análisis de las versiones que más se utilizan y las que han supuesto un mayor cambio para *Android*.

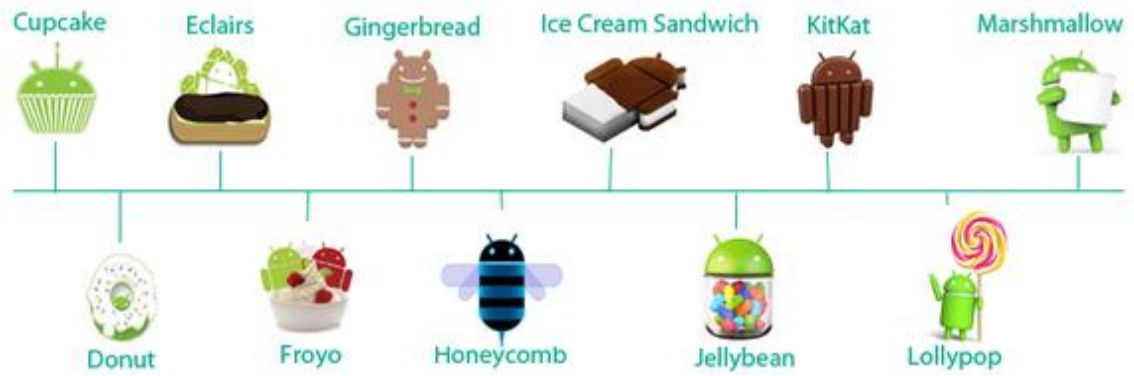


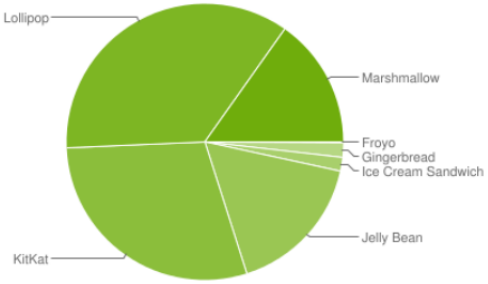
Figura 13. Cronología de las versiones de *Android*

En la figura 14 se pueden observar el porcentaje de dispositivos *Android* que utilizan cada versión del sistema operativo. Estos datos los publica *Google* periódicamente, obteniéndolos a través de su plataforma de aplicaciones *Google Play*, ya que el sistema obtiene la versión de *Android* que utiliza el dispositivo que descarga una aplicación y utiliza esta información para obtener estas estadísticas. Las versiones con menos de un 0.1% de uso no se muestran por ser consideradas irrelevantes.

Como podemos observar, las versiones más utilizadas son las tres más nuevas: *Kitkat*, *Lollipop* (que cuenta con las versiones 5.0 y 5.1) y *MarshMallow*. Un 79.9% de dispositivos *Android* utilizan estas versiones, siendo la versión 4.4 *KitKat* la más extendida, pese a ser la más antigua de las tres. [20]

Al haber tantas versiones de *Android*, este sistema operativo siempre ha tenido el problema de la fragmentación de versiones, ya que algunos dispositivos antiguos no se actualizan. Este problema parece haber sido en parte paliado a partir de la versión 4.4, ya que se han conseguido aglutinar el 80% de los dispositivos entorno a las tres últimas versiones.

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	1.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.6%
4.1.x	Jelly Bean	16	6.0%
4.2.x		17	8.3%
4.3		18	2.4%
4.4	KitKat	19	29.2%
5.0	Lollipop	21	14.1%
5.1		22	21.4%
6.0	Marshmallow	23	15.2%



Data collected during a 7-day period ending on August 1, 2016.

Any versions with less than 0.1% distribution are not shown.

Figura 14. Uso de las versiones de *Android*

Capítulo 4. Objetivos

En este capítulo se pasarán a describir los objetivos que se intentan alcanzar en este Trabajo de Fin de Grado. Como ya indica el propio título del trabajo, el objetivo principal de este es:

- Desarrollar una aplicación móvil para dispositivos *Android* que muestre eventos culturales geoposicionados, mostrando al usuario los eventos que tiene cerca de su ubicación en un radio determinado por el usuario.

Los objetivos generales durante la realización del proyecto son:

- Estudio de las bases de datos *MySQL*, su implementación en el entorno de desarrollo y la creación de una base de datos *MySQL* en un servidor.
- Conocer el lenguaje *SQL* y las bases de datos relacionales.
- Estudio del concepto de *open data* y *Smart City*, concretamente el caso de la ciudad de Valencia.
- Conocer del formato *JSON*, así como sus utilidades y el correcto tratamiento de estos archivos.
- Estudio del lenguaje de programación Java y de un entorno de desarrollo basado en este lenguaje.
- Familiarizarse con el repositorio de librerías *Apache Maven*.
- Estudio del procesamiento de lenguajes naturales *NLP* y su implementación en el sistema para geolocalizar eventos.
- Conocer el protocolo *HTTP* y la arquitectura de los servicios web.
- Estudio del sistema operativo *Android*, la programación de aplicaciones en *Android* y su entorno de desarrollo *Android Studio*.

Además, también se detallarán los objetivos del módulo geolocalizador de noticias y de la aplicación. Primero se describirán los objetivos del programa geolocalizador de noticias:

- Conexión a una base de datos MySQL ubicada en un servidor.
- Extracción de datos de archivos JSON, tratamiento de los datos y su almacenamiento en la base de datos.
- Extracción de noticias de feeds RSS proporcionados, correcto tratamiento de estas y su almacenamiento en la base de datos.
- Relacionar noticias no procesadas de la base de datos con los sitios almacenados en la base de datos, en función del procesado del texto de las noticias y almacenamiento de estas relaciones (*matches*) en la base de datos.
- Capacidad de seleccionar las noticias que de mostrarán en la aplicación, seleccionando a mostrar las mismas noticias que aparecen en los feeds RSS.

Por último, se detallarán los objetivos de la aplicación móvil:

- Capacidad de conectarse desde cualquier dispositivo *Android* al servidor web que proporcionará las noticias geoposicionadas.
- Capacidad de mostrar una lista con todas las noticias que se reciban del servidor.
- Capacidad de mostrar los detalles de la noticia seleccionada por el usuario.
- Capacidad de mostrar en un mapa todas las noticias geoposicionadas.
- Capacidad de mostrar en un mapa todas las noticias geoposicionadas que estén situadas dentro de un radio, seleccionado por el usuario (de 0.5 a 4.5 km, y todos los eventos) a partir de la ubicación del usuario.
- Capacidad de mostrar en un mapa el número de eventos que ocurren en un mismo sitio, pudiendo consultar el usuario todos los eventos de dicho sitio y los detalles de cada noticia.
- Capacidad de mostrar una interfaz amigable y adaptada a los diferentes tamaños de los dispositivos *Android*.

Capítulo 5. Desarrollo del sistema

En este capítulo de la memoria, se explicará y justificará cómo se ha desarrollado el sistema completo, es decir, la etapa del desarrollo y programación del software. Se detallará la arquitectura del sistema completo que hace posible el correcto funcionamiento de la aplicación móvil final, así como la explicación detallada de cada elemento del sistema desarrollado.

5.1 Arquitectura del sistema

En este apartado, se procederá a describir la arquitectura global del sistema desarrollado. Para esto, primero se presentará un escenario con la visión global y esquemática del sistema, y finalmente se describirá el orden cronológico que se ha seguido para su desarrollo.

5.1.1 Escenario global

En la figura que se muestra a continuación (Figura 15), se puede ver un esquema de la arquitectura global del sistema.

Figura 15. Arquitectura global del sistema

Se puede observar que el sistema está compuesto por cuatro elementos:

1. Un equipo donde se ejecutará el **programa** llamado ***RSSAgenda***, que es el encargado de crear la base de datos y actualizarla periódicamente. *RSSAgenda* cargará los *feeds RSS*, las noticias de estos feeds y los sitios de archivos *JSON*. Seguidamente, intentará relacionar a los sitios con las noticias y guardará esta información en la base de datos.
2. La **base de datos** del tipo *MySQL* que *RSSAgenda* irá actualizando. Estará ubicada en un servidor *MySQL*.
3. Un **servicio web** que se ejecutará en un servidor web. Este servicio web responderá a las peticiones de los usuarios de la aplicación móvil, accediendo a la base de datos anterior para poder responder a las peticiones. La respuesta se enviará en formato *JSON*.

4. La **aplicación móvil**, llamada ***Agenda Cultural de Valencia***, que se ejecutará en un terminal móvil *Android*. Para poder mostrar las noticias, la aplicación hará peticiones al servidor web mencionado anteriormente y tratará los datos obtenidos de la respuesta del servidor.

5.1.2 Cronología de desarrollo.

Como hemos explicado en el apartado 2. Gestión y organización, se ha optado por el método de desarrollo de *programación extrema (XP)*, por lo que este proyecto se ha desarrollado primando la adaptabilidad a las condiciones que iban surgiendo durante los meses de desarrollo de este trabajo de fin de grado.

El orden cronológico seguido para la realización del proyecto es el mismo orden que se seguirá en la explicación del desarrollo del sistema de este capítulo. Se han ido realizando pruebas unitarias durante el proceso de desarrollo, para ir comprobando la funcionalidad del software paso a paso y así asegurarse de que se cumplían los requisitos que el tutor había especificado durante la fase de desarrollo.

Estas pruebas unitarias han agilizado bastante el proceso de desarrollo, ya que es más fácil comprobar poco a poco la funcionalidad del sistema y después comprobar la funcionalidad completa, que desarrollar el sistema entero para después y comprobar que se cumplan los requisitos directamente. Este método ha reducido drásticamente el tiempo en la solución de los errores.

5.2 La base de datos

Para el correcto funcionamiento del sistema es necesaria la implementación de una base de datos, para almacenar permanentemente las noticias de los *feeds RSS*, los sitios y los *matches* entre noticias y sitios encontrados.

Se ha decidido implementar una base de datos del tipo *MySQL* en el servidor *MySQL* del laboratorio del *SATRD*, ya que este servidor se encuentra en funcionamiento permanente y se tiene un fácil acceso a los administradores de dicho servidor, para gestionar los posibles problemas que pudieran aparecer.

La base de datos se llamará *georss* y estará formada por 4 tablas: *feeds*, *noticias*, *sitios* y *match*. La estructura de las tablas de la base de datos estará definida por las tablas que se muestran a continuación:

NOTICIAS

Campo	Tipo	Auto-incremental
id	INT UNSIGNED	SI
title	TINYTEXT	
keywords	TINYTEXT	
description	MEDIUMTEXT	
body	MEDIUMTEXT	
timestamp	BIGINT UNSIGNED	
feed	INT UNSIGNED	
processed	BOOLEAN	
discarded	BOOLEAN	

Tabla 1. Estructura de la tabla noticias

SITIOS

Campo	Tipo	Auto-incremental
id	INT UNSIGNED	SI
name	VARCHAR (255)	
lat	DOUBLE	
lon	DOUBLE	

Tabla 2. Estructura de la tabla sitios

FEEDS		
Campo	**Tipo**	**Auto-incremental**
id	INT UNSIGNED	SI
name	VARCHAR (255)	
url	VARCHAR (255)	

Tabla 3. Estructura de la tabla feeds

MATCH		
Campo	**Tipo**	**Auto-incremental**
id_noticia	INT UNSIGNED	
Id_sitio	INT UNSIGNED	

Tabla 4. Estructura de la tabla match

Para la gestión de esta base de datos y la comprobación del correcto funcionamiento del sistema, se han empleado una serie de consultas *SQL*, que se muestran en la figura 16.

```
SELECT `noticias`.id AS id, title, body, name AS lugar,lat,lon,discarded FROM `noticias` INNER JOIN (`match` INNER JOIN `sitios` ON `match`.id_sitio=`sitios`.id)
        ON `noticias`.id=`match`.id_noticia ORDER BY timestamp DESC;
SELECT * FROM `match`;
SELECT * FROM `noticias` ORDER BY id DESC;
SELECT `noticias`.`id` AS `id`,`noticias`.`timestamp` AS `timestamp`,`noticias`.`title` AS `noticia no matcheada`,`noticias`.`keywords` AS `keywords`,
        `noticias`.`body` AS `body` FROM `noticias` WHERE (NOT(`noticias`.`id` IN (SELECT `match`.`id_noticia` FROM `match`)))
        ORDER BY `noticias`.`timestamp` DESC;
SELECT COUNT(name) AS numNoticias,name AS sitio FROM `noticias` INNER JOIN (`match` INNER JOIN `sitios` ON `match`.id_sitio=`sitios`.id)
        ON `noticias`.id=`match`.id_noticia GROUP BY name;
SELECT COUNT(name) AS numNoticias,name AS sitio FROM `noticias` INNER JOIN (`match` INNER JOIN `sitios` ON `match`.id_sitio=`sitios`.id)
        ON `noticias`.id=`match`.id_noticia WHERE `noticias`.discarded=0 GROUP BY name;
SELECT id_noticia AS no_valida FROM `match` WHERE id_noticia NOT IN (SELECT `noticias`.id FROM `noticias` INNER JOIN (`match` INNER JOIN `sitios`
        ON `match`.id_sitio=`sitios`.id) ON `noticias`.id=`match`.id_noticia);
SELECT `noticias`.id AS id, title, body, name AS lugar,timestamp,discarded FROM `noticias` INNER JOIN (`match` INNER JOIN `sitios`
        ON `match`.id_sitio=`sitios`.id) ON `noticias`.id=`match`.id_noticia WHERE discarded=0 ORDER BY timestamp DESC;
```

Figura 16. Consultas *SQL* para gestionar la base de datos

Estas consultas también nos permitirán obtener estadísticas de las noticias que se han conseguido o no *matchear*, así como observar las noticias que deberían mostrarse en la aplicación final.

