UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática

en Ingeniería del Software

Trabajo Fin de Grado

OpenDataApps

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática

en Ingeniería del Software

Trabajo Fin de Grado

OpenDataApps

Autor: Francisco Javier Corchado Bernal

Tutor: Álvaro Prieto Ramos

Resumen

En la actualidad existe un movimiento digital llamado Open Data al que se están sumando cada vez más instituciones de todo el mundo. Dicho movimiento consiste en poner los datos que administran estas instituciones a libre disposición de la sociedad.

Esto ha provocado que aumente considerablemente el número de desarrolladores software que construyen sus aplicaciones utilizando datos abiertos.

Uno de los problemas que existe actualmente es que es difícil saber qué conjuntos concretos de datos abiertos están siendo utilizados por aplicaciones y qué aplicaciones trabajan con datos abiertos y con cuáles conjuntos de datos en particular.

El objetivo de este TFG es satisfacer estas necesidades construyendo un portal web donde los desarrolladores software de aplicaciones que utilizan datos abiertos puedan registrar, categorizar y publicitar sus aplicaciones indicando, entre otra información, los conjuntos de datos abiertos que utilizan.

Con esta información, por un lado, las instituciones que publican sus datos pueden saber qué conjuntos de datos están teniendo más éxito y puede priorizar un tipo de datos sobre otro, por otro lado, los desarrolladores pueden saber qué conjuntos de datos están siendo más utilizados por otras empresas y consiguen una nueva vía para dar a conocer sus aplicaciones y, por último, a los usuarios se les ofrece una nueva forma de saber qué datos abiertos pueden ser más interesantes y también les facilita la búsqueda de aplicaciones que usan datos abiertos.

Contenido

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. Introducción

En una era donde la información es de vital importancia, han aparecido nuevas tecnologías y nuevos movimientos para facilitar el procesamiento y la transmisión de la misma.

Uno de los movimientos que ha aparecido es Open Data que es una filosofía en la que su objetivo es poner a disposición de la sociedad los datos que gestionan las administraciones públicas en formatos fáciles de manipular.

Gracias a esta nueva filosofía la sociedad puede aprovechar estos datos para generar valor económico. Así, se podrán construir sobre ellos nuevas ideas que generen nuevos datos, conocimientos y servicios que reporten beneficios económicos y sociales. A las empresas que consumen estos datos se les denomina infomediarios o reutilizadores.

Este nuevo movimiento ha producido un aumento en el número de desarrolladores de software que utilizan estos datos públicos en la construcción de sus proyectos.

Con este TFG se quiere seguir promoviendo este movimiento con la construcción de un portal web en el que los desarrolladores de aplicaciones que usan datos abiertos puedan publicarlas e indicar y concretar qué datos abiertos están utilizando.

1.1. Principios Open Data

Los principios [1] para abrir la información gubernamental sirven para evaluar hasta qué punto los datos públicos son abiertos y están disponibles a la sociedad. Los principios son los siguientes:

- Detallados

Los conjuntos de datos publicados por las instituciones y administraciones deben ser lo más completa posible, lo que refleja la totalidad de los que se registra sobre un tema en particular. Toda la información en bruto a partir de un conjunto de datos debe ser liberado al público, salvo en la medida necesaria para cumplir con la ley con respecto a la divulgación de información personal. Los metadatos que define y explica los datos que deben ser incluidos también.

- Primacía

Los conjuntos de datos publicados por el gobierno deben ser datos de fuente primaria. Esto incluye la información original recogido por el gobierno, los detalles sobre cómo se recogieron los datos y los documentos originales que registran la recolección de los datos. La difusión pública permitirá a los usuarios verificar que la información se recogió correctamente y registra con precisión.

- Actualizados

Los conjuntos de datos publicados por el gobierno deben estar a disposición del público en el momento oportuno. Siempre que sea posible, la información recogida por el gobierno debe ser puesto en libertad tan pronto como se recolecta y se recoge. Se debe dar prioridad a los datos cuya utilidad es sensible al tiempo.

- Automatizados

Los datos deben estar estructurados para que puedan ser procesadas de forma automática por un ordenador. Esta es una condición muy importante para que se puedan reutilizar los datos de una forma automática.

- Accesibles

Hay que hacer accesibles los datos al mayor número de usuarios posible. No debería existir ninguna restricción para todos aquellos que quieran hacer uso de los datos, ni en el propósito de uso.

- No discriminación

"No discriminación" se refiere a quién puede acceder a los datos y cómo deben hacerlo. Barreras para la utilización de los datos puede incluir requisitos de registro o de adhesión.

- Abiertos

Los formatos de los datos deben ser no propietarios; es decir, no pueden depender de una entidad o de una herramienta propietaria de una entidad. Como ejemplo, un formato abierto sería CSV o XML, mientras que formatos propietarios serían Word, Excel, etc.

- Libres

Los datos deben ser de uso 100% libre para los usuarios. Así, los datos deben estar libres de derechos, patentes, copyright y no estar sujetos a derechos de privacidad, seguridad o privilegios que puedan estar regladas por otras normas.

1.2. ¿Qué es un conjunto de datos o dataset?

Es una colección de datos relacionados susceptibles de ser reutilizados (bien por sí solos o bien mezclándolos con otros datos) para crear servicios de interés para los ciudadanos, empresas u organizaciones [2].

1.3. ¿Por qué Web?

Se denomina aplicación web a aquellas aplicaciones de software que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador [3].

Las aplicaciones web son muy populares debido a lo práctico del navegador web como cliente ligero y a la independencia del sistema operativo. Algunas ventajas de este tipo de aplicaciones son las siguientes:

- Ahorran costes de hardware y software

Sólo es necesario usar un ordenador con un navegador web y conectarse a Internet. Las aplicaciones basadas en web usan menos recursos que los programas instalados. Por otra parte las aplicaciones web no requieren canales de distribución como el software tradicional, lo que permite que su precio sea inferior al de los programas instalables.

- Fáciles de usar

Las aplicaciones web son muy sencillas de utilizar, sólo necesitará conocimientos básicos de informática para trabajar con ellas.

- Escalables y de rápida actualización

Existe solo una versión de la aplicación web en el servidor, por lo que no hay que distribuirla entre los demás ordenadores. El proceso de actualización es rápido y limpio. Las aplicaciones basadas en web no requieren que el usuario se preocupe por obtener la última versión ni interfieren en su trabajo diario para descargar, instalar y configurar últimas versiones.

- Provocan menos errores y problemas

Las aplicaciones web son menos propensas a colgarse y crear problemas técnicos debido a conflictos con hardware, con otras aplicaciones existentes, protocolos o con software personal interno.

1.4. Motivación

Las motivaciones que han promovido el desarrollo de este proyecto son las siguientes:

- El primer motivo que me ha influido para elegir este proyecto como TFG para completar mis estudios es que me atrae mucho el desarrollo web, independientemente de las tecnologías utilizadas. Además el desarrollo web tiene una gran salida hacia el mundo laboral.
- Este proyecto promueve la filosofía de Open Data que hoy en día está cobrando cada vez más importancia en nuestra sociedad como demuestra un estudio presentado por Capgemini Consulting [4], en el que se revela que cada vez más organismos públicos europeos están implementando una política de datos abiertos para que los ciudadanos los puedan utilizar y compartir libremente. Este tema es de gran ayuda para seguir completando mi formación.
- En un contexto de crisis económica como el actual, es importante poder priorizar la apertura de datos que favorezcan el crecimiento económico, de manera que las empresas puedan ser más competitivas en sentido genérico y esto repercuta a largo plazo en el crecimiento. Uno de los objetivos del desarrollo de este proyecto es precisamente intentar facilitar la labor de las administraciones públicas a la hora de priorizar los datos que van a hacer públicos [5].

- Hoy en día los usuarios solicitan muchas aplicaciones que ofrecen funcionalidades muy útiles para su vida cotidiana. Esto es posible gracias al uso que realizan las aplicaciones de datos publicados por administraciones públicas. Este portal es una herramienta de búsqueda de este tipo de aplicaciones para los usuarios.

1.5. Alcance

En este apartado se va a exponer hasta donde se quiere llegar en el desarrollo de este trabajo fin de grado.

- **Definir un sistema de información** para almacenar la información que se va a utilizar en el portal web. Toda lo relacionado con los usuarios de la aplicación y las aplicaciones y los datasets que publicarán los mismos.
- **Desarrollar el backend** del portal web para que se pueda atender todas las peticiones de los clientes o usuarios desde sus navegadores web.
- **Desarrollar el frontend** para que los usuarios puedan tener una buena interacción y una gran experiencia de usuario con el portal web.

1.6. Objetivos

El objetivo general de este proyecto es crear un portal web que sirva a los publicadores de conjuntos de datos abiertos para poder obtener información relevante en el momento de tomar la decisión de qué conjuntos de datos publicar y a los desarrolladores para, por un lado, poder decidir bien sobre qué aplicaciones desarrollar y, por otro lado, publicitar dichas aplicaciones en el portal una vez las tengan desarrolladas.

En cuanto al desarrollo e investigación los objetivos son:

- Hacer un estudio de los portales y repositorios de aplicaciones más conocidos para:
    - Analizar su arquitectura.
    - Estudiar qué características y detalles ofrecen sobre las aplicaciones que contienen.

  - Identificar qué propiedades serían deseables en estos portales y si las aplicaciones que van a contener trabajan con datos abiertos.

- Diseñar el portal web que mantenga información tanto de las aplicaciones que utilizan datos abiertos como los detalles de los datos abiertos utilizados. Para ello:
  - Se tomará como base el estudio realizado como primer objetivo.
  - Se analizarán las distintas tecnologías con las que poder construir la solución y se usarán las que mejor se acomoden a los requisitos del nuevo portal para su desarrollo.

2. Planificación del proyecto

La planificación realizada para el proyecto se ha dividido en las siguientes fases:

1. Fase 1: estudio de características de la información que ofrecen los conjuntos de datos abiertos para decidir qué información es deseable que las aplicaciones que los usan indiquen cuando hacen referencia a ellos. Este estudio se muestra en el capítulo 3.

2. Fase 2: estudio de repositorios web de aplicaciones más conocidos, teniendo en cuenta también aquellos que utilizan datos abiertos. En este estudio se realiza una comparativa entre los distintos portales analizados y se comprueba si ofrecen algunas de las características obtenidas en la fase 1 y, en caso afirmativo, cómo las ofrecen. Este estudio se detalla en el capítulo 3.

3. Fase 3: partiendo de los resultados de la fase 1 y la fase 2, se realiza un análisis de los casos de uso y se definen los requisitos de la aplicación de manera formal. En esta fase también se hace el diseño detallado de la solución propuesta. Este análisis se detalla en el capítulo 4.

4. Fase 4: definición de la arquitectura de la aplicación web. Al optarse por realizar una aplicación SPA (Single Page Application, más detalles en sección 4.4.2), la arquitectura está directamente relacionada con este tipo de aplicación. Así, en esta fase se investigan las arquitecturas de este tipo de aplicación web y se decide qué tecnologías utilizar para implementar el portal. Además, con los requisitos y los casos de uso ya definidos en la fase 3, se realiza también el diseño de la base de datos del sistema. Los resultados de esta fase se detallan en el capítulo 4.

5. Fase 5: con los resultados obtenidos y las decisiones tomadas en la fase 4 se construye la aplicación. En esta fase, como en cualquier desarrollo de un proyecto, han aparecido problemas cuyas soluciones tomadas se detallan en el capítulo 5.

6. Fase 6: documentación del proyecto. Esta fase ha sido transversal al resto de fases y se ha extendido durante todo el ciclo de vida del proyecto.

En la tabla 1 se muestra la planificación de horas esperadas para cada fase a principio del proyecto y las horas reales que se han dedicado.

Tabla 1 Relación de horas en cada fase del proyecto

Fase	**Horas Esperadas**	**Horas Reales**
Fase 1	10	15
Fase 2	40	32
Fase 3	20	25
Fase 4	35	28
Fase 5	150	185
Fase 6	25	35

3. Estado Del Arte

En este capítulo se muestran las características distintivas que deben ofrecer el portal web a desarrollar y el análisis y el estudio realizado sobre distintos repositorios y portales web de aplicaciones. Para la selección de los portales analizados se han seguido dos criterios. Por un lado, se han incluido aquellos portales que son referencia en alguna de las plataformas más habituales (Android, IOS, Windows, Mac, Linux, Web). Por otro lado, se han incluido portales que tienen alguna o varias de las características distintivas del portal a desarrollar.

3.1. Características

Las características que van a distinguir este portal web son las siguientes:

- Las aplicaciones que se dan de alta pueden estar disponibles en cualquiera de las plataformas más habituales y pueden proceder de cualquier país del mundo. Para cada aplicación:
    - Debe almacenar y mostrar el identificador de recursos uniforme URI de los datasets que utiliza.
    - Debe almacenar y mostrar una descripción de los datasets que utiliza.
- Debe ofrecer estadísticas del uso de los datasets.

3.2. Portales Web

En este apartado se describirá cada portal estudiado y se expondrán sus puntos fuertes y débiles con respecto al proyecto que se va a desarrollar en este TFG.

3.2.1. Google Play Store

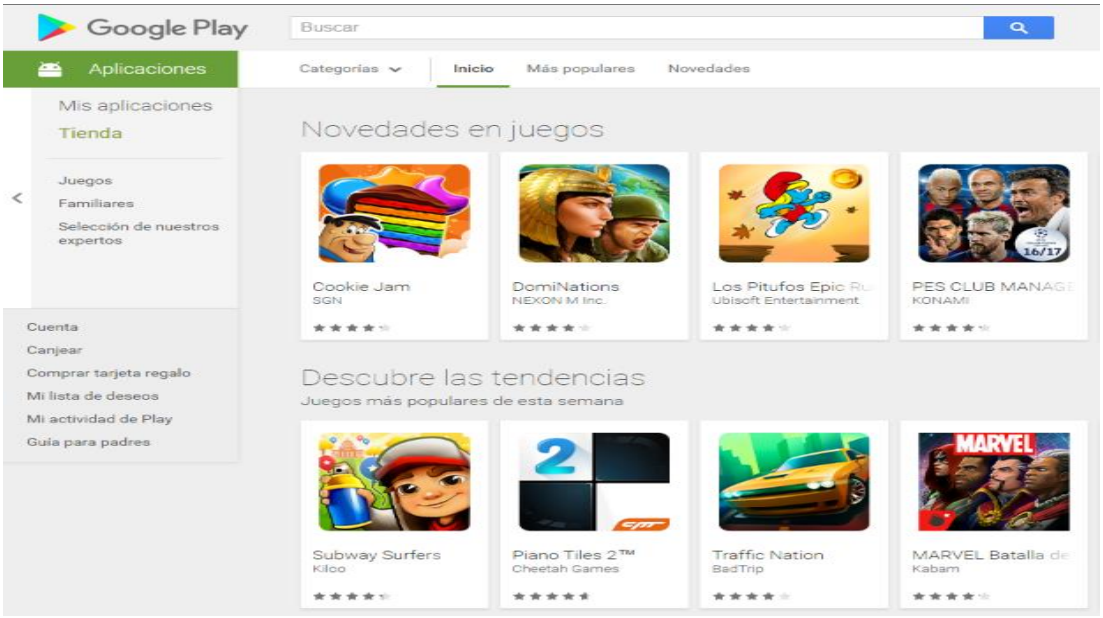


Figura 1 Portal web Google Play Store

Es una plataforma de distribución de aplicaciones que se ejecutan en sistemas operativos Android. Esta tienda online fue desarrollada por google y se ha elegido para el estudio porque es uno de los repositorios más utilizados en todo el mundo. Publicita aplicaciones de todo tipo, incluidas entre ellas las que utilizan datasets.

Puntos fuertes:

- Es uno de los repositorios más utilizados.
- Engloba aplicaciones de todo el mundo, por tanto se utilizarán datos abiertos de todo el mundo.
- Muestra mucha información sobre los desarrolladores y las aplicaciones.
- Interfaz muy amigable.
- El sistema donde se ejecutan las aplicaciones es el sistema operativo más utilizado para móvil del mundo. En la Figura 2 se puede ver el uso de los sistemas operativos móviles durante el año 2015.

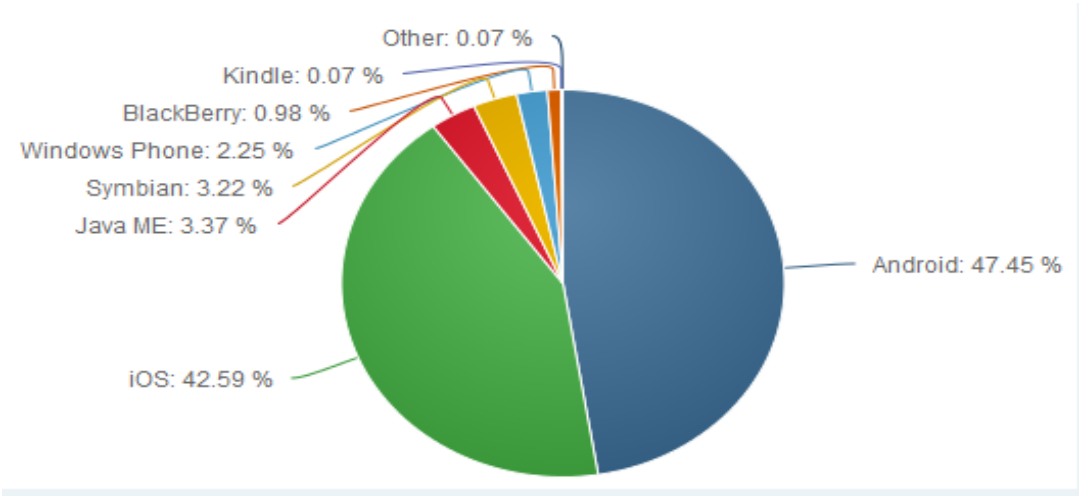


Figura 2 Gráfico del uso de los SO móviles durante el año 2015

Puntos débiles:

- Publicita todo tipo de aplicaciones y no hace ningún tipo de distinción entre las que usan datos abiertos y las que no.
- Entre la información que muestra sobre la aplicación no hace referencia a si utiliza datos abiertos y en caso de utilizarlos no da información sobre ellos.
- Tiene una categorización de aplicaciones demasiado amplia, aproximadamente 50 categorías diferentes.
- Se debe pagar cuota de desarrollador para poder publicar aplicaciones.

Este repositorio no cuenta con alguna de las características más importantes que se quieren cubrir en el proyecto:

- No suministra información sobre los datos abiertos utilizados por las aplicaciones y no muestra por tanto ninguna información relevante sobre los mismos.

3.2.2. Tienda De Windows

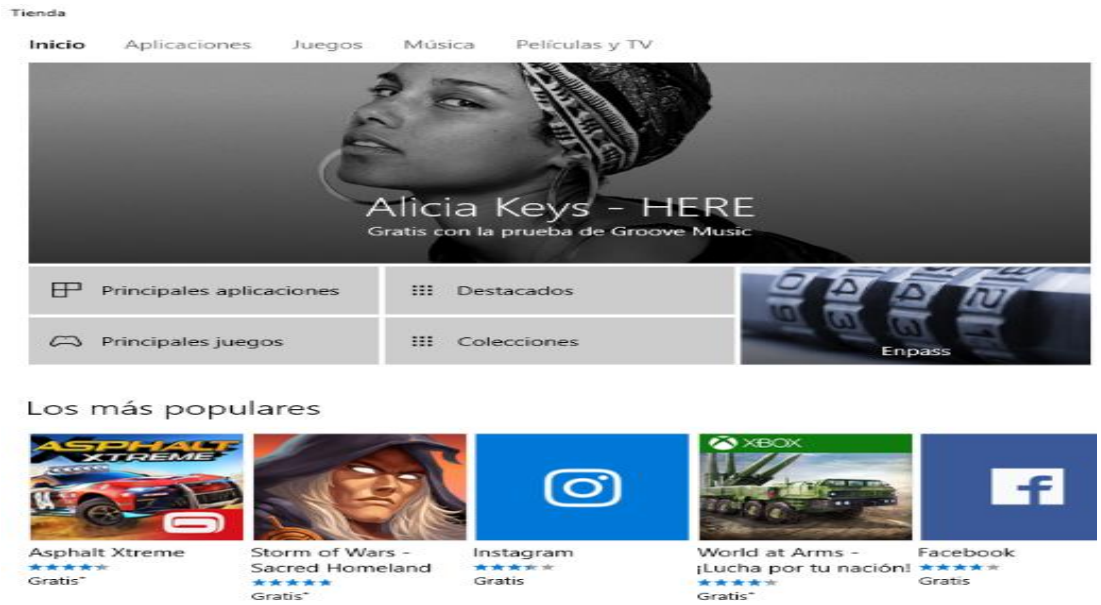


Figura 3 Tienda de Windows

Es una plataforma digital para la distribución de aplicaciones que se ejecutan en el sistema operativo Windows. Fue desarrollada por Microsoft y al igual que el repositorio anterior publicita aplicaciones de todo el mundo y de todo tipo entre las que se encuentran aplicaciones que utilizan datos abiertos.

Puntos fuertes:

- Tiene una interfaz de usuario muy amigable e intuitiva.
- Muestra bastante información sobre los desarrolladores y las aplicaciones.
- Distribuye aplicaciones de todo el mundo, por tanto puede ser que se utilicen datos abiertos de todo el mundo.
- El sistema operativo en el que se ejecutan sus aplicaciones es el más utilizado del mundo como podemos observar en el siguiente gráfico que muestra la evolución del uso de los sistemas operativos en el periodo comprendido entre los meses julio de 2015 y julio de 2016.

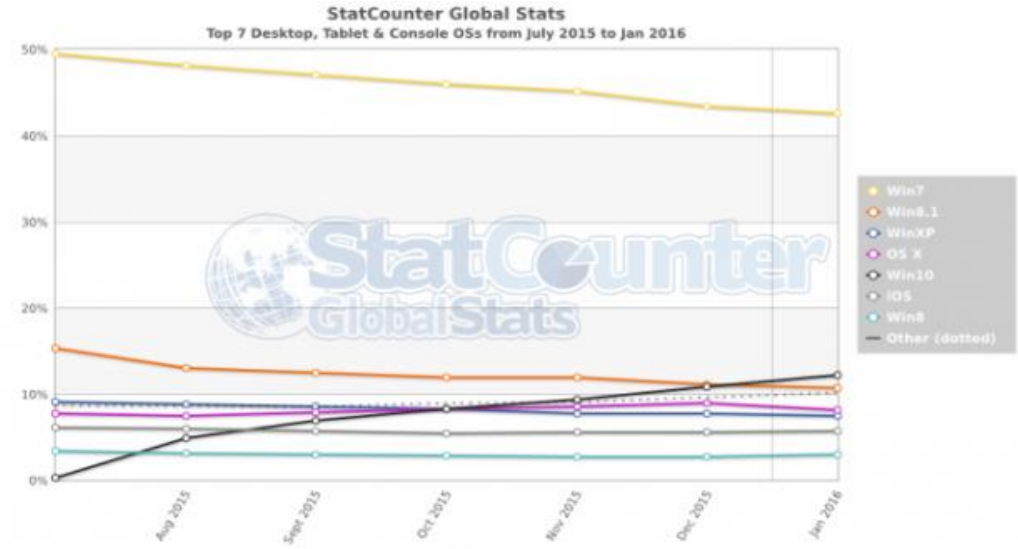


Figura 4 Gráfica del uso de los sistemas operativos julio 2015-julio 2016

Puntos débiles:

- No indica qué aplicaciones utilizan datos abiertos.
- En las aplicaciones que usan datos abiertos no indica qué datos abiertos utiliza ni muestra datos importantes sobre los mismos.
- Tiene una categorización muy amplia de las aplicaciones que distribuye.
- Se debe pagar cuota de desarrollador para poder distribuir las aplicaciones en dicha plataforma.

Del análisis de esta plataforma se puede deducir que:

- No es una buena referencia para el proyecto porque no le da ninguna relevancia al uso de datos abiertos utilizados en las aplicaciones.
- Esta plataforma no satisface las características más importantes que se quieren cubrir con el desarrollo de este proyecto.

3.2.3. App Store

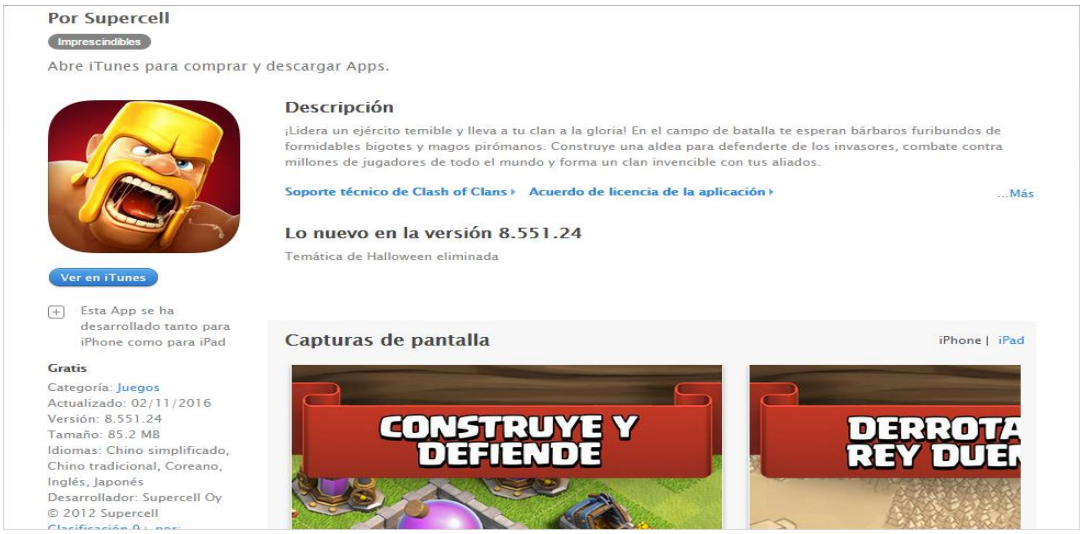


Figura 5 Portal web App Store

Es el Marketplace de aplicaciones para usuarios de Apple, a través del cual miles de desarrolladores de apps del mundo entero ofrecen sus productos y millones de usuarios pueden descargar aplicaciones gratuitas o de pago.

Puntos fuertes:

- Publica aplicaciones de todo el mundo lo que significa que se utilizan datos abiertos de todo el mundo.
- El sistema operativo en el que se ejecutan las aplicaciones es uno de los más utilizados y populares del mundo.
- Muestra bastante información sobre las aplicaciones y los desarrolladores de las mismas.

Puntos débiles:

- Tiene una interfaz gráfica bastante simple.
- Tiene una categorización de aplicaciones demasiado amplio.
- No indica de ninguna manera que aplicaciones utilizan datos abiertos.

- No muestra información sobre los datos abiertos que utilizan las aplicaciones que publicita el portal.

Este portal no reúne las características con las que contará este proyecto:

- No indica información sobre los datos abiertos que utilizan las aplicaciones
- No hace ningún tipo de distinción entre las aplicaciones que usan datos abiertos y las que no los utilizan.

3.2.4. Chrome Web Store

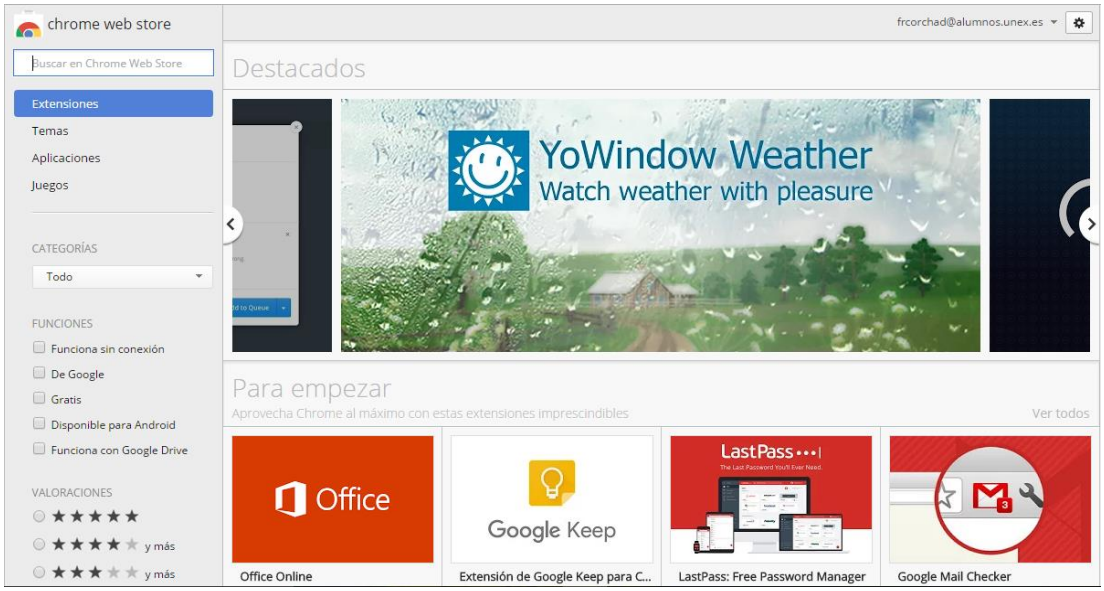


Figura 6 Portal Chrome Web Store

Es una tienda de aplicaciones web para el navegador de Google Chrome, y la cual fue desarrollada y mantenida por Google. A parte de aplicaciones también cuenta con extensiones y temas visuales para el navegador.

Puntos fuertes

- Es una web que publicita aplicaciones de todo el mundo.
- La web publicita aplicaciones que se ejecutan en uno de los navegadores más utilizados del mundo como demuestra el siguiente gráfico.

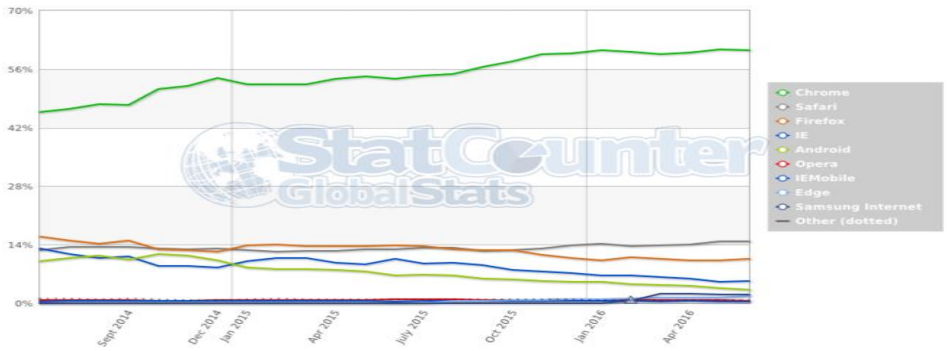


Figura 7 Gráfico sobre los navegadores más utilizados

- Publicita aplicaciones de todo el mundo, por tanto se puede suponer que se usan datos abiertos de todo el mundo.
- Tiene una interfaz bastante intuitiva y amigable.
- Proporciona bastante información y datos sobre las aplicaciones y los desarrolladores.

Puntos débiles

- No indica que aplicaciones utilizan datos abiertos y cuáles no.
- No proporciona datos sobre los datos abiertos utilizados por las aplicaciones que publicita.
- Tiene una categorización demasiado amplia.

Este portal no reúne las condiciones con las que se quieren que cuente el portal web de este proyecto:

- No indica ningún tipo de información sobre los datos abiertos que utilizan las aplicaciones

3.2.5. Data.gov

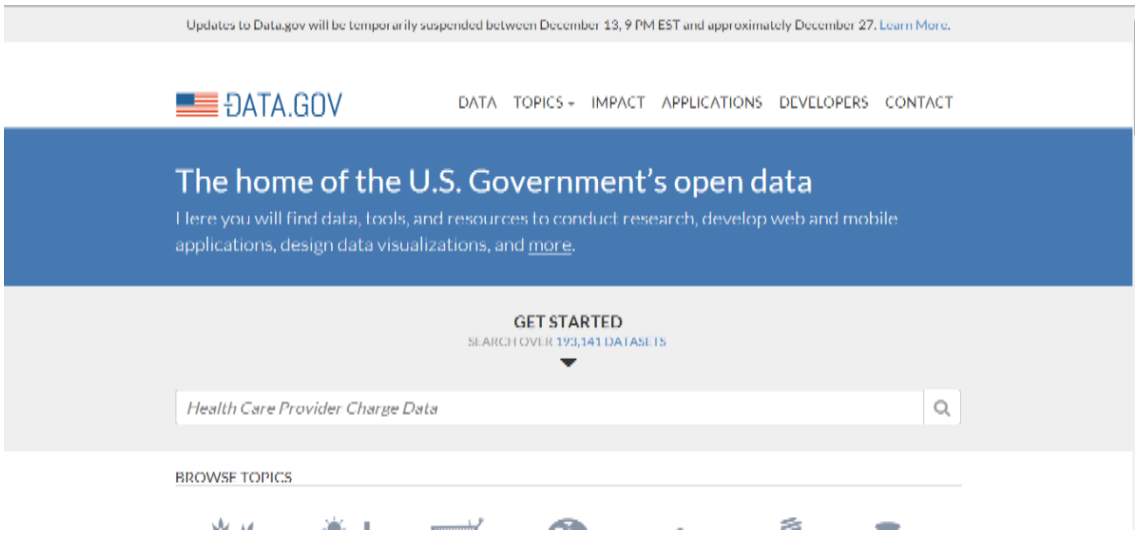


Figura 8 Portal Web Data.gov

Es un sitio web puesto en marcha por el gobierno de los EE.UU. en el año 2009. Su objetivo es facilitar y mejorar el acceso público a los conjuntos de datos generados por el poder ejecutivo del gobierno federal. Tiene una sección en la que publicitan aplicaciones que utilizan los conjuntos de datos abiertos que ponen a disposición del público y otra sección para los desarrolladores de las estas aplicaciones.

Puntos fuertes

- Solo publicitan aplicaciones que usan datos abiertos.
- Indican el conjunto de datos que utilizan las aplicaciones.

Puntos débiles

- No proporcionan ninguna descripción sobre los conjuntos de datos abiertos que utilizan las aplicaciones.
- No muestran estadísticas sobre los datos abiertos utilizados.
- No muestran mucha información sobre las aplicaciones y sobre los desarrolladores.
- La interfaz no es muy amigable.

- Los datos abiertos y las aplicaciones se reducen a los EE.UU.

Este portal se acerca un poco más a los que se quiere desarrollar en este proyecto, pero:

- Su ámbito se reduce a los EE.UU
- No proporciona toda la información relacionada con los conjuntos de datos abiertos que se quiere proporcionar en el portal.

3.2.6. Data.gov.uk

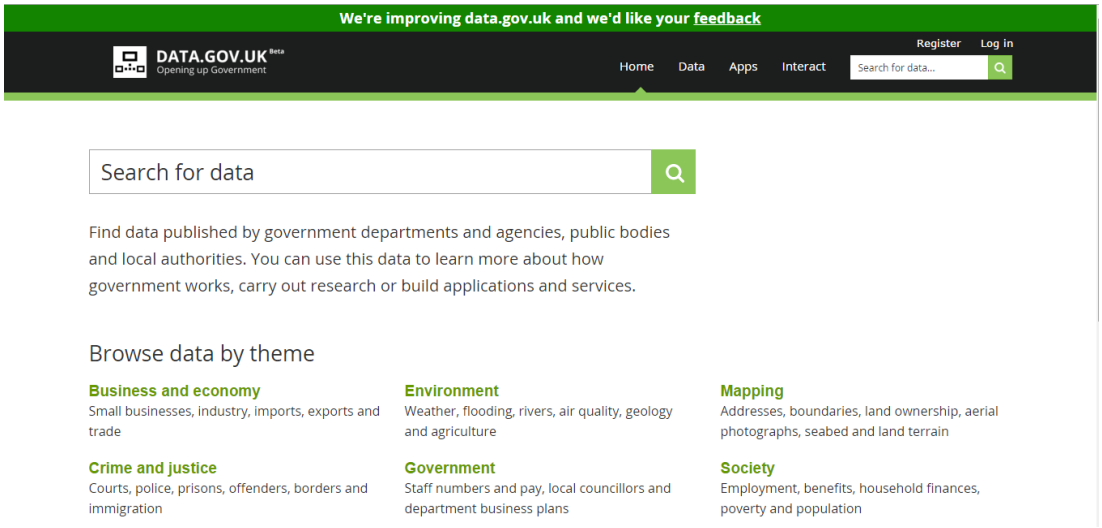


Figura 9 Portal Web Data.gov.uk

Es un proyecto del gobierno del Reino Unido para que los datos no personales del gobierno estén disponibles como datos abiertos. Se lanzó públicamente en enero del 2010. Tiene muchas secciones, entre ellas la sección de conjuntos de datos, aplicaciones, foros, librerías, etc.

Puntos fuertes

- Tiene una interfaz muy amigable.
- Solo publicita aplicaciones que utilizan datos abiertos.
- Indica los conjuntos de datos abiertos que utilizan las aplicaciones.

- Muestra bastante información respecto a las aplicaciones y a los desarrolladores de las mismas.
- Muestra estadísticas sobre los conjuntos de datos abiertos utilizados por los desarrolladores por sus aplicaciones.

Puntos débiles

- Las aplicaciones y los conjuntos de datos abiertos utilizados por estas se reducen al Reino Unido.
- No proporcionan una descripción directa sobre los conjuntos de datos abiertos utilizados por cada aplicación.

Este portal web es el que más se acerca a los requisitos de este proyecto:

- Las aplicaciones que publicitan usan datos abiertos, proporcionan bastante información tanto de las aplicaciones como de los conjuntos de datos que utilizan.
- Como desventaja hay que indicar que su ámbito de actuación se reduce al Reino Unido.

3.2.7. Datos.gob.es

Figura 10 Portal Web datos.gob.es

Es el portal web puesto en marcha por el gobierno de España para poder organizar y gestionar el Catálogo de Información del Sector Público. Como las anteriores tiene secciones de las aplicaciones y catálogo de datos.

Puntos fuertes

- Solo publicita aplicaciones que usan conjuntos de datos abiertos.
- Indican que conjuntos de datos utilizan las aplicaciones.
- Tiene una interfaz muy amigable para el usuario y renovada recientemente.
- Muestra bastante información sobre las aplicaciones.

Puntos débiles

- No muestra mucha información sobre los desarrolladores de las aplicaciones.
- Los conjuntos de datos y las aplicaciones que los utilizan se limitan a España.
- No muestran estadísticas sobre los datos abiertos utilizados.
- No proporcionan descripción sobre los datos abiertos utilizados directamente en la información de la aplicación.

Este portal no cumple con todas las características con las que se quieren contar en este proyecto:

- Los conjuntos de datos utilizados por las aplicaciones son solamente de España
- No muestra estadísticas sobre los conjuntos de datos y las aplicaciones.

3.3. Conclusiones

Después de realizar un estudio por todos estos portales web donde se publicitan aplicaciones de todo tipo y de todos los lugares del mundo, se ha llegado a la conclusión de que no hay ningún portal web en la actualidad que cumpla con todas la características con las que se quiere contar en el portal web que se va a desarrollar en este proyecto.

En la Tabla 2 se muestra un resumen con las características más importantes que aparecen en el apartado 3.1 de este capítulo y qué portales las cumples y cuáles no.

Como se puede observar no hay ningún portal que cuente con todas las características.

Tabla 2: Tabla de comparación de portales web

Información/ Repositorios	**Google Play Store**	**Tienda De Windows**	**App store**	**Chrome web store**	**data.gov**	**data.gov.uk**	**datos.gob**
Las aplicaciones proceden de cualquier país	**SI**	**SI**	**SI**	**SI**	**NO**	**NO**	**NO**
Almacenan y muestran el identificador URI de los datasets	**NO**	**NO**	**NO**	**NO**	**SI**	**SI**	**NO**
Almacenan y muestran una descripción de los datasets	**NO**	**NO**	**NO**	**NO**	**NO**	**NO**	**NO**
Ofrecen estadísticas del uso de los datasets	**NO**	**NO**	**NO**	**NO**	**NO**	**SI**	**NO**

4. Análisis y diseño

En este capítulo se muestran en primer lugar los requisitos de la aplicación, seguidos de los casos de uso antes de pasar a mostrar las primeras decisiones en cuanto a la solución propuesta. A continuación se presenta la arquitectura del sistema y el diseño de la base de datos para terminar explicando las razones para haber usado un API REST y los patrones de diseño utilizados.

4.1. Requisitos

Para cumplir con los objetivos que se han propuesto con el desarrollo de este proyecto se deben cumplir una serie de requisitos. Los requisitos software son la descripción de las características y las funcionalidades del sistema. En este apartado los dividiremos en requisitos funcionales, que son declaraciones de los servicios que proveerá el sistema y de la manera en que éste reaccionará a entradas particulares, y requisitos no funcionales, que son aquellos requerimientos que no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades emergentes de éste como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento.

4.1.1. Requisitos funcionales

En la tabla 2 se muestran los requisitos funcionales con los que debe contar la aplicación.

Tabla 3: Tabla de los requisitos funcionales de la aplicación

Identificador	**Descripción**
RF1.	Proporcionar un sistema que permita al usuario poder consultar la información relacionada con las aplicaciones y los datasets utilizados por estas.
RF2.	Ofrecer un sistema en el que el usuario pueda darse de alta en la aplicación mediante un registro.
RF3.	Debe haber alguna funcionalidad que permita al usuario autenticarse y posteriormente cerrar sesión.

RF4.	Debe ofrecer un buscador de aplicaciones avanzado a través de filtros para facilitar las búsquedas del usuario.
RF5.	Ofrecer un sistema para que los usuarios puedan votar las aplicaciones.
RF6.	Deberá disponer de una funcionalidad que permita al usuario realizar comentarios sobre las aplicaciones del portal.
RF7.	El usuario dado de alta en la aplicación debe poder dar de alta aplicaciones.
RF8.	El usuario que ha dado de alta aplicaciones debe poder editar la información de la misma cuando desee.
RF9.	El portal debe ofrecer un sistema para que el usuario pueda editar los datos de su perfil.
RF10.	El portal debe ofrecer un sistema para que el usuario pueda dar de baja su perfil en la aplicación.

4.1.2. Requisitos no funcionales

En la tabla 3 se exponen los requisitos no funcionales que debe ofrecer la aplicación que se quiere construir.

Tabla 4: Tabla de los requisitos no funcionales

Identificador	**Descripción**
RNF1.	La interfaz de la aplicación debe ser sencilla, intuitiva y usable.
RNF2.	El tiempo de respuesta de la aplicación debe ser rápido y el portal debe ser fluido.
RNF3.	El portal debe funcionar perfectamente en todas las últimas versiones de todos los navegadores web.

4.2. Casos de usos

Un caso de uso es una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema.

4.2.1. Descripción de actores

Se tratan de los roles que pueden jugar los agentes que interactúan con el sistema. Los roles son jugados por personas, dispositivos, u otros sistemas. Podríamos distinguir entre actores primarios, para los cuales el objetivo del caso de uso es esencial y actores secundarios, que interactúan con el caso de uso, pero cuyo objetivo no es esencial. En la figura 11 se pueden apreciar los diferentes actores de la aplicación.

- **Administrador**: Persona encargada de toda la gestión del portal. Controla el flujo de información y el buen funcionamiento de la aplicación. Administra el sistema completo.
- **Usuario anónimo**: Cualquier persona que interactúe con el sistema sin estar autenticado.
- **Usuario autenticado**: Cualquier persona que interactúe con el sistema utilizando un usuario registrado.

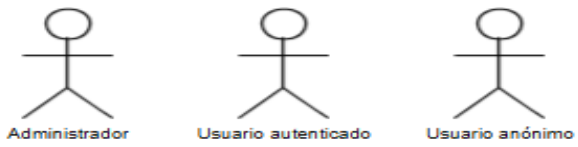


Figura 11 Actores de la aplicación

4.2.2. Diagrama y descripción de los casos de uso

En este apartado se expondrá el diagrama de casos de uso de la aplicación y las diferentes tablas explicativas de los mismos.

Los diagramas de caso de uso sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/o otros sistemas.

El diagrama de caso de uso de la aplicación se muestra en la figura 12.

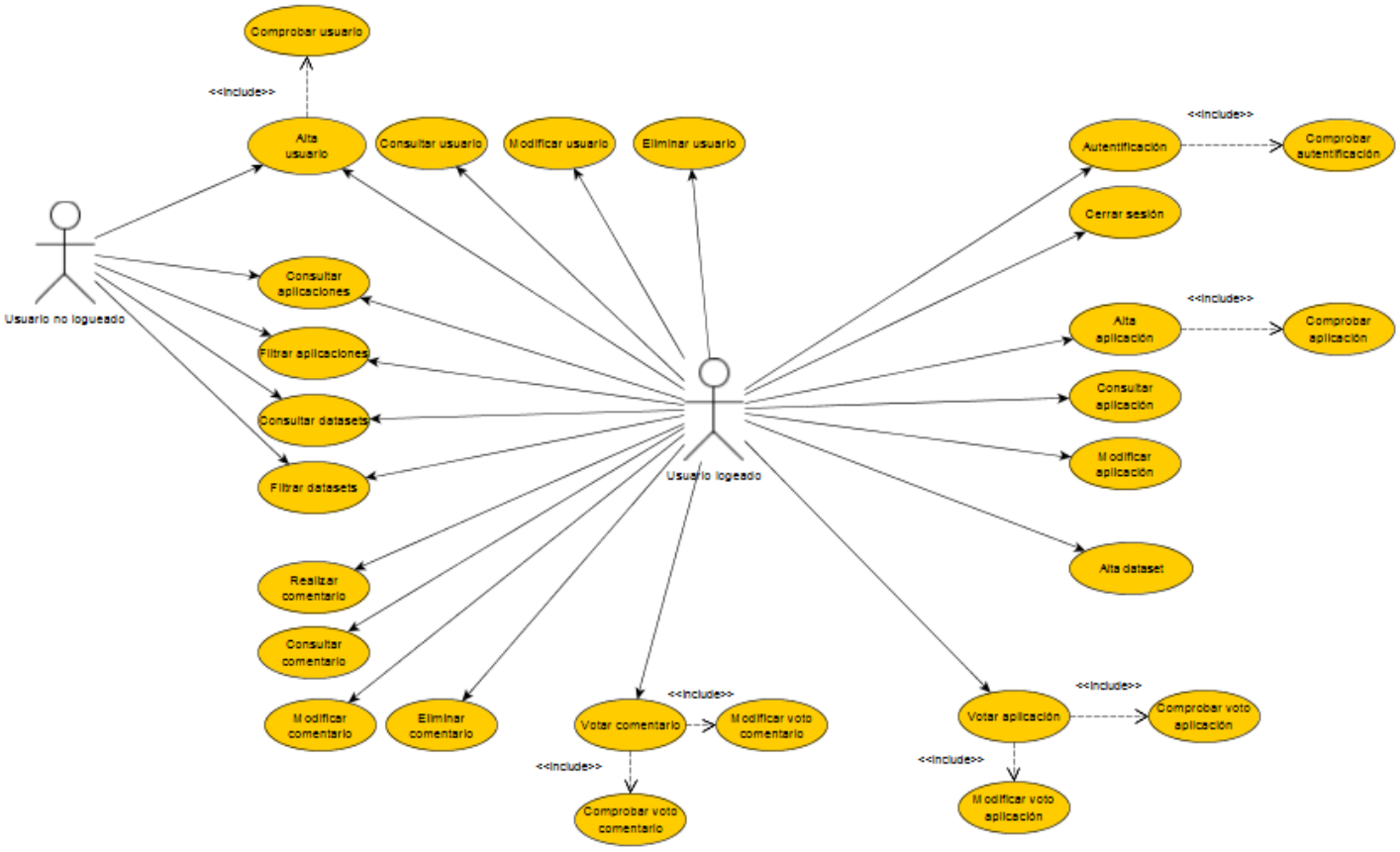


Figura 12 Diagrama casos de uso

A continuación se muestran las fichas que describen los casos de uso del diagrama.

Tabla 5: Caso de uso Autentificación

Nombre	**Autentificación**	
Descripción	Permite al usuario acceder a la aplicación como usuario autenticado.	
Precondición	El usuario debe estar registrado en la aplicación.	
Secuencia normal	Paso 1 2	Acción Introducir email o usuario y la contraseña. Comprobar si la autentificación es correcta.

		2a	Si no es correcta, el usuario sigue como usuario anónimo.
		2b	Si es correcta el usuario queda autenticado en la aplicación.
Postcondición	Una vez introducidos los datos y comprobada la autentificación el usuario queda autenticado.		
Excepciones	Paso	Acción	
	1	El usuario introduce datos incorrectos, entonces el sistema le avisará con un mensaje informando del error.	

Tabla 6: Caso de uso Comprobar autentificación

Nombre	**Comprobar autentificación**		
Descripción	Se comprueba si los datos introducidos por el usuario coinciden con los existentes en la base de datos.		
Precondición	Haber introducido email o usuario y la contraseña.		
Secuencia normal	Paso	Acción	
	1	Se buscan los datos del usuario en la base de datos por el email y usuario introducido.	
	2	Se compara la contraseña introducida por el usuario y la contraseña obtenida de la base de datos.	
		2a	Si no son iguales, El usuario no podrá ser autenticado.
		2b	Si son iguales el usuario queda autenticado en la aplicación.
Postcondición	El usuario queda autenticado.		
Excepciones	Paso	Acción	
	1	Si los datos introducidos no coinciden, el sistema avisará al usuario con un mensaje informando del error.	

Tabla 7: Caso de uso Cerrar sesión

Nombre	**Cerrar sesión**	
Descripción	El usuario cierra sesión en la aplicación.	
Precondición	El usuario debe estar autenticado.	
Secuencia normal	Paso	Acción
	1	El usuario debe pulsar sobre el texto logout de la aplicación.
Postcondición	El usuario volverá a la pantalla de inicio y quedará como usuario anónimo.	

Tabla 8: Caso de uso Alta usuario

Nombre	**Alta usuario**	
Descripción	Permite al usuario registrarse en la aplicación.	
Precondición		
Secuencia normal	Paso	Acción
	1	Rellenar adecuadamente el formulario de registro.
	2	Pulsar el botón de envío del formulario.
	2a	Si no existe el usuario, se da de alta en el sistema.
	2b	Si existe el usuario no pude ser dado de alta.
Postcondición	El usuario es dado de alta y guardado en la base de datos.	
Excepciones	Paso	Acción
	1	En caso de que el usuario que se esté dando de alta ya exista en la base de datos se le cancelará el alta y se le avisará del error.
	2	Si el usuario no introduce todos los datos correctamente

		para el registro se le informará del error.

Tabla 9: Caso de uso Comprobar usuario

Nombre	**Comprobar usuario**	
Descripción	Se comprueba si el usuario que quiere darse de alta existe en la base de datos.	
Precondición	Haber rellenado el formulario de registro correctamente.	
Secuencia normal	Paso 1	Acción Se consulta el usuario por el email y el usuario introducido.
	2a	Si el email o el usuario ya existen el usuario no podrá darse de alta.
	2b	Si el usuario o el email no existen se podrá efectuar el alta.
Postcondición	El usuario queda dado de alta en el sistema.	
Excepciones	Paso 1	Acción Si el usuario intenta darse de alta con un email que ya existe se le informará del error.

Tabla 10: Caso de uso Consultar usuario

Nombre	**Consultar usuario**
Descripción	Permite al usuario consultar los datos de su perfil. No se permite consultar los datos de otros usuarios.
Precondición	El usuario debe estar autenticado y existir en la base de datos.

Secuencia normal	Paso 1 2	Acción Iniciar sesión con el propio usuario. Pulsar sobre el nombre de usuario.
Postcondición	El usuario podrá visualizar los datos de su perfil en un formulario precargado. Esta vista será la misma que se utiliza para modificar el usuario.	
Excepciones		

Tabla 11: Caso de uso Modificar usuario

Nombre	**Modificar usuario**	
Descripción	Permite al usuario modificar los datos de su perfil.	
Precondición	El usuario debe existir en la base de datos y estar autenticado.	
Secuencia normal	Paso 1 2 3 4	Acción Acceder con su perfil a la aplicación. Pulsar sobre el nombre del usuario. Cambiar los campos a modificar en el formulario. Pulsar el botón que envía el formulario.
Postcondición	Una vez realizados los pasos anteriores el usuario quedará modificado en la base de datos.	
Excepciones	Paso 1	Acción Si el usuario no introduce todos los datos correctamente se le cancelará la modificación y se le informará del error.

