UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

APLICACIÓN DE ASISTENCIA EN COMPRAS

MEMORIA PRESENTADA POR:

Marc Semper Lloret

GRADO DE INGENIERÍA INFORMÁTICA

Convocatoria de defensa: *[indicar mes y año de presentación]*

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Título del trabajo:	APLICACIÓN DE ASISTENCIA EN COMPRAS
Nombre del autor:	Marc Semper Lloret
Nombre del tutor:	Javier Esparza Peidro
Fecha de entrega:	14/02/2019
Titulación:	Grado de Ingeniería Informática
Área del trabajo:	Programación
Idioma del trabajo:	Castellano
Palabras clave:	MongoDB, Express, Angular, NodeJS

Resumen

En la actualidad hay infinidad de productos y tiendas en el mercado, esto nos puede generar dudas sobre que comprar. Además, la tecnología y los mercados evolucionan muy rápido, constantemente salen a la venta nuevos productos que nos pueden resultar útiles y ni sabemos que existen.

Por lo tanto, el principal objetivo de este proyecto consiste en guiar a los usuarios en sus compras. Para ello se ha desarrollado una aplicación informática.

Como segundo objetivo, se ha tratado de aumentar los conocimientos sobre nuevas tecnologías que ofrezcan proyección en el mercado laboral al alumno.

Para desarrollar la aplicación se ha ido pasando por diferentes etapas, primero se ha llevado a cabo un análisis de requisitos, seguidamente se ha diseñado e implementado, y para finalizar se han realizado algunas pruebas.

El desarrollo se ha llevado a cabo usado **MongoDB** como base de datos, **NodeJS** con **Express** en la parte del servidor y **Angular** 6 en la parte del cliente.

Resum

En l'actualitat hi ha infinitat de productes i tendes en el mercat, açò moltes vegades ens crea dubtes sobre que comprar.

La tecnologia i els mercats evolucionen molt de pressa, constantment ixen a la venda productes que ens resultarien útils i ni sabem que existeixen.

Per tant, el principal objectiu d'aquest projecte consisteix en guiar els usuaris en les seues compres. Per això, s'ha desenrotllat una aplicació informàtica.

Com a segon objectiu, s'ha tractat d'augmentar els coneixements sobre tecnologies útils i relativament noves que oferisquen projecció en el mercat laboral a l'alumne.

Per a desenrotllar l'aplicació s'ha anat passant per diferents etapes, primer s'ha dut a terme una anàlisi de requisits, a continuació s'ha dissenyat e implementat, i per a finalitzar s'han realitzat algunes proves.

El desenrotllament s'ha dut a terme gastant **MongoDB** com a base de dades, **NodeJS** amb **Express** en la part del servidor i **Angular** 6 en la part del client.

Abstract

Currently there are many products and stores in the market. Technology and markets evolve very fast, they constantly sell products that would be useful to us and we do not even know they exist.

Therefore, the main objective of this project is to guide users in their purchases. For this, a computer application has been developed.

As a second objective, it has been tried to increase the knowledge on useful and relatively new technologies that offer projection in the labor market to the student.

In order to develop first an analysis of requirements has been carried out, then it has been designed and implemented, and finally some tests have been carried out

The development has been carried out using **MongoDB** as a database, **NodeJS** with **Express** in the server part and **Angular** 6 in the client part.

Índice de contenido

Contenido

1. Introducción

En esta parte se introducirá el proyecto, para ello se explicará a rasgos generales.

El objetivo de este proyecto consiste en guiar a los usuarios en sus compras. Para ello se ha realizado una aplicación con dicho fin.

Aunque hay muchas aplicaciones similares ninguna tiene nuestro enfoque. Por ello, es una buena oportunidad para aprovechar esta falta de competencia. A continuación, entraremos en más detalle.

1.1 Motivación

Las funcionalidades de recomendar productos no son para nada revolucionarias, muchas tiendas online recomiendan productos y permiten a sus usuarios hacer valoraciones.

En una tienda, solo se valoran los productos que dicha tienda vende. En cambio, la aplicación realizada debe permitir a los usuarios poder valorar y consultar sobre todos los productos de las tiendas más populares de internet.

Además de esto, a las tiendas les interesa que sus productos tengan buena imagen, por esta razón, se preocupan de tener buenas valoraciones y esto puede hacer que las valoraciones no sean suficientemente imparciales. Nuestra aplicación será diferente, ya que no se vende ningún producto, por lo tanto, las criticas serán más imparciales.

Para que esto quede más claro, pondremos como analogía las aplicaciones de críticas de cine y las plataformas que ofrecen contenido cinematográfico. Como, por ejemplo, Filmaffinity y Neflix. Cuando un usuario quiere ver una valoración de una película o serie no suele recurrir a Neflix, sino a un portal dedicado a críticas de cine (Filmaffinity), ya que las valoraciones suelen ser más fiables. Además, hay contenido que no está disponible en Neflix, en cambio en una aplicación de críticas de cine están todas las películas de todas las plataformas de contenido cinematográfico.

