UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. de Telecom., Sonido e Imagen

“Desarrollo de una aplicación web, con Front-end y Back-end, para compraventa de segunda mano”

TRABAJO FINAL DE GRADO

Autor/a:
Nieto Rodrigo, Jorge

Tutor/a:
Marín-Roig Ramón, José

GANDIA, 2016

Resumen

Desde el CDM (Centro De Movilidad) de la multinacional ‘Babel, sistemas de información’ se motiva al autor a diseñar y crear una página web de compraventa de productos de segunda mano la cual sirva como proyecto de fin de carrera.

La empresa tiene como objetivo para este proyecto:

- Implementar una web, tanto el front-end como su back-end.
- Implementar medidas de seguridad y su corroboración.
- Trabajar con metodologías agiles de programación como Scrum.
- Trabajar con Git para la creación de copias de seguridad.

La empresa española multinacional ‘Babel, sistemas de información’ presta servicios tecnológicos orientados a acompañar a grandes organizaciones en su proceso de transformación digital. Además, avalan su experiencia las 150 organizaciones que han depositado su confianza como Iberdrola, Airbus, grupo Santander, BBVA, ONO, el Ministerio de Justicia…

Así pues, con esta web, pretende ser el inicio de un proyecto para una nueva compañía. Una vez terminada la primera versión de prueba se pretende presentar para la obtención del proyecto y de un nuevo cliente.

Palabras clave: ‘Front-end’, ‘back-end’, ‘commit’, GitHub y SourceTree.

Abstract

From the CDM (Center Of Mobility) of the multinational 'Babel, information systems' is motivated to the author to designing and programe a web page of dealing product of the second hand which serves as project of end of career.

The company has as aim for this project:

- To implement a web, both the front-end and its back-end.
- To implement safety measures and its corroboration.
- To work with agile methodologies of programming as Scrum.
- To work with Git for the creation of safety copies.

The Spanish multinational company 'Babel, information systems' offers technological services orientated to accompanying big organizations in their process of digital transformation. In addition, their experience is supported by 150 organizations which have deposited their confidence, such as Iberdrola, Airbus, Santander Group, BBVA, ONO, el Ministerio de Justicia…

All in All, this web tries to be the beginning of a project for a new company. Once its first versions and tests are finished, it will be introduced, looking forward to a new client and a posible contract.

Keywords: ‘Front-end’, ‘back-end’, ‘commit’, GitHub and SourceTree.

Tabla de contenido

Tabla de ilustraciones

Capítulo 1.

Introducción

1.1 Descripción

El trabajo final de grado que me ocupa es la creación de una página web similar a la aplicación móvil de 'Wallapop', ya que su página web tiene restricciones en comparación con la aplicación móvil.

'Wallapop' es una aplicación para el móvil que permite comprar, vender y/o intercambiar artículos de segunda mano o nuevos con la ventaja de la geolocalización, es decir, permite buscar y ofrecer productos por cercanía y por zonas. Los usuarios se registrarán gratuitamente y podrán comenzar a ofertar sus productos.

En el proyecto que se va a explicar a continuación se crearán los servicios para el 'back-end' (servicios transparentes al usuario), se diseñarán las diferentes vistas del 'front-end' (interfaz gráfico de la página web) y sus correspondientes bases de datos, tanto para usuarios como para anuncios.

En la fase de preproducción se procederá a detallar los criterios técnicos y artísticos a seguir para el diseño de las vistas y se decidirá las tecnologías para cada sección del proyecto. En la producción se creará el repositorio en 'GitLab' (versión privada de GitHub), se crearán las vistas y se desarrollarán los controladores de cada vista. Finalmente, se describirá el testeo realizado y la última depuración.

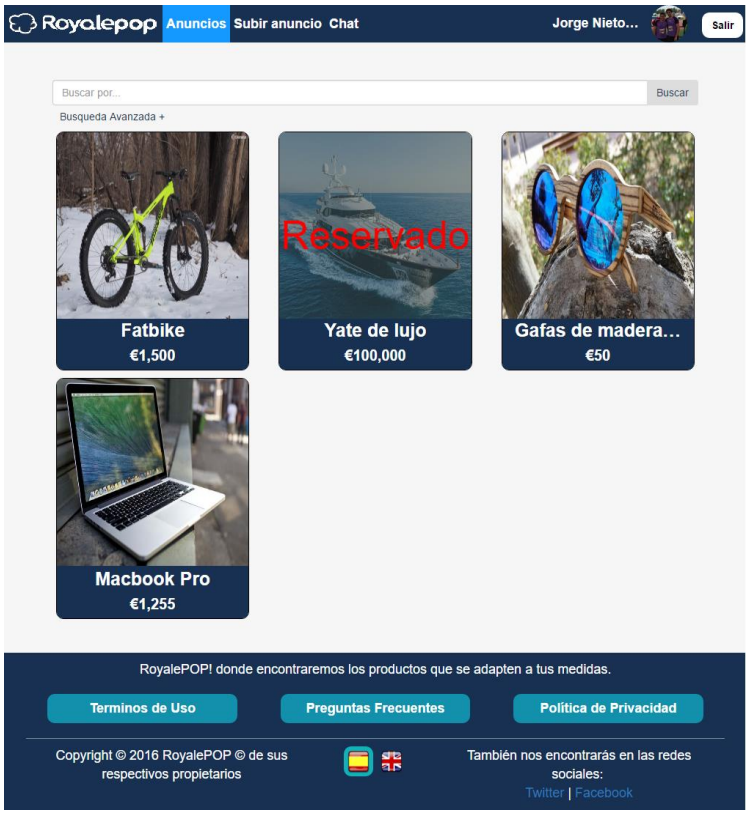

Ilustración 1. Main menú.

1.2 Objetivos

- Objetivos principales:
  - Diseñar una aplicación de '*front-end*'.
  - Diseñar una aplicación de '*back-end*'.
- Objetivos secundarios:
  - Implementación de métodos de seguridad.
  - Realización de testeo automático, en este caso con '*protractor*'.
  - Detección de fallos de seguridad.
  - Gestión de bases de datos.
  - Desarrollo de proyecto mediante 'Git' con '*SourceTree*' para el control de versiones.

1.3 Metodología de trabajo

Se usarán metodologías agiles de desarrollo como 'Scrum' ya que:

- Adopta una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.
- Se basa en la calidad del resultado más en el conocimiento tácito de las personas en equipos auto organizados, que en la calidad de los procesos empleados.
- Evita solapamiento de las diferentes fases del desarrollo, en lugar de realizar una tarea tras otra en un ciclo secuencial o en cascada.

Scrum es un modelo de referencia que define un conjunto de prácticas y roles.

Los roles principales son:

- El ***cliente*** recibe el producto y puede influir en el proceso, entregando sus ideas o comentarios respecto al desarrollo.
- ***Product Owner***, representa la voz del cliente. Se asegura de que el equipo trabaje de forma adecuada desde la perspectiva del negocio. Es **'**el jefe' responsable del proyecto.
- ***Scrum Master*** (SM) o facilitador de proyectos, es la figura que lidera los equipos en la gestión ágil de proyectos. Su misión es que los equipos de trabajo alcancen sus objetivos hasta llegar a la fase de ''*sprint' final*', eliminando cualquier dificultad que puedan encontrar en el camino. Es el encargado de hablar con el product owner para desarrollar el producto conforme a sus especificaciones.
- El ***equipo*** que ejecuta el desarrollo y demás elementos relacionados con él.

1.3.1 Proceso de desarrollo

Durante cada '*sprint'*, (un periodo entre una y cuatro semanas, magnitud definida por el equipo, siendo esta lo más corta posible), el equipo crea una serie de tareas potencialmente entregables (utilizable).

El *Product Owner*, en el Sprint Planning, identifica las tareas que quiere ver completados y se las muestra al equipo. Entonces, el equipo conversará con el Product Owner buscando la claridad y la magnitud adecuada para luego determinar la cantidad de ese trabajo que podrá comprometerse a completar durante el '*sprint*'.

En la siguiente captura se podrá observar el listado de tareas de un proyecto en '*Kunagi'* (plataforma para desarrollo de trabajo con metodología con srcum).

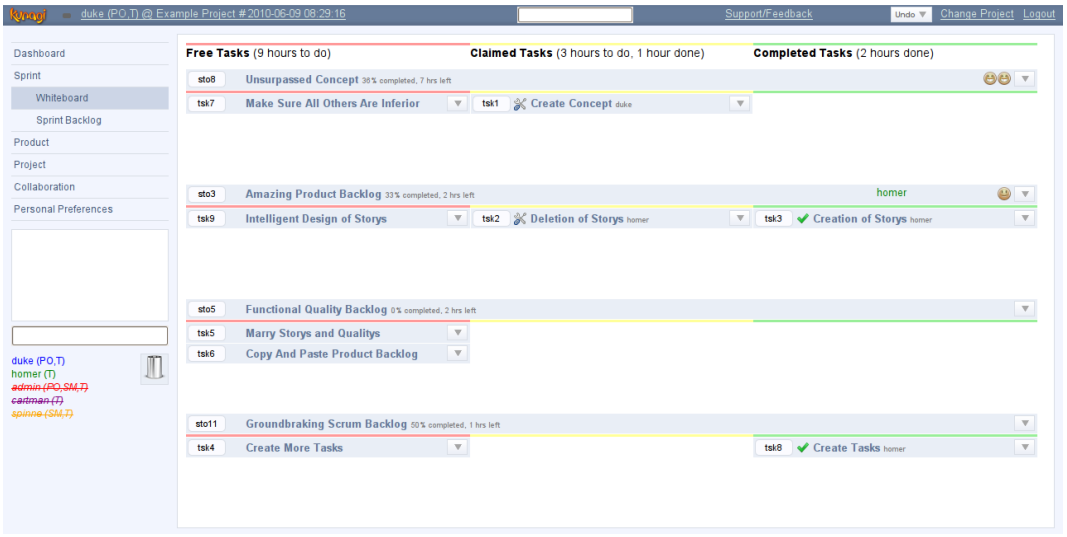

Ilustración 2. Listado de tareas de un 'sprint'.

En esta plataforma se pueden organizar las tareas por el equipo, añadir un tiempo a cada tarea, normalmente por el Scrum Master… y finalmente ver la gráfica del tiempo empleado para cada tarea a lo largo del '*sprint*'. Además, si quedasen tareas pendientes por terminar se añadirán automáticamente al inicio de la pila de tareas del siguiente '*sprint*'.