Tabla 12: Caso de uso Eliminar usuario

Nombre	**Eliminar usuario**
Descripción	Permite al usuario eliminar su cuenta de perfil de la aplicación. Al darse de baja las aplicaciones dadas de alta por el usuario no

	se eliminarán de la base de datos.	
Precondición	El usuario debe existir en la base de datos y estar autenticado.	
Secuencia normal	Paso	Acción
	1	Acceder con su perfil a la aplicación.
	2	Pulsar sobre el nombre de usuario.
	3	Pulsar sobre el botón que elimina el perfil.
Postcondición	Una vez seguido estos pasos el usuario quedará eliminado de la aplicación.	
Excepciones		

Tabla 13: Caso de uso Alta aplicación

Nombre	**Alta aplicación**		
Descripción	Permite al usuario dar de alta una aplicación en el sistema.		
Precondición	El usuario debe existir en la base de datos y estar autenticado.		
Secuencia normal	Paso	Acción	
	1	Rellenar adecuadamente el formulario de alta de aplicación.	
	2	Pulsar el botón de envío del formulario.	
		2a	Si no existe la aplicación se da de alta.
		2b	Si existe la aplicación no puede ser dada de alta.
Postcondición	La aplicación es dada de alta en el sistema.		
Excepciones	Paso	Acción	
	1	En caso de que la aplicación que está siendo dada de alta ya exista, se cancelará el alta y se informará del error al usuario.	
	2	Si el usuario no introduce todos los datos del alta de la aplicación correctamente será informado del error.	

Tabla 14: Caso de uso Comprobar aplicación

Nombre	**Comprobar aplicación**		
Descripción	Se comprueba si la aplicación que quiere darse de alta ya existe en el sistema.		
Precondición	Rellenar el formulario de alta de aplicación correctamente.		
Secuencia normal	Paso	Acción	
	1	Se consulta la aplicación en la base de datos.	
		2a	Si la aplicación existe en la base de datos no se podrá dar de alta.
		2b	Si la aplicación no existe en la base de datos se podrá dar de alta.
Postcondición	La aplicación queda dada de alta en el sistema.		
Excepciones	Paso	Acción	
	1	Si el usuario intenta dar de alta una aplicación que ya existe se informará del error.	

Tabla 15: Caso de uso Consultar aplicación

Nombre	**Consultar aplicación**
Descripción	Permite al usuario consultar los datos privados de la aplicación. No se permite consultar los datos privados de aplicaciones de otros usuarios.
Precondición	La aplicación debe existir en la base de datos y el usuario debe estar autenticado.

Secuencia normal	Paso	Acción
	1	Iniciar sesión con el perfil del propio usuario.
	2	Pulsar sobre el botón de acceso a sus aplicaciones.
	3	Pulsar sobre la aplicación que desea consultar.
Postcondición	El usuario podrá visualizar los datos de su aplicación en un formulario precargado. Esta vista será la misma que se utiliza para modificar la aplicación.	
Excepciones		

Tabla 16: Caso de uso Modificar aplicación

Nombre	**Modificar aplicación**	
Descripción	Permite al usuario modificar los datos de su aplicación.	
Precondición	La aplicación debe existir en la base de datos y el usuario debe estar autenticado.	
Secuencia normal	Paso	Acción
	1	Acceder con su perfil al sistema.
	2	Pulsar sobre el botón de acceso a sus aplicaciones.
	3	Pulsar sobre la aplicación que desea modificar.
	4	Cambiar los campos a modificar en el formulario.
	5	Pulsar el botón que envía el formulario.
Postcondición	Una vez realizados los pasos anteriores la aplicación quedará modificada en la base de datos.	
Excepciones	Paso	Acción
	1	Si el usuario no introduce todos los datos correctamente se le cancelará la modificación y se le informará del error.

Tabla 17: Caso de uso Realizar comentario

Nombre	**Realizar comentario**	
Descripción	Permite al usuario realizar un comentario sobre alguna aplicación.	
Precondición	El usuario y la aplicación deben existir en la base de datos y el usuario debe estar autenticado.	
Secuencia normal	Paso	Acción
	1	Rellenar adecuadamente el formulario para realizar un comentario.
	2	Pulsar el botón de envío del formulario.
Postcondición	El comentario será dado de alta en el sistema.	
Excepciones	Paso	Acción
	1	Si el usuario no introduce todos los datos del alta del comentario correctamente será informado del error.

Tabla 18: Caso de uso Consultar comentario

Nombre	**Consultar comentario**	
Descripción	Permite al usuario consultar los datos del comentario.	
Precondición	El usuario y la aplicación deben existir en la base de datos y el usuario debe estar autenticado.	
Secuencia normal	Paso	Acción
	1	Iniciar sesión con el propio usuario.
	2	Pulsar sobre el botón de acceso a sus comentarios.
	3	Pulsar sobre el comentario que desea consultar.
Postcondición	El usuario podrá visualizar los datos de su comentario en un formulario precargado. Esta vista será la misma que se utiliza para modificar el comentario.	
Excepciones		

Tabla 19: Caso de uso Modificar comentario

Nombre	**Modificar comentario**	
Descripción	Permite al usuario modificar los datos de su comentario.	
Precondición	El usuario y la aplicación deben existir en la base de datos y el usuario debe estar autenticado.	
Secuencia normal	Paso	Acción
	1	Acceder con su perfil al sistema.
	2	Pulsar sobre el botón de acceso a sus comentarios.
	3	Pulsar sobre el comentario que desea modificar.
	4	Cambiar los campos a modificar en el formulario.
	5	Pulsar el botón que envía el formulario.
Postcondición	Una vez realizados los pasos anteriores el comentario quedará modificado en la base de datos.	
Excepciones	Paso	Acción
	1	Si el usuario no introduce todos los datos correctamente se le cancelará la modificación y se le informará del error.

Tabla 20: Caso de uso Eliminar comentario

Nombre	**Eliminar comentario**	
Descripción	Permite al usuario eliminar el comentario del sistema.	
Precondición	El usuario y la aplicación deben existir en la base de datos y el usuario debe estar autenticado.	
Secuencia normal	Paso	Acción
	1	Acceder con su perfil a la aplicación.
	2	Pulsar sobre el botón de acceso a sus comentarios.
	3	Pulsar sobre el botón que elimina el comentario.

Postcondición	Una vez seguido estos pasos el comentario quedará eliminado de la aplicación.	
Excepciones		

Tabla 21: Caso de uso Filtrar aplicaciones

Nombre	**Filtrar aplicaciones**	
Descripción	Permite a los usuarios filtrar las aplicaciones del sistema por categorías, país, valoración de los usuarios, visitas y precio.	
Precondición		
Secuencia normal	Paso 1 2 3	Acción Acceder a la página principal. Pulsar el botón de aplicaciones. Filtrar según el criterio del usuario.
Postcondición	Una vez seguido estos pasos la lista de aplicaciones se corresponderá al filtro elegido.	
Excepciones		

Tabla 22: Caso de uso Consultar aplicaciones

Nombre	**Consultar aplicaciones**	
Descripción	Permite al usuario consultar los datos públicos de las aplicaciones del sistema.	
Precondición		
Secuencia normal	Paso 1 2	Acción Acceder a la página principal. Pulsar sobre la aplicación que desea consultar.
Postcondición	El usuario podrá visualizar los datos públicos de la aplicación.	
Excepciones		

Tabla 23: caso de uso Filtrar datasets

Nombre	**Filtrar datasets**	
Descripción	Permite a los usuarios filtrar los datasets utilizados por las aplicaciones del sistema por categorías, país y ciudad.	
Precondición	El usuario debe estar autenticado.	
Secuencia normal	Paso	Acción
	1	Iniciar sesión en la aplicación
	2	Acceder a la página principal.
	3	Pulsar el botón de conjunto de datos.
	4	Filtrar según el criterio del usuario.
Postcondición	Una vez seguido estos pasos la lista de datasets se corresponderá al filtro elegido.	
Excepciones		

Tabla 24: Caso de uso Consultar datasets

Nombre	**Consultar datasets**	
Descripción	Permite al usuario consultar los datos de los datasets de las aplicaciones del sistema.	
Precondición	El usuario debe estar autenticado.	
Secuencia normal	Paso	Acción
	1	Iniciar sesión en la aplicación.
	2	Acceder a la página principal.
	3	Pulsar sobre el dataset que desea consultar.
Postcondición	El usuario podrá visualizar los datos del dataset.	
Excepciones		

Tabla 25: Caso de uso Votar aplicación

Nombre	**Votar aplicación**	
Descripción	Permite al usuario votar una aplicación del sistema.	
Precondición	El usuario y la aplicación debe existir en la base de datos y el usuario debe estar autenticado.	
Secuencia normal	Paso	Acción
	1	El usuario se autentifica con su perfil.
	2	Se sitúa en la lista de aplicaciones.
	3	Pulsa el botón para realizar un voto sobre alguna de las aplicaciones de la lista.
	2a	Si el usuario ya ha votado a esa aplicación se modifica el voto anterior por el nuevo.
	2b	Si el usuario no había votado la aplicación se añade el nuevo voto.
Postcondición	El voto es dado de alta en el sistema.	
Excepciones	Paso	Acción
	1	En caso de que el usuario ya hubiera votado la aplicación el voto se modificará y se le mostrará un mensaje indicando que el voto ha sido modificado.

Tabla 26: Caso de uso Comprobar voto aplicación

Nombre	**Comprobar voto aplicación**
Descripción	Se comprueba si el usuario ya ha realizado un voto sobre la aplicación.
Precondición	El usuario debe votar la aplicación.

Secuencia normal	Paso	Acción	
	1	Se consulta si el voto existe en la base de datos.	
		2a	Si el voto existe se modifica por el nuevo voto.
		2b	Si el voto no existe se da de alta en la base de datos.
Postcondición	El voto queda dado de alta en el sistema.		
Excepciones	Paso	Acción	
	1	Si el usuario intenta dar de alta un voto que ya existe se modificará el voto por el nuevo y se informará al usuario.	

Tabla 27: Caso de uso Modificar voto aplicación

Nombre	**Modificar voto aplicación**	
Descripción	Permite al usuario modificar el voto de la aplicación.	
Precondición	El voto debe existir en la base de datos y el usuario debe estar autenticado.	
Secuencia normal	Paso	Acción
	1	El usuario se autentifica con su perfil.
	2	Se sitúa en la lista de aplicaciones.
	3	Pulsa el botón para realizar un voto sobre alguna de las aplicaciones de la lista.
	4	
Postcondición	Una vez realizados los pasos anteriores el voto quedará modificado en la base de datos.	
Excepciones		

Tabla 28: Caso de uso Votar comentario

Nombre	**Votar comentario**		
Descripción	Permite al usuario votar un comentario sobre alguna aplicación del sistema.		
Precondición	El usuario y la aplicación debe existir en la base de datos y el usuario debe estar autenticado.		
Secuencia normal	Paso	Acción	
	1	El usuario se autentifica con su perfil.	
	2	Se sitúa en la lista de aplicaciones.	
	3	Pulsa sobre alguna aplicación.	
		Pulsa el botón para realizar un voto sobre algún comentario de la aplicación.	
		2a	Si el usuario ya ha votado a ese comentario se modifica el voto anterior por el nuevo.
		2b	Si el usuario no había votado el comentario se añade el nuevo voto.
Postcondición	El voto es dado de alta en el sistema.		
Excepciones	Paso	Acción	
	1	En caso de que el usuario ya hubiera votado el comentario se modificará y se le mostrará un mensaje indicando que el voto ha sido modificado.	

Tabla 29: Caso de uso Comprobar voto comentario

Nombre	**Comprobar voto comentario**
Descripción	Se comprueba si el usuario ya ha realizado un voto sobre el comentario.
Precondición	El usuario debe votar el comentario.

Secuencia normal	Paso	Acción	
	1	Se consulta si el voto existe en la base de datos.	
		2a	Si el voto existe se modifica por el nuevo voto.
		2b	Si el voto no existe se da de alta en la base de datos.
Postcondición	El voto queda dado de alta en el sistema.		
Excepciones	Paso	Acción	
	1	Si el usuario intenta dar de alta un voto que ya existe se modificará el voto por el nuevo y se informará al usuario.	

Tabla 30: Caso de uso Modificar voto comentario

Nombre	**Modificar voto comentario**	
Descripción	Permite al usuario modificar el voto de un comentario.	
Precondición	El voto debe existir en la base de datos y el usuario debe estar autenticado.	
Secuencia normal	Paso	Acción
	1	El usuario se autentifica con su perfil.
	2	Se sitúa en la lista de aplicaciones.
	3	Pulsa sobre alguna aplicación.
	4	Pulsa el botón para realizar un voto sobre algún comentario de la aplicación.
Postcondición	Una vez realizados los pasos anteriores el voto quedará modificado en la base de datos.	
Excepciones		

4.3. Solución propuesta

En este apartado se expone un estudio sobre algunos tipos de aplicaciones web y sus arquitecturas. Estas arquitecturas y aplicaciones se tomarán como referencia para construir nuestra solución.

4.3.1. Tipos de aplicaciones web

Existen distintos tipos de aplicaciones web [6]. Los más importantes y conocidos hoy en días son los siguientes:

- **Aplicación web estática**. Es un tipo de aplicación que muestra muy poca información y está pensada para no generar e incluir nuevos contenidos. Suelen estar desarrolladas en HTML y CSS. No obstante, pueden incluir videos, banners y GIFS.

  Este tipo de aplicación web no serviría para la implementación de nuestro portal web por razones evidentes, ya que en nuestro caso sí que necesitamos actualizar la información constantemente.

- **Aplicación web dinámica**. Son mucho más complejas a nivel técnico que las anteriores. Utilizan bases de datos para cargar la información y los contenidos se van actualizando cada vez que el usuario accede a la web app. Existen muchos lenguajes de programación para aplicaciones web dinámicas como por ejemplo PHP y ASP, que son los más populares porque permiten una buena estructuración del contenido. Es muy sencillo actualizar los contenidos y se pueden añadir foros, o bases de datos. También el diseño de la web se puede cambiar y retocar.

  Este tipo de aplicación es una opción para tener en cuenta a la hora de elegir la aplicación web que vamos a construir, ya que cuenta con muchas de las características que necesitamos que tenga el portal para poder cumplir con todos los requisitos y poder construir todos los casos de uso.

- **E-commerce**. Es el tipo de aplicación web pensado para tiendas online. El desarrollo es más complejo al tener que crearse unas pasarelas de pago para tarjetas de crédito, PayPal, etc. Además de sincronizarse con la gestión de stocks y logística. Habrá que crear un panel de gestión donde se subirán los

productos y se irán actualizando o eliminándose, y donde se gestionaran los pedidos y pagos. Podemos encontrar miles de ejemplos, desde Amazon hasta El Corte Inglés.

Este caso no se ajusta para nada a las necesidades que tenemos en nuestro portal, ya que poco tiene que ver con una tienda online.

- **Portal web app**. Con el término portal, nos referimos a un tipo de aplicación en el que la página principal permite el acceso a diversos apartados, categorías o secciones. Puede haber de todo: foros, chats, correo electrónico, un buscador, zona de acceso con registro, contenido más reciente, etc.

  Esta aplicación web también tiene bastantes características con las que debe contar nuestro portal web.

- **Aplicación web animada**. Son aplicaciones web creadas con la tecnología Flash. Esta tipología de programación permite crear y presentar contenidos con efectos animados. Es una tecnología muy atractiva para desarrolladores y diseñadores. El problema que tienen las webs animadas es que no son útiles para mejorar el posicionamiento ni para optimizar el SEO; los buscadores no pueden leer correctamente las informaciones.

  Este tipo de aplicación es más para aplicaciones que tengan una gran nivel de exigencias en los requisitos relacionados con la interfaz gráfica. En nuestro caso el diseño no debe ser muy recargado y no es necesario que cuente con animaciones.

- **Aplicación web con "Gestor de Contenidos".** Perfecta para proyectos que necesitan actualizar su contenido constantemente. Tienen un gestor de contenidos (CMS) a través del cual el administrador y los editores pueden ir añadiendo los contenidos, realizando los cambios y actualizaciones, etc. Muchas empresas han optado por este tipo de aplicaciones web, por la facilidad de publicar contenidos.

  Se podría usar perfectamente en nuestro caso ya que cuenta con todas las propiedades y características necesarias para poder construir nuestro portal.

Una vez realizado el estudio de las aplicaciones web más populares que existen en este momento procedemos a su análisis. En la tabla 31 se muestra una comparativa en la que se puede apreciar que tipo de aplicación puede satisfacer nuestras necesidades a la hora de implementar la solución.

Tabla 31 Comparación aplicaciones webs

Tipo de aplicación web	Cumple con las características
Aplicación web estática	NO
Aplicación web dinámica	SI
E-commerce	NO
Portal web app	SI
Aplicación web animada	NO
Aplicación web con "Gestor de Contenidos"	SI

Como vemos existen tres tipos de aplicaciones en las que fijarnos. Basándonos en las descripciones de cada una de ellas en este mismo apartado, se llega a la conclusión de que la aplicación que mejor se adapta a nuestro portal es una combinación entre el tipo de **aplicación dinámica** y la de **portal web app** por los siguientes motivos:

- Necesitamos la base de datos para poder almacenar toda la información.
- El usuario necesita que el portal vaya actualizando la información constantemente.
- Existirá una interacción usuario aplicación que actualizará la información y la página web.
- La página principal deberá permitirá acceso a distintas secciones, apartados y categorías.

Los motivos por los que se ha descartado la opción de **Aplicación web con "Gestor de Contenidos"** son los siguientes:

- El principal motivo es porque aunque con un CMS podríamos satisfacer todos los requisitos, pero unos de los objetivos por los que elegí este TFG es porque quería enfocarlo más hacia la programación web y en este tipo de aplicación no se necesita programar.
- A la hora de poder personalizar el estilo de la página y tener que modificar el código HTML o CSS es más complejo.

4.4. Arquitectura

En este apartado del capítulo 4 se va a explicar todo lo relacionado con la arquitectura que tiene una aplicación web y la que va a tener la aplicación que se va a construir en este proyecto.

4.4.1. Arquitectura de una aplicación web

Una aplicación Web [7] es proporcionada por un servidor Web y utilizada por usuarios que se conectan desde cualquier punto vía clientes Web (browsers o navegadores). La arquitectura de un Sitio Web tiene tres componentes principales:

- Un servidor Web
- Una conexión de red
- Uno o más clientes

El servidor web distribuye páginas de información formateada a los clientes que las solicitan. Los requerimientos son hechos a través de una conexión de red, y para ello se usa el protocolo HTTP. Una vez que se solicita esta petición mediante el protocolo HTTP y la recibe el servidor web, éste localiza la página web en su sistema de archivos y la envía de vuelta al navegador que la solicitó.

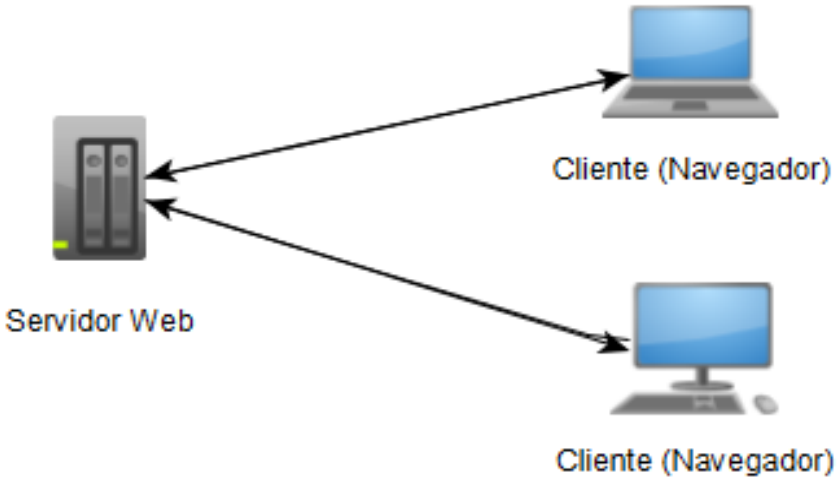


Figura 13 Arquitectura web básica

Esta es la arquitectura más básica y general de una aplicación web (Figura 13). En nuestra plataforma se quiere ir un poco más allá y en el siguiente punto de este capítulo se expondrá más concretamente la arquitectura de la aplicación web OpenDataApps.

4.4.2. Arquitectura SPA

Uno de los factores principales para la experiencia de usuario es el tiempo de carga. Para mejorarlo se conocen varias técnicas como minimizar archivos, retardar la ejecución de código que no se utilice, usar la caché…, pero aun así el navegador del cliente debe volver a parsear y ejecutar el código CSS y javascript, descargar y parsear el código HTML, reconstruir el DOM, renderizar la interfaz, por tanto el usuario ve como la página se construye mientras espera [8].

SPA es un nuevo enfoque para construir aplicaciones web. Todo el código se carga en la primera llamada o posteriormente de forma dinámica, sin recargar la página. La navegación se resuelve y ejecuta en el cliente. Las llamadas al servidor se ejecutan de forma asíncrona y la interfaz se construye en el cliente. Una aplicación muy popular construida con este tipo de arquitectura es Gmail [8].

4.4.2.1. ¿Qué diferencia hay respecto a aplicaciones web básicas?

Habitualmente la lógica de negocio (el código ejecutable) de aplicaciones web se realiza íntegramente en el lado del servidor, y se confía la propia naturaleza del

sistema de URLs el mostrar una "vista de aplicación" u otra. Por ej. en un e-commerce el carrito estará en una URL concreta, mientras que la pantalla de login estará bajo otra URL totalmente diferente. Para el navegador, cada URL diferente es completamente independiente del resto: Aunque tenga los mismos estilos y/o plantillas, estos tienen que volver a ser procesados desde cero. Esto, para la gran mayoría de páginas web dinámicas, implica que al cambiar entre vistas se sufrirá el problema de la latencia en la web [9].

Otra característica negativa de esta arquitectura es que el estado de la aplicación del cliente es difícil de mantener, teniendo que hacer auténticos malabarismos para poder gestionar una simple transferencia de información de una vista a otra.

Las **ventajas** de las aplicaciones SPA son las siguientes:

- Interfaz más sencilla.
- Mantenimiento más sencillo.
- Distribución de carga. El servidor puede ser más ligero.
- Comienzo del desarrollo más ágil.
- La interfaz es simplemente otro cliente.
- Se presenta muy bien para testing.
- Perfecto para combinar con aplicaciones móviles.

Las **desventajas** de las aplicaciones SPA son las siguientes:

- La primera carga puede ser lenta.
- El SEO se vuelve complejo.
- Requiere conocimientos adicionales y avanzados de javascript.
- Es más difícil de analizar.

En nuestro caso después de la investigación anterior se ha decidido construir la aplicación utilizando este tipo de arquitectura. Un motivo importante por el que se ha tomado esta decisión es por la experiencia personal adquirida en la asignatura de Programación en Internet, dónde se realizaron dos aplicaciones web: la primera de ellas basada en una arquitectura cliente/servidor simple utilizando servlets y JSPs y

la segunda una SPA utilizando Java en la parte de servidor y AngularJS en el cliente. Pude apreciar la mayoría de las ventajas que se comentan en este apartado, especialmente las ventajas relacionadas con el desarrollo.

4.4.3. Arquitectura de OpenDataApps

En la figura 14 se muestra la arquitectura que tendrá la aplicación web que se va a construir.

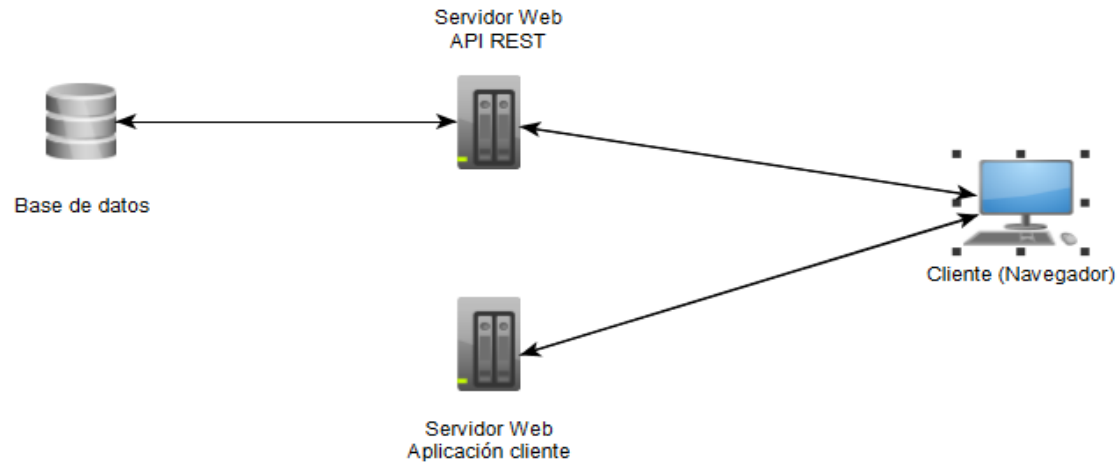


Figura 14 Arquitectura OpenDataApps

La arquitectura de la aplicación la componen los siguientes componentes:

- **Base de datos**: Es el sistema de información donde se almacenará toda la información perteneciente a la aplicación. Este sistema estará desplegado en un servidor de base de datos que será el encargado de intercambiar la información con el servidor web que tiene desplegada la API REST.
- **API REST**: Será el servicio web encargado de mapear los datos de la base de datos y enviarlos al cliente en el formato que se soliciten. También ejecutará parte de la lógica de negocio de la aplicación.
- **Aplicación cliente**: Será el servidor web donde se almacenan todos los ficheros necesarios para la primera descarga en el cliente. En este servidor no se ejecutará nada. A la hora de desplegar toda la aplicación a producción estos ficheros pueden estar en el mismo servidor donde está desplegada la API REST, pero se muestra así para que se vea claramente que la aplicación cliente y la API REST están totalmente desacopladas.

- **Cliente**: Es el navegador desde donde se hacen las peticiones a los servidores. La primera petición se realizará al servidor donde se almacenan todos los ficheros de la aplicación cliente y se descargarán en el mismo. Una vez descargados, en el navegador se empezará a ejecutar lógica de negocio. Esta lógica tendrá peticiones AJAX (asíncronas) a la API REST. Dependiendo de la navegación que realice el usuario se irá ejecutando una parte del código u otro.

4.5. Diseño de la base de datos del sistema

En la figura 15 se muestran todas las entidades que componen el diseño de la base de datos con sus propiedades y las relaciones entre ellas.

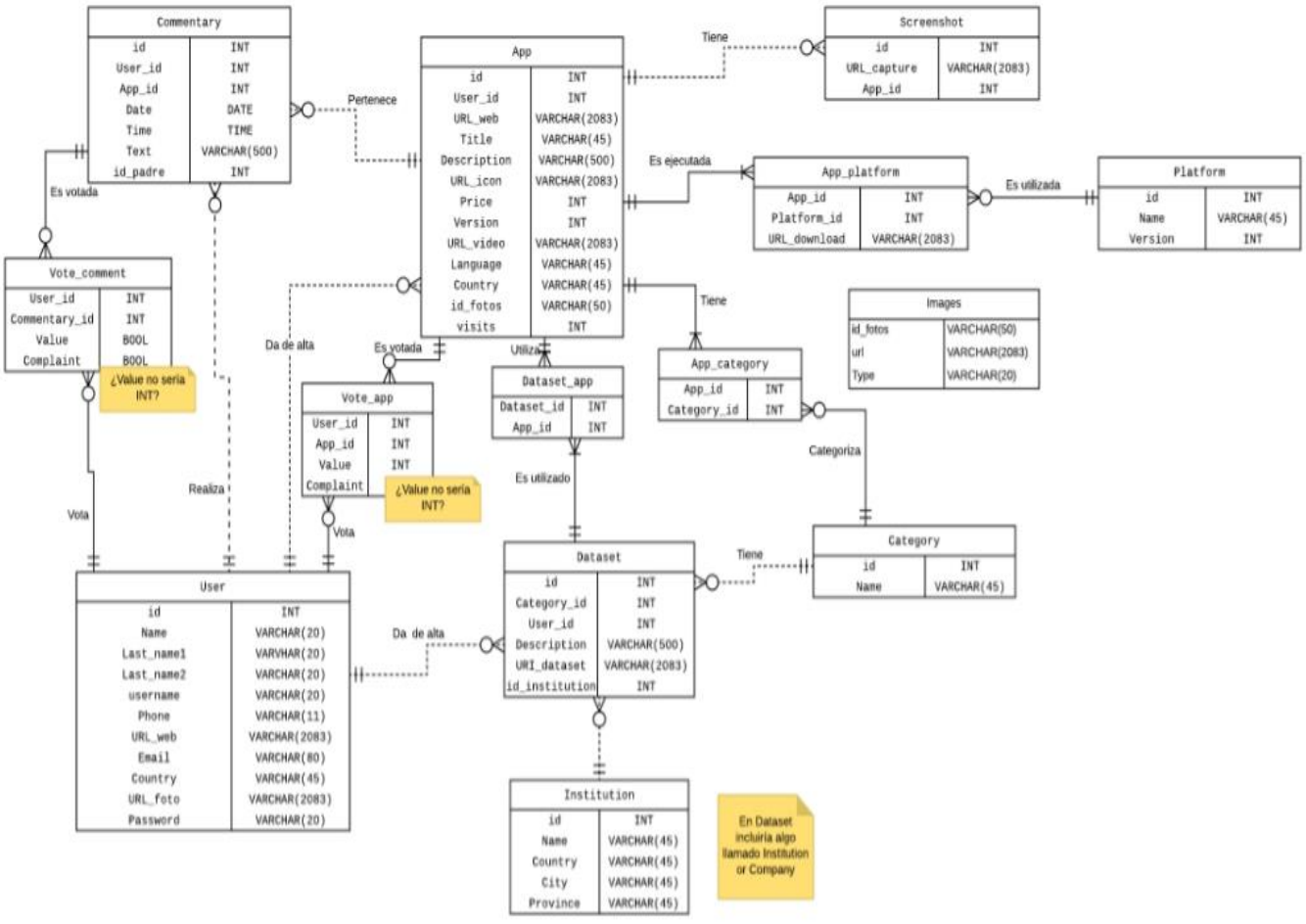


Figura 15 Diagrama Entidad/Relación Base de datos

4.5.1. User

Todas las personas que se registren en el portal serán usuarios. Una vez dados de alta en la base de datos podrán iniciar sesión en la aplicación y realizar todas las operaciones que le están permitidas a un usuario autenticado. La información que se almacena por cada usuario es la siguiente:

- Id: Es el id que les identifica en la base de datos.
- Name: Nombre de la persona que se da de alta.
- Last_name1: Primer apellido de la persona que se da de alta.
- Last_name2: Segundo apellido de la persona que se da de alta.
- Username: nombre de sesión
- Phone: Teléfono del usuario.
- Url _web: Url de su web de desarrollador si la tiene.
- Email: email del usuario.
- Country: País del usuario.
- Url_foto: Url de la foto de perfil de usuario. Este campo no se utiliza a día de hoy.
- Password: Contraseña del usuario para iniciar sesión en la aplicación.

4.5.2. Commentary

Los comentarios sobre aplicaciones que pueden realizar los usuarios autenticados que han iniciado sesión en la aplicación. La información que se almacenará sobre estos comentarios es la siguiente:

- Id: Identificador único.
- User_id: Id del usuario propietario del comentario.
- App_id: Id de la App sobre la que se realiza el comentario.
- Date: Fecha en la que se realiza el comentario.

- Time: Hora en la que se realiza el comentario.
- Text: El contenido del comentario.
- Id_padre: Id del comentario al que se está respondiendo en caso de ser respuesta.

4.5.3. Vote_comment

En esta entidad se almacena la información relacionada con un voto realizado sobre un comentario.

- User_id: Id del usuario que realiza el voto.
- Commentary_id: Id del comentario sobre el que se realiza el voto.
- Value: Valor del voto. En este caso solo podrá ser 0 (Negativo) o 1 (Positivo).

4.5.4. App

Se almacena toda la información relacionada con las aplicaciones que dan de alta los usuarios en el portal web. La información que se almacena es:

- Id: Identificador único de las aplicaciones.
- User_id: Id del usuario que da de alta la aplicación.
- Url_web: Url de acceso a la aplicación o de acceso a su descarga.
- Title: Título de la aplicación.
- Description: Descripción que explica que tema trata.
- Url_icon: Url del icono de la aplicación.
- Price: Precio de descarga de la aplicación.
- Version: Versión de la aplicación.
- Url_video: Url de vídeo promocional si se tiene.
- Language: Lenguaje de la aplicación.
- Country: País de la aplicación.

- Id_fotos: Id que se genera en el formulario donde se da de alta la aplicación para enlazar las imágenes con la aplicación.
- Visits: Número de visitas que ha recibido.

4.5.5. Vote_app

Entidad en la que se almacena toda la información sobre los votos que los usuarios realizan sobre las aplicaciones.

- User_id: Id del usuario que realiza el voto.
- App_id: Id de la app sobre la que se realiza el voto.
- Value: Valor del voto, que puede tomar un valor entero entre 1 y 5.
- Complaint: Valor de una posible denuncia.

4.5.6. Institution

En esta entidad se guarda toda la información de las instituciones que publican los datasets que han dado de alta los usuarios en el portal. Estas instituciones se darán de alta cuando se de alta a un dataset en caso de no existir la institución.

- Id: Identificador único de las instituciones.
- Name: Nombre del órgano de la institución.
- Country: País a la que pertenece la institución.
- City: Ciudad a la que pertenece la institución.
- Province: Provincia a la que pertenece la institución.

4.5.7. Dataset

Toda la información que se necesita para los conjuntos de datos que los usuarios den de alta en la aplicación. Se darán de alta cuando los usuarios estén dando de alta aplicaciones y los datasets no existan.

- Id: Identificador único de un dataset.
- Category_id: Id de la categoría a la que pertenece la información del conjunto de datos.
- User_id: Id del usuario que lo ha dado de alta en la aplicación.

- Description: Descripción del conjunto de datos.
- URI_dataset: URI para poder obtener la información que ofrece el dataset.
- Id_intitution: Id de la institución que ha publicado el conjunto de datos.

4.5.8. Dataset_app

En esta entidad se almacena los id que relacionan las apps con los datasets que utilizan.

- Dataset_app: Id del dataset.
- App_id: Id de la app.

4.5.9. Category

En esta entidad se almacena la información relacionada con las categorías con las que se categorizarán tanto las aplicaciones como los datasets dados de alta en la aplicación. Las aplicaciones podrán tener más de una categoría, en cambio los datasets solo podrán estar asociados a una categoría.

- Id: Identificador único de la categoría.
- Name: Nombre de la categoría.

4.5.10. App_category

Entidad que almacena los id que relacionan las aplicaciones con las categorías.

- App_id: Id de la app relacionada.
- Categroy_id: Id de la categoría relacionada.

4.5.11. Platform

Se guarda todas las propiedades necesarias para almacenar la información perteneciente a las plataformas en las que se pueden ejecutar las aplicaciones que dan de alta los usuarios en el portal.

- Id: Identificador único de plataformas.
- Name: Nombre de la plataforma.
- Version: Versión de la plataforma.

4.5.12. App_platform

Entidad que guarda los id que relaciona las aplicaciones dada de alta en el portal con las plataformas donde se pueden ejecutar.

- App_id: Id de la aplicación relacionada.
- Platform_id: Id de la plataforma relacionada.
- URL_download: URL de descarga en caso de existir.

4.5.13. Sreenshot

Entidad que almacena la información relevante con las capturas de las aplicaciones del portal.

- Id: Identificador único de captura.
- URL_capture: URL de la captura.
- App_id: Id de la app a la que pertenece.

4.5.14. Images

Esta tabla no está relacionada con ninguna otra entidad. Almacena las url de las imágenes subidas al servidor con un id de fotos que identifica al usuario o la aplicación a la que pertenece.

- Id_fotos: Es un identificador generado en cada formulario donde se da de alta una aplicación o un usuario. El objetivo es enlazar a la aplicación o al usuario con las imágenes que se suben al servidor en dicho formulario, ya que la subida se produce antes de que el usuario o la aplicación se den de alta en la base de datos y no se dispone de sus id para enlazarlos con las imágenes.
- Url: Es la url de la imagen subida al servidor.
- Type: Indica el tipo que puede ser capture en caso de ser una captura de una aplicación, icon si es el icono de una aplicación o user si es la imagen de perfil de un usuario.

Así queda el diseño de la base de datos. Como se puede observar ha sufrido algunos cambios, ya que hay tablas que se han creado en fases posteriores al análisis y diseño porque en dicho momento no se tuvieron en cuenta esas necesidades y hay entidades

que ahora mismo no se usan ya que en la siguientes fases surgieron problemas que hubo que solucionar creando otras alternativas.

4.6. API REST

Para la parte del servidor web se va a implementar y desplegar un servicio web denominado API REST.

REST [10] es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON. Es una alternativa en auge a otros protocolos estándar de intercambio de datos como SOAP (Simple Object Access Protocol), que disponen de una gran capacidad pero también mucha complejidad. A veces es preferible una solución más sencilla de manipulación de datos como REST.

El objetivo que se quiere conseguir con este servicio web es poder intercambiar y recibir información en el cliente y allí mismo que se ejecute una gran parte de la lógica de negocio.

Algunas de las ventajas que ofrece REST son las siguientes:

- **Separación entre el cliente y el servidor**: el protocolo REST separa totalmente la interfaz de usuario del servidor y el almacenamiento de datos. Eso tiene algunas ventajas cuando se hacen desarrollos. Por ejemplo, mejora la portabilidad de la interfaz a otro tipo de plataformas, aumenta la escalabilidad de los proyectos y permite que los distintos componentes de los desarrollos se puedan evolucionar de forma independiente. Además del beneficio comentado anteriormente de que el servidor donde se despliega va a ser más ligero y necesitará menos recursos que un servidor donde se despliegue una aplicación web dinámica.
- **Visibilidad, fiabilidad y escalabilidad**. La separación entre cliente y servidor tiene una ventaja evidente y es que cualquier equipo de desarrollo puede escalar el producto sin excesivos problemas. Se puede migrar a otros servidores o realizar todo tipo de cambios en la base de datos, siempre y cuando los datos de cada una de las peticiones se envíen de forma correcta. Esta separación facilita tener en servidores distintos el frontend y el backend

y eso convierte a las aplicaciones en productos más flexibles a la hora de trabajar.

- La API REST siempre es **independiente del tipo de plataformas o lenguajes**: la API REST siempre se adapta al tipo de sintaxis o plataformas con las que se estén trabajando, lo que ofrece una gran libertad a la hora de cambiar o probar nuevos entornos dentro del desarrollo. Con una API REST se pueden tener servidores PHP, Java, Python o Node.js. Lo único que es indispensable es que las respuestas a las peticiones se hagan siempre en el lenguaje de intercambio de información usado, normalmente XML o JSON. Incluso no depende del cliente que realice las peticiones, siempre que las haga utilizando los métodos correctos y pasando la información que se espera recibir.

4.7. Patrones de diseño

En este apartado se van a exponer los patrones de diseño que se van a utilizar en este proyecto.

4.7.1. Patrón MVC

Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores [11].

- **Modelo**: Es la capa donde se trabaja con los datos, por tanto contendrá mecanismos para acceder a la información y también para actualizar su estado.
- **Vista**: Las vistas, como su nombre nos hace entender, contienen el código de nuestra aplicación que va a producir la visualización de las interfaces de usuario.
- **Controlador**: Contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una compra, una búsqueda de información, etc.

En nuestra aplicación este patrón lo vamos a seguir tanto globalmente como en partes más específicas. A nivel general la API REST contendrá código de la parte de modelo para mapear los datos de la base de datos y de la parte de controlador para

ejecutar la lógica de negocio. De la parte de la vista se ocupará totalmente la aplicación que se descarga y se ejecuta en el cliente.

A nivel más específico la API REST estará dividida en MVC. El modelo serán las clases donde se mapeen los datos, El controlador todo el código que manipula los datos y las vista será el formato en el que se entreguen los datos. En la aplicación que se ejecuta en el cliente el modelo serán los objetos javascript donde se mapean los datos, el controlador toda la parte que manipula los datos de esos objetos y la vista todo lo relacionado con la interfaz de usuario.

4.7.2. Patrón DAO

Es bastante normal hacer aplicaciones que almacenan y recogen datos de una base de datos. Suele ser habitual, también, querer hacer nuestra aplicación lo más independiente posible de una base de datos concreta, de cómo se accede a los datos o incluso de si hay o no base de datos detrás. Nuestra aplicación debe conseguir los datos o ser capaz de guardarlos en algún sitio, pero no tiene por qué saber de dónde los está sacando o dónde se guardan [12].

El párrafo anterior explica cuál es el objetivo principal de usar el patrón DAO (Data Access Object). En nuestra aplicación queremos conseguirlo e implementarlo en la API REST que es la parte encargada de interactuar con el sistema de información de la aplicación.

5. Implementación

Este capítulo expone todo lo relacionado con la fase de implementación del proyecto OpenDataApps. Se ha optado por explicar algunas partes de este apartado con gran nivel de detalle porque se espera que en el futuro haya nuevas versiones del proyecto realizadas por distintas personas y se pretende facilitarles en lo posible la comprensión de la versión actual.

5.1. Herramientas colaborativas

Son las herramientas utilizadas en la comunicación entre personas, aunque ésta sólo sea de forma virtual. Estas nuevas herramientas de la comunicación nos han permitido de forma eficaz y rápida el traspaso de información y han acortado de una forma u otra las distancias. En el ámbito académico, en particular en educación superior, las herramientas de trabajo colaborativo son un potente recurso que fácilmente se adapta a diversas necesidades y objetivos [13].

En este proyecto se han utilizado las siguientes herramientas colaborativas:

- **Lucidchart**: Es una herramienta de diagramación basada en la web, que permite a los usuarios colaborar y trabajar juntos en tiempo real, en la que se puede crear entre otros muchos, diagramas Entidad/Relación.

  En una reunión que se tuvo en enero de 2017, el objetivo era comenzar a construir una primera versión del diagrama Entidad/Relación de la base de datos. En dicha reunión se llegó a la conclusión de que el diseño sufriría cambios en fases posteriores, por tanto, se necesitaba una herramienta colaborativa para ir viendo los cambios y decidir si esos cambios eran los correctos. En la reunión se buscó y miró las herramientas que nos podían servir para este propósito y al final se decidió usar lucidchart.

  En esta aplicación se puede ingresar con una cuenta de google y crear y compartir los diagramas rápidamente. También tiene una funcionalidad que aún está en fase beta para poder exportar el código SQL.

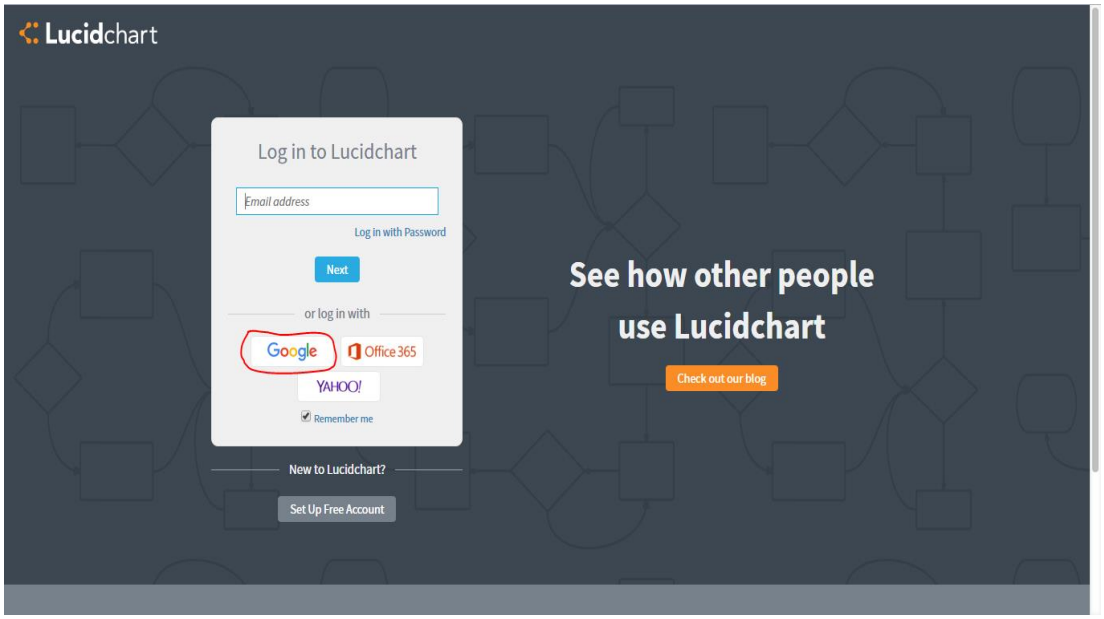


Figura 16 Acceso con google a lucidchart

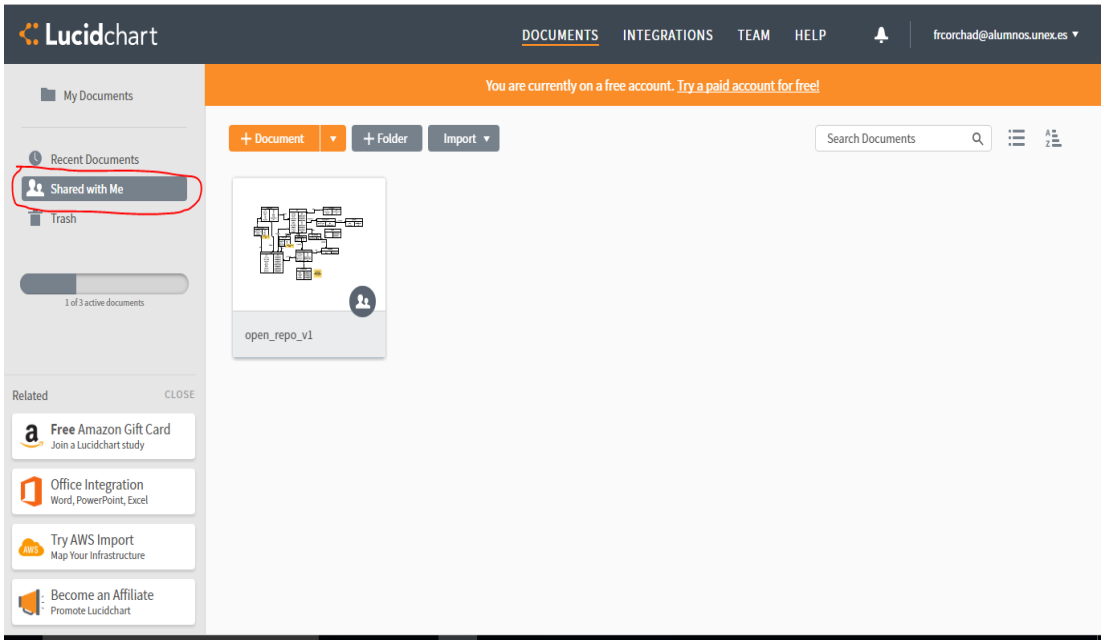


Figura 17 Panel de control de lucidchart

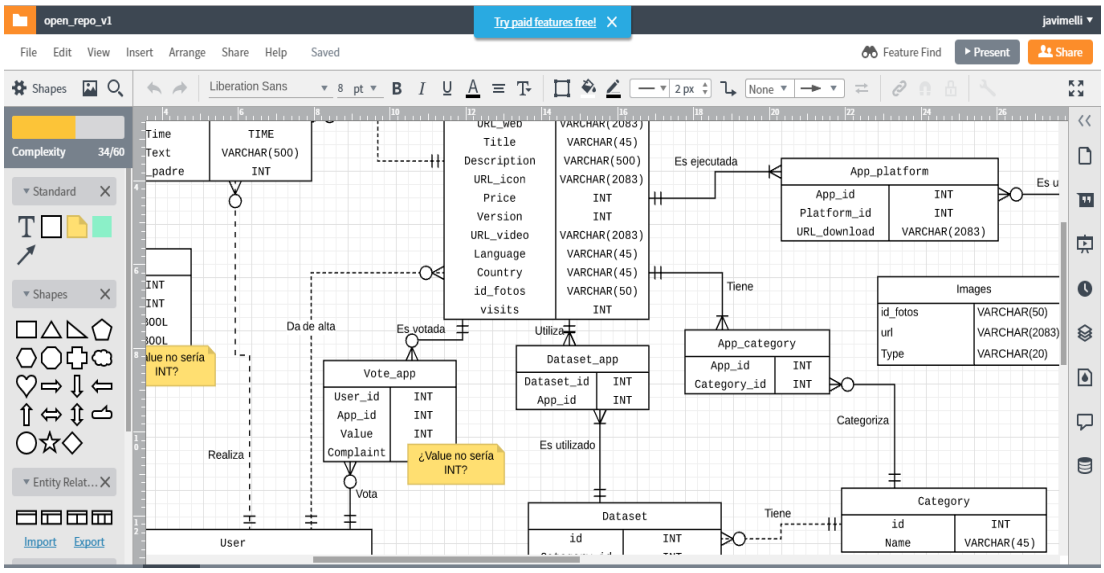


Figura 18 Diagrama E/R colaborativo

- **Control de versiones**: El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante [14].

  Debido al gran número de ficheros con los que cuenta tanto el servicio web REST como en la aplicación cliente y al gran número de líneas de código que los componen, será necesario utilizar un control de versiones por motivos de seguridad. Otro motivo importante es porque se va a trabajar en ordenadores diferentes y se tiene la necesidad de tener actualizado y actualizar todos los cambios que se produzcan en el proyecto.

  Existen varios tipos de software que permiten realizar un control de versiones sobre los proyectos, pero sin ninguna duda el más popular, sobre el que más documentación se escribe en la web y el más utilizado es **GIT.**

  GIT es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente [14].

  Para poder utilizarlo simplemente se descarga el ejecutable en su web oficial y se instala. En este desarrollo se ha utilizado Windows 10 en ambos

ordenadores, por tanto se tiene instalada la versión para este sistema operativo.

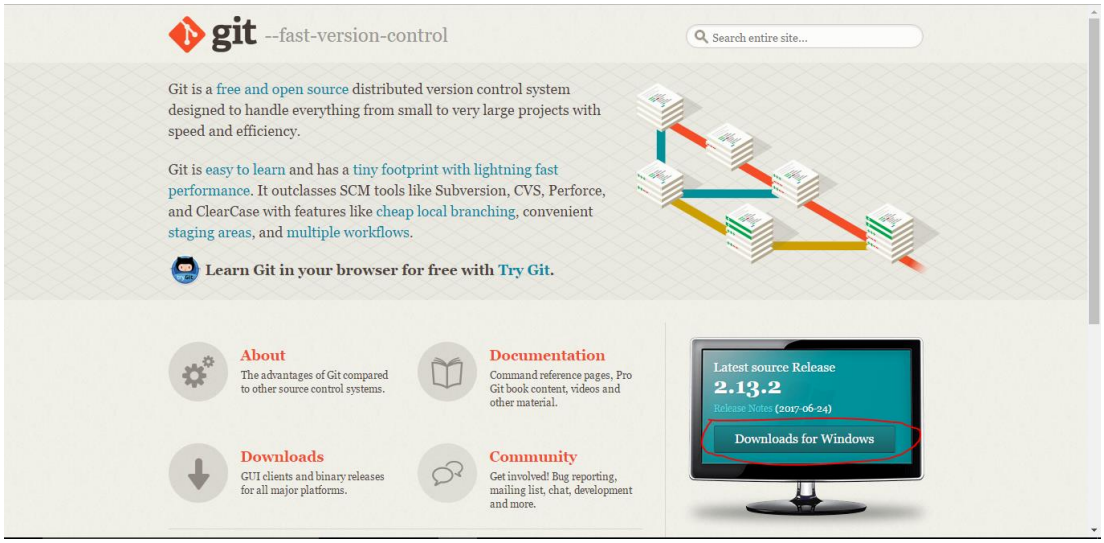


Figura 19 Página de descarga de GIT

Una vez instalado GIT necesitamos un repositorio remoto en la web para poder alojar nuestros proyectos. Para ello se va a utilizar **GitHub.**

Github es una plataforma creada para facilitar el desarrollo colaborativo de software, nos permite alojar proyectos en la web gratuitamente [15]. Para ello necesitamos crearnos una cuenta y un usuario. Una vez hecho esto, se crean los repositorios remotos en GitHub y los locales en los ordenadores donde se va a desarrollar. Por último, se enlazan los repositorios locales con el remoto.