5.3 RSSAgenda

El programa *RSSAgenda*, como ya se ha descrito anteriormente de forma breve, es el encargado de crear la base de datos y actualizarla periódicamente. *RSSAgenda* cargará los *feeds RSS*, las noticias de estos feeds y los sitios de archivos *JSON* previamente almacenados en el equipo. Finalmente, intentará relacionar los sitios con las noticias no procesadas mediante la técnica del procesado natural de lenguaje *NLP* y guardará esta información en la base de datos.

5.3.1 Escenario global

A continuación, se muestra un esquema global del funcionamiento de *RSSAgenda*.

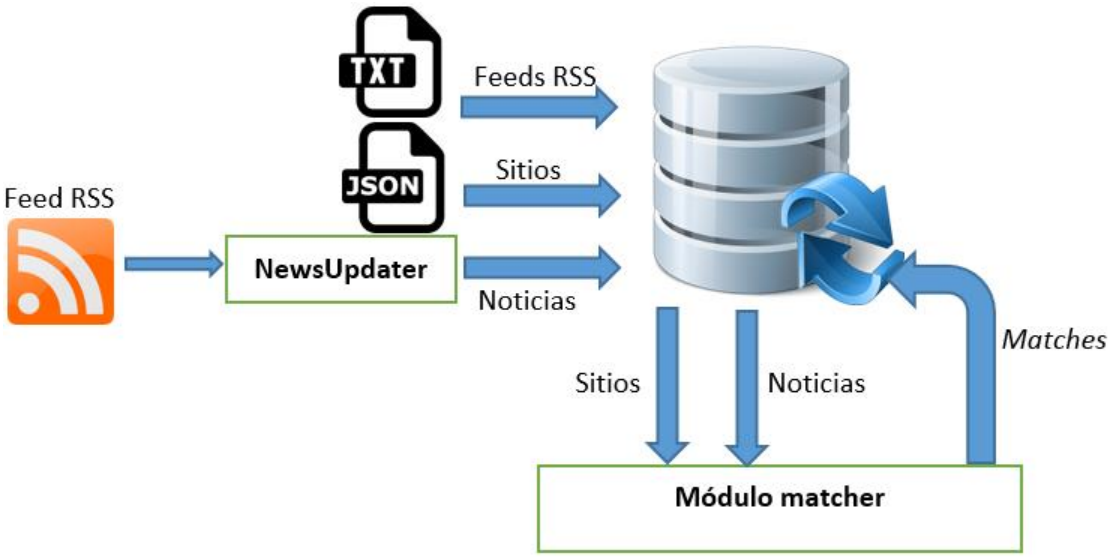


Figura 17. Funcionamiento global de *RSSAgenda*

Podemos diferenciar cuatro partes en el proceso de ejecución del software:

1) La conexión con la base de datos y la creación de las tablas.
2) La extracción de los sitios geoposicionados de archivos *JSON* y la extracción de los *feeds RSS* de un archivo de texto, para finalmente guardar los sitios y los feeds en la base de datos.
3) La extracción de noticias de los diferentes *feeds RSS* con sus detalles y su posterior almacenamiento en la base de datos. Además, la selección de las noticias que se mostrarán en la aplicación.
4) El establecimiento de una relación entre las noticias de los *feeds* y los sitios geoposicionados, es decir, la obtención de los *matches*.

En los apartados siguientes, se explicará con detalle la implementación de cada parte del software, para finalmente hacer un análisis de los resultados obtenidos de la ejecución del programa.

5.3.2 Clases de RSSAgenda

En este apartado se pasarán a detallar las clases que se han creado en el desarrollo del programa *RSSAgenda* con las siguientes figuras, con lo que se pretende que se observe una visión global del diagrama de clases del programa.

Primero, conviene explicar que *RSSAgenda* tiene una estructura de tres paquetes de clases llamados: *agenda.rss*, *agenda.rss.db* y *agenda.rss.matcher*, cuyas clases se pueden observar en la figura siguiente:

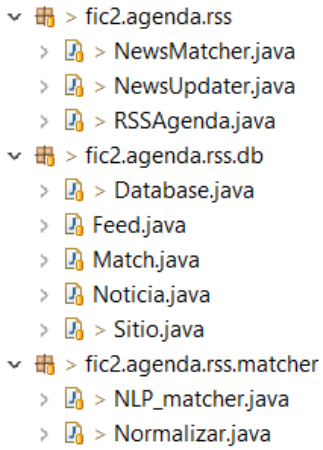


Figura 18. Estructura general de paquetes y clases de *RSSAgenda*

De cada clase de describen sus elementos que las definen, que son sus atributos y métodos. Además, también se indica el tipo de datos que retorna cada método, así como si el método o atributo es el tipo *Static*.

Finalmente, se presentará un diagrama con todas las clases y las interacciones entre ellas, para que se acabe de obtener una visión del funcionamiento de las clases del software desarrollado.