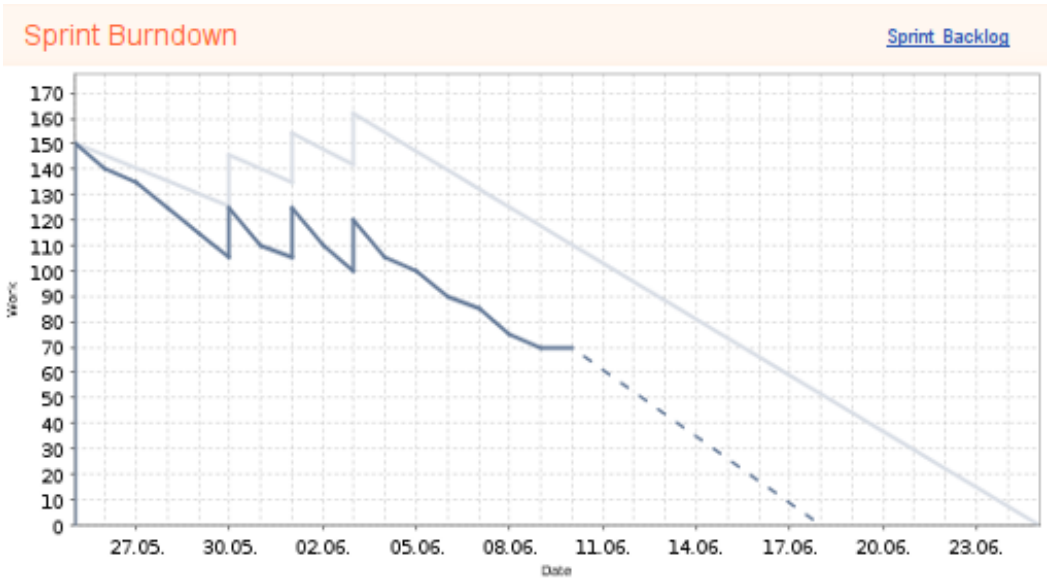

Ilustración 3. Sprint Burndown.

1.4 Etapas del proyecto

Este proyecto se ha realizado siguiendo las siguientes fases:

1. Etapa de formación: curso ‘web Development Bootcamp’.
2. Diseño de las diferentes pantallas con las que va a contar la aplicación.
3. Estudio de la organización del proyecto.
4. Creación del proyecto con sus respectivas carpetas y bibliotecas de ayuda.
5. Inicialización del proyecto en GitLab.
6. Creación de las bases de datos.
7. Implementación de las diferentes vistas.
8. Implementación del servidor, con los envíos de datos entre cliente y servidor.
9. Diseño de ‘CSS’ para darle formato ‘atractivo’ e intuitivo a la página.
10. Documentación del código.
11. Testeo de la aplicación con ‘protractor’.
12. Resolución de problemas de seguridad.
13. Testeo de la aplicación, por parte de un usuario externo el cual nunca haya probado la aplicación.
14. Resolución de problemas de problemas de interactuación con la página y las correspondientes mejoras en el ‘CSS’.
15. Segundo testeo de la aplicación por parte de un usuario externo, el cual nunca haya probado la aplicación.
16. Solucionar la interactuabilidad entre usuario y aplicación.

Nota: Para la creación de nuevos métodos, clase, archivo, etc. Y para la realización de cambios se ha utilizado siempre ‘Git’ con ‘SourceTree’ para el control de versiones.

Capítulo 2.

Tecnologías empleadas

En el desarrollo de aplicaciones es importante diferenciar las partes con las que el proyecto cuenta y emplear las tecnologías más apropiadas para cada parte del proyecto. Por este motivo hay que diferenciar dos apartados en el desarrollo web:

- Front-end
- Back-end, en la que está incluida la base de datos.

Además de estos dos apartados hay que tener en cuenta el control de versio*nes*, ya que normalmente se suele desarrollar en grupos, y el seguimiento del trabajo. Por estos motivos también se detallarán los programas empleados.

2.1 Front-end

En diseño de software el *front-end* es la parte del software que interactúa con el o los usuarios. En el ***Anexo 1*** se pueden observar las diferentes vistas de la aplicación, obteniendo así algunos ejemplos de front-end.

2.1.1 HTML5

HTML5 (*HyperText Markup Language*, versión 5) es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. HTML5 especifica dos variantes de sintaxis para HTML: una «clásica», HTML (text/html), conocida como *HTML5*, y una variante XHTML conocida como sintaxis *XHTML5* que deberá servirse con sintaxis XML (application/xhtml+xml).

Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo. Es un estándar del consorcio W3C desde el 5 de octubre de 2014.

Ilustración 4.
Logotipo HTML5.

Características de HTML5:

- Introduce etiquetas con significado semántico.
- Introduce multitud de APIs para JavaScript.
- Permite crear etiquetas propias.
- Incorpora etiquetas (canvas 2D y 3D, audio, vídeo) con codecs para mostrar los contenidos multimedia.
- Mejoras en los formularios. Nuevos tipos de datos (email, numero, url…) y facilidades para validar el contenido sin Javascript.
- Nuevas funcionalidades para arrastrar objetos como imágenes, Drag & Drop.
- Nuevos elementos multimedia: audio, video, WebWorker (Hilo de ejecución en paralelo), geoposicionamiento…

2.1.2 CSS

CSS (*Cascading Style Sheets, hojas de estilo en cascada*), es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML2 (y por extensión en XHTML). El World Wide Web Consortium (W3C) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los navegadores.

CSS se ha creado en varios niveles y perfiles. Cada nivel de CSS se construye sobre el anterior, generalmente añadiendo nuevas funciones al previo.

Los perfiles son, generalmente, parte de uno o varios niveles de CSS definidos para un dispositivo o interfaz particular. Actualmente, pueden usarse perfiles para dispositivos móviles, tabletas, impresoras o televisiones de cualquier tamaño.

CSS3 a diferencia de CSS2 fue una única especificación que definía varias funcionalidades, pudiendo estar dividida en varios documentos separados, llamados 'módulos'.

Cada módulo añade nuevas funcionalidades a las definidas en CSS2, de manera que se preservan las anteriores para mantener la compatibilidad.

Los trabajos en el CSS3, comenzaron a la vez que se publicó la recomendación oficial de CSS2, y los primeros borradores de CSS3 fueron liberados en junio de 1999.

Debido a la modularización del CSS3, diferentes módulos pueden encontrarse en diferentes estados de su desarrollo, de forma que, a fechas de noviembre de 2011, hay alrededor de cincuenta módulos publicados, tres de ellos se convirtieron en recomendaciones oficiales de la W3C en 2011: *'Selectores'*, *'Espacios de nombres'* y *'Color'*.

Algunos módulos, como *'Fondos y colores'*, *'Consultas de medios'* o *'Diseños multicolumna'* están en fase de 'candidatos', y considerados como razonablemente estables, a finales de 2011, y sus implementaciones en los diferentes navegadores son señaladas con los prefijos del motor del mismo.

Ilustración 5. Logotipo CSS3.

Propiedades de CSS:

- Propiedades de las Fuente, como tipo, tamaño, énfasis...
- Color de texto, fondos, bordes u otros elementos.
- Alineación de textos, imágenes, tablas u otros elementos.
- Propiedades de caja, como margen, borde, relleno o espaciado.
- Atributos del texto, como espaciado entre palabras, letras, líneas, etcétera.
- Propiedades de identificación y presentación de listas.
- Funcionalidades propias de las capas ( ) como de posicionamiento relativo/absoluto/fijo, niveles (z-index), etcétera.
- Soporte para las hojas de estilo auditivas.
- Texto bidireccional, sombras, etcétera.

2.1.3 Bootstrap

Bootstrap es el framework más popular para el desarrollo adaptativo de webs. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales.

En agosto del 2011, se liberó Bootstrap como código abierto. En febrero del 2012, se convirtió en el proyecto de desarrollo más popular de GitHub.

Bootstrap tiene un soporte relativamente incompleto para HTML5 y CSS3, pero es compatible con la mayoría de los navegadores web. La información básica de compatibilidad de sitios web o aplicaciones está disponible para todos los dispositivos y navegadores. Existe un concepto de compatibilidad parcial que hace disponible la información básica de un sitio web para todos los dispositivos y navegadores. Por ejemplo, las propiedades introducidas en CSS3 para las esquinas redondeadas, gradientes y sombras son usadas por Bootstrap a pesar de la falta de soporte de navegadores antiguos. Esto extiende la funcionalidad de la herramienta, pero no es requerida para su uso.

Desde la versión 2.0 también soporta diseños sensibles. Esto significa que el diseño gráfico de la página se ajusta dinámicamente, tomando en cuenta las características del dispositivo usados.

Características de Bootstrap:

- Sistema de rejilla siendo este un diseño adaptativo y fiable.
- Reconoce las hojas de estilo de CSS.
- Componentes re-usables.
- Permite añadir plugins de JavaScript.

Ilustración 6. Logotipo Bootstrap.

2.1.4 AngularJS

AngularJS, o simplemente Angular, es un framework de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

La biblioteca lee el HTML que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript. Los valores de las variables de JavaScript se pueden configurar manualmente, o son recuperadas de los recursos JSON estáticos o dinámicos.

AngularJS se combina con el entorno en tiempo de ejecución Node.js, el framework para servidor Express.js y la base de datos MongoDB para formar el conjunto MEAN (Acrónimo para: MongoDB, ExpressJS, AngularJS, NodeJS).

AngularJS está construido en torno a la creencia de que la programación declarativa es la que debe utilizarse para generar interfaces de usuario y enlazar componentes de software, mientras que la programación imperativa es excelente para expresar la lógica de negocio.

Este framework adapta y amplía el HTML tradicional para servir mejor contenido dinámico a través de un data binding bidireccional que permite la sincronización automática entre modelos y vistas. Como resultado, AngularJS pone menos énfasis en la manipulación del DOM (Document Object Model), mejora la testeabilidad y el rendimiento.

Objetivos a la hora de diseñar:

Ilustración 7. Logotipo AngularJS.

- Disociar la manipulación del DOM de la lógica de la aplicación. Esto mejora la capacidad de prueba del código.
- Considerar a las pruebas de la aplicación como iguales en importancia a la escritura de la aplicación. La dificultad de las pruebas se ve reducida drásticamente por la forma en que el código está estructurado.
- Disociar el lado del cliente de una aplicación del lado del servidor. Esto permite que el trabajo de desarrollo avance en paralelo, y permite la reutilización de ambos lados.
- Guiar a los desarrolladores a través de todo el proceso del desarrollo de una aplicación: desde el diseño de la interfaz de usuario, a través de la escritura de la lógica del negocio, hasta las pruebas.

Angular sigue el patrón MVC de ingeniería de software y alienta la articulación flexible entre la presentación, datos y componentes lógicos. Con el uso de la inyección de dependencias, Angular lleva servicios tradicionales del lado del servidor, tales como controladores dependientes de la vista, a las aplicaciones web del lado del cliente. En consecuencia, gran parte de la carga en el back-end se reduce, lo que conlleva a aplicaciones web mucho más ligeras.

2.2 Back-end

El back-end es el encargado del procesamiento de los datos de entrada desde el front-end, siendo este el controlador de la vista y realizando la conexión con la base de datos.

En este apartado se detallarán las tecnologías empleadas para los controladores y la creación de la base de datos.

2.2.1 MongoDB

MongoDB (de la palabra en inglés '*humongous*' que significa enorme) es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto.

