Universidad Carlos III de Madrid

Escuela Politécnica Superior

Trabajo Fin de Grado

IMPLEMENTACIÓN DE UN PORTAL WEB DE OFERTA/DEMANDA DE TAREAS

Roberto Gómez Díaz

Madrid – Septiembre 2015

Agradecimientos

Lo primero de todo, en este apartado me gustaría agradecer a mis amigos y a todas las personas que me han acompañado en el transcurso de estos años su apoyo, porque sin ellos no habría sido posible llegar hasta aquí.

Mención especial se merece mi familia, sobre todo mis padres, por haberme facilitado poder comenzar estos estudios universitarios y haberse esforzado para que pudiera acabarlos, tanto o más que yo, por ser mis *"compañeros de viaje"*, tanto en los buenos como en los malos momentos, sufriendo y alegrándose conmigo. Por eso, por haber sido el pilar que me ha permitido llegar hasta donde he llegado y porque sois un ejemplo de superación. ¡Gracias!

Papá, te he visto luchar con mamá para que todo saliera bien, junto con todos nosotros. Mamá, te he visto luchar para superar una enfermedad que, finalmente, ha podido contigo. Tú siempre decías que estabas orgullosa de mí, me encargaré de que, estés donde estés, puedas seguir estando orgullosa de mí.

Ambos me habéis enseñado que en la vida tienes que luchar con todo para conseguir aquello que deseas. Por eso, este trabajo de fin de grado os lo dedico, ya que sin vosotros llegar hasta aquí no hubiera sido posible.

No podría acabar este apartado sin hacer otra mención especial a los que han sido mis compañeros durante estos cuatro años. Bueno, comenzaron siendo eso, compañeros de clase, pero poco a poco se fueron convirtiendo en amigos, incluso a veces casi familia. Hemos compartido todos los momentos buenos y malos que nos han ocurrido a lo largo de estos cuatro años de universidad, nos hemos ayudado mutuamente para que nos fuera más fácil ir superando asignaturas, hemos pasado muchas horas juntos, tanto en la universidad como fuera de ella. Porque habéis hecho que estos cuatro años se me pasaran volando, porque únicamente el hecho haber conocido gente como vosotros ya es suficiente para que haya merecido la pena haber estudiado este grado, porque aunque no nos sigamos viendo, que estoy seguro que sí, siempre quedará en mí el bonito recuerdo de haber tenido la suerte de compartir con vosotros esta vida universitaria, porque sin vosotros puede que no hubiera llegado hasta este punto.

Por todo esto y porque han sido tantos momentos los que he vivido con vosotros a lo largo de estos cuatro años que no podía poner fin a esta etapa sin haceros una mención especial. ¡Gracias!

Resumen

La aplicación web que se va a implementar en este proyecto es un portal web de oferta y demanda de tareas en el que una persona se podrá registrar para publicar esas tareas que no puede hacer por algún motivo, como sacar a pasear al perro, por ejemplo. El usuario establecerá un precio máximo orientativo por el que la persona que realice la tarea será remunerada.

Además, ese usuario también podrá ofrecer sus servicios para realizar las tareas que publiquen otras personas. Para ofrecerse tendrá que pujar por dichas ofertas, estableciendo un importe por el que estaría dispuesto a realizar la tarea.

Una vez realizada la tarea, el usuario podrá valorar la actividad influyendo así en la reputación que tiene en la aplicación la persona que ha realizado la tarea, sumándole puntos, si la valoración es positiva, o restándole si es negativa.

Así mismo, cuando un usuario reciba la puja podrá decidir si la acepta o no en función de la reputación del pujador. Pudiendo decidir así si se fía o no de dicho usuario.

La aplicación también tendrá la posibilidad de establecer una imagen de perfil para que los usuarios se puedan reconocer más fácilmente entre ellos cuando se acepten las pujas y tengan que verse en persona, y, además, puedan crearse una primera impresión del usuario antes de aceptar o rechazar una puja.

Palabras Clave

Aplicación web, portal web, página web, oferta, usuario, ofertante, puja, pujador, tarea, publicar, publicación, servlet, clase, formulario, método.

Abstract

The web application which is going to develop in this bachelor's thesis is a web page of tasks' offer and demand in which people can register to publish those tasks that they cannot do for any reason, such as take their dog for a walk. The user who publish a task will set a maximum amount by which the published task will be paid.

Moreover, a user also may offer himself to do tasks which other people have published. To do it, this user will have to bid on these tasks, setting an amount by which he or she may do the job defined.

Once the job is complete, the user who published the task will be able to evaluate the job done. In this way, this user can influence in the reputation which, the user who has complete the task, has in the web application, improving or decreasing it, depending if the evaluation is good or not.

Likewise, when a user receive a bid, he will be able to decide whether to accept or not the offer according to the reputation of the bidder. In this way, he or she will be able to determine if the user is trustable or not.

The web application also has the possibility to set a profile image so that users can recognize easily between them, when they accept bids and have to see each other face to face. Also, this profile image allow them to build a first impression of the user before accepting or rejecting a bid.

Keywords

Web application, web portal, web page, offer, user, provider, bid, bidder, task, publish, publication, servlet, class, form, method.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

1. Introduction

Following is proceed to introduce the bachelor's thesis exposing the motivation and objectives of the thesis, as well as the stages of it and the technologies which have been used along the development.

1.1. Motivation and objectives

The primary objective of this thesis is to develop a web application of tasks' offer and demand, in which people could publish daily tasks which they cannot do in a precise moment, and the may find other people who may do that job instead.

The motivation of building a web application of this sort comes from the success of this sort of webs in the United States, as we will see in the examples which will be exposed at the point "2.3. *Webs análogas",* and it is thought that, in Spain, this sort of web application can be successful because there is not any web which can connect people who live in the same neighborhood to help each other.

The user who decide to publish a task will be able to set a maximum **amount** by which will be paid. In the other hand, the users interested in do it will be able to setting an amount which they consider right for the task. In this way is established a bid system by which the offerer can choose the right bid at that moment.

Moreover, the users will be able to evaluate the job done in the tasks which they have published, modifying the reputation of the users who help them. In this way, it is pretended that a user who publish a task has an a priori perspective about who is the right user to do the task basing on the reputation of him or her. Likewise, the users will need to strive if they want to use the web application actively to convince the rest of the users that they are trustable.

This point, along with the amount of the bid, which was explained previously, are two of the key points and more crucial in choosing the right user for a task.

1.2. Resources and technologies used

To develop the thesis, different hardware and software have been used, which are described next:

- Laptop with 8 GB of RAM, processor i5 @ 2,50 GHz, graphics card NVIDIA of 1GB and Windows 7 operative system.
- Apache Tomcat server: This open source server has been used to deploy the web application and test all the functionalities, since, being a servlets and JSPs container, it is one of the best choices to deploy the application.
- IDE *Eclipse* for Java EE developers: This open source environment has been used to program the web application because it brings too many facilities which allow saving some time when programming, besides, its integration with the server is very good, which saves a lot of troubles and time when testing the application.
- MySQL: This database has been used because his performance are similar compared to other databases by which must be paid a license, such as Oracle or SQL, while MySQL does not need any license because it is open source.
- Google Chrome.
- Microsoft Word.

1.3. Development stages

The development of the thesis has been carried out in different stages which are briefly described next:

1.3.1. Stage 1: Database design

The first thing which should be done is to design the database structure because, depending on the data which is needed to store, a determined tables and fields will be created.

To do this design, all the functionalities and the data, which need to be in the application for a right behavior once it is fully developed, have been analyzed a priori. Once the analysis is done, it gets the conclusion which the database needs three tables, which will be detailed in the point *"3.2. Database Design" to go* into greater depth.

1.3.2. Stage 2: Design of the application

Once the structure of the database is set, it is the time to design the structure of the web application. As before, depending on the functionalities which need to be added to the application, the number of the classes, servlets and JSP's will be increasing.

At least six classes will be needed, three classes to design all the connections to the database, and the other three to create objects for each user, offer and bid. In this way will be easier to handle all the fields which are in each table.

Moreover, servlets will be needed to handle the user's login, the register and the view of the published offers (from now on the words task and offer will be used indistinctly) and bids by each user, besides of the respective JSP's to show the results which are sent by the servlets. All the details will be explained in the point *"3.3. Design of the web application"*.

1.3.3. Stage 3: Implementation of the application

Once both initial structures (database and application) are designed, it is the time to begin the implementation of the first version of the web application.

The first thing which will be done will be build the develop environment to start to program the web application. To do that, the first thing to do is to download the software Eclipse and set up all the things which are necessary to start to program the design and test the application as the development comes forward.

1.3.4. Stage 4: Tests

This stage goes by in parallel with the previous one, because, as the development proceeds, the functionalities will be tested, because it is easier to find the mistake by doing little tests in the application while it is developed than at the end of the process.

However, as it will be detailed in the point *"4. Pruebas de la aplicación"*, also a global test of the whole application will be made when the development of is fully complete.

1.3.5. Stage 5: Making the report

When the development is complete, or near to complete, the preparation of the report is started.

1. Introducción

A continuación procedemos a introducir el trabajo de fin de grado exponiendo la motivación y los objetivos del mismo, así como las fases de las que se ha compuesto y las tecnologías utilizadas a lo largo de su desarrollo.

1.1. Motivación y objetivos

El principal objetivo de este proyecto es implementar un portal web de oferta y demanda de tareas, de forma que las personas puedan publicar tareas cotidianas que no puedan realizar en un momento determinado, pudiendo aparecer otros usuarios ofreciéndose a realizarla en su lugar.

La motivación de crear este tipo de página web viene por el éxito que están teniendo portales de este estilo en Estados Unidos, como veremos en los ejemplos que se expondrán en el apartado *"2.3 Webs análogas"*, y creemos que en España puede tener éxito también este modelo de página web, debido a que no existe ningún portal de este estilo que pueda conectar personas que vivan en la misma zona para ayudarse mutuamente.

El usuario que decida publicar una tarea podrá establecer un precio orientativo por el que se remunerará a la persona que quiera realizarla. Por otra parte, los usuarios interesados en realizarla podrán pujar por la tarea publicada estableciendo el importe que ellos consideran que se corresponde a la tarea que van a realizar. Establecemos así un sistema de subasta de ofertas para que el usuario ofertante pueda elegir la puja que más le convenga en ese momento.

Además, los usuarios podrán valorar la realización de las tareas que hayan publicado en la web, alimentando así la reputación de los usuarios que se ofrezcan a ayudar a su vecino. De esta forma, se pretende que un usuario que publique una tarea tenga una perspectiva a priori sobre qué usuario de los que hayan pujado por su tarea es el más adecuado para realizarla basándose en la reputación que tenga. Así mismo, los usuarios tendrán que esforzarse si quieren usar de forma

activa el portal, para, de este modo, conseguir que el resto de los usuarios puedan confiar en él para futuras tareas.

Este aspecto, junto con el importe de la puja explicado anteriormente son dos de los aspectos fundamentales del proyecto y los más determinantes a la hora de elegir el usuario idóneo para la tarea que ha publicado el usuario.

1.2. Medios y tecnologías utilizadas

Para el desarrollo del proyecto se han utilizado diferentes elementos de hardware y software, los cuales se describen a continuación:

- Portátil con 8 GB de RAM, procesador i5 @ 2,50 GHz, tarjeta gráfica NVIDIA de 1GB y sistema operativo Windows 7.
- Servidor Apache Tomcat: Se ha utilizado este servidor de código abierto para poder desplegar la aplicación y probar todas sus funcionalidades, ya que, al ser un contenedor de Servlets y JSPs, es una de las mejores opciones para proceder a desplegar la aplicación.
- IDE *Eclipse* para Java EE: Se ha usado este entorno de código abierto para programar todo el código de la aplicación web, ya que da bastantes facilidades que ahorran bastante tiempo a la hora de programar, además, su integración con el servidor Tomcat es muy buena, lo que ahorra también bastante tiempo a la hora de las pruebas de la aplicación.
- Base de datos MySQL: Se ha usado esta base de datos, debido a que sus prestaciones son similares a las de otras bases de datos por las que se debería pagar una licencia, como Oracle o SQL, mientras que para MySQL no es necesario adquirir ninguna licencia, ya que es de código abierto.
- Google Chrome.
- Microsoft Word.

1.3. Fases de desarrollo

La elaboración del proyecto se ha llevado a cabo en diferentes fases que describimos brevemente a continuación:

1.3.1. Fase 1: Diseño de la base de datos

Lo primero que se debe analizar es el diseño de la estructura que va a tener la base de datos, ya que, dependiendo de los datos que se necesiten almacenar, se crearan una cantidad de tablas y de campos determinada.

Para este diseño se han pensado, a priori, todas las funcionalidades que se quieren añadir a la aplicación y los datos que se requieren para que funcionen correctamente una vez esté completamente desarrollada. Una vez hecho éste análisis, se llega a la conclusión de que se necesitan tres tablas en la base de datos, las cuales se detallarán en el apartado *"3.2. Diseño de la base de datos"* con más profundidad.

1.3.2. Fase 2: Diseño de la aplicación

Una vez establecida la estructura de la base de datos se pasa a diseñar toda la estructura de la aplicación web. Al igual que antes, dependiendo de las funcionalidades que se quieran añadir a la aplicación el número de clases, servlets y JSPs irá aumentando.

Se necesitarán seis clases, tres para todos los accesos a la base de datos, y otras tres para crear objetos con cada usuario, tarea y puja. Y poder manejar más fácilmente todos los atributos de cada una de las tablas.

También se necesitarán servlets para manejar el inicio de sesión del usuario, el registro, la vista de sus ofertas (a partir de ahora usaremos la palabra oferta o tarea indistintamente) publicadas y de sus pujas, etc., además de sus respectivos JSPs para mostrar los resultados que se envían desde los servlets. Todo este diseño se explicará con detalle en el apartado *"3.3. Diseño de la aplicación web"*.

1.3.3. Fase 3: Implementación de la aplicación

Una vez diseñada tanto la estructura inicial de la base de datos como una estructura a priori de la aplicación con la que empezar, ya se podrá comenzar a implementar esa primera versión de la aplicación web.

Lo primero que se hará será montar el entorno de desarrollo para poder empezar a programar la aplicación web, para ello lo primero que se hará es descargar Eclipse y realizar todas las configuraciones oportunas para que se pueda

programar el diseño que hemos realizado e ir probando la aplicación según vayamos avanzando.

1.3.4. Fase 4: Pruebas

Esta fase va ocurre de forma paralela a la fase anterior, debido a que a medida que se va avanzando en la implementación de las funcionalidades se irán probando, ya que es más sencillo encontrar el fallo si se van haciendo pequeñas pruebas según avanza el proyecto.

No obstante, como se detallará en el apartado *"4. Pruebas de la aplicación"*, también se realizará una prueba global de toda la aplicación cuando se dé por finalizada la implementación de todas las funcionalidades.

1.3.5. Fase 5: Elaboración de la memoria

Cuando ya tenemos la aplicación web finalizada, o a punto de finalizar, comenzamos a elaborar la memoria.

2. Estado del arte

La evolución de internet en estos últimos años ha sido muy grande, en este apartado se intenta poner en contexto al lector sobre el estado actual y la evolución del mundo digital.

2.1. Evolución de internet

Vivimos en un mundo en el que, día tras día, ocurren nuevos avances y descubrimientos tecnológicos. La mayoría de estos avances nacen, principalmente, con la función de hacer la vida más fácil a las personas, por lo que, cada vez estamos más enganchados a internet.

Ejemplo de esto es la enorme cantidad de usuarios de internet que hay en el mundo, y, si nos centramos en los usuarios de España, nos damos cuenta que el 74,8% [1] de la población usa internet (datos obtenidos a fecha de Junio de 2014). En el siguiente mapa [2] se puede ver la distribución de los usuarios de internet por el mundo, que en Enero de este año era del 42%, evolucionando así un 7% desde Enero del 2014.

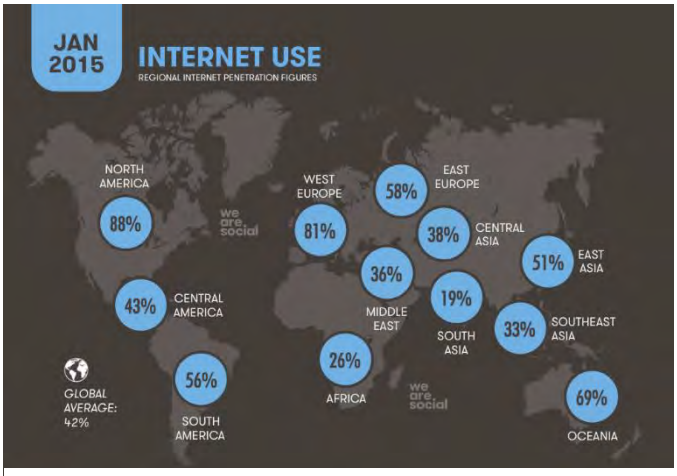

Figura 1. Usuarios de internet en el mundo

Además, si analizamos la evolución de internet con el paso de los años, se puede observar que estamos en una evolución constante y, probablemente, llegará el momento en el que toda la población use activamente internet.

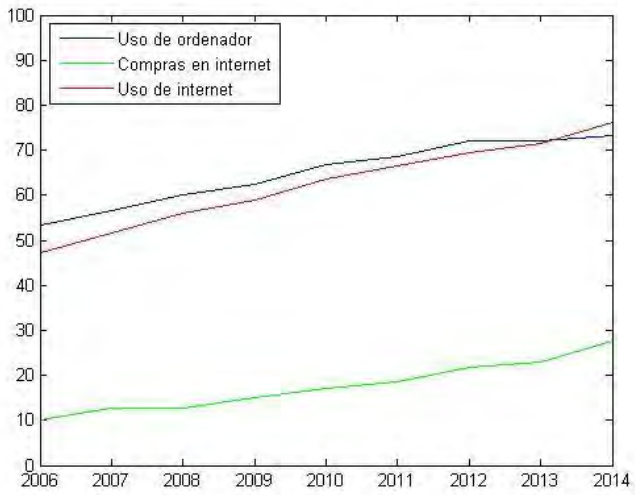

Figura 2. Evolución del uso de internet

En la gráfica anterior (datos obtenidos del Instituto Nacional de Estadística [3]), se puede observar la gran evolución del uso de internet y cómo la sociedad tiende a utilizar la tecnología para prácticamente cualquier cosa.

Otro factor importante que se puede observar en la gráfica es el aumento de las compras en Internet, que en 2014 llegaba prácticamente hasta el 30%, y hoy en día, sin duda que ese porcentaje lo estaremos superando. El aumento de este factor empezó en 2008, justo cuando la crisis económica estaba empezando. Esto no es casualidad, ya que, a menudo, podemos encontrar en la red muchos

productos a precios menores que en tiendas físicas. Por este motivo, el aumento no es casualidad, ya que todo lo que sirva para ahorrarse "unos eurillos" siempre es bienvenido.