Por lo tanto, nuestra aplicación tendrá una gran diversidad de productos con críticas imparciales (no hay ventas de por medio). Lo que queremos hacer es como un Filmaffinity pero de productos.

1.2 Estudio del mercado

En estos apartados veremos una aplicación similar a la que queremos implementar y marcaremos alguna diferencia.

Como se ha dicho anteriormente, Filmaffinity es una aplicación que se parecerá mucho a la nuestra, tanto en misión y visión como en formato.

Dicha aplicación es principalmente una página de votación y recomendación de

películas y series, un diario personal de lo que te gusta y lo que quieres ver, además de una red social de cine.

La aplicación que se desea crear es algo similar, si en el párrafo anterior cambiamos películas y series por productos encajaría perfectamente con la definición de la aplicación que se pretende crear.

Por lo tanto, Filmaffinity la podemos usar como un modelo a seguir, ya que debemos crear algo similar.

Pero Fimaffinity no es nuestra competencia, en cambio aplicaciones similares a Amazon sí lo podrían ser, Amazon hace prácticamente lo mismo que nosotros, la única diferencia entre este tipo de negocio y nosotros es el fin, mientras ellos son intermediarios con la finalidad de que se vendan productos, nosotros seremos solamente una página de opinión.

Nos diferenciaremos teniendo un mayor número de productos con mayor imparcialidad.

1.3 Objetivos

Uno de los objetivos de este proyecto consiste en guiar a los usuarios en sus compras.

Pero con este proyecto no sólo se pretende esto, sino mejorar las habilidades del alumno en el desarrollo de aplicaciones de software. Para lograr esto se han creado servicios y clientes de software simulando una aplicación real con cierta complejidad.

Además, se ha realizado todo el proceso de desarrollo de software sin ayuda, desde tener la idea hasta crear un producto en una versión inicial y operativa.

Hay que aclarar que no se llevarán a cabo todas las partes, recopilar todos los productos de internet es un trabajo muy grande para realizarlo en su plenitud y va más allá de este proyecto. Como se ha dicho anteriormente se implementará una versión inicial.

1.4 Estructura del resto de la memoria

Una vez introducido el proyecto pasaremos a entrar en más detalle sobre cómo resolveremos los problemas.

En primer lugar, analizaremos el problema y mencionaremos los requisitos previos necesarios para crear una aplicación que pueda cumplir las funcionalidades deseadas. En segundo lugar, describiremos el proceso de implementación de nuestra aplicación informática. En tercer lugar, mostraremos el resultado obtenido. Y para finalizar, llegaremos a una conclusión acerca de todo lo dicho anteriormente.

2. Análisis del problema

En este apartado presentaremos las funcionalidades de la aplicación, sin entrar en detalle sobre cómo resolverlas.

Para ello especificaremos los requisitos, estos describen que ofrece el sistema. Hay dos tipos de requisitos, los funcionales que especifican el funcionamiento del sistema y los no funcionales que marcan un número de características, como el rendimiento requerido por el sistema, como seguro debe ser … Es decir, mientras los requisitos funcionales definen qué debe hacer un sistema, los requisitos no funcionales definen cómo debe ser el sistema.

A continuación, se especificarán los requisitos en lenguaje natural.

2.1 Requisitos funcionales

1. Los usuarios deben poder registrarse en la aplicación mediante un formulario para poder ser identificados, como mínimo se le pedirá su email y nombre.
2. Ningún usuario que se haya registrado correctamente podrá acceder a la aplicación.
3. Para acceder a la aplicación los usuarios deberán autentificarse, para ello deberán rellenar un formulario donde se validarán sus datos.
4. Una vez un usuario se ha autentificado correctamente tendrá acceso al resto de la aplicación.
5. La página principal mostrará productos clasificados por indicadores útiles.
6. Los indicadores como mínimo deben describir productos que destaquen por tener mejores valoraciones, un mayor número de visitas y han sido añadidos recientemente.
7. No deberán haber más de 2 niveles de profundidad. Es decir, se debe poder acceder a todas las partes de la aplicación, mediante dos enlaces como mucho, indiferentemente del sitio en que te encuentres.
8. Se podrá buscar un producto específico.
9. Los productos se mostrarán por categorías.
10. Todos los usuarios deben poder modificar sus propios datos.
11. Todo producto debe tener un título, imagen, descripción y una valoración de los clientes. Además, deberán aparecer las críticas de todos los usuarios.
12. Los usuarios deben poder valorar los productos y escribir su opinión sobre estos.
13. Los usuarios podrán crear listas de productos.
14. Tanto las listas de productos como las valoraciones deben poder ser identificadas fácilmente, y en todo momento debe ser posible modificarlas o eliminarlas.
15. Se deben mostrar gráficos con los gustos de los usuarios, como mínimo deberán aparecer los votos y notas por categoría.
16. Los usuarios tendrán diferentes niveles de privilegios, por una parte, estarán los usuarios normales y por otra los administradores.
17. Un usuario con suficientes privilegios para ser administrador podrá añadir, actualizar, eliminar y ver cualquier producto, categoría u otro usuario registrado en la aplicación, de este modo debe poder gestionar toda la aplicación.
18. El usuario debe poder cerrar la sesión cuando lo desee.