Ilustración 8. Logotipo MongoDB.

MongoDB forma parte de la nueva familia de sistemas de base de datos NoSQL ('no sólo SQL, sino que es una base de datos con open source). En lugar de guardar los datos en tablas como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos en documentos similares a archivos JSON con un esquema dinámico (MongoDB utiliza una especificación llamada BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

MongoDB fue fundada en 2007 por DoubleClick, ShopWiki y Gilt Groupe. En DoubleClick había desafíos diarios con el tratamiento, el almacenamiento y con el escalado de los datos, por este motivo, DoubleClick escribió su propio software para solventar dichos problemas.

En 2009 MongoDB fue lanzado como un producto independiente y publicado bajo la licencia de código abierto AGPL (Affero General Public License, es una licencia derivada de la Licencia Pública General de GNU diseñada específicamente para asegurar la cooperación con la comunidad en el caso de software que corra en servidores de red).

En marzo de 2011, se lanzó la versión 1.4 y se consideró ya como una base de datos lista para su uso en producción.

2.2.2 Express

Express.js está basado en Connect, el cual es un framework extensible de manejo de servidores HTTP, provee *plugins* de alto rendimiento conocidos como *middleware*. Middleware es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos.

Ilustración 9. Logotipo Express.

Express es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones móviles y web, facilitando un rápido desarrollo basado en nodos.

Características de Express:

- Permite configurar middleware para responder a las peticiones HTTP.
- Define una tabla de enrutamiento que se utiliza para llevar a cabo diferentes acciones, basadas en el método HTTP y URL.
- Permite representar dinámicamente páginas HTML basadas en el envío de argumentos de las plantillas.

2.2.3 NodeJS

Node.js es una librería y entorno de ejecución de E/S (Entrada / Salida) dirigida por eventos y por lo tanto asíncrona que se ejecuta sobre el intérprete de JavaScript creado por Google V8.

Ilustración 10. Logotipo NodeJS.

Trabaja con un único hilo de ejecución que es el encargado de organizar todo el flujo de trabajo que se deba realizar.

Para poder trabajar de una forma óptima Node.js delega todo el trabajo en un pool de threads (hilos). Este pool de threads está construido con la librería libuv. Esta librería dispone de su propio entorno multithread (multitarea) asíncrono. Node.js envía el trabajo que hay que realizar al pool.

Cuando se trabaja con Node.js prácticamente toda la programación es asíncrona y se parece tanto a las clásicas llamadas **AJAX**, además dispone de una función de callback ya que prácticamente todo el trabajo se delega.

Actualmente, por defecto tiene un límite de memoria de 512 MB en sistemas de 32 bits, 1gb en sistemas de 64 bits. El límite se puede aumentar, pero se recomienda dividir el proceso en varios 'workers' si se llega a los límites.

La última versión usada en producción es la v0.12.7 (versión estable) liberada el 9 Julio de 2015. Recientemente se ha liberado la versión v4.0, uniendo Node.js e io.js.

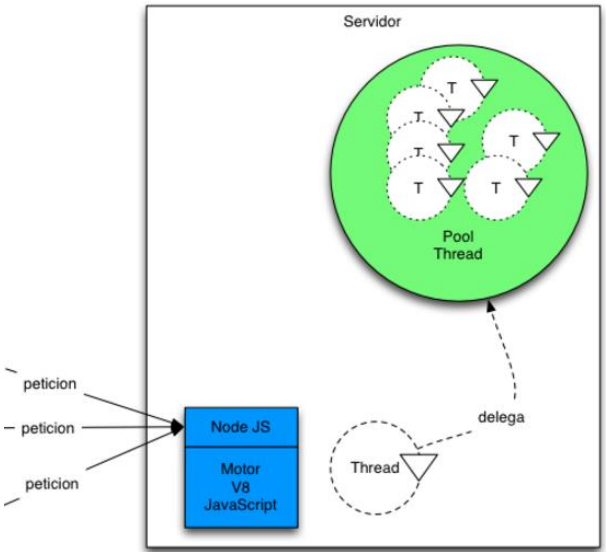

Ilustración 11. Hilos/Threads.

2.3 Control de versiones

2.3.1 Git y GitLab

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

Ilustración 12. Logotipo de Git.

GitLab es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git. Los repositorios se pueden crear públicos o privados.

Entre las características más importantes de Git se encuentran:

- **Cambio de ramas sin errores.** Crear una rama de prueba, realizar un par de commits, volver a donde ramificaba y fusionarlo/mergearlo.
- **Basado en roles**. Tener una rama que siempre contiene sólo lo que va a la producción, otra que se fusionan en el trabajo para las pruebas, y varios más pequeños para el día a día del trabajo.

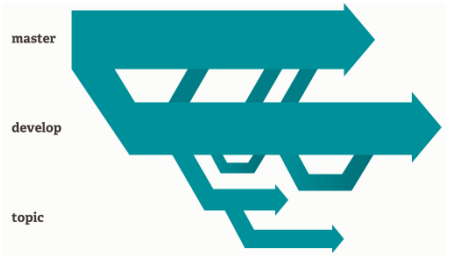

Ilustración 13. Diversificacion de Ramas en Git.

- **Basado en función del flujo de trabajo.** Posibilidad de crear nuevas ramas para cada nueva funcionalidad en las que se está trabajando. Mientras que se está desarrollando una nueva funcionalidad se puede cambiar de rama para observar los cambios realizados, volver a terminar la funcionalidad y una vez finalizada se puede eliminar cuando se mergean las funcionalidades.
- **Experimentación desechable.** Crear una rama para experimentar, pero al darse cuenta de que no va a funcionar, se suprime, abandonando el trabajo con nadie más volver a ver a él (incluso si usted ha empujado otras ramas en el ínterin).

2.3.2 SourceTree

SourceTree es una herramienta visual para el uso de DCVS (control de versiones distribuido, permite que múltiples desarrolladores de software puedan trabajar en un proyecto determinado sin que tengan que compartir una red común).

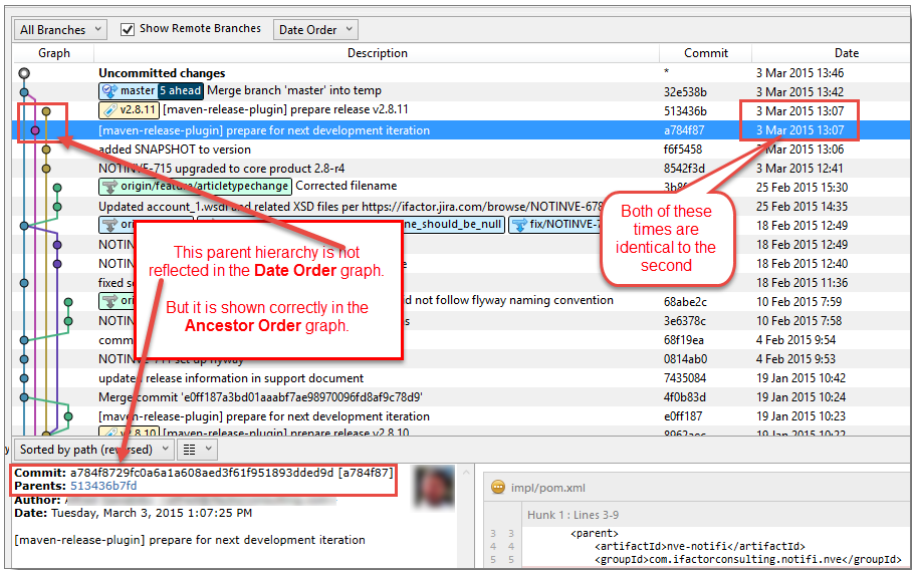

Ilustración 14. Ejemplo de ramas/branches en un proyecto.

2.4 Seguimiento del trabajo

2.4.1 Kunagi

Kunagi es una herramienta, de software libre, utilizada mediante una web para la gestión de proyectos integrada y de colaboración basado en Scrum.

No sólo ofrece la gestión de los documentos básicos de Scrum, sino también una gran variedad de datos adicionales. Además, incluye varias características para la facilidad de uso y la colaboración.

Kunagi pretende ser accesible y adecuado tanto para el desarrollo profesional y no profesional de proyectos de cualquier tamaño.

Capítulo 3.

Fase de estudio previo

3.1 Diseños previos

En este apartado se diseñó las diferentes vistas, aunque en producción podrían sufrir leves modificaciones. En el caso de esta página se van a realizar vistas similares para que al cambiar de dispositivo la página se ajuste según el ancho de la pantalla.

Para una mayor adaptabilidad a los dispositivos será recomendable empezar todos los diseños por las versiones móviles, posteriormente móviles en formato ancho, tablets y para finalizar ordenadores o grandes pantallas. El motivo de seguir este proceso para el diseño de las vistas se encuentra en la dificultad de adaptar todos los campos en una pantalla de tamaño pequeña y posteriormente las restantes, siendo estas más sencillas.

En este proyecto se ha usado el prototipo de rejillas de *Bootstrap* para una mayor adaptabilidad y sencillez a la hora de implementar el código.

En el ***'Anexo 1'*** se pueden ver los diseños de las diferentes vistas.

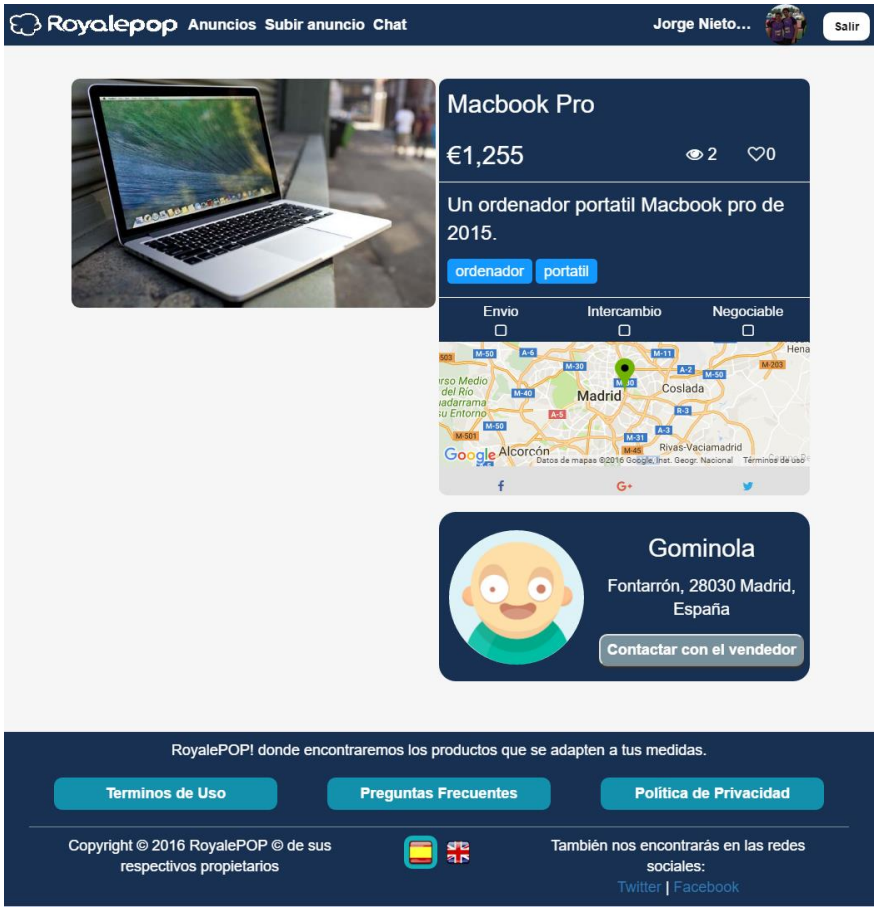

Ilustración 15. Detalle de un anuncio ajeno.