La introducción en el mercado de *smartphones* es otro de los factores que favorecen tanto el aumento de las compras por internet como el aumento del uso de internet, debido a que se puede acceder a cualquier sitio estés donde estés, ya sea para comprar algo, visitar una red social o cualquier contenido online. Esto es por lo que, en la gráfica, se ve un estancamiento en el uso del ordenador pero un gran aumento del uso de internet.

2.2. Redes sociales

Ya hemos visto que el uso de internet ha aumentado mucho en los últimos tiempos, esto nos ha llevado a que haya muchos usuarios cada vez más enganchados a las redes sociales, en muchos casos adictos o rozando la adicción a ellas.

La evolución en España de las redes sociales se ven claramente en el siguiente estudio [4] realizado a principios de 2015, donde se muestra que el 82% de las personas entre 18 y 55 años son usuarios de redes sociales. Además, se puede ver el liderazgo que tienen redes sociales como *Facebook* entre la población, ya que es una de las tres redes sociales más visitadas junto a *Youtube* y *Twitter*.

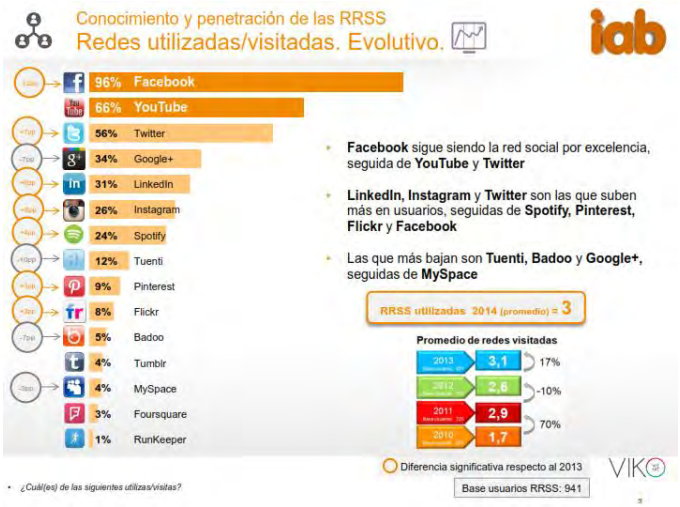

Figura 3. Redes sociales más utilizadas

2.3. Webs análogas

Este proyecto nace viendo el éxito que está teniendo este modelo en Estados Unidos. Un ejemplo de esto se puede ver en la red social *Nextdoor* [5], la cual fue fundada en 2010 y comenzó como una versión de prueba cerrada, creciendo hasta los 3500 barrios con este modelo. La idea principal era la de conectar a los vecinos para ayudarse mutuamente.

Su éxito ha sido descomunal, ya que actualmente esta red social es usada en más de 60000 barrios de Estados Unidos y está valorada en más de 1 billón de dólares.

Cuando un modelo de página web tiene éxito no tardan en aparecer algunos que, de alguna manera, intentan copiar el modelo ofreciendo el mismo servicio con algunas diferencias. Esto está ocurriendo con el sitio web *Helpyourneighbor* [6] que actualmente está en fase de pruebas como *Nextdoor* en sus comienzos. Por lo que parece, ambos van a intentar convivir en Estados Unidos, con el tiempo se verá si eso es posible o hay un único vencedor en la batalla.

Este modelo de página web aún no está presente en España, por lo que se piensa que puede tener éxito algún modelo similar y adaptado la sociedad y mentalidad de nuestro país, ya que no todas las personas ni todos los países son iguales.

3. Desarrollo de la aplicación

En este apartado se explicará detalladamente cómo ha sido todo el proceso de desarrollo de la aplicación web, además de todas las especificaciones técnicas de todos los elementos que la componen.

3.1. Entorno de desarrollo

Como ya se comentó en el apartado *"1.2. Medios y tecnologías utilizadas"* se ha usado el software Eclipse, para programar la aplicación web, junto con el servidor Apache Tomcat, para hacer funcionar la aplicación.

Los principales motivos por los que se han elegido estos dos softwares se exponen a continuación:

- Ambos tienen licencia de software libre, por lo que no hay que pagar ninguna licencia por su uso. Además, al ser licencias libres cualquier persona puede contribuir a su mejora informando de fallos en la aplicación, por lo que ambos van a estar mejorando continuamente.
- Ambos softwares tienen una gran compatibilidad con Java y JSP, que son los lenguajes que se van a usar para todo el desarrollo del proyecto.
- En lugar de Eclipse para el desarrollo de la aplicación web, también se podría haber usado otro software como *NetBeans*, siendo el resultado el mismo, ya que ambos tienen sus ventajas y sus desventajas, por lo que, el criterio que se puede seguir uno u otro es la familiarización con la interfaz de cada software.
- También se podría haber usado un bloc de notas para escribir el código en lugar de un IDE. Esta opción implicaría una evolución más lenta del desarrollo de la aplicación, ya que el uso de un IDE te permite la validación del código al instante pudiendo corregir inmediatamente los errores de sintaxis. Además, te permite depurar el código para seguir el código línea a línea y buscar los errores, lo que, en mi opinión es la mayor ventaja del uso de un IDE frente al bloc de notas tradicional.
- La elección de Apache Tomcat en lugar de otros servidores como *Jetty* o Glassfish, que también son compatibles con los lenguajes que se van a utilizar, también se basa en que es un servidor estable, su configuración es muy sencilla a la hora de desplegar una aplicación web, además de ser el servidor Java EE más utilizado como muestra la siguientes estadísticas de *Plumbr* [7].

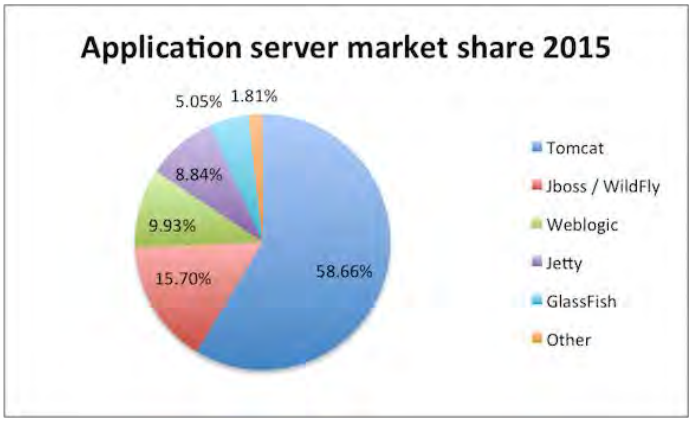

Figura 4. Servidores más usados

3.2. Diseño de la base de datos

A la hora de diseñar la base de datos se tienen que tener en cuenta todos los datos que se deben almacenar para poder consultarlos posteriormente por la aplicación web en el momento necesario.

Por ello, se crea una tabla de usuarios con un campo para introducir las características y los datos de cada uno de ellos. Los campos que posee la tabla se detallan en la siguiente tabla:

Campo	Tipo de dato	Descripción
ID	Entero de incremento automático (Clave Primaria)	Campo que facilitará la interacción entre las tablas, siendo más rápido buscar resultados en consultas conjuntas de tablas al buscar un número entero.
Fecha de Registro	*DateTime*	Almacena la fecha en la que se registra el usuario para que se pueda tener un control sobre la cantidad de registros en un intervalo determinado. Esto permitirá en un futuro poder sacar estadísticas sobre el uso de la aplicación.
Usuario	*Varchar*	Almacena el nombre de usuario con el que entrará en la página web, y por el que el resto de usuarios le podrán identificar.
Nombre	*Varchar*	Almacena el nombre del usuario para poder mostrarlo en la aplicación cada vez que entre en ella o se vea su cuenta.

Apellidos	*Varchar*	Almacena el apellido del usuario para poder mostrarlo en la aplicación cada vez que el usuario vea la información de su cuenta y para diferenciar entre usuarios con el mismo nombre.
Localidad	*Varchar*	Almacena la localidad del usuario para que, si lo desea el usuario, el resto de personas puedan ver la localidad en la que se encuentra y así puedan pujar por su oferta o no basándose en la cercanía a la que se encuentren de la localidad.
Localidad Publica	Entero de 1 byte	Campo booleano que almacena si el usuario quiere que la localidad sea pública o no. Por defecto el valor es true. Si el valor del campo es 0 se considera *'FALSE'*, en caso contrario será *'TRUE'*.
Provincia	*Varchar*	Guarda la provincia en la que se encuentra el usuario. Este campo sí que es público, ya que es un campo necesario para que el resto de usuarios tengan una orientación acerca del lugar donde se realizará la tarea.
País	*Varchar*	Almacena el país en el que se encuentra el usuario. Este valor también es público al igual que la provincia.

Email	*Varchar*	Almacena el correo electrónico que el usuario introduce al registrarse en la aplicación. Podrá ser visto por el resto de usuarios para poder ponerse en contacto con él.
Teléfono	*Varchar*	Almacena el teléfono del usuario. Al igual que el correo electrónico, éste campo podrá ser visto por el resto de usuarios para que puedan ponerse en contacto con él
Reputación	Entero de 1 byte	Almacena la reputación del usuario. Su valor por defecto es 0, lo que consideraremos como una reputación nula.
Salt	*Varchar*	Almacena el salt que se genera automáticamente en el registro del usuario.
Hash	*Varchar*	Almacena el hash correspondiente al usuario. Al igual que el salt, éste campo es generado automáticamente en el registro. La utilidad de este campo y del campo del salt la comentaremos con más detalle en el apartado *"3.3. Diseño de la aplicación web".*
MimeType	*Varchar*	Almacena el Content-Type de la cabecera MIME que se genera cuando el usuario sube una imagen para mostrarla como imagen de perfil.

Figura 5. Tabla de usuarios

También se crea una tabla de ofertas, donde se irán insertando las ofertas que vaya publicando cada usuario. La descripción detallada de todos los campos se puede ver en la siguiente tabla:

Campo	Tipo de dato	Descripción
ID	Entero de incremento automático (Clave Primaria)	Campo con el mismo objetivo que en la tabla anterior de usuarios.
ID de usuario	Entero (Clave Foránea)	Campo que hace referencia al ID del usuario que ha publicado la oferta de forma que con un número entero podemos localizar fácilmente al usuario mediante una consulta conjunta de ambas tablas.
Titulo	*Varchar*	Almacena el título que el usuario ha puesto a la tarea.
Descripción	*Text*	Guarda la descripción detallada acerca de la tarea publicada.
Precio	*Decimal*	Almacena el importe máximo por el que el usuario que ha publicado la oferta remunerará al que decida realizar su tarea.
Fecha de publicación	*DateTime*	Almacena la fecha de publicación de la tarea para que los usuarios puedan tener información acerca de la fecha en que se publicó. Además, con este dato, en el futuro, se podrían realizar estadísticas acerca de los momentos en los que se usó más la aplicación.

Fecha de inicio	*Date*	En este campo se almacena la fecha en la que el usuario que ha publicado la tarea ha decidido que se inicie.
Fecha de fin	*Date*	Este campo es similar al anterior, ya que almacena la fecha en la que el usuario ha decidido que finalizará la tarea realizada.
Hora	*Time*	En este campo almacenamos la hora en la que el usuario estima que se debe realizar la tarea publicada. Tanto este campo como los dos anteriores se supone que deben ser estimaciones del usuario acerca de las fechas en las que cree se debe realizar la tarea

Figura 6. Tabla de ofertas

Por último, también es necesario crear una tabla de pujas con el mismo fin que se ha creado la tabla de ofertas, ir introduciendo las pujas que vayan publicando los usuarios. A continuación vemos en detalle los campos de la tabla:

Campo	Tipo de dato	Descripción
ID	Entero de incremento automático (Clave Primaria)	Campo con el mismo objetivo que en las tablas anteriores.
ID de usuario	Entero (Clave Foránea)	Al igual que en la tabla anterior, este campo hace referencia al ID del usuario que ha publicado la puja.

ID de oferta	Entero (Clave Foránea)	Con este campo se hace referencia al ID de la tarea a la que la puja está referida.
Precio	*Decimal*	Almacena el importe que el usuario que ha pujado considera que le tienen que pagar por realizar esa tarea.
Descripción	*Text*	Al igual que la tabla anterior, este campo guarda una descripción detallada de la puja por si el usuario lo considera oportuno.
Estado	*Enum*	Este campo almacena el estado en el que se encuentra la puja. Puede tomar tres posibles valores: - Aceptado: La puja ha sido aceptada. - Rechazado: La puja ha sido rechazada. - Espera: No han contestado a la puja.
Votado	Entero de 1 byte	Campo booleano que almacena si la tarea que ha realizado el pujador, en caso de que haya sido aceptada su puja, ha sido votada por el usuario.
Bien realizado	Entero de 1 byte	Campo booleano que está relacionado con el anterior. Almacena si la votación de la tarea ha sido buena o mala. Si el valor del campo es verdadero significará que ha sido favorable la votación.

Figura 7. Tabla de pujas

En cuanto a los tipos de datos que se han usado, en algunos casos se podrían haber usado otras alternativas que hubieran sido igualmente válidas. A continuación se explica por qué se han elegido los tipos de datos expuestos anteriormente y no otras alternativas:

- Varchar: Se podría haber usado el tipo Char, sin embargo el tipo de dato Varchar tiene la ventaja que en la base de datos ocupa en memoria los bytes del texto que se introduzca más un byte adicional que almacena la longitud del texto, mientras que el tipo de dato Char es de longitud fija y va a ocupar en la base de datos los bytes que se hayan definido.
- Text: Se ha usado este valor para los campos que se prevea que puedan ocupar más de 255 bytes como las descripciones, ya que en versiones antiguas de MySQL los tipos de datos Varchar únicamente podían contener hasta 255 bytes, mientras que este tipo de dato puede contener hasta 2^{16} bytes.
- Enteros: Se han usado estos tipos de datos ya que permiten introducir una cantidad de datos bastante grande (2^{16}). Además, si fuera insuficiente, se podría modificar el tipo de datos a un entero grande, que permite 2^{32} registros distintos.

Por último, en las tres tablas su clave primaria es el ID propio de cada una de ellas. Éste será el que se use para relacionar las tablas, por lo que el resto de tablas tendrán claves foráneas con referencias al ID de la tabla con la que se vayan a relacionar.

En la siguiente imagen se puede observar cómo queda la estructura definitiva de la base de datos.

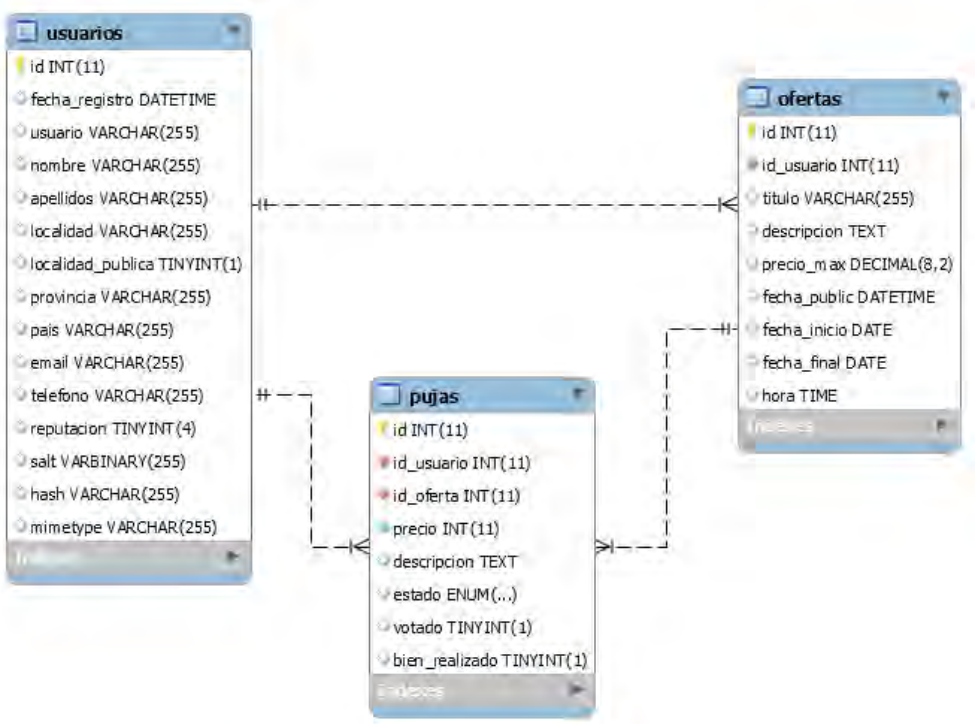

Figura 8. Base de datos

3.3. Diseño de la aplicación web

Para el diseño de la aplicación lo primero que se ha hecho es estructurar todas las clases en cuatro paquetes para que sea más fácil controlar la aplicación y el código sea más manejable. Por tanto, la aplicación web tiene los siguientes paquetes:

- Paquete com.uc3m.TFGRGD.DAO.
- Paquete com.uc3m.TFGRGD.Models.
- Paquete com.uc3m.TFGRGD.Servlets.
- Paquete com.uc3m.TFGRGD.Utils.

Además, la aplicación contendrá páginas web dinámicas creadas con JSP que nos ayudarán al diseño de la aplicación para mostrar al usuario lo que quiere ver en cada momento determinado. Esto se verá en detalle en el apartado *"3.3.5. JSPs"*.

La lógica que se seguirá para el diseño de la aplicación la podemos ver en la siguiente imagen:

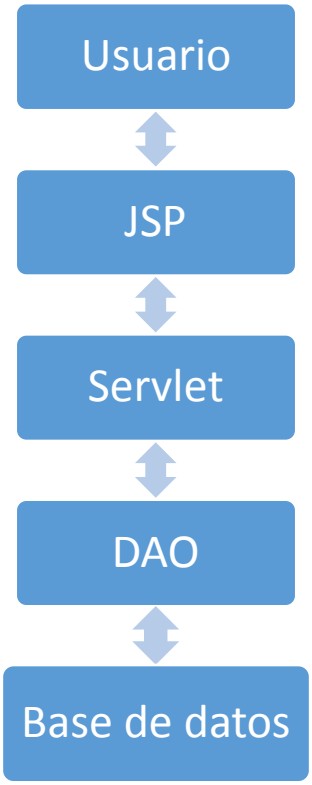

Figura 9. Lógica de diseño

Como vemos, la única parte de la aplicación capaz de interactuar con la base de datos serán las clases del paquete DAO, debido a que acceder a la base de datos directamente desde otros sitios como los servlets o los JSP sería inseguro para la aplicación web ante posibles ataques.