2.2 Requisitos no funcionales

1. La aplicación deberá ser segura, es decir, deberá haber medidas para que no se pueda robar, manipular o usar indebidamente los datos.
2. El nuevo sistema debe desarrollarse aplicando patrones y recomendaciones de programación que incrementen la seguridad de datos.
3. Las contraseñas deberán ir cifradas.
4. El tiempo de aprendizaje por un usuario será menor a 5 minutos.
5. La aplicación mostrará mensajes de error informativos en caso de que los datos introducidos por el usuario sean erróneos.
6. Se le informará al usuario cuando la aplicación este cargando los datos.
7. El usuario obtendrá feedback si las acciones se realizan correctamente.
8. La parte Web de la aplicación deberá tener un diseño responsive, es decir, la interface gráfica se debe adaptar a cualquier tipo de pantalla.
9. La aplicación deberá tener la mayor escalabilidad posible, tanto horizontalmente como verticalmente.
10. Se deben poder conectar muchos usuarios al mismo tiempo.
11. Se deben poder almacenar una gran cantidad de datos con un buen rendimiento.
12. La aplicación deberá ser multiplataforma.

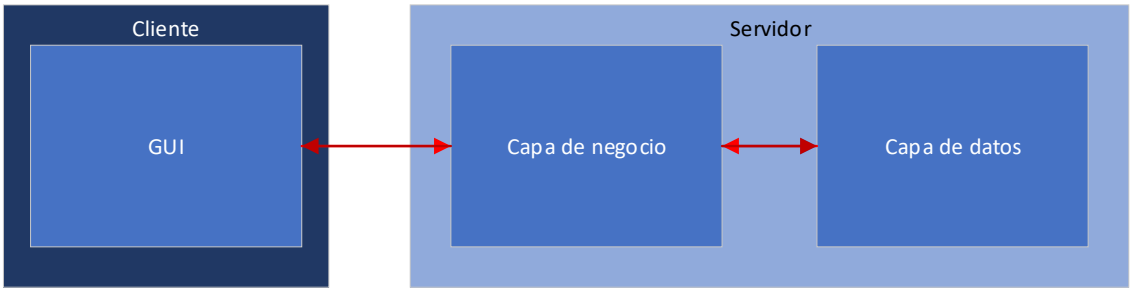
3. Solución del problema

Una vez analizado el problema se planteará una solución para satisfacer los requisitos tanto funcionales como no funcionales. Para ello describiremos como debe ser la arquitectura del software y se comentarán sus diversas capas junto a las tecnologías usadas en cada capa.

3.1 Arquitectura de la aplicación

Definir la arquitectura de la aplicación sirve para planificar y proyectar la estructura, el funcionamiento e interacción entre las partes del software.

Para desarrollar la aplicación usaremos una arquitectura multicapa cliente/servidor, es decir, las tareas se repartirán entre un cliente y un servidor, el cliente realizará peticiones y el servidor le responderá.