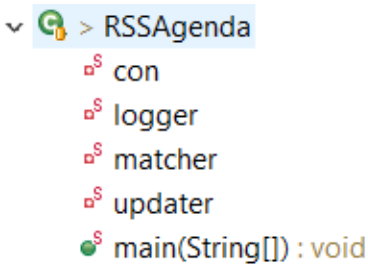


Figura 19. Clase RSSAgenda

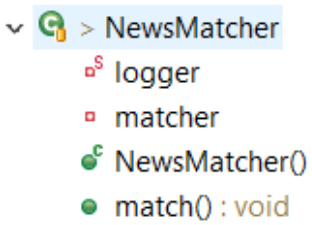


Figura 20. Clase NewsMatcher

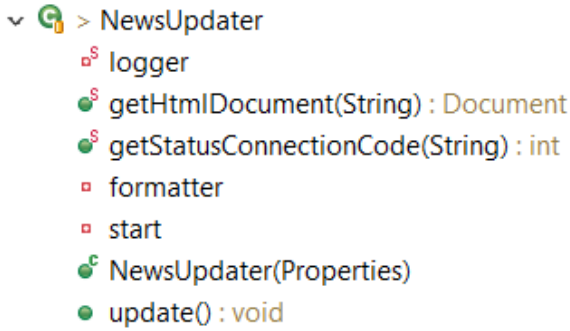


Figura 21. Clase NewsUpdater

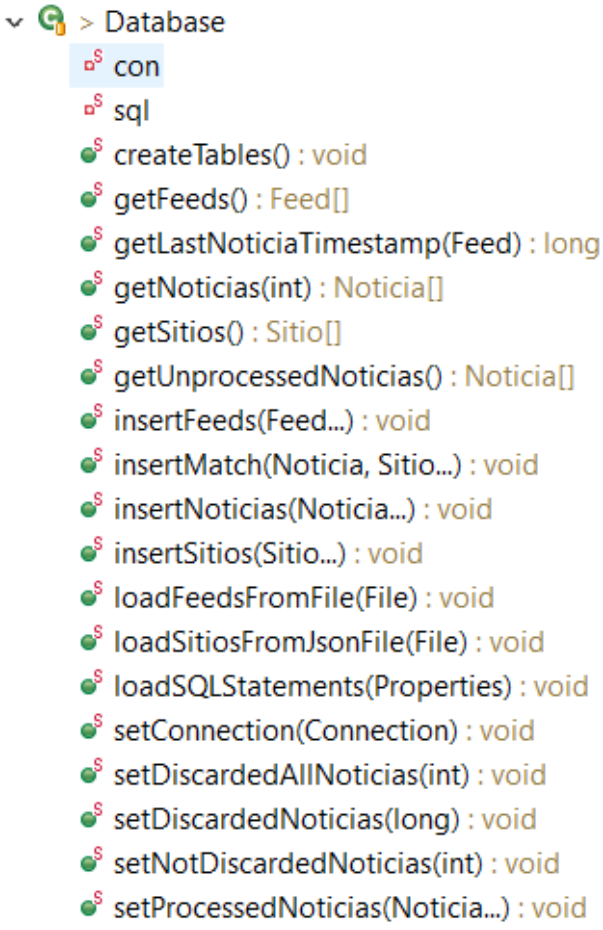


Figura 22. Clase Database

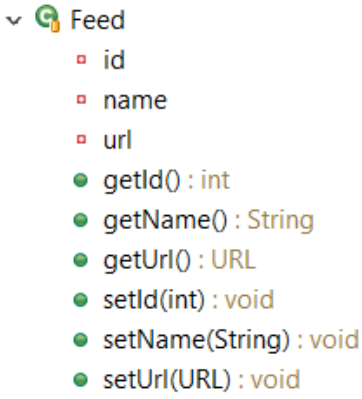


Figura 23. Clase Feed

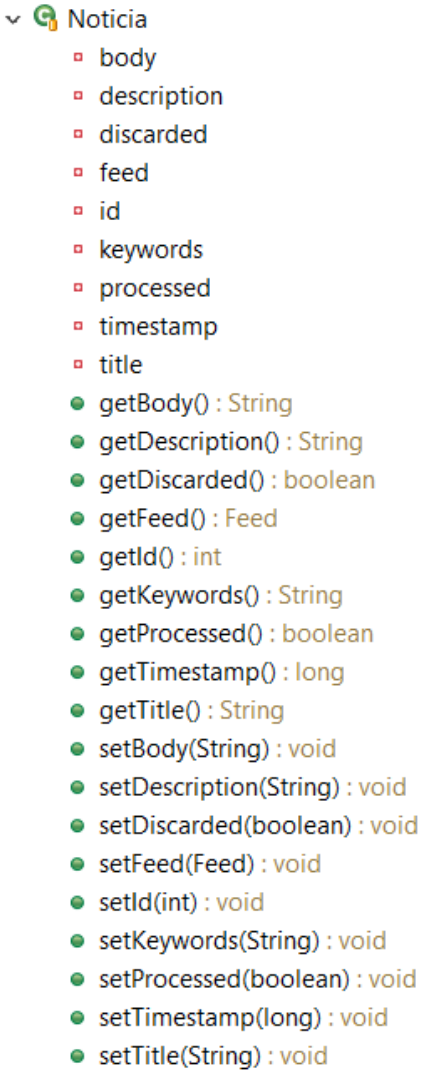


Figura 24. Clase Noticia

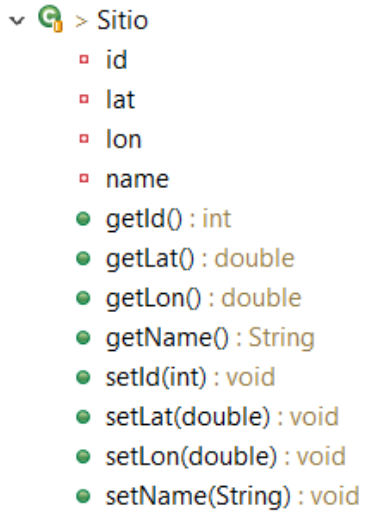


Figura 25. Clase Sitio

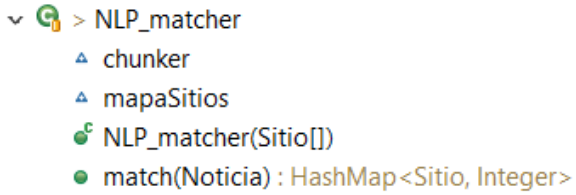


Figura 26. Clase NLP_matcher

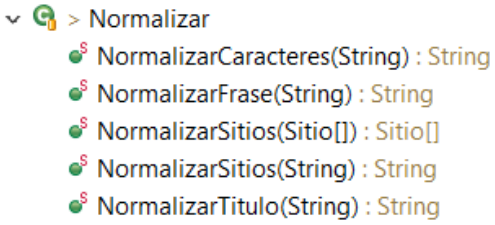


Figura 27. Clase Normalizar

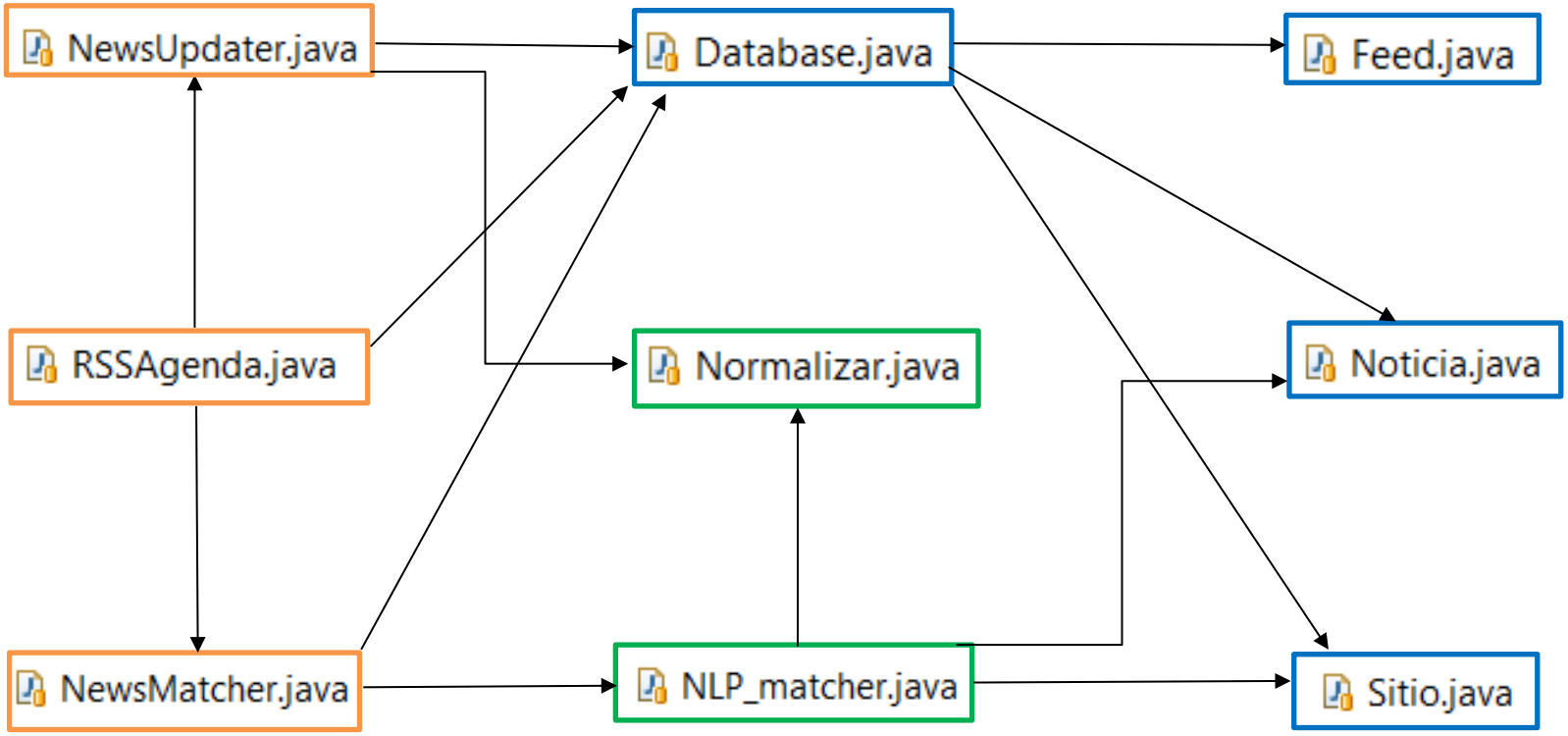


Figura 28. Relación entre clases de *RSSAgenda*

5.3.3 Inicialización de la base de datos

El primer paso en el desarrollo del programa *RSSAgenda*¸ es la conexión con la base de datos ubicada en el servidor del laboratorio del *SATRD* y la creación de las tablas que tendrá la base de datos. La estructura de las tablas será la que se ha mostrado en el apartado 5.2, y las instrucciones *SQL* utilizadas para su creación son las siguientes:

```
CREATE TABLE IF NOT EXISTS noticias (id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT, title TINYTEXT NOT NULL, keywords TINYTEXT NOT NULL,
                        description MEDIUMTEXT NOT NULL, body MEDIUMTEXT NOT NULL, timestamp BIGINT UNSIGNED NOT NULL,
                        feed INT UNSIGNED NOT NULL, processed BOOLEAN NOT NULL, discarded TINYINT NOT NULL);
CREATE TABLE IF NOT EXISTS sitios (id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT, name TINYTEXT NOT NULL, lat DOUBLE NOT NULL,
                        lon DOUBLE NOT NULL);
CREATE TABLE IF NOT EXISTS feeds (id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT, name VARCHAR(255) NOT NULL, url VARCHAR(255) NOT NULL,
                        UNIQUE(name, url));
CREATE TABLE IF NOT EXISTS `match` (id_noticia INT UNSIGNED, id_sitio INT UNSIGNED, UNIQUE(id_noticia,id_sitio));
```

Figura 29. Instrucciones *SQL* utilizadas para la creación de las tablas de la base de datos

5.3.4 Carga de feeds y sitios

Una vez comprobada la correcta conexión con la base de datos y la correcta creación de las tablas, el siguiente paso será insertar en la base de datos los *feeds* de los que se extraerán las noticias y los sitios geoposicionados.

Para seleccionar los *feeds RSS* de los que extraer las noticias, se ha hecho previamente un proceso de selección para que las noticias se adecuen a los eventos culturales que se quieren mostrar en la aplicación. Para esto se han analizado los diferentes *feeds RSS* de periódicos como *Las Provincias* o *Levante-EMV*¸ pero finalmente se ha decido dar por válidos solo los siguientes *feeds*:

1. La agenda de eventos del mismo Ayuntamiento de Valencia (http://www.valencia.es/ayuntamiento/agenda_accesible.nsf/agenda.xml).
2. La agenda de eventos de la web del turismo de la ciudad de Valencia (http://www.visitvalencia.com/es/Rss.aspx?tipo=agenda&idi=es).

Los enlaces de estos canales de noticias se guardarán en un fichero de texto, para que *RSSAgenda* se encargue de cargar este fichero y guardar estos *feeds* en la base de datos.

El Ayuntamiento de Valencia dispone de una web de datos abiertos (http://gobiernoabierto.valencia.es) dentro del marco del proyecto de convertir a Valencia en una *Smart City*, proporcionando a los ciudadanos transparencia y datos sobre la ciudad, algunos de ellos en tiempo real [2]. Para este proyecto se han seleccionado dos conjuntos de datos de esta web: *Equipamientos municipales* (bibliotecas, teatros, museos, …) y *Monumentos turísticos*. Además, se ha decido crear manualmente un nuevo conjunto de datos llamado *NuevosSitios* con diferentes sitios que no aparecen en los otros conjuntos de datos.

Estos datos abiertos se pueden descargar en diferentes formatos, pero se ha decidido descargarlos en formato *JSON*, ya que es un formato simple y fácil de tratar para el desarrollador. Para poder tratar los datos de estos archivos, se ha implementado un módulo lector de datos *JSON* utilizando la librería *gson* de *Google*¸ en el que se seleccionará solo la información más relevante de este conjunto de datos: el **nombre** del sitio, su identificador único (**id**) y las **coordenadas geográficas**, que están dadas en formato *UTM*, por lo que es necesario convertirlas al formato de latitud y longitud con la ayuda de la librería *WorldWind* de la *NASA*, ya que *Google Maps* trabaja con el formato de latitud y longitud. Finalmente, esta información se almacenará en la base de datos.

5.3.5 Carga y actualización de las noticias

Con la carga de los *feeds* y de los sitios geoposicionados, para tener los tres elementos sobre los que buscar *matches*, solo falta implementar un módulo que lea las noticias de los *feeds* y las guarde en la base de datos: el módulo *NewsUpdater*. Además de la carga de noticias, este módulo también seleccionará las noticias de cada *feed* que se mostrarán en la aplicación, ya que no sería lógico que se mostrarán eventos ya realizados en la aplicación.

Para poder implementar este módulo, es necesario estudiar el código *XML* de los dos *feeds* que se han seleccionado para que proporcionen las noticias al sistema. Ambos *feeds* tiene una estructura general muy similar, pero al observarse diferencias en la descripción de los detalles de cada entrada o noticia del *feed*, por lo tanto, se ha optado por tratar cada *feed* por separado, ya que al haber solo dos *feeds* es una tarea sencilla y de resultados muy efectivos.

Cada noticia tiene nueve atributos:

- **Id**: identificador numérico único de cada noticia.
- **Title**: el titular de la noticia.
- **Description**: breve descripción de la noticia que aparece en el *feed* de noticias.
- **Keywords**: palabras clave de la noticia que se utilizarán para en el proceso de *matcheo* de la noticia con los sitios. Solo es útil para el *feed* del Ayuntamiento de Valencia.
- **Body**: el cuerpo de la noticia, donde aparece toda la información y detalles.
- **Timestamp:** número de segundos que han pasado desde el instante 00:00:00 UTC del 1 de enero de 1970 hasta el instante de publicación de la noticia. Se podría entender como una marca temporal.
- **Feed:** *feed* al que pertenece la noticia.
- **Processed:** booleano que indica si la noticia ha sido procesada, es decir, si ya ha pasado por el proceso de intentar relacionarla con un sitio en el módulo *NewsMatcher*.
- **Discarded:** booleano que indica si la noticia se muestra en la aplicación o si ha sido descartada.

Para el tratamiento de estos *feeds*, se ha utilizado la herramienta *Rome Feeds*, incluyendo las librerías *Rome* y *Rome Utils* en el proyecto de *RSSAgenda*. Con esta herramienta se pueden extraer las características de los *feeds RSS*, y lo más importante: sus noticias con los detalles que aporta el *feed* de cada una de ellas. Esto es posible gracias a que esta herramienta incorpora un lector de *XML* y métodos con los que se extrae la información automáticamente.

Para ahorrar tiempo de procesamiento y evitar cargar noticias antiguas, se ha establecido como fecha de partida el 1 de enero de 2016 a las 00:00 horas, por lo que las noticias anteriores a esta fecha no serán tratadas por el módulo *NewsUpdater*. Además, este módulo solo tratará las noticias con una fecha de publicación posterior a la de la última noticia de cada *feed* guardada en la base de datos, es decir, con un valor de *timestamp* mayor.

El siguiente paso será dar valor a los atributos de las noticias con la herramienta descrita anteriormente. El título, la descripción, el feed y el *timestamp* de las noticias se establecen con la información extraída de los *feeds*¸ mientras que los atributos *processed* y *discarded* se establecen como falso y verdadero (son booleanos), respectivamente. Las noticias recién añadidas no estarán ni procesadas ni se mostrarán en la aplicación por defecto.

Las noticias del *feed* del Ayuntamiento de Valencia no tienen una descripción breve de la noticia, sino que su descripción solo contiene la repetición del titular, por lo que ha habido que buscar una alternativa para obtener el cuerpo de la noticia. Sin embargo, estas noticias tienen un enlace desde el que se puede acceder a la vista completa de la noticia en la página web del ayuntamiento. Al tener todas las noticias de esta web la misma estructura, se ha optado por utilizar una herramienta conocida como *html parser* para extraer el cuerpo de la noticia de la página web. Además, en la página web de cada noticia aparecen unos campos denominados *keywords*¸ que

son unas palabras clave relacionadas con la noticia, y que pueden ser de gran utilidad a la hora de relacionar estar noticias con los sitios mediante la técnica de *NLP*. Estos *keywords* también se extraerán utilizando el *Jsoup Html Parser*.

Respecto al *feed* de *TurisValencia*, al no tener las noticias un link para acceder a la vista detallada de la noticia, no se ha podido emplear la técnica del *html parser*. Al atributo *keywords* se le ha asignado como valor la palabra "no" y el cuerpo de la noticia se ha obtenido de la descripción de cada noticia que aporta el propio *feed RSS*.

Una vez se han cargado todas las noticias nuevas con sus atributos de los *feeds*, estas noticias se insertarán en la base de datos.

Figura 30. Funcionamiento de la carga de noticias de *NewsUpdater*

La otra parte del módulo *NewsUpdater* es la selección de noticias que se mostrarán en la aplicación *Android*. En una aplicación donde se muestran eventos culturales de una ciudad, no resulta lógico ni conveniente de cara al usuario final mostrar eventos que ya han tenido lugar, por lo que es necesario implementar una función que seleccione los eventos que mostrará la aplicación.