Para que la aplicación pueda conectar con la base de datos es necesario crear un fichero *"context.xml"*, tal y como se define en la documentación [8] de Apache Tomcat. En este fichero habrá un elemento *"Context"*, cuyo hijo será un elemento *"Resource"* con todos los atributos necesarios para conectar a la base de datos, los cuales definimos a continuación:

Atributo	Función
Name	Nombre del elemento *"Resource"*.
auth	Especifica dónde se maneja el login al recurso. Puede ser en la aplicación o en el servidor. Por seguridad, se manejará en el servidor, por lo que su valor es Container

type	Clase java que se usará para buscar este objeto Resource. En este caso la clase será javax.sql.DataSource
username	Nombre de usuario de la base de datos
password	Contraseña del usuario de la base de datos
driverClassName	Especifica el driver que se usará para conectar. Se usará el driver JDBC de MySql.
url	Especifica la dirección URL a la base de datos. Tendrá el formato siguiente: *"jdbc:mysql://[host]:[port][/[database]]"* como se especifíca en la documentación de *MySql* [9]
maxActive	Máximo número de conexiones activas
maxIdle	Máximo número de conexiones inactivas

Figura 10. Diseño del fichero context.xml

Adicionalmente a este fichero, es necesario otro llamado *"web.xml"* con un elemento *"web-app"* en el que se hará referencia a cada servlet de la aplicación, además de hacer referencia al elemento *"Resource"* del fichero *"context.xml"* que acabamos de definir mediante un elemento *"resource-ref"*.

Además, la estructura de carpetas de la aplicación web será la que se define en la documentación [10] de Apache Tomcat, ya que es el servidor que se usará. De modo que se tendrá que configurar el IDE que se use, en este caso será Eclipse, para que compile las clases en el directorio adecuado de forma que Tomcat sea capaz de desplegar la aplicación web. A continuación mostramos una imagen con la estructura de carpetas generada por el IDE Eclipse y una breve descripción del contenido de cada una de ellas:

Figura 11. Estructura de la aplicación

Como vemos en la anterior imagen, hay tres carpetas principales:

- Carpeta *"src"*: Aquí se encuentra todo el código fuente de la aplicación, con una carpeta por cada paquete mencionado anteriormente, en cada una de estas carpetas irá el código fuente de sus correspondientes paquetes.
- Carpeta *"doc"*: Aquí se encuentra toda la documentación *Javadoc* de la aplicación.
- Carpeta *"WebContent"*: Estos son los ficheros que habría que subir al servidor Tomcat para que funcione la aplicación. Aquí nos encontramos con todos los JSP de la aplicación, además de las siguientes subcarpetas:
    - Carpeta *"css"*: Aquí se encuentra la hoja de estilos. En la subcarpeta *"fonts"* se encuentran los ficheros de las fuentes que usa la aplicación.
    - Carpeta "*images"*: Aquí se encuentran todas las imágenes que usa la aplicación en algún momento. En la subcarpeta "*profile"* se guardan las imágenes de perfil de cada usuario con un nombre

específico que veremos más adelante en el apartado *"3.3.4 Paquete Servlets"*.

- Carpeta *"js"*: Aquí están todos los ficheros *JavaScript* de la aplicación.
- Carpeta *"META-INF"*: Aquí se encuentra el fichero context.xml mencionado anteriormente.
- Carpeta *"WEB-INF"*: Aquí se encuentra el fichero *"web.xml"*, además de las siguientes subcarpetas:
  - Carpeta *"lib"*: Aquí se encuentran las librerías adicionales que necesita la aplicación. Aquí se encuentra el conector de *MySql* y la librería "*tika-app-1.0"* de Apache para detectar el *MimeType* de cada imagen subida al servidor.
  - Carpeta *"clases"*: Aquí está todo el código fuente compilado de la aplicación.

3.3.1. Paquete DAO

El nombre del paquete, cuyas siglas (DAO) significan Data Access Object, se debe a que aquí se encuentran todas las clases que tengan acceso a la base de datos. Cada una tiene un atributo privado llamado "*connection*" para almacenar la conexión a la base de datos del objeto cuando es creado.

En este proyecto se crean tres clases en función de la tabla que se vaya a consultar a la base de datos. En las tres hay tres métodos comunes para controlar la conexión a la base de datos, estos métodos son los siguientes:

Método	Función que realiza
Constructor	Constructor del objeto. Llama al método *connect()*.
private void connect()	Conecta a la base de datos especificada en el "*context.xml*"
public void close()	Cierra la conexión con la base de datos.

Figura 12. Descripción de los métodos comunes del paquete DAO

En cuanto a la seguridad, los ataques que podría sufrir la aplicación y que se deben tener en cuenta a la hora de programar las clases que incluyen este paquete son los ataques por inyección SQL.

Estos ataques pueden producir que un usuario malintencionado pueda hacer una consulta para averiguar las contraseñas de los usuarios o para conocer la estructura de la base de datos de la aplicación.

Para evitar, en la medida de lo posible, estos tipos de ataques todas las consultas que se hagan en estas clases utilizarán la clase "*PreparedStatement*" de *Java,* usando así parámetros para cada consulta que hagamos en los que tendremos que indicar el tipo de dato que es cada parámetro. De esta forma evitaremos que un usuario pueda introducir caracteres "extraños" como "- -" que invalidan el resto de la consulta.

A continuación podemos ver un ejemplo de consulta con *"PreparedStatement"*:

```
            PreparedStatement query =
connection.prepareStatement("SELECT hash FROM usuarios WHERE
(usuario=?)");
            query.setString(1, usuario);
            ResultSet rs = query.executeQuery();
```

Para todas las operaciones relacionadas con la tabla de ofertas se usará la clase OfertaDAO que describimos a continuación:

Método	Función que realiza
public List<Oferta> buscarOfertas(String fechaPrimera, String fechaSegunda)	Busca las ofertas entre dos fechas dadas.
public boolean eliminarOferta(int idOferta, int idUsuario)	Elimina una oferta.
public List<Oferta> listarOfertas(int idUsuario)	Busca las ofertas del usuario dado.

public List<Oferta> listarOfertasUltimoMes(int idUsuario)	Busca las ofertas del usuario en el último mes.
public Oferta mostrarOferta(int id)	Busca una oferta determinada.
public int ofertar(int idUsuario, String titulo, String descripcion, String precio, String fechaInicio, String fechaFin, String hora)	Inserta la oferta.

Figura 13. Descripción de la clase OfertaDAO

Se usará otra clase, PujaDAO, para todas las operaciones relacionadas con la tabla de pujas, a continuación describimos en detalle esta clase:

Método	Función que realiza
public int buscarOfertaPuja(int idPuja)	Busca la oferta a la que está referida la puja.
public boolean eliminarPuja(int idPuja, int idOferta, int idUsuario)	Elimina una puja.
public List<Puja> listarPujas(int idOferta)	Busca las pujas de la oferta seleccionada.
public List<Puja> listarPujasUsuario(int idUsuario)	Busca las pujas del usuario seleccionado.
public boolean modificarEstado(int idPuja, int estado)	Modifica el estado de la puja.
public int pujar(int idUsuario, String idOferta, String precio, String descripcion)	Inserta la puja.

Figura 14. Descripción de la clase PujaDAO

Por último, habrá otra clase UsuarioDAO para todas las operaciones relacionadas con la tabla de usuarios.

Método	Función que realiza
public Usuario buscarUsuario(String usuario)	Busca el usuario por su nombre de usuario.
public Usuario buscarUsuarioPorId(int idUsuario)	Busca el usuario por su id.
public boolean comprobarCorreo(String email)	Comprueba si existe el correo en la base de datos.
public boolean comprobarUsuario(String usuario)	Comprueba si existe el usuario en la base de datos.
private String getHash(String usuario)	Busca el hash del usuario por su nombre de usuario.
public String getHashPorId(int id)	Busca el hash por el id del usuario.
public int getReputacion(int id)	Coge la reputación del usuario.
public byte[] getSalt(int id)	Coge el salt del usuario.
public boolean modificarDatos(int id, String contraseña, String localidad, String provincia, String pais, boolean publico, String email, String telefono, String mimeType)	Cambia los datos del usuario por los que ha indicado por los pasados por parámetro.
public boolean modificarReputacion(int idPuja, int id, boolean bien, boolean mal)	Modifica la reputación de un usuario. Si bien es true, se sumará 5 y si mal es true se le restará 5 a la reputación del usuario. El

	rango en el que variará será de -125 hasta 125.
public void register(String usuario, String contrasena, String nombre, String apellidos, String localidad, String provincia, String pais, String email, String telefono, String mime)	Inserta un usuario nuevo en la base de datos.

Figura 15. Descripción de la clase UsuarioDAO

3.3.2. Paquete Models

En este paquete se incluyen todas las clases que actuarán como objetos modelo para manejar toda la información de la base de datos. Así mismo, se crea una clase por cada tabla de la base de datos.

Cada clase tiene, principalmente, un atributo por cada columna de la base de datos, de forma que podremos acceder a los valores de la base de datos en la aplicación de una forma intuitiva y fácil mediante estos objetos. Sin embargo, no todos las clases van a tener exactamente los mismos atributos que columnas tiene la tabla en la base de datos, ya que, puede haber valores de algunas columnas que, por seguridad, no va a ser necesario almacenar en un objeto, o, puede ser necesario almacenar otras columnas adicionales debido a que se quiera mostrar al usuario datos relacionados de otra tabla.

Además, habrá un método *get* y otro *set* por cada atributo de cada una de las clases para poder obtener el valor o modificarlo con uno u otro método, respectivamente.

A continuación podemos ver los atributos que tiene cada clase para ver de forma detallada todo lo explicado en el párrafo anterior.

Clase Oferta:

Atributo	Función
Id	Almacena el valor del id de la tabla ofertas que tiene la tarea.

idUsuario	Almacena el valor del id de usuario que publica la oferta.
Usuario	Almacena el nombre de usuario correspondiente al id de usuario que contenga el objeto.
Fecha	Almacena la fecha de publicación de la oferta.
fechaInicio	Almacena la fecha de inicio de la oferta.
fechaFin	Almacena la fecha de finalización de la oferta.
Hora	Almacena la hora de realización de la oferta.
Título	Guarda el título de la oferta.
descripción	Guarda la descripción.
Precio	Guarda el importe máximo por el que se va a remunerar la tarea.

Figura 16.Atributos de la clase Oferta.

Además de todas las columnas de la tabla ofertas, se ha añadido un atributo adicional para almacenar el nombre del usuario y mostrarlo en la aplicación web.

Clase Puja:

Atributo	Función
Id	Almacena el valor del id de la tabla pujas.
idUsuario	Almacena el valor del id de usuario que publica la puja.
usuario	Almacena el nombre de usuario correspondiente al id de usuario que contenga el objeto.
idOferta	Almacena el id de la oferta.
precio	Guarda el importe que se ofrece en la puja.

descripción	Guarda la descripción.
estado	Almacena el estado actual de la puja.
votado	Guarda si ha sido votada la realización de la tarea.
tareaBienHecha	Almacena el valor de la votación de la tarea realizada.

Figura 17. Atributos de la clase Puja

Al igual que en la clase anterior, además de un atributo por cada columna de la clase pujas, se ha añadido un atributo extra para almacenar el nombre de usuario.

Clase Usuario:

Atributo	Función
Id	Almacena el valor del id de la tabla de usuarios
fechaRegistro	Fecha de registro del usuario en la aplicación.
Usuario	Nombre de usuario.
Nombre	Nombre completo del usuario
Apellidos	Apellidos del usuario.
Localidad	Localidad del usuario.
Provincia	Provincia del usuario.
País	País del usuario.
localidadPublica	Almacena si la localidad del usuario es pública en la aplicación o no.
Email	Email del usuario.
Teléfono	Teléfono del usuario.

Reputación	Reputación actual del usuario.
mimeType	Almacena el tipo de fichero que ha subido el usuario como imagen de perfil.
imagenPerfil	Almacena el nombre que tiene la imagen de perfil del usuario en el servidor.

Figura 18. Atributos de la clase Usuario

Nótese que en el objeto *"Usuario"* no se almacena ni el *hash* ni el *salt* del usuario por seguridad, ya que no nos interesa que esté almacenado en ningún sitio mientras dure la sesión del usuario.

Además, se ha añadido un campo extra con el nombre que contiene el fichero de la imagen de perfil del usuario para poder mostrarla en la aplicación. El nombre de la imagen del usuario se asigna realizando el hash del nombre de usuario, de forma que el nombre del fichero será dicho hash con la extensión que tuviera la imagen que subió el usuario. De esta forma, nos aseguramos que el nombre del fichero de la imagen va a ser único para cada usuario y, si consiguen ver el nombre del fichero, no van a sacar ninguna información relevante acerca de la aplicación.

En cuanto a los tipos de dato de cada uno de los atributos, se han elegido en función del tipo de dato elegido para cada columna de la base de datos. La correspondencia entre el tipo de datos en lenguaje *Java* y los tipos de datos en *MySql* los podemos ver en la siguiente tabla:

Tipo de datos MySql	Tipo de datos Java
Int	*Int*
DateTime / Date	*Timestamp*
Time	*Time*
Decimal	*BigDecimal*
Varchar / Text / Enum	*String*

TinyInt(1)	*Boolean*
TinyInt(4)	*Int*

Figura 19. Correspondencias entre los diferentes tipos de datos.

3.3.3. Paquete Utils

En este paquete se encuentran todas las clases que son necesarias para el funcionamiento de la aplicación pero no necesitan acceder a la base de datos ni son objetos modelo. De modo que, al no necesitar muchas funciones de este estilo, se crea una sola clase *"Utils"* con dos funciones que describimos a continuación:

- public int comprobarContrasena(String clavePrueba, String usuario)

  Este método tiene como objetivo comprobar si la contraseña pasada como parámetro como *clavePrueba* cumple con los requisitos de la aplicación que son los siguientes:

  - Debe tener una longitud entre 8 y 20 caracteres.
  - No puede contener el nombre de usuario, por este motivo se le pasa como parámetro al método el nombre de usuario.
  - Debe contener, al menos, una mayúscula.
  - Debe contener, al menos, una minúscula.
  - Debe contener, al menos, dos números.

  El método devuelve un número entero distinto dependiendo de la condición que no se cumpla.

- *public String hash(String password, byte[] salt)*

  Este método se encarga de realizar el *hash* de la contraseña del usuario antes de insertar nada en la base de datos. El hash se realiza con el algoritmo SHA-256 usando un *salt* que se genera aleatoriamente cuando el usuario se va a registrar, además, su longitud también es variable.

  De este modo nos protegemos de una forma bastante segura ante ataques que intenten averiguar la contraseña de algún usuario para suplantar su

identidad, ya que, al atacante no le sirve de nada tener precalculados todos los posibles *hash* con todas las combinaciones de palabras que pueda usar un usuario como contraseña, debería de calcular el hash de todas esas combinaciones concatenando todas las combinaciones posibles de *salts.*

Por este motivo, se ha decidido que el *salt* sea de longitud variable, ya que, si fuera fija, y alguien adivinara la longitud podría calcularse todos los posibles valores del *hash* y suplantar la identidad de cualquier usuario. Sin embargo, al ser de longitud variable lo tendría más complicado.

Esta función hash se usará también para conseguir el nombre del fichero de la imagen de perfil de un usuario, se hará el *hash* del nombre de usuario sin ningún *salt,* ya que no es una información tan comprometida como la contraseña.

3.3.4. Paquete Servlets

En este paquete se encuentran todos los servlets de la aplicación web, son los encargados de recoger toda la información que envía el usuario mediante los formularios y redirigirle hacia donde sea conveniente.

En cada servlet, si no se accede a él con el método HTTP que ha sido diseñado, se producirá un código de error HTTP 405 [11]. A continuación se va a proceder a describir todas las operaciones que se realizan en cada uno de los servlets que hay en la aplicación.

- Servlet Autenticar:

  Este servlet es el encargado de autenticar a los usuarios cuando intentan iniciar sesión en la aplicación web. Para comprobar si la contraseña es correcta se cogerá el *hash* de la base de datos y se comparará con el generado por la contraseña introducida y el *salt* guardado en la base de datos. La llamada a este servlet puede acabar de tres formas distintas:

  - El anteriormente mencionado error HTTP 405 que ocurre cuando se ha intentado acceder a este servlet mediante una petición GET. Cabe destacar que este error únicamente ocurrirá poniendo la URL del servlet directamente en el navegador. Debido a que se envía

información sensible como la contraseña y el usuario, se usa el método POST en lugar de GET, ya que es algo más segura.

- La redirección a la siguiente página ocurrirá cuando el usuario ya esté registrado en la base de datos y haya introducido todos los datos correctamente. En el apartado *"3.3.5 JSPs"* se verá un mapa de la aplicación completa en el que se podrá observar qué página sucede a cual.

- Cuando los datos no se introduzcan correctamente o el usuario no esté registrado en la base de datos se mostrará un mensaje de error indicando que son incorrectos los datos.

En la siguiente imagen podemos ver el diagrama de flujo que se ha seguido para el diseño de este servlet.

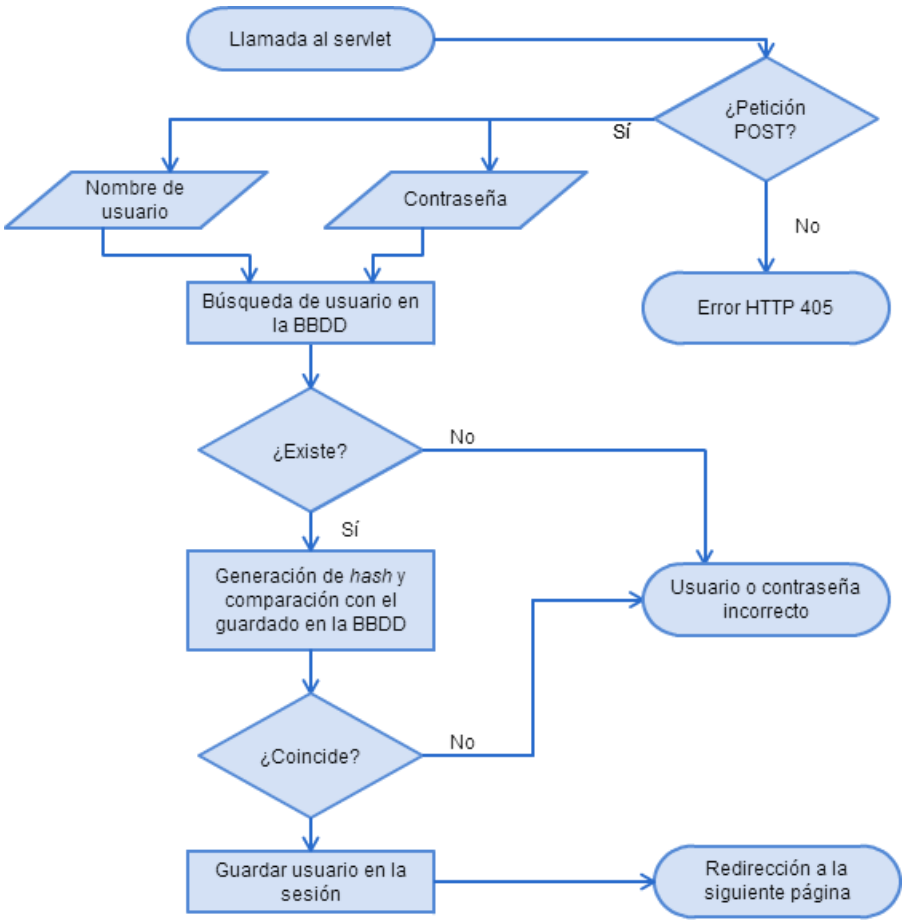

Figura 20. Diagrama de flujo del servlet Autenticar

- Servlet BuscarOferta

  Este servlet será el encargado de buscar ofertas entre dos fechas en la base de datos, ya que el usuario debe ser capaz de hacer una búsqueda de las tareas que han publicado otros usuarios para poder pujar por ellas. El servlet devolverá las ofertas que hay entre esas dos fechas en la base de datos a la página correspondiente para que sean mostradas al usuario y así pueda continuar usando la aplicación.