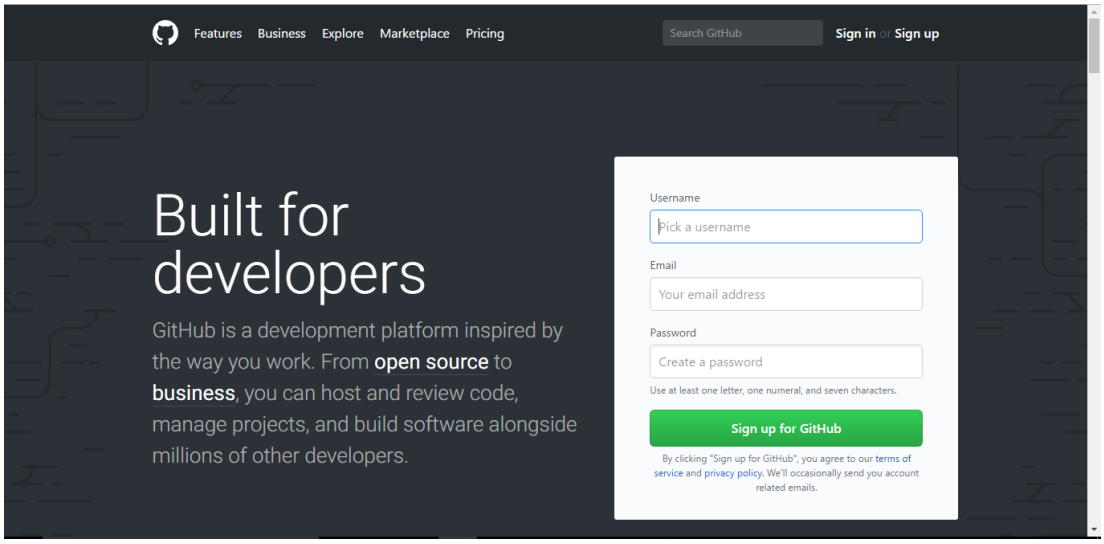


Figura 20 Página de inicio de GitHub

En este proyecto se han creado dos repositorios remotos, uno para la API REST y otro para el proyecto cliente.

El repositorio de la API REST es **appsrepository** y su url de enlace es https://github.com/javimelli/appsrepository.git.

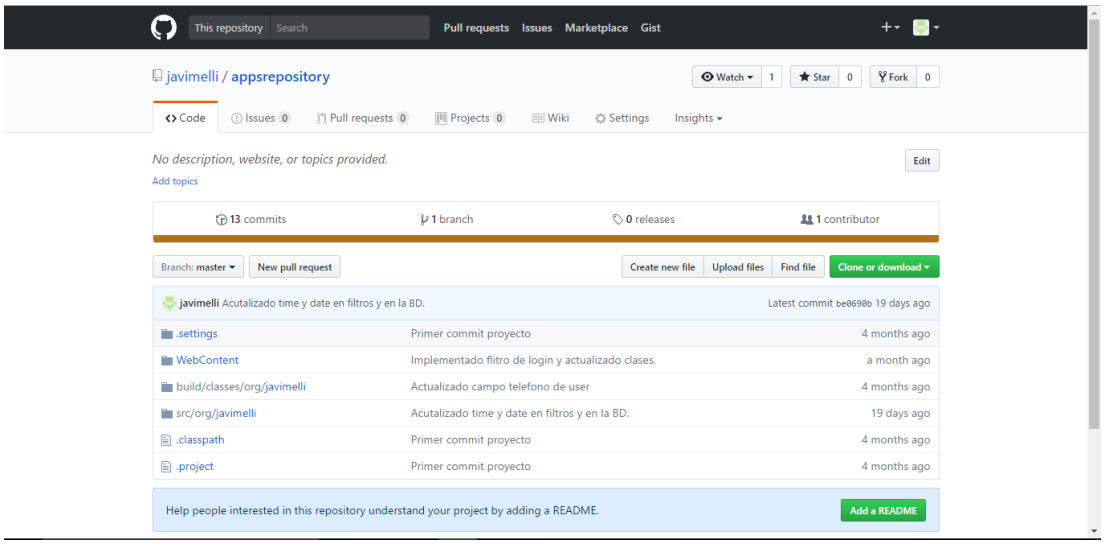


Figura 21 Repositorio remoto appsrepository

El repositorio de la aplicación cliente es **appsrepository_client** y su url de enlace es https://github.com/javimelli/appsrepository_client.git.

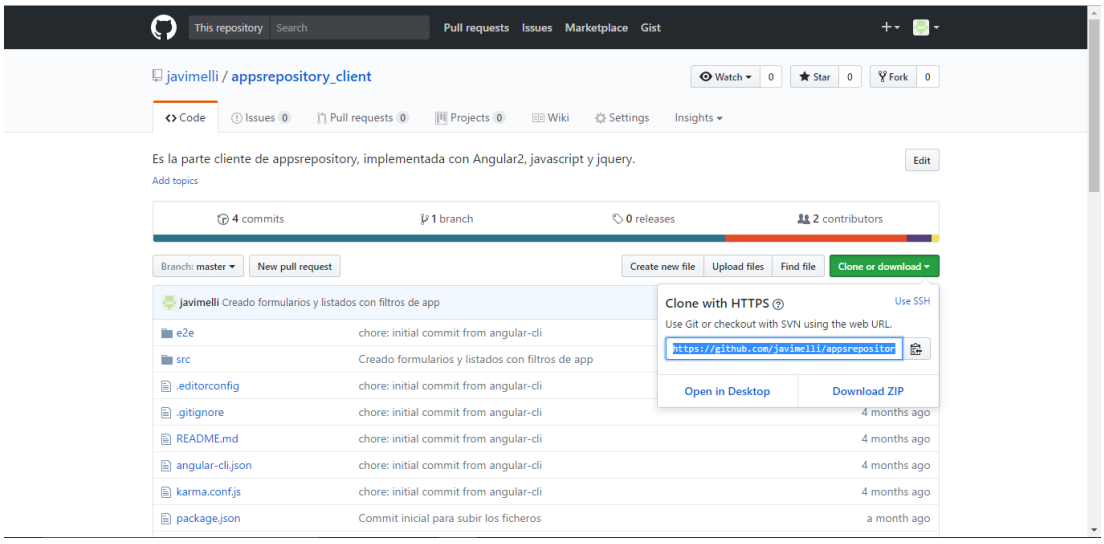


Figura 22 Repositorio remoto appsrepository_client

5.2. Construcción del sistema de información

Como se ha comentado en el apartado de objetivos de este documento en este proyecto, se necesitará un sistema de información donde almacenar, leer y actualizar toda la información de la aplicación. Para ello se construirá una base de datos.

En primer lugar, para la construcción del sistema, la primera pregunta es si usar un sistema de gestión de base de datos SQL o NoSQL. Veamos algunas diferencias entre ellas [16].

- SQL permite combinar de forma eficiente diferentes tablas para extraer información relacionada, mientras que NoSQL no lo permite o muy limitadamente.
- NoSQL permite distribuir grandes cantidades de información; mientras que SQL facilita distribuir bases de datos relacionales.
- SQL permite gestionar los datos junto con las relaciones existentes entre ellos; en NoSQL no existe este tipo de utilidades.
- NoSQL permite un escalado horizontal sin problemas – por su capacidad de distribución-; mientras que escalar SQL resulta más complicado.

Teniendo en cuenta estas diferencias se decide que el sistema gestor de base de datos debe ser SQL, ya que en este proyecto se necesita que exista relación entre tablas y la información que se almacenará en ellas.

5.2.1. Comparación Sistemas gestores de base de datos

Existen varios sistemas gestores de base de datos SQL. En la tabla 32 se realiza una comparación de los más populares [17].

Tabla 32 Comparativa sistemas gestores de base de datos

Sistema Gestor BD	**Características**	**Ventajas**	**Desventajas**	**Opinión**
DB2	Es propietario. Es propietario IBM. Integra XML de forma nativa. Es relacional. Arquitectura similar a Oracle. El SQL es muy potente. Tiene muchos años.	Multiplataforma Elimina tareas rutinarias, esto permite menor uso de recursos hardware. Tiene una versión gratuita llamada DB2 Express-C. Escalable. Estable.	No es tan robusto como Oracle. Puede ser caro.	Es un DBMS que puede ser caro por el hecho de no ser tan robusto en comparación con otros sistemas de gestión de bases de datos.
MySQL	Propietaria y pública. Portabilidad.	Fácil de aprender y utilizar. Multiplataforma. Código abierto. Fácil configuración. Veloz a realizar operaciones.	El soporte para disparadores es muy básico No soporta algunas conversiones de datos. Los privilegios de las tablas no se borrar de forma automática.	Mysql es uno de los DBMS más populares que hay y es debido al hecho que además de ser eficiente es de código libre y gratuito en algunas versiones y también incluye versiones de pago.
Oracle	Propietaria. Portable. Compatible. Alto rendimiento.	DBMS popular. Oracle ofrece porte técnico. Permite la gestión de múltiples bases de datos.	Una mala configuración ofrece resultados desfavorables.	Oracle es un BAMS de paga que tiene como beneficio su fiabilidad y su soporte.

PostgreSQL	Incluye herencia entre las tablas. Incorpora estructuras de arrays.	Ahora en costos. Instalación limitada. Estabilidad. Gran capacidad de almacenamiento.	Lento en inserciones y actualizaciones. Ofrece soporte en línea.	Tiene características específicas que los hacen especial para ciertas necesidades.
SQLite	Dominio público. DBMS relacional. Algunos lenguajes de programación lo incluyen en sus módulos o bibliotecas.	Multiplataforma. Muchos lenguajes de programación tienen soporte o módulos para sqlite. Pequeño tamaño.	Su límite es de 2 terabytes su base de datos. En algunas versiones los tipos de datos los asigna a los valores individuales y no a columnas esto en ocasiones no permite se portable a otras bases de datos.	Es una buena alternativa como DBMS en especial para aplicaciones, por poner un ejemplo de sus aplicaciones Mozilla Firefox, blackberry, android, Skype el reproductor clementine guardan sus datos en sqlite.
InterBase	Propietario. Arquitectura única. El lenguaje de procedimientos y trigger es muy potente.	Para Microsoft Windows y Linux. Permite hacer copias de seguridad en caliente. Tiene cercanía al estándar SQL.	No permite realizar particiones. No es popular.	Su principal inconveniente es su arquitectura única y su poca popularidad pero a pesar de eso este DBMS tiene características muy poderosas que lo hace único.
Microsoft SQL Server	Propietario. Integra nuevas herramientas. Recuperación de datos eficaz y rápida Portabilidad.	Para Windows. Soporte de transacciones. Estabilidad. Seguridad. Soporte de	Utiliza muchos recursos computaciones como memoria RAM. Es de pago.	La principal ventaja es su fiabilidad a la hora de recuperar datos.

		procedimientos almacenados. Entorno gráfico.		

En una de las reuniones se comentó que una de las características con las que debía contar el SGBD (Sistema gestor de base de datos) es que fuera público. Teniendo en cuenta esta característica y analizando la comparación anterior los sistemas de base de datos que nos quedan para elegir se reduce a MySQL, SQLite y Postgre SQL.

Por motivos de uso la decisión se reduce a MySQL y PostgreSQL, ya que no se tiene suficiente experiencia previa con SQLite.

Al final se toma la decisión de usar **MySQL** por los siguientes motivos:

- Como se puede apreciar en la tabla de comparación, MySQL es más rápida en algunas operaciones vitales como inserciones y actualizaciones.
- Durante la investigación realizada se han encontrado más fuentes de consulta en la web sobre MySQL que sobre PostgreSQL.

5.2.2. Construcción

Una vez decidido el sistema gestor de base datos (MySQL) comenzamos a construir la base de datos que se utilizará para almacenar, leer y actualizar la información que se va a utilizar en el portal web.

En primer lugar, se debe descargar e instalar el servidor de base de datos de la página oficial.

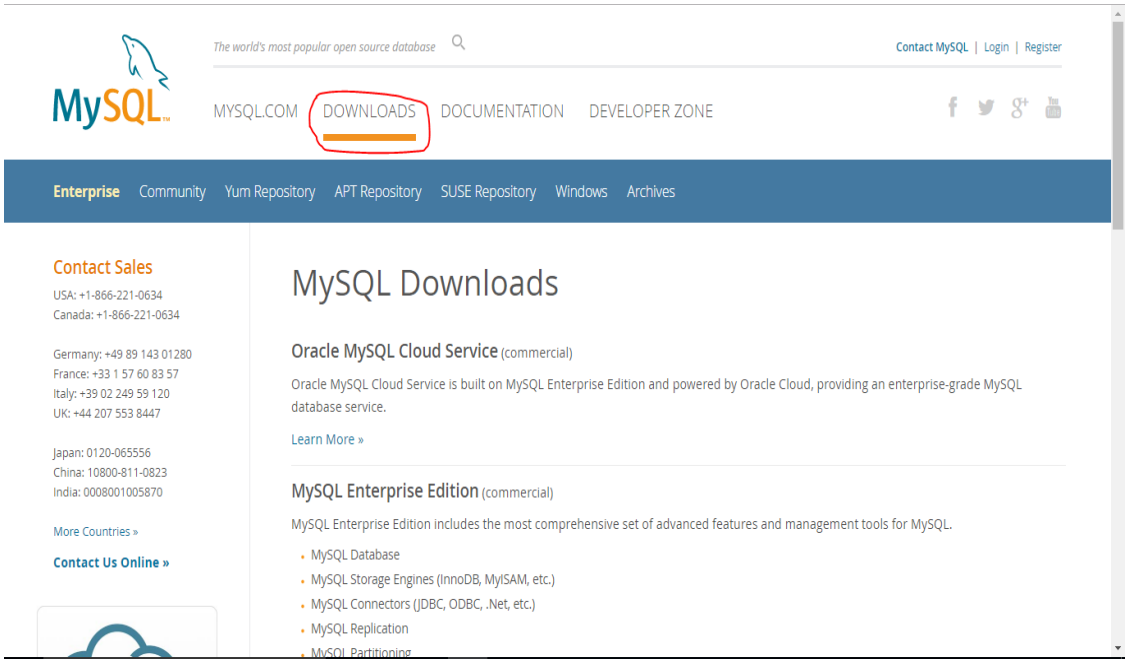


Figura 23 Página de descarga MySQL

Como se ha comentado en apartados anteriores, la fase de desarrollo se llevará a cabo en ordenadores diferentes. Por tanto, sería un problema utilizar servidores de base de datos instalados localmente en cada ordenador, ya que cada servidor tendría información diferente y cada modificación en el diseño implicaría hacerlo en cada servidor instalado.

La solución ha sido crear la base de datos en un servidor MySQL en la nube y gratuito llamado **db4free**.

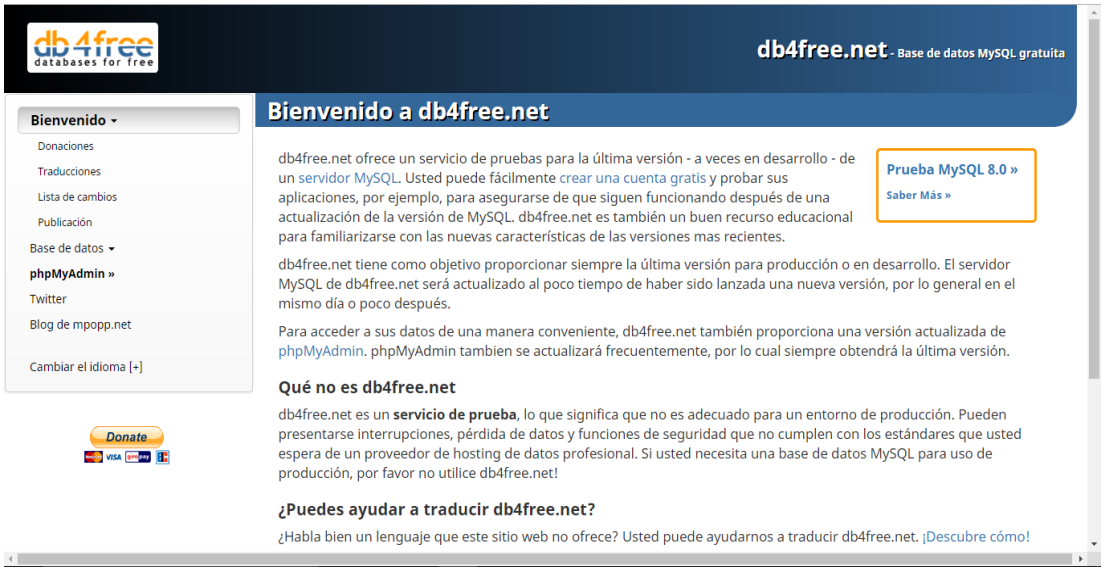


Figura 24 Página inicial db4free

Para conectar con un servidor de base de datos MySQL existen tres clientes:

- Línea de comandos: Existe un cliente por línea de comandos que permite realizar todas las operaciones necesarias sobre el servidor MySQL.
- PhpMyAdmin: Es el cliente web de MySQL para conectar con el servidor. Dispone de una interfaz gráfica muy sencilla e intuitiva. Uno de los problemas de db4free es que la conexión es bastante lenta y utilizar este tipo de cliente la ralentiza mucho más.
- Workbench: Es el cliente de escritorio de MySQL. Es una opción muy buena y la que se ha utilizado para la construcción del sistema.

Workbench proporciona una característica muy buena y clave para tomar la decisión de utilizar esta herramienta que es la de poder crear un diagrama Entidad/Relación y a partir de él construir la base de datos en el servidor.

Para crear la base de datos en primer lugar, se creó una conexión al servidor db4free utilizando las credenciales que obtuvimos al darnos de alta en su página web.

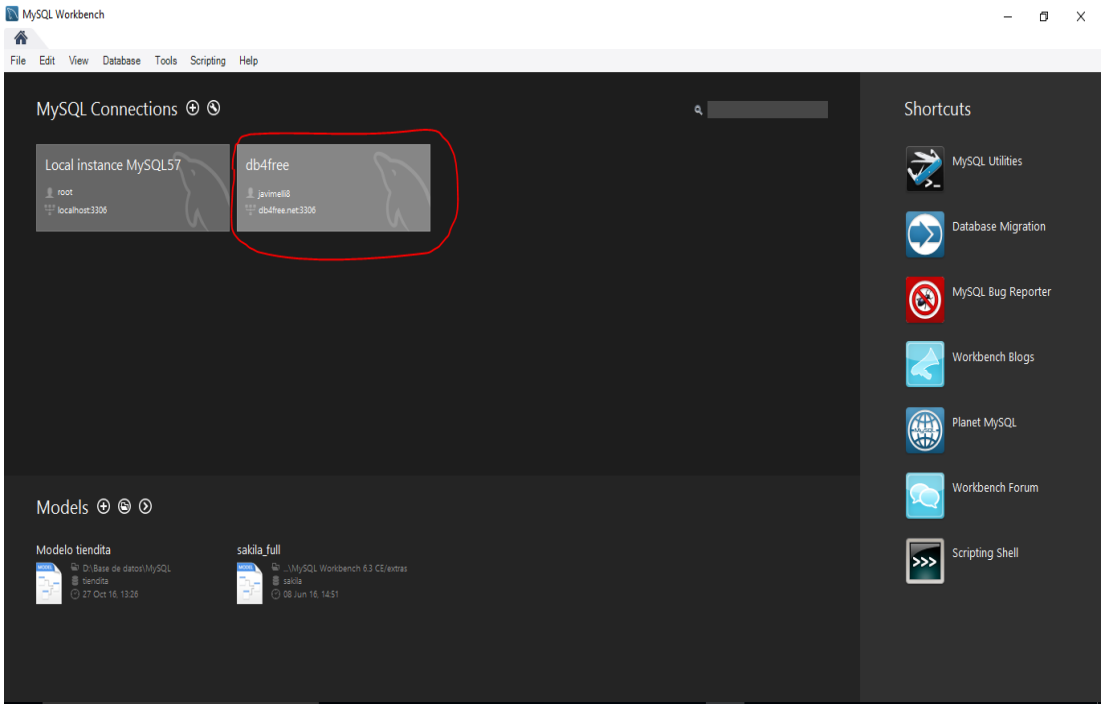


Figura 25 Conexión servidor db4free

En la figura 26 se muestra el panel de administración del servidor. En el panel izquierdo se poder ver todas las opciones y las bases de datos que existen en el servidor. La única que tenemos creada es la de appsrepository que es la base de datos

que utilizaremos para el proyecto. En el panel central se van escribiendo y ejecutando las sentencias SQL.

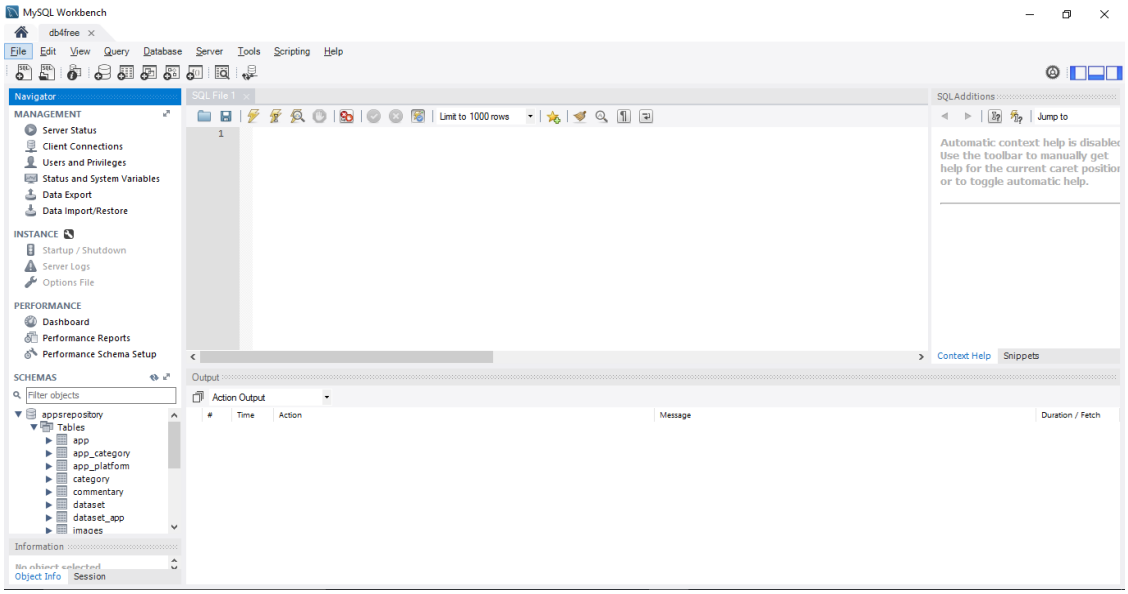


Figura 26 Panel de trabajo Workbench

5.3. Desarrollo de la parte del servidor web (back-end)

Este apartado documenta todo lo relacionado con la fase de implementación del servicio web. Este servicio será el encargado de interactuar directamente con el sistema de información.

5.3.1. Tecnología

Para la construcción del servicio web se pueden usar diferentes lenguajes de programación como pueden ser PHP, Python, Ruby, Java y Node.js.

El lenguaje de programación influirá poco en el rendimiento del servicio. Lo que más influirá es el uso de patrones de diseño y realizar una programación optimizada.

5.3.1.1. Java

Teniendo en cuenta el párrafo anterior el lenguaje de programación que se ha elegido para la construcción del servicio es **Java,** ya que en la asignatura de programación en internet, que se estudia en este grado, construimos una API REST con este lenguaje.

Figura 27 Icono Java

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo [18].

Vamos a comentar algunas ventajas y desventajas [19] de este lenguaje de programación.

Ventajas

- Lenguaje Multi-plataforma: El código que es escrito en java es leído por un intérprete, por lo que su programa andará en cualquier plataforma.
- Manejo automático de la memoria. (para los que vienen de C/C++). El manejo de la memoria se hace automáticamente y utilizando el garbage collector.
- Su uso es gratuito.
- Es unos de los lenguajes más utilizados por los programadores, por tanto existe mucha información. El siguiente gráfico de la figura 28 muestra el uso de los lenguajes de programación actualmente.

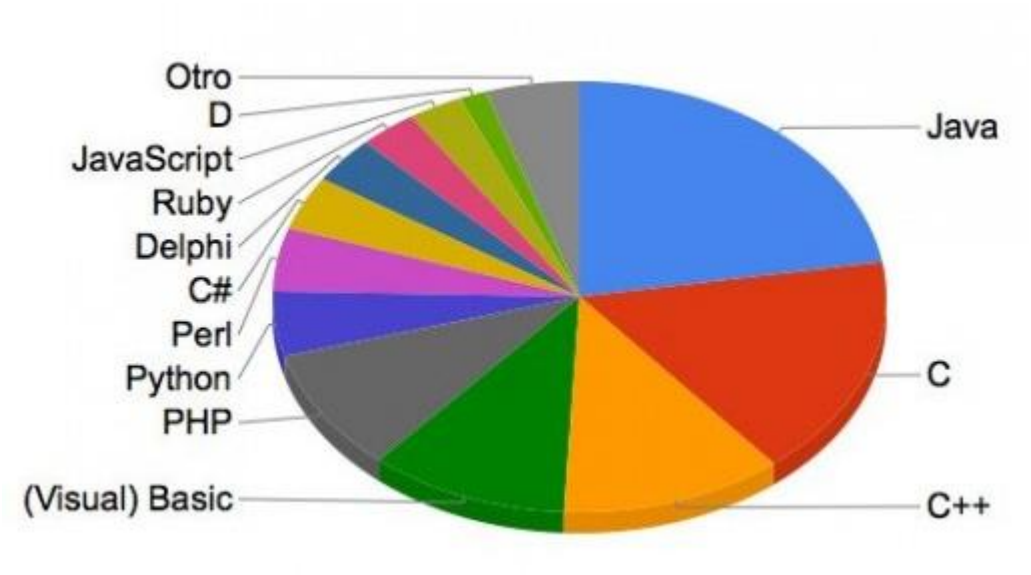


Figura 28 Uso de los lenguajes de programación

Desventajas

- Algunas implementaciones y librerías pueden tener código rebuscado.
- Algunas herramientas tienen un costo adicional.
- Lentitud al ejecutar algunas de sus aplicaciones.

Para programar con Java se necesita instalar en el ordenador donde se vaya a desarrollar el JDK. Java Development Kit o (JDK), es un software que provee herramientas de desarrollo para la creación de programas en Java. Puede instalarse en una computadora local o en una unidad de red. También se necesita el JRE o máquina virtual para poder ejecutar los programas que se escriban con Java. Para ello vamos a la página oficial y descargamos el paquete completo y lo instalamos. En este caso se ha descargado la versión 1.8.

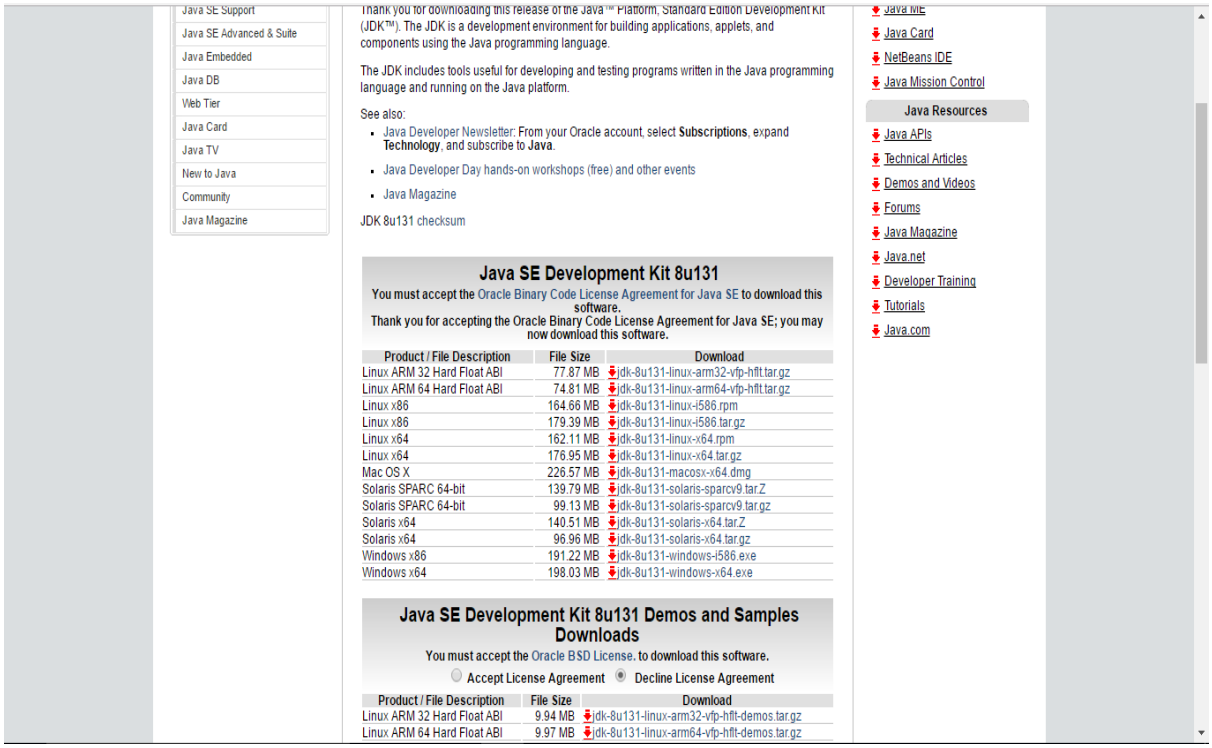


Figura 29 Página inicial descarga JDK

Elegimos el paquete dependiendo del sistema operativo donde se vaya a instalar. Por último debemos configurar las variables de entorno con los ejecutables del JDK. Se deben actualizar la variable PATH y crear la variable JAVA_HOME añadiendo el directorio bin del JDK descargado.

5.3.1.2. Eclipse

Para desarrollar cualquier aplicación en Java existen muchos IDEs. Los más conocidos son Eclipse e Intellij. La elección de un IDE u otro se basa totalmente en la comodidad para desarrollar. Despendiendo de cada desarrollador elegirán unos u otros.

En mi caso y porque es con el que he desarrollado durante todo el grado elegiré Eclipse. Se descarga desde la página oficial y simplemente se hace click sobre el ejecutable descargado para abrirlo y poder utilizarlo. Para este desarrollo se ha utilizado la versión NEON.

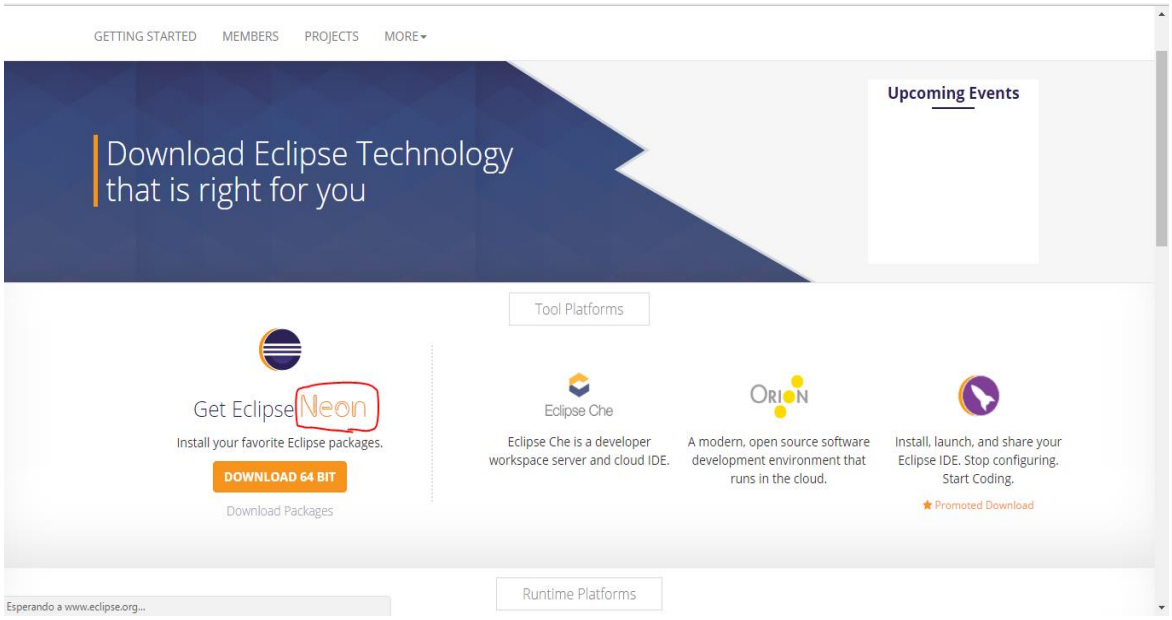


Figura 30 Página de descarga de Eclipse

5.3.1.3. Tomcat

Para poder ejecutar cualquier aplicación web, en nuestro caso el servicio web API REST, se necesita un servidor web. El servidor web que vamos a utilizar es **Tomcat**.

Tomcat [19] es un servidor Web con soporte para servlets y JSPs. Trae incluido el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor Web Apache. A partir de la versión 4.0, Tomcat utiliza el contenedor de servlets Catalina.

Tomcat puede funcionar como servidor Web por sí mismo. Al principio de su desarrollo existió la percepción de que la utilización de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con mínimos requisitos de velocidad y gestión de transacciones. Actualmente ya no existe esa percepción y Tomcat es usado como servidor Web independiente en entornos con alto nivel de tráfico y alta disponibilidad.

En primer lugar, para poder utilizar Tomcat se debe ingresar en la página oficial y descargar la versión que se desee. Para este desarrollo se va a utilizar la versión 9.0.

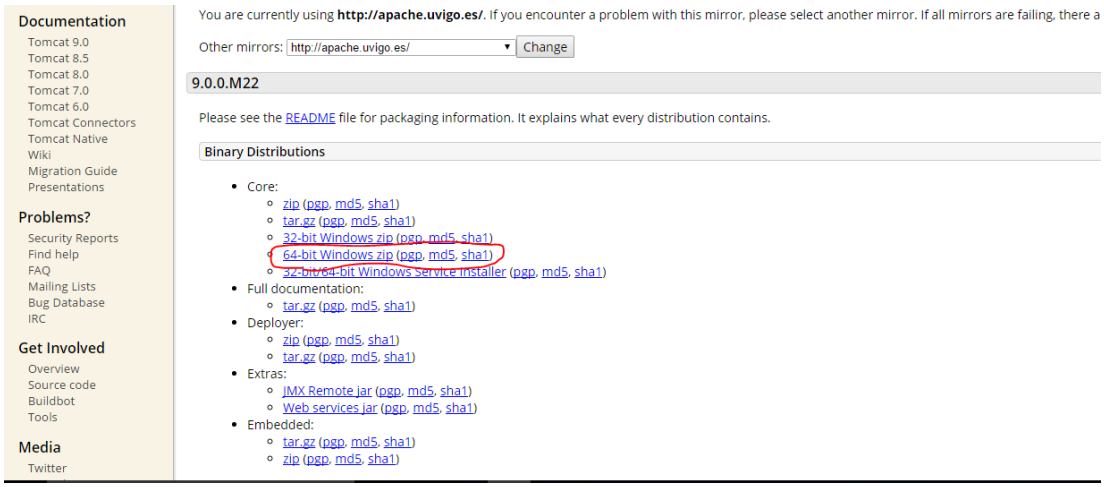


Figura 31 Página de descarga de Tomcat

Como se puede apreciar en la figura 31 la versión se ha descargado es para el sistema operativo Windows.

Una vez descargado en el ordenador simplemente hay que desplegar las aplicaciones en la carpeta webapps y lanzar el servidor con el script startup.bat situado en la carpeta bin.

Usar este tipo de despliegue de la aplicación implicaría que cada vez que se realizaran cambios en el proyecto java, para poder probarlos habría que empaquetar la aplicación de nuevo y desplegarla en el servidor. Pues bien, otro de los motivos por los que se ha elegido eclipse como IDE para desarrollar el servicio web, es porque ofrece una funcionalidad en la cual se puede embeber un servidor web y así desarrollar y probar las aplicaciones de forma retroalimentada.

Para ello simplemente debemos ir a la pestaña de servers en eclipse como se muestra en la figura 32, pulsar el botón derecho del ratón y hacer click en new server.

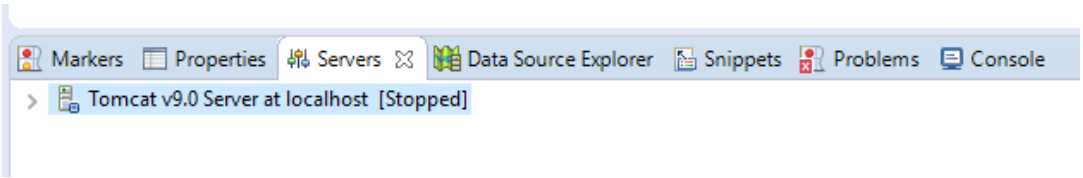


Figura 32 Pestaña server Eclipse

Aparecerá un menú en el que tendremos que indicar la versión del servidor que vamos a utilizar.

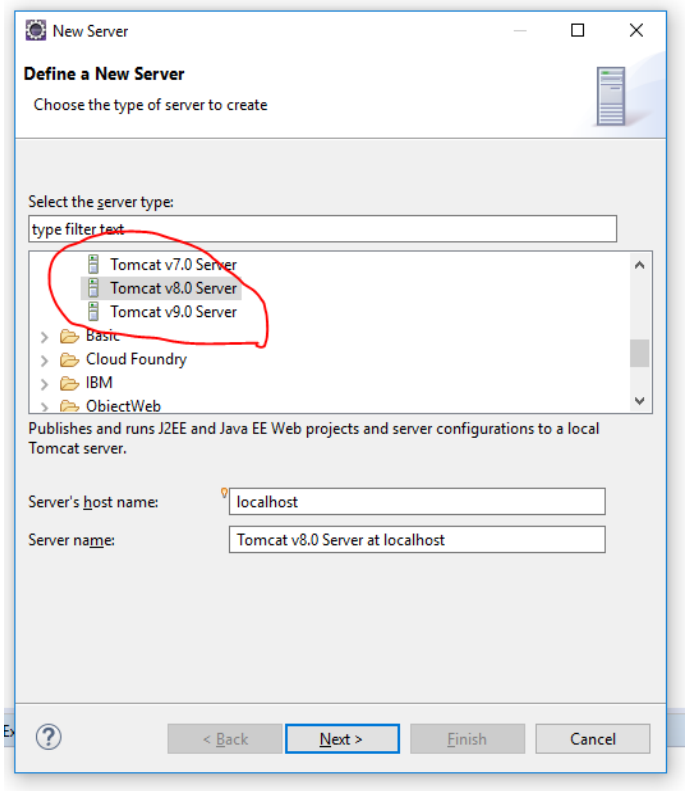


Figura 33 Eclipse versión servidor

El siguiente paso será indicar el directorio bin del servidor que se ha descargado en nuestro ordenador.

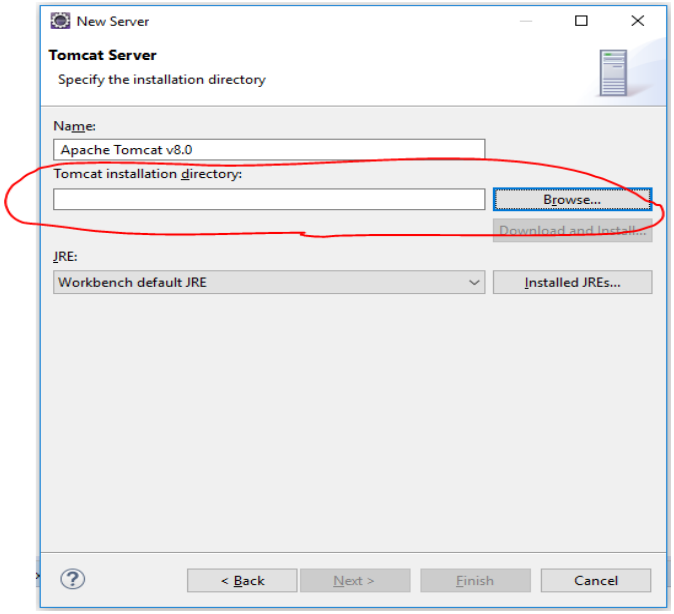


Figura 34 Directorio servidor Eclipse

Por último, se eligen las aplicaciones que vamos a desplegar en el servidor que acabamos de embeber en el IDE.

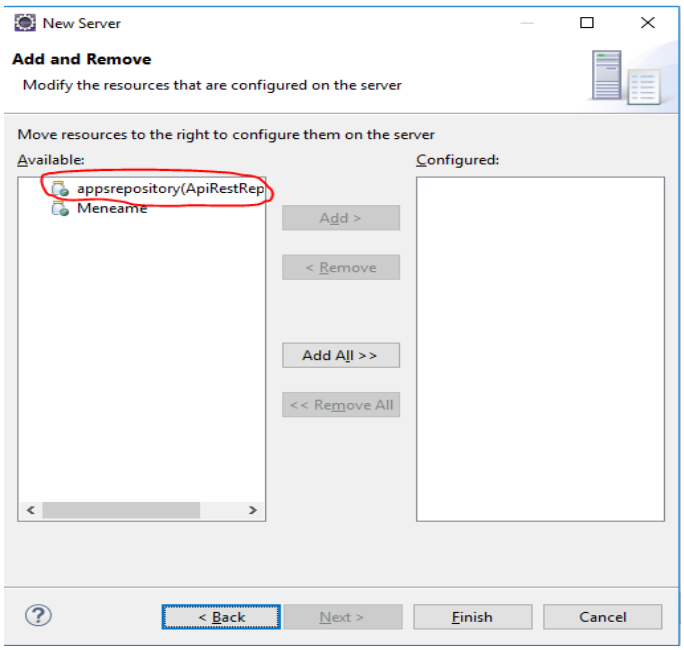


Figura 35 Aplicaciones Tomcat Eclipse

Ya estamos en condiciones de lanzar el servidor y probar las aplicaciones que hay desplegadas en él.

5.3.2. Librerías

En este apartado se van a listar y describir todas las librerías que se han utilizado en el proyecto Java para poder construir el servicio web API REST. En la figura 36 aparecen las librerías importadas en el proyecto.

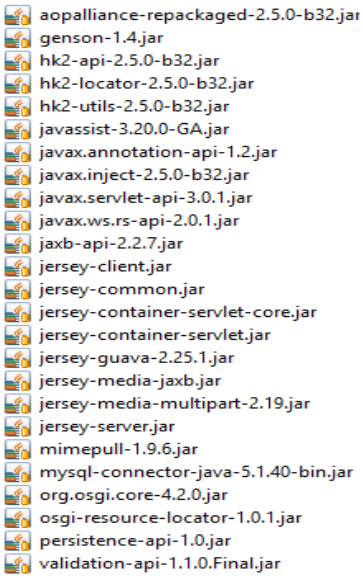


Figura 36 Librerías API REST

5.3.2.1. Jersey

Para implementar una API REST en Java hay principalmente dos formas. Utilizando un framework de Java llamado Spring o utilizar las librerías de jersey.

No puede decirse que la curva de aprendizaje de Spring sea suave debido a que introduce conceptos relativamente novedosos como son la AOP o la IoC y a la cantidad de configuración que hay que manejar [21]. Y teniendo en cuenta que en el grado he utilizado las librerías de jersey la decisión de utilizar esta librería está clara.

Para poder utilizar estas librerías simplemente vamos a la página oficial y las descargamos.

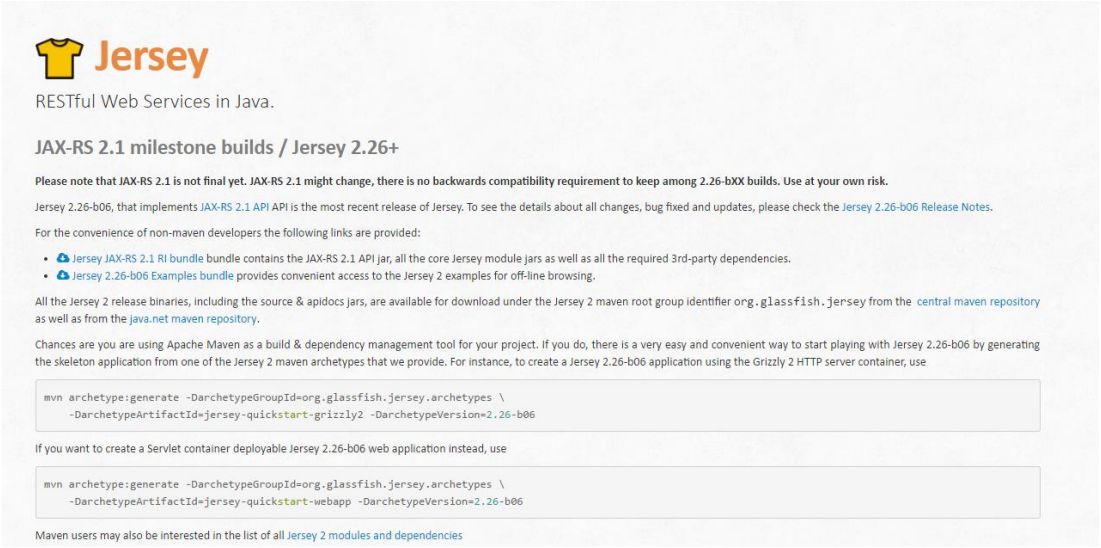


Figura 37 Página oficial jersey

Una vez descargadas las librerías las importamos en nuestro proyecto Java y ya podremos utilizar todas las anotaciones.

Veamos unos ejemplos de anotaciones que podemos usar con Jersey [22]:

- @GET: Indica que el método anotado corresponde a una petición HTTP GET.
- @POST: Indica que el método anotado corresponde a una petición HTTP POST.
- @HeaderParam: Enlaza una cabecera http al parámetro de un método.
- @HttpMethod: Asocia un método con el nombre de un método HTTP.

- @Path: Identifica la URI de una clase o método que sirve las peticiones.
- @ProduceMime: Define el/los tipo(s) MIME que los métodos producen.
- @QueryParam: Enlaza un parámetro de la petición HTTP con un parámetro del método java anotado.

5.3.2.2. Mysql-connector-java

Para que nuestro servicio web pueda consultar y actualizar información de la base de datos necesitamos tener una conexión a la misma. Como nuestra base de datos es MySQL necesitaremos utilizar esta librería.

```
@WebListener
public class ListenerApplication implements ServletContextListener {

    private static final Logger logger = Logger.getLogger(ServletContextListener.class.getName());
    //CONSTANTES PARA LA CONEXIÓN CON LA BASE DE DATOS
    private static final String HOST = "jdbc:mysql://db4free.net/appsrepository";
    private static final String USER = "javimelli8";
    private static final String PASSWORD = "corchado";

    public ListenerApplication() {

    }

    public void contextDestroyed(ServletContextEvent event)  {

    }

    public void contextInitialized(ServletContextEvent event)  {

        logger.info("-------------------------------------Creating DB-----------------------------------");
        Connection conn = null;

        //Cargamos el driver de mysql
        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
```

Figura 38 Conexión base de datos

En la figura 38 se puede observar cómo se utiliza esta librería para conectar con la base de datos.

5.3.2.3. Genson

El formato en el que recibirá y producirá información el servicio web será JSON. Para que java pueda parsear los datos que recibe y envía a formato JSON, se necesitará las funcionalidades que ofrece esta librería.

5.3.3. Estructura de la API REST

En este apartado se mostrará la estructura del proyecto de la API REST y se describirán los apartados más importantes. En la figura 39 se puede apreciar la estructura paquetes del proyecto Java en el IDE Eclipse.

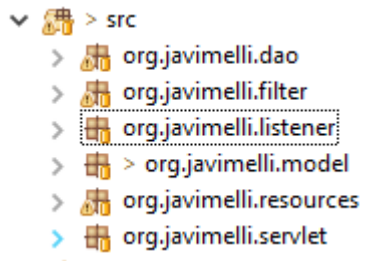


Figura 39 Paquetes Java API REST

5.3.3.1. Dao

En la figura 40 aparecen todos los ficheros que componen el paquete dao.

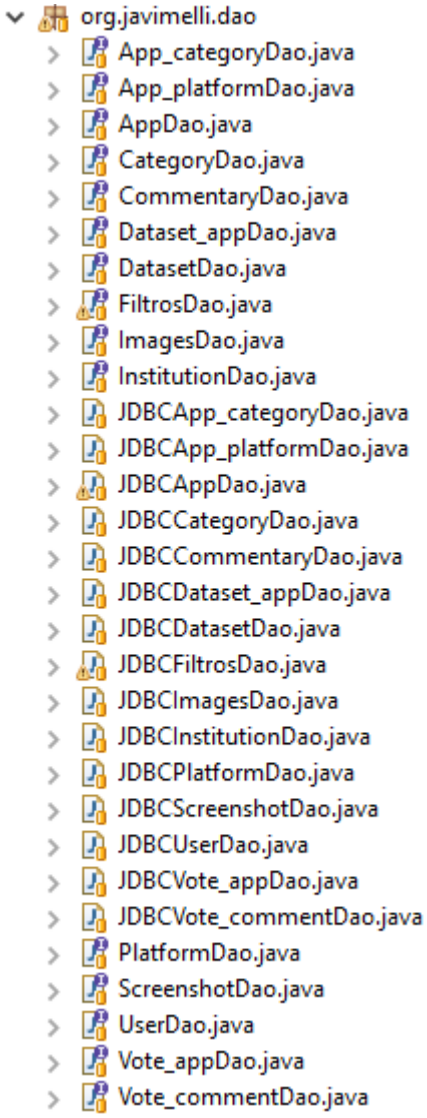


Figura 40 Ficheros Dao

En este paquete se encuentran todos los ficheros que implementan la comunicación con la base de datos. Se utiliza el patrón DAO como se ha comentado en capítulos anteriores.

Las interfaces contienen las cabeceras de los métodos y cada interfaz tiene una clase que implementa sus métodos, los cuáles construyen las consultas para poder obtener y actualizar la información de la base de datos.

5.3.3.2. Filter

El paquete filter contiene los ficheros que se muestran en la figura 41.

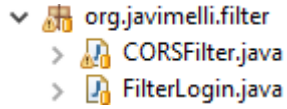


Figura 41 Ficheros paquete Filter

Estos ficheros implementan dos filtros que filtran todas las peticiones que recibe el servicio.

El FilterLogin controla las peticiones que pueden o no pueden pasar dependiendo de si el usuario es autenticado.

El CORSFilter añade parámetros a las cabeceras de las respuestas HTTP para poder controlar la política de CORS.

5.3.3.3. Listener

En la figura 42 se muestra el fichero que compone el paquete listener.

Figura 42 Fichero paquete Listener

Este fichero implemente un listener que se ejecuta al iniciar la aplicación. Se guarda la conexión de la base de datos en el contexto de la aplicación.

5.3.3.4. Model

En la figura 43 se pueden apreciar los ficheros que componen el paquete model.

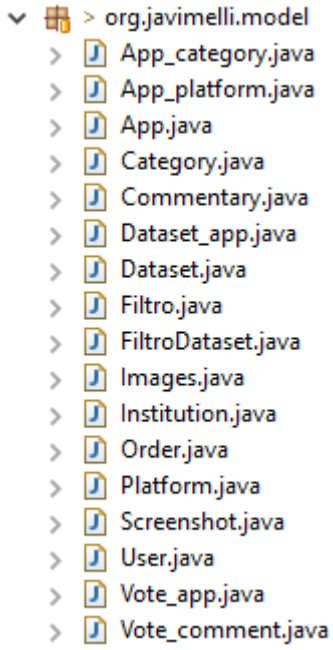


Figura 43 Ficheros paquete model

Estos ficheros contiene todas las clases donde se mapeará la información que proviene tanto de la base de datos como la que viene de peticiones HTTP de la aplicación cliente.

5.3.3.5. Resources

En la figura 44 se muestran los ficheros que componen el paquete resources.