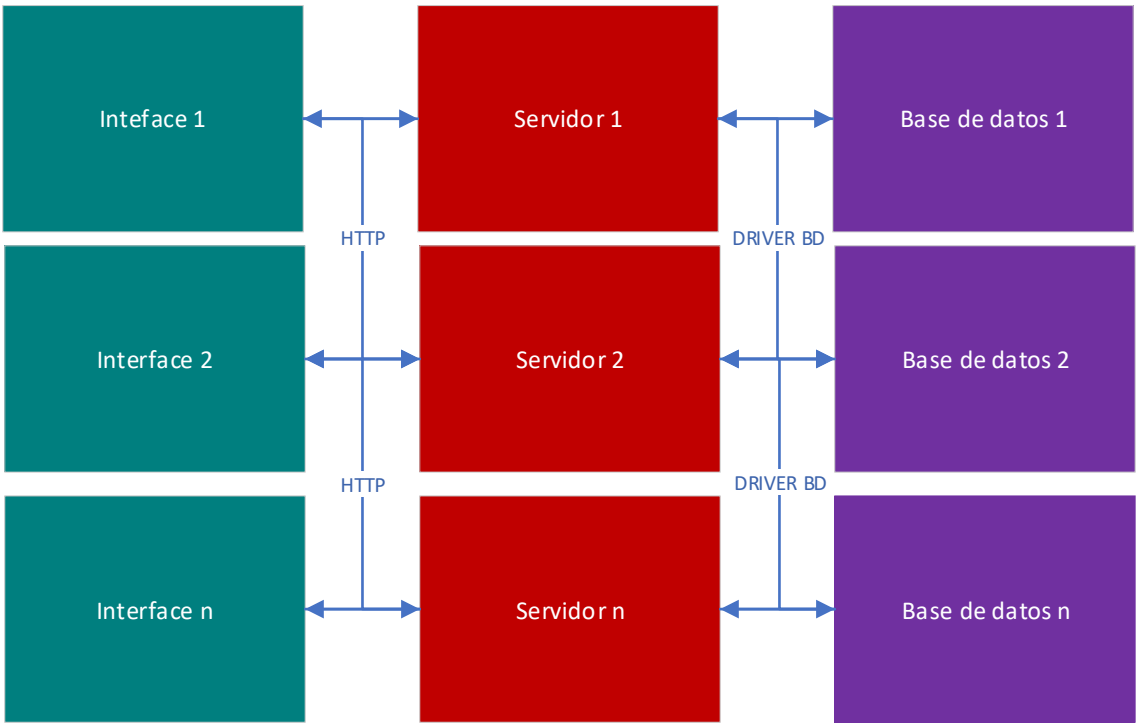


Esta arquitectura puede ser desplegada en una sola computadora, pero en nuestro caso cada parte se ejecutará en un sitio diferente. Esto garantizarán en cierta medida escalabilidad, independencia y seguridad. Seguidamente explicaremos las tres capas más detalladamente.

La arquitectura se separa en tres partes principales, la GUI, la capa de negocio y la capa de datos. En nuestro caso particular la GUI se ejecutará en el dispositivo del cliente, la capa de negocio en un servidor y la capa de datos en otro.

Por lo tanto, tendremos una aplicación distribuida. Para el cliente, la capa de negocio al igual que la capa de datos serán una caja negra, sólo podrá saber que realiza peticiones a un servidor, pero no la estructura de nuestro sistema.

En el siguiente diagrama se muestra un ejemplo sobre cómo podría funcionar nuestra aplicación.



3.1.1 GUI (graphical user interface)

Esta capa se encargará de interactuar con el usuario mediante gráficos. En nuestro caso se ha realizado una interface web. Como ya se ha dicho anteriormente nuestra arquitectura garantiza la independencia entre capas, esto quiere decir que perfectamente podríamos tener una interface móvil, otra de escritorio y otra web, todas conectadas a una o varias capas de negocio. La conexión se realizará mediante el protocolo HTTPS. Cabe mencionar que esta capa se ejecutara en el hardware del cliente. Por lo cual dispondremos de una capacidad de computo y almacenamiento bastante limitado.

3.1.2 Capa de negocio

La capa de negocio atenderá las peticiones HTTPS que lleguen desde la GUI. Para ello se implementará una API REST, ésta es una interfaz HTTPS donde se intercambiarán datos JSON (en nuestro caso).

La API REST, nos permitirá ver, actualizar, insertar y borrar en todos los registros de base de datos para usuarios con privilegios suficientes. Además, nos devolverá indicadores útiles como productos populares, últimos productos… etc. Y por supuesto las recomendaciones de cada usuario.
Para hacer posible todo lo anteriormente dicho se usarán peticiones HTTPS como GET, POST, UPDATE y DELETE.

Como ya se ha nombrado, esta capa funcionará coma una caja negra, es decir, llegarán datos, los tratará y los devolverá sin que los usuarios sepan que ha pasado de por medio. De este modo garantizamos que nadie pueda plagiar o manipular el código.

Esta capa se ejecutará en un servidor, por lo cual, tendremos una buena capacidad de computo para ejecutar toda la lógica necesaria, como, por ejemplo, ejecutar algoritmos, atender un gran número de peticiones... etc.

La capa de negocio no dispone de persistencia de datos en disco, solo se encarga de la lógica, por lo cual es necesario que se guarden los datos en algún sitio. Por esto, se conectará mediante un driver a una capa de datos.