  Al igual que antes, se accederá a este servlet mediante el método POST, debida a que proporciona mayor seguridad que el método GET y a la hora del diseño, la complejidad no aumentaba al elegir uno u otro.

  A continuación podemos ver el diagrama de flujo de este servlet, el cual es más sencillo en comparación con el anterior.

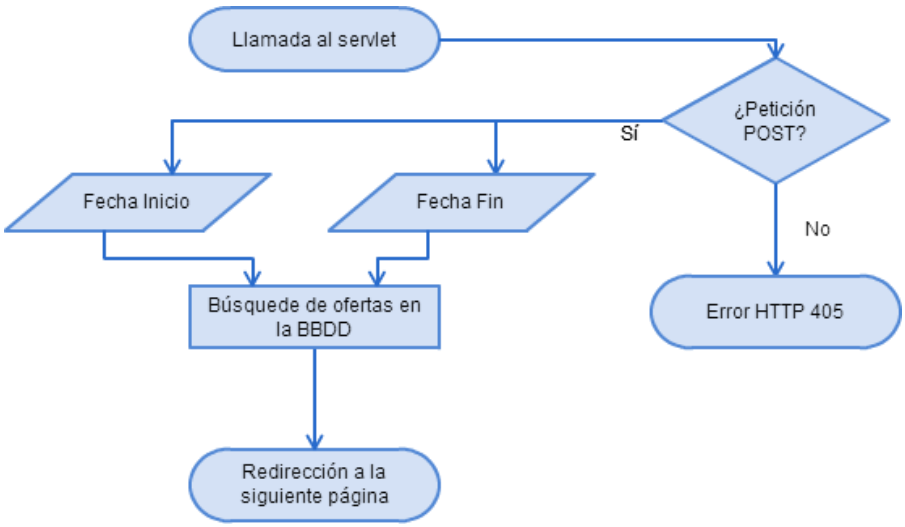

Figura 21. Diagrama de flujo del servlet BuscarOferta

- Servlet Desconectar

  Este servlet será el encargado de cerrar la sesión del usuario en la aplicación. La única acción que realiza este servlet es borrar todos los datos que haya en la sesión del usuario y redireccionarle a la página principal.

  En este servlet, el usuario recibirá el error HTTP 405 cuando intente acceder mediante el método POST a la página web, ya que el servlet se ha diseñado para que se acceda a él mediante el método GET.

- Servlet Detalles

  Este servlet se encarga de buscar la oferta que él haya seleccionado en la aplicación y enviar al usuario la información completa de la misma.

  Al igual que el servlet anterior se ha diseñado para acceder a él mediante el método GET debido a la simplicidad a la hora del diseño de la página web, en comparación con el método POST, a la poca cantidad de parámetros que es necesario enviarle al servlet y a que su ejecución no implica la modificación de datos almacenados en la base de datos.

  A continuación podemos observar el diagrama de flujo de este servlet para ver el funcionamiento del mismo.

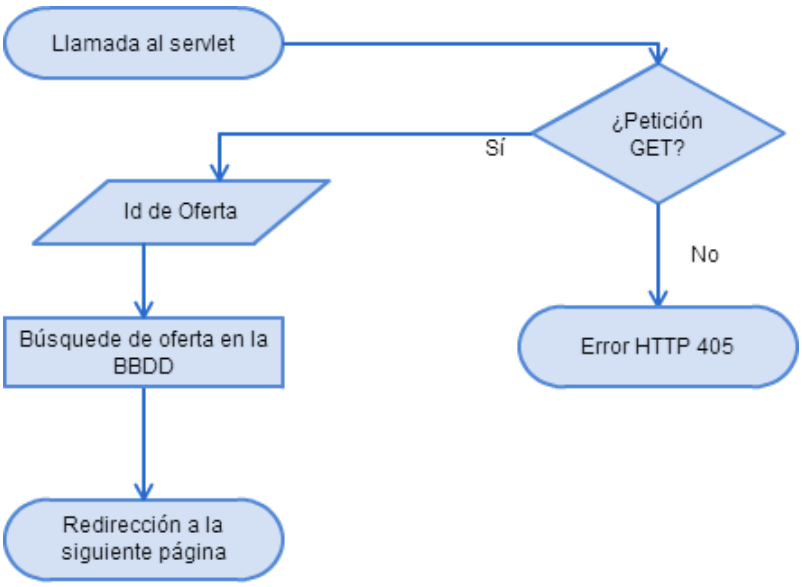

Figura 22. Diagrama de flujo del servlet Detalles

- Servlet Eliminar

  Este servlet va a ser el encargado de gestionar la eliminación de la base de datos de una puja o de una oferta en función de los parámetros que reciba.

  Se ha usado un mismo servlet para eliminar tanto pujas como ofertas, en lugar de un servlet para cada acción, para evitar la duplicidad de código en la aplicación y tener dos servlets que realicen funciones similares. Por este motivo, gestionando de una forma correcta y eficiente los parámetros recibidos en el mismo servlet, se pueden manejar ambas operaciones sin

necesidad de crear otro más. De esta forma se ahorran recursos y se evita duplicar código, como hemos dicho anteriormente.

La ejecución de este servlet, a diferencia del anterior, sí que implica la modificación de datos almacenados en la base de datos, por lo que para acceder a él sí que se usará el método POST debido a que es más segura que el método GET, ya que los parámetros no son enviados como parte de la URL y es más difícil acceder a ellos.

En la siguiente imagen podemos observar el diagrama de flujo de este servlet, viendo de una forma más gráfica su funcionamiento.

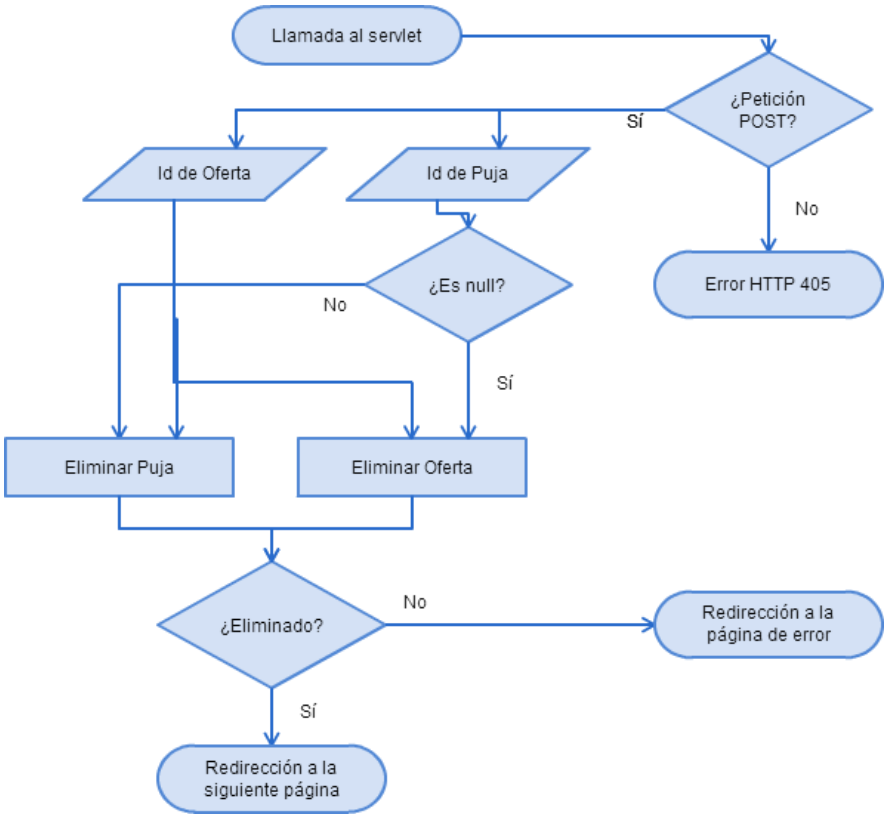

Figura 23. Diagrama de flujo del servlet Eliminar

- Servlet EstadoPuja

Este servlet es el encargado de modificar el estado de una puja, es decir, cuando un usuario quiere aceptar o rechazar una puja, este servlet será el encargado de recoger su petición y realizar todas las operaciones necesarias para realizar la petición del usuario.

Al igual que en el servlet anterior, se usará el método POST en lugar de GET por los mismos motivos, la seguridad y su ejecución implica modificaciones en la base de datos.

Cuando finaliza todas las acciones necesarias, se servlet redirecciona al usuario a la página desde la que se envió la petición, de forma que puede ver los cambios al instante.

A continuación se puede observar el diagrama de flujo que sigue este servlet para entender mejor su funcionamiento.

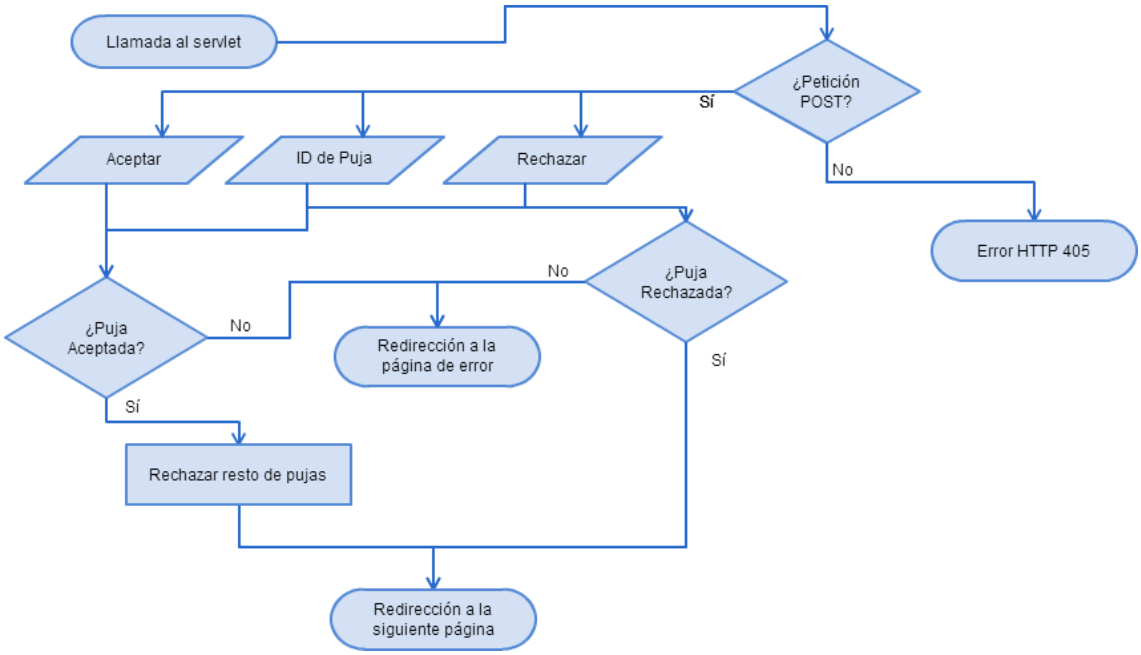

Figura 24. Diagrama de flujo del servlet EstadoPuja

- Servlet InformacionUsuario

Este servlet se encarga de recoger toda la información de un usuario en concreto de la base de datos para mostrarla en la correspondiente página.

El acceso a este se realiza mediante el método GET, por los mismos motivos que en el servlet Detalles, la simplicidad en el diseño, la poca cantidad de parámetros necesarios y su ejecución no implica la modificación de la base de datos.

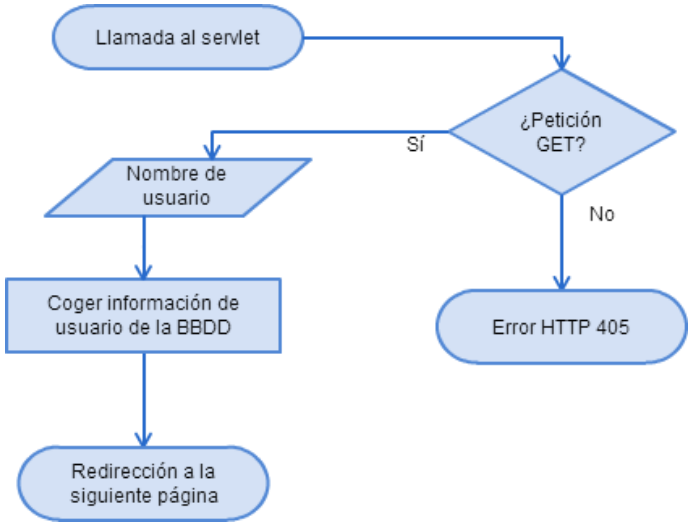

Figura 25. Diagrama de flujo del servlet InformacionUsuario

- Servlet Inicio

  Este servlet se encarga de mostrar sacar de la base de datos la información que el usuario ve en la página principal de la aplicación justo después de autenticarse, como la reputación actual del mismo y las tareas que ha publicado en el último mes.

  Para acceder aquí se usará el método GET, debido a que el usuario podrá acceder a este desde cualquier lugar de la aplicación, ya que es el servlet que controla la página principal de la misma. Además, no recibe ningún parámetro, sino que lo único que necesita lo obtiene de la sesión, ya que es donde está almacenado el usuario que está autenticado.

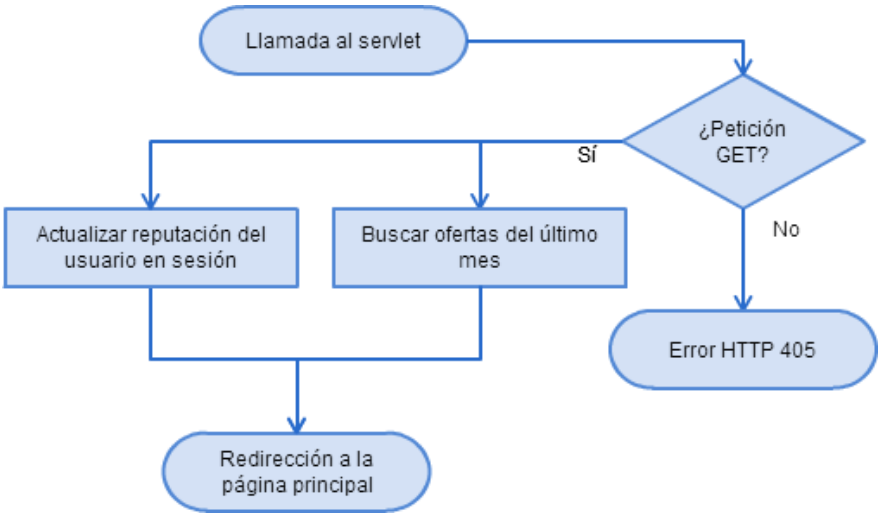

Figura 26. Diagrama de flujo del servlet Inicio

- Servlet MiCuenta

  Este servlet es el encargado de mostrar todos los datos del usuario en sesión. Estos datos ya se encuentran en la sesión, sin embargo, se ha diseñado el servlet para que se actualice la reputación que está almacenada en la sesión cada vez que se acceda a él y el usuario pueda ver el valor más actual posible.

  Esto se realiza de esta forma debido a que, si la cantidad de usuarios aumenta, el valor de la reputación puede variar mucho mientras un usuario está usando la aplicación por lo que con este diseño se intenta que el usuario tenga a la vista los valores más actuales posibles.

  El método HTTP mediante el que se accede al servlet es el método GET, ya que no recibe ningún parámetro, por lo que por simplicidad se usará este método.

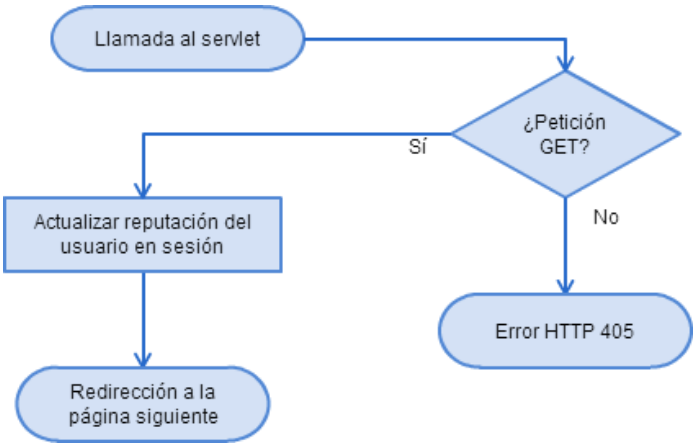

Figura 27. Diagrama de flujo del servlet MiCuenta

- Servlet MisOfertas

  Este servlet se encarga de mostrar todas las ofertas publicadas por el usuario que está autenticado. Es similar al servlet BuscarOfertas, con la única diferencia de que no recibe ningún parámetro ya que lo único que necesita lo coge de la sesión.

  Al igual que en el servlet anterior, se usará el método GET para acceder a él por simplicidad.

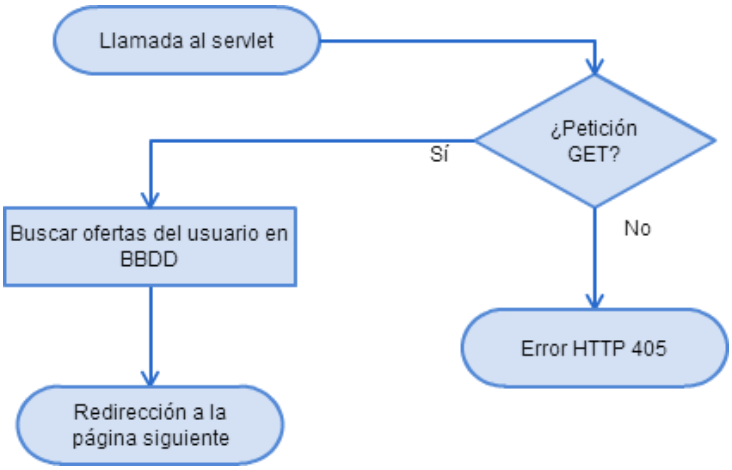

Figura 28. Diagrama de flujo del servlet MisOfertas

- Servlet MisPujas

  Este servlet es el encargado de mostrar todas las pujas que haya realizado el usuario. Es prácticamente idéntico al anterior salvo que en lugar de buscar en la base de datos todas las tareas busca todas las pujas.