En un primer momento se pensó en mostrar las noticias durante 15 días desde su fecha de publicación (o *timestamp* en el caso de este proyecto), pero esta solución no era conveniente, ya que se seguían mostrando eventos pasados y no se mostraban algunos eventos que aún tenían vigencia pasados estos 15 días, como exposiciones o ciclos de conciertos.

Finalmente, se optó por mostrar las mismas noticias que aparecen en los *feeds RSS* de los que se han extraído, marcando primero las noticias como descartadas para luego marcar como no descartadas solo las que aparecen en los *feeds RSS* después de un proceso de comparación.

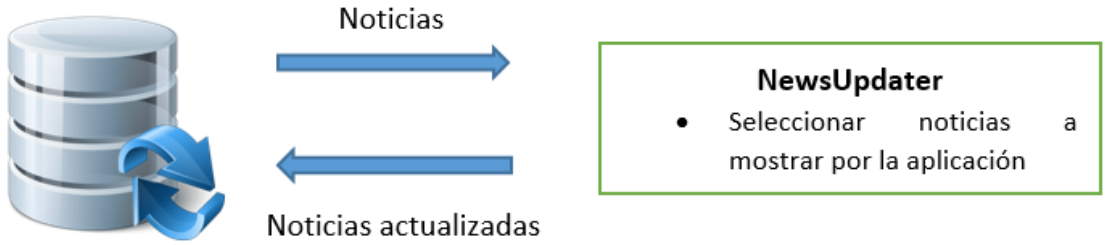


Figura 31. Funcionamiento de la actualización de noticias de *NewsUpdater*

5.3.6 Módulo matcher

Hasta este punto, el programa *RSSAgenda* es capaz de almacenar en la base de datos *feeds RSS*, sitios de archivos *JSON* y noticias extraídas de *feeds RSS*. Con esto, la funcionalidad del programa no es la que se pretende, ya que para la aplicación final se necesitan tener las noticias geoposicionadas y no una simple lista de noticias de eventos. Por lo tanto, en el módulo final de *RSSAgenda* se intentarán relacionar las noticias no procesadas de la base de datos con los sitios geoposicionados de la base de datos mediante el procesado del texto de las noticias, concretamente mediante la técnica de *NLP* y el *Chunking*.

Estas relaciones o *matches* se intentarán conseguir detectando menciones de sitios en las noticias. Por ejemplo, en un evento en el que leemos en el cuerpo de la noticia que tiene lugar en Plaza del Ayuntamiento, el programa debe de ser capaz de *matchearla* con el sitio *Plaza del Ayuntamiento*, y así obtendrá una noticia geoposicionada, ya que el sitio con el que se relaciona la noticia tiene una latitud y una longitud (es decir, una coordenada geográfica) asociadas.

El esquema siguiente explica el funcionamiento del módulo *matcheador* de noticias con los sitios:

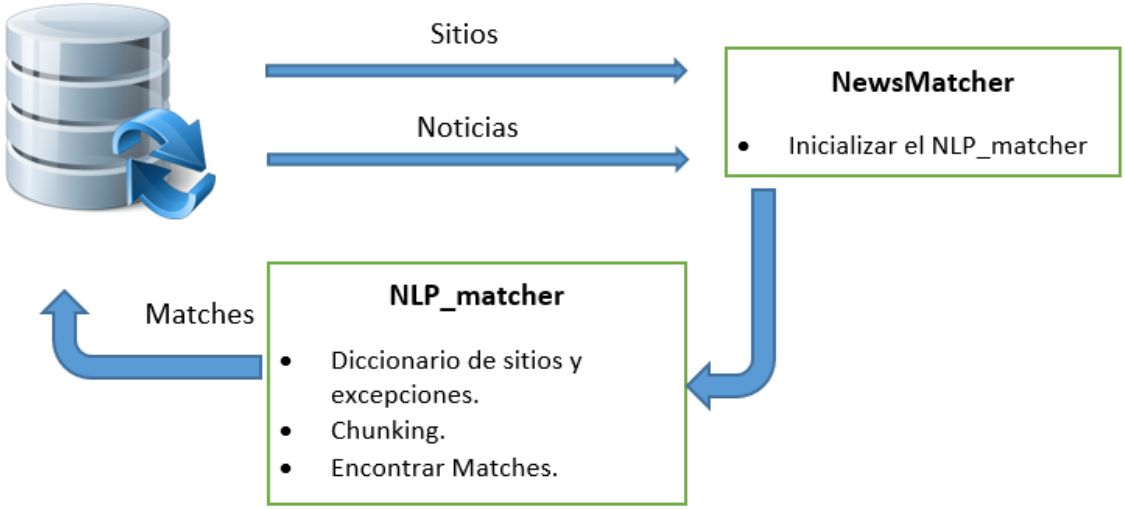


Figura 32. Funcionamiento del módulo *matcher*

En primer lugar, es necesario obtener los sitios y las noticias no procesadas de la base de datos. Con estas dos colecciones de elementos, ya se puede implementar el módulo *NLP_matcher*, que será el que relacionará las noticias con los sitios que aparecen en ellas.

Para el establecimiento de estas relaciones, se ha optado por la técnica de *procesamiento de lenguajes naturales (NLP)* con la ayuda de la herramienta *Lingpipe* (herramienta desarrollada para la técnica de *NLP* y el análisis de texto), que intentará reconocer los sitios que aparecen en las noticias y así detectar los *matches*. Para esto, es necesario normalizar los sitios y los *keywords* o la descripción de las noticias, es decir, pasar el texto a mayúsculas y eliminar los acentos. Además, también se ha creído conveniente traducir algunas palabras que aparecen en valenciano al castellano para evitar futuros problemas, como las palabras teatre (teatro) o universitat (universidad).

Otro aspecto importante para obtener éxito en el futuro uso de *Lingpipe* para obtener los *matches*, es crear un diccionario con todos los sitios de la base de datos para ayudar a esta herramienta a reconocer los sitios en los *keywords* o la descripción de las noticias. Además, también es muy recomendable incluir excepciones en este diccionario, ya que en las noticias estos sitios no siempre aparecen con el mismo nombre con el que están guardados en la base de datos. Estas excepciones se han ido añadiendo una a una después de ejecutar numerosas pruebas unitarias del software desarrollado, como marca uno de los principios de la *programación extrema*, método que se ha seguido durante el proceso de desarrollo.

```
if(sitio.getName().contains("PALAU DE LES ARTS"+" \"REINA SOFIA\"")){
    dictionary.addEntry(new DictionaryEntry<String>(sitio.getName(), Integer.toString(sitio.getId())));
    dictionary.addEntry(new DictionaryEntry<String>(sitio.getName().replaceAll("PALAU DE LES ARTS"+" \"REINA SOFIA\"", "PALAU DE LES ARTS"),Integer.toString(sitio.getId())));
}else if(sitio.getName().contains("MUSEO DE LAS CIENCIAS"+" \"PRINCIPE FELIPE\"")){
    dictionary.addEntry(new DictionaryEntry<String>(sitio.getName(), Integer.toString(sitio.getId())));
    dictionary.addEntry(new DictionaryEntry<String>(sitio.getName().replaceAll("MUSEO DE LAS CIENCIAS"+" \"PRINCIPE FELIPE\"", "MUSEO DE LAS CIENCIAS"),Integer.toString(sitio.ge
}else if(sitio.getName().contains("TEATRO RIALTO")){//la filmoteca del ivac se encuentra en el Teatro Rialto
    dictionary.addEntry(new DictionaryEntry<String>(sitio.getName(), Integer.toString(sitio.getId())));
    dictionary.addEntry(new DictionaryEntry<String>(sitio.getName().replaceAll("TEATRO RIALTO", "FILMOTECA DEL IVAC"),Integer.toString(sitio.getId())));
```

Figura 33. Ejemplo de excepciones de sitios

Con este diccionario creado, el siguiente paso para poder utilizar la herramienta *Lingpipe* y la técnica del *chunking* será la creación de un objeto llamado *chunker*¸ que estará compuesto por el diccionario que se ha creado, un diccionario de caracteres y construcciones propias del lenguaje indoeuropeo y unos parámetros configurables relacionados con el *chunking*, que se han dejado en los valores por defecto recomendados.

Una vez establecidos y configurados los parámetros para poder utilizar estas herramientas para el *chunking*, el siguiente paso será buscar los *matches*. Como se ha explicado anteriormente, para esto se hará un tratamiento diferente para las noticias según al *feed* que pertenezcan. En las noticias que pertenezca *feed* del Ayuntamiento de Valencia se intentarán detectar las menciones de sitios en los *keywords* normalizados de la noticia, mientras que en las noticias que pertenezcan al *feed* de *TurisValencia* se intentarán buscar menciones en el cuerpo de la noticia.

El proceso de *chunking* dará como resultado una puntuación entre el 0 y 4 indicando el nivel de afinidad del nombre del sitio con la palabra (del cuerpo o de los *keywords* de la noticia) seleccionada como *match*. Una puntuación de 0 indica un alto nivel de afinidad (misma palabra) mientras que una puntuación de 4 indica un nivel bajo de afinidad.

Durante este proceso, hay bastantes noticias para las que se encuentran *matches* o coincidencias con más de un sitio. Por lo tanto, se ha visto como necesario la creación de un algoritmo para que cada noticia se quede con el *match* con menor puntuación, es decir, con el sitio con el que hay un mayor nivel de coincidencia.

Para finalizar el proceso, solo se insertarán en la base de datos los *match* con una puntuación menor que 1, para evitar posibles *matches* defectuosos o que no coincidan en su totalidad.

5.3.7 Análisis de resultados

Una vez se ha explicado completamente el proceso de desarrollo y el funcionamiento del programa *RSSAgenda*, conviene analizar los resultados obtenidos en su ejecución. La siguiente figura detalla la información proporcionada por el programa durante su ejecución:

```
05/sep/2016 13:49:00,519 - RSSAgenda: Loading configuration from file: configuration.properties
05/sep/2016 13:49:00,520 - RSSAgenda: Loading database configuration from file: mysql.properties
05/sep/2016 13:49:00,521 - RSSAgenda: Connecting to database...
05/sep/2016 13:49:00,763 - RSSAgenda: Loading SQL statements...
05/sep/2016 13:49:00,764 - RSSAgenda: Loading feeds from file: rssagendaculturalvalencia.txt
05/sep/2016 13:49:05,272 - NewsUpdater: Retrieving news from: Agenda Ajuntament de València-http://www.valencia.es/ayuntamiento/agenda_accesible.nsf/agenda.xml
05/sep/2016 13:49:05,419 - NewsUpdater: Agenda Ajuntament de València: 30 entries
05/sep/2016 13:49:06,356 - NewsUpdater: Inserting: 1 new noticias
05/sep/2016 13:49:06,435 - NewsUpdater: Seleccionando noticias a mostrar...
05/sep/2016 13:49:08,244 - NewsUpdater: Noticias a mostrar actualizadas
05/sep/2016 13:49:08,245 - NewsUpdater: Retrieving news from: Agenda VLC-http://www.turisvalencia.es/Rss.aspx?tipo=agenda&idi=es
05/sep/2016 13:49:10,886 - NewsUpdater: Agenda VLC: 116 entries
05/sep/2016 13:49:10,891 - NewsUpdater: Inserting: 0 new noticias
05/sep/2016 13:49:10,892 - NewsUpdater: Seleccionando noticias a mostrar...
05/sep/2016 13:49:14,923 - NewsUpdater: Noticias a mostrar actualizadas
05/sep/2016 13:49:14,923 - NewsUpdater: 1 noticias insertadas de todos los feeds
05/sep/2016 13:49:14,938 - NewsMatcher: 1 new noticias to match
05/sep/2016 13:49:14,938 - NewsMatcher: 1/1 searching matches for noticia 936
keyword de la noticia: ,VALENCIA, AYUNTAMIENTO, MUSICA, AYUNTAMIENTO. PLAZA DEL AYUNTAMIENTO CORO Y RONDALLA DEL CENTRO MUNICIPAL DE ACTIVIDADES PARA PERSONAS
score de chunkarray[0] 3.0 vs score de chunkarray[0+1] 0.0
index min score=1
score de chunkarray[1] 0.0 vs score de chunkarray[1+1] 2.0
index min score=1
MIN SCORE FINAL=0.0
Sitio matcheado: AYUNTAMIENTO. PLAZA DEL AYUNTAMIENTO
05/sep/2016 13:49:15,009 - NewsMatcher: Match found!
05/sep/2016 13:49:15,111 - NewsMatcher: 1/1 news matched
05/sep/2016 13:49:15,111 - RSSAgenda: Done
```

Figura 34. Proceso de ejecución de *RSSAgenda*

Desde que el programa *RSSAgenda* se ha terminado de desarrollar y ha funcionado correctamente, se han insertado en la base de datos 620 noticias, de las cuales se han conseguido *matchear* con éxito 595, es decir, un 95.97% de las noticias totales, una cifra muy alta y que confirma la fiabilidad del programa desarrollado, así como el cumplimiento de los requisitos iniciales. Sólo se han quedado sin *matchear* 25 noticias de 620, es decir, un 4.03% de las noticias totales.