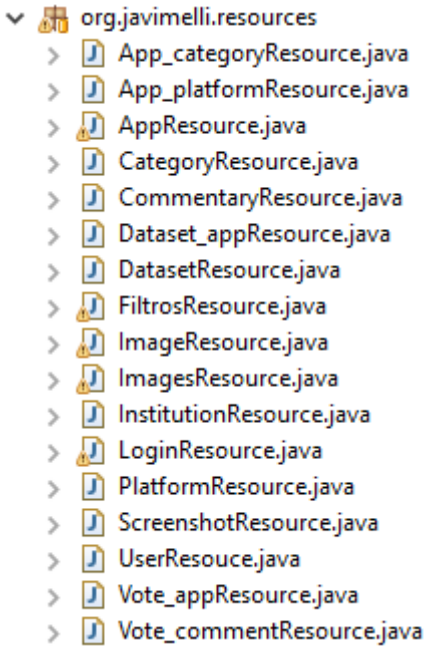


Figura 44 Ficheros que componen el paquete resources

En estas clases se implementan los métodos que se ejecutarán cuando se les invoque desde la aplicación cliente. Aquí es donde se usan las anotaciones de la librería jersey y se invocan los métodos del paquete dao para realizar las operaciones sobre la base de datos.

5.3.4. Implementación

En este apartado se va a comentar cuál ha sido el proceso que se ha seguido para la implementación de la API REST.

5.3.4.1. Configuración

La primera parte de la implementación se dedicó a crear la configuración que establece la conexión con la base de datos. Para ello se creó un listener que se ejecuta cada vez que arranque la aplicación.

```
@WebListener
public class ListenerApplication implements ServletContextListener {

    private static final Logger logger = Logger.getLogger(ServletContextListener.class.getName());
    //CONSTANTES PARA LA CONEXIÓN CON LA BASE DE DATOS
    private static final String HOST = "jdbc:mysql://db4free.net/appsrepository";
    private static final String USER = "javimelli8";
    private static final String PASSWORD = "corchado";
```

Figura 45 Constantes listener

En la figura 45 se muestran las constantes del listener. Estas contantes almacenan las credenciales para poder conectar con la base de datos. Se guardan en constantes para conseguir independencia y desacoplamiento en el código con el objetivo de que si se cambia de servidor la base de datos solo haya que modificar estos datos de conexión.

```
public void contextInitialized(ServletContextEvent event)  {

    logger.info("-------------------------------------Creating DB---------------------------------");
    Connection conn = null;

    //Cargamos el driver de mysql
    try {
        Class.forName("com.mysql.jdbc.Driver");
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }

    try{
        //Cogemos el contexto de la aplicación
        ServletContext sc = event.getServletContext();

        //Creamos la conecxión con los datos de la base de datos
        conn = DriverManager.getConnection(HOST,USER,PASSWORD);

        //Guardamos la conexión en el contexto de la aplicación
        sc.setAttribute("dbConn", conn);

        logger.info("-------------------------Base de datos creada-----------------------------");
    }catch(SQLException e) {
        e.printStackTrace();
    }
}
```

Figura 46 contextInitialized listener

En la figura 46 se muestra la implementación del método contextInitialized que se ejecutará al arrancar la aplicación. Este método lo que hace es registrar el diver de mysql y realizar una conexión pasándole las constantes con las credenciales de la figura 45.

Por último, dentro del try catch se guarda la conexión como atributo en el contexto de la aplicación con el nombre dbConn. Ahora esta conexión será accesible desde cualquier punto de la aplicación.

5.3.4.2. Filtros

En la aplicación se han implementado dos filtros. El CORSFilter.java y FilterLogin.java.

CORSFilter

La misión de este filtro es añadir parámetros a la cabecera HTTP de las repuestas que emite el servicio web. El objetivo es cumplir la política CORS de peticiones de origen cruzadas.

```
/**
 * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
 */
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {


    System.out.println("Pasando filtro");
    HttpServletResponse resp=(HttpServletResponse) response;
    resp.setHeader("Access-Control-Allow-Origin", "http://localhost:4200");
    resp.setHeader("Access-Control-Allow-Methods", "GET, POST, DELETE, PUT, OPTIONS, HEAD");
    resp.setHeader("Access-Control-Max-Age", "3600");
    resp.setHeader("Access-Control-Allow-Headers", "Origin, Accept, X-Requested-With, Content-Type, Access-Control-Request-Method, Access-Control-Request-Headers");
    resp.setHeader("Access-Control-Allow-Credentials", "true");
    // pass the request along the filter chain


    chain.doFilter(request, response);
}
```

Figura 47 Método doFilter del filtro CORSFIlter

Como vemos este método añade los siguientes parámetros a la respuesta que se emiten desde el servidor:

- Acces-ControlAllow-Origin: Este parámetro indica el dominio donde estará permitido realizar peticiones al servicio.
- Acces-Control-Allow-Methods: Indica los métodos que están permitidos en las peticiones.
- Acces-Control-Max-Age: Indica los segundos en los que la información que trae el paquete es válida.
- Acces-Control-Allow-Headers: Indica los parámetros de cabecera que estarán permitidos.
- Acces-Control-Allow-Credentials: Las cookies de credenciales y sesión estarán permitidas.

FilterLogin

El objetivo de este filtro es controlar el acceso a la información de los usuarios que están autenticados y los anónimos. Controla todas las peticiones que llegan al servicio como se puede ver en la figura 48.

```
/**
 * Servlet Filter implementation class FilterLogin
 */
@WebFilter(dispatcherTypes = {DispatcherType.REQUEST }
,           urlPatterns = { "/resources/*" })
public class FilterLogin implements Filter {
```

Figura 48 Dispacher FilterLogin

En la figura 49 se aprecia la implementación del método doFilter de este filter.

```
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
    // TODO Auto-generated method stub
    // place your code here

    HttpServletRequest req = (HttpServletRequest) request;
    HttpServletResponse res = (HttpServletResponse) response;

    HttpSession session = ((HttpServletRequest)request).getSession(true);
    logger.info("Comprobamos si hay sesion" + session.getId());//Si no hay sesion a logearse
    //chain.doFilter(request, response);

    if((req.getRequestURI().contains("/resources/apps")&&(req.getMethod().equalsIgnoreCase("GET"))) ||
            (req.getRequestURI().contains("/resources/apps")&&(req.getMethod().equalsIgnoreCase("DELETE"))) ||
            req.getRequestURI().contains("/ApiRestRepositorio/resources/session") ||
            req.getRequestURI().contains("/ApiRestRepositorio/resources/categorys") ||
            req.getRequestURI().contains("/ApiRestRepositorio/resources/images/id_fotos/") ||
            req.getMethod().equalsIgnoreCase("OPTIONS") ||
            req.getRequestURI().contains("/ApiRestRepositorio/resources/users") ||
            req.getRequestURI().contains("/ApiRestRepositorio/resources/filtros") ||
            req.getRequestURI().contains("/ApiRestRepositorio/resources/votes_apps/averageVotes/") ||
            req.getRequestURI().contains("/ApiRestRepositorio/resources/datasets") ||
            req.getRequestURI().contains("/ApiRestRepositorio/resources/institutions")){
            System.out.println("Entamos en dofilter" + req.getRequestURI());
            chain.doFilter(request, response);
    }else{
        System.out.println("Entamos en dofilter" + req.getRequestURI());
        if(session.getAttribute("user") == null){
            System.out.println("Entamos en login");
            res.setStatus(401);
        }
        else{
            System.out.println("Entamos en dofilter" + req.getRequestURI());
            chain.doFilter(request, response);
        }
    }
```

Figura 49 doFilter FilterLogin

Como vemos se dejan pasar algunas peticiones específicas. Todos son peticiones GET, lo que quiere decir que devolverán información, pero no la modificarán. En caso de que no sea una petición que se deja pasar siempre se comprueba si el usuario está autenticado, en caso afirmativo pasa el filtro y en caso negativo no pasa y se devuelve un estado HTTP 401 que indica que el usuario no está autenticado y se necesita autentificación para poder realizar esa operación.

5.3.4.3. Sesión

Una sesión [23] es un conjunto de interacciones que tienen lugar en su sitio web en un periodo determinado. Por ejemplo, una única sesión puede contener varias páginas vistas, eventos, interacciones sociales y transacciones de comercio

electrónico. Se puede decir que una sesión es el elemento que engloba las acciones del usuario en su sitio web.

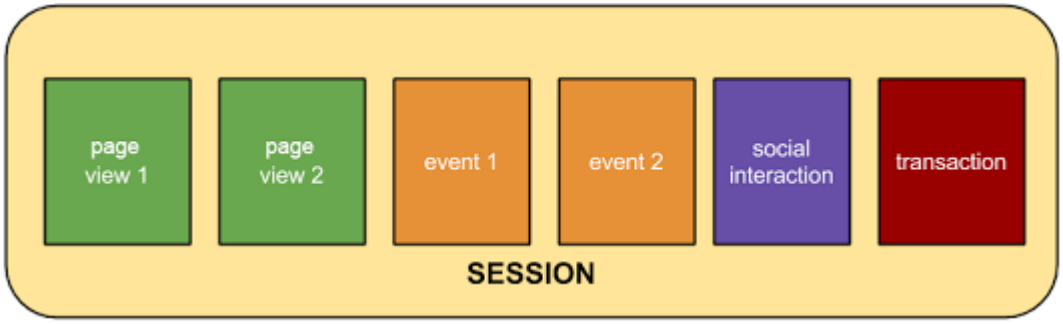


Figura 50 Acciones sesión

Un único usuario puede abrir varias sesiones, que pueden ocurrir el mismo día o en varios días, semanas o meses. En cuanto finaliza una sesión, es posible empezar una sesión nueva.

Para solventar este problema en la plataforma Java se usa de forma muy habitual la clase HttpSession [24] que tiene una estructura de HashMap (Diccionario) y permite almacenar cualquier tipo de objeto en ella.

El funcionamiento del sistema de sesiones es relativamente sencillo. Cada vez que un usuario crea una session accediendo a una página (que la genere) se crea un objeto a nivel de Servidor con un HashMap vacío que nos permite almacenar la información que necesitamos relativa a este usuario. Realizado este primer paso se envía al navegador del usuario una Cookie que sirve para identificarle y asociarle el HashMap que se acaba de construir para que pueda almacenar información en él. Este HashMap puede ser accedido desde cualquier otra página permitiéndonos compartir información. En la figura 51 se aprecia este proceso.

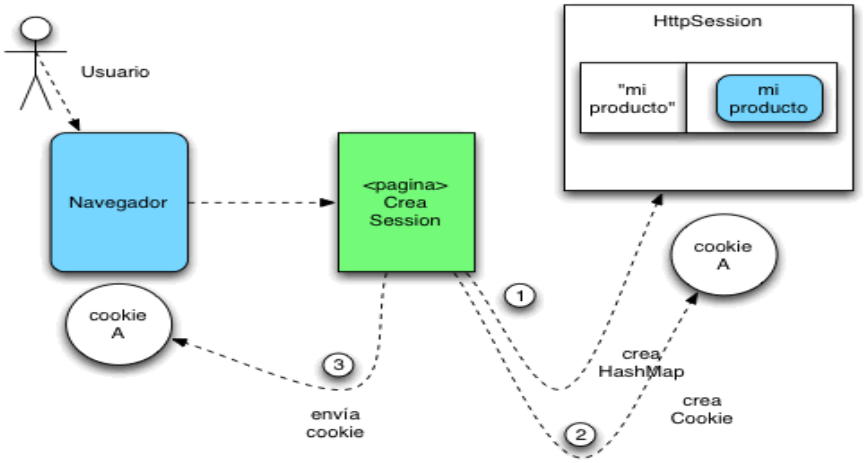


Figura 51 Sistema de cookies sesión

Cada sesión de usuario que se conecta a nuestra aplicación es individual y la información no es compartida entre ellos. Así pues cada usuario dispondrá de su propio HashMap en donde almacenar la información que resulte útil entre páginas.

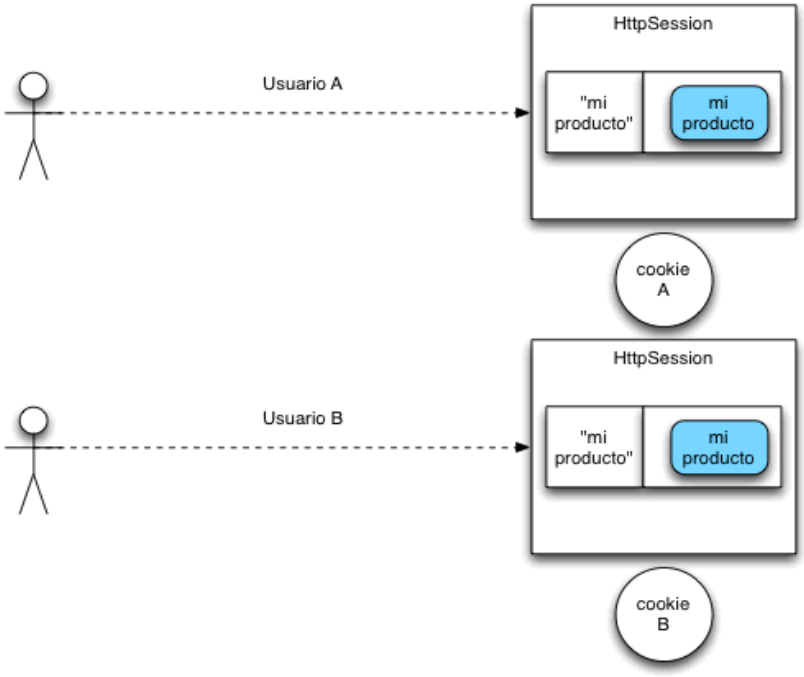


Figura 52 Usuarios y sesiones

En nuestra aplicación web se considera que la sesión de un usuario se abre cuando entra en la web por primera vez y se cerrará cuando cierren sesión mediante un logout o cuando transcurran 30 minutos, que es el tiempo en el que se cierra por defecto.

Una vez entendido lo que se quiere implementar y conseguir, pasamos a describir la implementación.

```
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
    // TODO Auto-generated method stub
    // place your code here

    HttpServletRequest req = (HttpServletRequest) request;
    HttpServletResponse res = (HttpServletResponse) response;

    HttpSession session = ((HttpServletRequest)request).getSession(true);
    logger.info("Comprobamos si hay sesion" + session.getId());//Si no hay sesion a logearse
    //chain.doFilter(request, response);
```

Figura 53 Inicio de sesión

En la figura 53 se aprecian algunas líneas de código del FilterLogin que se ha descrito en este mismo capítulo. Como se ha mencionado, todas las peticiones que se realicen al servicio pasarán por este filtro. Pues bien, la primera petición que pase por aquí abrirá la sesión. Para ello se crea un objeto de la clase HttpSession llamado session en el que se guarda la sesión que devuelve el método getSession(true) que al recibir true como parámetro lo que hará será abrir una sesión en caso de que no se hay creado aún y en caso de que exista la devuelve. Creando todo el proceso que se ha explicado y se aprecia en la figura 51.

Ahora en el objeto session podremos guardar información relacionada con las acciones de navegación que realiza el usuario en el portal.

Para cerrar sesión el usuario del portal ejecutará o realizará un evento en el cliente que provocará una petición al servicio web y que este procesará y provocará la ejecución del método que se puede apreciar en la figura 54. En este método se ejecuta el método getSession(false), pero en este caso se le pasa un boolean a false que lo que quiere decir es que este método en caso de que exista una sesión la cerrará.

```
@GET
@Path("/logout")
public boolean logout(@Context HttpServletRequest request){
    boolean logout = false;

    HttpSession session = request.getSession(false);
    if(session != null){
        session.invalidate();
        logout = true;
    }

    return logout;
}
```

Figura 54 Método logout

5.3.4.4. Autentificación

Como ya se ha comentado en este documento las acciones que un usuario podrá realizar en la aplicación dependerá de si está autenticado o si por el contrario es anónimo. Esto implica que se realice un control sobre las operaciones que un usuario puede realizar.

Este control se realiza utilizando el FilterLogin.java. Cuando llega una petición se comprueba si se solicita información accesible para todos los usuarios, en caso de ser así la petición pasará el filtro y el usuario podrá seguir con la operación que estaba realizando. Si se da el caso que la petición que viene de la aplicación cliente solicita información reservada para usuarios autenticados o solicita ejecutar funciones con modificaciones en la base de datos se comprueba si en el HashMap asociando al usuario existe un atributo llamado user que será un objeto que guarda la información del usuario autenticado. Este control realizado en el FilterLogin.java se puede apreciar en la figura 55.

```
if((req.getRequestURI().contains("/resources/apps")&&(req.getMethod().equalsIgnoreCase("GET"))) ||
        (req.getRequestURI().contains("/resources/apps")&&(req.getMethod().equalsIgnoreCase("DELETE"))) ||
        req.getRequestURI().contains("/ApiRestRepositorio/resources/session") ||
        req.getRequestURI().contains("/ApiRestRepositorio/resources/categorys") ||
        req.getRequestURI().contains("/ApiRestRepositorio/resources/images/id_fotos/") ||
        req.getMethod().equalsIgnoreCase("OPTIONS") ||
        req.getRequestURI().contains("/ApiRestRepositorio/resources/users") ||
        req.getRequestURI().contains("/ApiRestRepositorio/resources/filtros") ||
        req.getRequestURI().contains("/ApiRestRepositorio/resources/votes_apps/averageVotes/") ||
        req.getRequestURI().contains("/ApiRestRepositorio/resources/datasets") ||
        req.getRequestURI().contains("/ApiRestRepositorio/resources/institutions")){
        System.out.println("Entamos en dofilter" + req.getRequestURI());
        chain.doFilter(request, response);
}else{
    System.out.println("Entamos en dofilter" + req.getRequestURI());
    if(session.getAttribute("user") == null){
        System.out.println("Entamos en login");
        res.setStatus(401);
    }
    else{
        System.out.println("Entamos en dofilter" + req.getRequestURI());
        chain.doFilter(request, response);
    }
}
```

Figura 55 Control usuario autenticado

Para almacenar el atributo user en el HashMap de su sesión asociada, el usuario debe enviar una petición (que será provocada por alguna acción que realice el usuario en la aplicación cliente) al servicio web con sus credenciales. Esta petición ejecutará el método que aparece en la figura 56. Este método comprueba que el usuario existe en la base datos y si es así comprueba que las credenciales son correctas.

```
@POST
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public User starSession(User user, @Context HttpServletRequest request){

    User userSession = null;

    Connection conn = (Connection) sc.getAttribute("dbConn");
    UserDao userDao = new JDBCUserDao();
    userDao.setConnection(conn);

    User userDB = userDao.get(user.getUsername());

    //Compobamos que el usuario exite en la base de datos y evitamos un nullpointerexception a la hora
    //de acceder al username delk userDB
    if(userDB.getUsername() != null){
        if(userDB.getUsername().equals(user.getUsername()) && userDB.getPassword().equals(user.getPassword())){

            HttpSession session = (HttpSession) request.getSession();
            session.setAttribute("user",userDB);
            userSession = (User) session.getAttribute("user");
            userSession.setPassword("");
            System.out.println("Get ID Session: "+session.getId()+" User: "+userSession.getUsername()+" Password: "+userSession.getPassword());
        }
    }

    return userSession;
}
```

Figura 56 Método de autentificación

5.3.4.5. Subida de Imágenes

En la aplicación web, el usuario autenticado podrá subir imágenes al servidor ejecutando las siguientes operaciones:

- **Subir imagen de perfil**: Esta operación se puede realizar cuando el usuario se registre en el portal o una vez registrado y autenticado podrá sustituir la imagen de perfil.
- **Subir icono de aplicación**: Esta operación se podrá realizar cuando un usuario autenticado esté dando de alta o editando una aplicación.
- **Subir capturas de aplicación**: Esta operación se podrá realizar cuando un usuario esté dando de alta o editando una aplicación.

Por tanto, en el servidor se podrá realizar la subida de tres tipos de imágenes. Con el fin de poder identificar las imágenes con sus usuarios y con sus aplicaciones se ha creado el árbol de directorios que aparece en la figura 57.

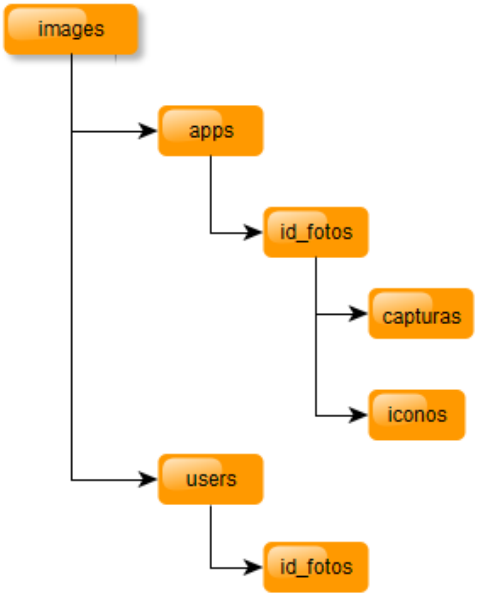


Figura 57 Árbol de directorios para las imágenes

Cuando un usuario realiza cualquiera de las operaciones que se han descrito, desde la aplicación cliente, esta realiza una petición al servicio web mediante una petición HTTP que ejecutará el método de la figura 58.

```
@POST
@Consumes(MediaType.MULTIPART_FORM_DATA)
public Response uploadFile(@FormDataParam("file") InputStream uploadedInputStream,
                @FormDataParam("file") FormDataContentDisposition fileDetail,
                                                @Context HttpServletRequest request
                ){
        String url = null;//String que enviaremos en la respuesta, para añadirle el domino del servidor
        String img = request.getParameter("img");
        String id_form = request.getParameter("id_form");
        String url_location = null;
        try{
                if(img.equals("user")){
                        url = this.FILE_LOCATION.concat("\\users\\" + id_form + "\\");
                        url_location = this.DIR_SERVER + url;
                }else{
                        if(img.equals("capture")){
                                url = this.FILE_LOCATION.concat("\\apps\\" + id_form + "\\capturas\\");
                                url_location = this.DIR_SERVER + url;
                        }else{
                                url = this.FILE_LOCATION.concat("\\apps\\" + id_form + "\\iconos\\");
                                url_location = this.DIR_SERVER + url;
                        }
                }
                File file = new File(url_location);
                if(file.mkdirs()){
                        System.out.println("Directorio creado");
                }
                //Creamos un fichero con un flujo de salida del programa con la location correcta
                FileOutputStream out = new FileOutputStream(url_location + fileDetail.getFileName());
                int read = 0;
                //Buffer para almacenar los Bytes leidos del InputStream pasado como parámetro
                byte[] buffer = new byte[1024];
                //Leemos los Bytes del InputStream pasado por parámetro de un MB en un MB hasta que termine  de leerse el fichero.
                while((read = uploadedInputStream.read(buffer)) != -1){
                        out.write(buffer, 0, read);
                }
                //Escribe los datos intermedios
                out.flush();
                //Cerramos definitivamente el flujo de salida
                out.close();
        }catch(IOException e){
                e.printStackTrace();
        }

        Response res;
        res = Response.ok(url + fileDetail.getFileName()).build();

        return res;
}
```

Figura 58 Método subida de imágenes al servidor

El método recibe un parámetro con el que se puede comparar si es un usuario, icono o captura. Dependiendo del valor del parámetro se va formando el path correcto para poder almacenar la imagen en el árbol de directorios que se ha explicado anteriormente.

Una vez generado el path se irá procesando la imagen mediante la apertura de un flujo de entrada sobre la imagen y un flujo de salida para almacenarlo en el path que se acaba de generar.

Por último, el mismo evento de subir una imagen al servidor provoca la ejecución del método de la figura 59.

```
@POST
@Consumes(MediaType.APPLICATION_JSON)
public Response postApp(Images img, @Context HttpServletRequest request){

    Response res;

    //Obtenemos la conexion a la base de datos del contexto de la aplicacion
            Connection conn = (Connection) sc.getAttribute("dbConn");
            ImagesDao imagesDao = new JDBCImagesDao();
            imagesDao.setConnection(conn);

    int id = 9999;

            id = imagesDao.addImages(img);

            //Creamos una respuesta
            res = Response //return 201 y la localizacion del nuevo recurso
                    .ok(Integer.toString(id))
                    .contentLocation(
                                    uriInfo
                                    .getAbsolutePathBuilder()
                                    .path(Integer.toString(id))
                                    .build())
                    .build();

            return res;
}
```

Figura 59 Método de almacenamiento de url de imagen

Este método guarda un registro en la entidad imagen de la base de datos almacenando el tipo de la imagen que se acaba de subir al servidor pudiendo ser user, icon o capture, almacenando la url generada y almacenando el id_fotos correspondiente al usuario o la aplicación.

5.3.4.6. Entidades

En este apartado se va a describir el proceso que se ha seguido para construir los métodos que atienden las peticiones correspondientes a las entidades que forman el sistema de información.

Para entender bien este proceso y que sea lo más claro posible se va a utilizar como ejemplo la entidad app.

La primera clase que se implementa es la clase en la que se van a mapear los datos de la base de datos, por tanto debe tener todos los atributos correspondientes a la entidad en la base de datos y sus getters y setters. En la figura 60 se puede apreciar los atributos de la clase app que se corresponde con la entidad app.

```
public class App {

    private int id;
    private int user_id;
    private String url_web;
    private String title;
    private String description;
    private String url_icon;
    private int price;
    private int version;
    private String url_video;
    private String language;
    private String country;
    private String id_fotos;
    private String date;
    private String time;
```

Figura 60 Atributos app

El siguiente paso es crear la clase que implementará los métodos en el que se construirán las consultas y operaciones SQL y que se ejecutarán en la base de datos, utilizando el patrón DAO.

Para ello en primer lugar, se crea una interfaz con las cabeceras de los métodos que van a utilizarse para construir las operaciones. En la figura 61 se muestra la interfaz correspondiente a la entidad app.

```
package org.javimelli.dao;

import java.sql.Connection;

public interface AppDao {

    public List<App> getAppsAll();
    public List<App> getAppsByLmit(int numRegs,int init);
    public List<App> getAppsByCountry(String country);
    public List<App> getAllByOwner(int owner);
    public App getById(int id);
    public int addApp(App app);
    public boolean save(App app);
    public boolean delete(int id);
    public void setConnection(Connection conn);
}
```

Figura 61 Interfaz DAO app

En la figura 62 se muestran las constantes de la clase que implementa la interfaz AppDao.

```
//CONSTANTES TABLAS
private static final String tblApp = "app";
//CONSTANTES ATRIBUTOS TABLA
private static final String atrId = "id";
private static final String atrUser_id = "user_id";
private static final String atrUrl_web = "url_web";
private static final String atrTitle = "title";
private static final String atrDescription = "description";
private static final String atrUrl_icon = "url_icon";
private static final String atrPrice = "price";
private static final String atrVersion = "version";
private static final String atrUrl_video = "url_video";
private static final String atrLanguage = "language";
private static final String atrCountry = "country";
private static final String atrId_fotos = "id_fotos";
private static final String atrDate = "date";
private static final String atrTime = "time";
```

Figura 62 Constantes de tabla y atributos

Estas constantes se utilizarán en los strings que guardan las instrucciones SQL. El objetivo es conseguir el máximo desacoplamiento de código posible y si se produce un refactor en algún nombre de tabla o atributos en las entidades de la base de datos solo debamos cambiar el valor de las constantes y no todas las instrucciones SQL a las que afecte el refactor.

En la figura 63 se muestra el método que devuelve todos los registros de la entidad app.

```
@Override
public List<App> getAppsAll() {

    if (conn == null) return null;

    ArrayList<App> apps = new ArrayList<App>();
    try {
        Statement stmt;
        ResultSet rs;
        synchronized(conn){
          stmt = conn.createStatement();
          String sql = "select * from "+tblApp;
          System.out.println(sql);
          rs = stmt.executeQuery(sql);
        }
        while ( rs.next() ) {
            App app = new App();
            app.setId(rs.getInt(atrId));
            app.setUser_id(rs.getInt(atrUser_id));
            app.setUrl_web(rs.getString(atrUrl_web));
            app.setTitle(rs.getString(atrTitle));
            app.setDescription(rs.getString(atrDescription));
            app.setUrl_icon(rs.getString(atrUrl_icon));
            app.setPrice(rs.getInt(atrPrice));
            app.setVersion(rs.getInt(atrVersion));
            app.setUrl_video(rs.getString(atrUrl_video));
            app.setLanguage(rs.getString(atrLanguage));
            app.setCountry(rs.getString(atrCountry));
            app.setId_fotos(rs.getString(atrId_fotos));
            app.setTime(rs.getString(atrTime));
            app.setDate(rs.getString(atrDate));

            apps.add(app);
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
    return apps;
}
```

Figura 63 método getAppsAll

Este método primero se crea un ArrayList de App donde se guardará el resultado de la consulta SQL ya mapeado a código Java. Utilizando la conexión a la base de datos que se guarda en el contexto de la aplicación durante la ejecución del listener, lazamos la consulta almacenada en el string sql, en el cual se utiliza la constante de tabla tblApp. El resultado se almacena en un Resultset que iremos recorriendo y por cada registro primero mapeamos este en una app y segundo guardamos esa app en el ArrayList que devuelve el método.

```java
@Override
public int addApp(App app) {

    int id=-1;
    if (conn != null){

        //CREACIÓN DE LA HORA Y LA FECHA
        Date date = new Date();
        System.out.println(date.getTime());
        DateFormat timeFormat = new SimpleDateFormat("HH:mm:ss");
        DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        System.out.println("Hora: "+timeFormat.format(date));
        System.out.println("Fecha: "+dateFormat.format(date));

        Statement stmt;
        try {
            stmt = conn.createStatement();
            String sql = "INSERT INTO "+tblApp +" ("+atrId+","+atrUser_id+","+atrUrl_web+","+atrTitle+","+atrDescription+","+atrUrl_icon+","
                    + ""+atrPrice+","+atrVersion+","+atrUrl_video+","+atrLanguage+","+atrCountry+","+atrId_fotos+","+atrTime+","+atrDate+") VALUES("
                    +app.getId()+","
                    +app.getUser_id()+",'"
                    +app.getUrl_web()+"','"
                    +app.getTitle()+"','"
                    +app.getDescription()+"','"
                    +app.getUrl_icon()+"',"
                    +app.getPrice()+","
                    +app.getVersion()+",'"
                    +app.getUrl_video()+"','"
                    +app.getLanguage()+"','"
                    +app.getCountry()+"','"
                    +app.getId_fotos()+"','"
                    +timeFormat.format(date)+"','"
                    +dateFormat.format(date)+"')";
            System.out.println(sql);
            stmt.executeUpdate(sql,Statement.RETURN_GENERATED_KEYS);

            ResultSet genKeys = stmt.getGeneratedKeys();

            if (genKeys.next())
                id = genKeys.getInt(1);

        } catch (SQLException e) {

            e.printStackTrace();
        }
    }

    return id;
}
```

Figura 64 Método que inserta app en la base de datos

En la figura 64 se puede observar el método que inserta una app en la base de datos.

Se recibe la app desde el cliente por parámetro y ya mapeada a un objeto Java. Este objeto contendrá todos los valores que se deben insertar en el registro de la entidad app.

En primer lugar, como la app al ser dada de alta guarda fecha y hora, se utilizan las funciones de Java getTime() y SimpleDateFormat() para obtener la fecha y la hora del sistema y darle un formato correcto para poder almacenarlo en sus campos correspondientes de MySQL sin problemas.

El siguiente paso es formar la sentencia SQL, utilizando las contantes de la clase para los nombres de atributos y tablas y la app pasada por parámetro para los valores del nuevo registro. Una vez formada la sentencia en un string este se ejecuta utilizando la conexión con la base de datos.

```
@Override
public boolean save(App app) {

    boolean save = false;
    if(conn != null){
        Statement stmt;

        try{
            stmt = conn.createStatement();
            String sql = "UPDATE "+ tblApp +" SET "+atrId+"="+app.getId()+","
                                            +atrUser_id+"="+app.getUser_id()+","
                                            +atrUrl_web+"='"+app.getUrl_web()+"',"
                                            +atrTitle+"='"+app.getTitle()+"',"
                                            +atrDescription+"='"+app.getDescription()+"',"
                                            +atrUrl_icon+"='"+app.getUrl_icon()+"',"
                                            +atrPrice+"="+app.getPrice()+","
                                            +atrVersion+"="+app.getVersion()+","
                                            +atrUrl_video+"='"+app.getUrl_video()+"',"
                                            +atrLanguage+"='"+app.getLanguage()+"',"
                                            +atrCountry+"='"+app.getCountry()+"'"
                                            +" WHERE "+atrId+"="+app.getId();
            System.out.println(sql);
            stmt.executeUpdate(sql);
            save = true;
        }catch(SQLException e){
            e.printStackTrace();
        }
    }
    return save;
}
```

Figura 65 Método que actualiza una app en la base de datos

El método de la figura 65 sigue el mismo procedimiento que el anterior, pero en este caso la sentencia que se forma en el string es un UPDATE para realizar una modificación en un registro de la entidad app.

```
@Override
public boolean delete(int id) {

    boolean done = false;
    if (conn != null){

        Statement stmt;
        try {
            stmt = conn.createStatement();
            String sql = "DELETE FROM "+tblApp+" WHERE "+atrId+"="+id;
            System.out.println(sql);
            stmt.executeUpdate(sql);
            done= true;
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    return done;
}

@Override
public void setConnection(Connection conn) {
    this.conn = conn;

}
```

Figura 66 Método que borra un registro de la base de datos

En el método que se muestra en la figura 66 se construye una sentencia SQL para borrar un registro de la entidad app. Para ello se utiliza en el WHERE de la sentencia DELETE el id pasado por parámetro al método que corresponde con la app que el usuario desea eliminar.

Una vez mostrado e implementado un CRUD de operaciones sobre la entidad app de nuestra base de datos el siguiente paso es crear la clase que implementa los métodos que se ejecutan cuando se realiza una petición desde el cliente y a su vez ejecuta los métodos con las sentencias SQL que acabamos de describir.

En esta nueva clase es donde utilizamos las anotaciones de la librería jersey para poder crear los distintos path que ejecutan los métodos y para indicar que tipo de recursos consumen o producen en caso de hacerlo.

```java
@Path("/apps")
public class AppResource {

    @Context//Obtenemos el contexto de la aplicacion
    ServletContext sc;
    @Context//Obtenemos el URI completo de la solicitud
    UriInfo uriInfo;
```

Figura 67 Path de una clase jersey

En la figura 67 se aprecia la declaración de la clase AppResource y sus atributos. Lo primero que hay que hacer es para cada clase de este tipo diferenciar unas de otras con un path diferente. En este caso para ejecutar cualquier método de esta clase el path o url de la petición HTTP del cliente debe contener localhost:8080/appsrepository/resources/**apps**.

```java
@GET
@Produces(MediaType.APPLICATION_JSON)
public List<App> getApps() {

    //Obtenemos la conexion a la base de datos del contexto de la aplicacion
    Connection conn = (Connection) sc.getAttribute("dbConn");
    AppDao appDao = new JDBCAppDao();
    appDao.setConnection(conn);

    List<App> listApp = appDao.getAppsAll();

    return listApp;
}
```

Figura 68 Método que atiende una petición GET

El método de la figura 68 se ejecutará si el servicio recibe una petición con la url localhost:8080/appsrepository/resources/apps y utiliza el método GET. Como vemos lo que hace es invocar al método que hemos descrito en este mismo apartado y que aparece en la figura 63 y devuelve las apps en formato JSON al cliente.

```
@POST
@Consumes(MediaType.APPLICATION_JSON)
public Response postApp(App app, @Context HttpServletRequest request){

    Response res;

    //Obtenemos la conexion a la base de datos del contexto de la aplicacion
    Connection conn = (Connection) sc.getAttribute("dbConn");
    AppDao appDao = new JDBCAppDao();
    appDao.setConnection(conn);

    HttpSession session = (HttpSession) request.getSession();
    User user = (User) session.getAttribute("user");

    int id = 9999;

    if(user != null){
        app.setUser_id(user.getId());
        id = appDao.addApp(app);
    }



    //Creamos una respuesta
    res = Response //return 201 y la localizacion del nuevo recurso
        .ok(Integer.toString(id))
        .contentLocation(
                uriInfo
                .getAbsolutePathBuilder()
                .path(Integer.toString(id))
                .build())
        .build();

    return res;
}
```

Figura 69 método que atiende una petición POST

El método de la figura 69 se ejecutará si recibe una petición con la url localhost:8080/appsrepository/resources/apps y solicita el método POST. En este caso recibe información en formato JSON que es mapeada a un objeto Java y que se le pasa al método que hemos descrito en este apartado y que aparece en la figura 64.

```
@PUT
@Path("/{appId: [0-9]+}")
@Consumes(MediaType.APPLICATION_JSON)
public Response putUser(@PathParam("appId") int id, App appEdit,  @Context HttpServletRequest request){

    Response res = null;

    Connection conn = (Connection) sc.getAttribute("dbConn");
    AppDao appDao = new JDBCAppDao();
    appDao.setConnection(conn);

    HttpSession session = (HttpSession) request.getSession();
    User user = (User) session.getAttribute("user");

    boolean save = false;

    if(user != null){

        App app = appDao.getById(appEdit.getId());

        if(app == null){
            //Lazamos EXCEPTION
        }else{
            if(app.getId() == appEdit.getId()){
                save = appDao.save(appEdit);
            }
            else{
                //LANZAMOS EXCEPTION DE ID
            }
        }
    }

    res = Response.ok(save).build();

    return res;
}
```

Figura 70 Método que se ejecuta con una petición PUT

El método de la figura 70 realiza la misma operación que el anterior, pero en este caso con el método PUT e invocando su método correspondiente que aparece en la figura 65.

```
@DELETE
@Path("/{appId: [0-9]+}")
public void deleteUserId(@PathParam("appId") int id){

    Connection conn = (Connection) sc.getAttribute("dbConn");
    AppDao appDao = new JDBCAppDao();
    appDao.setConnection(conn);

    App app = appDao.getById(id);

    if(app == null){
        //Lazamos EXCEPTION
    }else{
        appDao.delete(id);
    }
}
```

Figura 71 Método que se ejecuta con una petición DELETE

El método de la figura 71 se ejecuta ante una petición con la url localhost:8080/appsrepository/resources/apps y utilizando el método DELETE. Este

invoca el método de la figura 66 para que este ejecuta la sentencia SQL DELETE como ya hemos explicado en este apartado.

Se puede dar la circunstancia de que en una misma clase necesitemos ejecutar métodos que solicitan el mismo método HTTP, por ejemplo GET. En este caso es tan sencillo como añadirle la etiqueta de jersey @Path y construir una url distinta y única para la petición que provoca la ejecución del método. Veamos el ejemplo en la figura 72.

```
@GET
@Path("/country/{country}")
@Produces(MediaType.APPLICATION_JSON)
public List<App> getAppByCountry(@PathParam("country") String country,  @Context HttpServletRequest request){

    Connection conn = (Connection) sc.getAttribute("dbConn");
    AppDao appDao = new JDBCAppDao();
    appDao.setConnection(conn);

    List<App> app = null;
    app = appDao.getAppsByCountry(country);

    return app;
}
```

Figura 72 Método con @Path

Todo este proceso que se ha explicado en este apartado se ha llevado a cabo con todas las entidades que componen el diseño de la base de datos y se han implementado todas las funcionalidades que se han ido necesitando, pero siempre siguiendo este proceso paso a paso y utilizando el patrón DAO.

5.3.4.7. Enrutamiento de métodos

En este apartado se muestra la tabla 33 que contiene las urls de las peticiones del cliente para ejecutar los distintos métodos de la API REST.

Tabla 33 Enrutamiento ENDPOINT

PATH	**MÉTODO**
apps/	GET
/apps/country/{country}	GET
/apps/owner/{appId}	GET
/apps/limit/{num}	GET
/apps/{appId}	GET

/apps/	POST
/apps/	PUT
/apps/	DELETE
/apps_categorys/{appId}	GET
/apps_category/category/{categoryId}	GET
/apps_category/categoryId/país/{country}	GET
/apps_categorys/	POST
/apps_categorys/{id}	PUT
/apps_categorys/{id}	DELETE
/apps_platforms/app/{appId}	GET
/apps_platforms/platform/{platformId}	GET
/apps_platforms/	POST
/apps_platforms/{id}	PUT
/apps_platforms/{id}	DELETE
/categorys/	GET
/categorys/{categoryId}	GET
/commentarys/	GET
/commentarys/owner/{ownerId}	GET
/commentarys/app/appId	GET
/commentarys/{commentaryId}	GET
/commentarys/	POST
/commentarys/{id}	PUT
/commentarys/{id}	DELETE
/datasets_apps/dataset/{datasetId}	GET
/datasets_apps/app/{appId}	GET

/datasets_apps/	POST
/datasets_apps/{id}	PUT
/datasets_apps/	DELETE
/datasets_apps/{appId}	DELETE
/datasets/	GET
/datasets/{datasetId}	GET
/datasets/user/{datasetId}	GET
/datasets/category/{datasetId}	GET
/datasets/institution/{datasetId}	GET
/datasets/	POST
/datasets/{id}	PUT
/datasets/{id}	DELETE
/filtros/{busq}	GET
/filtros/dataset/{busq}	GET
/filtros/	POST
/filtros/datasets	POST
/image/	POST
/image/	DELETE
/images/	GET
/images/id_fotos/{Id_fotos}	GET
/images/	POST
/images/{id}	PUT
/images/{id}	DELETE
/institutions/	GET
/institutions/{institutionsId}	GET

/institutions/	POST
/institutions/{id}	PUT
/institutions/{id}	DELETE
/session/	GET
/session/logout	GET
/session/	POST
/platforms/	GET
/platforms/{platformId}	GET
/screenshots/{screenshotId}	GET
/screenshots/app/{appId}	GET
/screenshots/	POST
/screenshots/	DELETE
/users/{userId}	GET
/users/{username}	GET
/users/	GET
/users/	POST
/users/{id}	PUT
/users/{id}	DELETE
/votes_apps/user/{userId}	GET
/votes_apps/app/{appId}	GET
/votes_apps/{userId}/{appId}	GET
/votes_apps/countComplaint/{appId}	GET
/votes_apps/averageVotes/{appId}	GET
/votes_apps/	POST
/votes_apps/	PUT

/votes_apps/	DELETE
/votes_comments/user/{userId}	GET
/votes_comments/comment/{commentId}	GET
/votes_comments/{userId}/{commentId}	GET
/votes_comments/countComplaint/{commentId}	GET
/votes_comments/countVotePositive/{commentId}	GET
/votes_comments/countVoteNegative/{commentId}	GET
/votes_comments/{id}	POST
/votes_comments/{id	PUT
/votes_comments/	DELETE

5.3.4.8. Filtro de búsqueda

En la aplicación web se va a implementar un buscador avanzado de aplicaciones y un buscador avanzados de datasets.

Para el buscador de aplicaciones los **filtros** con los que dispondrá el usuario serán los siguientes:

- Categoría
- País

El listado podrá ser ordenado por:

- Visitas
- Media de votaciones
- Precio

Para el buscador de datasets los filtros de los que dispondrá el usuario serán:

- Categoría
- País
- Ciudad

Para el filtro de búsqueda se han implementado dos métodos. Uno que formará la consulta para el listado de apps y otro que formará la consulta para el listado de datasets.

```
@Override
public List<App> filtrosApp(Filtro filtros) {

    List<App> apps = new ArrayList<App>();

    String atributos = "SELECT *";
    String tables = " FROM " + tblApp + " a ";
    String where = "";
    String orderBy = "";
    String limit = "LIMIT " + filtros.getInit() + ", " + filtros.getNum();

    boolean hayWhere = false;

    //Recorremos los filtros
    if(filtros.getCategoria() != 9999){
        tables += "INNER JOIN "+tblApp_category+" appc ON a."+atrId+"=appc."+atrtblApp_categoryApp_id+" ";
        where += "WHERE appc."+atrtblApp_categoryCategory_id+"="+filtros.getCategoria()+" ";
        hayWhere = true;
    }

    if(!filtros.getPais().equals("")){
        if(hayWhere)
            where += "AND a.country='"+filtros.getPais()+"' ";
        else
            where += "WHERE a.country='"+filtros.getPais()+"' ";
    }

    //Recorremos los orders
    if(filtros.isVistas()){
        orderBy = "ORDER BY visits "+filtros.getOrder()+" ";
    }

    if(filtros.isPrecio()){
        orderBy = "ORDER BY price "+filtros.getOrder()+" ";
    }
    if(filtros.isMedia()){
        atributos += ", (SELECT AVG("+atrvote_commentValue+") FROM vote_app WHERE app_id=a.id) as average";
        orderBy += "ORDER BY average "+filtros.getOrder()+" ";
    }

    String sql = null;
    if(hayWhere)
        sql = atributos + tables + where + orderBy;
    else
        sql = atributos + tables + where + orderBy + limit;

    if(conn != null){
        Statement stmt;

        try{
            stmt = conn.createStatement();
            //String sql = "SELECT * FROM "+tblApp+ " LIMIT " + init + ", " + numRegs;
            System.out.println(sql);
            ResultSet rs = stmt.executeQuery(sql);
```

Figura 73 Método filtro apps

El método de la figura 73 recibe un parámetro de la clase Filtros que es una clase con los filtros por los que un usuario puede filtrar apps. En la figura 74 se muestran los atributos de la clase Filtros.

```java
public class Filtro {

    private int categoria;
    private String pais;
    private boolean vistas;
    private boolean precio;
    private boolean media;
    private String order;
    private int num;
    private int init;
```

Figura 74 Atributos Filtros

Según los valores que traiga la clase Filtros pasado por parámetro al método de la figura 73 y mediante comprobaciones generará la consulta que se ajusta a todos los parámetros introducidos por el usuario y devolverá las apps devueltas al ejecutar la consulta.

```java
@Override
public List<Dataset> filtrosDataset(FiltroDataset filtros) {

    if (conn == null) return null;

    List<Dataset> datasets = new ArrayList<Dataset>();

    String atributos = "SELECT *";
    String tables = " FROM " + tblDataset + " d ";
    String where = "";

    boolean hayWhere = false;
    boolean hayJoin = false;

    if(filtros.getCategoria() != 9999){
        where += "WHERE d." + atrDatasetCategory_id +"="+filtros.getCategoria();
        hayWhere = true;
    }

    if(!filtros.getCiudad().equals("")){
        tables += " INNER JOIN " + tblInstitution + " i ON d." + atrDatasetInstitution + "=i." + atrInstitutionId + " ";
        hayJoin = true;
        if(hayWhere){
            where += " AND i." + atrInstitutionCity +" = '" + filtros.getCiudad() + "'";
        }else{
            where += "WHERE i." + atrInstitutionCity +" = '" + filtros.getCiudad() + "'";
        }
        hayWhere = true;
    }

    if(!filtros.getPais().equals("")){
        if(!hayJoin){
            hayJoin = true;
            tables += " INNER JOIN " + tblInstitution + " i ON d." + atrDatasetInstitution + "=i." + atrInstitutionId + " ";
        }
        if(hayWhere){
            where += " AND i." + atrInstitutionCountry + " = '" + filtros.getPais() + "'";
        }else{
            where += "WHERE i." + atrInstitutionCountry + " = '" + filtros.getPais() + "'";
        }
    }

    String sql = atributos + tables + where;
    System.out.println(sql);

    try {
        Statement stmt;
        ResultSet rs;
        synchronized(conn){
          stmt = conn.createStatement();
          System.out.println(sql);
          rs = stmt.executeQuery(sql);
        }
```

Figura 75 Método que filtra datasets

El método de la figura 75 sigue el mismo proceso que el de la figura 73 pero basándose en los filtros que un usuario puede usar para filtras datasets. En la figura 76 se muestran los atributos de la clase FiltrosDatasets que son los que se les pasa por parámetro a esta consulta y en los que se basará por tanto, para construir la consulta que devuelva los datasets de la base de datos que se ajusten con los parámetros.

```
public class FiltroDataset {

    private int categoria;
    private String pais;
    private String ciudad;
```

Figura 76 Atributos filtros dataset

Para ambos buscadores se ha implementado un buscador de palabras qué realizará la búsqueda tanto en las descripciones como en los títulos de aplicaciones y datasets. Para su implementación se ha utilizado una sentencia de MySQL llamada MATCH AGAINST.

Esta instrucción nos permitirá realizar búsquedas sobre el índice definido al igual que lo haríamos en google. Es muy útil ya que nos evita tener que programarlo o adaptar frames en nuestro sitio.

Para poder utilizar esta sentencia sobre los campos description y title de ambas entidades, primero se tuvo que crear un Índice FULL TEXT sobre los campos. Para crearlo se hay seguido el código de la figura 77.

```
MySQL
1 CREATE FULLTEXT INDEX nombre_indice ON nombre_tabla(columna [columna2...]);
```

Figura 77 Index fulltext

```
@Override
public List<App> BusqApp(String busq) {

    List<App> listApps = new ArrayList<App>();

    if(conn != null){
        Statement stmt;

        try{
            stmt = conn.createStatement();
            String sql = "SELECT * FROM "+tblApp+" WHERE MATCH("+atrTitle+","+atrDescription+") AGAINST('"+busq+"')";
            System.out.println(sql);
            ResultSet rs = stmt.executeQuery(sql);
            while(rs.next()){
                App app = new App();
                app.setId(rs.getInt(atrId));
                app.setUser_id(rs.getInt(atrUser_id));
                app.setUrl_web(rs.getString(atrUrl_web));
                app.setTitle(rs.getString(atrTitle));
                app.setDescription(rs.getString(atrDescription));
                app.setUrl_icon(rs.getString(atrUrl_icon));
                app.setPrice(rs.getInt(atrPrice));
                app.setVersion(rs.getInt(atrVersion));
                app.setUrl_video(rs.getString(atrUrl_video));
                app.setLanguage(rs.getString(atrLanguage));
                app.setCountry(rs.getString(atrCountry));
                app.setId_fotos(rs.getString(atrId_fotos));

                listApps.add(app);
            }
        } catch (SQLException e){
            e.printStackTrace();
        }
    }

    return listApps;
}
```

Figura 78 Método MATCH AGAINST aplicaciones

En el método e la figura 78 se implementa el buscador por palabras de aplicaciones. Como se puede apreciar recibe como parámetro un string que es utilizado en la consulta con el MATCH AGAINST.

```
@Override
public List<Dataset> BusqDataset(String busq) {

    if (conn == null) return null;

    ArrayList<Dataset> datasets = new ArrayList<Dataset>();
    try {
        Statement stmt;
        ResultSet rs;
        synchronized(conn){
          stmt = conn.createStatement();
          String sql = "select * from "+tblDataset+" WHERE MATCH("+atrTitle+","+atrDescription+") AGAINST('"+busq+"')";
          System.out.println(sql);
          rs = stmt.executeQuery(sql);
        }
        while ( rs.next() ) {
            Dataset dataset = new Dataset();
            dataset.setId(rs.getInt(atrDatasetId));
            dataset.setCategory_id(rs.getInt(atrDatasetCategory_id));
            dataset.setUser_id(rs.getInt(atrDatasetUser_id));
            dataset.setDescription(rs.getString(atrDatasetDescription));
            dataset.setUri_dataset(rs.getString(atrDatasetUri_dataset));
            dataset.setInstitution_id(rs.getInt(atrDatasetInstitution));
            dataset.setTitle(rs.getString(atrDatasetTitle));

            datasets.add(dataset);
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
    return datasets;
}
```

Figura 79 Método MAATCH AGAINST datasets

El método de la figura 79 sigue el mismo proceso que el de la figura 78, pero utilizando la constante de la tabla de datasets y almacenando el resultado en un ArrayList de datasets.

5.4. Desarrollo de la parte del cliente (front-end)

En este apartado se explica todo lo relacionado con la fase de implementación de la parte que se ejecutará en el navegador web del cliente. Esta parte de la aplicación web se descargará en el cliente en una primera petición que realice al servidor que contiene los archivos de esta parte. Una vez descargado los archivos, la aplicación cliente irá realizando peticiones al servicio web como ya se ha comentado en capítulos anteriores.

5.4.1. Tecnología

En este apartado se listarán y explicarán las tecnologías que se han utilizado para la construcción de la aplicación cliente. Como se ha comentado en el apartado 4.4 del capítulo 4 análisis y diseño de este documento, la arquitectura de la aplicación web será SPA. Para construir la parte cliente de este tipo de aplicaciones existen varios frameworks de javascript. Veamos algunos de ellos.