  El acceso se realiza mediante el método GET por simplicidad, al igual que en el servlet anterior. No se ha reutilizado el código del servlet anterior para unirlos en uno solo, como ocurre en el servlet Eliminar visto anteriormente, debido a que no se reciben parámetros que nos permitan diferenciar de una manera fácil lo que el usuario quiere visualizar, sino que se deberían añadir parámetros adicionales con el único fin de identificar la información que el usuario quiere ver, lo que se considera una técnica ineficiente, por lo que, debido a la poca carga de código que tienen estos servlets se ha decidido crearlos por separado.

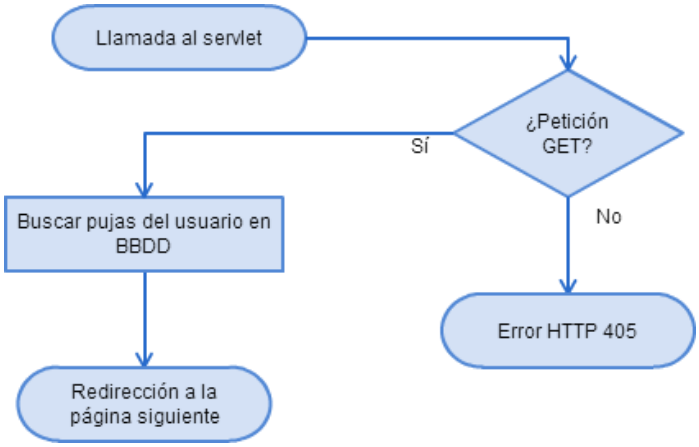

Figura 29. Diagrama de flujo del servlet MisPujas

- Servlet ModificarDatos

  Este servlet se encargará de modificar los datos del usuario que esté autenticado.

  El usuario podrá modificar cualquier dato de su cuenta salvo el nombre de usuario, por lo que se han debido tomar las suficientes precauciones para evitar que algún atacante pueda hacerse con el control completo de la cuenta del usuario. Para evitarlo se han tomado las siguientes precauciones:

  - Comprobar que el usuario en sesión y el usuario del que se quieren modificar los datos es el mismo

  - La parte más crítica de este servlet es la posibilidad de que el usuario pueda enviarle una nueva contraseña para cambiarla por la anterior. Por lo que el usuario deberá introducir la contraseña actual y se comprobará si coincide con la base de datos, además deberá introducir dos veces la contraseña nueva.

  Todas estas comprobaciones y el diseño del servlet se pueden ver de forma más clara y gráfica en el siguiente diagrama de flujo.

  Como se ve en el siguiente diagrama, el método de acceso al mismo será el método POST debido a la gran cantidad de parámetros que envían y a su confidencialidad, en el caso de que se produzca el cambio de contraseña.

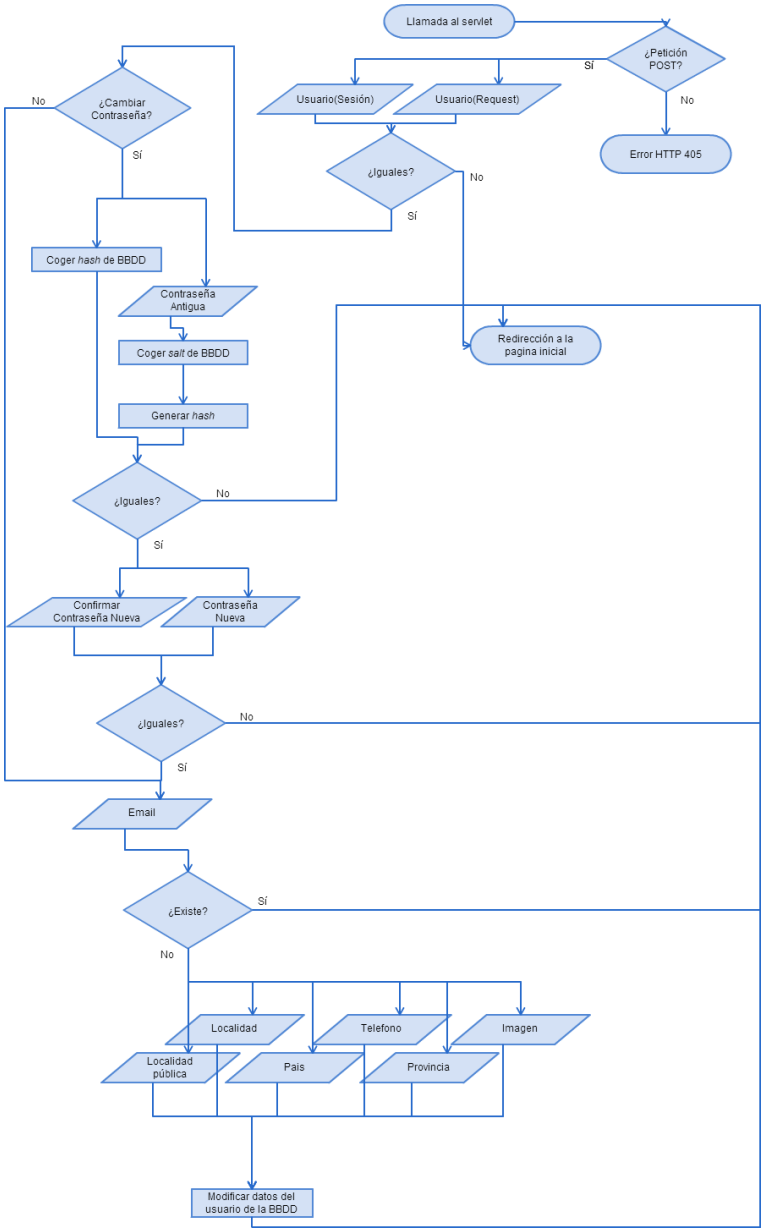

Figura 30. Diagrama de flujo del servlet ModificarDatos

- Servlet Ofertar

  Este servlet se encargará de recoger los datos introducidos por el usuario y publicar la tarea insertándola en la base de datos.

  Se usará el método POST para acceder a él, debido a que se envían muchos parámetros, además de ser más seguro, ya que realiza modificaciones en la base de datos y, si se usara el método GET, un atacante podría intentar modificar los datos de la tarea de forma más fácil.

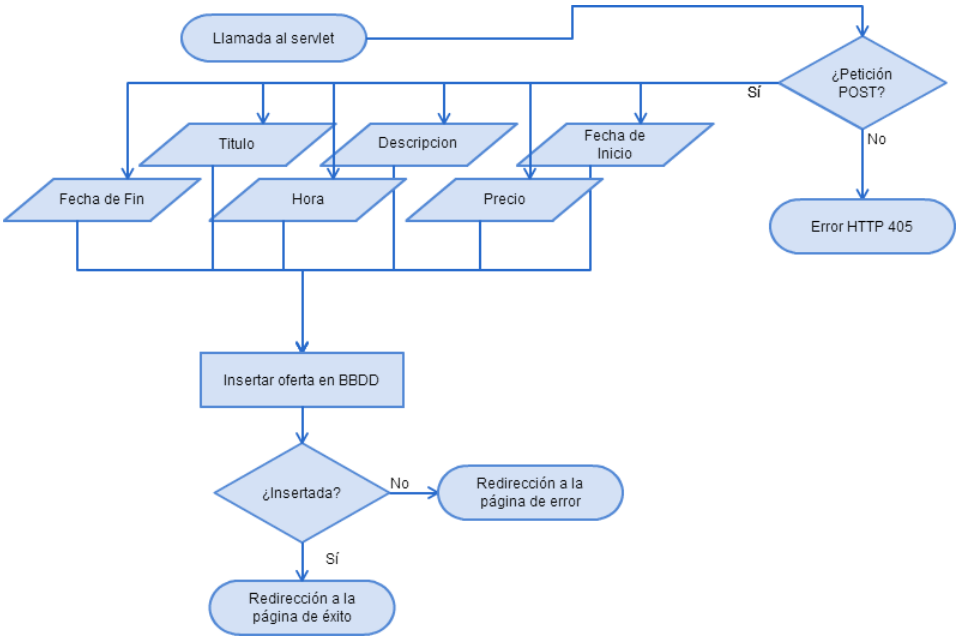

Figura 31. Diagrama de flujo del servlet Ofertar

- Servlet Pujar

Este servlet es muy parecido al anterior con la única diferencia de que en lugar de insertar una oferta se inserta una puja.

Para acceder, al igual que en el servlet anterior, se usará el método POST por los mismos motivos, ya que la longitud de la petición puede ser demasiado larga si la descripción de la puja se extiende mucho, lo que, en el caso de usar el método GET, puede dar problemas.

A continuación vemos el diagrama de flujo, prácticamente idéntico al anterior.

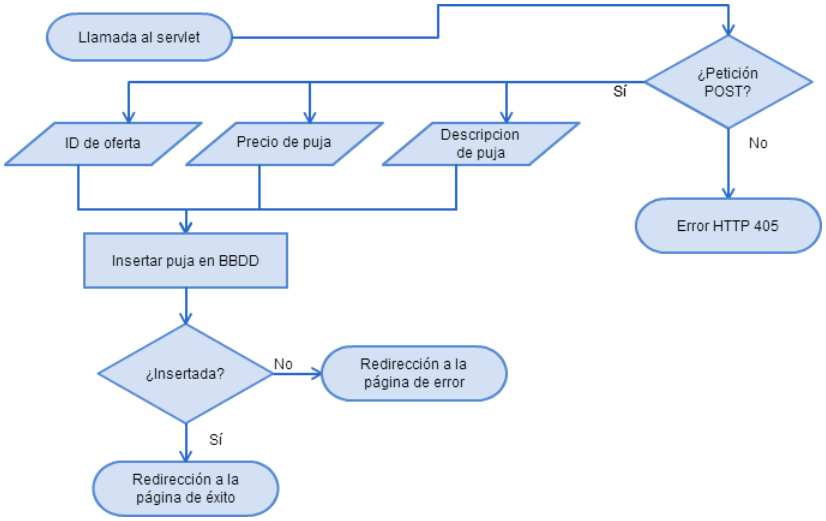

Figura 32. Diagrama de flujo del servlet Pujar

- Servlet Pujas

  Este servlet será el encargado de obtener las pujas de una determinada oferta de la base de datos para, posteriormente, mostrárselas al usuario en la aplicación y así pueda decidir cuál de ellas aceptar.

  Se usará el método GET para acceder a él, debido a que su ejecución no implica modificaciones en la base de datos.

  El servlet no solo enviará a la página de destino las pujas de una oferta determinada, sino que también enviará la información de la oferta completa, de forma que el usuario pueda ver ambas cosas para tener una visión global de todo.

  En la siguiente imagen vemos el diagrama de flujo, muy similar a alguno visto anteriormente.

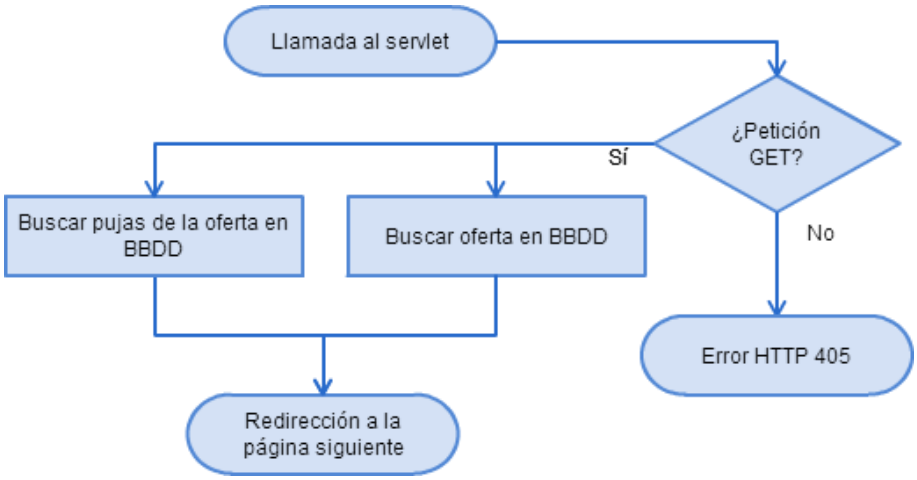

Figura 33. Diagrama de flujo del servlet Pujas

- Servlet Registro

  Este servlet se encargará de registrar usuarios en la aplicación web. Es muy similar al servlet ModificarDatos visto anteriormente.

  Las únicas diferencias de este servlet con ModificarDatos son:

  - En este servlet no es necesaria la parte de cambiar la contraseña debido a que el usuario no está en la base de datos todavía.

- Es necesario realizar una comprobación adicional, ya que al ser un nuevo usuario, es necesario comprobar si ya existe ese nombre de usuario en la base de datos aparte de la comprobación de la existencia del correo electrónico en la base de datos, como ya se hacía en ModificarDatos.

El método que se usa para acceder a él es el método POST debido a que se están enviando datos importantes como la contraseña, por lo que se necesitan unos mínimos de seguridad. Además, la cantidad de datos que se mandan haría que la petición GET fuera demasiado grande, pudiendo provocar problemas en muchos navegadores.

La siguiente imagen puede servir para entender mejor el funcionamiento del servlet.

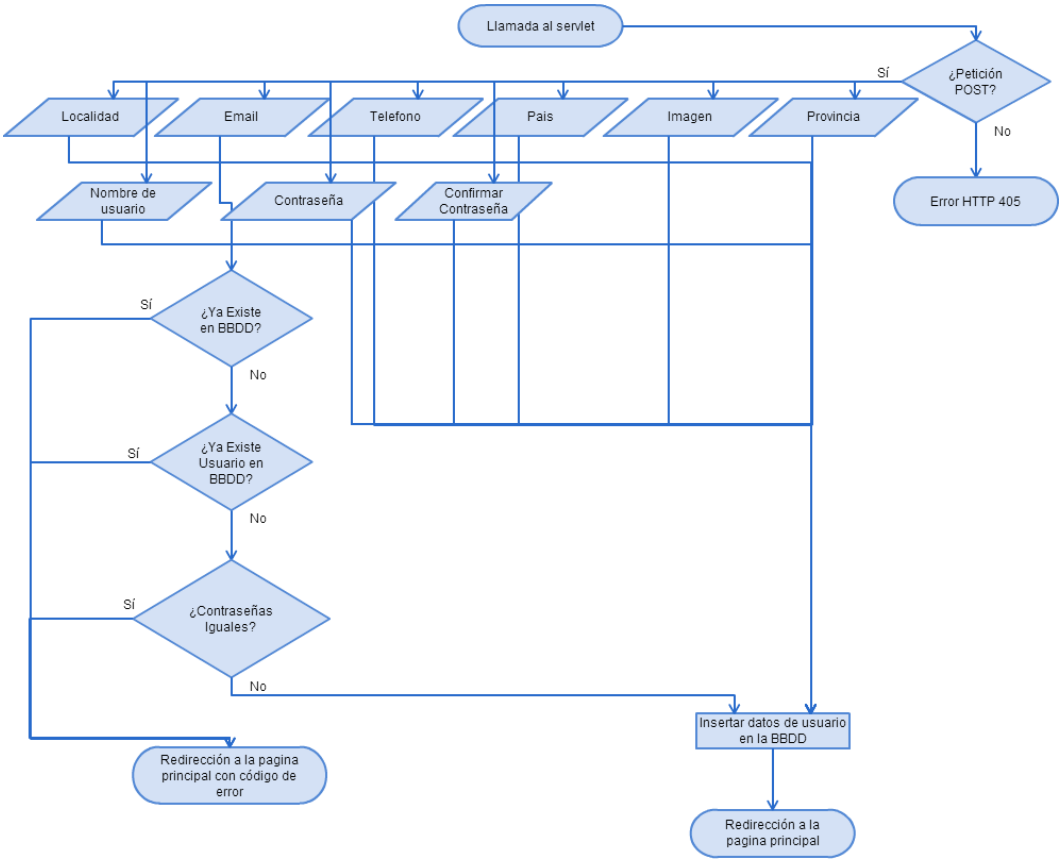

Figura 34. Diagrama de flujo del servlet Registro

- Servlet Votar

Este servlet va a ser el encargado de modificar la reputación de un usuario cuando su puja haya sido aceptada y el usuario ofertante haya votado la actividad realizada.

El método que se usa para acceder es el método POST, ya que implica la modificación de datos, además de no complicar el diseño frente a la elección de un método GET, por lo que se decide usar este método por ser más seguro.

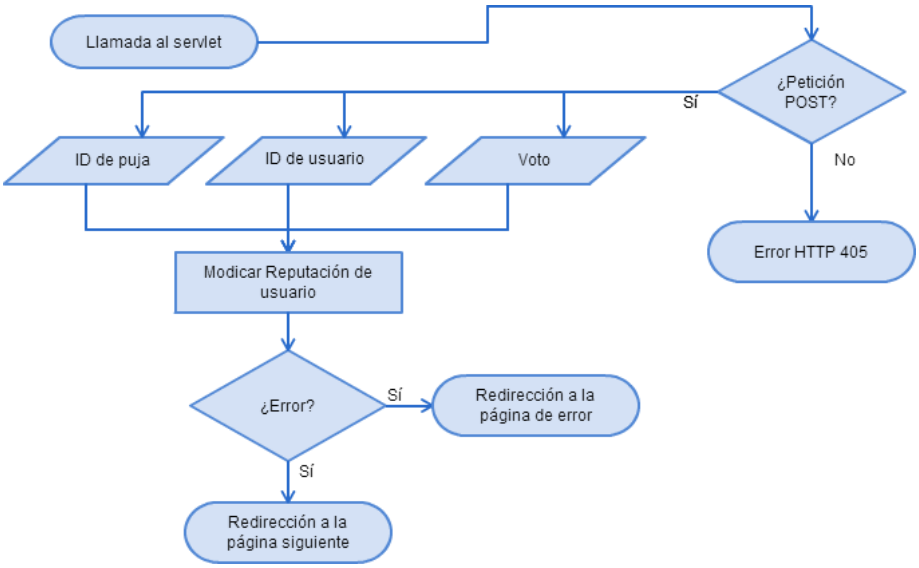

Figura 35. Diagrama de flujo del servlet Votar

3.3.5. JSPs

JSP (JavaServer Pages) [12] es la tecnología que se ha decidido utilizar para crear páginas web dinámicas, por lo que, cada una de las páginas web de la aplicación tendrá la extensión *".jsp"* en lugar de la extensión *".html"* de las páginas web estáticas.

En todas las páginas que se han diseñado, salvo la página principal que ve el usuario antes de estar autenticado y la página de error, se ha incluido una excepción para comprobar si el usuario sigue en la sesión, de forma que evitamos que alguien intente acceder a alguna parte de la aplicación de forma manual saltándose el paso de autenticarse.

Además, en las mismas páginas mencionadas anteriormente, se ha incluido una barra con un menú, de forma que desde cualquier parte de la aplicación se

puede acceder a prácticamente cualquier sitio. Al final de este apartado se podrá ver cuál es la relación entre cada una de las páginas y su relación con cada uno de los servlets. A continuación describimos brevemente cada una de las páginas JSP que se han diseñado en la aplicación.

- buscaroferta.jsp

  Esta página será la encargada de mostrar al usuario el formulario para buscar las ofertas que hay entre dos fechas que introduzca.