Existen diversas causas por las que algunas noticias no se consiguen relacionar con algún sitio en concreto, pero analizando los resultados las dos causas principales podrían ser:

- El sitio donde ocurre el evento no está en la en los archivos *JSON* proporcionados por la web de datos abiertos del ayuntamiento, y el sitio no se ha considerado lo suficientemente relevante para incluirlo manualmente en el archivo *NuevosSitios.JSON*.
- El evento no ocurre en un sitio en concreto, como, por ejemplo, un desfile que recorre las calles de la ciudad o un espectáculo itinerante.

En resumen, con estas estadísticas obtenidas mediante consultas *SQL* ejecutadas en la base de datos, se puede afirmar que el programa cumple claramente con los objetivos y requisitos definidos por el cliente/tutor a lo largo del proceso de desarrollo.

5.4 El servicio web

No es conveniente realizar una conexión *JDBC* directamente desde el código *Java* de la aplicación *Android* a la base de datos, ya que *JDBC* presenta muchos problemas cuando se implementa en las aplicaciones *Android*. Otro motivo de peso importante es por la seguridad, ya que al utilizar *JDBC* el usuario y la contraseña que permiten el acceso a la base de datos están escritos en plano en el código fuente.

Por estas razones es conveniente la creación de un servicio web simple que actúe como puente entre la base de datos y la aplicación *Android.* El dispositivo *Android* que ejecute la aplicación enviará una petición *HTTP* con el método *GET* al servicio web, este realizará la consulta a la base de datos, y enviará la respuesta *HTTP* que contendrá los datos que se pedían en formato *JSON* al dispositivo móvil.

Este servicio web estará ubicado en el servidor del laboratorio del *SATRD* y tendrá dos recursos distintos:

- **Agenda Resource:** devuelve todos los eventos que se han conseguido *matchear* y que estén seleccionados para mostrarse en la aplicación.
- **Matches Cercanos Resource:** devuelve los *matches* no descartados que se encuentran dentro de un radio euclídeo determinado respecto a un punto geográfico determinado.

5.4.1 Estructura general

Para el desarrollo del servicio web, se ha aprovechado un servicio web ya programado de código abierto en el que las respuestas del servidor a las peticiones se envían en formato *JSON*. Este servicio web solo tiene programada la estructura, así que es necesario personalizarlo con los recursos que debe ofrecer y con la correcta conexión y operaciones con la base de datos.

La estructura se muestra a continuación:

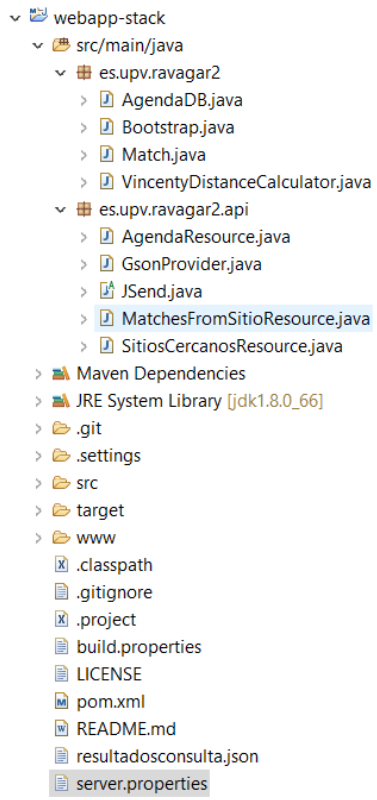


Figura 35. Estructura del proyecto del servicio web

La estructura de las clases que sí se han desarrollado y no estaban en el proyecto de código abierto son las que se muestran a continuación, con sus atributos y sus métodos:

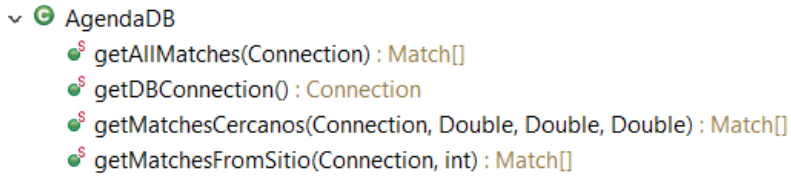


Figura 36. Clase AgendaDB

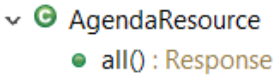


Figura 37. Clase Agenda Resource

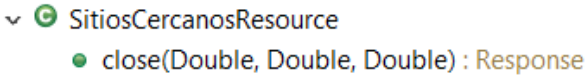


Figura 38. Clase SitiosCercanosResource

5.4.2 La clase Match

Para el retorno de los datos en el archivo *JSON*, se ha creado un objeto llamado *Match*, los atributos del cuál serán los elementos que se devuelvan como respuesta en el archivo *JSON*. Como se puede ver en la figura 33, estos elementos serán: el **título** y el **cuerpo** de la noticia, el **lugar** donde ocurre y la **latitud** y la **longitud** que indican el punto geográfico exacto de este lugar.

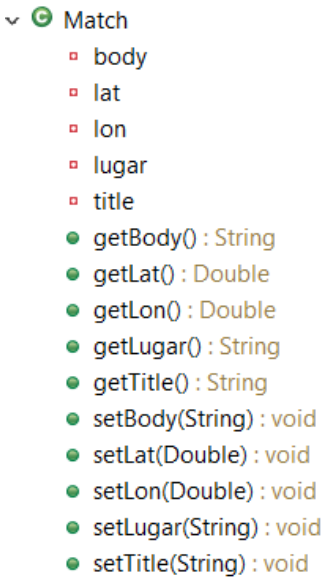


Figura 39. Clase Match

5.4.3 Agenda Resource

Este recurso del servicio web tiene como objetivo devolver todos los eventos que se han conseguido *matchear* y que estén seleccionados para mostrarse en la aplicación. Para esto se ha considerado que la mejor opción es utilizar el método *GET*, no siendo necesario pasar parámetros con la petición *HTTP GET*.

El enlace para la petición de este recurso es http://158.42.188.5:8080/agenda/api/agenda/all y el formato de la respuesta del servidor a la petición se puede observar en la siguiente imagen:


```
{
  "data": [
    {
      "lat": 39.46972276696454,
      "lon": -0.3771533083573021,
      "title": "Coro y Rondalla del Centro Municipal de Actividades para Personas Mayores (CMAPM) San Marcelino. ",
      "lugar": "AYUNTAMIENTO. PLAZA DEL AYUNTAMIENTO",
      "body": "Coro y Rondalla del Centro Municipal de Actividades para Personas Mayores (CMAPM) San Marcelino.. \nFECHA: jueves 8 de septiembre de 2016, alas 12.00 horas ENTRADA libre. Aforo limitado PROGRAMA: 1º Valencia Mia. 2º Amor que viene cantando. 3º Marinero. 4º Torre Vieja. 5º La Jota de mi balcón. 6º Las Espigadoras, (La Rosa del Azafrán) 7º ¡Hay! mi sombrero. 8º La Hiedra. 9º Los Gitanos. 10º María la Portuguesa. 11º Francisco Alegre. 12º Aromas de Valencia. 13º ¡Hay! Jalisco no te rajes. 14º Libre. ( Homenaje a Nino Bravo) 15º Las Tardes del Ritz. 16º Si tu me dices. 17º Como la espiga del trigo. 18º Viva Valencia ( Poupurri folclórico Valenciano.). \n"
    },
    {
      "lat": 39.47834715477485,
      "lon": -0.3835001276157386,
      "title": "Festival I Cicle de recitals de guanyadors i pianistes valencians del Concurs Internacional de Piano de València Premi Iturbi ",
      "lugar": "CENTRO CULTURAL LA BENEFICENCIA",
      "body": "Festival I Cicle de recitals de guanyadors i pianistes valencians del Concurs Internacional de Piano de València Premi Iturbi. \nFECHAS: 15,16,19,20,21,22,23,24,y 29 de septiembre HORARIO:19.30 horas PRECIO: Entradas gratuitas. \nConcerts a la Beneficència. \n"
    },
```


Figura 40. Respuesta del servidor

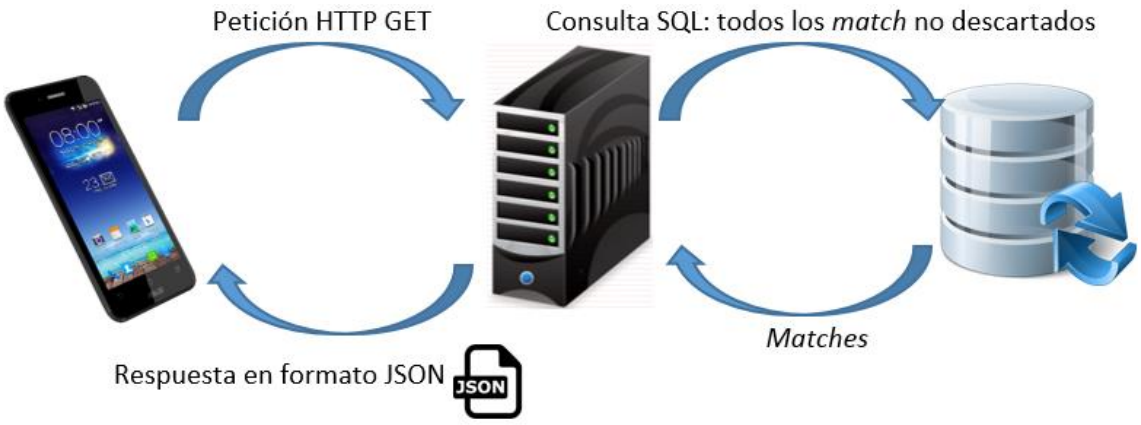


Figura 41. Esquema del funcionamiento del *Agenda Resource*

5.4.4 Matches Cercanos Resource

Este recurso del servicio web se ha diseñado utilizando el método *GET* con tres parámetros a introducir por el usuario en la petición:

- Un **radio** de búsqueda en quilómetros.
- Una coordenada geográfica definida por una **latitud** y una **longitud**.

La respuesta a la petición serán los *matches* que se encuentren dentro del radio determinado respecto al punto geográfico indicado por la latitud y longitud introducidas. La base de datos proporcionará todos los *matches* no descartados y la función que se ejecutará en el servidor seleccionará los *matches* que cumplan los requisitos especificados en la petición.

El enlace para la petición de este recurso es http://158.42.188.5:8080/agenda/api/matchescercanos/all?lat={latitud}&lon={longitud}&rad={radio} , por ejemplo:

http://158.42.188.5:8080/agenda/api/matchescercanos/all?lat=39.473488&lon=-0.376296&rad=5

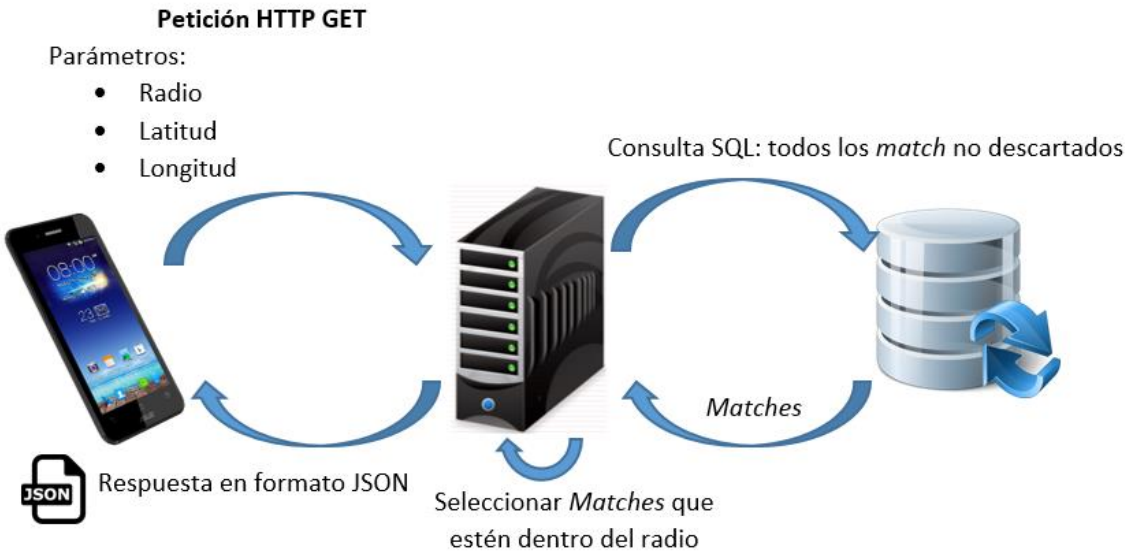


Figura 42. Esquema del funcionamiento del *Matches Cercanos Resource*

5.5 La aplicación *Agenda Cultural de Valencia*

Llegados a este punto, ya se disponen de los datos necesarios para que funcione la aplicación móvil, por lo que el siguiente paso será desarrollar la aplicación para dispositivos *Android*, con una interfaz amigable para el usuario y que cumpla los requisitos definidos.