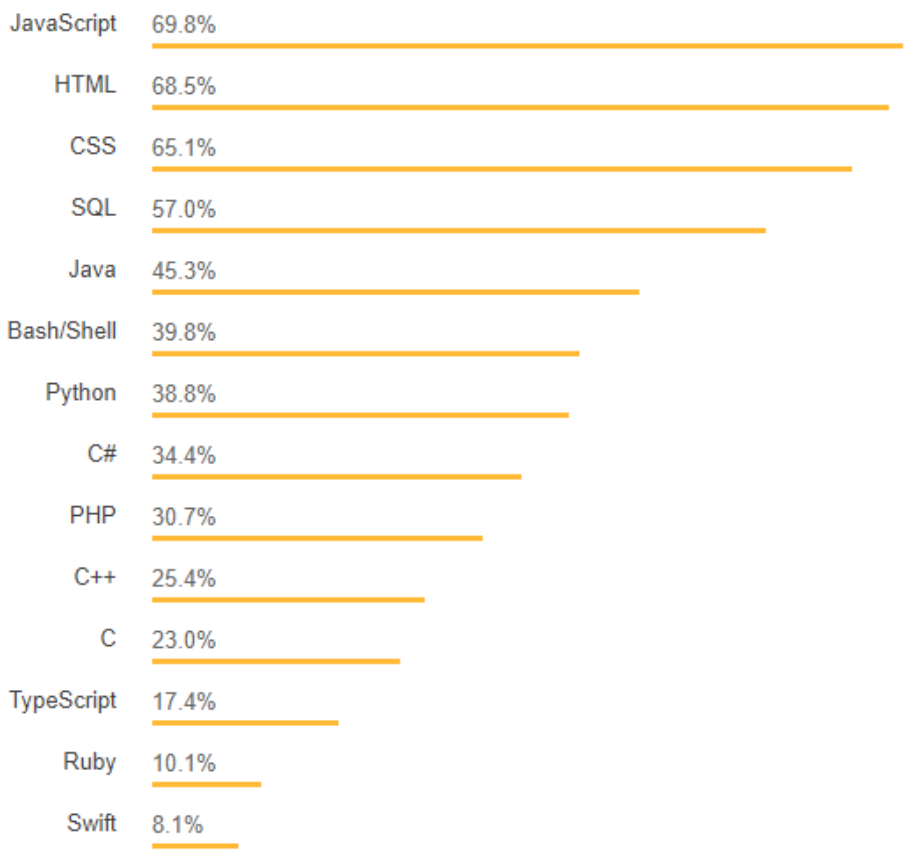
3.1.3 Capa de datos

Aquí haremos que nuestros datos persistan en cuanto al tiempo mediante el uso de bases de datos. La capa de negocio usará los datos que se guarden en esta capa para funcionar.

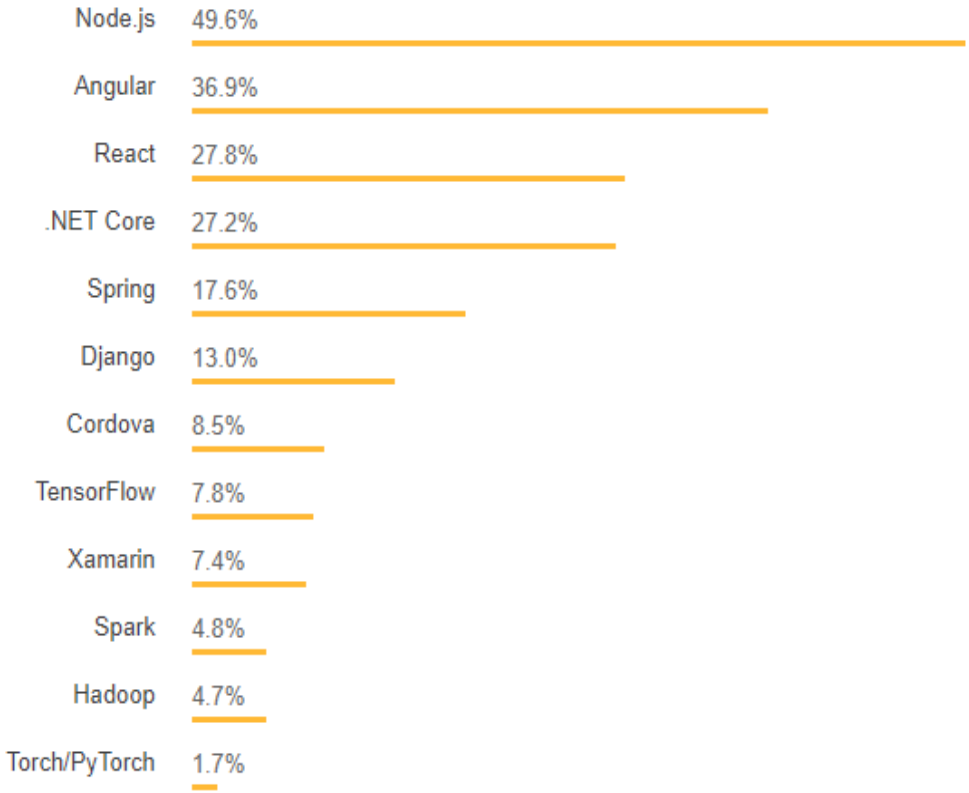
3.2 Tecnologías utilizadas

En el siguiente apartado comentaremos y justificaremos las tecnologías usadas. Para elegir una tecnología nos fijaremos en diversos factores. En primer lugar, mediremos el grado de actividad de la comunidad. Esto es importante, a mayor comunidad, más probabilidad hay de que alguien que haya tenido el mismo problema que nosotros y ya se haya resuelto. Uno de los sitios más conocidos de preguntas y respuestas acerca de programación es Stack Overflow. A continuación, se mostrarán algunas estadísticas de Stack Overflow.