  Además, en la misma página se mostrarán las ofertas que haya devuelto el servlet BuscarOferta. Si el servlet no envía ninguna oferta a la página se mostrará una opción para que el usuario, si lo desea, pueda volver a buscar ofertas entre otras dos fechas.

- cuentausuario.jsp

  Esta página es la encargada de mostrar la información de la cuenta de otro usuario de la aplicación.

  Al contrario que la página anterior, esta página no es tan dinámica, ya que no hay ningún formulario que envíe información a un servlet para que éste devuelva datos para que sean mostrados. Lo único dinámico de esta página es la localidad del usuario en concreto, ya que, en función de si ese usuario ha decidido que sea o no pública, se mostrará o no en esta página.

- detalles.jsp

  En esta página se muestra toda la información completa de una tarea en concreto, además de las pujas que tenga la oferta.

  Al principio, cuando el usuario entre en esta página, únicamente verá la información completa de la oferta. Para ver las pujas de esa oferta, siempre y cuando el usuario que ha publicado la oferta y el que esté viendo la página sea el mismo, habrá un botón mediante el que se mostrarán las pujas que haya en ese momento sobre esa tarea.

  Cuando las pujas se muestran en la página, además de ver la información de las mismas, el usuario podrá realizar varias acciones sobre ellas:

  - Aceptar la puja

- Rechazar la puja
- Si la puja ha sido aceptada, el usuario también podrá votar la tarea realizada como buena o mala, de forma que se modificará la reputación del pujador.

- error.jsp

  Esta página será la que muestre la aplicación cuando ocurra un error, como intentar acceder a una página antes de autenticarse, por ejemplo, o cualquier error interno de la aplicación.

  Hay que destacar que usando la aplicación en condiciones normales, esta página no debería aparecer en ningún momento, sino que únicamente aparecerá en casos excepcionales.

- exito.jsp

  Esta página será la encargada de indicar al usuario que la oferta o la puja se ha publicado correctamente.

- index.jsp

  Esta es la página principal de la aplicación que ve cualquier usuario antes de estar autenticado.

  Aquí nos encontramos dos formularios, uno mediante el que el usuario podrá autenticarse, y otro mediante el que podrá registrarse en la aplicación si todavía no es usuario.

  La parte dinámica de esta página son los mensajes de error o de éxito que se le pueden mostrar al usuario al intentar autenticarse o registrarse, ya que si se introducen mal algunos datos el usuario es necesario que sepa qué fallo ha cometido.

- inicio.jsp

  Esta va a ser la página principal de la aplicación, una vez el usuario se haya autenticado en ella.

Aquí se mostrará la reputación del usuario y las tareas que haya publicado en el último mes. En el caso de que no haya publicado ninguna tarea en el último mes, se le mostrará un botón para acceder directamente a la página con un formulario para publicar una oferta. Si, por el contrario, al usuario sí que se le muestran tareas, tendrá la opción de ver los detalles y de eliminar cada una de las ofertas mostradas.

- micuenta.jsp

  Esta es la encargada de mostrar al usuario la información de su cuenta, además de permitirle modificarla si lo desea.

  La parte dinámica de esta página es la misma que en la página inicial, mensajes de error en el caso de que se intenten modificar los datos y se introduzcan mal.

- misofertas.jsp

  Esta página se encargará de mostrar al usuario todas sus ofertas publicadas.

  Las funcionalidades de esta página son las mismas que en la parte en que se muestran las ofertas del último mes de la página inicio.jsp, ver los detalles y la posibilidad de eliminar cada una de las ofertas mostradas.

  En el caso de que no se muestren ofertas porque el usuario no haya publicado ninguna, se mostrará un botón para acceder a un formulario que le permita hacerlo, al igual que ocurría en la página inicio.jsp.

- mispujas.jsp

  Esta página es muy similar a la anterior. Con la única diferencia que se mostrarán todas las pujas del usuario.

  Las acciones que puede realizar el usuario desde esta página es eliminar las pujas que no hayan sido contestadas, sin embargo, las que ya han sido aceptadas o rechazadas no podrán ser eliminadas.

  Si el usuario no ha pujado por ninguna oferta todavía, y, por consiguiente, no se le muestran pujas en esta página, se mostrará un enlace a la página

buscaroferta.jsp para que pueda buscar ofertas y así pujar por alguna de ellas.

- oferta.jsp

  Esta página es la encargada de mostrar al usuario el formulario para que pueda publicar la tarea que desee.

  Aquí no tenemos varias opciones de visualización al igual que en las otras páginas. Simplemente muestra el formulario.

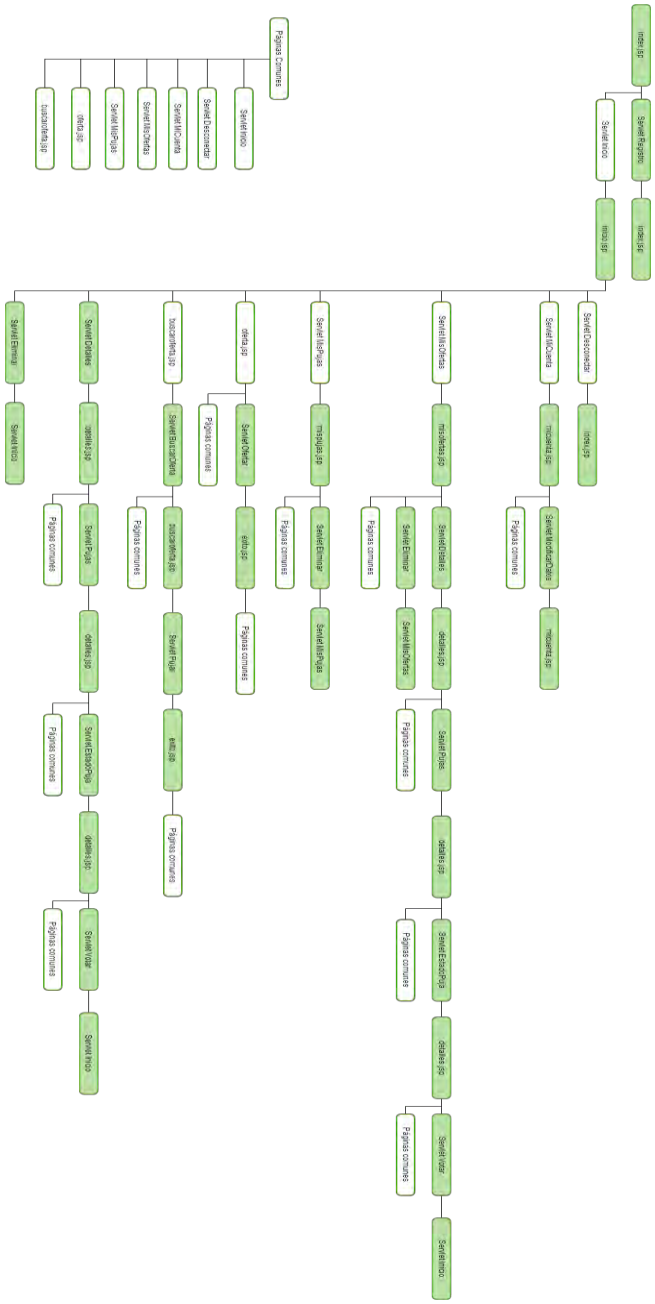

Figura 36. Mapa del sitio web

3.3.6. Extras

Para que la vista de la página sea más atractiva y agradable al usuario se han usado algunas funciones de *JavaScript* y *jQuery*, de forma que se producen algunos efectos visuales. A continuación vamos a describir algunos de los efectos más importantes de la aplicación:

- Calendario: A la hora de elegir la fecha, en algunos navegadores como Firefox o Internet Explorer no son compatibles con el elemento HTML *input* de tipo *date*. De modo que, si son algunos de estos navegadores, estos elementos por el elmento *Datepicker* que proporciona la interfaz de usuario de *jQuery*. Así, el usuario podrá seleccionar las fechas usando un calendario, aunque esté usando otro navegador que no sea Google Chrome. Para el funcionamiento de este elemento ha sido necesario añadir a las páginas los ficheros de *jQuery-UI* [13], los cuales están online y no es necesario descargarlos.

- Reputación: Para mostrar al usuario el valor de la reputación de una forma más atractiva, se ha decidido recurrir a unos emoticonos *Twemoji* [14], para mostrarlos en función de si la reputación es buena o mala, y a un plugin de *jQuery* de terceros llamado *jQuery Knob* [15]. Este plugin permitirá dibujar una semicircunferencia que estará más o menos coloreada en función del valor de la reputación del usuario. A continuación podemos ver un ejemplo de reputación neutra, buena y mala, respectivamente, para que quede más claro cuál es la visualización que tiene el usuario.

Figura 37. Reputación neutra de un usuario

Figura 38. Buena reputación de un usuario

Figura 39. Mala reputación de un usuario

4. Pruebas de la aplicación

A medida que se va avanzando con el desarrollo de la aplicación se irán probando las distintas funcionalidades implementadas para, de este modo, poder ir corrigiendo los errores que vayan surgiendo de forma más sencilla.

Una vez finalizado todo el desarrollo se hará una prueba completa de toda la aplicación.

4.1. Entorno de pruebas

Para las pruebas parciales de cada funcionalidad se usará Eclipse para desplegar la aplicación en Tomcat, de forma que podremos usar el depurador en caso de fallo, para así poder corregir de una forma más rápida los errores que pudieran surgir, siguiendo línea a línea el código de la aplicación.

Para la prueba final de la aplicación, en lugar de usar Eclipse, se usará el servidor Tomcat únicamente, copiando el contenido de la carpeta *"WebContent"* del proyecto de Eclipse a una carpeta nueva que creemos en la carpeta *"webapps"* del directorio donde se encuentre el servidor Tomcat. Posteriormente se arrancará el servidor ejecutando el fichero *"startup.bat"*, si se usara el sistema operativo Linux se ejecutaría el fichero *"startup.sh"*.

4.2. Pruebas

Para probar el funcionamiento completo de la aplicación se crearán tres usuarios, uno que será el ofertante, es decir, el que publique las tareas, y otros dos pujadores, uno se usará para simular que realiza bien una tarea, y otro para simular que realiza una tarea mal.

Para abrir la aplicación se accederá a la URL "http://localhost:8080/TFG/". Posteriormente, para probar la funcionalidad del registro, primero se introducirá la contraseña del usuario mal para probar que se

recibe un mensaje de error y luego se rellenarán todos los datos correctamente para completar el registro. En las siguientes tablas podemos ver los pasos que se han seguido en estas pruebas:

Pasos	Resultado
1- Rellenar los datos del formulario de registro en la página principal introduciendo una contraseña que no cumpla los criterios. 2- Pulsar el botón *"Unirme"*	Se recibe un mensaje de error.

Figura 40. Prueba de registro incorrecto

Pasos	Resultado
1- Rellenar los datos del formulario de registro en la página principal correctamente. 2- Pulsar el botón *"Unirme"*	Se recibe el mensaje *"Registro correcto".*

Figura 41. Prueba del registro correcto.

Posteriormente, se procederá a registrar los otros dos usuarios, como se ha dicho antes, y a probar la funcionalidad de autenticarse como usuarios. Al igual que antes se harán dos pruebas, una introduciendo mal el usuario o la contraseña, para ver si se recibe el mensaje de error, y otra introduciendo los datos bien para entrar a la aplicación.

Pasos	Resultado
1- Introducimos el usuario y la contraseña en la parte superior de la página principal introduciendo una contraseña errónea. 2- Pulsar el botón *"Entra"*	Se recibe un mensaje de error.

Figura 42. Prueba de autenticación incorrecta

Pasos	Resultado
1- Introducimos el usuario y la contraseña correctos en la parte superior de la página principal. 2- Pulsar el botón *"Entra"*	Se entra a la aplicación

Figura 43. Prueba de autenticación correcta

Una vez dentro de la aplicación, se procederá a publicar una tarea. Para llegar hasta el formulario hay dos opciones, pulsar el botón de la página principal o desplegar el menú *"Acciones"* y pulsar *"Publicar Oferta"*.

Pasos	Resultado
1- Pulsar el botón "Publicar Oferta" de la página de inicio o el que aparece en el menú desplegable *"Acciones"* 2- Rellenar los datos del formulario 3- Pulsar el botón *"Publicar"*	Inserción de la oferta en BBDD y redirección a éxito.jsp

Figura 44. Prueba de publicación de oferta

Una vez esté publicada la oferta se procederá a entrar en el apartado *"Mis Ofertas"* para probar la funcionalidad de los detalles.

Pasos	Resultado
1- Pulsar el botón *"Mis ofertas"* del menú desplegable *"Bienvenido Ofertante"*. 2- Pulsar el botón *"Detalles"* de la oferta que se acaba de publicar.	Vista de todos los datos de la oferta.

Figura 45. Prueba la vista de los detalles de la oferta

Antes de cerrar la sesión del ofertante se probará a modificar algún dato de su cuenta como el correo electrónico y la contraseña, para comprobar que funciona correctamente. Además, también se probará la funcionalidad del cierre de sesión, ya que para asegurar que la contraseña ha sido cambiada es necesario cerrar sesión y volver a entrar con la nueva contraseña.

Pasos	Resultado
1- Pulsar el botón *"Mi cuenta"* del menú desplegable *"Bienvenido Ofertante".* 2- Cambiar el correo electrónico de los datos que se muestran en pantalla. *3-* Pulsar el botón *"Cambiar contraseña".* 4- Introducir la contraseña actual en el campo *"Contraseña antigua".* 5- Introducir la contraseña nueva en los campos *"Contraseña nueva"* y *"Confirmar contraserña nueva".* 6- Pulsar el botón "Modificar datos" para confirmar el formulario. 7- Pulsar el botón *"Cerrar sesión".* 8- Volver a entrar a la aplicación usando la nueva contraseña. 9- Pulsar el botón *"Mi cuenta"* del menú desplegable *"Bienvenido Ofertante".*	Vista de los datos de mi cuenta con el nuevo correo.

Figura 46. Prueba del cierre de sesión y la modificación de datos de la cuenta.

Una vez hecho esto se cerrará la sesión del ofertante y se iniciará con uno de los pujadores siguiendo los pasos de la *Figura 43.* Una vez dentro de la aplicación, lo primero que se hará será buscar la oferta que hay publicada para, posteriormente, pujar por ella.

Pasos	Resultado
1- Pulsar el botón *"Buscar ofertas"* del menú desplegable *"Acciones".* 2- Introducir las fechas entre las que se quiere buscar la oferta. *3-* Pulsar el botón *"Pujar".* 4- Rellenar el precio y la descripción de la puja. *5-* Pulsar el botón *"Pujar".*	Inserción de la puja en BBDD y redirección a éxito.jsp

Figura 47. Prueba de la búsqueda de ofertas y la opción de pujar

Se volverá a realizar otra puja a la misma oferta, siguiendo los pasos descritos anteriormente, para probar la funcionalidad de eliminar una puja.

Además, se publicará una oferta siguiendo los pasos descritos en la *Figura 44*, de este modo se podrá probar también la funcionalidad de eliminar una oferta. Una vez publicadas ambas se procederá a eliminarlas siguiendo los siguientes pasos.

Pasos	Resultado
1- Pulsar el botón *"Mis ofertas"* del menú desplegable *"Bienvenido Ofertante".* *2-* Pulsar el botón *"Eliminar".*	Vista de Mis ofertas sin ninguna oferta publicada.

Figura 48. Prueba de la funcionalidad de eliminar ofertas

Pasos	Resultado
1- Pulsar el botón *"Mis pujas"* del menú desplegable *"Bienvenido Ofertante".* *2-* Pulsar el botón *"Eliminar".*	Vista de Mis pujas sin la puja que se ha eliminado.

Figura 49. Prueba de la funcionalidad de eliminar pujas

Posteriormente, se iniciará sesión con el usuario ofertante para aceptar y rechazar las pujas para probar estas funcionalidades.

Pasos	Resultado
1- Seguir los pasos de la *Figura 43* y *Figura 45,* respectivamente. 2- Pulsar el botón *"Ver pujas".* 3- Pulsar el botón *"Aceptar"* de una de las pujas. *4-* Pulsar el botón *"Bien" en la puja aceptada.*	Se acepta la puja seleccionada y se rechaza de forma automática el resto de pujas. Tras votar favorablemente la tarea se redirecciona a la página principal.

Figura 50. Prueba de aceptar una puja y votación favorable.

Pasos	Resultado
1- Seguir los pasos de la *Figura 43* y *Figura 45,* respectivamente. 2- Pulsar el botón *"Ver pujas".* 3- Pulsar el botón *"Rechazar"* de una de las pujas. *4-* Pulsar el botón *"Aceptar" en la puja restante.* *5-* Pulsar el botón *"Mal" en la puja aceptada.*	Se rechaza una puja y se acepta la otra. Tras votar negativamente la tarea se redirecciona a la página principal.

Figura 51. Prueba de rechazar y aceptar una puja y votación desfavorable.

Posteriormente, se aprovechará para ver la vista de los datos de la cuenta de otro usuario, para comprobar que está todo correctamente y no se pueden modificar los datos de un usuario que no sea el que está en sesión.

Pasos	Resultado
1- Seguir los pasos de la *Figura 43* y *Figura 45,* respectivamente. 2- Pulsar el botón *"Ver pujas".* 3- Pulsar sobre el nombre del usuario que ha realizado una de las puja.	Vista de los datos del usuario correspondiente.

Figura 52. Prueba de la vista de los datos del usuario.

Para finalizar las pruebas, se procederá a autenticarse en la aplicación con cada uno de los pujadores para comprobar que uno tiene mala reputación porque su puja ha sido votada como desfavorable, detalle que se puede ver viendo los detalles de la puja en el apartado *"Mis pujas"* de la cuenta del usuario, y otro tiene buena reputación porque la votación sobre su tarea ha sido positiva.

5. Conclusiones

Se puede decir, que el objetivo principal de implementar un sitio web de oferta/demanda de tareas se ha cumplido. Ahora los usuarios tienen la posibilidad de ofrecer a otras personas tareas remuneradas que ellos no pueden realizar por algún motivo.

En cuanto al cumplimiento de los requisitos iniciales que se establecieron al comienzo del proyecto y que debía tener la aplicación se deben evaluar por separado para ver si se han cumplido.

Se acordó que por cada una de las tareas ofertadas se establecería un sistema de pujas mediante el que los usuarios pudieran mostrar su interés en realizar la misma y el ofertante pudiera decidir qué usuario es el adecuado para realizar su tarea.

Se propuso que el usuario ofertante pondría una remuneración máxima de la tarea publicada para que cada uno de los usuarios que estuvieran interesados no pudiera pedir un importe desorbitado por la realización de la tarea.

También se estableció que debía implementarse un sistema de reputación por dos motivos, para aumentar la competitividad entre los usuarios por tener una buena reputación y esforzarse en la buena realización de las tareas; y para que el ofertante tuviera algún factor más con el que decidir sobre el usuario que es el más adecuado para la realización de su tarea.