Figura 80 Backbone.js

- **Backbone.js**, este pequeño framework permite construir aplicaciones usando Javascript siguiendo el patrón MVC (modelo-vista-controlador).

Figura 81 Ember

- **Ember.js** es un framework JavaScript para crear aplicaciones web del lado del cliente (código abierto). Está basado en la arquitectura modelo-vista-controlador.

Figura 82 React

- **React.js** React es una biblioteca escrita en JavaScript, desarrollada en Facebook para facilitar la creación de componentes interactivos, reutilizables, para interfaces de usuario. Se utiliza en Facebook para la producción de componentes, e Instagram está escrito enteramente en React [25].

Figura 83 AngularJS

- **AngularJS** es sin duda el framework de javascript MCV más popular del mercado y creado por Google lo que asegura un gran éxito sin ninguna duda.

En la web existe infinidad de documentación y tutoriales relacionada con este framework.

Figura 84 Angular 2

- Angular 2 es la segunda versión de AngularJS. Es un framework para desarrollo SPA que permite extender el HTML con etiquetas propias. La interfaz está basada en componentes y se recomienda usar TypeScript.

La elección de que framework utilizar puede parecer sencilla en un principio porque como he comentado en capítulos anteriores AngularJS no solo es que sea el framework más asentado y conocido, además en la asignatura de programación en internet que se imparte en este grado, se realizó una aplicación utilizando este framework, por tanto tengo conocimientos bastantes sólidos.

Sin embargo, Al buscar información sobre angular 2 nos entró bastante curiosidad por aprender este framework. Han salido bastantes versiones betas antes de que en Septiembre de 2016 saliera la primera versión estable de este framework. Por lo que se ha podido investigar tiene bastantes diferencias con su predecesor AngularJS y mejora muchas funcionalidades de este.

Teniendo en cuenta que uno de los objetivos principales por los que se decidió realizar este proyecto era formarse en el desarrollo web, realizarlo con Angular 2 implicaría aumentar conocimientos en esta materia, aunque utilizar este framework implique un aumento de horas invertidas en el proyecto, ya que antes de utilizarlo se necesita tiempo para aprenderlo.

El tutor del TFG en este aspecto dio total libertad para elegir que framework utilizar, con el único requisito de que las tecnologías que se utilicen en el proyecto deben ser gratuitas.

Finalmente el framework elegido para construir la parte cliente es Angular2. Los motivos por los que se ha elegido son los siguientes:

- Como ya se ha comentado es un crecimiento importante en el desarrollo de un programador web.
- Se prevé que este framework será el más utilizado a corto y medio plazo.
- Mejora muchas funcionalidades de su framework predecesor AngularJS.
- El soporte que se le dará en los próximos años será bastante importante. Esto implicará que la fase de mantenimiento de la aplicación web sea más fácil y llevadera.

Algunas desventajas que implica desarrollar con este framework son:

- Como se ha comentado implica horas extra para poder adquirir los conocimientos necesarios para poder desarrollar con él.
- Es un framework bastante nuevo, por tanto no hay tanta documentación en la web como puede existir ahora mismo sobre AngularJS.

5.4.1.1. Angular2

Una aplicación Angular 2 se desarrolla a base de crear componentes. Generalmente tendrás un árbol de componentes que forman tu aplicación y cada persona lo podrá organizar de su manera preferida. Siempre existirá un componente padre y a partir de ahí podrán colgar todas las ramas que sean necesarias para crear tu aplicación [26].

El párrafo anterior describe la característica más nueva e importante de Angular2. En este apartado comentaremos algunos de los conceptos más importantes de este framework.

Controlador: Será el encargado de implementar la lógica de negocio del componente y de enlazar el controlador con la vista.

Un controlador siempre debe llevar el código que se aprecia en la figura 85. El selector indica cuál es el nombre que se utilizará en el código HTML para utilizar este componente. En este caso su etiqueta será <proyecto-angular2-app></proyecto-angular2-app>. El templateUrl indica el fichero que contiene las etiquetas propias HTML que se mostrará donde utilizamos la etiqueta del controlador y con las que

interactuará, se puede ver en la figura 86 de este documento. Por último el styleUrls indica el fichero que contendrá el código CSS con el que se dará estilo a la vista.

```
@Component({
  moduleId: module.id,
  selector: 'proyecto-angular2-app',
  templateUrl: 'proyecto-angular2.component.html',
  styleUrls: ['proyecto-angular2.component.css']
})
```

Figura 85 Component Angular2

```
<h1>
  {{title}}
</h1>
<p [hidden]="!visible">
  Adiós
</p>
<button (click)="decirAdios()">Decir adiós</button>
```

Figura 86 Vista controlador Angular2

Un controlador como se ha comentado implementará la lógica de negocio relacionada con la vista. Para ello a parte del código que ya hemos explicado, un controlador implementa una clase que envuelve toda la lógica de negocio correspondiente. Siguiendo con este ejemplo en la figura 87 se aprecia la lógica de negocio del componente que estamos usando como ejemplo.

```
export class ProyectoAngular2AppComponent {
  title = 'Manual de Angular 2 de DesarrolloWeb.com';
  visible = false;
  decirAdios() {
    this.visible = true;
  }
}
```

Figura 87 Clase Angular 2

Aunque en javascript no existen clases porque no es un lenguaje orientado a objetos y no es un lenguaje tipado, ya hemos comentado en este capítulo que Angular2 utiliza TypeScript.

El navegador no puede ejecutar código TypeScript directamente porque no lo entiende, solo entiende javascript. Angular2 realiza una transpilación del código TypeScript y lo convierte a javascript que si lo puede ejecutar el navegador del cliente. Por tanto todo el código javascript que se escriba en los métodos de la clase del controlador también será válido.

Angular2 está modularizado y necesita una configuración inicial bastante extensa. Existen herramientas que facilitan este trabajo y te ayudan a crear un proyecto esqueleto ya configurado. La herramienta oficial de angular y la que se ha utilizado para esta labor es **Angular-cli**.

Angular CLI nos ahorrará escribir mucho código y nos permitirá partir de un esquema de aplicación avanzado y capaz de facilitar los flujos de desarrollo, depuración, testing o deploy [26].

Es una herramienta NodeJS, es decir, para poder instalarla necesitaremos contar con NodeJS instalado en nuestro sistema operativo, algo que podemos conseguir muy fácilmente yendo a la página oficial de node.js y descargando el instalador para nuestro sistema.

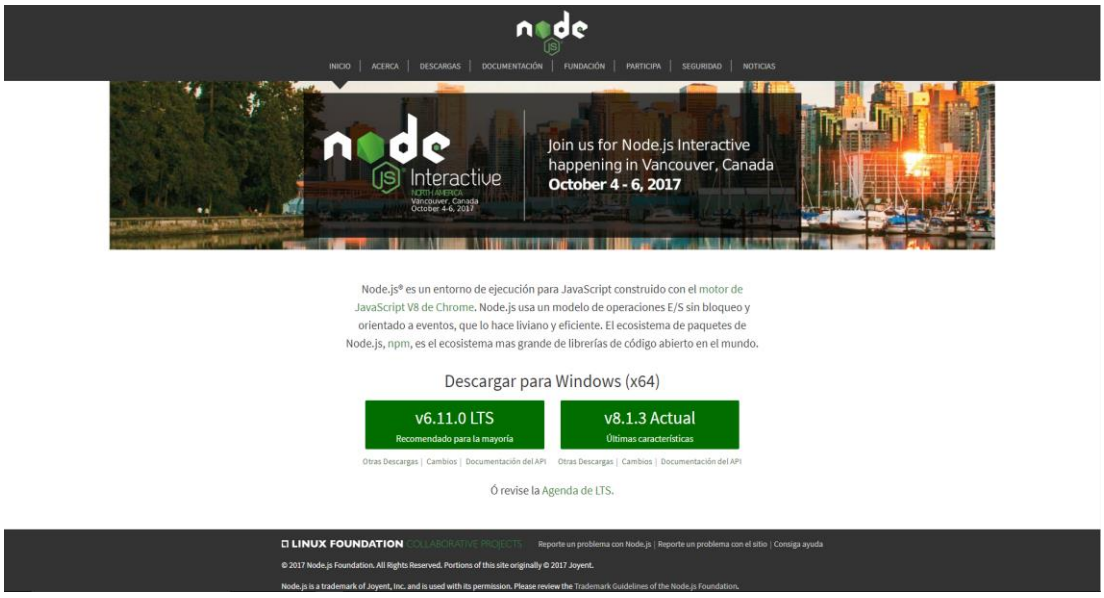


Figura 88 Página NodeJS

Una vez instalado NodeJS ya podremos usar su gestor de paquetes npm para instalar Angular-cli. Nos situamos en la línea de comando y ejecutamos el comando de la figura 89.

```
npm install -g angular-cli
```

Figura 89 Comando angular-cli

Para crear el esqueleto del proyecto Angular2, después de instalar Angular-cli debemos ejecutar el comando de la figura 90.

```
ng new mi-nuevo-proyecto-angular2
```

Figura 90 Comando proyecto Angular2

Ya estaríamos en condiciones de poder trabajar con nuestro proyecto Angular2.

Para poder ejecutar la aplicación, Angular-cli proporciona un comando en el que lanza un servidor con retroalimentación para poder ver los cambios durante el desarrollo. Para lanzarlo simplemente desde la línea de comandos navegamos hasta el directorio del proyecto y ejecutamos el comando **ng serve**. Una vez ejecutado, comenzará la transpilación del código TypeScript. Terminada la transpilación si no hay errores podremos ejecutar la aplicación en el navegador web utilizando localhost:4200. 4200 es el puerto por defecto, se puede cambiar si el usuario lo necesita.

5.4.1.2. Sublime Text 2

Existen varios IDEs y editores de texto con los que poder desarrollar en Angular2 usando TypeScript. Algunos IDEs como Eclipse tienen plugins para TypeScrit y Angular2. En la figura 91 podemos ver algunos IDEs y editores que podemos usar.

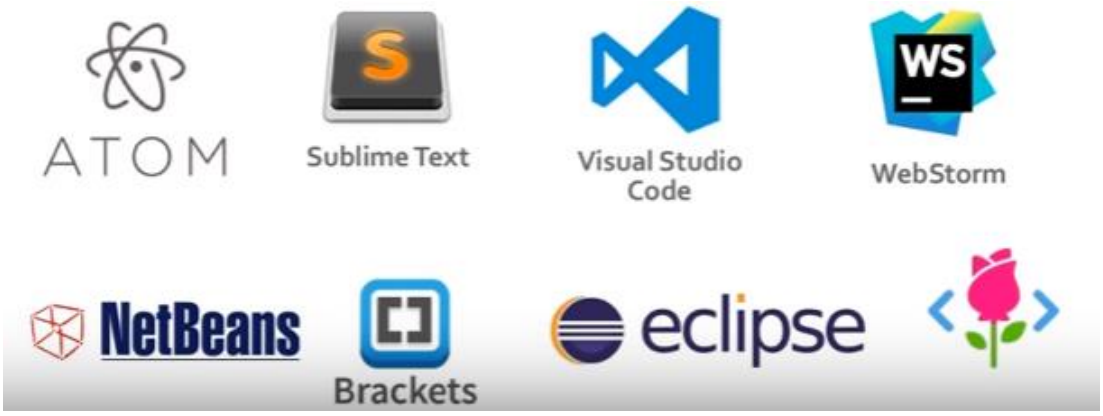


Figura 91 Editores e IDEs Angular2

La elección de usar un editor o IDE concreto no afecta al rendimiento de la aplicación, simplemente es cuestión de comodidad y gustos del desarrollador. En mi caso he elegido Sublime Text 2 para desarrollar esta parte de la aplicación.

5.4.1.3. Bootstrap

Con el objetivo de seguir ampliando los conocimientos sobre el desarrollo web, se ha decido usar uno de los frameworks CSS más utilizados y solicitados. Veamos algunas de las ventajas y desventajas de este framework [28].

Ventajas:

- Cuenta con un mantenimiento y actualización realizados por Twitter: esto no quiere decir que esta herramienta sea perfecta, pero gran parte del trabajo interno ya está llevado a cabo por sus creadores.
- Ofrece un paquete de elementos web personalizables: con Bootstrap puedes diseñar una web jugando con sus elementos compuestos por diferentes combinaciones de HTML, CSS y Javascript, de manera que las piezas siempre encajan.
- Se integra con librerías JavaScript.
- Es una herramienta de uso ágil y sencillo: facilita enormemente el diseño de interfaces y además incluye por defecto una plantilla bastante optimizada.
- Contiene tutoriales: este framework facilita mucha documentación para resolver dudas tanto a principiantes como a desarrolladores expertos.

Desventajas:

- es necesario adaptarse a su forma de trabajo, si bien su curva de aprendizaje es liviana, deberás comprender y familiarizarte con su estructura y nomenclatura. Esto implica horas extras en el proyecto como el caso de aprender Angular2.
- debes adaptar tu diseño a un grid de 12 columnas, que se modifican según el dispositivo. aquí empiezan los problemas, bootstrap por defecto te trae anchos, márgenes y altos de línea, y realizar cambios específicos es por decir, un poco tedioso.

5.4.2. Estructura de la aplicación cliente

En este apartado se va a describir la estructura de la parte del proyecto que se descarga y ejecuta en el cliente. Como se ha descrito en apartados anteriores hemos usado la herramienta Angular-cli para generar un proyecto esqueleto de Angular2. En la figura 92 se aprecia la estructura de la aplicación cliente.

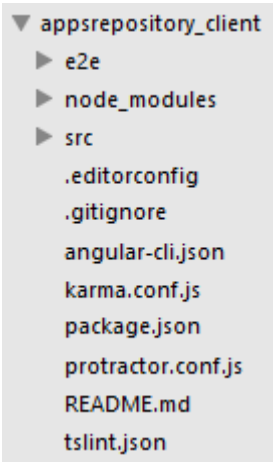


Figura 92 Estructura aplicación cliente

- El directorio **e2e** contiene los ficheros y directorios necesarios para poder realizar testing sobre proyecto. Su contenido se aprecia en la figura 93.

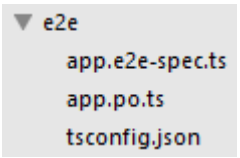


Figura 93 Ficheros e2e

- En **node_modules** se guardan todos los módulos que se van a utilizar en el proyecto. Si queremos descargar algún nuevo módulo se descargará en este directorio. En la figura 94 se aprecian algunos de los módulos que contiene este directorio.

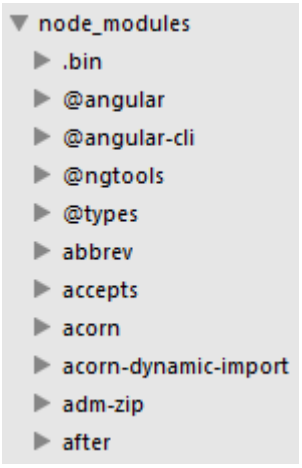


Figura 94 Directorio node_modules

- El directorio **src** contiene todos los ficheros y directorios que implementan los componentes de la aplicación.

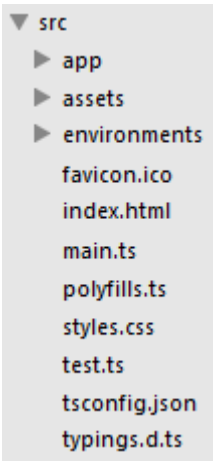


Figura 95 Directorio src

Este es el directorio más importante, es donde crearemos los componentes de la aplicación. El index.html es el fichero en el cual se inicia la aplicación y se invoca el componente raíz. El contenido de este fichero se puede ver en la figura 96. Como se ve contiene los link a los ficheros externos como por ejemplo el CDN de la librería de Bootstrap. Si nos fijamos dentro de la etiqueta body del html hace referencia al componente raíz de la aplicación que es app-root. Dentro del html del componente raíz existirán etiquetas que hacen referencia a otros componentes de la aplicación.

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>AppsOpenData</title>
    <base href="/">

    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" type="image/x-icon" href="assets/favIcon.png">

  <link href="" rel="stylesheet">
  <link href="https://cdnjs.cloudflare.com/ajax/libs/ng2-bootstrap/1.6.0/ng2-bootstrap.umd.min.js" rel="stylesheet">
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">

  <script src="http://code.jquery.com/jquery-3.1.1.min.js"></script>

</head>
<body>
  <app-root>Loading...</app-root>
</body>
</html>
```

Figura 96 index.html

El contenido del directorio app es el que se muestra en la figura 97. Cada directorio contiene los ficheros correspondientes a un componente. Como hemos explicado antes son el controlador, la vista y el fichero de estilos. Además hay un directorio llamado service que contiene los servicios que implementan métodos comunes a varios componentes. Los ficheros que corresponden al componente raíz están en este directorio.

Figura 97 Directorio app

Otro directorio importante dentro del directorio app es assets que contiene todos los recursos estáticos como imágenes o iconos que usa la aplicación. Su contenido se aprecia en la figura 98.

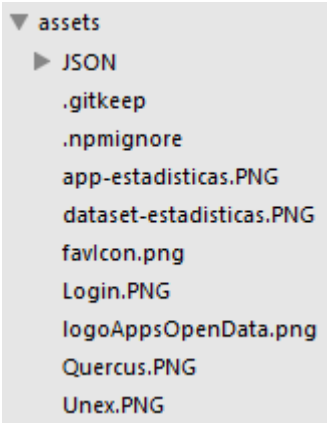


Figura 98 Directorio assets

Los demás ficheros tanto del directorio src como del directorio app son ficheros de configuración generados por Angular-cli y que no se han tenido que modificar durante el desarrollo.

5.4.3. Implementación

En este apartado se describirá la implementación de la aplicación cliente. Como se ha comentado en primer lugar, se realizó un proyecto esqueleto y a partir de él se empezó a trabajar. Como ya se ha explicado en este documento, Angular2 se construye a partir de componentes. En la figura 99 se muestra el árbol de componentes que se ha construido.

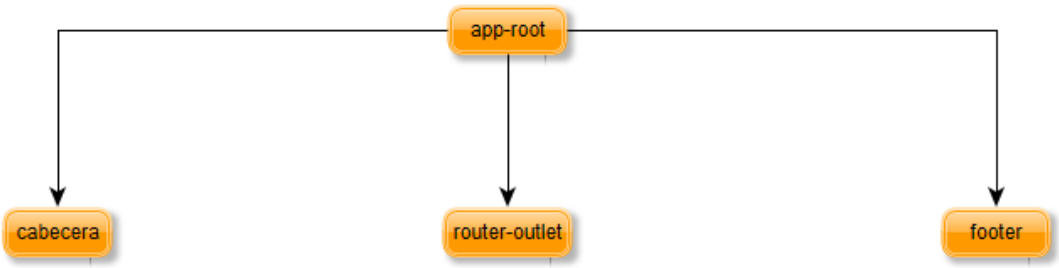


Figura 99 Árbol de componentes

Como vemos la aplicación parte de un componente raíz llamado app-root y que ha sido generado por Angular-cli. A partir de este componente vamos llamando y construyendo la aplicación.

5.4.3.1. Creación de componentes

A partir del componente app-root debemos crear nuestros propios componentes y enlazarlos y utilizarlos en la aplicación. Para ello se deben modificar varios archivos. Para la creación de los componentes en primer lugar, se debe ejecutar el comando ng m g nombreNuevoModulo. Este comando ejecutado en el directorio del proyecto generará un directorio con el contenido de la figura 100.

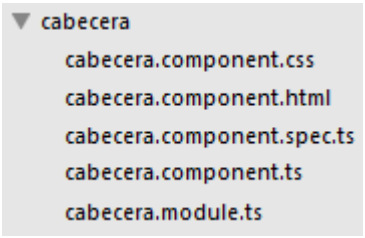


Figura 100 Módulo cabecera

En este caso de ejemplo estamos mostrando el proceso seguido con el componente cabecera, pero es el mismo proceso seguido con todos los componentes. Como se puede apreciar en la figura 100 se crean los ficheros correspondientes al controlador (cabecera.component.ts), vista (cabecera.component.html) y estilos (cabecera.component.css). El resto de ficheros no se han utilizado.

Una vez creado hay que configurar la aplicación para que reconozca este componente y pueda ser usado por otros componentes. Para ello abrimos el fichero app.module.ts que es el descriptor de despliegue de la aplicación e importamos el controlador en este caso el cabecera.component.ts.

```
import { CabeceraComponent } from './cabecera/cabecera.component';
```

Figura 101 import cabecera

Como vemos en la figura 101 el nombre con el que se importa es el nombre de la clase del controlador (CabeceraComponent).

El siguiente paso es añadirlo al apartado declarations del NgModule como se muestra en la figura 102.

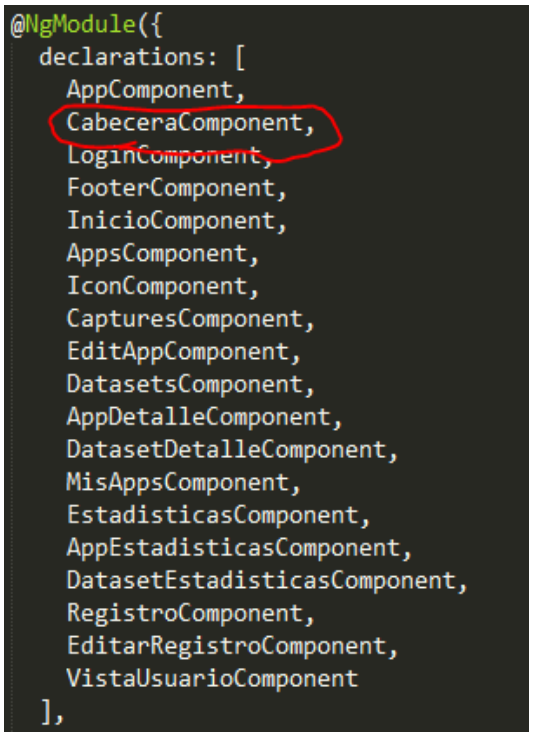


Figura 102 Declarations cabecera

Una vez realizado este proceso ya se podrá usar el componente cabecera en cualquier lugar de la aplicación. Este proceso se ha seguido para construir todos los componentes de la aplicación.

5.4.3.2. Router-outlet

Como se puede observar en la figura 99 de este capítulo, el árbol de componentes tiene un componente llamado router-outlet. No es un componente generado por nosotros utilizando el proceso que se ha explicado en el apartado anterior.

Como ya se ha explicado una aplicación SPA no depende de la URL para cargar una página nueva completa. Este componente irá cambiando su contenido dependiendo de la URL sin cambiar la página completa. Esto se puede hacer gracias a al routing de Angular2.

Para poder implementarlo se debe crear un fichero llamado app.routing.ts. En el definiremos las distintas rutas de la aplicación y el componente que ocupará la vista con esa URL. En la figura 103 se puede ver las urls y los componentes de nuestra aplicación.

```
const appRoutes = [
    { path: '', component: InicioComponent, },
    { path: 'login', component: LoginComponent, },
    { path: 'app', component: AppsComponent,},
    { path: 'app/:id', component: EditAppComponent,},
    { path: 'app-detalle/:id', component: AppDetalleComponent,},
    { path: 'dataset-detalle/:id', component: DatasetDetalleComponent,},
    { path: 'mis-apps', component: MisAppsComponent,},
    { path: 'datasets', component: DatasetsComponent,},
    { path: 'estadisticas', component: EstadisticasComponent,},
    { path: 'app-estadisticas', component: AppEstadisticasComponent,},
    { path: 'dataset-estadisticas', component: DatasetEstadisticasComponent,},
    { path: 'registro', component: RegistroComponent,},
    { path: 'editar-perfil', component: EditarRegistroComponent,},
    { path: 'vista-usuario/:id', component: VistaUsuarioComponent,}
]
```

Figura 103 Rputing aplicación

Veamos algunos ejemplos de cómo funciona el routing de angular. En la figura 104 la URL es localhost:4200. Lo que quiere decir que se corresponde con la URL / y si nos fijamos en la figura 103 la URL / se corresponde con el componente InicioComponent.

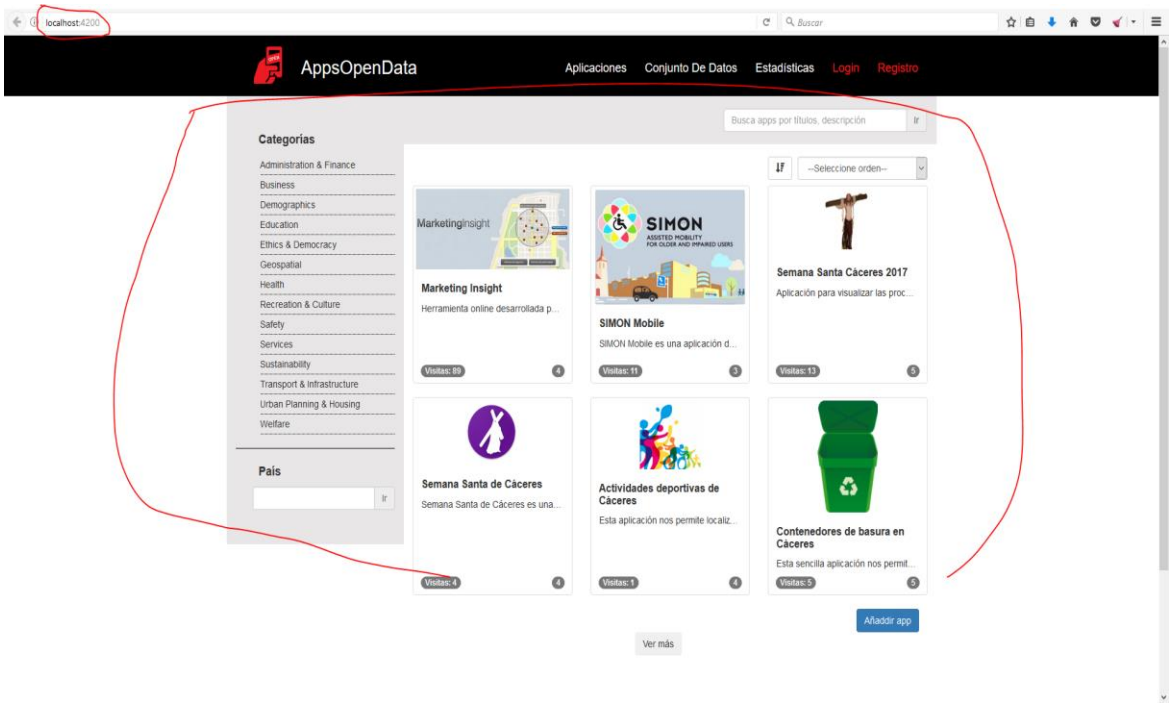


Figura 104 Página principal

Por tanto, en esta situación con la URL / el árbol de componentes quedaría como se puede apreciar en la figura 105.

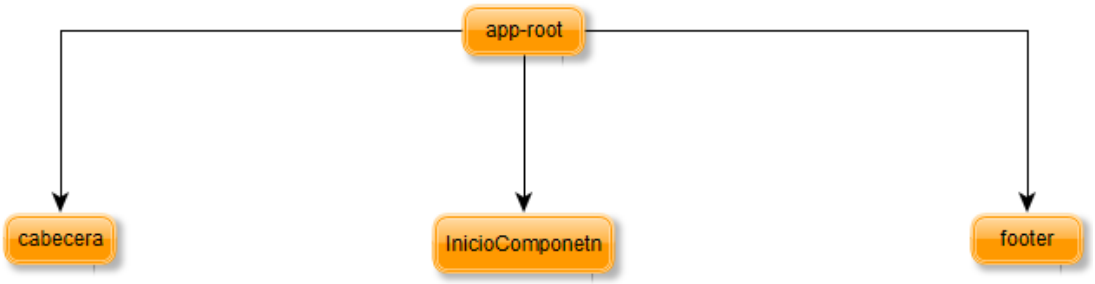


Figura 105 InicoComponent

Este sería el proceso de navegación de la aplicación. Como vemos en la figura 106 la aplicación se divide en el componente cabecera, este componente siempre va a estar, el componente route-outlet que cambiará según la URL, en este ejemplo está la URL / y el componente InicioComponent, y por último el componente footer que siempre estará al igual que cabecera.

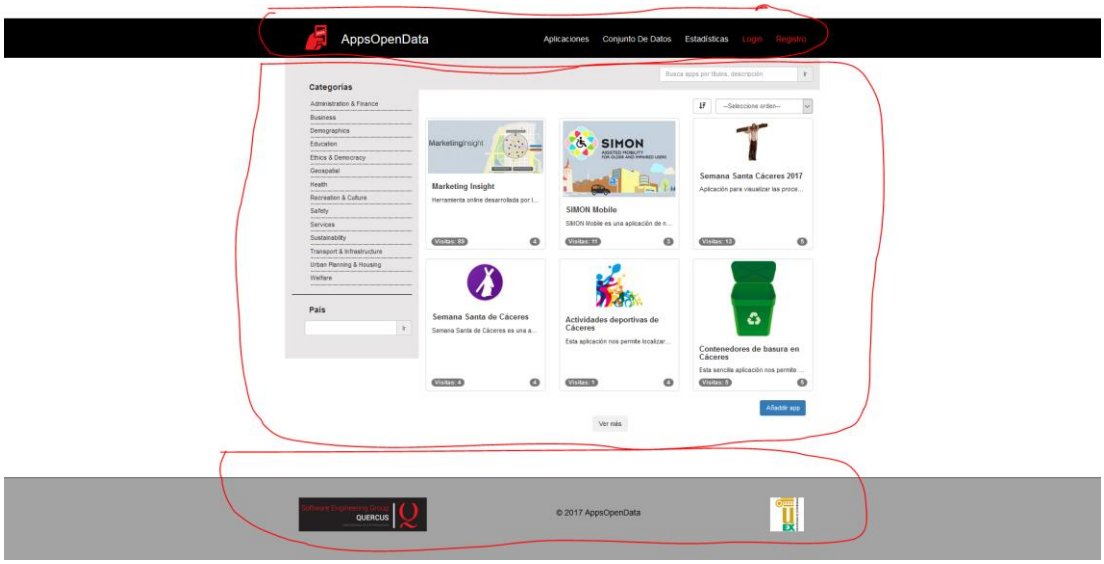


Figura 106 División componentes

Para que un usuario cambie de vista deberá navegar por la aplicación y para crear un enlace que cambie la URL y cargue su componente correspondiente se debe utilizar el código que se muestra en la figura 107. No se utiliza el atributo href, se debe utilizar el elemento routerLink indicando la URL que se corresponda con el componente que queremos que se cargue en cada caso. En este ejemplo y si nos fijamos en la figura 103 el componente que se cargará si el usuario pincha en este enlace es LoginComponent.

```
<li *ngIf="login"><a [routerLink]="['/login']" class="enlace_login navbar-brand elemento_cabecera">Login</a></li>
```

Figura 107 routerLink

5.4.3.3. Servicios

Los servicios en Angular2 son clases que se podrán instanciar y ejecutar sus métodos en cualquier controlador de un componente de la aplicación.

Según los desarrolladores de Angular 2 es buena práctica separar el código que realiza una petición AJAX usando HTTP de los controladores de componentes, y así se ha implementado en nuestra aplicación. Por cada clase existente en el paquete resources (figura 44, apartado 5.3.3.5 de este documento) del servicio web se ha creado un servicio que implementará los métodos que realizarán peticiones a los métodos de la API REST.

Se va a explicar el proceso que se ha seguido para crear los servicios de la aplicación. Para ello vamos a utilizar como referencia el app.service.ts.

Se ha creado un directorio llamado service dentro del directorio app como ya hemos descrito en este capítulo. En este directorio creamos un fichero en el cual declaramos una clase como se puede apreciar en la figura 108.

```
export class AppService {
```

Figura 108 Declaración de clase

En esta clase creamos una constante llamada urlBase que se corresponde con el path de la clase del servicio web a la que queremos hacer peticiones como se puede ver en la figura 109

```
private urlBase:string = 'http://localhost:8080/ApiRestRepositorio/resources/apps';
```

Figura 109 urlBase

Para realizar peticiones HTTP Angular2 dispone de su propio cliente HTTP. Para ello deberemos primero importar el módulo en el fichero (figura 110) y declarar un atributo en el constructor de la clase utilizando este módulo como se aprecia en la figura 111.

```
import { Http, Response, Headers, Request, RequestOptions } from '@angular/http';
```

Figura 110 Importación del cliente HTTP

```
constructor(private http:Http){}
```

Figura 111 Atributo HTTP

El siguiente paso será implementar los métodos que realizarán las peticiones AJAX al servicio web. En la figura 112 se pueden ver los métodos que implementan este servicio.

```
getApps(){
    console.log('Ejecutando servicio getApps');
    let headers = new Headers();
    let options = new RequestOptions({ headers: headers, withCredentials: true});
    return this.http.get(this.urlBase,options);
}

getAppsById(id){
    console.log('Ejecutando servicio getApps');
    let url = this.urlBase + "/" + id;
    let headers = new Headers();
    let options = new RequestOptions({ headers: headers, withCredentials: true});
    return this.http.get(url,options);
}

getAppsByOwner(id){
    console.log('Ejecutando servicio getAppsByOwner');
    let url = this.urlBase + "/owner/" + id;
    let headers = new Headers();
    let options = new RequestOptions({ headers: headers, withCredentials: true});
    return this.http.get(url,options);
}

getAppsByLimit(num, init){
    console.log('Ejecutando servicio getAppsByCountry');
    let url = this.urlBase + "/limit/" + num + "/" + init;
    let headers = new Headers();
    let options = new RequestOptions({ headers: headers, withCredentials: true});
    return this.http.get(url,options);
}

getAppsByCountry(country){
    console.log('Ejecutando servicio getAppsByCountry');
    let url = this.urlBase + "/country/" + country;
    let headers = new Headers();
    let options = new RequestOptions({ headers: headers, withCredentials: true});
    return this.http.get(url,options);
}

putApp(app,id){
    console.log('Ejecutando servicio putApp');
    let url = this.urlBase + "/" + id;
    let headers = new Headers();
    let options = new RequestOptions({ headers: headers, withCredentials: true});
    return this.http.put(url,app,options);
}

postApp(app){
    console.log('Ejecutando servicio postApps');
    let headers = new Headers();
    let options = new RequestOptions({ headers: headers, withCredentials: true});
    return this.http.post(this.urlBase,app,options);
}
```

Figura 112 Métodos servicio app

Si nos fijamos en la figura 112 el primer método getApps() realiza una petición a la URL que guarda el atributo urlBase. En este caso el método realiza una petición a la API REST y esta ejecutará el método que se corresponde con el path y usando el

método GET. En este ejemplo el método ejecutado es el que devuelve todas las apps de la base de datos.

Para que un controlador de un componente pueda invocar los métodos de un servicio primero debe importar el servicio como se muestra en la figura 113.

```
import { AppService } from './../service/app.service';
```

Figura 113 importar servicio

Una vez importado creamos un atributo en el constructor de la clase del controlador que instancie la clase del servicio.

```
constructor(private appservice:AppService,
```

Figura 114 Atributo servicio

El último paso es invocar el método de la clase del servicio como se muestra en la figura 115. Se debe realizar una subscripción al método ya que es una petición AJAX. El primer parámetro que se le pasa a la subscripción es un método callback que se ejecutará cuando se reciba una respuesta correcta. El segundo parámetro es otra función que se ejecutará en caso de error.

```
public getApps() {
  this.appservice.getAppsByLimit(this.num,this.init).subscribe(
    response => {
      console.log(response.json());
      this.apps = response.json();
      console.log(this.apps);
      for(let i=0; i<this.apps.length; i++){
        this.imagesService.getImagesById_fotos(this.apps[i].id_fotos).subscribe(
          response => {
            this.images = response.json();
            for(let j=0; j<this.images.length; j++){
                if(this.images[j].type == "icon"){
                    this.iconos[this.apps[i].id_fotos] = this.globals.HOST + this.images[j].url;
                    console.log(this.iconos[this.apps[i].id_fotos]);
                }
            }
          },
          error => {

          }
        );
        this.vote_appService.getAverageByApp(this.apps[i].id).subscribe(
          response => {
            this.apps[i].average = response.text();
            console.log(this.apps[i].average);
          },
          error => {