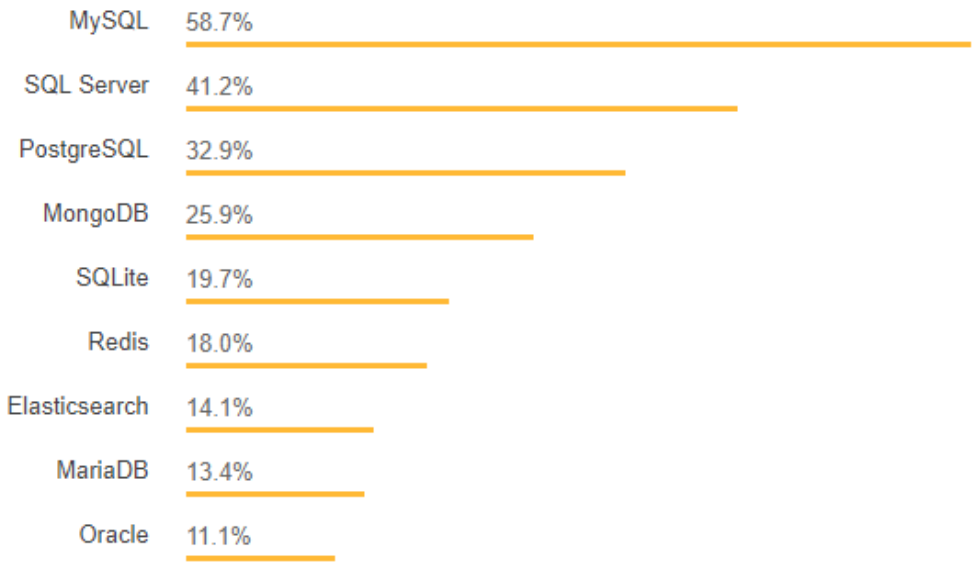
- Lenguajes de programación más populares



- Frameworks, librerías y herramientas más populares.



- Bases de datos más populares



Como podemos ver en los gráficos JavaScript y sus tecnologías como node o angular están en los mejores lugares. Por lo tanto, en 2018 hay una comunidad bastante activa con un gran número de preguntas junto a sus respuestas.

Por otro lado, tenemos el índice TIBOE, que es un indicador sobre la popularidad de los lenguajes de programación.

Las calificaciones del índice TIBOE se basan en el número de ingenieros calificados, cursos y proveedores externos en todo el mundo. Los motores de búsqueda populares como Google, Bing, Yahoo, Wikipedia, Amazon, YouTube y Baidu se usan para calcular dichas calificaciones.

Dec 2018	Dec 2017	Change	Programming Language	Ratings	Change
1	1		Java	15.932%	+2.66%
2	2		C	14.282%	+4.12%
3	4	▲	Python	8.376%	+4.60%
4	3	▼	C++	7.562%	+2.84%
5	7	▲	Visual Basic .NET	7.127%	+4.66%
6	5	▼	C#	3.455%	+0.63%
7	6	▼	JavaScript	3.063%	+0.59%
8	9	▲	PHP	2.442%	+0.85%
9	-	⇈	SQL	2.184%	+2.18%
10	12	▲	Objective-C	1.477%	-0.02%
11	16	⇈	Delphi/Object Pascal	1.396%	+0.00%
12	13	▲	Assembly language	1.371%	-0.10%
13	10	▼	MATLAB	1.283%	-0.29%
14	11	▼	Swift	1.220%	-0.35%

En las estadísticas de TIBOE podemos ver cómo según ellos Java es el lenguaje más popular del mundo. La popularidad es importante, usar una tecnología desconocida te va a dar menos garantías y fiabilidad.

Además del índice TIBOE tenemos IEEE SPECTRUM, donde para realizar el ranking usan conferencias acerca de un lenguaje específico, proyectos de código abierto y ofertas de trabajo. Sus proveedores de información, son plataformas como GitHub, Reddit o las tendencias de Google.

Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C++		99.7
3. Java		97.5
4. C		96.7
5. C#		89.4
6. PHP		84.9
7. JavaScript		82.6
8. Go		76.4
9. Scala		72.1
10. Ruby		71.4
11. HTML		71.2
12. Perl		57.4
13. Swift		53.9

Como este proyecto es creado por un estudiante también es interesante que las tecnologías usadas puedan dar cierta proyección al alumno. Por esto, también se tendrá en cuenta la demanda en el mercado, para ello, recurriremos a InfoJobs, que es una aplicación famosa para buscar empleo. Según InfoJobs estas son las tecnologías más demandadas de la actualidad.

Obviamente no sólo nos podemos centrar en que tan popular es un lenguaje ni en la actividad de la comunidad, hay otros factores que se deben valorar, aunque todos estos datos nos pueden ayudar a tomar una decisión.

Otro factor a tener en cuanta es la curva de aprendizaje. Al conocer una tecnología, aunque no sea la mejor opción te puede ahorrar mucho tiempo. No es lo mismo en cuanto a tiempo empezar a aprender de 0 que usar una tecnología que ya se conoce.

El alumno conoce JavaScript y Java, por lo tanto, algunos de sus entornos serán la mejor opción para desarrollar la aplicación. Tanto Java como JavaScript tienen un buen lugar en todos los rankings mostrados anteriormente y pueden satisfacer las necesidades del proyecto.

De los dos lenguajes que nos quedan se elegirá JavaScript, para realizar una página Web y dotarla de dinamismo se necesita JavaScript, por lo tanto, este lenguaje lo tenemos que usar sí o sí en la parte del cliente. JavaScript nos proporcionará cierta comodidad, ya que, tanto en la parte del cliente como en la parte del servidor se está usando el mismo lenguaje.

Además, ocupa los primeros puestos en Stack Overflow, esto quiere decir que actualmente es muy usado y dispone de una comunidad muy activa.

Más adelante se verán algunas diferencias más entre JavaScript y Java, sobre todo en la parte del servidor.