En cuanto a los conocimientos puestos en práctica en el transcurso de este proyecto, se han usado todos los que se han aprendido a lo largo de la carrera en todas las asignaturas relacionadas con la programación. Sin embargo, los conocimientos que más se han usado han sido los adquiridos en la asignatura *"Aplicaciones web"* de este último curso, ya que, sin haberla cursado dudo que la finalización de este proyecto hubiera sido posible, al menos, en el tiempo que se ha realizado. Al igual que los adquiridos en la carrera, también han sido de ayuda los obtenidos en mi corta e incipiente etapa profesional.

Por tanto, teniendo en cuenta todos los aspectos anteriores, se puede concluir que este proyecto ha concluido satisfactoriamente.

5.1. Futuras mejoras

Aunque la aplicación funciona correctamente en la actualidad, en el futuro se podrían realizar mejoras en ella añadiendo más funcionalidades como las que se detallan a continuación:

- Añadir un apartado de nuevas pujas para que el usuario pueda ver si ha habido alguna novedad en alguna de sus ofertas o, por el contrario, no ha habido ninguna puja nueva.

- Relacionado con lo anterior, se podían introducir notificaciones cuando una oferta del usuario ofertante haya recibido una puja, mientras el usuario esté en la sesión.

- Actualmente, la única forma de comunicarse que tienen los usuarios entre ellos es mediante el correo electrónico o el número de teléfono que tiene cada uno en su perfil. Sin embargo, se podría añadir la posibilidad de permitir a los usuarios de comunicarse dentro de la misma aplicación por medio de mensajes privados o mediante mensajes públicos dentro de cada oferta publicada.

- Se podría establecer un sistema en el que los usuarios pudieran ver el precio de la puja más alta por una oferta. De este modo se pretendería aumentar la competitividad dentro de la aplicación para que el ofertante no salga perdiendo. En caso de que se hiciera esta mejora, se establecería un periodo de pruebas en el que se pediría opinión sobre la misma a los usuarios de la aplicación y, de este modo, decidir si la mejora sería definitiva o no.

- Crear un formulario de contacto mediante el que los usuarios se puedan comunicar con el administrador del sitio web para posibles dudas, consultas o sugerencias sobre la aplicación web.

- A parte de la valoración de una tarea como buena o mala, se podría añadir la opción de que un usuario estableciera la opinión sobre el que ha realizado la tarea, de forma que el resto puedan ver porqué ese usuario tiene mala reputación.

Para que la experiencia del usuario sea agradable en todo momento, se tendrá en cuenta su opinión, ya que el éxito de una aplicación web de este estilo se basa en que la usen la mayor cantidad de usuarios posibles.

El objetivo de estas mejoras es ir convirtiendo el sitio web, poco a poco, en una red social de oferta/demanda de tareas, donde los usuarios puedan comunicarse de una forma fácil, para realizar o encontrar a alguien que pueda realizar tareas sencillas por menos precio que cobraría un profesional y que les permitan ahorrarse esos “eurillos” de los que se hablaba en la Introducción.

5. Conclusions

You could say that the primary objective, which was implement a web portal of tasks' offer and demand, is accomplished. Now, people have the possibility of offer remunerated tasks that they cannot do for any reason.

As regard the compliance with the initial requirements which were set at the beginning of the thesis, it must be evaluated one by one to check if have been accomplished.

It was decided that, for each published task, will be established a bid system by which the user will be able to show interest in do the job and the offerer could decide who is the right user to do the task.

It was proposed that the offerer needs to be able to set the maximum amount by which the task will be paid. In this way, other interested users will not be able to demand an exorbitant amount for the task.

Also was established that must be implemented a reputation system for two reasons, to increase competition between users for having a good reputation and effort themselves at doing a good job when they do a task; and so that the offerer has an extra factor which they can decide about who is the right person to do the job.

As regards the knowledge implemented along the development of the thesis, have been used all of which have been learned along the bachelor's degree in the programming-related subjects. However, the knowledge which have been more used are all which have been learned in the subject "Web applications" that has been coursed in this last year, because, if this subject would not have been coursed, the ending of this thesis would not have been possible, at least, in the moment which the thesis have been complete. As well as the knowledge learned in the degree, also have been helpful all that have been learned along my short and emerging professional path.

Therefore, considering all the preceding points, it can be conclude that this thesis has been complete successfully.

5.1. Future improvements

Although the application is working properly nowadays, future improvements could be made to add more features such as those which are detailed next:

- Add an option of new bids so that the user can see if there has been, or not, a new bid since the last time they enter in the application.

- Connected with the previous point, it could introduced notifications when an offer has received a new bid while the user is logged on the application.

- Nowadays, the only way the users have to communicate themselves is by the email or the phone number that they have in their profile. However, it could be added the possibility of exchange private messages, or public messages inside each published offer.

- It could be established a system by which the user could see the amount of the highest bid on a task. In this way, it would be hoped to increase competition inside the application so that the offerer don't lose out. If this improvement were made, would be established a test period by which the users could give their point of view and, in this way, decide the future of the improvement.

- To build a contact form by which the users could communicate with the administrator of the web application in case they had any doubt or suggestions about the application.

- Apart from the rating of the work as good or bad, the option that the user can add an opinion about this work can be added so that the rest of the users of the web application can see why this user has a bad reputation.

For a nice experience of the user, his/her opinion will be consider at any time, because the success of this sort of web applications is based on a large quantity of users. Usually, the more users the better future has the application.

The purpose of these improvements is to create, little by little, a social network of tasks' offer and demand, where users can communicate between them easily to do, or find someone who can do, simple tasks by spending less money than by hiring a professional and allow them to save those "little euros" which were told in the Introduction.

6. Glosario

IDE: Entorno de desarrollo integrado. Software informático que proporciona un entorno de desarrollo para facilitar al usuario todo el proceso de desarrollo de una aplicación/software.

Eclipse: Es un tipo de IDE. En este proyecto en concreto se ha usado el software Eclipse para desarrolladores Java EE.

RAM: Memoria de acceso aleatorio (en inglés, *Random Access Memory*). Es la memoria interna del ordenador donde se van almacenando diferentes datos para posteriormente ser utilizados.

Procesador: Es la parte del hardware del ordenador que interpreta todas las instrucciones de los programas.

Servidor: Software que funciona como contenedor de ficheros que son accesibles por distintos usuarios. Normalmente este software se encuentra en un ordenador que únicamente realiza esta función, por lo que en este caso, también se le llama servidor al equipo en el que se encuentra instalado el software.

Cliente: Parte de la aplicación que se ejecuta en un ordenador que no es el servidor, sino que accede mediante una dirección al contenido que se encuentra en el servidor.

Tomcat: Servidor de páginas JSP y servlets bajo la licencia de la undación Apache.

Clase: Fichero *".java"* en el que, normalmente, se define un tipo de objeto con unas determinadas propiedades.

Servlet: Fichero *".java"* que se utiliza para recoger determinadas acciones que se produzcan en el navegador web del cliente.

Java: Lenguaje de programación orientado a objetos.

JSP: Tecnología que usa el lenguaje Java para crear páginas web dinámicas.

HTML: Lenguaje con el que se elaboran las páginas web

Driver: Software que permite la interacción con otros dispositivos. En este proyecto, por ejemplo, se usa el driver JDBC para que la aplicación conecte con la base de datos.

Javadoc: Lenguaje utilizado para documentar todas las clases escritas en lenguaje Java.

JavaSript: Lenguaje de programación que se ejecuta en la parte del cliente.

jQuery: Biblioteca de JavaScript.

MIME: Extensiones multipropósito de correo de internet (en inglés, *Multipurpose Internet Mail Extensions*). Son una serie de normas que se siguen en el intercambio de ficheros en internet.

URL: Dirección mediante la que se accede a determinados recursos en internet.

GET: Método HTTP que se usa para pedir un recurso a una determinada URL. La solicitud no tiene cuerpo, por lo que toda la información se pasa en la URL. También se guarda información en cabeceras y cookies.

POST: Método HTTP similar al GET, con la diferencia que los datos se pasan en el cuerpo de la petición.

7. Anexo

7.1. Manual de usuario

Al entrar a la aplicación el usuario se podrá registrar, rellenando el formulario y pulsando el botón "UNIRME", o entrar en la aplicación, rellenando el usuario y la contraseña y pulsando el botón "ENTRAR", en el caso de que ya se haya registrado anteriormente.

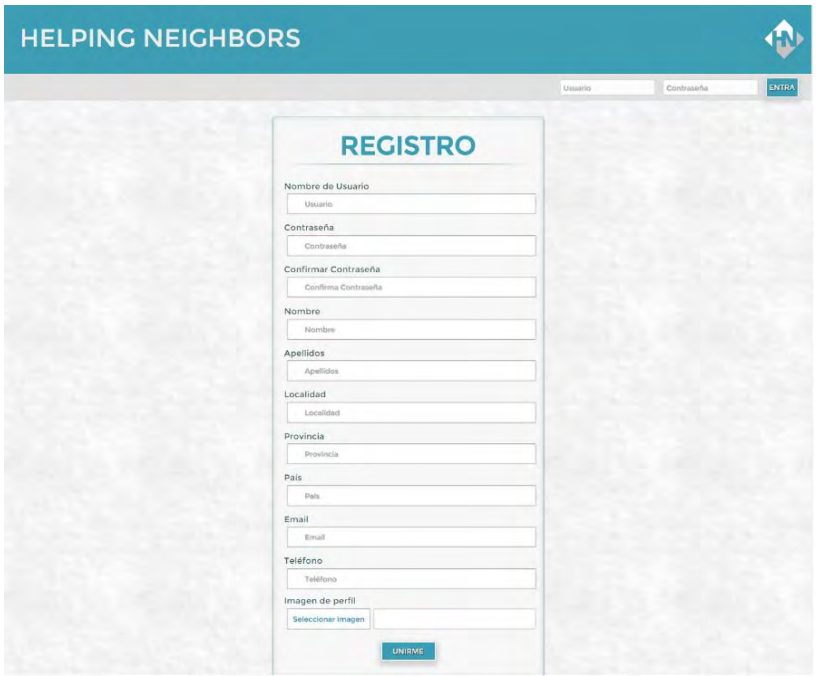

Figura 53. Pantalla de inicio (index.jsp)

Si el usuario, al intentar autenticarse o registrarse en la aplicación, introduce algún dato erróneo, aparecerá un mensaje de error.

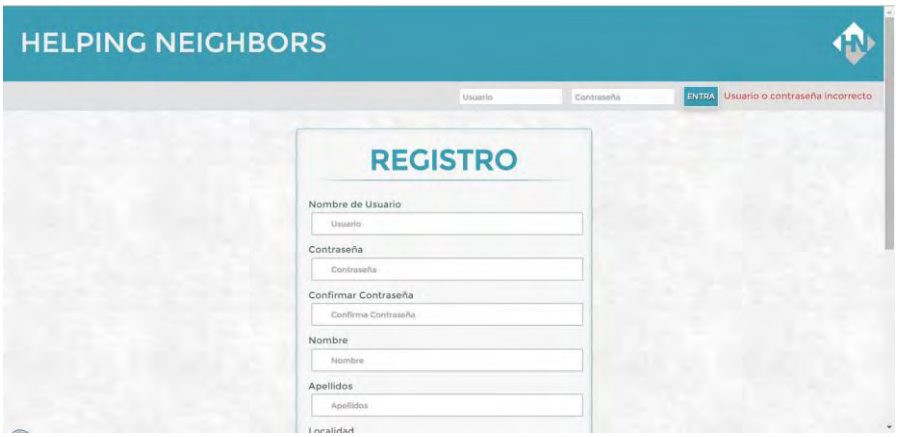

Figura 54. Mensaje de error al autenticarse (index.jsp)

Figura 55. Mensaje de error al registrarse (index.jsp)

En el caso de que el registro de un nuevo usuario sea satisfactorio, se mostrará un mensaje indicando al usuario que puede autenticarse en la aplicación.

Figura 56. Pantalla principal, menú de la izquierda desplegado (inicio.jsp)

Figura 57. Pantalla principal, menú de la derecha desplegado (inicio.jsp)

Una vez el usuario entre en la aplicación podrá realizar varias acciones mediante los menús desplegables de la barra superior. Si el usuario ha publicado

alguna tarea en el último mes se le mostrará en esta misma pantalla una lista de las mismas, junto con dos botones para cada una de ellas, un botón "Eliminar" por si el usuario quiere eliminar la tarea; y otro botón "Detalles" para ver todos los datos de la oferta. Estas funcionalidades se describirán más adelante.

En el menú desplegable "Acciones" se encuentra la opción "Publicar oferta" publicar oferta mediante la que se accederá a la siguiente página.

Figura 58. Pantalla de publicar oferta (oferta.jsp)

Una vez se rellenen los datos de la oferta se pulsará el botón "Publicar" y, si la oferta ha sido publicada, se accederá a la siguiente pantalla para informar al usuario que todo ha ido bien.

Figura 59. Pantalla de éxito (éxito.jsp)

En el menú desplegable "Acciones" también se encuentra la opción "Buscar oferta" mediante la que se accederá a una pantalla para buscar ofertas entre dos fechas.

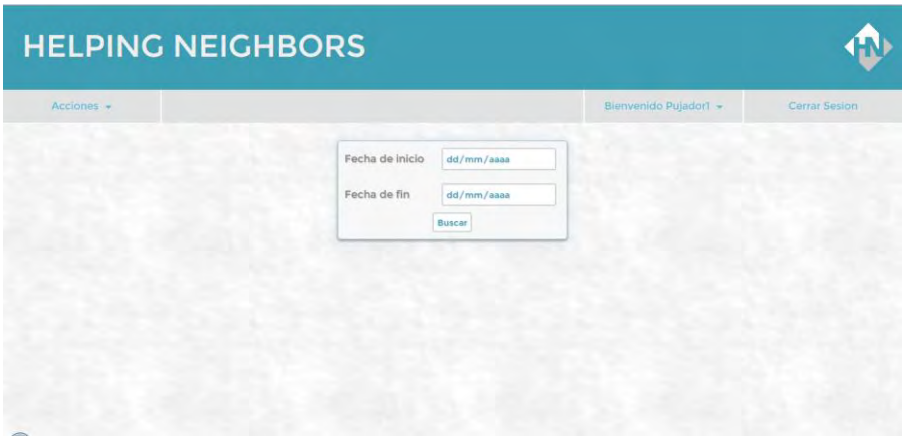

Figura 60. Pantalla de buscar ofertas (buscaroferta.jsp)

Una vez el usuario introduzca las fechas entre las que quiere buscar ofertas pulsará el botón "Buscar".

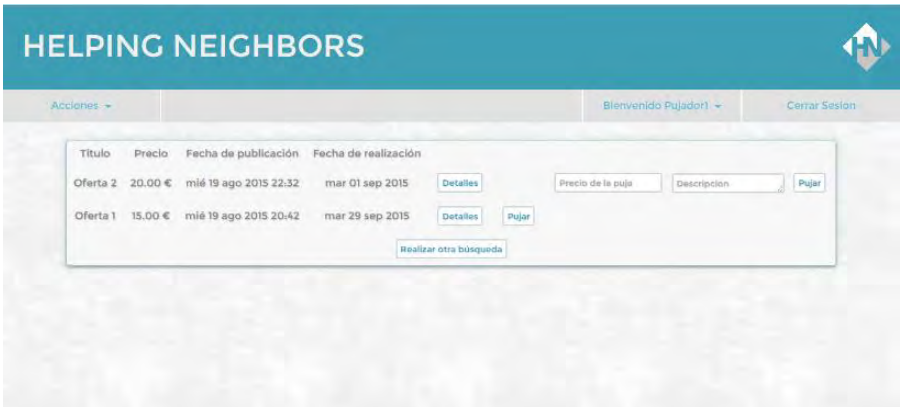

Figura 61. Resultados después de buscar ofertas (buscaroferta.jsp)

Cuando se muestren los resultados, el usuario podrá ver los detalles de la oferta o pujar por ella. Pulsando el botón "Pujar" la primera vez aparecerán en la fila dos campos para introducir el importe de la puja y una descripción de la misma en el caso de que el usuario lo crea conveniente. Para confirmar la puja el usuario pulsará el botón "Pujar" de la fila correspondiente.

También podrá pulsar el botón "Realizar otra búsqueda" para volver a la pantalla de la Figura 60 si los resultados no le satisfacen o desea realizar otra búsqueda.

En el menú desplegable de la derecha el usuario se encuentra con tres opciones, una de ellas es la opción "Mi cuenta" mediante la que accederá a la siguiente pantalla.

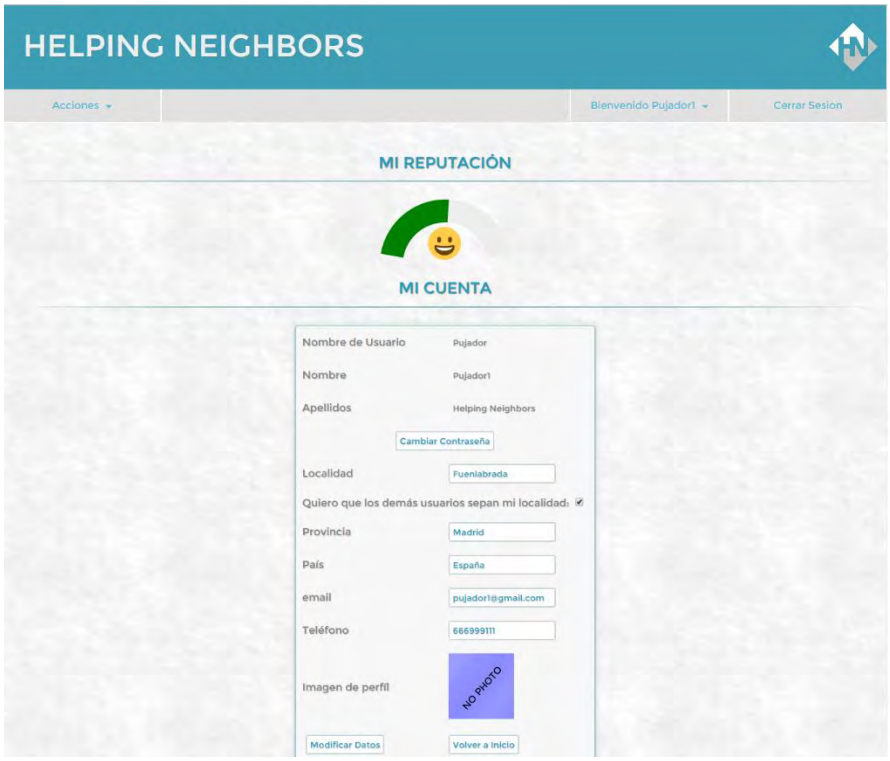

Figura 62. Pantalla de la cuenta del usuario autenticado (micuenta.jsp)

Aquí el usuario podrá cambiar cualquiera de los datos de su cuenta si lo desea, además de poder cambiar la contraseña pulsando el botón "Cambiar contraseña", mediante el que aparecerán los cuadros a rellenar para el cambio de contraseña. Una vez haga todos los cambios pulsará el botón "Modificar datos" para confirmarlos. Si desea volver para atrás pulsará el botón "Volver a inicio" para volver a la pantalla de las Figura 56 y Figura 57.