3.2 Arquitectura del proyecto

En esta fase previa se diseña la arquitectura del proyecto la cual consiste en la creación del directorio de carpetas y en una primera estructura de la base de datos. En la siguiente captura se puede observar el directorio general del proyecto.

A continuación, se detallan las carpetas más importantes del proyecto:

- Carpeta **'utils'** la cual se encuentra incluida en **'public':** Contienes las distintas carpetas relativas al back-end (controladores, archivos *'js'*) y sus vistas (archivos *'html'* de front-end).
- Carpeta **'routes':** contiene los diferentes archivos encargados de realizar las diferentes peticiones.
- **'*app.js*':** Archivo con declaración de rutas para el servidor.
- ***'inicializateDB.js':*** Archivo ejecutable para la inicialización de la base de datos con valores predeterminados de la carpeta BBDD.
- ***'README.md':*** Es el archivo más importante de abrir y leer, ya que cuenta con las instrucciones del proyecto.

En el **anexo 2.1** se explica con detalle el directorio de carpetas y la utilidad de cada una de ellas.

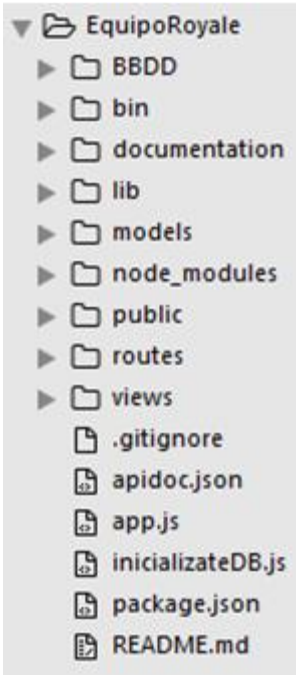

Ilustración 16. Carpeta general del proyecto.

Capítulo 4.

Fase de producción

4.1 Base de datos

En este apartado se detallará la estructura de la base de datos y los campos que contiene cada una.

Para el proyecto desarrollado se ha tenido en cuenta tres características, se tienen una parte con usuarios, los cuales se registran para acceder y posteriormente unos anuncios, además de un chat para la comunicación entre usuarios. Por este motivo se hicieron tres bases de datos. Dos de ellas interconectadas entre sí para que al realizar búsquedas posteriormente fuera más sencillo y una para guardar todos los mensajes que se envían los usuarios.

Se disponen de tres archivos '.json' los cuales contienen las directrices para la creación de todos los archivos '.json' que constituirán la base de datos.

- announcements.json
- chat-model.js
- user-model.js

La ***base de datos de anuncios*** está compuesta por los siguientes campos:

- **Título del anuncio:** de tipo *String*, con un mínimo de 5 y un máximo de 50.
- **Descripción:** de tipo *String* y un mínimo de 10.
- **Imagen:** de tipo *String*.
- **Tags:** de tipo Object. Es un array de 5 tags/etiquetas como máximo.
- **Precio:** de tipo *Number*, con un mínimo de 1 y un máximo de 12.
- **Propietario:** de tipo *String*. Este campo se asigna desde el back-end cuando un usuario sube un nuevo anuncio.
- **Identificador del propietario:** de tipo *String*. Se asigna desde el back-end y es el identificador único del usuario.
- **Opciones:** está compuesto por 3 campos de tipo *Boolean* (verdadero o falso): *'Envío'*, *Intercambio'* y *'precio negociable'*.
- **Vendido:** de tipo *Boolean.*
- **Reservado:** de tipo *Boolean.*
- **Favoritos:** de tipo *Array, compuesto por anuncios*.
- **Localización:** está compuesta por 2 campos de tipo *String*: '*latitud'* y '*longitud'*.

- **Número de visitas:** de tipo *Number*, cada vez que se visita un anuncio este número se incrementa en 1 unidad.
- **Categoría:** por defecto hay una serie de categorías definidas que el usuario solo podrá seleccionar. Campo es de tipo String y las categorías son las siguientes:
    - Tecnología e Informática
    - Coches y Motos
    - Deporte y Ocio
    - Muebles, Decoración y Jardín
    - Consolas y Videojuegos
    - Libros, Películas y Música
    - Moda y Accesorios
    - Juguetes, Niños y Bebés
    - Inmobiliaria
    - Electrodomésticos
    - Servicios
    - Otro

La ***base de datos del chat*** está compuesta por los siguientes campos:

- **Nombre:** de tipo *String* y es único.
- **Mensajes:** es un *Array* compuesto por 3 campos:
    - **Identificador:** de tipo *String*.
    - **Mensaje:** de tipo *String*.
    - **Fecha:** de tipo *Date* (fecha).
- **Participantes:** compuesto por 2 campos:
    - **Propietario:** de tipo *String* y *required* (es necesario que este).
    - **Interesado:** de tipo *String* y *required*.
- **Producto:** de tipo *String* y *required*.

La ***base de datos de usuarios*** está compuesta por los siguientes campos:

- **Nombre de usuario:** de tipo *String,* solo en minúsculas, con un mínimo de 4 y un máximo de 24.
- **Email:** de tipo *String* y con un tamaño máximo de 50 caracteres.
- **Contraseña:** de tipo *String.*
- **Imagen:** de tipo *String.*
- **Localización:** está compuesta por 2 campos de tipo *String*: '*latitud'* y '*longitud'*.
- **Código postal:** de tipo *Number*, con un tamaño de 5 caracteres.
- **Fecha:** de tipo *String.*
- **Número de teléfono:** es de tipo *Number*, con un tamaño de 9 caracteres.
- **Tipo de cuenta:** de tipo *String,* solo en minúsculas y cuenta con un campo adicional para determinar si la cuenta esta registrada por 'Facebook', 'Google' o registro 'normal'
- **Rol:** de tipo String, solo en minúsculas, cuenta con un campo para diferenciar si es 'usuario' o 'administrador' aunque por defecto se completa con 'usuario'.

4.2 Front-end

Cuando se hace referencia a front-end se hace referencia a la los elementos de la pagina, los cuales pueden interactuar con el usuario. Desde la pantalla de login, de registro… hasta las barras de carga y los mensajes de alerta que aparecen cada vez que se realiza alguna peticion.

Por este motivo en este apartado se profundizaré en los elementos con los que el usuario puede interactuar en cada pantalla, ya que hay que tener en cuenta que cada formulario y cada campo lleva una validacion para evitar guardar en la base de datos valores no deseados y evitar tambien el hackeo de la pagina, aunque de esta ultima referencia hablaré en mas profundidad en el apartado de back-end.

```
<div class="row row-facebook-google">
    <div class="col-sm-6 col-sm-offset-3">
        <button id="facebook" class="logRRSS logFacebook"
        ng-click="authenticate('facebook')">
        </button>
    </div>
    <div class="col-sm-6 col-sm-offset-3">
        <button id="google" class="logRRSS logGoogle"
        ng-click="authenticate('google')">
        </button>
    </div>
</div>
```

Ilustración 17. Código HTML de front-end.

Ilustración 18. Visualización de código de front-end.

Acciones que los usuarios pueden realizar:

- ***Registrarse*** tanto por Facebook, Google+ o registro manual rellenando un formulario (Anexo 1.2).
- Acceder si están registrados, ***loguearse*** (Anexo 1.1).
- ***Cambiar de idioma*** entre 'español' e 'ingles'.
- Acceder al menú inferior, también llamado ***'footer'***, para cambiar de idioma, acceder a las redes sociales de la empresa o acceder a:
    - Términos de uso.
    - Preguntas frecuentes.
    - Política de privacidad.

Una vez registrados y logueados pueden acceder a la **página principal** para:

- Ver un ***listado de anuncios*** (Anexo 1.3).
- Acceder al ***detalle de un anuncio*** (Anexo 1.5).
    - Cuando se accede a un anuncio de otro usuario se dispondrá de la opción de contactar con el anunciante a través del chat y añadirlo a favoritos. En este caso se denominará como **anuncio ajeno**.
    - En el caso de que sea un **anuncio propio** cuenta con las siguientes opciones:
        - ✓ Reservar.
        - ✓ Editar.
        - ✓ Eliminar.
- Ir al ***perfil de usuario*** (Anexo 1.8).
- Acceder al **chat** (Anexo 1.10) en el cual se pueden diferenciar dos apartados:
    - Un listado de chats con los que se ha conectado anteriormente.
    - Una ventana para chatear.
- Navegar por el **menú superior** y pudiendo acceder a:
    - El **logotipo** redirige a la **página principal**.
    - **Anuncios** redirige a la **página principal**.
    - **Nuevo Anuncio** redirige al formulario para subir un nuevo anuncio (Anexo 1.7).
    - **Chat** redirige a una nueva página con el listado de chats.
    - **Nombre de usuario** y **foto** redirigen al perfil del propio usuario (Anexo 1.8.1).
    - **Salir** cierra la sesión del usuario y redirige a la página de loguin.
- Realizar una **búsqueda**, cuenta con un campo de escritura en la parte superior para realizar una búsqueda, el cual busca en el título del anuncio.

- Realizar ***búsquedas avanzadas*** de anuncios (Anexo 1.4).
    - Por precio, de una cantidad mínima hasta una máxima.
    - Por categoría.
    - Por distancia, desde la ubicación del usuario hasta la que el considere, en kilómetros.
    - Según la posibilidad de que el vendedor envíe el producto, sea intercambiable o si el precio es negociable.
    - Búsqueda de alguno de los tags del anuncio.
    - Ordenarlos de 'mayor a menor' precio o de 'menor a mayor'.

El campo de búsqueda es independiente al de búsqueda avanzada. Se puede realizar una búsqueda global o una búsqueda más detallada, pero nunca las dos a la vez ya que son dos funcionalidades diferentes.

En el ***perfil de usuario*** (Anexo 1.8.1) se puede:

- **Editar los datos personales** dependiendo del registro que se ha realizado, ya que si es por Facebook o Google+ no se podría modificar ni la contraseña ni el correo, en este caso redirige a una página diferente (Anexo 1.9).
- Modificar la ***ubicación*** en el mapa.
- Cambiar la ***foto de perfil***.
- Ver los anuncios **en venta**, los **vendidos** y sus **favoritos**.