La aplicación se llamará *Agenda Cultural de Valencia* y tendrá dos opciones principales:

- La visualización de todos los eventos en una lista, y la posterior visualización de los detalles del evento que se seleccione.
- La visualización de los eventos en un mapa, agrupados por el sitio donde ocurren, que se encuentren dentro de un radio determinado de la ubicación del usuario.

Para que la aplicación cumpla con los requisitos y se puedan implementar estas dos opciones, es necesario disponer de una conexión a internet en el dispositivo y obtener unos datos fiables de la ubicación del dispositivo, haciendo uso del servicio de ubicación del mismo dispositivo.

Antes de avanzar con la explicación de la aplicación, es necesario introducir algunos conceptos relacionados con *Android*, como las **actividades** y los **layouts**. Una aplicación está formada por un conjunto de elementos básicos de visualización conocidos como actividades (que son clases de java), que en un lenguaje coloquial serían las pantallas de la aplicación. Por otra parte, un *layout* es un conjunto de vistas agrupadas de una determinada forma, y son archivos *XML*. Cada actividad tiene un *layout* asociado que define la vista de la actividad.

A continuación, se explicará el proceso de desarrollo de la aplicación, empezando desde la estructura general del proyecto, hasta la explicación detallada de los elementos que se han personalizado.

5.5.1 Estructura general del proyecto

Android Studio es la herramienta que la mayoría de los desarrolladores utilizan para producir los proyectos de aplicaciones *Android*, ya que es la herramienta oficial de la plataforma e incluye mayores facilidades para el desarrollo de apps. Además, está continuamente actualizada con las nuevas versiones de *Android* y las novedades en la funcionalidad y en el diseño que estas incluyen.

Conviene centrarse en los elementos de un proyecto de *Android Studio* más importantes relacionadas con el código *Android*. Estas carpetas son: *manifests* (contiene el archivo *AndroidManifest.xml)*, *java* (contiene las clases de java del proyecto) y *res* (contiene varias carpetas recursos, como *layouts*, valores en archivos *XML* o imágenes).

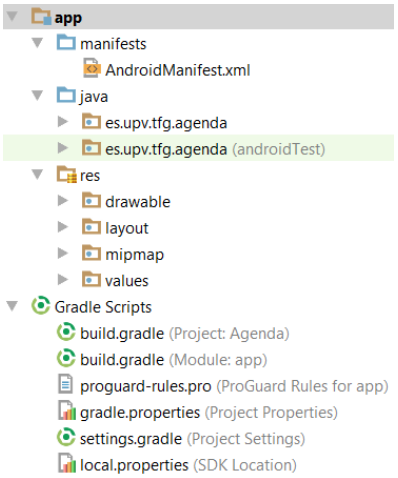


Figura 43. Estructura del proyecto en *Android Studio*

El archivo *AndroidManifest.xml* describe la aplicación *Android*. En él se indican las actividades, las intenciones y lo más importante: la versión mínima de *Android* para poder ejecutar la aplicación, que se ha decidido que sea el *SDK 16*, es decir, la versión 4.1 que abarca el 96.6% del total de dispositivos *Android*. Otro aspecto importante que se define en este archivo, son los permisos que dispondrá la aplicación, que se detallarán más adelante.

Una vez se ha visto la estructura general del proyecto, es conveniente mostrar con detalle las clases de *Java* y los *layout* que se han utilizado, aspectos que muestran las siguientes figuras:

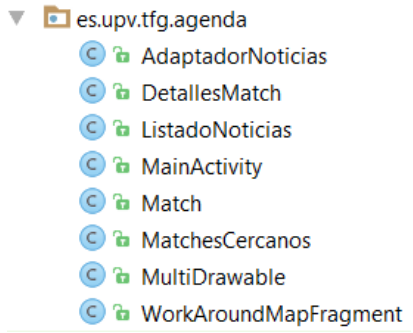


Figura 44. Clases de la aplicación

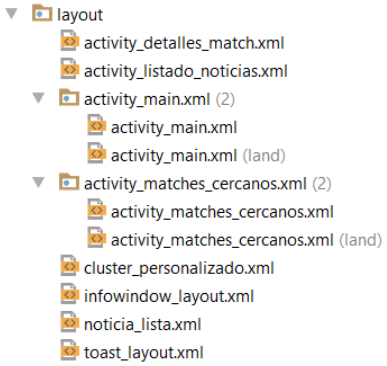


Figura 45. *Layouts* de la aplicación

Como se puede observar en la figura 45, para la actividad principal y para la actividad de *matches cercanos* se han utilizado dos *layouts*. Uno de estos *layout* es para la vista de la actividad cuando el dispositivo en el que se está ejecutando está en una posición vertical, mientras que el otro *layout* define la vista cuando el dispositivo está en posición horizontal. Esto se hace para que cuando el usuario gire el dispositivo, se siga mostrando una interfaz visualmente agradable y adaptada a la orientación del dispositivo que elija el usuario.

5.5.2 Permisos de la aplicación

Como se ha explicado en el apartado anterior, los permisos de los que dispondrá la aplicación se definen en el documento *AndroidManifest.xml*. Esta aplicación gozará de dos permisos esenciales para su funcionamiento:

- Permiso de acceso a Internet y el estado de la red (permisos *INTERNET* Y *ACCESS_NETWORK_STATE*).
- Permiso de acceso a la ubicación precisa del dispositivo (permiso *ACCESS_FINE_LOCATION*).

Hasta la versión de *Android* 6.0, se pedía a los usuarios que aceptaran conceder los permisos que pedía la aplicación en el momento de su instalación, de modo que si el usuario no aceptaba conceder todos los permisos requeridos la aplicación no podría utilizarse.

En *Android 6.0* los permisos se dividen en normales y peligrosos. El usuario debe de aprobar la concesión de los permisos normales en el momento de la instalación de la app, como se hacía en las versiones anteriores. La gran diferencia reside en la concesión de los permisos peligrosos, que no se conceden en el momento de la instalación, sino que la misma aplicación consultará al usuario si quiere conceder el permiso en el momento de utilizarlo. Además, en el menú de opciones del dispositivo *Android*, se pueden conceder o revocar cada uno de los permisos que pide la aplicación.

En el caso de la aplicación de este proyecto, en dispositivos *Android 6.0* se ha necesitado pedir el permiso peligroso de *ubicación precisa* al usuario para el correcto funcionamiento de la aplicación. La siguiente figura muestra la petición del permiso de ubicación precisa en *Android 6.0*:

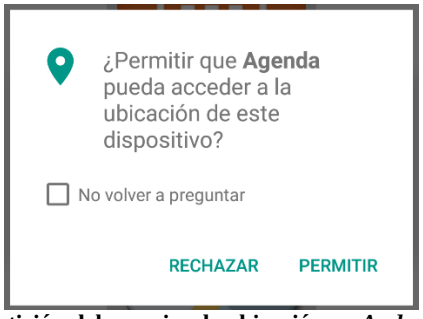


Figura 46. Petición del permiso de ubicación en *Android 6.0*

5.5.3 Ubicación

La funcionalidad más importante de la aplicación es mostrar los eventos geoposicionados que se encuentran cerca de la ubicación del usuario, por lo tanto, es necesario obtener la posición geográfica en la que se encuentra el dispositivo por medio de los servicios de ubicación del dispositivo.

La plataforma *Android* dispone de un sistema de posicionamiento que combina varias tecnologías:

- Sistema de localización basado en *GPS*, solo funciona cuando el dispositivo tiene visión directa de los satélites. Es el más preciso y es ideal para su uso en el exterior.
- Sistema de localización basado en la información proporcionada por los puntos de acceso *WI-FI* y por las torres de telefonía celular. Es menos preciso que el *GPS*, pero obtiene mejores resultados para espacios interiores, sin visión directa de los satélites.

Como la aplicación está pensada para su uso en todo tipo de sitios, se ha optado por combinar ambos métodos de obtención de la ubicación del dispositivo, dejando que sea el propio sistema *Android* el que elija en cada momento el mejor proveedor de información sobre la ubicación. Para este proceso, es necesario el permiso a la ubicación precisa del dispositivo (*ACCESS_FINE_LOCATION)*, ya que incluye el acceso todos los tipos de ubicación.

El sistema se encargará de actualizar la información de la ubicación cuando se detecte que el usuario ha cambiado su posición. Además, se ha creído conveniente la implementación de unos avisos para el usuario respecto a la ubicación, ya que sin unos datos concretos de ubicación es imposible que la aplicación funcione correctamente. Concretamente se han creado tres avisos que se muestran en la figura 47, que avisan al usuario cuando los servicios de ubicación del dispositivo están apagados, cuando no se tienen datos de la ubicación y cuando el dispositivo está obteniendo los datos de la ubicación.

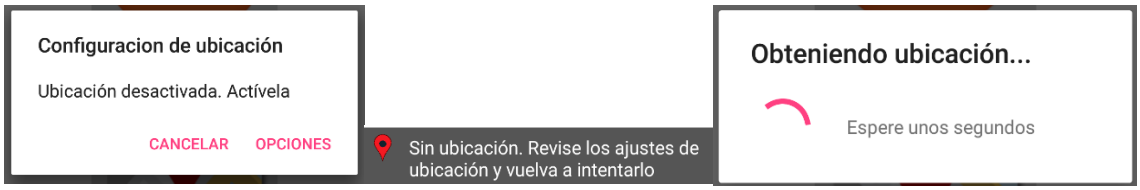


Figura 47. Avisos sobre la ubicación

5.5.4 Obtención de los eventos geoposicionados

Los datos principales en la aplicación desarrollada son los eventos geoposicionados obtenidos anteriormente y que están guardados en la base de datos, por lo tanto, es necesario implementar un módulo para hacer una petición al servicio web y que éste devuelva la información solicitada.

En *Android* es recomendable realizar las conexiones *HTTP* fuera del hilo principal de ejecución de la aplicación, ya que si el hilo principal de ejecución permanece bloqueado más de 5 segundos se muestra al usuario un mensaje de avisando de que la aplicación no responde.

La clase *AsyncTask* permite realizar un proceso que se ejecuta en un hilo secundario y cuyo resultado se quiere utilizar en el hilo de la interfaz del usuario. Por lo tanto, se van a implementar dos clases *AsyncTask*, una clase para cada método del servicio web. La siguiente figura muestra la clase *GetMatchesFromServer*, que extiende a una *AsyncTask* y tiene como objetivo hacer una petición *GET* al método *Matches Cercanos Resource* del servicio web, para que le devuelva en formato *JSON* los *Matches* dentro de un radio determinado respecto a la ubicación que ha proporcionado el dispositivo.

```
class GetMatchesFromServer extends AsyncTask<Double, Void, Match[]> {
    Match[] matches = null;
    ProgressDialog progressDialog;

    @Override
    protected void onPreExecute(){
        progressDialog = new ProgressDialog(MatchesCercanos.this);
        progressDialog.setTitle("Cargando noticias...");
        progressDialog.setMessage("Espere unos segundos");
        progressDialog.show();
    }

    @Override
    protected Match[] doInBackground(Double... parametros) {
        try {
            URL url = new URL("http://158.42.188.5:8080/agenda/api/matchescercanos/all?lat="+MainActivity.lat+"&lon="+MainActivity.lon+"&rad="+parametros[0]);
            HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
            InputStream in = new BufferedInputStream(urlConnection.getInputStream());
            matches = MainActivity.loadMatchesFromJsonFileServer(in);
            urlConnection.disconnect();
        } catch (Exception e) {
            e.printStackTrace();
            startActivity(new Intent(MatchesCercanos.this, MainActivity.class));
        }
        return matches;
    }

    @Override
    protected void onPostExecute(Match[] m) { progressDialog.dismiss(); }
}
```

Figura 48. Ejemplo de una *AsyncTask*

El servicio web enviará como respuesta los datos en formato *JSON*, por lo que también es necesaria la implementación de un módulo que sea capaz de leer y tratar la información que se recibe en formato *JSON*. Para esto, se ha adaptado el módulo lector de datos *JSON* de *RSSAgenda*, ya que el cometido final es el mismo.