          }
        );
      }
    },
    error => {
      console.log(error);
    }
  );
}
```

Figura 115 Petición AJAX

Este es el proceso que se ha seguido para la implementación de cada servicio de la aplicación cliente. En la tabla 34 se muestra la correspondencia de los servicios de la aplicación cliente con las clases del paquete resources del servicio web.

Tabla 34 correspondencia serviciso angular clases API REST

Servicio Angular2	**Clase API REST**
app.service.ts	AppResource.java
app_category.service.ts	App_categoryResource.java
app_platform.service.ts	App_platformResouce.java
categroy.service.ts	CategoryResource.java
commentary.service.ts	CommentaryResource.java
dataset.service.ts	DatasetResource.java
dataset_app.service.ts	Dataset_appResource.java

filtros.service.ts	FiltrosResource.java
images.service.ts	ImagesResource.java
institution.service.ts	InstitutionResource.java
platform.service.ts	PlatformResource.java
estadísticas.service.ts	EstadisticasResource.java
session.service.ts	LoginResource.java
user.service.ts	UserResource.java
vote_app.service.ts	Vote_appResource.java
vote_comment.service.ts	Vote_commentResource.java

5.4.3.4. Componentes

En este apartado listaremos y describiremos los componentes que forman la aplicación.

- **Cabecera:** Este componente será fijo en la aplicación, lo que quiere decir que estará en la pantalla independientemente de la URL. Se puede apreciar en la figura 116.

Figura 116 Componente cabecera

Este componente muestra el icono de la aplicación y el título. En una lista apilada hacia la derecha muestra enlaces a las tres secciones que tiene el portal que son aplicaciones, conjunto de datos y estadísticas. En caso de que el usuario que está navegando sea anónimo mostrará los enlaces de login y registro como se ve en la figura 116 y en caso de estar autenticado mostrará el nombre de usuario y la foto de perfil como se aprecia en la figura 117.

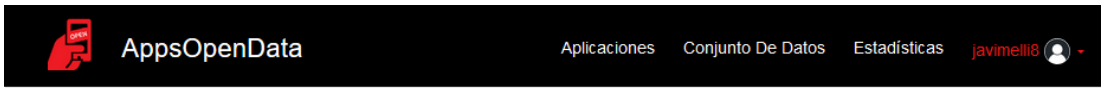


Figura 117 Componente cabecera 2

- **Footer:** Al igual que el componente cabecera el componente footer siempre estará presente independientemente de la URL de navegación. En la figura 118 se puede observar el componente.

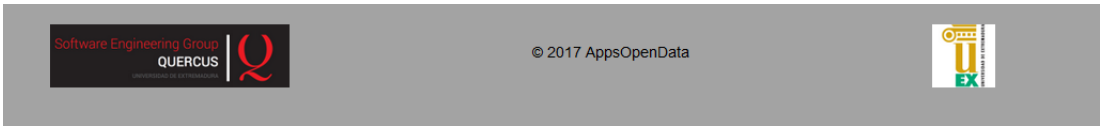


Figura 118 Componente footer

Destacar en este componente que el año que muestra está implementado dinámicamente con JavaScript, ya que actualizarlo puede ser una de las tareas que se olviden fácilmente en la fase de mantenimiento.

- **InicioComponent:** Este componente se corresponde con la url /. En la figura 119 se puede ver la vista de este componente.

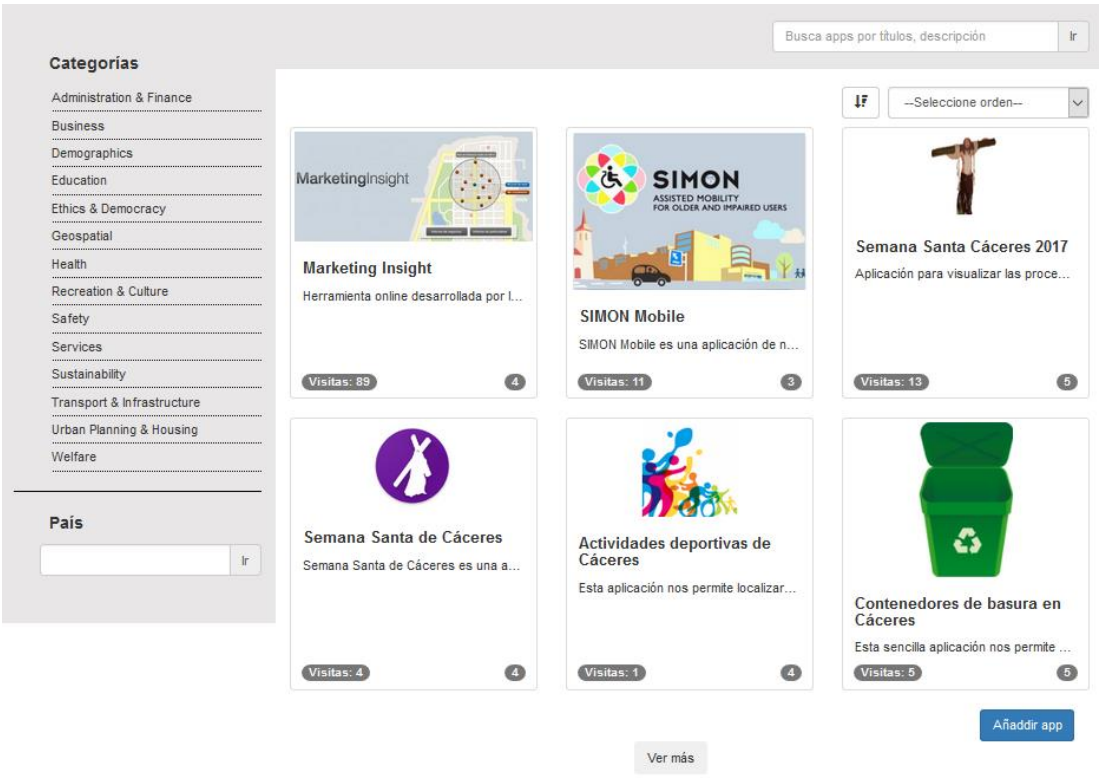


Figura 119 Componente InicioComponent

Está compuesto por un panel izquierdo en el que se muestran las categorías que se obtienen mediante peticiones a la API REST, un campo para introducir el país, Trae 6 aplicaciones mediante peticiones AJAX y también cuenta con botones que tienen el evento click escuchando para ejecutar los métodos

correspondientes y que están implementados en el controlador del componente.

- **App-detalle:** Este componente ocupa el router cuando la URL es /app-detalle/{idApp}.

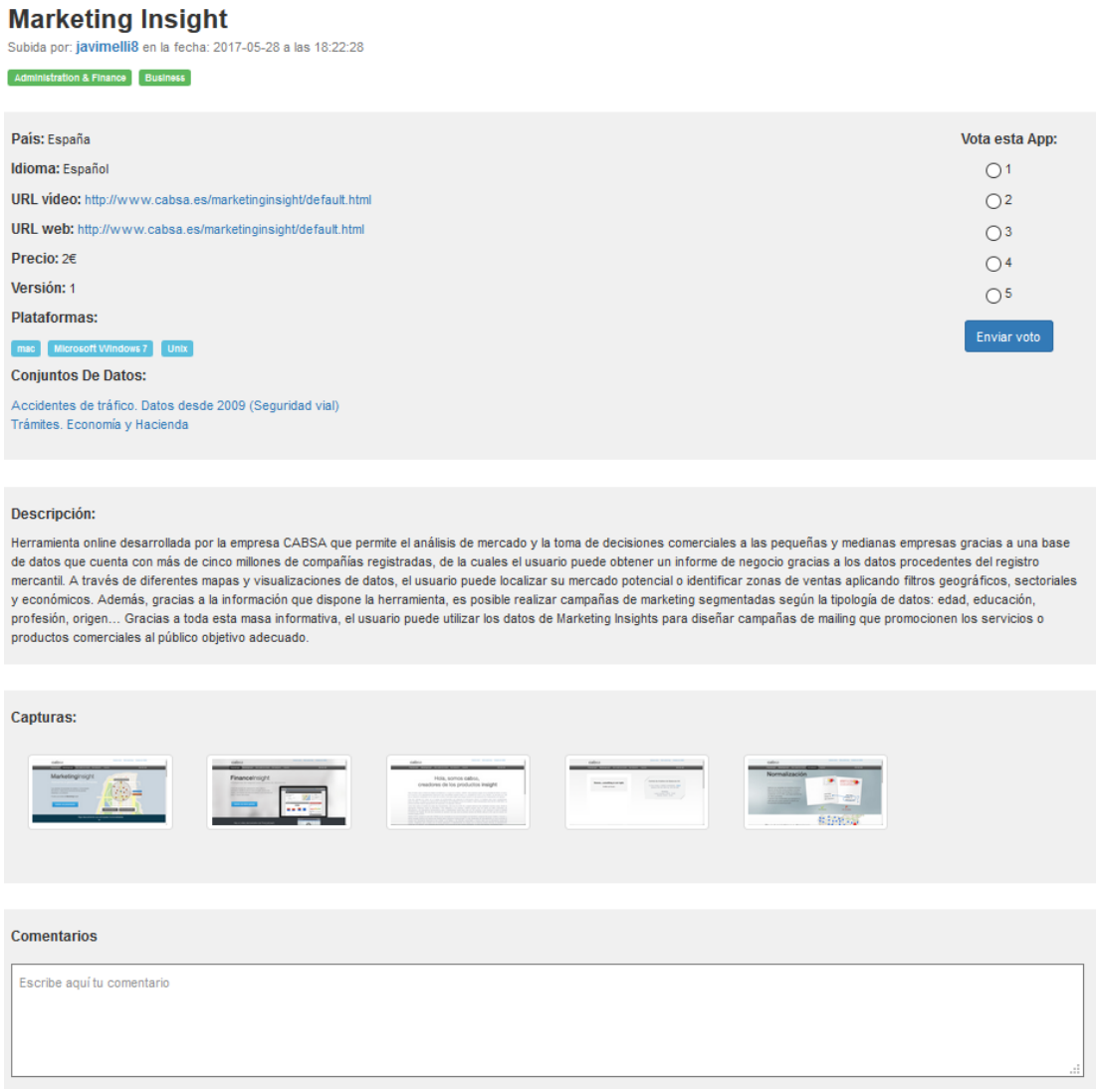
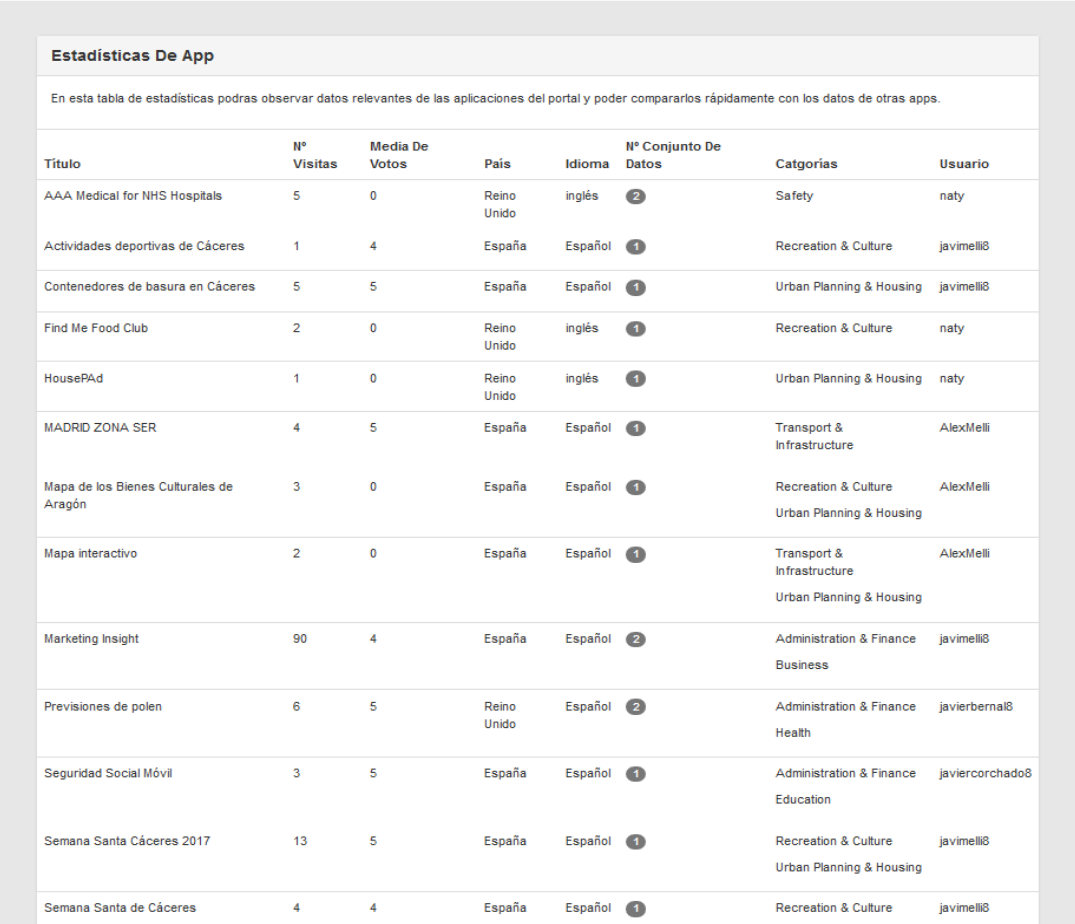


Figura 120 App-detalle componente

El objetivo de este componente es que cuando el usuario pinche en alguna aplicación en el listado de aplicaciones, obtenga todos los datos de la aplicación mediante al API REST y mostrarlo en el formato que se ve en la figura 120.

- **App-estadisticas:** Este componente se corresponde con la URL /app-estadisticas/. Su objetivo obtener la información de todas las aplicaciones del

portal vía AJAX y mostrarlas en el formato que se puede apreciar en la figura 121.



Estadísticas De App

En esta tabla de estadísticas podras observar datos relevantes de las aplicaciones del portal y poder compararlos rápidamente con los datos de otras apps.

Título	Nº Visitas	Media De Votos	País	Idioma	Nº Conjunto De Datos	Catgorías	Usuario
AAA Medical for NHS Hospitals	5	0	Reino Unido	inglés	2	Safety	naty
Actividades deportivas de Cáceres	1	4	España	Español	1	Recreation & Culture	javimelli8
Contenedores de basura en Cáceres	5	5	España	Español	1	Urban Planning & Housing	javimelli8
Find Me Food Club	2	0	Reino Unido	inglés	1	Recreation & Culture	naty
HousePAd	1	0	Reino Unido	inglés	1	Urban Planning & Housing	naty
MADRID ZONA SER	4	5	España	Español	1	Transport & Infrastructure	AlexMelli
Mapa de los Bienes Culturales de Aragón	3	0	España	Español	1	Recreation & Culture Urban Planning & Housing	AlexMelli
Mapa interactivo	2	0	España	Español	1	Transport & Infrastructure Urban Planning & Housing	AlexMelli
Marketing Insight	90	4	España	Español	2	Administration & Finance Business	javimelli8
Previsiones de polen	6	5	Reino Unido	Español	2	Administration & Finance Health	javierbernal8
Seguridad Social Móvil	3	5	España	Español	1	Administration & Finance Education	javiercorchado8
Semana Santa Cáceres 2017	13	5	España	Español	1	Recreation & Culture Urban Planning & Housing	javimelli8
Semana Santa de Cáceres	4	4	España	Español	1	Recreation & Culture	javimelli8

Figura 121 Componente App-estadisticas

- **Apps:** Este componente se corresponde con la URL /app/. Su objetivo es construir un formulario en el que un usuario pueda insertar una aplicación, dataset e instituciones y pueda subir al servidor el icono y las capturas de la aplicación que quiere dar de alta. Toda esta información debe darse de alta en la base de datos mediante la interacción con la API REST.

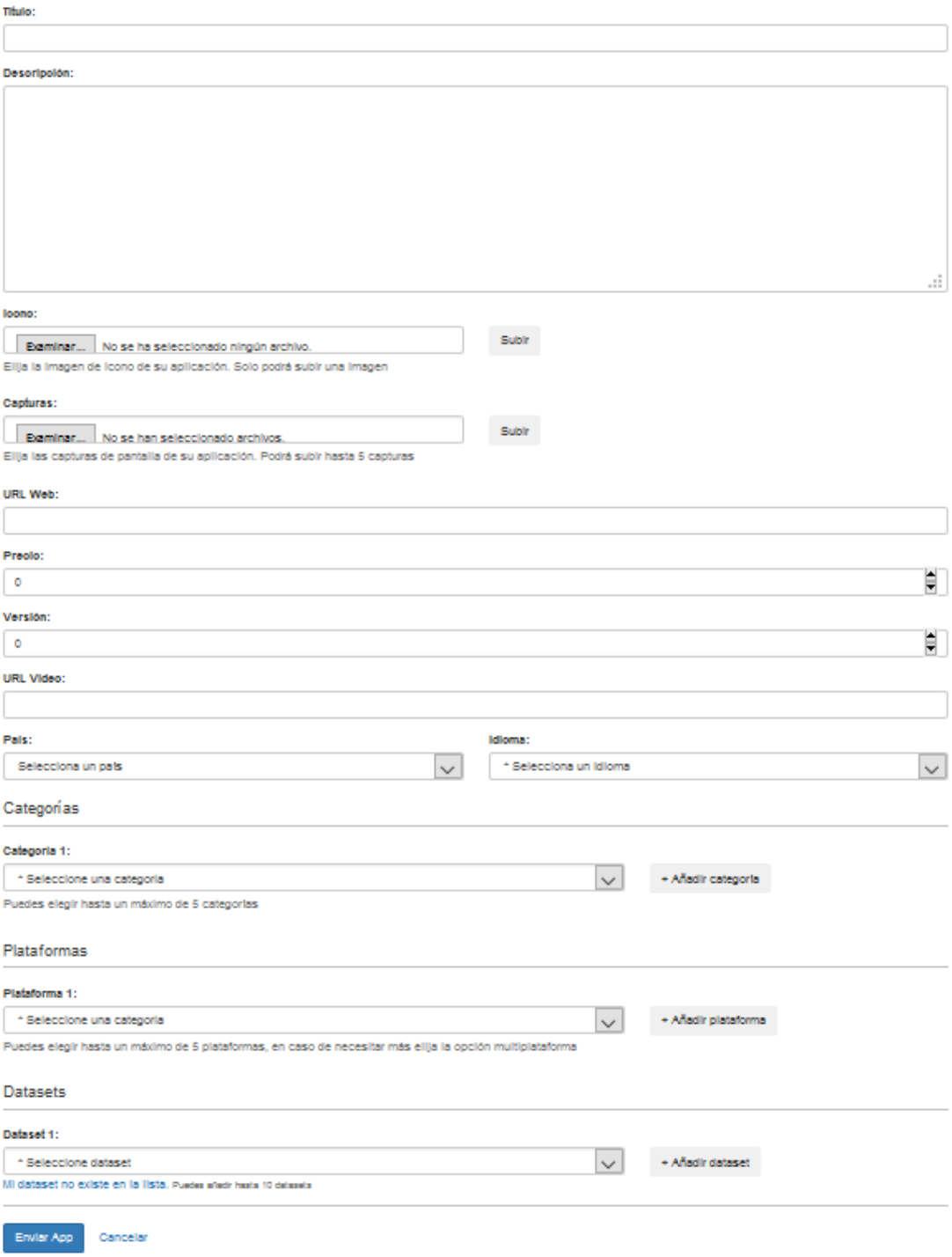


Figura 122 Componente App

- **Captures:** Este componente es utilizado en el HTML del componente App. Su objetivo es realizar el control e implementar la subida de las capturas en el servidor.

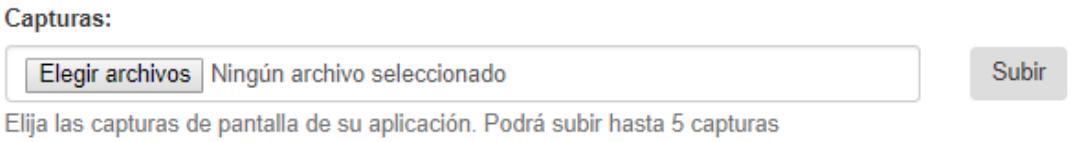


Figura 123 Componente capture

- **Icon:** Este componente es utilizado en el HTML del componente app. El objetivo es realizar el control e implementar la subida del icono al servidor de la aplicación que se quiere dar de alta.

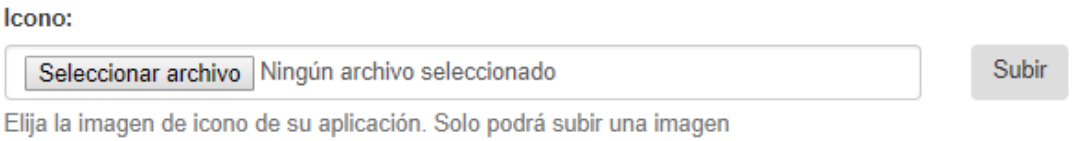


Figura 124 Componente icon

- **Dataset-detalle:** Este componente se relaciona con la URL /dataset-detalle/{idDataset}. Su objetivo es obtener los datos del dataset de la base de datos mediante peticiones a la API REST y mostrar la información en la vista con el formato de la figura 125.

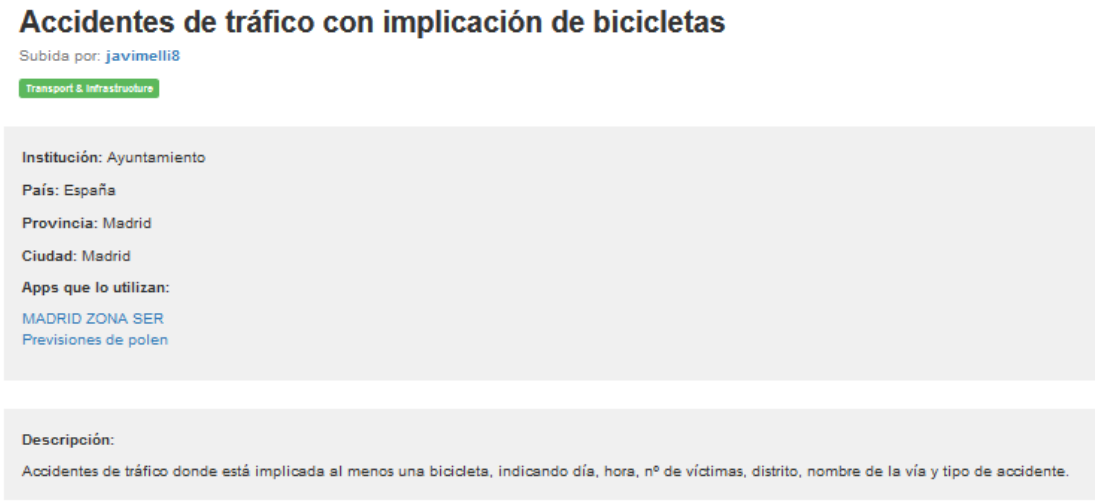


Figura 125 Componente Dataset-detalle

- **Datasets:** Este componente se corresponde con la URL /datasets/ y su objetivo es obtener los datasets de la base de datos y mostrarlos en la vista con el formato que se aprecia en la figura 126. También obtendrá las categorías e implementará dos campos para poder recoger los filtros del usuario.

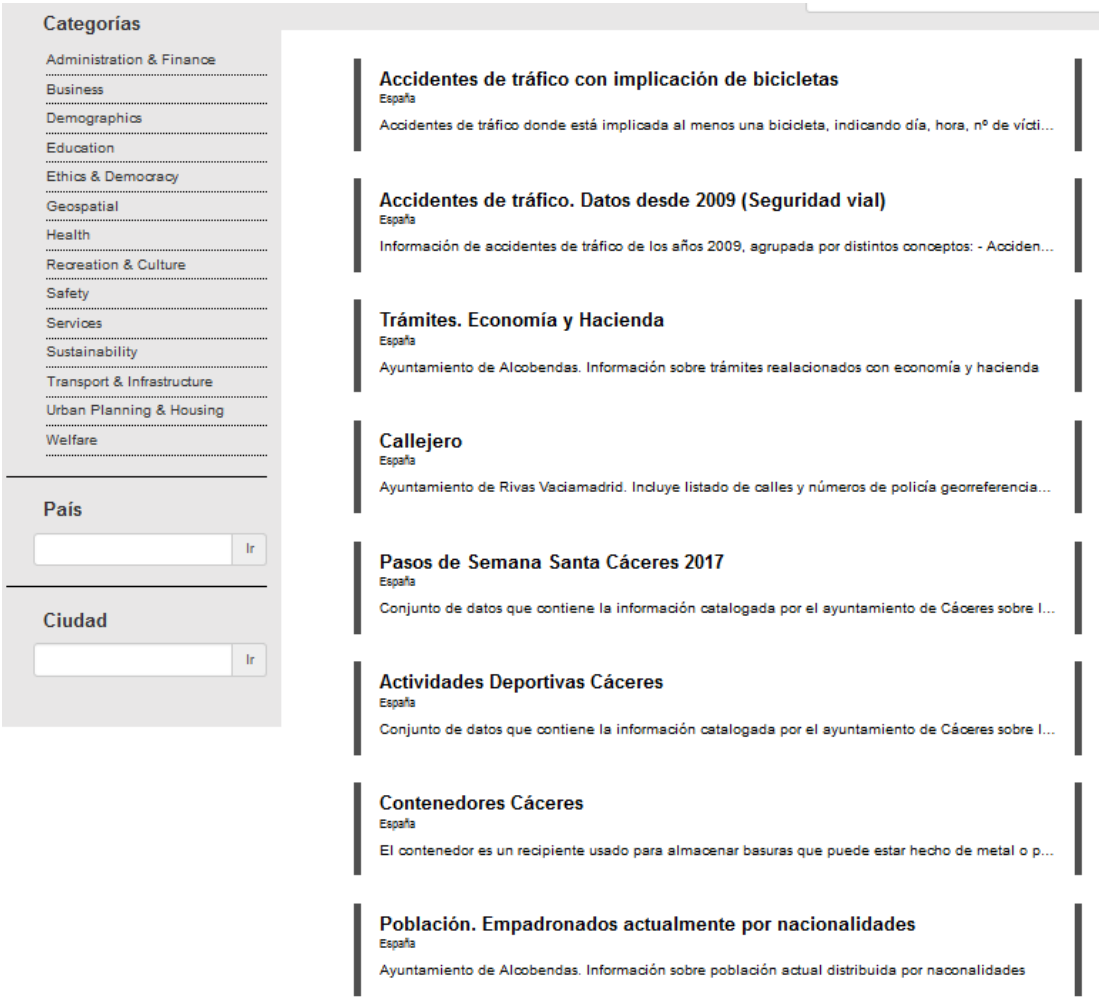


Figura 126 Coponente dataset

- **Edit-app:** La URL que se corresponde con este componente es /app/{idApp}. El objetivo es construir un formulario como el que construye el componente app, pero en este caso se debe precargar los datos de la app que tiene como id el pasado por parámetro en la URL.

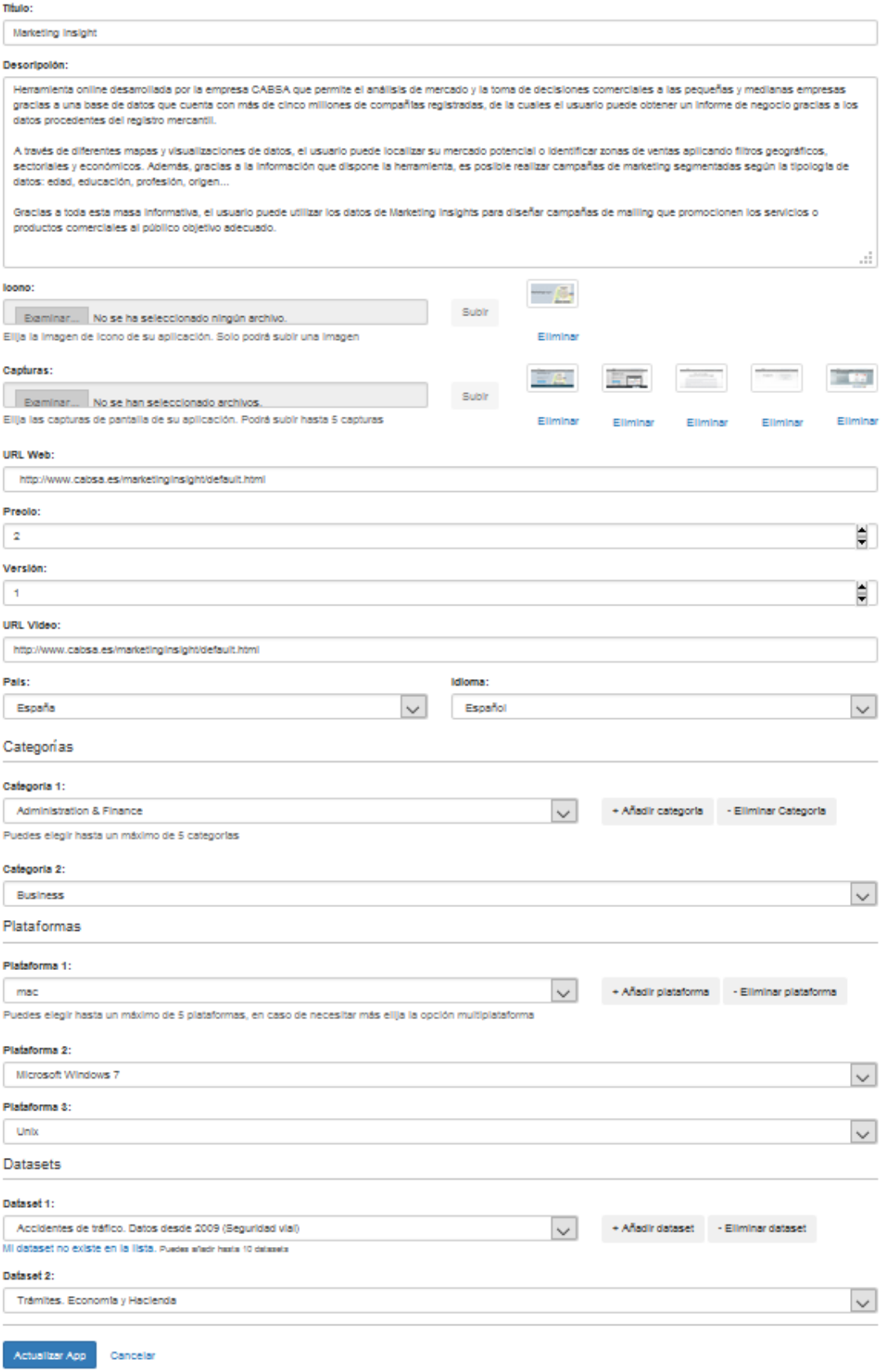


Figura 127 Componente edit-app

- **Registro:** Este componente se corresponde con la URL /registro/. El objetivo del componente es construir un formulario en el que un usuario pueda darse de alta en la aplicación y subir su foto de perfil al servidor. La información del usuario será introducida en la base de datos satisfactoriamente y la foto de perfil debe subirse correctamente al servidor.

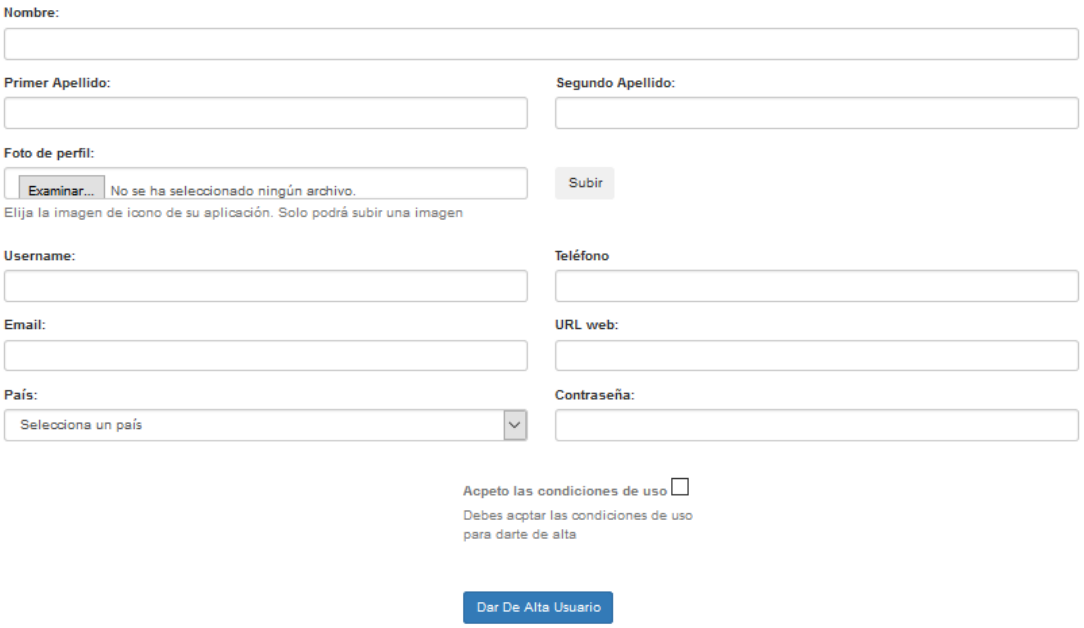


Figura 128 Componente registro

- **Editar-registro:** Este componente se corresponde con la URL /editar-perfil/. El objetivo de este componente es generar un formulario igual que el del componente registro, pero con los datos del usuario que tiene el mismo id que el usuario que está almacenado en la sesión.

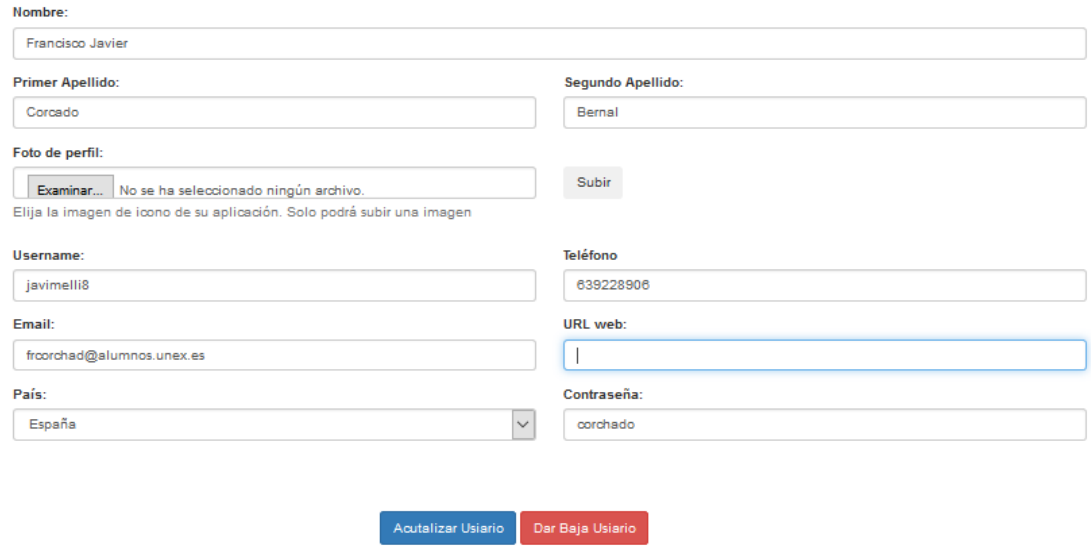


Figura 129 Componente editar-registro

- **Estadisticas:** La URL que se corresponde con este componente es /estadísticas/. Su objetivo es mostrar la vista de la figura 130 en la que hay dos enlaces. Uno cargará el componente app-estadisticas y otro el componente dataset-estadisticas en el router.

Figura 130 Componente estadísticas

- **Mis-apps:** Este componente se corresponde con la URL /mis-apps/. El objetivo de este componente es crear una vista en la que se carguen un listado con las aplicaciones que ha dado de alta el usuario que está almacenado en la sesión y proporcionarle un enlace a cada aplicación que cargue el componente edit-app con la app que ha pinchado.

Figura 131 Componente edit-app

Toda la información que necesitan los componentes de la aplicación la obtienen mediante peticiones asíncronas a la API REST utilizando los servicios como ya se ha explicado en este capítulo.

Para mostrar la información que obtienen los controladores de la API REST en la vista se realiza como se muestra en la figura 132.

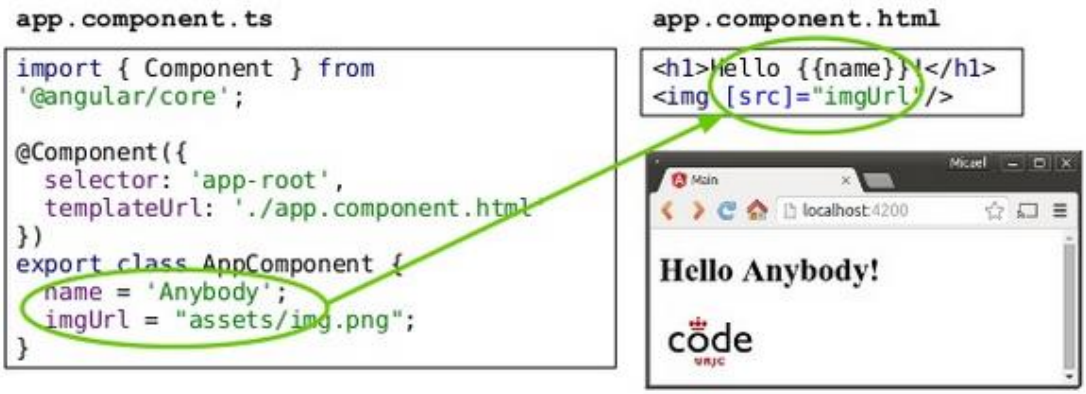


Figura 132 binding angular 2

Para capturar los eventos del usuario y poder ejecutar métodos implementados en el controlador del componente se realiza como en el ejemplo de la figura 133.

app.component.ts

```
import {Component} from '@angular/core';

@Component({
  selector: 'app',
  templateUrl: './app.component.html'
})
export class AppComponent {

  name = 'Anybody';

  setName(name:string){
    this.name = name;
  }
}
```

app.component.html

```
<h1>Hello {{name}}!</h1>

<button (click)="setName('John')">
    Hello John
</button>
```

Figura 133 Captura de eventos angular 2

Para recoger los datos de la vista en variables declaradas en el controlador se realiza como se muestra en la figura 134. Esta operación siempre se realiza utilizando elementos de un formulario HTML para que el usuario pueda introducir información.

app.component.ts

```
import {Component} from '@angular/core';

@Component({
  selector: 'app',
  templateUrl: './app.component.html'
})
export class AppComponent {

  name = 'Anybody';

  setName(name:string){
    this.name = name;
  }
}
```

app.component.html

```
<input type="text" [(ngModel)]="name">

<h1>Hello {{name}}!</h1>

<button (click)="setName('John')">
    Hello John
</button>
```

Figura 134 recoger datos de la vista al controlador

5.5. Problemas encontrados y soluciones

En este apartado se describen los problemas más importantes que se han encontrado durante la fase de desarrollo de la aplicación web y las soluciones a estos problemas. No se han tenido en cuenta los típicos problemas que hay en el desarrollo de cualquier aplicación, solamente los problemas que ha paralizado la implementación un tiempo considerable.

5.5.1. Sesión

Como se ha explicado en este documento hay que tener un control de la sesión de usuario. El funcionamiento se ha explicado en el apartado 5.3.4.3 del capítulo 5. Para resumir, cuando un usuario entra a usar la aplicación por primera vez el servidor crea una nueva sesión en el servidor y envía una cookie al navegador del usuario. A partir de ese momento cada petición que reciba el servidor del usuario vendrá con esa cookie que contiene un hash con el que será identificado.

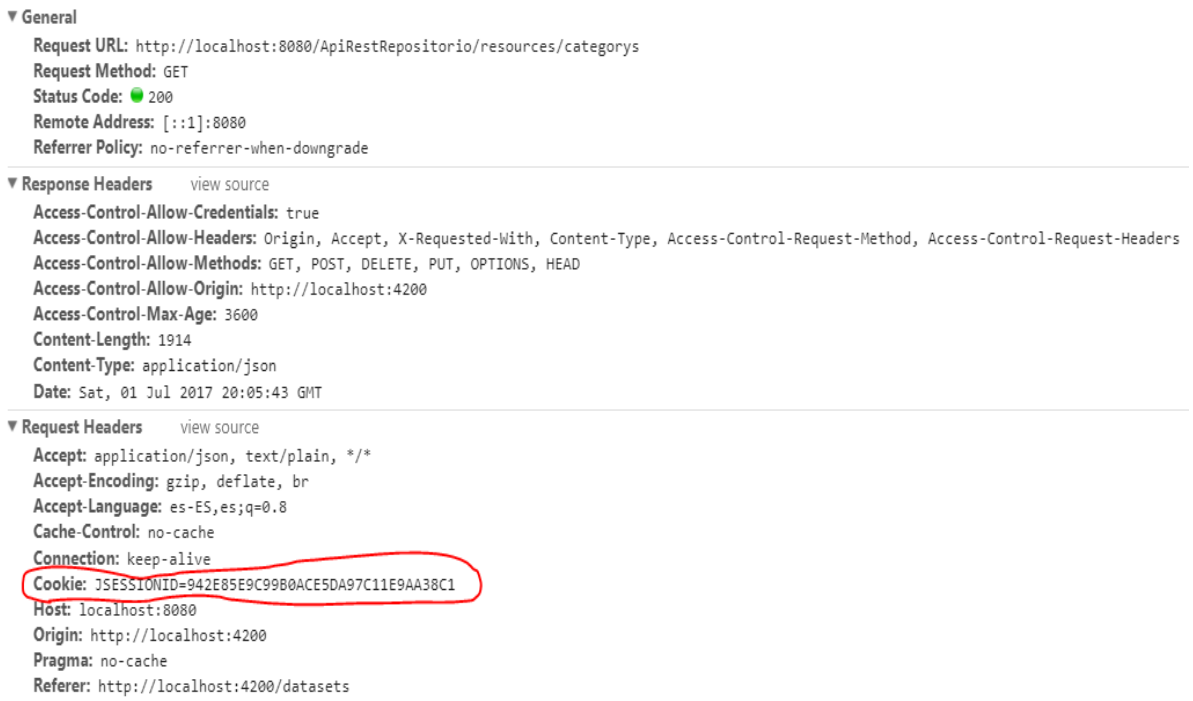


Figura 135 problema sesión

Como se puede apreciar en la figura 135 es una petición que se realiza a la API REST para traer las categorías de la base de datos. En esta petición va la cookie con el hash que nos identificará en el servidor porque ya se ha iniciado una sesión previamente.

El problema es que cuando se estaba implementando la sesión, el usuario no enviaba esta cookie con el hash identificativo, por tanto en cada petición se abría una sesión diferente y no se podía realizar ningún tipo de control.

El problema venía provocado por la política CORS de orígenes cruzados.

Por razones de seguridad, los exploradores restringen las solicitudes HTTP de origen cruzado iniciadas dentro de un script. Por ejemplo, XMLHttpRequest sigue la

política de mismo-origen. Por lo que, una aplicación usando XMLHttpRequest (en el caso de nuestro proyecto las peticiones asíncronas a la API REST) solo puede hacer solicitudes HTTP a su propio dominio [29].

Nosotros en el entorno de pruebas la API REST está desplegada en el domino localhost:8080 y la aplicación cliente en localhost:4200. Al ser puertos diferentes ya son dominios diferentes.

La solución fue añadir el parámetro withCredentials en la cabecera de las peticiones que se realizan desde el cliente. En la figura 136 se puede apreciar cómo se añade a la petición.

```
getAppsByLimit(num, init){
    console.log('Ejecutando servicio getAppsByCountry');
    let url = this.urlBase + "/limit/" + num + "/" + init;
    let headers = new Headers();
    let options = new RequestOptions({ headers: headers, withCredentials: true});
    return this.http.get(url,options);
}
```

Figura 136 parámetro withcredentials

5.5.2. Peticiones asíncronas

El problema que se va a comentar está muy relacionado con el anterior. El problema venía provocado por la política CORS y se trataba de que por cada petición realizada desde el cliente al servidor daba un error cross-origin.

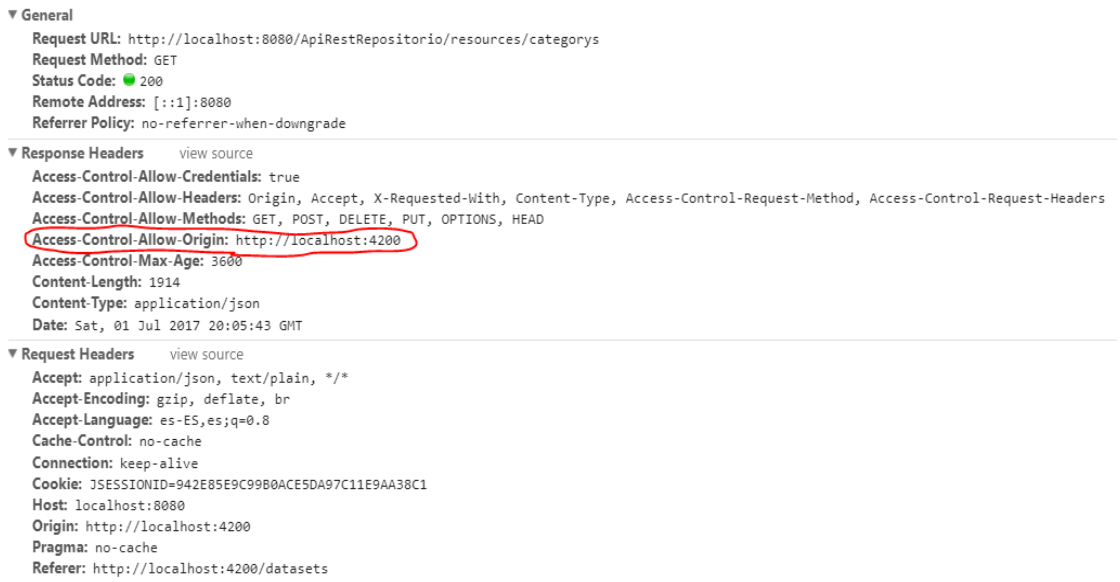


Figura 137 Problema cross origin

Como se puede observar en la figura 137 el parámetro que está destacado indica que se aceptan peticiones del origen con el dominio localhost:4200.

Este parámetro y todos los Acces-control faltaban en las respuestas del servidor.

La solución fue implementar el filtro CORSFilter que se ha explicado en el apartado 5.3.4.2 de este documento. Lo que hace es añadir estos parámetros a las cabeceras de respuesta de todas las peticiones que recibe la API REST.

5.5.3. Subida de imágenes

El problema que se ha encontrado en la subida de imágenes al servidor, es que en los formularios, al realizar la inserción de un usuario o una aplicación, la subida de las imágenes de perfil, iconos o capturas se realizaba independiente a la inserción de la aplicación o del usuario en la base de datos.

Esto implica que no se podía relacionar la imagen con su aplicación o con su usuario porque todavía no estaban dados de alta en la base de datos y por tanto no existía el id de estos para poder relacionarlos.

La solución fue insertar un campo nuevo tanto en la entidad usuario como en la de app llamado id_fotos. Este id_fotos es un identificador formado por día, mes, año, hora, minutos, segundos y un aleatorio de cuatro cifras que se genera al cargarse el formulario. En la figura 138 se muestra la generación del id_fotos.