Solo se podrán realizar cambios en el perfil de usuario si este corresponde con el usuario logueado, usuario propio, sino solo podrá visualizar el perfil sin poder editar ningún campo.

A la hora de la programación se ha optado por crear un archivo '*app.js*' incluido en la carpeta '*utils*'. En dicho archivo se encuentra una serie de relaciones con un **estado**. Dicho estado tiene es el nombre el cual relaciona una **url** con un **templateUrl/vista**, siendo este último un archivo '*.html*' el cual se representa visualmente.

En la siguiente captura se puede observar las tres líneas más importantes del código incluidas el archivo *'index.html'*:

- ***'ng-include':*** cargar el archivo correspondiente a la barra del menú superior.
- ***'ui-view':*** redirige a la vista correspondiente.
- ***'ng-include':*** cargar el archivo correspondiente al menú inferior o *'footer'*.

```
<div ng-include="'utils/menu.html'"></div>
<div class="main" id="main" ng-style="estilo">
    <div class="container row">
        <div class="col-xs-12">
            <div class="mainContent" ui-view></div>
        </div>
    </div>
</div>
<div id='footerID' ng-include="'utils/footer.html'"></div>
```

Ilustración 19. Archivo 'index.html' de referencia.

A continuación, se observa parte del código del archivo ***'app.js'***. En dicho archivo se detallarán una serie de estados los cuales hacen referencia a una URL y una vista. Al hacer referencia a una vista se dispondrá de la ubicación concreta de dicha vista.

Una vez nombradas las funciones a grandes rasgos se detallarán las diferentes vistas explicando su formación para la adaptación según el tamaño del dispositivo, los formularios de entrada y sus funcionalidades.

```
$stateProvider
    .state(states.home, {
        url: paths.home,
        templateUrl: 'authentification.html'
    })
    .state(states.error, {
        url: paths.error,
        templateUrl: 'utils/error/error404.html'
    })
```

Ilustración 20. Ejemplo del archivo 'app.js'.

4.2.1 Login

Por defecto al acceder a la web se mostrará la vista para poder loguearse (Anexo 1.1). En el caso de no estar registrado se puede acceder por Facebook, Google o cambiar de pestaña para el registro, el cual se explica en el siguiente apartado.

Valores de entrada del formulario de ***loguin****:*

- **Email:** tiene que ser un email con un máximo de 50 caracteres.
- **Contraseña:** tiene que ser un *String* de entre 4 y 16 caracteres.

Todos los campos han de estar rellenados y con los parámetros correctos para que se active el botón de ***'Entrar'*** y así realizar la petición.

Siempre está disponible el menú inferior o ***'footer'*** en el cual se da la opción de cambiar de idioma, acceder a las redes sociales de la empresa o acceder a otras tres vistas, que son:

- Términos de uso.
- Preguntas frecuentes.
- Política de privacidad.

Por defecto la web está en castellano, pero se puede cambiar a ingles simplemente pulsando sobre la bandera inglesa del footer.

4.2.2 Registro

Pantalla a la cual se accede para registrarse en la base de datos. Hay que tener en cuenta que solo se guardarán los datos si los campos están rellenados de forma correcta y el email no está en la base de datos.

*Valores de entrada del formulario de **registro**.*

- **Nombre:** tiene que ser un String de entre 4 y 24 caracteres.
- **Contraseña:** tiene que ser un String de entre 4 y 16 caracteres.
- **Email:** tiene que ser un email con un máximo de 50 caracteres.
- **Número de Teléfono:** tienen que ser números, con un campo cuyo valor está entre 9 y 999999999.

Si algún campo no está con dichos parámetros aparecerá debajo de dicho recuadro un error avisando del error. Si todos los campos están de forma correcta se activará el botón de ***'Regístrate'*** para enviar la petición.

4.2.3 Menú principal

En esta vista se observarán todos los anuncios en venta en diferentes tarjetas. Cada tarjeta cuenta con la imagen, el título y su precio.

Al acceder a esta página hay un ligero cambio en la barra de menú superior y es la activación de las diferentes pestañas por las que se puede navegar. Este cambio se ha producido porque el usuario se ha *logueado* de forma correcta (Anexo 1.3).

- Al pulsar al ***logo*** y a ***'Anuncios'*** redirige a la página principal con la vista de todos los anuncios.
- ***'Subir anuncio'*** redirige a una nueva página (Anexo 1.7).
- ***'Chat'*** redirige a una nueva página, (Anexo 1.10).
- Al pulsar sobre el **nombre de usuario** o **foto de perfil** redirige al perfil de usuario (Anexo 1.8.1).
- ***'Salir'*** cierra la sesión y redirige a la página inicial para loguearte.

No solo en la página principal se puede navegar a diferentes páginas con el menú, también está la opción de ver de forma más detallada un anuncio al pulsar sobre él, también se puede realizar una búsqueda de forma más detallada.

4.2.4 Búsqueda y búsqueda avanzada

Existen dos tipos de búsqueda diferentes:

Al rellenar el *campo de **búsqueda*** y pulsar el botón '*buscar*' se realiza una búsqueda sobre el campo de título de los anuncios.

También está la opción de desplegar un formulario con todos los campos con los que cuenta un anuncio (Anexo 1.4), siendo esto una **búsqueda avanzada**.

Si se rellena algún campo se realizar la búsqueda concatenada de todos los campos rellenados o seleccionados. No es obligatorio rellenar todos los campos, aunque la prioridad de ordenación, en el caso de que no sea por precio, será por fecha de subida. En el anexo 1.4 se puede ver una ilustración con todos los campos de la búsqueda avanzada.

4.2.5 Detalle de un anuncio

En el detalle de un anuncio se pueden observar tres zonas, las cuales se adaptarán según el tamaño de la pantalla del dispositivo en el cual se visualice (Anexo 1.5).

1. La imagen del producto, la cual ajusta su tamaño de forma automática.
2. Un bloque en el cual está la descripción completa del producto.
3. Un bloque con la información del propietario del anuncio.

En el bloque de descripción del anuncio cuenta con los siguientes campos:

1. Título del anuncio.
2. Precio del producto.
3. Número de visitas.
4. Botón para añadir a favoritos.
5. Descripción del anuncio.
6. Tags.
7. Opciones de envío.
8. Mapa de posición.
9. Botones para compartir en redes sociales.

En el detalle de un anuncio hay varias diferencias según la propiedad del anuncio. Existen dos opciones posibles, la opción de que anteriormente haya subido un anuncio, con lo que es un **anuncio propio**, o, se puede acceder a un anuncio de cualquier usuario, un **anuncio ajeno**.

A continuación, se detallarán las diferencias según la propiedad del anuncio.

4.2.5.1 Detalle de un anuncio propio

En el caso de acceder a un anuncio subido anteriormente por el propio usuario, aparecerán los siguientes botones al final del anuncio (Anexo 1.5.1):

- **Reservar**. Si se pulsa en 'Reservar' aparecerá una capa oscura con un título avisando de su reserva. Dicho aviso estará disponible en el menú principal y en el detalle del anuncio.
- **Editar**. Al editarlo redirige a una nueva pestaña con los datos que se han rellenado anteriormente. El formulario es el mismo que se ha usado para subir un anuncio nuevo (Anexo 1.6).
- **Borrar**. Eliminará el anuncio de la base de datos.

También está la diferencia de no aparecer el botón para añadir a **favoritos**, únicamente aparecerá el número de visitas dicho anuncio.

4.2.5.2 Detalle de un anuncio ajeno

En este caso, como se ha comentado anteriormente en el detalle del anuncio estará disponible el botón para añadir a favoritos y el número de visitas, además de toda la información del anuncio (Anexo 1.5.2).

Al ser un anuncio ajeno se puede contactar con el usuario mediante el chat pulsando sobre ***'contactar con el vendedor'***. Si no se ha pulsado sobre este botón no se añadirá a la lista de chats.

4.2.6 Editar anuncio

En esta vista se puede variar cualquier campo del anuncio. Dicho formulario está configurado para que todos los campos necesiten estar completados y de forma correcta con los valores del anuncio. Se pueden eliminar los valores del campo, pero han de ser reemplazados por otros para que el formulario pueda ser enviado para actualizarse en la base de datos.

Si algún dato estuviera con otros valores inapropiados avisa del error debajo de dicho recuadro y se desactiva el botón de **'Guardar'** para no enviarlo a la base de datos.

Para detallar más los datos de entrada del formulario se detallarán en el siguiente apartado, ya que es el mismo formulario de nuevo anuncio.

4.2.7 Nuevo anuncio

En este apartado se va a detallar el formulario de registro de un **Nuevo Anuncio**. En el caso de acceder por primera vez, la página redirige al perfil de usuario para escoger una ubicación en el mapa y así poder asignársela posteriormente a los anuncios de forma automática.

*Valores de entrada del formulario para un **nuevo anuncio**:*

- **Título:** tiene que ser de tipo texto de entre 5 y 50 caracteres.
- **Descripción:** tiene que ser de tipo texto de entre 9 y 160 caracteres.
- **Imagen:** tiene que seleccionarse un archivo de tipo imagen con un tamaño máximo de 20MB.
- **Precio:** tiene que ser un número y como mínimo tiene que tener un valor.
- **Tags:** se ha usado una librería para separarlos. Tiene un límite de 5 tags, cada tag con un tamaño mínimo de 3 y un máximo de 16 caracteres. Estos se separan entre sí mediante la barra espaciadora.
- **Categoría:** se ha usado un desplegable con las clases ya establecidas y únicamente hay que seleccionar la categoría que más se aproxime al producto.
- **Opciones:** en este apartado hay tres casilleros seleccionables para escoger la forma de realizar la transacción. No hay ninguna obligación de seleccionar algún casillero.
    - ✓ **Envío**
    - ✓ **Intercambio**
    - ✓ **Precio Negociable**

Si se han rellenado de forma correcta todos los parámetros no aparecerá ningún error y se activará el botón de ***'subir Producto'***. Una vez subido el producto la web redirige a la página principal para ver toda la lista de anuncios, añadiendo este último a continuación del último.

4.2.8 Perfil de usuario

Dependiendo del usuario se puede acceder a su propio perfil, **usuario propio**, o al perfil de otro usuario, **usuario ajeno**. Con esta diferenciación se pueden llevar a cabo diferentes acciones, las cuales se explican en los siguientes dos apartados.

4.2.8.1 Perfil de usuario propio

Se define como usuario propio cuando en la barra superior del menú aparece el nombre de usuario al igual que en el perfil de usuario. Al haber accedido a su propio perfil cuenta con la administración de sus datos personales y de sus anuncios.

Para llegar a esta página se puede pulsar sobre la imagen o el nombre que aparece en la barra superior del menú.

Al poder editar sus datos personales cuenta con algunos derechos, los cuales son:

Ilustración 21. Perfil de usuario propio.