Con la obtención de los datos del servidor y el correcto tratamiento de estos datos, la aplicación ya dispone de los eventos geoposicionados, por lo que el siguiente paso será mostrarlos en el mapa y en una lista con una interfaz atractiva para el usuario.

5.5.5 *Lista de noticias*

Una de las funcionalidades de la aplicación es mostrar una lista con todas las noticias y los sitios donde ocurren, para luego mostrar con más detalle la noticia que el usuario seleccione. Se ha visto como mejor solución la creación de una lista de noticias con un estilo personalizado para que sea más atractiva para el usuario, descartando las listas por defecto de *Android*.

La lista personalizada mostrará el titular de cada noticia, el nombre del sitio donde ocurre y el icono de los *feed RSS* de cada elemento de la lista.

Android dispone de una vista llamada *ListView*, que consiste en una lista deslizable verticalmente de varios elementos, donde cada elemento puede definirse como un *layout*. Para cargar las noticias en una *ListView* totalmente personalizada para la aplicación, es necesario la creación de una clase llamada *adaptador* totalmente personalizada. El *adaptador* se encargará de cargar los datos en la lista (titulares de las noticias y nombres de los sitios) y de definir el estilo personalizado en cada elemento de la lista.

La figura 49 muestra la lista de noticias personalizada:

Figura 49. *ListView* personalizada

5.5.6 *Google Maps, agrupación de marcadores y mapa de calor*

La funcionalidad más importante de la aplicación es la muestra de los eventos geoposicionados en el mapa dentro de un radio determinado. En consecuencia, es necesario mostrar un mapa al usuario con los eventos marcados en el sitio donde tienen lugar, con una interfaz que sea atractiva para el usuario y con una funcionalidad fácil y a la vez intuitiva.

La *API* de *Google Maps* proporciona un servicio de cartografía online que se puede usar en las aplicaciones *Android*, por lo que es conveniente el uso de la herramienta *Google Maps* para mostrar y trabajar con mapas en una aplicación *Android*. *Google Maps*, a diferencia de *Android*, no es un software de código libre, por lo que es necesario obtener una clave de *Google* e insertarla en el archivo *AndroidManifest.xml* para poder utilizar esta herramienta.

Los eventos geoposicionados se obtendrán de la respuesta del servidor web, después de tratarlos con el módulo lector de elementos *JSON*, por lo que el siguiente paso es colocarlos en el mapa con los datos de latitud y longitud del sitio donde ocurren estos eventos. En los mapas de *Google Maps* es posible colocar marcadores en un punto geográfico, por lo que los eventos se han colocado en el mapa mediante marcadores.

En un mismo sitio pueden ocurrir más de un evento, por lo que al colocar los eventos en el mapa mediante marcadores, en un sitio con más de un evento se solaparían muchos marcadores en un mismo punto geográfico. Para solucionar este problema se ha decidido agrupar estos marcadores mediante elementos llamados *clusters*, es decir, grupos de marcadores.

La librería *Google Maps utils* permite agrupar marcadores en *clusters* de manera automática, agrupando y desagrupando marcadores cuando el usuario varíe el nivel de zoom del mapa. Con la ayuda de esta librería, también se han personalizado totalmente los *clusters* y marcadores cambiando su aspecto, y lo más importante, cada *cluster* muestra el número de eventos que contiene, y estará remarcado de color rojo cuando el *cluster* solo contenga eventos de un mismo sitio.

Los marcadores personalizados serán el icono de los *feed RSS*, mientras que los *clusters* combinarán esta imagen dependiendo del número de marcadores que contengan. Esta personalización se puede ver en la figura 50:

Figura 50. Marcadores y *clusters* personalizados

Cada marcador tiene una ventana de información asociada que se muestra cuando el usuario lo selecciona. En este caso se ha decidido que esta ventana de información muestre el título del evento y el sitio donde ocurre, pudiendo el usuario acceder a los detalles del evento seleccionando esta ventana. También se ha personalizado totalmente esta ventana, como muestra la figura 51:

Figura 51. Marcador y ventana de información personalizadas

Para mostrar al usuario un indicador de las zonas donde tienen lugar muchos eventos, se ha creído conveniente la introducción de un mapa de calor en el mapa donde se muestran los eventos, ya que un mapa de calor ofrece una información visual muy potente. La librería *Google Maps utils* permite la creación de mapas de calor automáticamente, siendo la misma herramienta la que gestiona el mapa de calor cuando se cambia el nivel de zoom del mapa.

El mapa de calor que se ha implementado pintará las zonas que contienen más eventos de color rojo, mientras que las zonas con menos eventos estarán coloreadas de color verde claro. La figura 52 muestra un ejemplo de este mapa de calor:

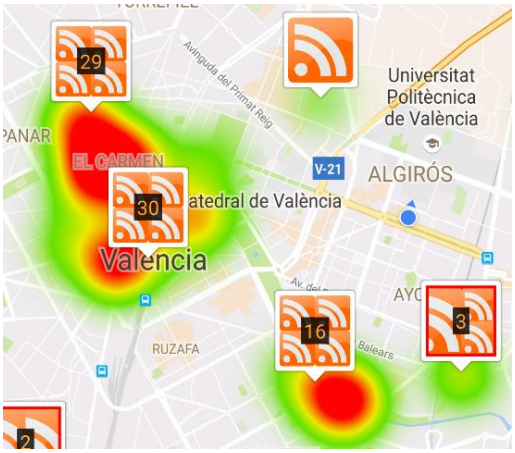


Figura 52. Ejemplo del mapa de calor implementado

5.5.7 Barra de selección del radio

El usuario debe de poder seleccionar el radio de búsqueda de los eventos a partir de su ubicación, por lo que es necesario implementar algún elemento para que el usuario seleccione este radio.

Para la selección del radio se ha decidido implementar una barra deslizante, con la que el usuario pueda seleccionar el radio de búsqueda dependiendo de la posición en la que se encuentre el seleccionador de la barra.

En *Android* este elemento es conocido como *Seekbar*, que permite personalizar las acciones que se deben producir en la aplicación cuando se detecta un cambio de posición del seleccionador de la barra. Se ha marcado un radio mínimo de 250 metros y un radio máximo que abarque todos los eventos de la ciudad de Valencia, con una diferencia de 500 metros entre cada posición de la barra, menos en las primeras posiciones que pasa de 250 metros a 500 metros de radio.

Cuando el usuario cambie la posición de la barra, se enviará una petición al servidor para obtener los eventos actualizados para el radio que ha determinado el usuario.

Figura 53. Barra deslizante de selección del radio en diferentes posiciones

Por último, en la figura 54 se muestra un ejemplo con el mapa completo de la aplicación.

Figura 54. Visión global del mapa de la aplicación

Capítulo 6. Guía de uso de la aplicación

El propósito de este capítulo es ofrecer una guía de uso de la aplicación *Agenda Cultural de Valencia* que se ha desarrollado en este trabajo de fin de grado. Se pretende ofrecer una vista de la funcionalidad completa de la aplicación, así como una guía sobre la navegación completa por las pantallas de la aplicación con esquemas y capturas de pantalla realizadas en un dispositivo *BQ Aquaris M5* que tiene instalado la versión *Android 6.0 Marshmallow.*

El esquema de navegación de la aplicación está compuesto por 6 vistas diferentes, cuya relación se puede ver en el esquema de la figura 55:

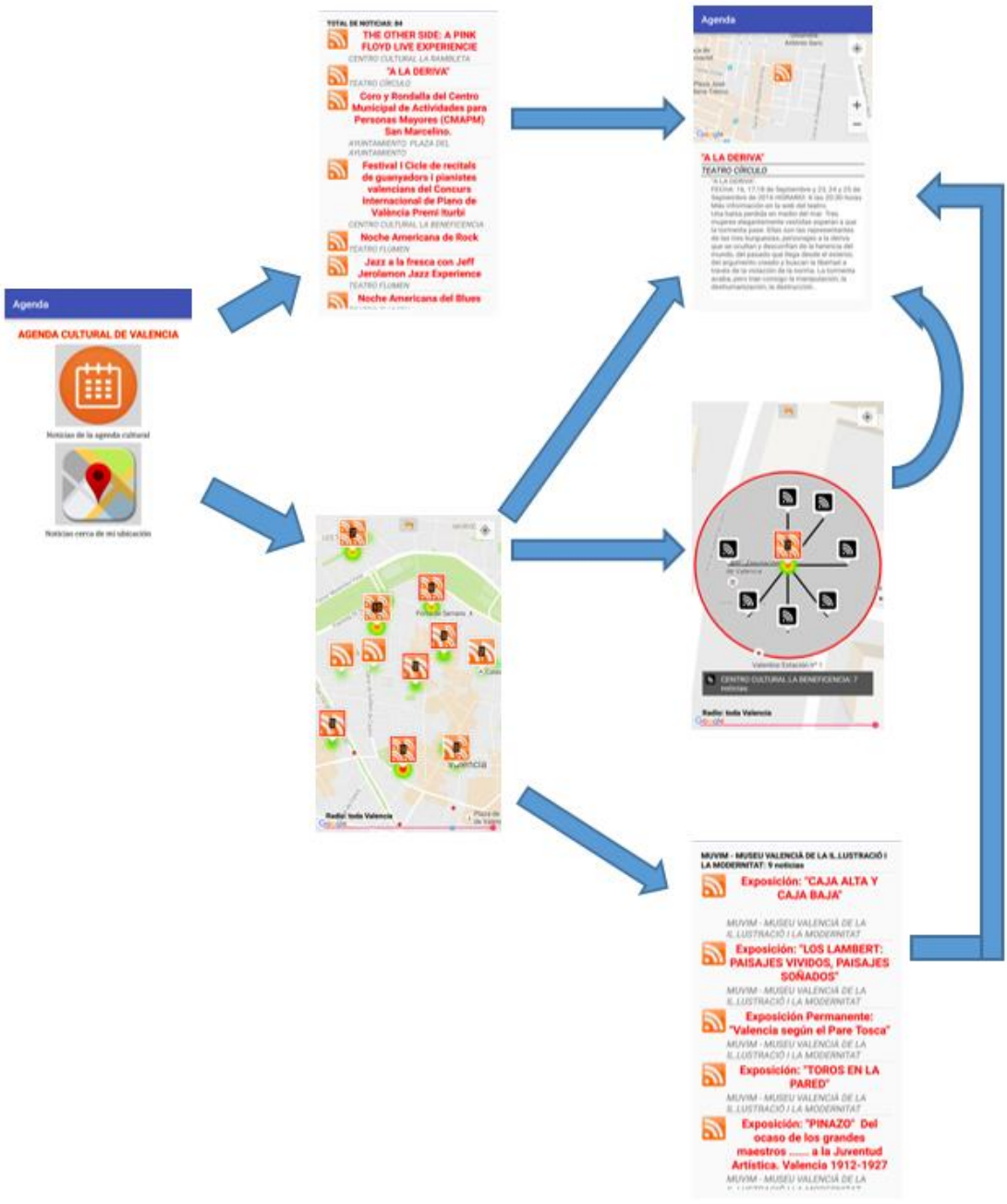


Figura 55. Esquema de navegación de la aplicación

En primer lugar, el usuario accederá al menú principal de la aplicación, donde se le pedirá que active los servicios de ubicación con un aviso si los tiene desactivados, mientras que si los servicios de ubicación del dispositivo están activados se mostrará un aviso mientras el dispositivo intenta obtener datos de su ubicación. Estos avisos se han mostrado en la figura 47.

Figura 56. Menú principal

El menú principal de la aplicación tendrá dos opciones:

1. **Noticias de la agenda cultural:** al seleccionar esta opción, se mostrará una lista con todos los eventos de la agenda cultural de la ciudad, sin ningún tipo de filtrado.

2. **Noticias cerca de mi ubicación:** esta opción permite acceder a un mapa en el que se mostrarán los eventos geoposicionados dentro de un radio determinado. Por defecto, se mostrarán todos los eventos de la ciudad.

La siguiente vista es la primera opción del menú principal, que es una lista con todas las noticias actuales de la agenda cultural, mostrándose el total de noticias que hay actualmente.

De cada noticia se muestra el titular en un formato de letra más grande y llamativo, y el sitio donde ocurre este evento. Al lado de cada noticia aparece siempre el mismo icono, el de los *feed RSS*, que son la fuente de donde se han extraído todas las noticias.

El usuario puede seleccionar la noticia en la que esté interesado para que se muestren los detalles.