```
var aleatorio = Math.round(Math.random() * (9999 - 1000) + 1000);
var date = new Date();
this.id_fotos = date.getDate()+"_"+date.getMonth()+"_"+date.getFullYear()+"_"+date.getHours()+"_"+date.getMinutes()+"_"+date.getSeconds()+"_"+aleatorio;
this.app.id_fotos = this.id_fotos;
console.log(this.app.id_fotos);
```

Figura 138 Generación id_fotos

Se ha creado la entidad nueva images que guarda el id_fotos, la url y el tipo como ya se ha descrito en este documento.

Ahora el proceso será el siguiente.

- Al entrar el usuario al formulario se genera el id_fotos.
- Al subir la imagen al servidor el id_fotos se guarda en el controlador y se inserta el registro en la tabla images. La imagen es subida en el directorio de imágenes que se ha explicado en el apartado 5.3.4.5 y se puede apreciar en la figura 57. Como se puede ver se usa el id_fotos en el directorio para

discriminar las imágenes de una aplicación de las de otras y la imagen de perfil de un usuario de la de otro.

- Por último, cuando el usuario de alta a la aplicación o al usuario, en estos registros se almacena el id_fotos.

Con este proceso ya quedan enlazadas las imágenes con sus propietarios mediante el id_fotos.

En la edición ya no se genera ningún id_fotos, porque ya no se necesita. Ya sabemos que imágenes de perfil pertenece a que usuario y que iconos y capturas pertenecen a que aplicaciones.

6. Manual de usuario

En este apartado se expone todo el manual de usuario de la aplicación web OpenDataApps con el fin de facilitar la navegación por el portal.

Se realizará una navegación por todo el portal, mostrando imágenes y explicando todas las funcionalidades.

6.1. Requisitos

Para poder realizar una navegación por el portal se necesitan tener los siguientes requisitos:

- Tener acceso a internet.
- Tener instalado un navegador web con el que acceder al portal.

6.2. Guía de la aplicación

El primer paso es abrir un navegador web y acceder al portal a través de la URL. Se le abrirá la página principal que aparece en la figura 139.

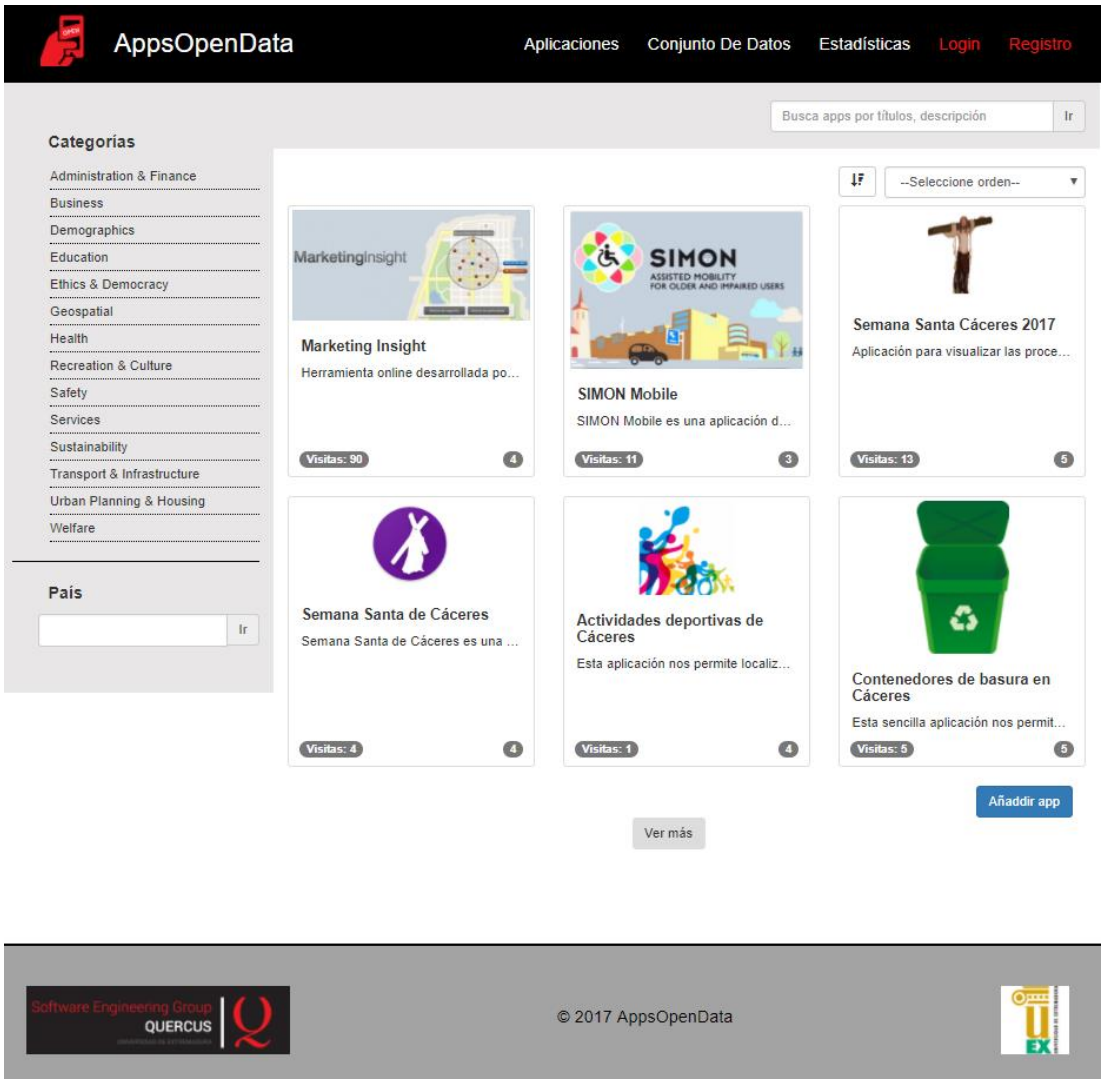


Figura 139 Página inicial

Esta será la página de bienvenida indistintamente esté autenticado o anónimo.

En esta vista se mostrará un listado de aplicaciones que inicialmente estará ordenado por el orden en el que se han insertado en la base de datos. Hay un botón al final del listado con el texto “Ver más” que si es pulsado ampliará el número de aplicaciones del listado y otro botón en la parte inferior derecha con el texto “Añadir app” que si es pulsado y usted está autenticado le mostrará un formulario para dar de alta una aplicación, si por el contrario usted es anónimo le mostrará un formulario de login.

6.2.1. Filtrado de aplicaciones

En la página inicial usted podrá filtrar las aplicaciones por categorías si pincha en algunas de ellas en el panel izquierdo. En la figura 140 se puede apreciar un filtro por la categoría Recreation & Culture.

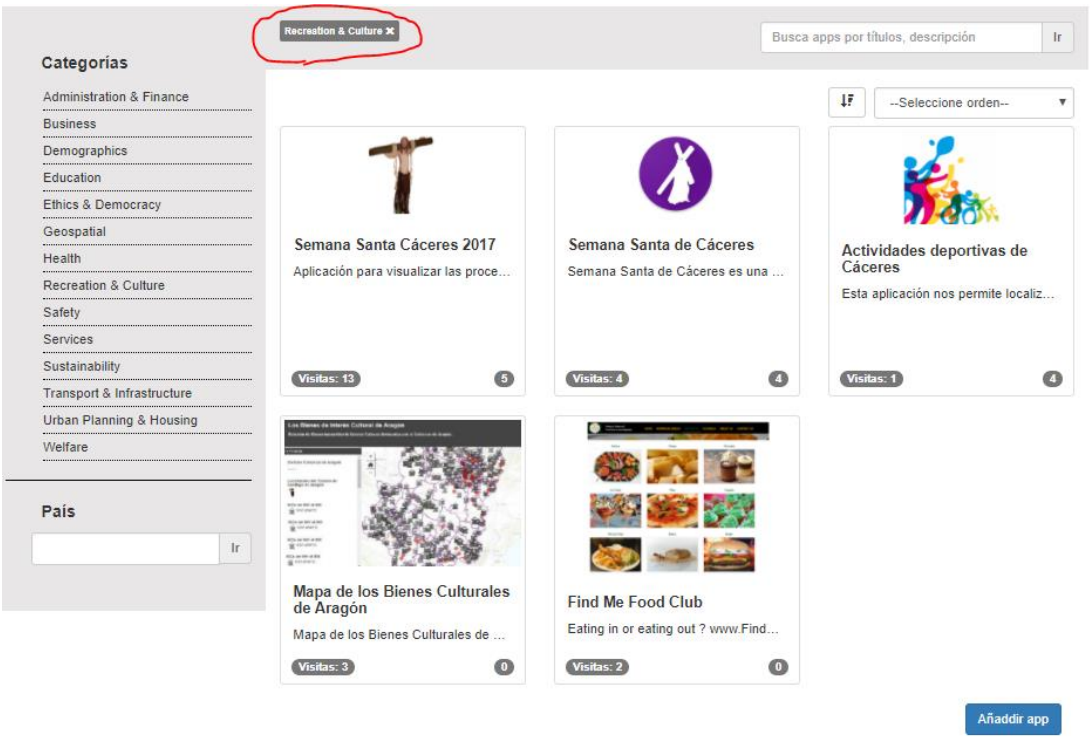


Figura 140 Filtro Recreation & Culture

También podrá filtrar aplicaciones por país introduciendo el nombre del mismo en el capo que aparece en el panel izquierdo debajo de las categorías. En la figura 141 se muestra un filtrado por el país España.

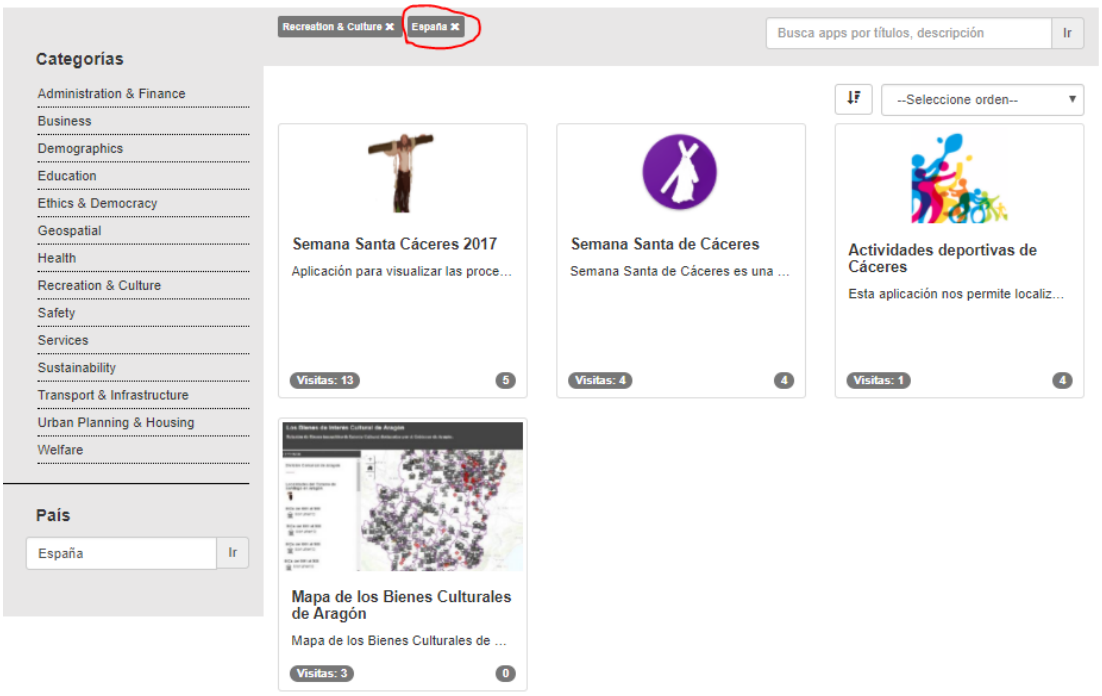


Figura 141Filtrado España

Para eliminar cualquiera de los filtros solo deberá pulsar el símbolo X situado a la derecha de las etiquetas que aparecen cuando realiza un filtro en la parte superior del listado y se han señalado en las figuras 140 y 141.

Una vez tenga el listado que desea, usted podrá ordenarlo por el número de visitas recibidas, la media de votaciones y el precio de la aplicación. Podrá ordenarlo ascendentemente y descendentemente.

Para ello usted debe seleccionar un orden en el selector que aparece en la parte superior derecha del listado y elegir si desea que el orden sea descendente o ascendente pulsando en el icono situado en la izquierda del selector. En la figura 142 aparece un listado ordenado por el número de visitas y ordenado descendentemente.

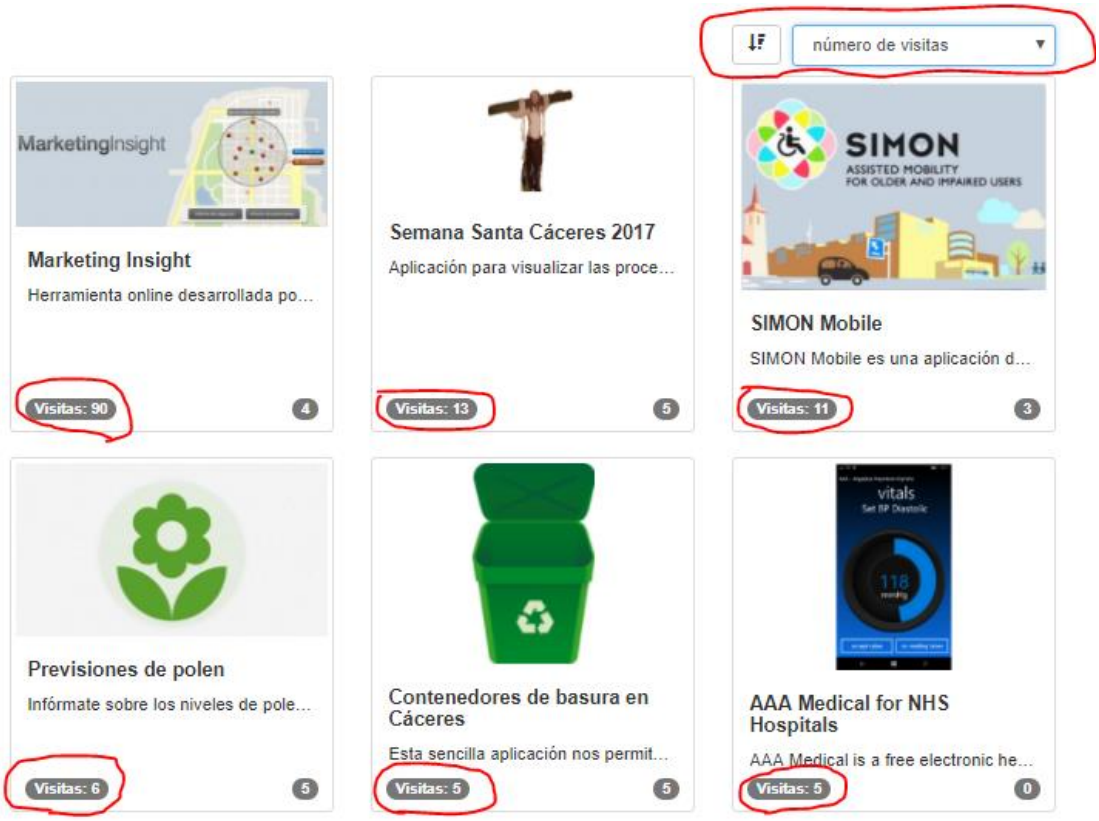


Figura 142 Orden número de visitas descendente

Si usted no desea filtrar un listado de aplicaciones y necesita una búsqueda más directa, se le proporciona un buscador por palabras situado encima del selector de orden. En el podrá introducir la palabra que deseé y se realizará una búsqueda de esa palabra sobre el título y la descripción de todas las aplicaciones existente en la base de datos. En la figura 143 se realiza una búsqueda de la palabra semana.

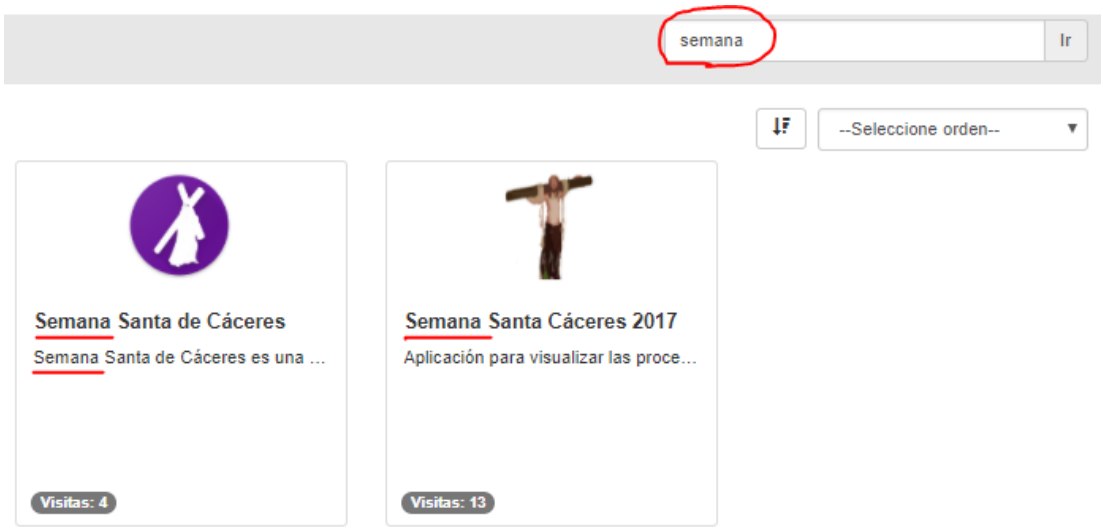


Figura 143 Búsqueda por la palabra semana

Una vez usted haya encontrado la aplicación que desee o simplemente quiera conocer más datos sobre alguna, solo deberá pulsar sobre ella y le aparecerá una vista con datos más detallados sobre la aplicación. Véase en la figura 144.

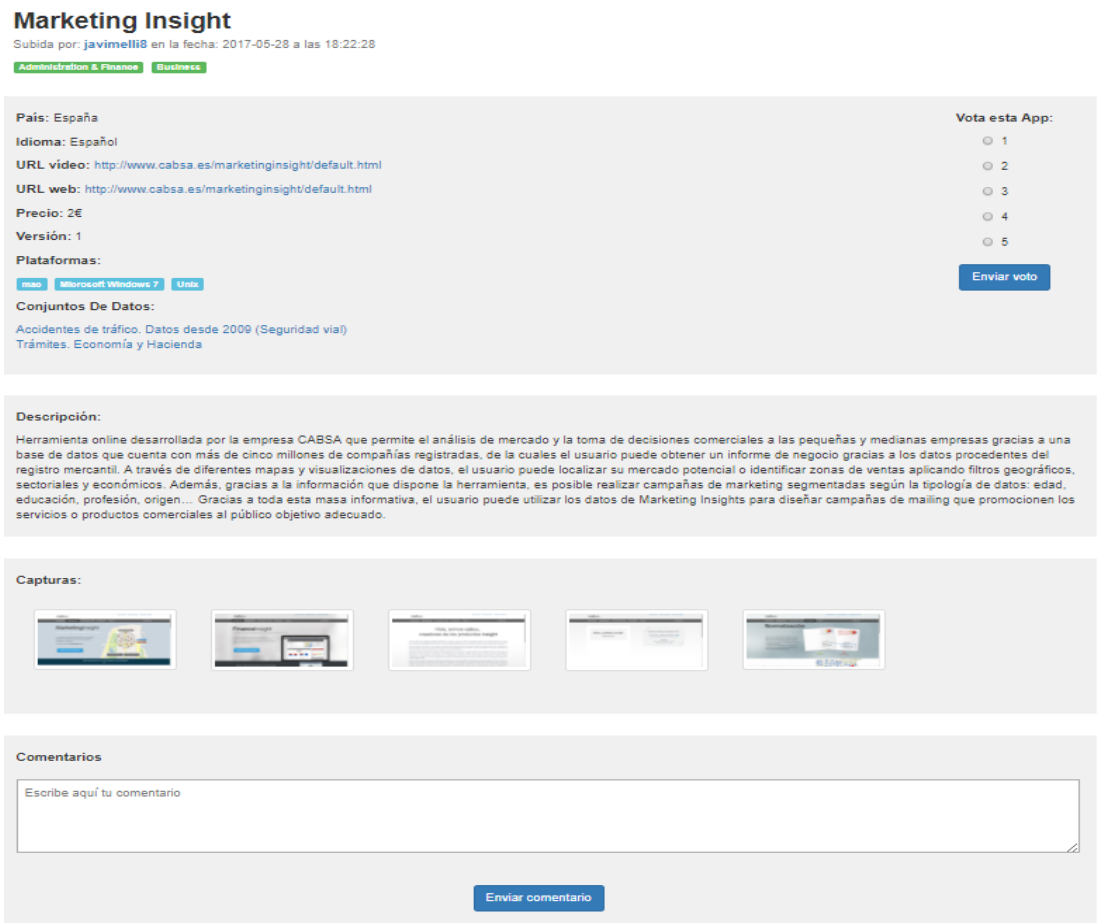


Figura 144 Vista aplicación detalle

6.2.2. Filtrado de conjunto de datos

Si usted lo que desea es ver un listado de datasets concretos simplemente debe pulsar en la sección "conjunto de datos" situado en la cabecera superior del portal y le aparecerá una página como la de la figura 145.

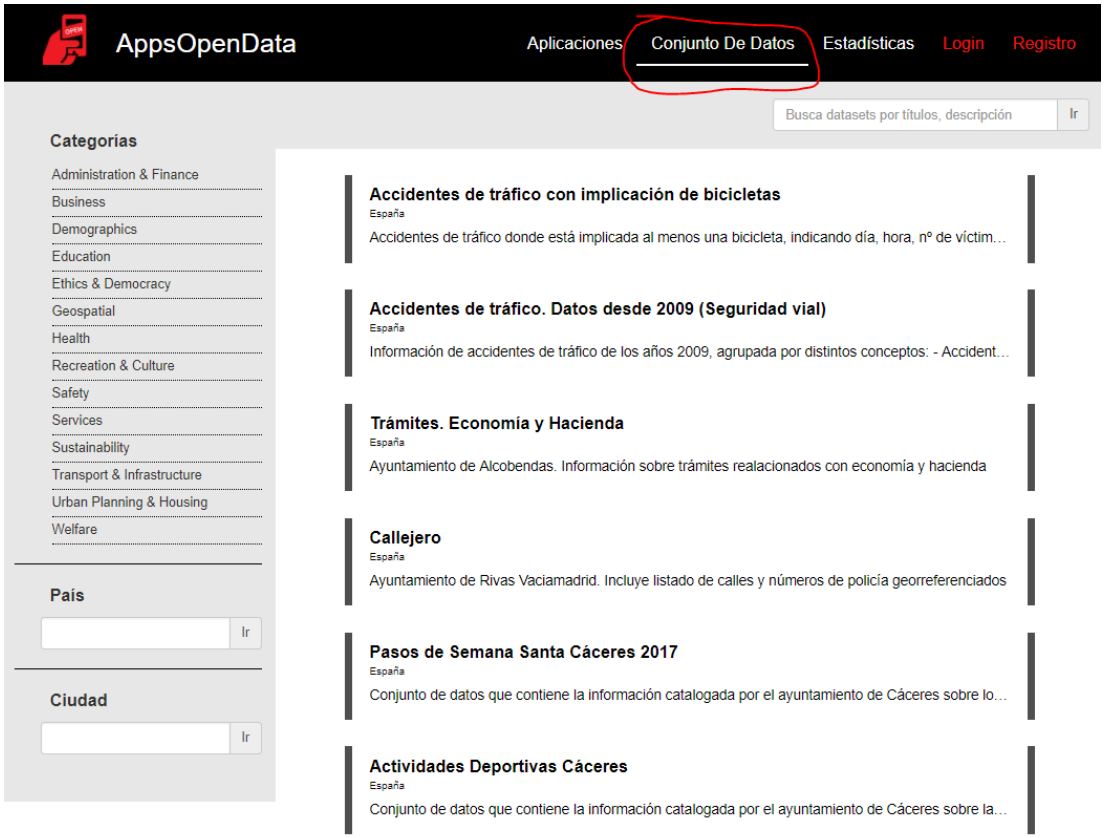


Figura 145 Listado de datasets

Podrá filtrar datasets por categorías pinchando en alguna categoría situada en el panel izquierdo de la página. En la figura 146 se muestra un listado filtrado por la categoría Transport & Infrastructure.

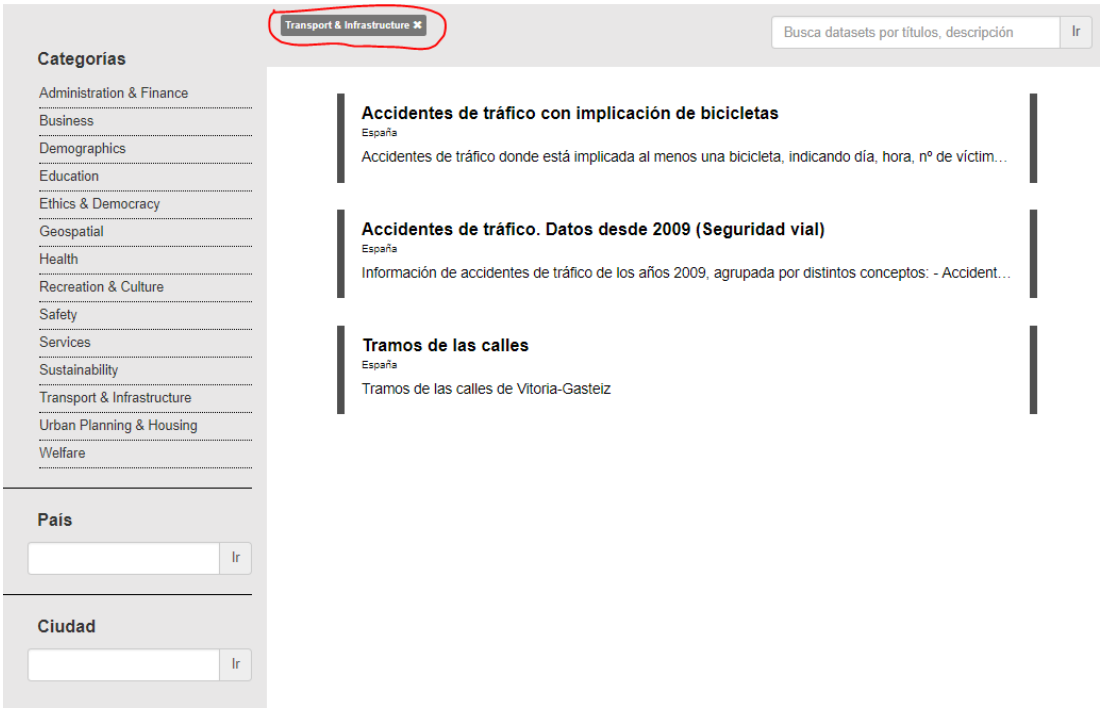


Figura 146 Filtrado de datasets por categoría

Usted también podrá filtrar datasets por país y ciudad. Para ello deberá introducir el país y la ciudad por la que quiera filtrar en los campos situados debajo de las categorías en el panel izquierdo de la página. En la figura 147 e realiza un filtrado por el país España y la ciudad de Madrid.

Figura 147 Filtrado por país y ciudad

Si usted necesita realizar una búsqueda por palabra la aplicación le proporciona un buscador situado en la parte superior derecha del listado. Hay debe introducir la palabra por la que desea buscar y la aplicación realizará una búsqueda de la palabra en los títulos y descripciones de todos los datasets dados de alta en la base de datos. En la figura 148 se muestra una búsqueda de la palabra semana.

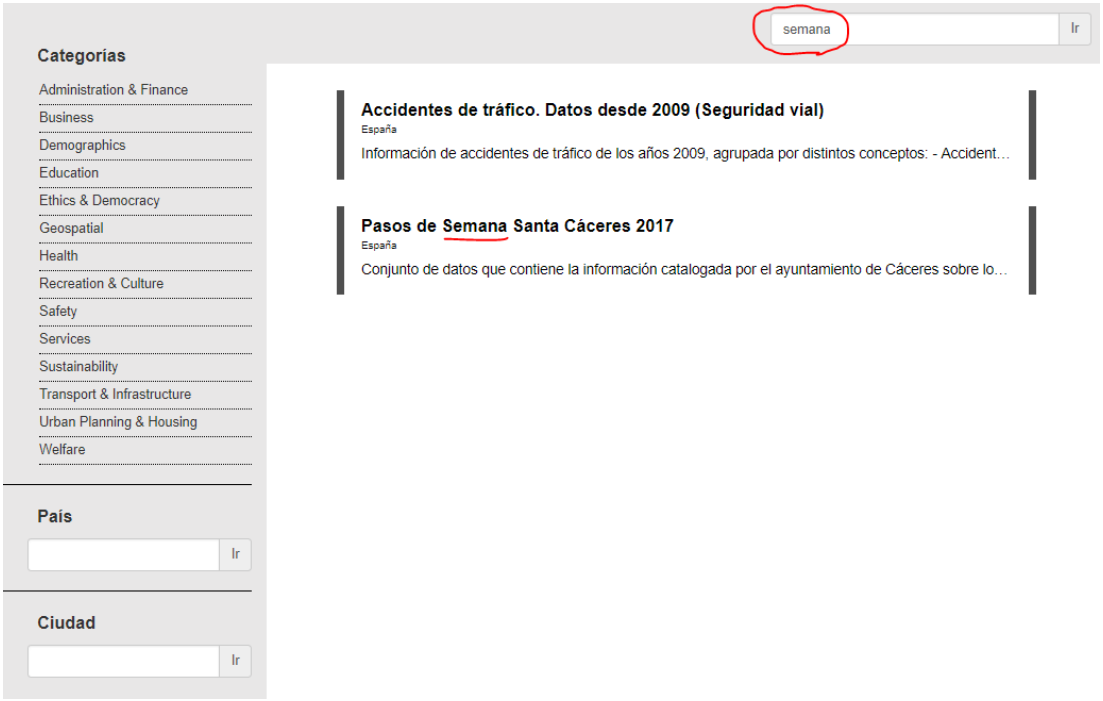


Figura 148 Búsqueda por palabras datasets

Todos los conjuntos de datos mostrados contendrán la palabra semana aunque en el listado no se pueda apreciar. Si usted pulsa sobre un dataset aparecerá una página con datos más detallados sobre el dataset consultado y podrá comprobar como contiene la palabra por la que se realizó la búsqueda, en este caso la palabra semana. En la figura 149 se puede apreciar la página de detalle del primer dataset que se muestra en el listado de la figura 148.

Figura 149 Página detalle dataset

6.2.3. Registro

Para poder darse de alta en la aplicación y acceder a todas las funcionalidades del portal deberá pulsar sobre el apartado “Registro” situado en la parte derecha de la cabecera superior del portal.

Le aparecerá un formulario como el de la figura 150. Usted deberá rellenarlo y enviarlo correctamente para poder ser dado de alta.

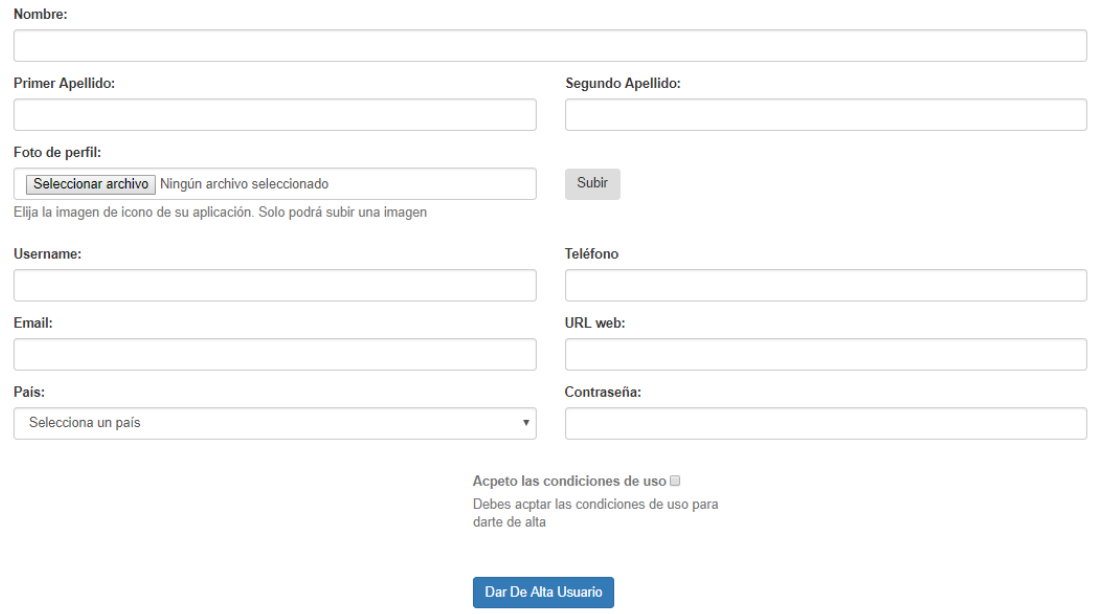


Figura 150 Formulario registro

Para rellenar este formulario correctamente deberá tener cuenta que debe rellenar todos los campos obligatorios. Si no es así al pulsar sobre “Dar De Alta Usuario” se le marcarán en rojo los campos obligatorios. Además deberá marcar el checkbox que indica que aceptas las condiciones de uso.

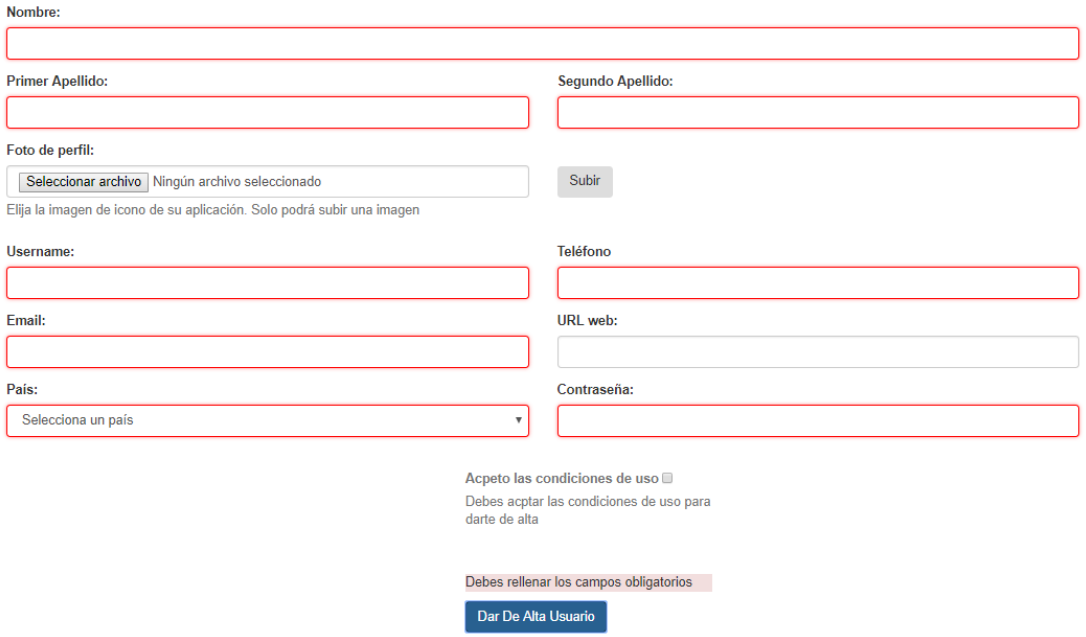


Figura 151 Validaciones registro

Una vez cumplido estos requisitos será dado de alta en la aplicación y se le mostrará la página de login para que inicie sesión como usuario autenticado.

6.2.4. Inicio se sesión como usuario autenticado

Para iniciar sesión como usuario autenticado usted deberá ingresar en el formulario de login pulsando sobre “Login” situado en la cabecera superior del portal. Le aparecerá un formulario como el que se muestra en la figura 152.

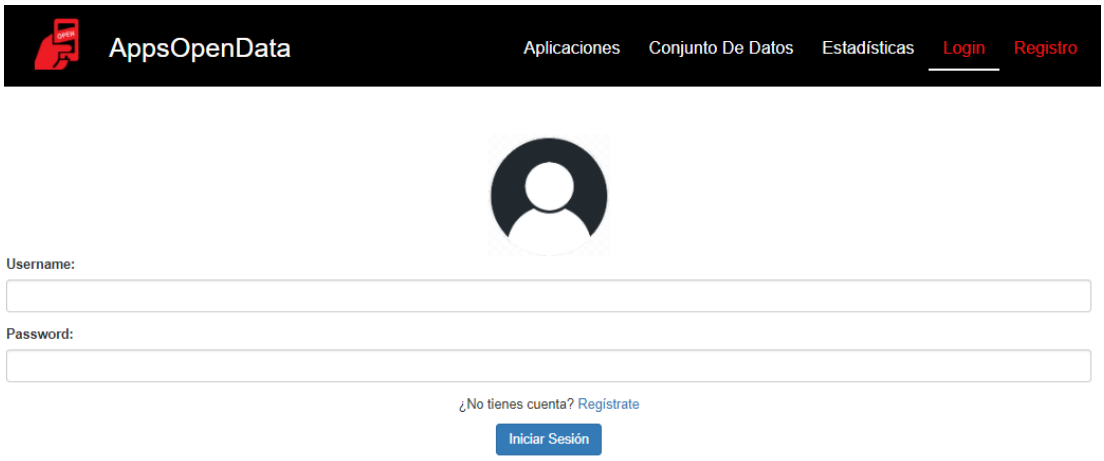


Figura 152 Formulario Login

Usted debe ingresar su username y password en el formulario y pulsar sobre Iniciar Sesión. Si son correcto se le mostrará la página principal con los cambios marcados en la figura 153 y ya podrá disfrutar de todas las funcionalidades que le están permitidas a un usuario autenticado.

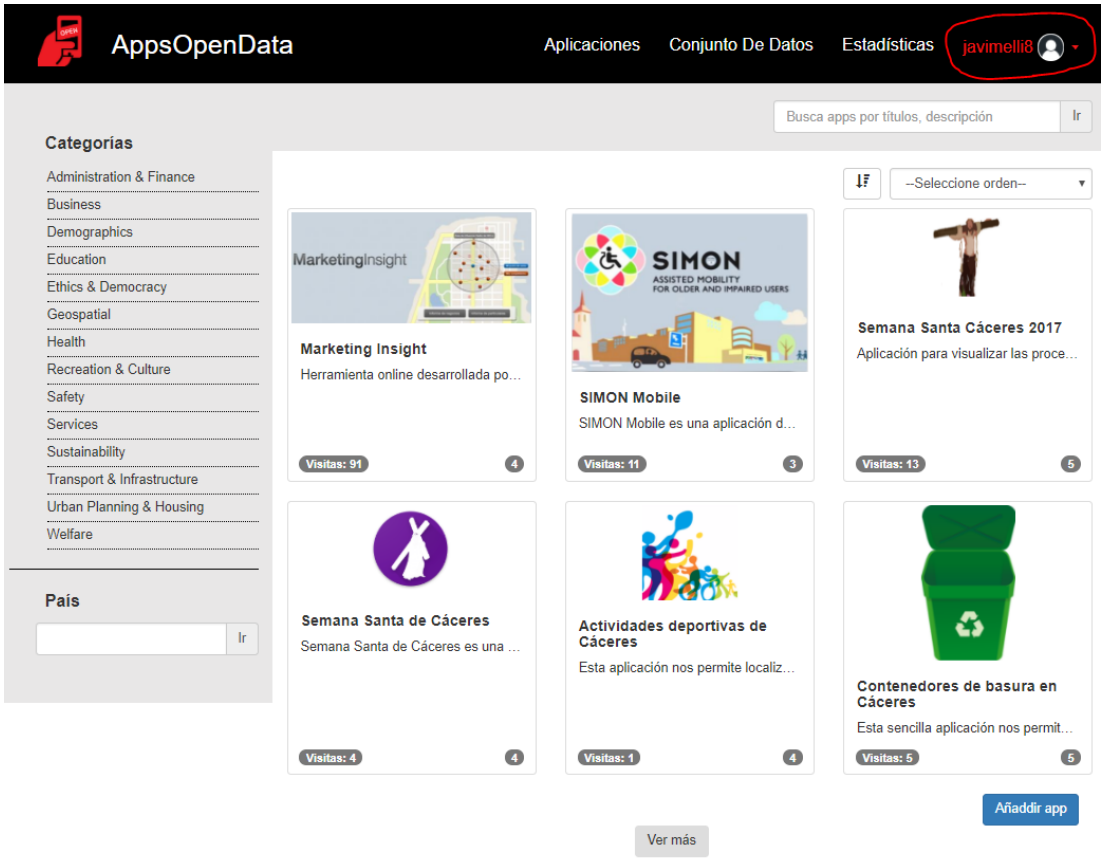


Figura 153 Página inicial usuario autenticado

En caso de que el username o la password introducidas sean incorrectas se le indicará con el mensaje que se puede apreciar en la figura 154.

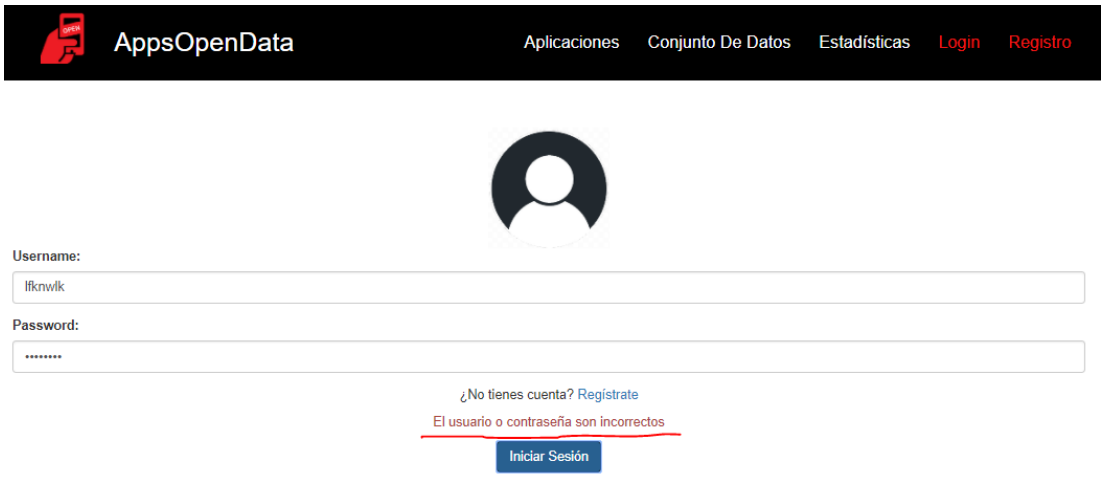


Figura 154 Formulario incorrecto Login

6.2.5. Edición de perfil

Para poder editar sus datos de perfil deberá iniciar sesión como se le indica en el apartado 6.2.4 de este manual. Una vez iniciada sesión deberá abrir el desplegable que aparece en la parte derecha de la cabecera superior del portal con el nombre de su username y elegir la opción editar perfil.

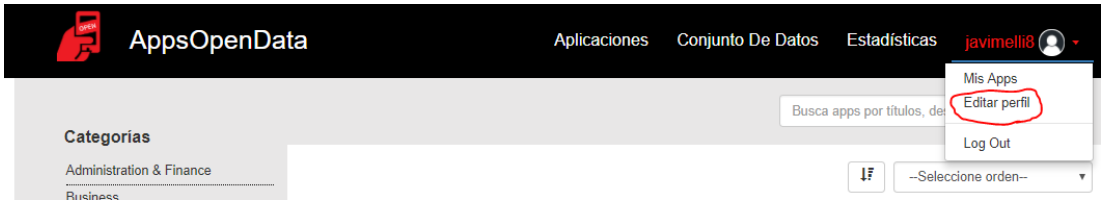


Figura 155 Editar perfil

Se le mostrará un formulario similar y con los mismos controles de validaciones que el explicado en el apartado 6.2.3. de este manual, pero con sus datos de perfil precargados en los campos del formulario como se muestra en la figura 156.

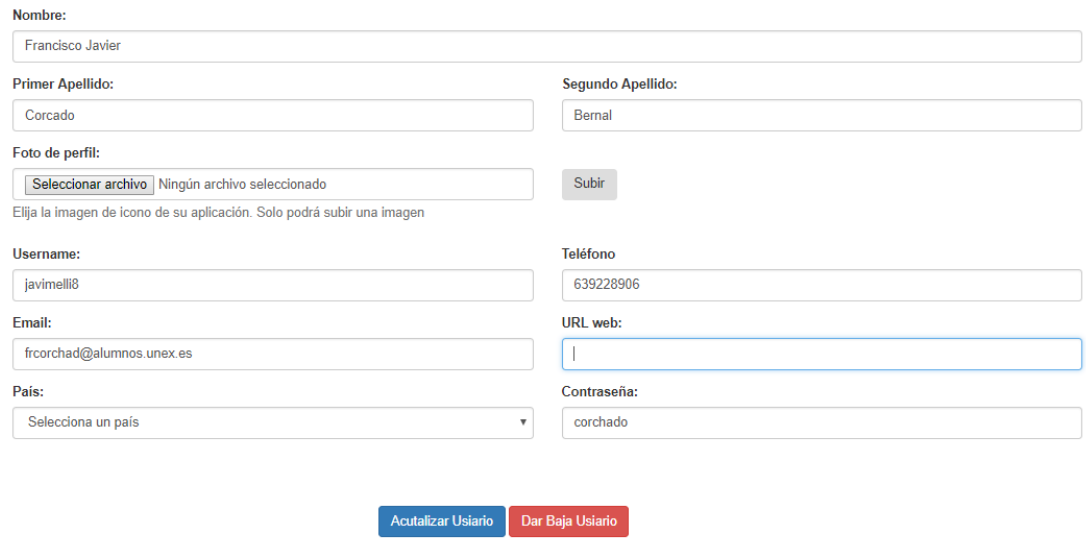


Figura 156 Datos precargados edición de perfil

En este formulario usted podrá modificar los datos de perfil y actualizarlo pulsando el botón "Actualizar Usuario". También podrá darse de baja en la aplicación pulsando el botón "Dar Baja Usuario". En este caso le aparecerá la página principal y ya pasará a ser usuario anónimo.

6.2.6. Alta aplicación

Para poder dar de alta una aplicación, deberá iniciar sesión como usuario autenticado como se explica en el apartado 6.2.4 de este manual. Una vez realizado este paso deberá pulsar el botón "Añadir app" situado en la parte inferior derecha de la página principal que se muestra en la figura 157.

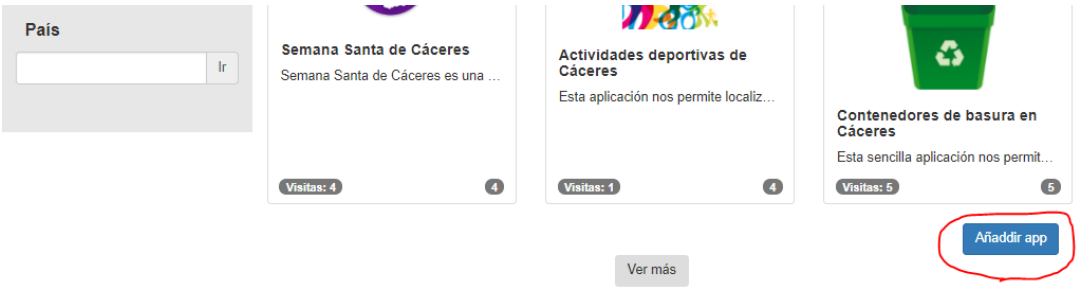


Figura 157 Botón Añadir app

Una vez pulse el botón le aparecerá un formulario como el que se muestra en la figura 158.

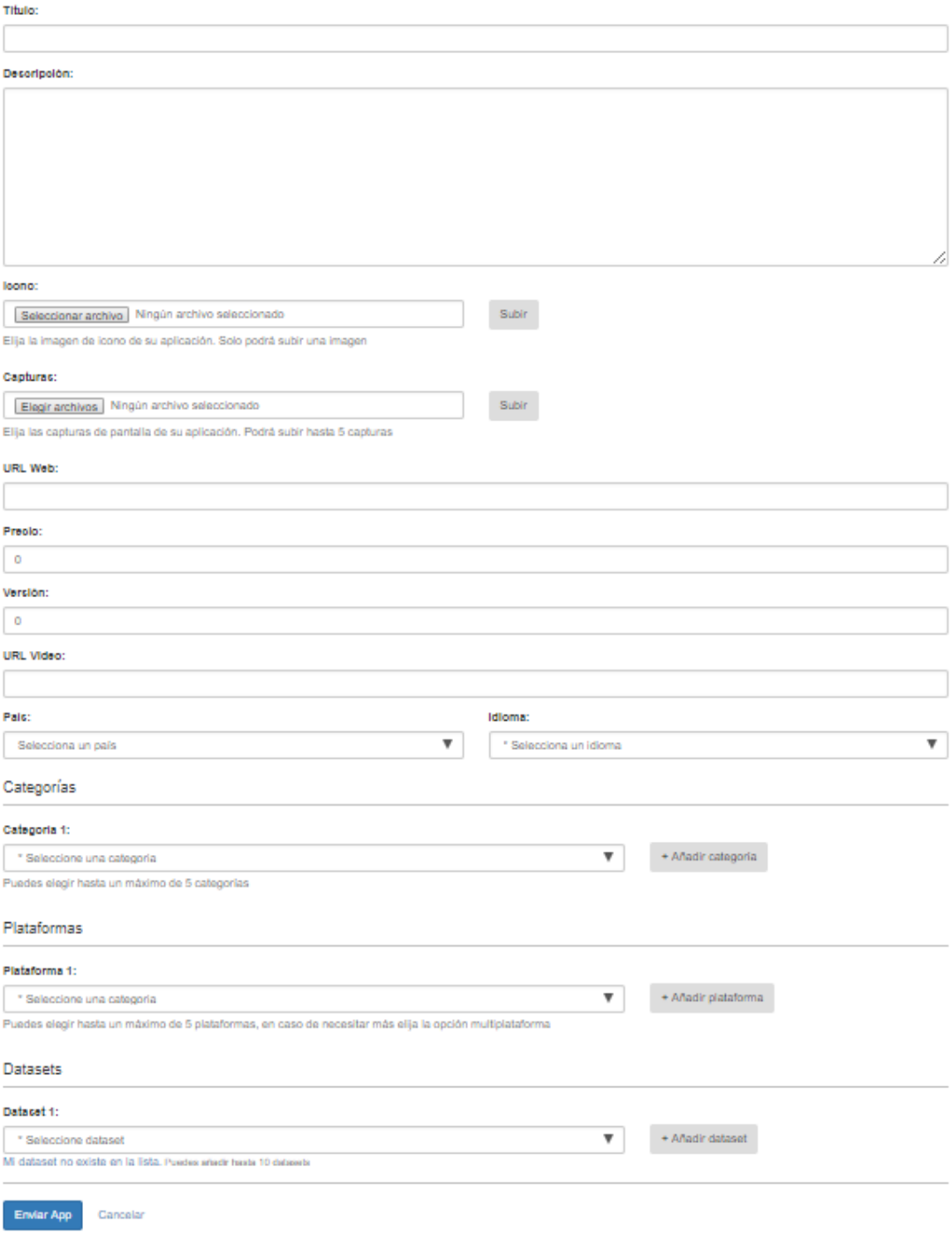


Figura 158 Formulario alta app

Este formulario está preparado para dar de alta las aplicaciones. Si se deja los datos obligatorios sin rellenar y pulsa el botón “Enviar App” se le marcará en rojo los campos que debe rellenar como se muestra en la figura 159.

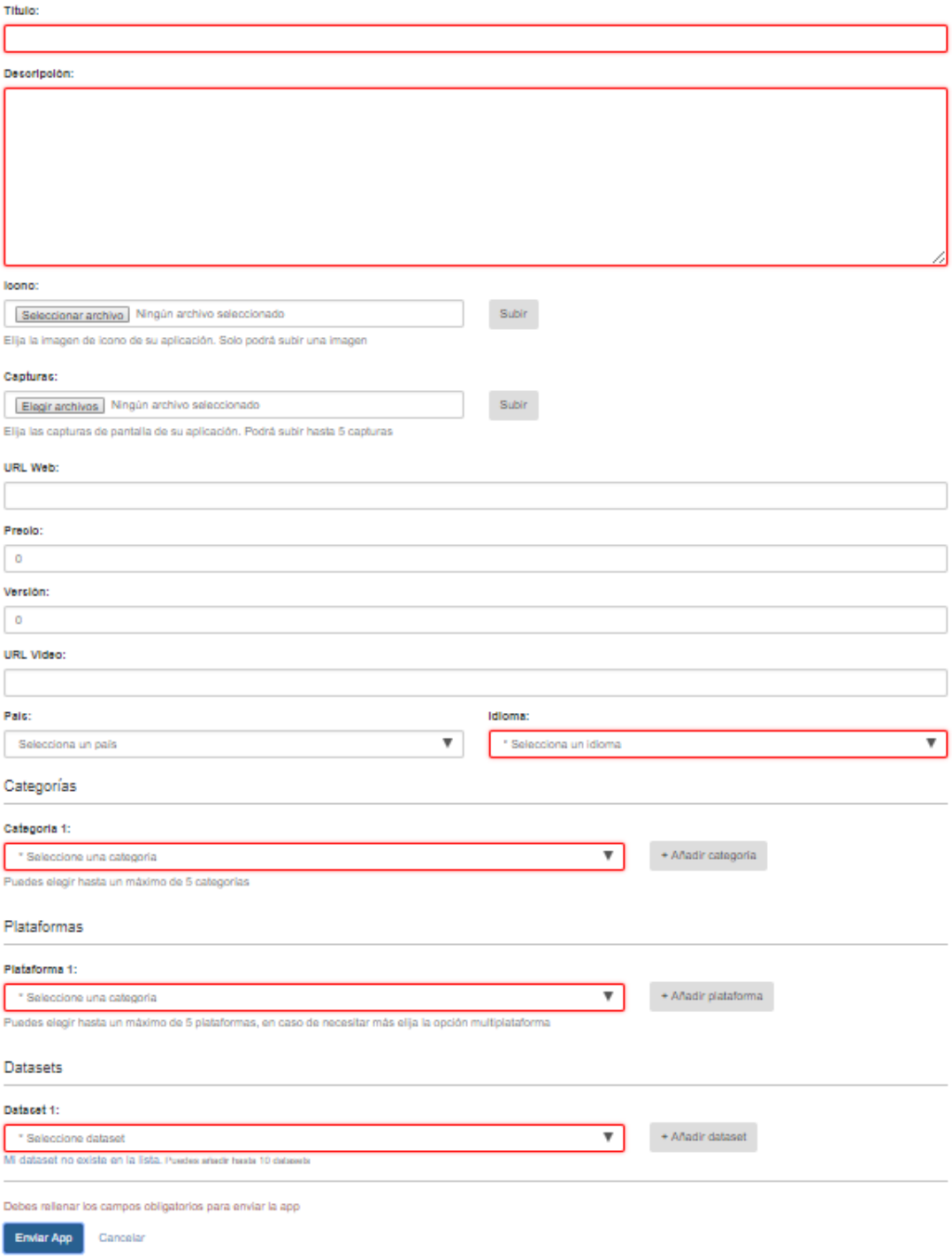


Figura 159 Validación formulario alta app

Usted solo podrá dar de alta una imagen de icono y hasta un máximo de 5 capturas. En caso de querer subir más le aparecerá un mensaje indicando el error. Ver en la figura 160.

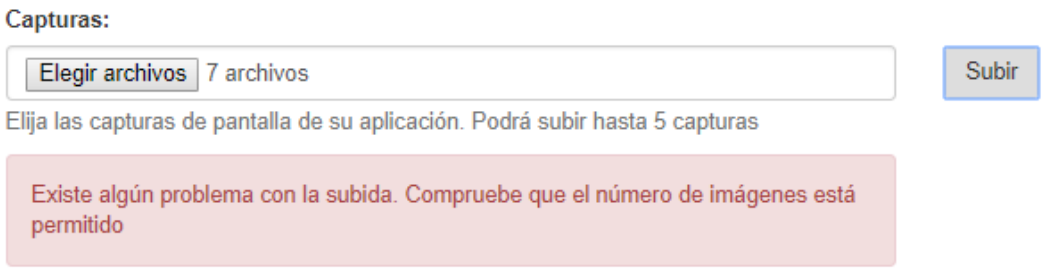


Figura 160 Error subir más capturas de las permitidas

A la hora de elegir las categorías, podrá indicar hasta un máximo de cinco. Para añadirlas o eliminarlas simplemente deberá pulsar los botones "+ Añadir categoría" y "- Eliminar categoría".

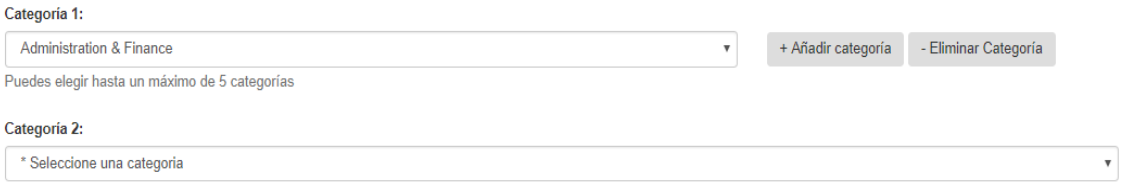


Figura 161 Añadir categoría

Para elegir las plataformas en las que se puede ejecutar la aplicación deberá pulsar los botones "+ Añadir plataforma" y "- Eliminar Plataforma", podrá elegir hasta un máximo de cinco.

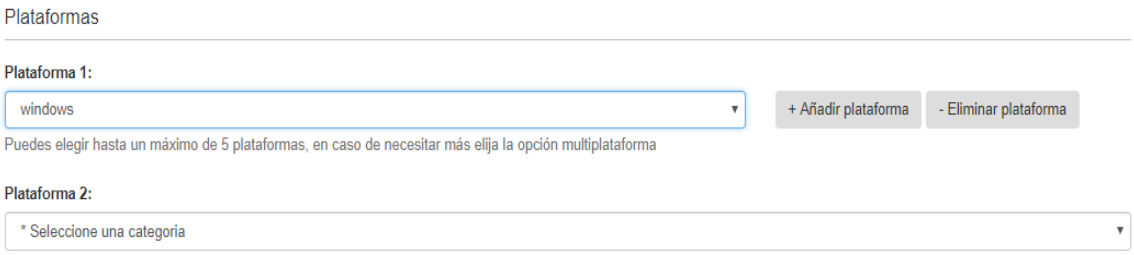


Figura 162 Añadir plataformas

Cuando usted se disponga a elegir los datasets que utiliza su aplicación, se puede encontrar con la situación de sus dataset no estén dados de alta en la aplicación. Para solucionar esta situación usted debe pulsar el texto que aparece en color azul de bajo del selector de datasets que aparece en la figura 163.

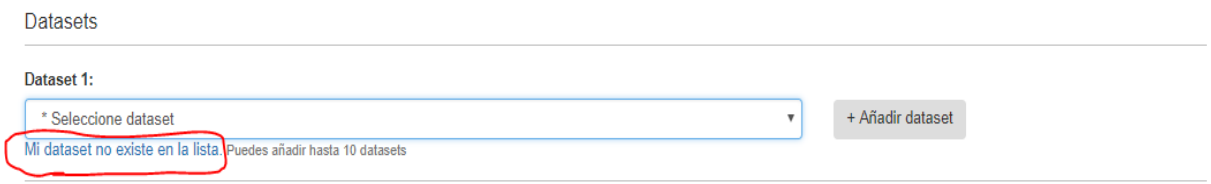


Figura 163 Mi dataset no existe

Una vez pulsado se le desplegará un formulario en el cual deberá rellenar los campos obligatorios como mínimo y dar de alta el dataset. Una vez dado de alta ya le aparecerá en los selectores de dataset automáticamente y podrá selecionarlo.

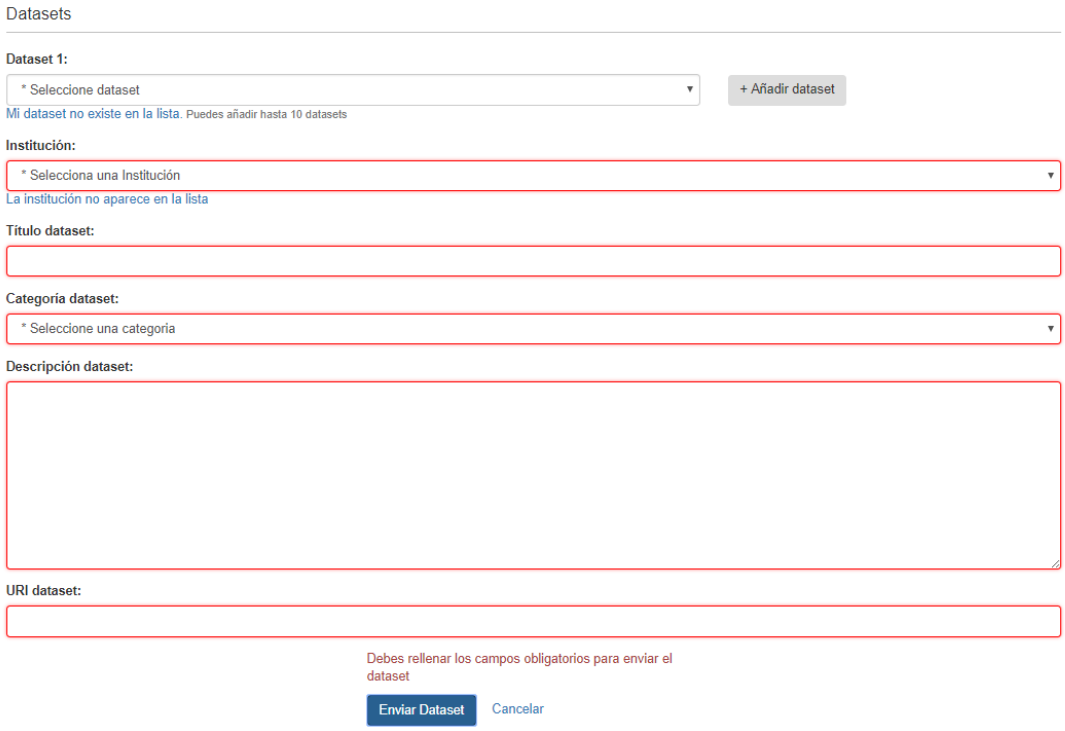


Figura 164 Formulario dataset

Dentro del formulario de dar de alta un dataset se puede encontrar con la misma situación a la hora de elegir la institución que publica los datos. Esta situación se resuelve exactamente igual que la del dataset. Usted debe pulsar el texto en azul debajo del selector de institución que aparece en la figura 165.

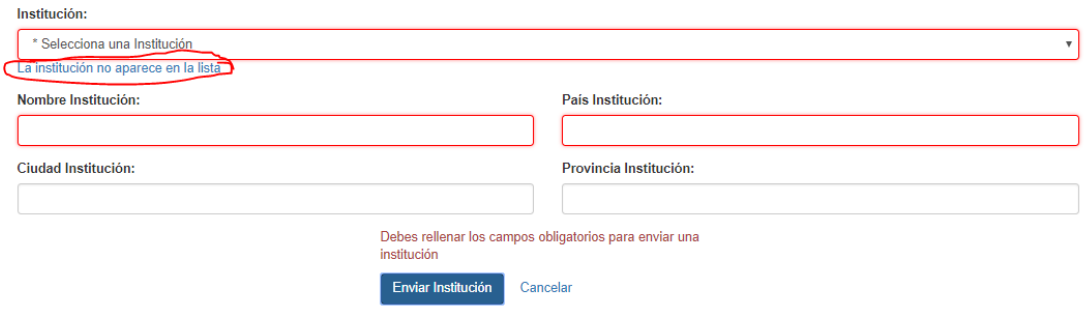


Figura 165 Formulario institución

Una vez dada de alta la institución satisfactoriamente respetando los campos obligatorios podrá seleccionar la nueva institución el selector y dar de alta el dataset.

Usted podrá seleccionar hasta un máximo de diez datasets pulsando en los botones "+ Añadir dataset" y "- Eliminar dataset".

Figura 166 Añadir dataset

Una vez relleno el formulario correctamente solo le queda pulsar el botón "Enviar App" y esperar a que le salga un mensaje de que la operación se ha realizado correctamente.

6.2.7. Editar Aplicación

Para poder editar las aplicaciones que ha dado de alta lo primero que debe hacer es seguir los pasos del apartado 6.2.4 de este manual para ser reconocido como usuario autenticado. Una vez realizado estos pasos usted deberá desplegar el desplegable situado en la parte derecha de la cabecera del portal con su nombre de usuario y pulsar sobre "Mis Apps".

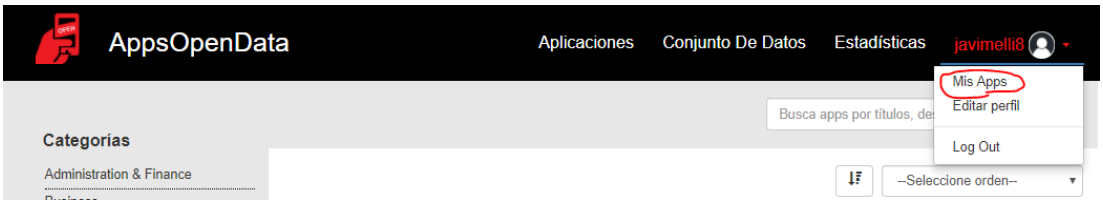


Figura 167 Mis Apps

Una vez realizado estos pasos se le abrirá una vista con un listado de las aplicaciones que ha dado de alta y un símbolo en forma de lápiz a su derecha (figura 168). Pulsando en ese símbolo le aparecerá un formulario igual al descrito en el apartado 6.2.6 de este manual, con sus mimas funcionalidades y validaciones, pero con los datos de la aplicación que ha seleccionado en el listado precargados.

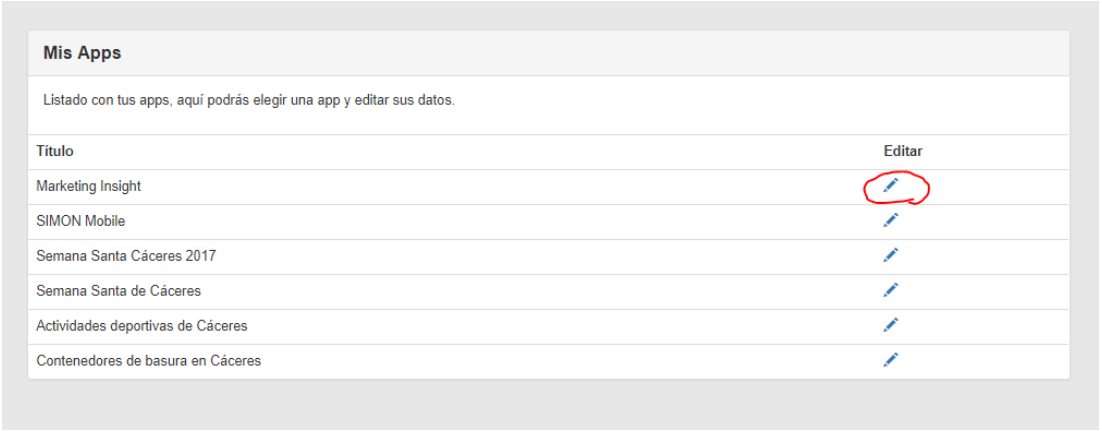


Figura 168 Listado mis apps

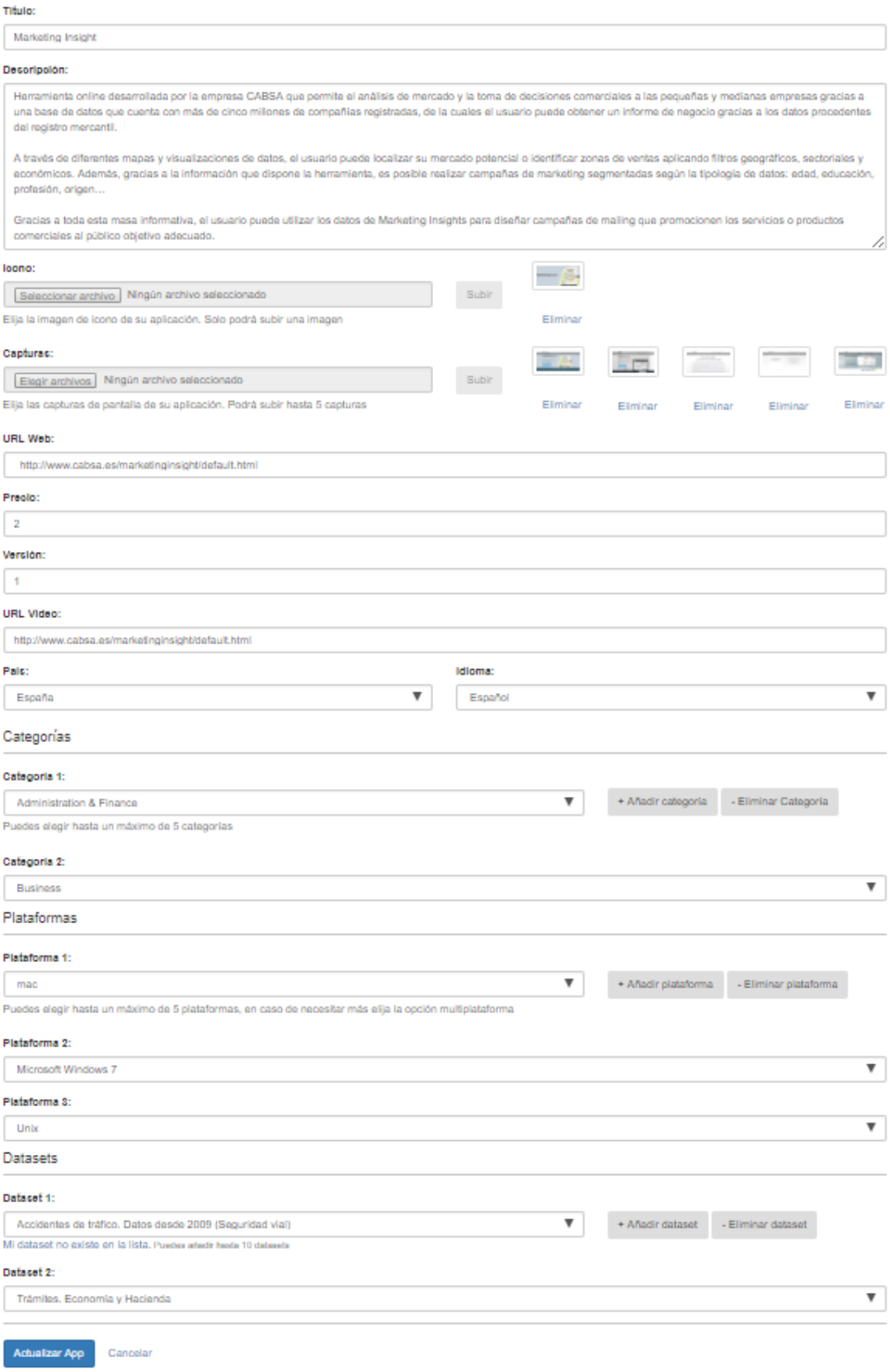


Figura 169 Formulario edición app

En este formulario usted podrá modificar todos los datos de su aplicación y eliminar y volver a subir nuevas imágenes tanto de capturas como el icono de la aplicación.

6.2.8. Valorar aplicación

Para poder valorar una aplicación usted deberá seguir los pasos que se sigue en el apartado 6.2.4 de este manual para ser usuario autenticado. Una vez realizado este paso deberá pulsar sobre la aplicación que desea valorar y en si vista de detalle aparece en la parte superior derecha una apartado para valorar la aplicación.

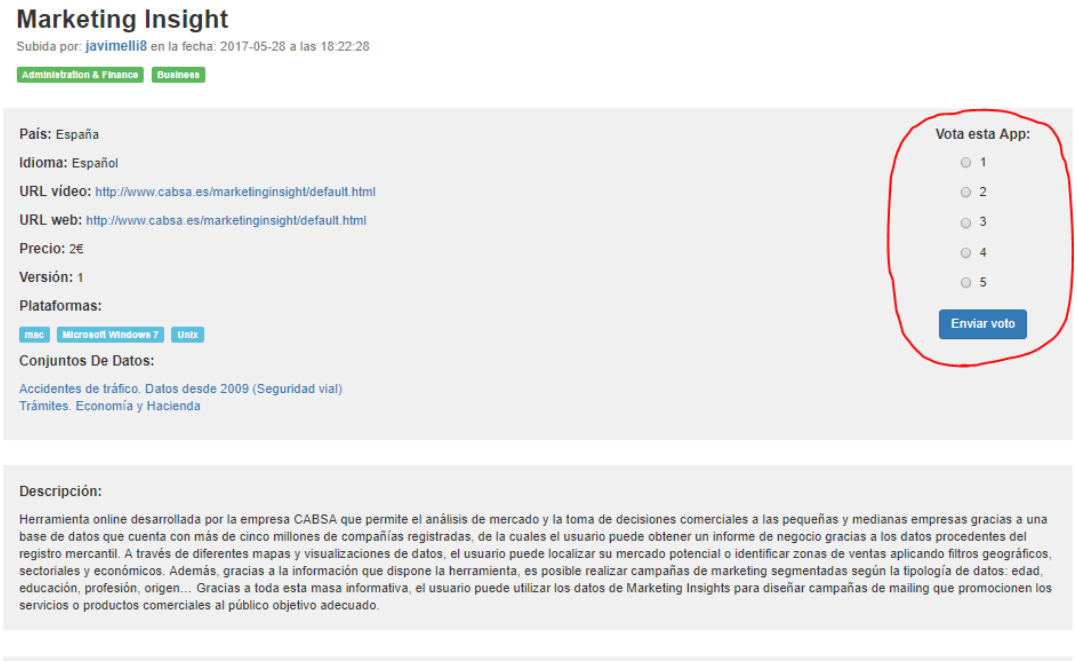


Figura 170 Valoración de aplicación

Usted debe seleccionar el valor que le quiere dar a la aplicación, pudiéndola valorar con un valor del uno al cinco. Una vez seleccionado el valor debe pulsar el botón “Enviar voto”. Si usted ya había valorado esta aplicación previamente, su voto se actualizará y en caso de ser una nueva valoración se contará y se dará de alta en la aplicación.

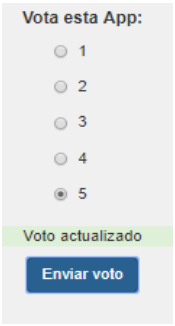


Figura 171 Voto actualizado

6.2.9. Realizar comentario a una aplicación

Para poder comentar las aplicaciones dadas de alta en el portal usted deberá seguir los pasos que se indican en el apartado 6.2.4 de este manual. Una vez sea usuario autenticado deberá pulsar sobre la aplicación que desea comentar y en si vista de detalle desplazarse al final de la página. Allí encontrará un formulario para realizar su comentario.

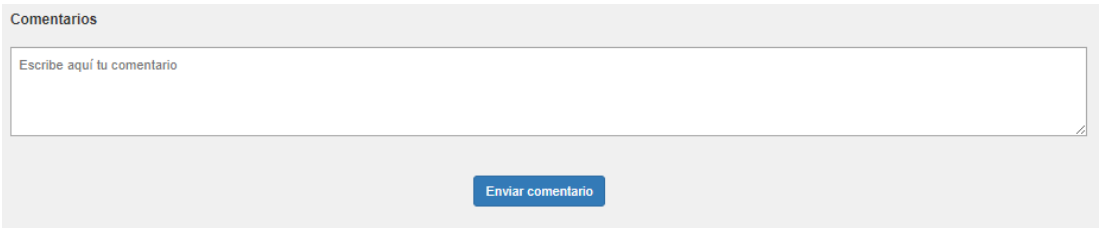


Figura 172 Insertar comentario

Una vez haya llegado a este punto lo único que deberá hacer es escribir el comentario que desee en el formulario y pulsar en el botón “Enviar comentario”. Una vez insertado aparecerá en la lista de comentarios situada justo debajo de este formulario.

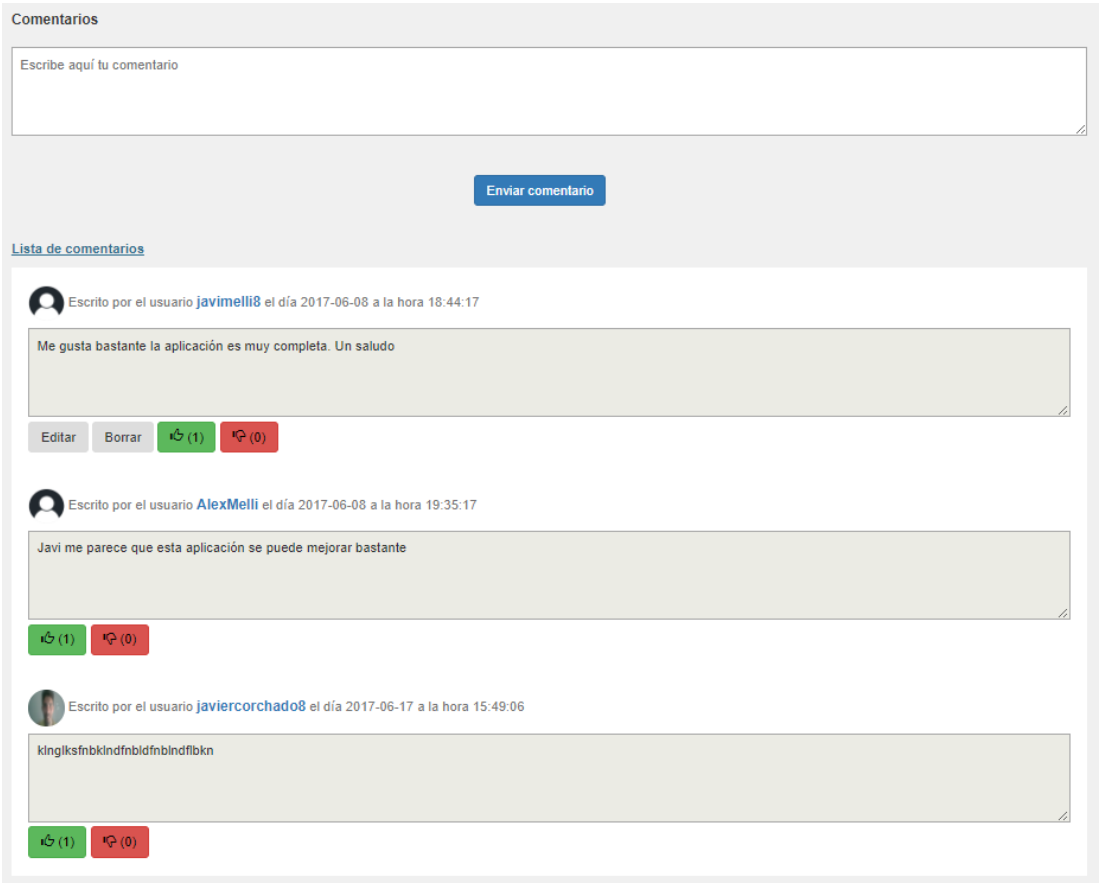


Figura 173 Lista de comentarios

6.2.10. Editar y borrar comentario

Para poder editar y eliminar comentarios que usted haya realizado sobre aplicaciones debe seguir los pasos que se le indican en el apartado 6.2.4 de este manual. Una vez sea usuario autenticado deberá pulsar sobre la aplicación donde se realizó el comentario y en su vista de detalle situarse al final de la página en la lista de comentarios.

En los comentarios que usted sea propietario aparecerán los botones “Editar” y “Borrar”. Si usted pulsa sobre el botón de “Editar” el comentario podrá ser editado y aparecerá un nuevo botón “Actualizar comentario”, pulsando sobre él se actualizará el comentario. Si en cambio usted desea borrar el comentario solo deberá pulsar el botón “Borrar” y desaparecerá de la lista de comentarios de la aplicación.

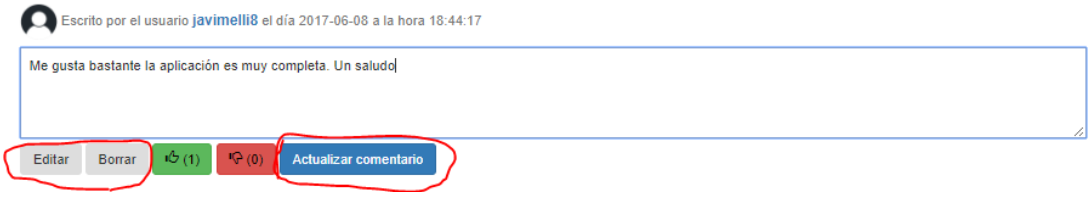


Figura 174 Edición de comentario

6.2.11. Valoración comentario

Para poder valorar un comentario usted deberá seguir los pasos que se le indican en el apartado 6.2.4 de este manual. Una vez usted sea usuario autenticado deberá pulsar sobre la aplicación que tiene el comentario que desea valorar y desplazarse al final de la página donde se encuentra la lista de comentarios de la aplicación.

En cada comentario aparecerán dos botones uno de color verde y otros de color rojo. Si pulsa sobre el botón verde estará realizando una valoración positiva del comentario, sin embargo si pulsa el botón rojo estará realizando una votación negativa.

Solo existirá una valoración por usuario. Esto quiere decir que si usted pulsa varias veces sobre el botón verde solo se contará una valoración positiva al igual que si lo hiciera con el botón rojo. Lo que sí está permitido es cambiar la valoración del comentario.

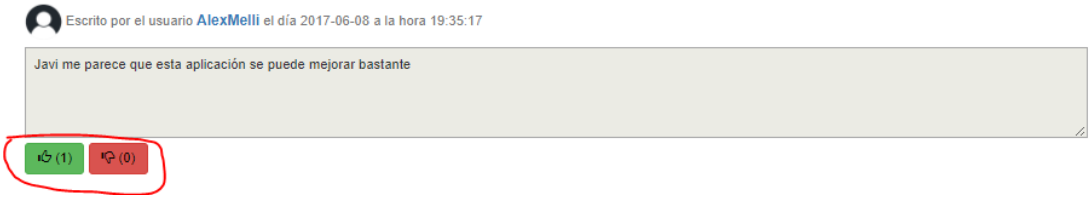


Figura 175 Valoración comentario

6.2.12. Visualización de estadísticas

Para poder visualizar tanto las estadísticas de las aplicaciones y los conjuntos de datos abiertos en la aplicación usted deberá seguir los pasos indicados en el apartado 6.2.4 de este manual. Una vez pase a ser usuario autenticado deberá pulsar en el apartado estadísticas que se encuentra en la cabecera superior de portal.

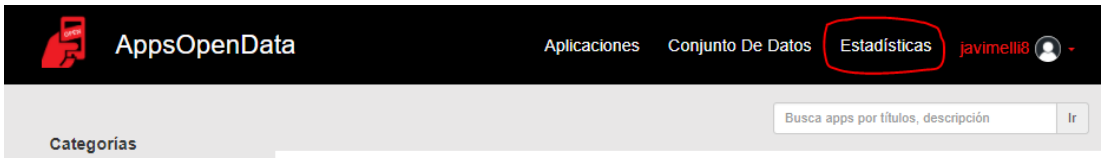


Figura 176 Cabecera estadísticas

Una vez pulse en esa sección le aparecerá una vista como la que se muestra en la figura 178.

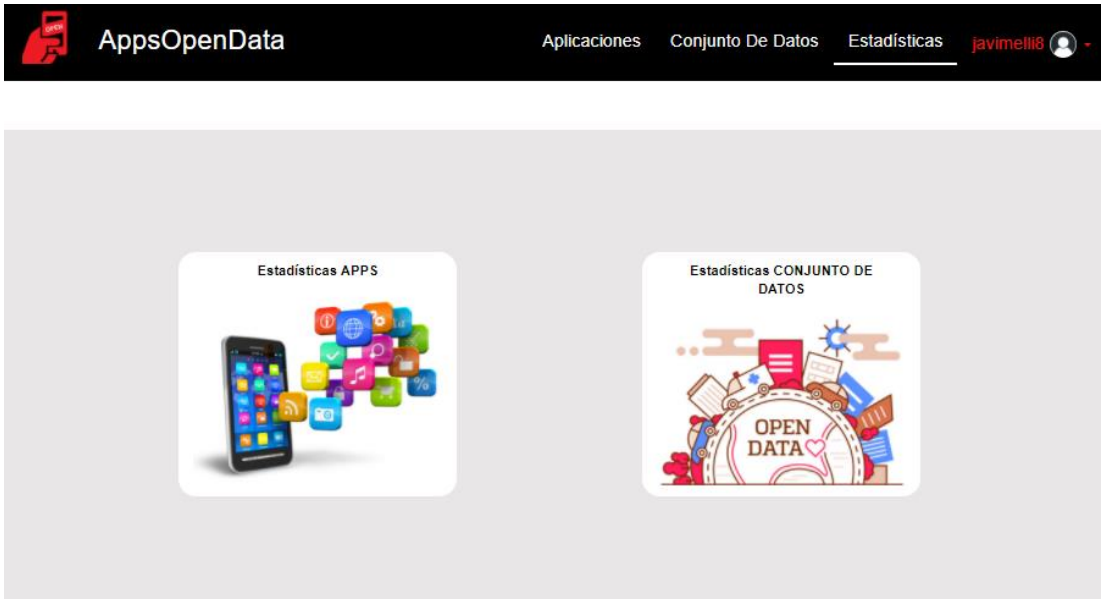
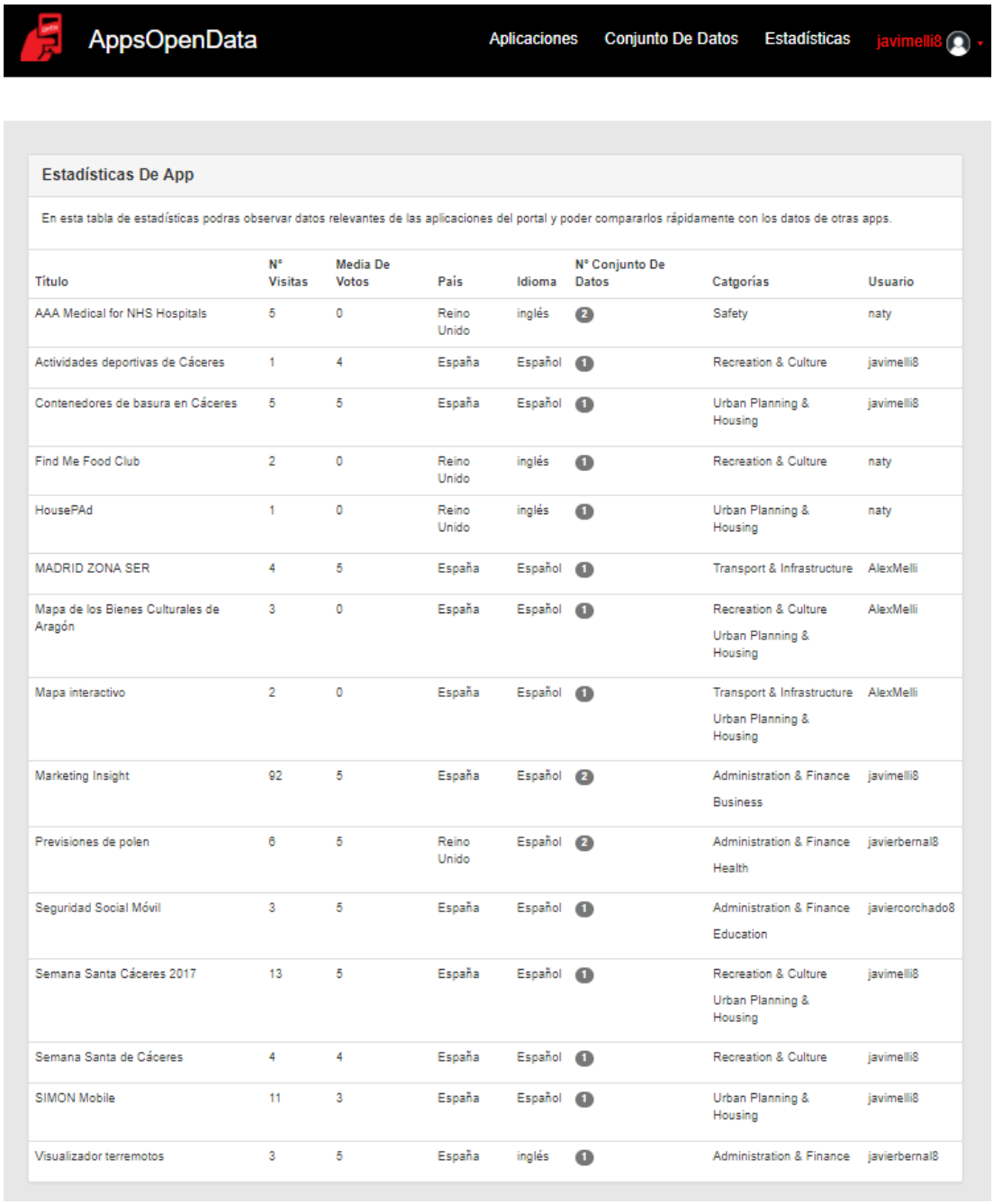


Figura 177 Elección de estadísticas

Si usted desea visualizar las estadísticas de las aplicaciones del portal debe pulsar sobre la imagen que contiene el texto “Estadísticas APPS”. Una vez pulsada esta imagen le aparecerá la vista de la figura 178.



Título	N° Visitas	Media De Votos	País	Idioma	N° Conjunto De Datos	Catgorías	Usuario
AAA Medical for NHS Hospitals	5	0	Reino Unido	inglés	2	Safety	naty
Actividades deportivas de Cáceres	1	4	España	Español	1	Recreation & Culture	javimelli8
Contenedores de basura en Cáceres	5	5	España	Español	1	Urban Planning & Housing	javimelli8
Find Me Food Club	2	0	Reino Unido	inglés	1	Recreation & Culture	naty
HousePAd	1	0	Reino Unido	inglés	1	Urban Planning & Housing	naty
MADRID ZONA SER	4	5	España	Español	1	Transport & Infrastructure	AlexMelli
Mapa de los Bienes Culturales de Aragón	3	0	España	Español	1	Recreation & Culture Urban Planning & Housing	AlexMelli
Mapa interactivo	2	0	España	Español	1	Transport & Infrastructure Urban Planning & Housing	AlexMelli
Marketing Insight	92	5	España	Español	2	Administration & Finance Business	javimelli8
Previsiones de polen	6	5	Reino Unido	Español	2	Administration & Finance Health	javierbernal8
Seguridad Social Móvil	3	5	España	Español	1	Administration & Finance Education	javiercorchado8
Semana Santa Cáceres 2017	13	5	España	Español	1	Recreation & Culture Urban Planning & Housing	javimelli8
Semana Santa de Cáceres	4	4	España	Español	1	Recreation & Culture	javimelli8
SIMON Mobile	11	3	España	Español	1	Urban Planning & Housing	javimelli8
Visualizador terremotos	3	5	España	inglés	1	Administration & Finance	javierbernal8

Figura 178 Estadísticas aplicaciones

Con el fin de que usted pueda realizar una búsqueda y una comparación rápida y efectiva entre las aplicaciones dadas de alta en el portal, se facilita este listado ordenado por orden alfabético según el título de las aplicaciones. Se muestra información relevante como el número de visitas, media de las valoraciones, categorías y el dato seguramente más importante que es el número de conjunto de datos abiertos que utiliza. Este último dato se ha destacado con un sombreado respecto a los demás.

Si usted desea visualizar las estadísticas de los conjuntos de datos abiertos dados de alta en la aplicación debe situarse en la vista que muestra la figura 177 de este apartado y pulsar sobre la imagen que contiene el texto “Estadísticas CONJUNTOS DE DATOS”. Pulsando sobre esta imagen le aparecerá una vista como la de la figura 179.

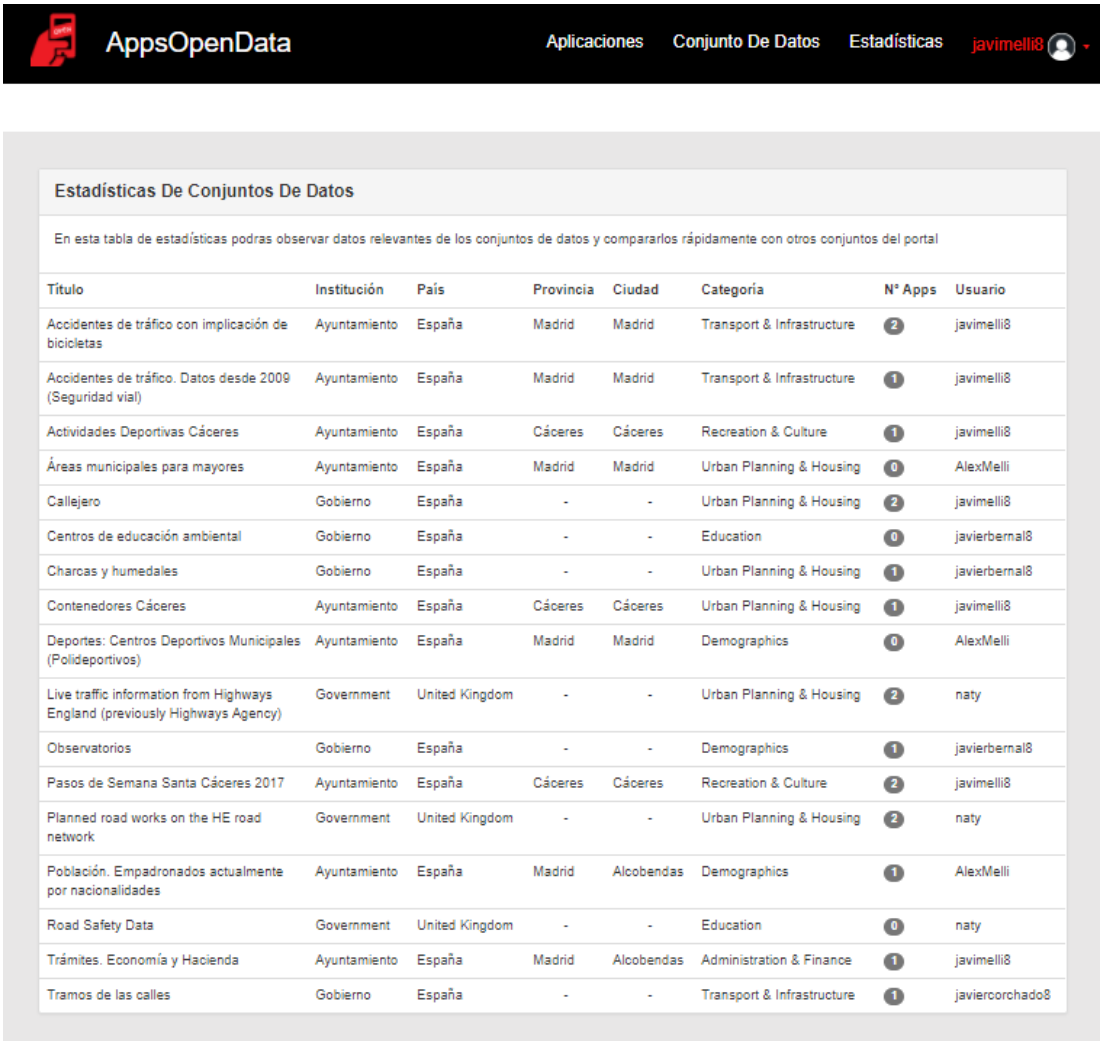


Figura 179 Estadísticas conjuntos de datos

Aparecerá un listado en el que usted podrá realizar una búsqueda rápida de los conjuntos de datos que desea consultar, ya que el listado está ordenado alfabéticamente por el título de los datasets. Se muestra información muy relevante entre la que hay que destacar el número de aplicaciones que utiliza el dataset. Este dato tiene un sombreado para destacarlo del resto de la información.

7. Posibles mejoras

En este apartado se van a comentar algunas posibles mejoras para futuras versiones:.

- Teniendo en cuenta que hoy en día la gran mayoría de las personas utilizan el móvil para navegar por la web, podría ser aconsejable realizar una versión optimizada del portal para móviles. Así, podría realizarse un diseño adaptable 100% para dispositivos móviles, realizar una carga perezosa de imágenes y reducir el número de librerías que se cargan.
- Para los listados que se muestran en las estadísticas tanto de aplicaciones como de datasets, podría implementarse una exportación de datos a algún documento tipo JSON, CSV, ODS o XLS.
- Se podría realizar una auditoría de seguridad con el fin de encontrar posibles fallos de seguridad en la aplicación y solucionarlos.
- Desplegar la aplicación en un servidor con servicio HTTPS para que la comunicación entre el cliente y el servidor esté cifrada.
- Implementar funcionalidades de comentarios y valoraciones de los datasets.
- Realizar un estudio SEO e implantarlo en la aplicación para conseguir un buen posicionamiento entre los buscadores de internet.
- Implementar robots captcha que controle la inserción por parte de posibles robots maliciosos desarrollados por usuarios no deseados.

8. Conclusiones

En este capítulo se describirán las conclusiones obtenidas después de haber afrontado este trabajo fin de grado.

8.1. Objetivos

En este apartado se va a analizar si se ha cumplido con los objetivos que se marcaron inicialmente para este proyecto.

Así, en cuanto a los objetivos que se marcaron con respecto al estudio de los portales se han cumplido con los siguientes:

- Analizar su arquitectura
- Estudiar qué características y detalles ofrecen sobre las aplicaciones que contienen
- Identificar qué propiedades serían deseables en estos portales y si las aplicaciones que van a contener trabajan con datos abiertos.

En cuanto al diseño del portal se ha cumplido con los siguientes objetivos.

- Tomar como base el estudio realizado como primer objetivo.
- Analizar las distintas tecnologías con las que poder construir la solución y usar las que mejor se acomodan a los requisitos del nuevo portal para su desarrollo.