Actualmente JavaScript dispone de un popular marco de trabajo conocido como la pila MEAN. Las tecnologías que forman la pila MEAN son MongoDB, Express, Angular y Node. Estas son las tecnologías más destacables, pero también se han usado otras tecnologías como Boostrap, JWT, Google Signin, Git, GitHub, Moongose... y más.

A continuación, se explicarán algunas de las tecnologías usadas capa por capa.

3.2.1 GUI

Como no podría faltar en cualquier web `para estructurar la página web se usará HTML.

HTML significa HiperText Markup Language (Lenguaje de Marcación de Hipertexto) es un lenguaje que se utiliza comúnmente para establecer la estructura y contenido de un sitio web, tanto de los textos como objetos e imágenes. Los archivos desarrollados en HTML usan la extensión .html.

El lenguaje de HTML funciona por medio de "etiquetas" que describen la apariencia o función del texto enmarcado.

HTML es un lenguaje que carece de estilo, sólo nos proporciona una estructura, para hacer amigable la página Web y mejorar la usabilidad necesitaremos una tecnología que mejore el estilo de nuestra Web.

Las siglas CSS corresponden a la expresión inglesa Cascading StyleSheets, que puede traducirse como “Hojas de estilo en cascada”. El concepto se utiliza en el ámbito de la informática para referirse a un lenguaje empleado en el diseño gráfico.

El lenguaje CSS permite presentar, de manera estructurada, un documento que fue escrito en un lenguaje de marcado. Se usa especialmente en el diseño visual de un sitio web cuando las páginas se hallan escritas en XML o HTML.

Por esto, este lenguaje es perfecto para mejorar el diseño de nuestra página. Para facilitar el desarrollo se usarán algunos componentes ya creados por la comunidad.

Bootstrap es un framework desarrollado y liberado por Twitter que tiene como objetivo facilitar el diseño web. Permite crear de forma sencilla webs de diseño adaptable, es decir, que se ajusten a cualquier dispositivo y tamaño de pantalla y siempre se vean igual de bien. Es Open Source o de código abierto, por lo que se puede usar de forma gratuita y sin restricciones.

Y con todo lo dicho ya podemos crear una página Web, el problema es que todo lo descrito anteriormente es estático y necesitamos dinamismo, en nuestra aplicación aparecerán listas dinámicas, objetos iterativos, entre otros … Además del dinamismo también se necesitará cierta lógica para mostrar los datos correctamente, hacer peticiones HTTPS …

Para conseguir todo esto necesitamos un lenguaje que pueda manejar operaciones lógicas. El lenguaje que se usara para desarrollar esta parte es TypeScript.

TypeScript es un lenguaje de programación de alto nivel que implementa muchos de los mecanismos más habituales de la programación orientada a objetos, este lenguaje nos puede ofrecer grandes beneficios deseables en aplicaciones grandes.

La característica fundamental de TypeScript es que compila en Javascript nativo, por lo que se puede usar en todo proyecto donde se esté usando Javascript. Dicho con otras palabras, cuando se usa TypeScript en algún momento se realiza su compilación, convirtiendo su código a Javascript común. El navegador, o cualquier otra plataforma donde se ejecuta Javascript, nunca llegará a enterarse que el código original estaba escrito en TypeScript, porque lo único que llegará a ejecutar es el Javascript resultante de la compilación.

En resumen, TypeScript es lo que se conoce como un "superset" de Javascript, aportando herramientas avanzadas para la programación que traen grandes beneficios a los proyectos.

Angular 6 es un framework para TypeScript desarrollado por Google. Angular nos facilitará la creación de aplicaciones SPA, estas son aplicaciones de una sola página, es decir, una aplicación web donde la navegación y la carga de datos son totalmente dinámicas y las llamadas a la capa de datos son asíncronas, por lo tanto, no es necesario refrescar el navegador en ningún momento.

Este framework nos será de gran utilidad para desarrollar la parte que se ejecutará en el cliente, nos proporcionará una estructura clara y nos garantizará diversas funcionalidades, como reutilización de código, un buen rendimiento, estandarización ...

Angular 6 se combinará con bibliotecas como JQuery para obtener más funcionalidades. JQuery es una librería famosa para JavaScript que nos facilitará en gran medida el trabajo. Ya que nos permite un fácil manejo del DOM, además de añadir más dinamismo a nuestra Web.

Principalmente se ha optado por Angular porque el alumno ya conocía la tecnología, Angular actualmente es muy usado en el desarrollo de aplicaciones Web, esto se ve reflejado en la gran cantidad de demanda, Angular es usado por compañías como YouTube, PayPal, Google, AWS... entre otros.

Su principal competidor es React, que es una biblioteca de JavaScript. React nos brinda mayor libertad ya que no es un framework, pero no nos proporciona una estructura sólida a la que seguir. React es usado por empresas como Facebook, Instagram, Netflix, Yahoo, Whatsapp, Dropbox, Microsoft… por lo cual esto es una garantía que es una tecnología bastante útil. VueJS es otra tecnología similar a Angular pero actualmente tiene mucha menos cuota de mercado, sus creadores no son una gran corporación y es una tecnología más nueva pero con muy buenas críticas.