- Se puede seleccionar sobre el mapa para cambiar la ubicación, moviendo el localizador y posteriormente guardarla usando el botón ***'Guardar mi posición'***, el cual no aparece en el otro caso.
- Debajo de la ubicación aparece un botón para editar los datos personales, dicho apartado se explicará de forma más detallada en el apartado 5.2.9.
- Al ubicar el puntero sobre la imagen aparece un texto avisando de su posible modificación. Únicamente se pueden añadir imágenes.
- Botón ***'Subir anuncio'*** además de estar en el menú superior también aparece la opción de redirigir para añadir un nuevo anuncio.

La diferencia más importante es la aparición de tus anuncios ***'favoritos'*** en el perfil propio, ya que al acceder a un usuario ajeno no aparece esta opción en la parte inferior (Anexo 1.8.1).

4.2.8.2 Perfil de usuario ajeno

Al acceder a otro perfil de usuario no se podrá editar nada, solo se puede ver la información personal, la ubicación y la otra gran diferencia, no se ven los anuncios favoritos del usuario (Anexo 1.8.2).

4.2.9 Editar perfil

Existe un único formulario a la hora de editar el perfil de usuario, el cual se diferencia según la forma de acceder a la página, tanto por Facebook, Google o manual.

Al obtener los datos de Facebook y Google se ha decidido por no almacenar los datos en la base de datos sin su cifrado por lo que la contraseña no se podrá variar al editar el perfil si se accedió con estas cuentas. El email tampoco se podrá variar ya

que al ser el de una cuenta oficial y realizar la petición tiene que mantenerse por si la próxima vez se quiere acceder directamente.

4.2.9.1 Edición de perfil normal

En este punto se detallará el formulario para la edición de los datos personales de un usuario registrado de forma manual, cuyo rol es 'normal'.

*Valores de entrada del formulario para la **edición del perfil**:*

- **Nombre de usuario:** de tipo *texto* de entre 4 y 32 caracteres.
- **Contraseña antigua:** de tipo *contraseña* de entre 4 y 16 caracteres además de coincidir con la registrada en la base de datos.
- **Nueva contraseña:** de tipo *contraseña* de entre 4 y 16 caracteres.
- **Email:** de tipo *email* con un tamaño máximo de 50 caracteres.
- **Fecha de nacimiento:** de tipo *date*, la fecha mínima es 1900-01-01 y el formato den fecha es: aaaa-nmm-dd.
- **Código postal:** de tipo *Number* con un tamaño requerido de 5 números.

En el caso de las contraseñas no variará en la base de datos si la contraseña antigua no coincide con la de la base de datos o no se hubiera escrito nada en el campo contraseña antigua.

En la base de datos se pueden ver todas las contraseñas cifradas, aunque es relativamente sencillo descifrarlas obteniendo el patrón y la clave variable.

4.2.9.2 Edición de perfil, cuenta Facebook o Google.

En la base de datos los administradores únicamente podrán ver el correo sin cifrar, la contraseña se puede ver cifrada obteniendo así mayor privacidad.

En el caso de acceder desde Facebook o Google no se admite el cambio de contraseña ni email por venir estos datos desde cuentas oficiales, por lo que los campos de edición serán los siguientes:

*Valores de entrada del formulario para la **edición del perfil**:*

- **Nombre de usuario:** de tipo *texto* de entre 4 y 32 caracteres.
- **Fecha de nacimiento:** de tipo *date* con fecha mínima de 1900-01-01 y el formato de fecha es: aaaa-mm-dd.
- **Código postal:** de tipo *Number* con un tamaño requerido de 5 números.

4.2.10 Chat

Para acceder a dicha funcionalidad se puede realizar desde dos lugares:

- Al pulsar mediante la pestaña de ***'chat'*** del menú superior, dicha pestaña redirige al listado de los chats.
- Al acceder al chat mediante el botón de ***'contactar con el vendedor'*** en un anuncio ajeno. En esta opción añade una nueva conversación al listado del chat.

Para implementar dicha funcionalidad se dispondrá de un menú con el listado de productos contactados y un recuadro con el listado de los mensajes.

Dicha funcionalidad realiza el envío de mensajes entre dos usuarios relacionados con el producto que se desea. Se dispondrá la opción de conversar con el mismo usuario tantas veces como anuncios tenga sin vender.

Como se observa en la siguiente captura en el listado del chat se encuentra la imagen del producto, el título y el nombre del usuario propietario. Además, en el apartado del chat se ha utilizado una biblioteca, la cual esta explicada en el apartado de bibliotecas, para la inserción de la fecha en cada mensaje enviado.

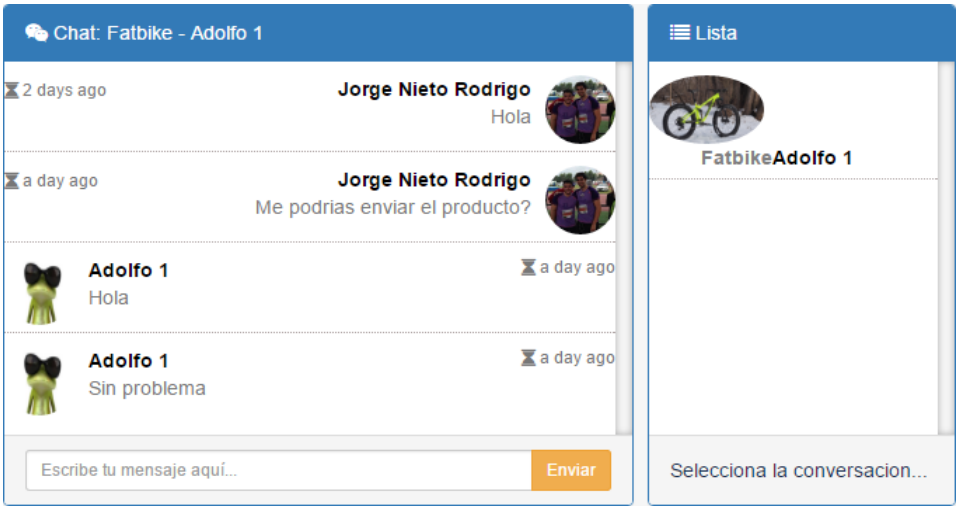

Ilustración 22. Vista de un chat.

4.3 Back-end

En este apartado se explicará de forma resumida el funcionamiento interno de la página, la realización de peticiones, sus medidas de seguridad y las bibliotecas de ayuda.

En la siguiente captura se puede observar un pequeño ejemplo de los métodos usados para recibir la información desde el usuario (front-end), su procesamiento y el envío de información al navegador del usuario.

```
$scope.getClassForItem = function(item, type) {
};
$scope.$on("$stateChangeSuccess", function(evt, currentRoute) {
    $scope.selectedItem = $state.current.name || '';
});
//CAPTURA DE EVENTO
$scope.$on("userEnterRequest", function(evt) {
    $scope.getUser();
});
$scope.$on("imagenProfileChange", function(evt) {
    $scope.getUser();
});
$scope.getUser = function() {
};
```

Ilustración 23. Ejemplo de código de back-end.

4.3.1 Peticiones http

Desde 1990, el protocolo HTTP (Protocolo de transferencia de hipertexto) es el protocolo más utilizado en Internet. La versión 0.9 sólo tenía la finalidad de transferir los datos a través de Internet (en particular páginas Web escritas en HTML). La versión 1.0 del protocolo (la más utilizada) permite la transferencia de mensajes con encabezados que describen el contenido de los mensajes.

El propósito del protocolo HTTP es permitir la transferencia de archivos (principalmente, en formato HTML) entre un navegador (el cliente) y un servidor web localizado mediante una cadena de caracteres denominada dirección URL.

Funcionamiento paso a paso de las peticiones:

1. El cliente (front-end) crea y envía una petición.

La petición es un mensaje de texto creado por un cliente (por ejemplo, un navegador, una aplicación para dispositivos móviles, etc.) en un formato especial conocido como HTTP. El cliente forma la petición con los datos correspondientes y envía la petición.

2. El servidor captura el evento y lo procesa.

3. El servidor devuelve una respuesta.

La respuesta HTTP contiene el recurso solicitado, así como otra información acerca de la respuesta. En el ***'Status Code'*** devuelve el estado de la petición HTTP (200 OK en este caso) de la respuesta, también ***'Request Method'*** es importante ya que diferencia la acción a desarrollar en el servidor.

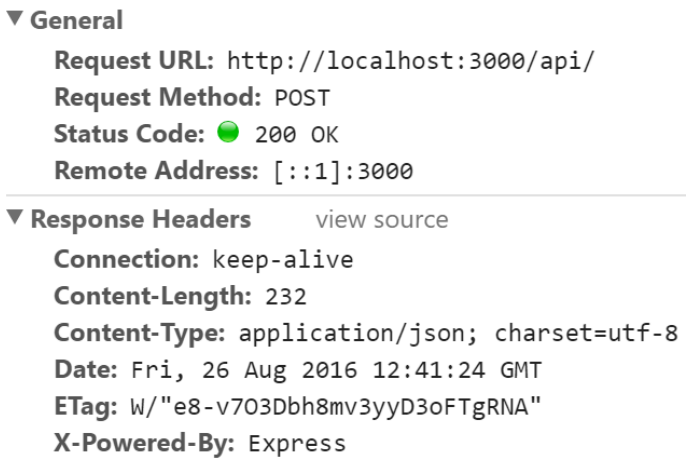

Ilustración 24. Resolución de una petición http.

En el caso de desear más información detallada del proceso seguido por una petición se puede observar la ilustración 25 en la cual detalla los pasos.

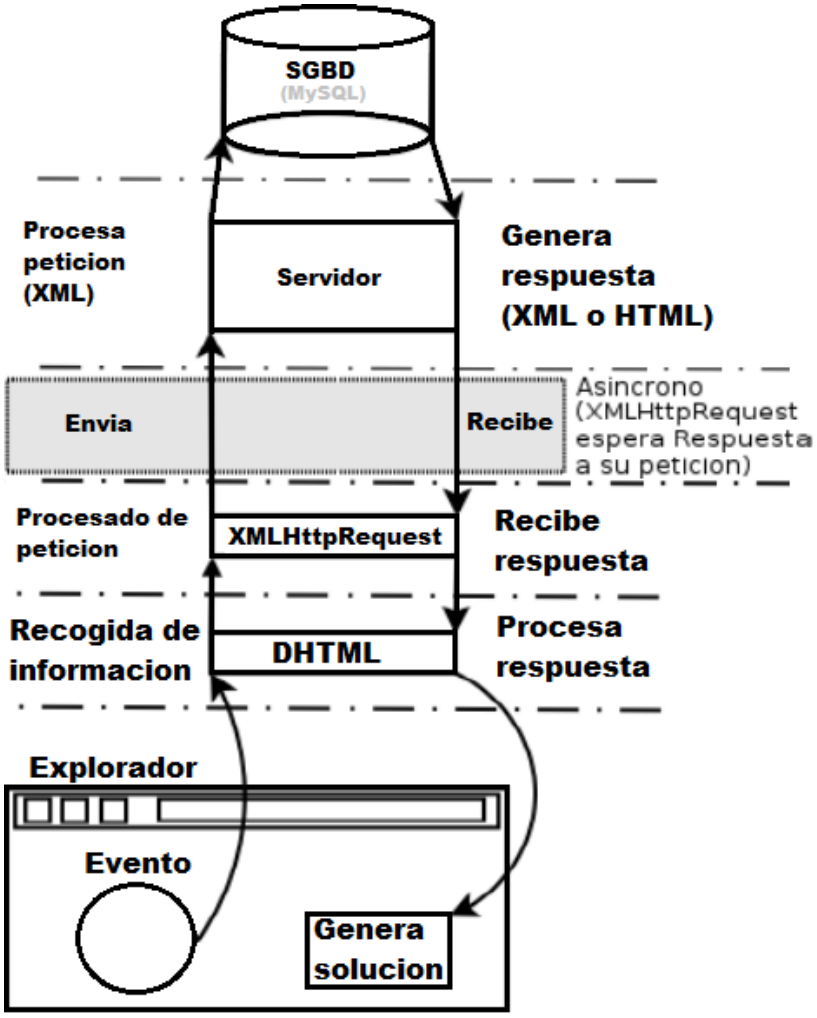

Ilustración 25. Proceso de una petición http.