Teniendo en cuenta los objetivos que se marcaron en el apartado 1.6 de este documento se puede comprobar que se ha cumplido con todos los objetivos marcados al inicio.

Por último, se ha desarrollado el portal cumpliendo con todos los requisitos y casos de uso que se definieron en la fase de análisis y diseño.

8.2. Conclusiones personales

A nivel personal, estoy muy satisfecho con la construcción de este portal web. Después de realizar este trabajo puedo asegurar que tanto la elección del TFG como la del tutor han sido totalmente acertadas.

Una de las motivaciones más grande que tenía para realizar este portal era aumentar mis conocimientos sobre el desarrollo web y puedo asegurar que se ha conseguido. Así, además de haber reforzado considerablemente los conocimientos previos adquiridos durante el estudio del Grado, la experiencia obtenida durante la realización de este TFG ha aumentado mis conocimientos sobre tecnologías de última generación dentro de este campo como son Angular2 y Bootstrap. Como valoración global me ha parecido un trabajo bastante completo ya que ha implicado construir una aplicación desde cero, pasando por todas las fases del ciclo de vida de un proyecto software. Esto ha implicado que he tenido que aplicar y utilizar muchos conocimientos adquiridos en las asignaturas del Grado como por ejemplo el ciclo de vida de un proyecto software, programación, redes para la comunicación servidor/cliente, expresiones regulares, etc.

También he aprendido a usar mejor los patrones de diseño y a valorar aún más la importancia de su uso, especialmente en proyectos complejos.

Como reflexión final, este trabajo fin de grado me ha servido para adquirir nuevas competencias y reforzar conocimientos, lo cual espero que haga que mis desarrollos futuros sean de más calidad. Esto lo agradezco tanto desde el punto de vista personal como laboral.

9. Anexos

En este apartado se van a mostrar imágenes de las hojas de cálculo que se realizaron durante el estudio de los portales web.

En las figuras 180, 181 y 182 se muestra la hoja de cálculo que se utilizó para analizar la información que se solicita para dar de alta una aplicación en los portales estudiados.

Información / Repositorios	Google Play Store	Tienda De Windows	App store	Chrome web store	data.gov	data.gov.uk	datos.gob	APP	Descripción
Pagar Cuota	SI	SI	SI	SI	NO	NO	NO	NO	Es la cuota que se debe pagar como desarrollador y poder publicar las apps en los repositorios.
Nombre de la persona	SI	SI	SI	SI	NO	NO	SI	SI	Nombre de la persona que se va a dar de alta como de desarrollador.
Dirección	SI	SI	SI	SI	NO	NO	NO	NO	Dirección de la persona que será el desarrollador.
Código postal	SI	SI	SI	SI	NO	NO	NO	NO	Código postal de la persona que va a ser el desarrollador.
Provincia	SI	SI	SI	SI	SI	NO	NO	NO	Provincia de la persona que va a ser el desarrollador.
Ciudad	SI	SI	SI	SI	SI	NO	NO	NO	Ciudad de la persona que va a ser el desarrollador.
Nº tarjeta de crédito	SI	SI	SI	SI	NO	NO	NO	NO	Nº tarjeta de crédito de la que será extraida la cuota de desarrollador.
Titular tarjeta crédito	SI	SI	SI	SI	NO	NO	NO	NO	Titular de la tarjeta de crédito con la que se realizará el pago de la cuota de desarrollador.
Nombre desarrollador o anunciante	SI	SI	SI	SI	NO	SI	SI	SI	Nombre público con el que será conocido el desarrollador por los usuarios del repositorio.
Página web	SI	NO	SI	SI	SI	SI	SI	SI	Página web con contenido referente al desarrollador o a la aplicación. Puede ser un enlace a la aplicación como tal si es web o a el repositorio donde adquirir la aplicación por ejemplo play store.
Correo electrónico	SI	SI	SI	SI	NO	SI	SI	SI	Correo electrónico del desarrollador.
Teléfono	SI	SI	SI	NO	NO	NO	SI	SI	Teléfono del desarrollador.
País de la cuenta	SI	SI	SI	SI	SI	NO	SI	SI	Páis del desarrollador.
Idioma	SI	SI	SI	SI	NO	NO	NO	SI	Idioma de la aplicación.
Título	SI	SI	SI	SI	SI	SI	SI	SI	Título de la aplicación.
Descripción breve	SI	NO	NO	NO	NO	NO	SI	SI	Descripción breve sobre el contenido y el funcionamiento de la aplicación.

Figura 180 Información solicitada 1

Descripción completa	SI	SI	SI	SI	SI	SI	NO	NO	Descripción amplia del contenido y el funcionamiento de la aplicación.
Descripción de la última actualización	NO	SI	NO	NO	NO	NO	NO	NO	Descripción de las nuevas funcionalidades y el nuevo contenido de la última actualización o versión de la aplicación.
Palabras clave	NO	SI	NO	NO	NO	SI	NO	NO	Palabras claves que se asocian al contenido de la aplicación.
Capturas de pantalla	SI	SI	SI	SI	NO	SI	NO	SI	Capturas de pantalla de la aplicación en funcionamiento. Se pueden pedir varios tamaño para los distintos dispositivos.
Icono	SI	SI	SI	SI	SI	SI	SI	SI	Icono de la aplicación. Se pueden pedir varios tamaños para los distintos dispositivos donde puede funcionar la aplicación.
Imagen destacada	SI	NO	NO	NO	NO	NO	NO	NO	Imagen destacada de la aplicación.
Enlace a vídeo promocional	SI	NO	SI	SI	NO	NO	NO	SI	Enlace hacia un video que haga publicidad obre nuestra aplicación.
Tipo de aplicación	SI	SI	SI	SI	NO	NO	NO	NO	En este caso se refiere a si es una aplicación o un juego.
Categoría	SI	SI	SI	SI	NO	SI	SI	SI	Indica la categoría y subcategoría a la que pertenece la aplicación.
Calificación del contenido	SI	NO	SI	NO	NO	NO	NO	NO	Se refiere a calificar el contenido según su madurez, es decir, para que personas es apto dependiendo de su contenido.
Política de privacidad	SI	NO	NO	SI	NO	NO	NO	NO	Política de privacidad, en caso de tenerla.
Precio o gratis	SI	SI	SI	SI	NO	SI	SI	SI	El precio que tiene bajar e instalar la aplicación en un dispositivo. Puede cambiar según el país en el que se distribuya.

Figura 181 Información solicitada 2

Países de distribución	SI	SI	SI	SI	NO	NO	NO	NO	Países en los que se distribuirá la aplicación.
ID de aplicación	NO	NO	SI	SI	NO	NO	NO	NO	ID que identifica de manera única la aplicación.
Certificado de producción	NO	NO	SI	NO	NO	NO	NO	NO	Certificado de seguridad de la aplicación.
Código de la aplicación	NO	NO	NO	NO	NO	NO	NO	NO	Código que implementa la aplicación. (En caso de ser de código libre)
Ejecutable de la aplicación	SI	SI	SI	SI	NO	NO	NO	NO	Ejecutable de la aplicación, el build que se debe distribuir para poder instalarla y ejecutarla en los distintos dispositivos. War, jar, puede ser un zip también como en caso de chrome…
Versión	NO	NO	SI	SI	NO	NO	NO	SI	La versión de la aplicación.
Plataforma de la aplicación	NO	NO	NO	NO	NO	NO	SI	SI	Plataforma en la que se debe ejecutar la aplicación.
Identificador de recursos uniforme URI de los data set utilizados	NO	NO	NO	NO	SI	SI	NO	SI	La URI que da acceso a los datos utilizados en la aplicación.
Descripción de los data set utilizados	NO	NO	NO	NO	NO	NO	NO	SI	Descripción de los datos de los data set utilizados en la aplicación.
Muestra estadísticas del uso de los datos abiertos utilizados por las	NO	NO	NO	NO	NO	SI	NO	SI	
*NOTA: OpenCity Apps no aparece en esta lista ya que es un grupo que se dedica a construir aplicaciones que utilizan datos abiertos para ayudar al gobierno de									

Figura 182 Información solicitada 3

En las figuras 183, 184 y 185 se muestra la hoja de cálculo que se utilizó para analizar la información que se muestra sobre las aplicaciones en los portales estudiados.

Información / Repositorios	Google Play Store	Tienda De Windows	App store	Chrome web store	OpenCity Apps	data.gov	data.gov.uk	datos.gob	Descripción
Pagar Cuota	NO	NO	NO	NO	NO	NO	NO	NO	Es la cuota que se debe pagar como desarrollador y poder publicar las apps en los repositorios.
Nombre de la persona	NO	NO	NO	NO	NO	NO	NO	NO	Nombre de la persona que se va a dar de alta como de desarrollador.
Dirección	NO	NO	NO	NO	NO	NO	NO	NO	Dirección de la persona que será el desarrollador.
Código postal	NO	NO	NO	NO	NO	NO	NO	NO	Código postal de la persona que va a ser el desarrollador.
Provincia	NO	NO	NO	NO	NO	NO	NO	NO	Provincia de la persona que va a ser el desarrollador.
Ciudad	NO	NO	NO	NO	NO	NO	NO	NO	Ciudad de la persona que va a ser el desarrollador.
Nº tarjeta de crédito	NO	NO	NO	NO	NO	NO	NO	NO	Nº tarjeta de crédito de la que será extraida la cuota de desarrollador.
Titular tarjeta crédito	NO	NO	NO	NO	NO	NO	NO	NO	Titular de la tarjeta de crédito con la que se realizará el pago de la cuota de desarrollador.
Nombre desarrollador/es o anunciante	SI	SI	SI	SI	SI	NO	SI	SI	Nombre público con el que será conocido el desarrollador por los usuarios del repositorio.
Página web	SI	NO	SI	SI	SI	SI	SI	SI	Página web con contenido referente al desarrollador o a la aplicación. Puede ser un enlace a la aplicación como tal si es web o a el repositorio donde adquirir la aplicación por ejemplo play store.
Correo electrónico	NO	NO	NO	NO	NO	NO	NO	NO	Correo electrónico del desarrollador.
Teléfono	NO	NO	NO	NO	NO	NO	NO	NO	Teléfono del desarrollador.
País de la cuenta	NO	NO	NO	NO	NO	NO	NO	NO	Páis del desarrollador.
Idioma	SI	SI	SI	SI	SI	NO	NO	NO	Idioma de la aplicación.
Título	SI	SI	SI	SI	SI	SI	SI	SI	Título de la aplicación.
Descripción breve	SI	NO	NO	No	NO	NO	NO	SI	Descripción breve sobre el contenido y el funcionamiento de la aplicación.
Descripción completa	SI	SI	SI	SI	SI	SI	SI	NO	Descripción amplia del contenido y el funcionamiento de la aplicación.

Figura 183 Información mostrada 1

Descripción de la última actualización	NO	SI	NO	NO	NO	NO	NO	NO	Descripción de las nuevas funcionalidades y el nuevo contenido de la última actualización o versión de la aplicación.
Palabras claves	NO	SI	NO	NO	NO	NO	SI	NO	Palabras claves que se asocian al contenido de la aplicación.
Capturas de pantalla	SI	SI	SI	SI	SI	NO	SI	NO	Capturas de pantalla de la aplicación en funcionamiento. Se pueden pedir varios tamaño para los distintos dispositivos.
Icono	SI	SI	SI	SI	NO	SI	SI	SI	Icono de la aplicación. Se pueden pedir varios tamaños para los distintos dispositivos donde puede funcionar la aplicación.
Imagen destacada	SI	NO	NO	NO	NO	NO	NO	NO	Imagen destacada de la aplicación.
Enlace a vídeo promocional	SI	NO	SI	SI	NO	NO	NO	NO	Enlace hacia un video que haga publicidad obre nuestra aplicación.
Tipo de aplicación	SI	SI	SI	SI	NO	NO	NO	NO	En este caso se refiere a si es una aplicación o un juego.
Categoría	SI	SI	SI	SI	NO	NO	SI	SI	Indica la categoría y subcategoría a la que pertenece la aplicación.
Calificación del contenido	SI	NO	SI	NO	NO	NO	NO	NO	Se refiere a calificar el contenido según su madurez, es decir, para que personas es apto dependiendo de su contenido.
Política de privacidad	NO	NO	NO	NO	NO	NO	NO	NO	Política de privacidad, en caso de tenerla.
Precio o gratis	SI	SI	SI	SI	SI	NO	SI	SI	El precio que tiene bajar e instalar la aplicación en un dispositivo. Puede cambiar según el país en el que se distribuya.
Países de distribución	SI	SI	SI	SI	NO	NO	NO	NO	Países en los que se distribuirá la aplicación.
ID de la aplicación	NO	NO	NO	NO	NO	NO	NO	NO	ID que identifica de manera única la aplicación.
Certificado de producción	NO	NO	NO	NO	NO	NO	NO	NO	Certificado de seguridad de la aplicación.
Código de la aplicación	NO	NO	NO	NO	SI	NO	NO	NO	Código que implementa la aplicación. (En caso de ser de código libre)

Figura 184 Información mostrada 2

Ejecutable de la aplicación	NO	NO	NO	NO	NO	NO	NO	NO	Ejecutable de la aplicación, el build que se debe distribuir para poder instalarla y ejecutarla en los distintos dispositivos. War, jar, puede ser un zip también como en caso de chrome…
Versión	NO	NO	SI	SI	NO	NO	NO	NO	La versión de la aplicación.
Plataforma de la aplicación	NO	NO	NO	NO	NO	NO	NO	SI	Plataforma en la que se debe ejecutar la aplicación.
Identificador de recursos uniforme URI de los data set utilizados	NO	NO	NO	NO	NO	SI	SI	NO	La URI que da acceso a los datos utilizados en la aplicación.
Descripción de los data set utilizados	NO	NO	NO	NO	NO	NO	NO	SI	Descripción de los datos de los data set utilizados en la aplicación.

Figura 185Información mostrada 3

En las figuras 186, 187, 188, 189, 190 y 191 se muestra la hoja de cálculo utilizada para estudiar la categorización de aplicaciones que se realiza en los portales web estudiados.

Categorias / Repositorios	Google Play Store	Tienda De Windows	App Store	Chrome web store	data.gov.uk	datos.gob	Descripción
General							
Bares y restaurantes	No	Si	NO	No	No	No	Incluye todas la categorías que categorizan aplicaciones relacionadas con restaurantes, bares, pub, cafeterías…
Bebida comida	NO	NO	SI	SI	No	No	Incluye todas la categorías que categorizan las aplicaciones relacionadas con la alimentación.
Bibliotecas y demos	SI	No	NO	NO	No	No	Es una categoría especial que solo tiene google play store.
Catálogos	NO	NO	SI	No	No	No	Categoriza aplicaciones relacionadas con cátalogos de distintos temas.
Cómics	Si	No	NO	No	No	No	Categoriza aplicaciones relacionadas con cómics.
Compras	SI	Si	SI	SI	No	No	Categoriza aplicaciones relacionas con temas de compras.
Comunicación	SI	No	NO	SI	No	No	Aplicaciones que son utilizadas para establecer algún tipo de comunicación.
Deportes	SI	Si	SI	SI	No	No	Aplicaciones relacionadas con el deporte.
Diseño multimedia	No	Si	NO	No	No	No	Categoriza aplicaciones utilizadas en el diseño multimedia.
Educación	SI	Si	SI	SI	SI	No	Aplicaciones relacionadas con la educación.
Empresa	SI	Si	NO	SI	No	No	Engloba todas la aplicaciones relacionadas con el tema de empresas. Herramientas que son útiles para ellas.

Figura 186 Categorización 1

Entretenimiento	SI	SI	SI	SI	No	No	Engloba topdas las aplicaciones que tienen como fin el entretenimiento independientemente el tema que tratan. También se le puede llamar ocio.
Estilo de vida	SI	Si	SI	SI	No	No	Categoriza todas la aplicaciones que tienen que ver con el estilo de vida de los usuarios. Por ejemplo donde suelen comer, comprar ropa…
Finanzas	SI	Si	SI	SI	SI	SI	Aplicaciones relacionadas con las finanzas y la economía.
Fotografía	SI	Si	SI	SI	No	No	Aplicaciones relacionadas con la fotografía.
Gobierno y política	No	Si	NO	No	SI	No	Aplicaciones relacionadas con la política en general.
Herramientas	SI	Si	SI	SI	No	No	Engloba todas la categorías que categorizan las aplicaciones relacionadas con herramientas o utilidades.
Heramientas de desarrollo	No	Si	NO	SI	No	No	Engloba las apliciones utilizadas para el desarrollo.
Libros y obras de consulta	SI	Si	SI	SI	No	No	Engloba todas las aplicaciones relacionadas con libros, referencias, enciclopedias y consultas.
Mapas y navegación	SI	Si	SI	SI	SI	No	Aplicaciones relacionadas con mapas, ubicación y navegación.
Medicina	SI	Si	SI	No	No	No	Aplicaciones relacionadas con la medicina.
Medio ambiente	NO	NO	NO	NO	SI	SI	Aplicaciones que tratan el tema relacionados con el medio ambiente.

Figura 187Categorización 2

Música y audio	SI	Si	SI	SI	No	No	Aplicaciones relacionadas con la música y el audio.
Negocios	NO	NO	SI	SI	No	No	Aplicaciones relacionadas con los negocios, marketing, relación con el cliente, temas jurídicos...
Noticias y revistas	SI	si	SI	SI	No	No	Aplicaciones relacionada con la prensa y las revistas.
Películas, cine y TV	SI	SI	SI	SI	No	No	Apartado para las aplicaciones relacionadas con el cine, películas y televisión.
Personalización	SI	Si	NO	No	No	No	Aplicaciones relacionadas con el tema de personalización de imágenes, fondos, móvil…
Productividad	SI	Si	SI	SI	No	No	Aplicaciones relacionadas con temas de productividad como por ejemplo las herramientas de oficina.
Reproductores y editores de vídeo	SI	Si	SI	SI	No	No	Aplicaciones relacionadas con la reproducción y edicción de videos.
Salud y bienestar	SI	Si	SI	SI	SI	SI	Aplicaciones relacionadas son la salud y el bienestar
Sector público	NO	NO	NO	NO	NO	SI	Aplicaciones relacionadas con el sector público.
Seguridad	No	Si	NO	No	No	No	Aplicaciones que abordan el tema de seguridad.
Social	SI	Si	SI	SI	SI	SI	Aplicaciones relacionadas con las redes sociales y la sociedad en general.

Figura 188 Categorización 3

Tiempo	SI	Si	SI	SI	SI	No	Aplicaciones que tratan el tema del tiempo meteorológico.
Viajes y guías	SI	Si	SI	SI	No	SI	Aplicaciones de viajes y guías de los mismos.
Juegos							
Acción	SI	Si	SI	SI	No	No	Categoriza los juegos con una temática de acción.
Arcade	SI	No	SI	SI	No	No	Categoriza los juegos de arcade.
Aventura	SI	Si	SI	No	No	No	Juegos que tienen que ver con aventuras.
Avatar	NO	Si	NO	No	No	No	Juegos de avatar.
Batalla, varios jugadores	No	SI	NO	No	No	No	Juegos en los que se puede jugar más de un jugador y tienen una temática de batalla.
Carreras	SI	SI	SI	No	No	No	Juegos con la temática de carreras.
Cartas	SI	No	SI	SI	No	No	Juegos relacionados con las cartas.
Casino	SI	SI	SI	No	No	No	Juegos relacionados con juegos típicos de los casinos.
Casual	SI	No	NO	No	No	No	Juegos de distintas temáticas y que son los más casuales.
Clásicos	No	SI	NO	No	No	No	Los juegos más clásicos.
Compañero	No	Si	NO	No	No	No	Se refiere a los juegos en los que se puede jugar más de un jugador.
Dado	NO	NO	SI	No	No	No	Juegos en los que se utilizan dados.
Deportes	SI	Si	SI	SI	No	No	Juegos relacionados con el deporte.
Educativos	SI	Si	SI	No	No	No	Juegos relacionados con la educación.

Figura 189 Categorización 4

Estrategia	SI	Si	SI	SI	No	No	Juegos de estrategia. Normalmente suelen ser de batallas o guerras.
Juegos de mesa	SI	Si	NO	SI	No	No	Juegos relacionados con los típicos juegos de mesa.
Juegos de rol	SI	Si	SI	SI	No	No	Juegos relacionados con el rol de personajes o personas que participan en el juego.
Lucha	No	SI	NO	No	No	No	Juegos relacionados con la temática de la lucha.
Música	SI	Si	SI	NO	No	No	Juegos relacionados con la música.
Otros	No	Si	NO	No	No	No	Es una categoría que categoriza a los juegos que no se han categorizado en ninguna de las otras categorías.
Palabras	SI	Si	SI	No	No	No	Juegos relacionados con las palabras.
Plataformas	No	Si	NO	No	No	No	Juegos de plataformas. Suelen ser con saltos para ir sorteando distintos obstáculos.
Preguntas y respuestas	SI	No	NO	No	No	No	Juegos en los que se utilizan las preguntas y respuestas como elementos principales del mismo.
Puzles	SI	Si	NO	SI	No	No	Juegos en los que se deben resolver puzles.
Rompecabezas	NO	NO	SI	SI	No	No	Juegos relacionados con los rompecabezas.
Simulación	SI	Si	SI	No	No	No	Juegos de simulación de aviones, coches…
Tablero	NO	NO	SI	No	No	No	Juegos en los que se utilizan tableros.
Tiros	No	SI	NO	No	No	No	Juegos relacionados con tiros. Puede ser con pistolas, arcos…

Figura 190 Categorización 5

Trivial	NO	NO	SI	No	No	No	Juegos en los que aparecen un trivial como elemento del mismo.
Vuelo	No	Si	NO	No	No	No	Juegos relacionados con vuelos.
Familia							
Hasta 5 años	SI	No	NO	No	No	No	Juegos con un contenido apto para personas con edad superior a 5 años.
Entre 6 y 8 años	SI	NO	NO	No	No	No	Juegos con un contenido apto para personas con edad entre 6 y 8 años.
Para más de 9 años	SI	No	NO	No	No	No	Juegos con un contenido apto para personas con edad mayor a 9 años.
Acción y aventura	SI	Si	NO	No	No	No	Son aplicaciones con un contenido familiar y relacionado con la acción y aventura.
Creatividad	SI	No	NO	No	No	No	Aplicaciones con contenido familiar y creativo.
Educación	SI	Si	SI	No	No	No	Aplicaciones relacionadas con la educación y con contenido familiar.
Juegos mentales	SI	No	NO	No	No	No	Juegos con un contenido familiar y relacionados con juegos mentales.
Juegos simulados	SI	Si	NO	No	No	No	Juegos con un contenido familiar y simulaciones.
Música y vídeo	SI	Si	NO	No	No	No	Aplicaciones relacionadas con la música y video y además con contenido familiar.
Niños y familia	No	Si	SI	SI	No	No	Aplicaciones con contenido apto para todos los miembros de cualquier familia.
*NOTA: OpenCity Apps no aparece en la lista ya que sus aplicaciones no las tiene categorizadas.							
*NOTA: data.gov no aparece en esta clasificación porque las aplicaciones no están categorizadas por categorías. Las tienen clasificadas por agencias y departamentos del gobierno.							

Figura 191 Categorización 6

Referencias

[1] Principios Open Datas. https://sunlightfoundation.com/policy/documents/ten-open-data-principles/, consultado el 10 de octubre de 2016.

[2] ¿Qué es un conjunto de datos? https://www.lasrozas.es/el%20ayuntamiento/datos%20abiertos, consultado el 10 de octubre de 2016.

[3] Qué es una aplicación web. http://rua.ua.es/dspace/handle/10045/16994, consultado el 11 de octubre de 2016.

[4] Estudio de Capgemini Consulting. https://www.es.capgemini.com/noticias/el-nuevo-portal-de-datos-europeo-principal-avance-hacia-un-mercado-con-potencial-de-miles, consultado el 12 de octubre de 2016.

[5] Directiva 2013/37/UE del Parlamento Europeo y del Consejo, de 26 de junio de 2013, por la que se modifica la Directiva 2003/98/CE relativa a la reutilización de la información del sector público. «DOUE» núm. 175, de 27 de junio de 2013, DOUE-L-2013-81251. Consultado el 15 de octubre de 2016.

[6] Los distintos tipos de aplicaciones web: http://fp.uoc.edu/blog/los-tipos-de-aplicaciones-web-que-existen/. 20 de junio de 2017.

[7] Aplicación Web: https://programacionwebisc.wordpress.com/2-1-arquitectura-de-las-aplicaciones-web/. Consultado el 20 de junio de 2017.

[8] Arquitectura SPA: https://es.slideshare.net/dcslides/spa-25806613.

[9] Arquitectura SPA: https://cink.es/blog/2013/10/07/spa-un-paradigma-de-arquitectura-de-aplicaciones-web-en-auge. Consultado el 22 de junio de 2017.

[10] API REST: https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos. Consultado el 22 de junio de 2017.

[11] Patrón MVC: https://desarrolloweb.com/articulos/que-es-mvc.html. Consultado el 26 de junio de 2017.

[12]PatrónDAO: http://chuwiki.chuidiang.org/index.php?title=Patr%C3%B3n_DAO. Consultado el 26 de junio de 2017.

[13]_Herramientas_colaborativas: https://es.wikipedia.org/wiki/Herramienta_de_trabajo_colaborativo. Consultado el 26 de junio de 2016.

[14] Control de versiones: https://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones. Consultado el 27 de junio de 2017.

[15] GitHub: https://www.freelancer.es/community/articles/github-como-puede-ayudar. Consultado 27 de junio de 2017.

[16] SLQ vs NOSQL: https://www.facilcloud.com/noticias/sql-vs-nosql-which-one-should-i-use/. Consultado el 27 de junio de 2017.

[17]_Comparación-SGBD: http://desarrollowebydesarrolloweb.blogspot.com.es/2015/02/tabla-comparativa-de-los-sistemas.html. Consultado el 27 de junio de 2017.

[18] Java: http://cs.ictea.com/knowledgebase.php?action=displayarticle&id=8790. Consultado el 27 de junio de 2017.

[19]-Ventajas_y_desventajas_de_Java: http://adsi.wikia.com/wiki/8._%C2%BFCu%C3%A1les_son_las_ventajas_y_desventajas_de_usar_JAVA%3F_%C2%BFCu%C3%A1nto_cuesta%3F. Consultado el 28 de junio de 2017.

[20] Tomcat: https://www.ecured.cu/Servidor_Tomcat. Consultado el 28 de junio de 2017.

[21] Curva de aprendizaje Spring: http://blog.neodoo.es/2007/12/16/comparativa-de-frameworks-spring/. Consultado el 28 de junio de 2017.

[22] Anotaciones jersey: https://www.adictosaltrabajo.com/tutoriales/jersey-rest/. Consultado el 28 de junio de 2017.

[23] Sesión: https://support.google.com/analytics/answer/2731565?hl=es. Consultado el 30 de junio de 2017.

[24] HttpSession: http://www.arquitecturajava.com/usando-java-session-en-aplicaciones-web/. Consultado el 30 de junio de 2017.

[25] React.js: https://devcode.la/blog/como-funciona-reactjs/. Consultado el 1 de junio de 2017.

[26] Angular2: https://desarrolloweb.com/articulos/concepto-teorico-componente-angular2.html. Consultado el 1 de junio de 2017.

[27] Angular-cli: https://desarrolloweb.com/articulos/angular-cli.html. Consultado el 1de junio de 2017.

[28] Ventajas y desventajas Bootstrap. http://tecnologiaenvivo.com/bootstrap-ventajas-y-desventajas/. Consultado el 1 de junio de 2017.

[29]_Política-CORS: https://developer.mozilla.org/es/docs/Web/HTTP/Access_control_CORS. Consultado el 1 de junio de 2017.