Figura 57. Lista de eventos de la agenda cultural

Figura 58. Detalles de un evento

Esta vista muestra los detalles de un evento. Como se puede ver en la figura 58, se muestran el título, sitio y detalles del evento para proporcionar al usuario una mayor información sobre el citado evento.

Otro aspecto importante es la inclusión de un mapa con un marcador personalizado que indica el punto geográfico donde tendrá lugar el evento. El usuario puede interactuar libremente con el mapa.

Cuando el usuario selecciona la segunda opción del menú principal, la aplicación mostrará todos los eventos en el mapa, como muestra la figura 59. Estos eventos están agrupados en *clusters*, que van mostrando los eventos que contienen cuando el usuario amplía el mapa, y se van agrupando cuando el usuario disminuye el mapa.

El usuario puede seleccionar un marcador para ver más detalles de la noticia en la ventana de información del marcador. Si quiere ver los detalles de la noticia, puede pulsar esta ventana para acceder a los detalles de la misma en una vista como la de la figura 58.

Respecto a los *clusters*, los que contienen noticias del mismo sitio están remarcados de color rojo, y el usuario puede pulsarlos para que se muestren las noticias que contiene el *cluster*. Si el *cluster* contiene menos de 8 noticias, las noticias se mostrarán en el mapa como muestra la figura 60, con unos marcadores de un estilo diferente, pero con la misma funcionalidad. Por otra parte, si el *cluster* tiene más de 8 noticias, se dirigirá al usuario a una nueva vista, que mostrará las noticias en una lista personalizada, como muestra la figura 61.

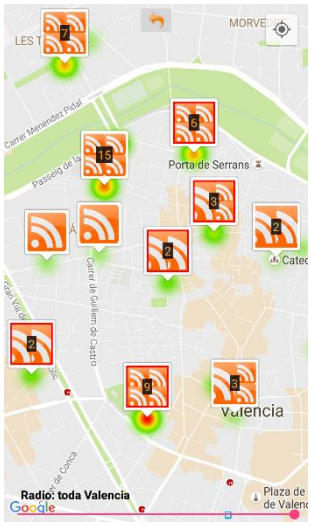


Figura 59. Mapa con los eventos

Figura 60. Eventos de un *cluster* con menos de 7 noticias

Figura 61. Eventos de un *cluster* con más de 8 noticias

Al seleccionar un *cluster* con noticias de diferentes sitios, se avisará al usuario con un mensaje para que amplíe el mapa para obtener más detalles, como muestra la figura 62.

Figura 62. Aviso de un *cluster* de noticias de diferentes sitios

El usuario puede seleccionar el radio de búsqueda de los eventos respecto a su ubicación actual con la barra deslizante que aparece en la vista principal del mapa (figura 59). Si las noticias tardan en recibirse del servidor, aparecerá un mensaje diciendo al usuario que espere a que se carguen las noticias.

Para finalizar con la vista principal del mapa, es conveniente explicar la funcionalidad de los botones que aparecen en el mapa. El botón con el icono de la flecha y el botón con el icono de la ubicación, tienen como finalidad centrar la vista del mapa en el punto geográfico de la ubicación del dispositivo, que estará determinado por un círculo azul. La diferencia entre estos botones, es que el primero fija un nivel de zoom determinado, mientras que pulsando el segundo botón se mantiene el nivel de zoom que había fijado el usuario.

Para finalizar esta guía de uso de la aplicación, conviene mencionar una característica que está activado por defecto en *Google Maps* y que se ha decido dejar activa por su gran utilidad. Al pulsar un marcador en alguno de los mapas de la aplicación, en la esquina inferior derecha de la pantalla aparecen dos botones y pulsando uno de estos dos botones, se abrirá la aplicación *Google Maps* del dispositivo.

Con la pulsación del primer botón, en el mapa de la aplicación *Google Maps* se muestra cómo ir desde la ubicación del dispositivo hasta el evento que se había seleccionado, como muestra la figura 63.

Si se pulsa el segundo botón, solo se muestra el evento con un marcador en el mapa de la aplicación *Google Maps*.

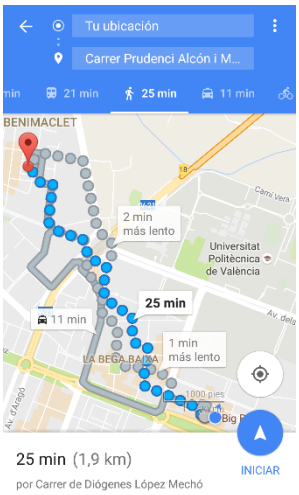


Figura 63. Cómo llegar a un evento

Capítulo 7. Conclusiones y líneas futuras

7.1 Conclusiones

En este trabajo final de grado se ha realizado un programa que consigue geoposicionar automáticamente noticias de eventos culturales gracias a los sitios geoposicionados del portal de datos abiertos del Ayuntamiento de Valencia. Estos datos han servido para el desarrollo de una aplicación en *Android* que muestra estas noticias en el mapa, indicando los eventos que ocurren alrededor del usuario dentro de un radio de búsqueda determinado.

La primera parte de este proyecto, que es el programa *RSSAgenda*, ha sido todo un reto, ya que es la primera vez que se presentaba el desafío de realizar un proyecto tan grande en *Java*, y, además, del que se esperaban unos resultados concretos.

Los primeros éxitos en la etapa de desarrollo de este primer programa han llegado con la inserción correcta de los primeros datos en la base de datos, especialmente de las primeras noticias con todos sus atributos correctamente insertados. Gracias a esto, se han adquirido unos conocimientos más amplios de las bases de datos ubicadas en servidores reales, y se ha abierto una visión del mantenimiento real de estos servidores, con sus problemas y sus correspondientes soluciones.

Estudiar la técnica del *procesado natural de lenguaje (NLP)* ha sido muy interesante, ya que en el mundo de la tecnología cada vez se está estudiando más la interacción de los humanos con las máquinas mediante el lenguaje natural del ser humano. En este proyecto solo se ha utilizado la técnica del *chunking*, más propia del tratamiento de textos y una parte pequeña del *NLP*, pero es una interesante primera toma de contacto con este campo de estudio.

Otra parte interesante desde el punto de vista de un estudiante ha sido el desarrollo y puesta en funcionamiento de un servicio web en un servidor real, ya que durante todo el grado se han estudiado los diferentes protocolos de internet y su uso, pero en la realización de proyectos de este tipo es donde se ve la aplicación real y la gran utilidad de estos protocolos, de los que a menudo los alumnos solo tienen una visión abstracta.

La otra gran parte y la parte final de este proyecto es el desarrollo de la aplicación en para dispositivos con el sistema operativo *Android*. La aplicación ha ido adquiriendo funcionalidades paso a paso, partiendo desde la simple muestra por pantalla de las noticias recibidas del servidor hasta la versión final de la aplicación. Este proceso ha servido para familiarizarse con el desarrollo de aplicaciones *Android* y comprender con más profundidad el funcionamiento de este sistema operativo.

La personalización de los elementos de la aplicación ha ocupado un tiempo razonable del tiempo de desarrollo total. Durante el desarrollo de la aplicación se han ido personalizando todos los elementos de la misma, partiendo desde elementos por defecto de la plataforma hasta elementos totalmente personalizados como pueden ser las *ListView* o los marcadores con sus ventanas de información del mapa. Esta etapa de personalización se ha hecho visualizando la aplicación desde el punto de vista del usuario final, ya que las aplicaciones gráficamente más atractivas suelen tener un mayor éxito que las aplicaciones más funcionales, pero no tan atractivas.

Respecto a los resultados obtenidos, cabe destacar el gran éxito que se ha obtenido en la continua ejecución de *RSSAgenda*, ya que es capaz de geoposicionar automáticamente la inmensa mayoría de los eventos con un sitio definido, alcanzando un éxito en el *match* en un 95.97% de los casos. Por esta razón, la aplicación final se ve como una herramienta fiable para consultar los eventos culturales de la ciudad de Valencia y como una demostración más de cómo las *Smart City* pueden facilitar la vida de los ciudadanos proporcionándoles información personalizada y en tiempo real.

A título personal, este proyecto me ha proporcionado una experiencia inmejorable dentro del mundo del desarrollo del software y del desarrollo de aplicaciones móviles, ya que además de la experiencia y conocimiento adquiridos, el proyecto me ha servido para obtener una visión global del trabajo en el mundo profesional en el ámbito de la tecnología. Además, la satisfacción de poder utilizar una aplicación que ha desarrollado uno mismo después de meses de duro trabajo es muy grata.

7.2 Líneas futuras

El desarrollo de una aplicación para smartphones abre un abanico infinito de propuestas y mejoras, ya que las mismas aplicaciones se van mejorando con futuras versiones y actualizaciones, ya que cuando los usuarios utilizan la aplicación proporcionan un *feedback* al desarrollador con la propuesta de posibles mejoras y la detección de algunos errores.

La primera y más lógica propuesta de mejora en el futuro es el desarrollo de la aplicación para más plataformas móviles aparte de *Android*. *Android* es la plataforma con más usuarios del mercado, pero si se quiere llegar a todo tipo de usuarios es conveniente el desarrollo de la aplicación para dispositivos *IOS* (*iPhones* y *iPads)*, ya que los dispositivos de *Apple* tienen un gran mercado y han conseguido cautivar a muchos usuarios. Otras plataformas para las que se podría desarrollar la aplicación sería *Windows Phone*, o el desarrollo para plataforma web directamente.

Otra mejora que conseguiría acercar la aplicación a muchos más usuarios sería la traducción de la aplicación a diferentes idiomas. La aplicación en si sería fácil de traducir a varios idiomas, ya que *Android* facilita la traducción de las aplicaciones mediante el uso de archivos *XML* con la traducción del texto a diferentes idiomas. Sin embargo, las noticias se tienen que geoposicionar en el programa *RSSAgenda*, por lo que habría que adaptar este programa para que guardara noticias, sitios y *matches* en los diferentes idiomas. Además, sería necesario que los *feeds* de los que se extraen las noticias y los sitios del portal de datos abiertos estuvieran traducidos a diferentes idiomas.

Centrándose en el programa *RSSAgenda* que geoposiciona eventos en función del reconocimiento de texto de los mismos, podría aplicarse la funcionalidad de este programa a otro tipo de eventos y noticias, no solo eventos culturales. Esto sería la semilla de una multitud de aplicaciones basadas en el geoposicionamiento de elementos, como podría ser una aplicación de monumentos turísticos de la ciudad de Valencia que recogiera noticias e información sobre ellos.

Glosario de términos

SATDR	Grupo de Sistemas y Aplicaciones de Tiempo Real Distribuidos
ISO	International Organizarion for Standardization
IEEE	Institute of Electrical and Electronics Engineers
ANSI	American National Standards Institute
VLCi	Valencia Ciudad inteligente
SQL	Structured Query Language
JDK	Java Development Kit
SDK	Software Development Kit
XML	eXtensible Markup Language
RSS	Really Simple Syndication
JSON	JavaScript Object Notation
NLP	Natural Language Processing
IOS	Iphone Operating System
HTTP	Hypertext Transfer Protocol
API	Aplication Programming Interface
GPS	Global Positioning System

Bibliografía

[1] http://wearesocial.com/uk/special-reports/digital-in-2016

[2] http://gobiernoabierto.valencia.es

[3] **Laboratorio Nacional de Calidad del Software del INTECO**: *Ingeniería del software: metodologías y ciclos de vida*. 2009

[4] https://es.wikipedia.org/wiki/Programaci%C3%B3n_extrema

[5] https://www.java.com/es/download/faq/whatis_java.xml

[6] https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

[7] https://es.wikipedia.org/wiki/Base_de_datos_relacional

[8] https://es.wikipedia.org/wiki/MySQL

[9] http://www.mysql.com/why-mysql/

[10] http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02149.pdf

[11] https://es.wikipedia.org/wiki/Extensible_Markup_Language

[12] https://es.wikipedia.org/wiki/RSS

[13] http://json.org/

[14] https://www.microsoft.com/en-us/research/group/natural-language-processing/

[15] https://es.wikipedia.org/wiki/Procesamiento_de_lenguajes_naturales

[16] **Breck Baldwin y Krishna Dayanidhi**: *Natural Languaje Processing with Java and LingPipe Cookbook*. https://books.google.es/books?id=5MSiBQAAQBAJ&printsec=frontcover&hl=es#v=onepage&q&f=false [Online]

[17] http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[18] https://es.wikipedia.org/wiki/Android

[19] https://www.android.com/intl/es_es/history

[20] https://developer.android.com/about/dashboards/index.html

[21] **Jesús Tomás Gironés**: *El Gran Libro de Android, 5ª edición*. Marcombo, 2016