Métodos de peticiones:

- **GET**: Usado para la petición de datos.
- **POST**: Envío de datos al programa ubicado en la URL definida.
- **PUT**: Envío de datos a la URL especificada.
- **DELETE**: Borrado de recursos ubicados en la URL especificada.

A continuación, se detallará la forma de realizar peticiones, especificando los datos que han de llevar las peticiones.

```
router.get('/', function(req, res) {
});     2  1            3   4

router.post('/announ', function(req, res) {
});
```

Ilustración 26. Ejemplo de una petición.

Detalle de la ilustración 25:

1. Dirección de la petición.
2. Método de la petición.
3. Datos de entrada.
4. Datos de salida/respuesta del servidor.

Códigos de respuesta:

Los códigos de estatus están especificados por el RFC 2616, y algunos fragmentos en los estándares RFC 2518, RFC 2817, RFC 2295, RFC 2774 y RFC 4918; otros no están estandarizados, pero son comúnmente utilizados.

Ejemplo de códigos de respuesta:

1XX: Respuestas informativas.

2XX: Peticiones correctas.

3XX: Redirecciones.

4XX: Errores del cliente.

5XX: Errores del servidor.

```
app.use(function(req, res, next) {
    var err = new Error('Not Found');
    err.status = 404;
    next(err);
});
```

Ilustración 27. Código de respuesta 404.

Además de los códigos estandarizados se pueden crear códigos de respuesta propios, estos siempre han de estar comentados en el código y detallados en los manuales de la web para facilitar su comprensión a los demás desarrolladores.

```
if (err) {
    return res
        .status(500)
        .send({ result: false, err: err });
}
```

Ilustración 28. Devolución de un error 500 en la petición.

4.3.2 Medidas de seguridad

Es necesario proteger la información enviada desde el cliente hasta el servidor. Por este motivo se ha optado por la utilización de un *'token'* o código de seguridad para las peticiones. Dicho 'token' acompaña siempre a las peticiones para corroborar que el usuario está logueado, registrado en la base de datos y no es un intento de intrusión.

Para la creación de dicho 'token' se ha utilizado la biblioteca *'satellizer'* la cual recibe las credenciales de autenticación básicas de la solicitud dada. La autorización está en la cabecera, se analiza y si la cabecera no es válida o esta indefinida devuelve un error, sino devuelve un objeto con dos campos: nombre/email y contraseña.

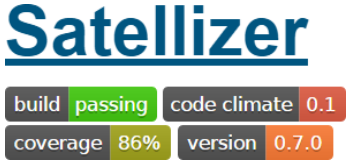

Ilustración 29. Biblioteca de seguridad básica.

La biblioteca anterior ofrece cierta seguridad, pero al ser una biblioteca pública y gratuita con lo que es fácil descifrar los datos si se conoce el algoritmo de creación del *'token'*. Por este motivo se ha insertado una variable secreta, en un punto determinado del token, y un tiempo máximo con lo que, si está más de 2 horas en la web, el 'token' expirará y se cerrará la sesión automáticamente.

En cada petición se comprueba que el 'token' esté (sea correcto y no haya expirado) para seguir en la web.

En caso de usar la web de forma profesional y lanzarse al mercado habría que implementar el último proceso de seguridad. Al realizar el primer envío de información, tanto de loguin como de registro, se realiza una petición http la cual lleva todos los datos sin procesar por este motivo sería necesaria la implementación de un envío de datos mediante HTTPS.

El sistema HTTPS utiliza un cifrado basado en SSL/TLS para crear un canal cifrado (cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente) más apropiado para el tráfico de información sensible que el protocolo HTTP. De este modo se consigue que la información sensible (usuario y contraseñas normalmente) no puedan ser usadas por otra persona que haya conseguido interceptar la transferencia de datos de la conexión, ya que lo único que obtendrá será un flujo de datos cifrados que le resultará imposible de descifrar.

No se ha implementado con HTTPS por el precio de la licencia ya que está web aún no ha sido comercializada.

4.3.3 Bibliotecas de ayuda

Para la implementación de muchas funciones de esta aplicación es necesario el soporte de aplicaciones externas que incorporan funciones que resuelven problemas concretos tanto del font-end como del back-end.

Dichas bibliotecas se alojan en la carpeta ***'bower_components'*** y a continuación se describen las utilizadas para el desarrollo de este proyecto:

-angular: este módulo se incluye por defecto y contiene los componentes básicos de AngularJS.

-angular-bootstrap-switch: se usa para la verificación de formularios.

-angular-socket-io: este módulo es usado por la página web para que el ciclo de gestión con angular, es capaz de crear circuitos virtuales para las peticiones entre el front-end y el back-end.

-angular-toastr: este módulo es usado para la creación de avisos de mensajes, tanto de error, de aviso o mensajes correctos (200 OK).

-angular-translate y **angular-translate-loader-static-files:** estos módulos son usados para la traducción de la web mediante la carga asíncrona de datos mediante archivos '. json'.

-angular-ui-router: este módulo es usado para actualizar la URL del navegador. Proporciona una máquina de estados para la gestión de las transiciones entre los estados de la aplicación y las URL.

-angularjs-slider: este módulo es usado para la inserción de barras deslizantes en la aplicación.

-bootstrap: este módulo es usado para generar los archivos CSS, JavaScript y las fuentes del proyecto.

-bootstrap-switch: este módulo es usado para la inserción de casilleros de verificación y diferentes tipos de botones especiales ya preconfigurados.

-font-awesome: este módulo es usado para la inserción de diferentes iconos y los diferentes estilos de letras.

-ihover: este módulo es usado para la activación de acciones cuando el ratón pasa por encima del objeto configurado. Se puede usar para avisar de mensajes cuando el ratón pasa por encima de un objeto.

-intro: este módulo es usado para la función de 'visita guiada' de la página, la cual realiza una visita por la web indicando las funcionalidades y explicándola, paso a paso, de forma más detallada.

-moment: este módulo es usado para la inserción de fecha y hora. En el proyecto se ha usado para la inserción de dichos parámetros en el chat, para así poder saber cuándo se envió dicho mensaje.

-ng-file-upload: este módulo es usado para cargar imágenes en la web. Esta biblioteca es capaz de editar una imagen, guardarla en el servidor y posteriormente usarla en la web.

-ng-tags-input: este módulo es usado para la edición y separación de los diferentes Tags/etiquetas. Es configurable el método de separación, el estilo de los Tags, el fondo de color, etc.

-ngmap: este módulo es usado para la inserción de los mapas y su utilización de forma rápida y sencilla. Para su uso es necesario tener una clave de desarrollador, la cual se obtiene de forma gratuita.

-satellizer: es sencillo de usar, de extremo a extremo. Es un módulo de autenticación basado en un 'token' para AngularJS con OAuth con soporte integrado para Google, Facebook, LinkedIn, Twitter, Instagram, GitHub, Bitbucket, Yahoo, Twitch, Microsoft (Windows Live) proveedores, así como de correo electrónico y contraseña de inicio de sesión. Sin embargo, no está limitado a las opciones de inicio de sesión por encima, de esto se puede añadir cualquier OAuth 1.0 o OAuth 2.0 de proveedor por pasar información específica del proveedor de la aplicación de configuración de bloques.

Capítulo 5.

Evaluación de la aplicación

En este apartado se explicará de forma breve la forma de trabajar con '*protractor*' y la realización de los test.

A la hora de definir un sprint se ha de tener en cuenta el tiempo de dicho sprint y las funcionalidades que se van a implementar. Una vez se han terminado dichas funcionalidades será necesaria la implementación de dichos test.

Si la funcionalidad es demasiado difícil de automatizar el test o es demasiado corta como para realizar un test será necesario realizar el test de forma manual. Dicho motivo no implica que el test no se tenga que probar ni añadir en el formulario de los test.

Para cada funcionalidad no se ha de implementar un único test, es necesario realizar todos los test para corroborar que realiza todas las acciones, tanto las que son correctas como la comprobación de los mensajes de error y alerta.

En el anexo 2.2 se hace referencia al archivo de Excel como ejemplo de archivo para el testeo de una aplicación ya que es imprescindible realizar un documento de testeo además de la automatización de estos.

Se hará necesario la declaración de identificadores únicos para cada campo o botón que se tenga que usar, ya que en protractor se hará necesaria la utilización de valores de entrada y salida como acciones. En la siguiente captura se observa los diferentes elementos:

1. Campo de texto. Escribe en dicho campo el valor asignado.
2. Comparación, de URL, la actual con la definida.
3. Acción de pulsar, en esta captura pulsar sobre el botón de 'loguin'.

```
function logIn() {
  1 email.sendKeys('gominola@hotmail.com');
    passLog.sendKeys('jesus');
  2 expect(browser.getCurrentUrl()).toEqual('http://localhost:3000/');
  3 element(by.id('entrar')).click();
}
```

Ilustración 30. Funcionalidades en un test.

Al tener diferentes sprint y diferentes funcionalidades se ha tenido que definir tres archivos con los test correspondientes, además de un archivo de configuración para realizar de forma secuencial los diferentes test:

1. Archivo de configuración de los test, conf.js (Capítulo 6.1).
2. Test de acceso, spec.js (Capítulo 6.2).
3. Test de anuncios, detailProfile.js (Capítulo 6.3)
4. Test del perfil de usuario, profile.js (Capítulo 6.4).

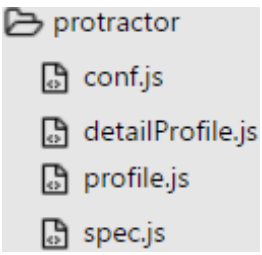

Ilustración 31. Ubicación de los test.