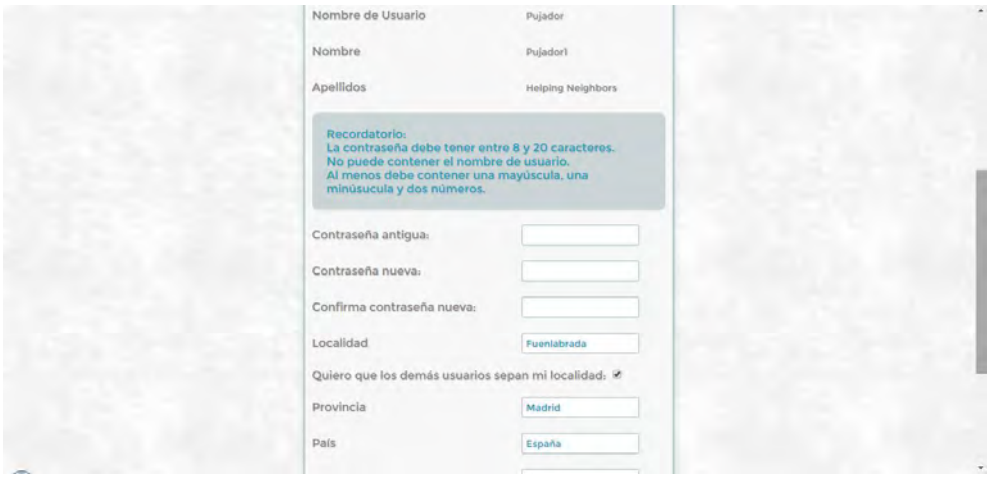

Figura 63. Cuadros de cambiar contraseña (micuenta.jsp)

En el menú desplegable de la derecha también tendrá la opción "Mis ofertas" mediante la que accederá a la siguiente pantalla.

Figura 64. Pantalla de las ofertas del usuario autenticado (misofertas.jsp)

En esta pantalla el usuario podrá eliminar las ofertas que desee mediante el botón "Eliminar" correspondiente a una oferta, además de ver los detalles de la misma mediante el botón "Detalles", accediendo a la siguiente pantalla.

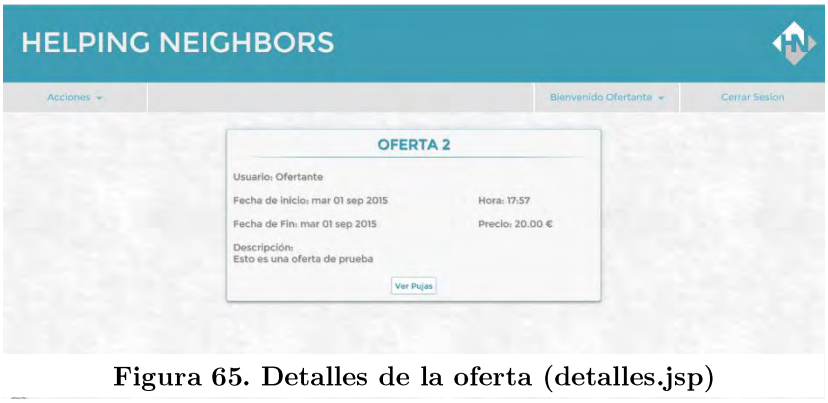

Figura 65. Detalles de la oferta (detalles.jsp)

En esta pantalla el usuario puede ver las pujas que han realizado sobre su oferta pulsando el botón "Ver pujas", de forma que la vista de la pantalla cambiará mostrando también las pujas de la oferta.

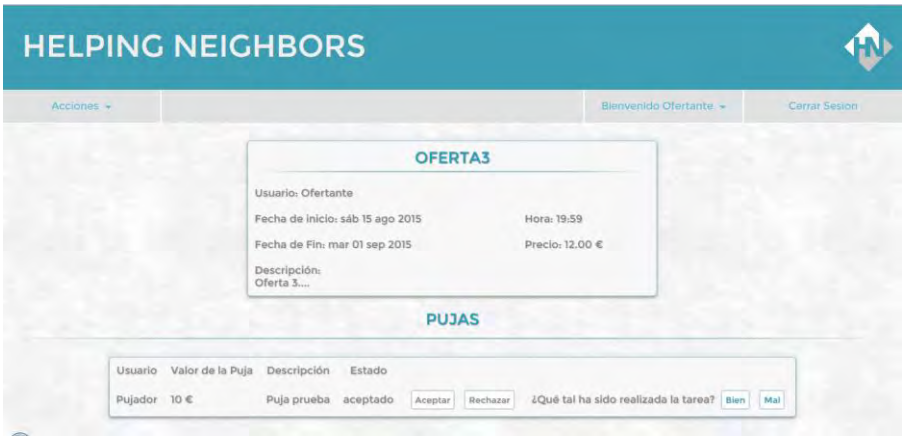

Figura 66. Detalles de la oferta y pujas (detalles.jsp)

Pulsando el nombre del usuario que ha realizado la puja se puede acceder a visualizar la información del pujador.

Figura 67. Información de un usuario (cuentausuario.jsp)

La última opción que se encontrará en el menú desplegable de la derecha será "Mis pujas" mediante la que podrá ver toda la información relevante a las pujas que haya realizado en la siguiente pantalla.

Figura 68. Pujas del usuario

En esta pantalla tendrá la opción de eliminar la puja en el caso de que no haya sido aceptada ni rechazada. Además, pinchando en el título de la oferta pujada podrá ver los detalles de la oferta correspondiente.

Por último, el usuario podrá cerrar sesión pulsando el botón "Cerrar sesión" de la barra superior, de forma que volverá a la pantalla de la Figura 53.

7.2. Presupuesto

Aquí se va a analizar el coste medio que puede tener el proyecto.

- Hosting y dominio durante 1 año: 200€
- Ordenador para desarrollar la aplicación: 700€
- Horas de trabajo: 300 horas x 20 €/hora = 6000€

Importe medio total para desarrollar la aplicación: 6900€

7.3. Extended abstract

The purpose of the thesis is to develop a web application of tasks' offer/demand. In this way, people who cannot do daily tasks', such as take their dog for walk or a deep cleaning of their home, or who are looking for help to do less daily tasks, such as paint their home or assemble a furniture, will be able to publish the tasks and wait until other users show their interest in do it for a respective price. Thus, one user will not have to do that task and the other will win some money by helping others.

This application is thought to connect people who are almost close, living in the same area, but they don't know each other. Thus, they will be able to get to know one another by using the web application. In this way, the neighbors of an area will have the possibility to know each other better and, if they deem it appropriate, ask them for help when necessary.

The origin of the aim of developing this application is the success of this sort of webs in the United States. The best example of this is the social network for neighborhoods named *"Nextdoor"*, which currently is used in over 69000 neighborhoods across the country, according to their website.

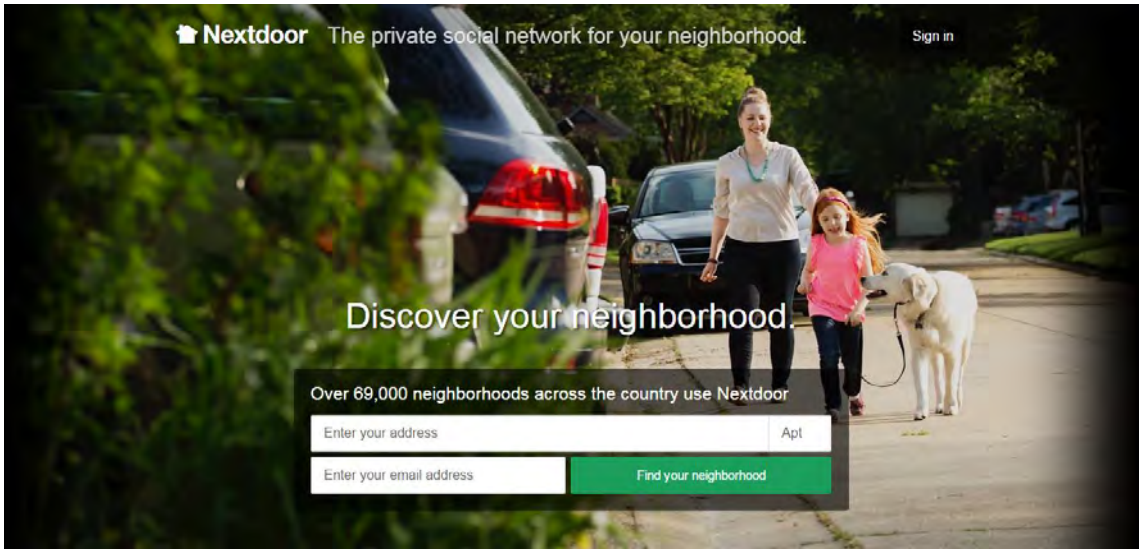

Figura 69. Pantalla principal de *Nextdoor*

Therefore, seeing the success of this web, it is thought that is worth to develop a website similar to *"Nextdoor"* in our country, because, in many cases, the neighbors of an area do not know each other and they cannot ask for help to other local people. This problem should not exist by using this application because the users will be able to ask for help to their neighbors as, even if we do not realize, there is always nice people around us.

The first time people open the web application they are going to need to register in the website by filling the form in the main page with some information such as their location, username or password, among others. When they submit the information filled in the form, these data will be inserted in the database in the corresponding table.

When users are registered in the application, they will be able to log in the web application and access to all the functionalities. So also, they will be able to publish the tasks above named. To do this, they need to set a maximum amount by which the job will be paid to the user who decide to do it. Also, they have to set the time and date. When the information is submitted, these data is sent to the database to be stored.

A user maybe is interested at helping their neighbors doing some tasks which have been published by other users. To do this, a bid system have been implemented in the application. With this system, a task can receive bids from different users so as the offerer have to decide which bid accept on the basis of the characteristics of the bids and the relevant information of the user, such as his/her reputation.

To bid for the tasks, the users have to look for them by setting two dates between which will be found. After this, they have to bid for the task that they want to do. In this bid they have to set the amount they have to earn for doing the task, and this cannot be higher than the amount set by the offerer. Also, they can set a short description of the bid to inform about something the other user needs to know about this bid.

To develop this system is necessary a table in the database to store all the information related with the bid, such as the price, description above named or its state, in addition to the keys to link the bids with the task and the user who is the bidder.

This system is similar to *Ebay* bidding system. However, instead of setting a minimum amount for the item which is going to be published, in this application a maximum amount is set. In this way, while in *Ebay* the bidders only are allowed to bid for a higher amount than the minimum price of the item, in this web application they only are allowed to set a lower amount than the compensation set in the task. As we can see, is a sort of "*Ebay* bidding reversed system".

Moreover, the users have the possibility to choose a profile image either in the register process or once they are registered by modifying their account information. This feature adds a personal touch to their account and allows getting a first impression of the users.

Also, it adds a way to get to know between users in case they need to meet each other because a bid has been accepted. This feature is thought to be necessary in a web app of those characteristics because some people may be more reluctant to accept bids from people who they have not seen ever in their entire life. So this can be used as one of the points which should be considered to accept bids.

To develop this feature is necessary to store all the pictures uploaded by the users so that they can be shown later in the application, and resize them because the users can be upload a large-size image which can overload the server it is not resized. Also, the image must have a unique name for each user in the application so it can be linked automatically by its name to a user. These are the three necessary steps to develop this functionality.

Another key feature is the reputation system developed in the web application. Thanks to this system the users can see if another user is reliable or not according to their reputation because, if the user has a bad reputation, it will mean they have not taken the tasks, which they have done, very seriously and it has not been completed successfully while, if their reputation is good, it means they have given their best in the most of the tasks which they have completed.

Multiple websites have this system of rating basing on the opinion of the users. The most similar to the reputation system exposed above is which we find in *"BlaBlaCar"*. In this website, the users can rate the journey they have done with a user by giving their opinion of it. In this way, the next time that this user publish a journey, the other users who want travel with they can see their reputation and the opinions of the other users who have already traveled with them and, basing on this criteria, decide if they want travel with them or they want to look for another available journey to the same destiny.

As we can see, the reputation system and *"BlaBlaCar"* system are very similar, but in this case, the quality of the job whish have been done in the tasks will be rated instead of journeys. To develop this system, it is necessary to store both the reputation of the users and the opinion of the work accomplished in the database.

In the next figure we can see the resulting database after develop all the key features exposed before and all the web application.

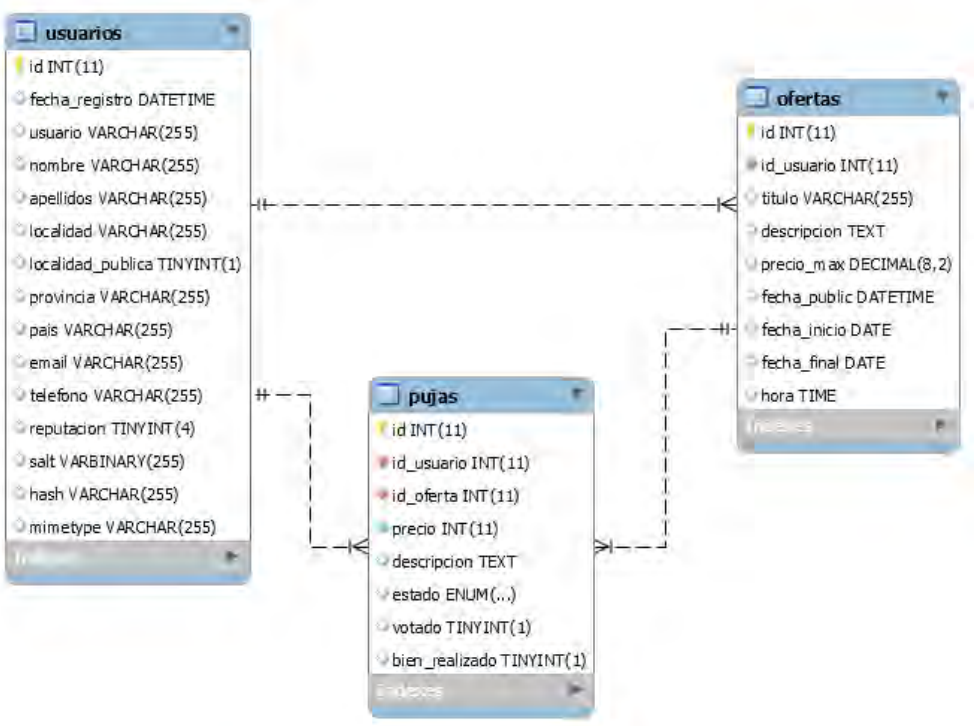

Figura 70. Base de datos

Also, it is necessary a user interface to allow the user interact with the database and can introduce their data in it by filling the forms mentioned in the previous paragraphs. Equally important is the look of this interface because this is the only thing the users are going to see when they are using the web application. So this interface needs to be simple to use and with an attractive design.

Figura 71. Pantalla inicial

The first version of this web application has the main features which were exposed at the beginning of the thesis as the minimum requirements. The aim of this version is show the idea of the web application.

However, in the future, many improvements can be made to turn the web application into a sort of social network in which people who are in the same area can publish tasks that they cannot do at any moment and wait for anyone living around who is interested in doing the job.

The main functionality to turn the application into a social network is to develop a way users can communicate internally in the web application. This functionality can be private messages between users or messages inside each published task so that the offerer and the bidders can communicate among themselves to exchange information between them. Both ways of communication can be implemented to enrich the application.

Another improvement to make the application more user-friendly is to implement notifications when the user receive a new bid for one of their offers or another change in the website which is related to them. This improvement will modernize the web application and will enrich the design of the web application.

Finally, the opinion of the users is necessary to be considered because for the success of the social networks is crucial that the users like the application because the more users the application have more successful it will have.

8. Bibliografía

[1] Internet World Stats, «Internet World Stats,» [En línea]. Available: http://www.internetworldstats.com/stats4.htm. [Último acceso: 24 Julio 2015].

[2] S. Kemp, «Digital, Social & Mobile Worldwide in 2015,» We are social, 21 Enero 2015. [En línea]. Available: http://wearesocial.net/blog/2015/01/digital-social-mobile-worldwide-2015/. [Último acceso: 25 Julio 2015].

[3] «Instituto Nacional de Estadística,» [En línea]. Available: http://www.ine.es/. [Último acceso: 25 Julio 2015].

[4] IAB Spain, «Slideshare,» 28 Enero 2015. [En línea]. Available: http://es.slideshare.net/elogia/estudio-redes-sociales-iab-spain-2015?from_action=save. [Último acceso: 25 Julio 2015].

[5] «Nextdoor,» [En línea]. Available: https://nextdoor.com/. [Último acceso: 25 Julio 2015].

[6] «Help your neighbor,» [En línea]. Available: http://www.helpyourneighbor.com/about. [Último acceso: 26 Julio 2015].

[7] N. Salnikov-Tarnovski, «Most popular Java EE containers: 2015 edition,» 15 Abril 2015. [En línea]. Available: https://plumbr.eu/blog/java/most-popular-java-ee-containers-2015-edition. [Último acceso: 18 Julio 2015].

[8] Apache, «Apache Tomcat 8 Users guide - JNDI Datasource HOW-TO,» [En línea]. Available: https://tomcat.apache.org/tomcat-8.0-doc/jndi-datasource-examples-howto.html. [Último acceso: 15 Agosto 2015].

[9] MySql, «Driver/Datasource Class Names, URL Syntax and Configuration Properties for Connector/J,» [En línea]. Available: http://dev.mysql.com/doc/connector-j/en/connector-j-reference-configuration-properties.html. [Último acceso: 15 Agosto 2015].

[10] Apache, «Apache Tomcat 8 Application Developer's Guide,» [En línea]. Available: http://tomcat.apache.org/tomcat-8.0-doc/appdev/deployment.html. [Último acceso: 15 Agosto 2015].

[11] R. Fielding, U. Irvine y J. Gettys, «[RFC2616] Hypertext Transfer Protocol -- HTTP/1.1,» 1999. [En línea]. Available: https://www.ietf.org/rfc/rfc2616.txt. [Último acceso: 28 Julio 2015].

[12] Oracle, «JavaServer Pages Technology,» [En línea]. Available: http://www.oracle.com/technetwork/java/javaee/jsp/index.html. [Último acceso: 15 Agosto 2015].

[13] «jQuery UI,» jQuery, [En línea]. Available: http://code.jquery.com/ui/. [Último acceso: 15 Agosto 2015].

[14] «Twitter Emoji,» [En línea]. Available: https://twitter.github.io/twemoji/. [Último acceso: 15 Agosto 2015].

[15] A. Terrien, «jQuery Knob,» [En línea]. Available: http://anthonyterrien.com/knob/.