5.1 Configuración de los test

Archivo **conf.js**, cuenta con la configuración general para los test. Dicho archivo será usado para ejecutar de forma automática todos los archivos con los test organizados según las funcionalidades. En la siguiente captura se puede observar:

- Framework usado.
- Ruta para acceder a los test.
- Archivos correspondientes a los test, se realizan de forma lineal según estén escritos.
    - **spec.js**
    - **detailProfile.js**
    - **profile.js**.
- En *'multiCapabilities'* se añadirán los diferentes buscadores en los cual se van a realizar los test automáticos. En este proyecto solo se han realizado las pruebas en *'Chrome'* por ser el navegador más utilizado.

```
exports.config = {
    framework: 'jasmine',
    seleniumAddress: 'http://localhost:4444/wd/hub',
    suites: {
        basicLoginTest: 'spec.js',
        detailAndEdit: 'detailProfile.js',
        profile: 'profile.js'
    },
    multiCapabilities: [{
        browserName: 'chrome'
    }]
}
```

Ilustración 32. Archivo de configuración de los test.

5.2 Test de acceso

Archivo **spec.js,** cuenta con los test para comprobar el correcto funcionamiento de la funcionalidad de '*loguin'* y '*registro'*. Además, se han de realizar las pruebas correspondientes para intentar acceder al menú principal sin loguarse usando las URL.

Dicho archivo cuenta con una serie de variables declaradas al inicio las cuales se obtendrán en los campos del front-end y se reutilizarán en los diferentes test de dicho archivo.

- En **beforeEach** se realizará la operación de abrir el navegador y maximizarlo.
- Las **funciones** que se encuentran a continuación están diseñadas para realizar el acceso o registro.
- Para finalizar este archivo se observa que hay una serie de **iteraciones,** las cuales prueban toda la parte de acceder a la aplicación.

Test realizados en ‘spec.js’:

- Registro correcto.
- Intentar acceder con un email incorrecto.
- Loguearse de forma correcta y salir.
- Intento de acceder con una URL correcta sin estar logueado.
- Acceder con Google.
- Intento de acceder con una URL incorrecta estando logueado.

```
describe('Protractor Testing RoyalePOP', function() {
    var username = element(by.id('usr'));
    var password = element(by.id('pw'));
    var email = element(by.id('email'));
    var number = element(by.id('tel'));
    var userLog = element(by.id('username'));
    var passLog = element(by.id('pass'));

    beforeEach(function() {…
    });

    function register() {…
    function logIn() {…
    function footer() {…
    function google() {…
    function facebook() {…

    it('test register', function() {…
    it('test incorrect login', function() {…
    it('test correct login', function() {…
    it('test footer', function() {…
    it('test google', function() {…
    it('testing the access to an unexisted url', function() {…
});
```

Ilustración 33. Archivo de test 'spec.js'.

5.3 Test de anuncios

Archivo **detailProfile.js,** cuenta con los test para comprobar el correcto funcionamiento de los anuncios y una mínima parte del perfil de usuario.

Dicho archivo cuenta con todos los test de los anuncios, el detalle de los anuncios y las posibles verificaciones según la propiedad del anuncio.

Test realizados en ‘detailProfile.js’:

- Login correcto.
- Editar perfil de usuario (cambiar nombre de usuario y solo contraseña antigua).
- Editar perfil de usuario (cambiar nombre de usuario y contraseñas, cambio OK).
- Editar perfil de usuario (cambiar nombre de usuario incorrecto y contraseñas, cambio OK).

- Subir anuncio (Añadir título, descripción, precio e imagen) y aviso de error.
- Acceder a la edición de un anuncio, cambiar el título por uno menor al valor del formulario y guardarlo (no se podrá realizar).
- Añadir anuncio a 'reservado' y a 'favoritos'.
- Reservar un anuncio y corroborar si la variable está en dicho estado.
- Añadir anuncio a 'favoritos' y eliminarlo de 'favoritos' posteriormente.
- Eliminar un anuncio propio.
- Cambiar de idioma.
- Búsqueda avanzada la cual no encuentra nada.
- Búsqueda avanzada correcta.

```
describe('Protractor Testing RoyalePOP', function() {
    var path = require('path');
    var email = element(by.id('email'));
    var password = element(by.id('pass'));

    beforeEach(function() {

    function logIn() {
    function uploadFile() {

    it('test login', function() {
    it('edit profile oldpw without newpw', function() {
    it('edit profile with bad oldpw', function() {
    it('edit profile with oldpw and newpw correctly', function() {
    it('upload an advert', function() {
    it('access to my detail announcement and edit', function() {
    it('access to profile detail click on favs, vent and sold', function() {
    it('access to detail announcement (mine) and click reserve', function() {
    it('access to detail announcement (not mine) of and click fav', function() {
    it('access to detail announcement (mine) and delete', function() {
    it('language change', function() {
    it('advanced search fail', function() {
    it('advanced search success', function() {
});
```

Ilustración 34. Archivo de test 'detailProfile.js'.

5.4 Test del perfil de usuario

Archivo **profile.js,** cuenta con los test para comprobar el correcto funcionamiento del perfil de usuario. Además, se realizan las pruebas correspondientes para intentar acceder sin registrarse usando las URL.

Dicho archivo cuenta con una serie de variables declaradas al inicio las cuales se obtendrán en los campos del front-end y se reutilizarán en los diferentes test de dicho archivo.

Funciones reutilizables en los test:

- Registrar correctamente a un usuario.
- Comparación de URLs al acceder al perfil de usuario propio.
- Cambiar contraseña de forma correcta y corroborar las URL en cada paso OK. Redirige a la página correcta.

- Cambiar contraseña de forma correcta y corroborar las URL en cada paso (KO). Esto ocurre en el caso de que no redirija a la página asignada.
- Intento de acceder al menú principal sin loguearse.
- Comprobación de redirección si se realizase un intento de acceso al perfil de usuario sin loguearse.
- Comprobación de redirección si se realizase un intento de edición de un perfil de usuario sin loguearse.
- Comprobación de redirección si se realizase un intento de acceso al menú principal sin loguearse.
- Comprobación de redirección si se realizase un intento de acceso al formulario para crear un nuevo anuncio sin loguearse.

Test realizados en 'profile.js':

- Loguearse y corroborar si dicha URL es la correspondiente.
- Acceder al perfil se un usuario y corroborar si dicha URL es la correspondiente. Comprobándose con la URL correcta.
- Acceder al perfil se un usuario y corroborar si dicha URL es la correspondiente. Comprobándose con una URL incorrecta.
- Loguarse, corroborar que dicho usuario cierra la sesión y redirige a la URL correspondiente. Dicha URL será la esperada.
- Comprobación de redirección si se realizase un intento de acceso al perfil de usuario sin loguearse mediante la utilización de URL.
- Comprobación de redirección si se realizase un intento de edición de un perfil de usuario sin loguearse mediante la utilización de URL.
- Comprobación de redirección si se realizase un intento de acceso al menú principal sin loguearse mediante la utilización de URL.
- Comprobación de redirección si se realizase un intento de acceso al formulario para crear un nuevo anuncio sin loguearse mediante la utilización de URL.

```
describe('Protractor Testing RoyalePOP', function() {
    var userLog = element(by.id('username'));
    var passLog = element(by.id('password'));
    var newPass = element(by.id(''));
    var oldPass = element(by.id(''));
    browser.get('http://localhost:3000/');
    beforeEach(function() {…
    function register() {…
    function viewProfile() {…
    function editProfileOk() {…
    function editProfileKo() {…
    function UrlLogueado() {…
    function UrlProfile() {…
    function UrlEditProfile() {…
    function UrlAnouncments() {…
    function UrlNewAnouncment() {…

    it('test: register', function() {…
    it('test: view profile', function() {…
    it('test: Change password OK', function() {…
    it('test: Change password KO', function() {…
    it('test: URL logeuado Ok', function() {…
    it('test: URL SIN loguarse PROFILE', function() {…
    it('test: URL SIN loguarse EDIT PROFILE', function() {…
    it('test: URL SIN loguarse ANOUNCMENTS', function() {…
    it('test: URL SIN loguarse NEW ANOUNCMENT', function() {…
});
```

Ilustración 35. Archivo de test 'profile.js'.

Capítulo 6.

Conclusiones y futuras líneas de trabajo

Este proyecto ha sido desarrollado como primera versión para realizar una demostración a una empresa. Al ser una primera versión, se concluirá con la llegada a dicho objetivo, aunque quede abierto para futuras ampliaciones y mejoras como:

- Reestructuración de la web.
- Nuevo diseño/estilo.
- Rediseño en los mensajes de alerta y creación de nuevos mensajes de aviso.
- Inclusión de menús de ayuda en los margenes del menú principal.
- Puntuación de usuarios al vender sus productos.
- Mejoras en el chat, tanto de diseño como de inclusión en el listado de chats.
- Mejoras de seguridad, tanto para acceder como para el envío de información confidencial.
- Implementación de Google analitycs para comprobar el correcto funcionamiento de la web y su interacción con el usuario.

Aunque se han quedado muchas opciones por añadir a la web hay que tener en cuenta los objetivos fijados con anterioridad. Esta web seguirá desarrollándose una vez concluida dicho trabajo y podrá verse concluida en pocos meses.

Capítulo 7.

Bibliografía

- Curso ***'web Development Bootcamp'***, cuyas transparencias fueron facilitadas por los responsables de este. Estas transparencias no se pueden adjuntar en forma de anexo por ser propiedad de la empresa ***'KeppCoding'***.
- **Protractor**, consultado el 5 de mayo de 2016 http://www.protractortest.org/#/

Bibliotecas de ayuda:

Las bibliotecas de ayuda nombradas con anterioridad se han instalado mediante la utilización de bower con los siguientes comandos:

```
$ bower install (nombre de biblioteca)

$ npm install (nombre de biblioteca)
```

Consultadas en el mes de marzo de 2016:

- **angular:** https://angularjs.org/
- **angular-bootstrap-switch:** https://github.com/frapontillo/angular-bootstrap-switch
- **angular-socket-io:** https://github.com/btford/angular-socket-io
- **angular-toastr:** https://github.com/Foxandxss/angular-toastr
- **angular-translate:** https://angular-translate.github.io/
- **angular-translate-loader-static-files:**

https://www.npmjs.com/package/angular-translate-loader-static-files

- **angular-ui-router:** https://angular-ui.github.io/ui-router/site/#/api/ui.router
- **angularjs-slider:** http://angular-slider.github.io/angularjs-slider/
- **bootstrap:** http://getbootstrap.com/
- **bootstrap-switch:** http://www.bootstrap-switch.org/
- **font-awesome:** http://fontawesome.io/
- **ihover:** http://gudh.github.io/ihover/dist/
- **intro:** http://introjs.com/
- **moment:** http://momentjs.com/
- **ng-file-upload:** https://github.com/danialfarid/ng-file-upload
- **ng-tags-input:** http://mbenford.github.io/ngTagsInput/

- **ngmap:** https://ngmap.github.io/
- **satellizer:** https://github.com/sahat/satellizer