Actualmente Angular tiene detrás una empresa como Google, esto da cierta seguridad. Además, si buscamos ofertas de empleo en España, Angular es la tecnología más buscada por las empresas para desarrollos Web, todo esto son puntos a favor de Angular.

Otra opción sería hacerlo a pelo, es decir todo desde 0, esta opción nos ofrecería mayor libertad en el desarrollo, pero, no compensaría en cuanto a tiempo.

Todas las tecnologías son buenas y sirven para el mismo propósito, cada una tiene ventajas e inconvenientes, pero estas son irrelevantes si ya se conoce una tecnología concreta.
En Internet hay un montón de comparativas y al final todas llegan al mismo punto, todas las tecnologías para la GUI web conocidas son buenas si las sabes usar.

Por lo tanto, la elección ha sido tomada por el conocimiento del alumno sobre Angular y por los buenos puestos en los gráficos acerca de la popularidad y actividad de la comunidad.

3.2.2 Capa de negocio

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google

Principalmente se ha elegido Node porque en lenguajes como Java y PHP, cada conexión genera un nuevo hilo que potencialmente viene acompañado de 2 MB de memoria. En un sistema que tiene 8 GB de RAM, esto da un número máximo teórico de conexiones concurrentes de cerca de 4.000 usuarios. A medida que crecen las conexiones necesitaremos más y más RAM, por lo tanto, se necesitan muchos recursos para atender a muchos clientes.

Node resuelve este problema. En lugar de generar un nuevo hilo para cada conexión (junto a la memoria acompañante), cada conexión dispara una ejecución de evento dentro del proceso del motor de Node. Node también afirma que nunca se quedará en punto muerto, porque no se permiten bloqueos y porque no se bloquea directamente para llamadas E/S. Node afirma que un servidor que lo ejecute puede soportar decenas de miles de conexiones concurrentes.

Por lo tanto, Node es la mejor opción actual en el mercado en cuanto a escalabilidad, gracias a necesitar menos recursos que el resto de tecnologías. Además de su escalabilidad también tiene un buen rendimiento gracias al uso de motor V8 de Google.

Node es modular, por lo tanto podemos ir añadiendo módulos que nos proporcionen funcionalidades deseadas, a continuación comentaremos algunos de los módulos elegidos.

ExpressJS es un framework para Node que nos proporciona innumerables ventajas como por ejemplo la opción de crear rutas para crear una API REST, esto será esencial para que la GUI se pueda comunicar con el servidor.

Con ExpresJS usaremos body-parser que es un middleware para extraer los cuerpos de las peticiones HHTPS.

También usaremos express-fileupload que es otro middleware para manejar archivos.

JWT (json web token) es otro de los elegidos, este es un estándar para la creación de tokens de acceso basado en JSON y que nos brinda una transmisión de información segura, todas las peticiones HTTPS (Menos el registro y el login) que se hagan de la GUI al servidor necesitarán un token válido para funcionar, esto nos garantiza que ningún intruso pueda colarse sin haberse validado.

El módulo bcryptjs es otro de los elegidos, lo usaremos para aumentar la seguridad, su principal funcionalidad es el cifrado y la creación de hashes.

Y para la autenticación por Google el módulo google-auth-library es la opción elegida.

Para manejar mejor la capa de datos se ha usado un ODM llamado Mongoose. Un ODM nos permite modelar nuestra base de datos desde NodeJS y nos facilitará el tema del mapeado e independencia de datos.

A moongose lo podemos complementar con módulos como mongoose-unique-validator, este módulo nos permite poder hacer validaciones de campos únicos.

3.2.3 Capa de datos

MongoDB es una de las bases de datos más rápidas del mercado, al no ser relacional no tiene que hacer tantas comprobaciones de integridad, referencia…etc. Las bases de datos relacionales funcionan muy lentamente si tenemos una cantidad de datos muy grandes, en nuestro caso como ya se ha comentado anteriormente será así. Por lo tanto, MongoDB es la mejor opción para grandes tamaños de datos.
Gracias a su buena escalabilidad horizontal el rendimiento no disminuirá si la cantidad de datos es muy grande, así que principalmente por temas de volumen y rendimiento elegimos MongoDB.

3.2.4 Otros

Para facilitar el desarrollo usaremos Git que es un software de control de versiones. Nos será útil para compartir el código en caso de que fuera necesario. Para realizar backups usaremos GitHub que es un repositorio online para alojar proyectos utilizando Git.

Para registrar todas las rutas de nuestra API usaremos Postman. Y como editor de texto Visual Studio Code.

3.2 Diseño e implementación

En esta parte ya ha llegado el momento de mostrar y explicar el resultado de lo que se ha creado en este proyecto.

Tanto la parte del Cliente (Angular) como la parte del Servidor (Node) se han podido desplegar en los servicios de publicación web avanzados de la UPV.
En cambio, no había soporte para la base de datos, por lo tanto, para la capa de datos (MongoDB) se ha usado AWS, ya que, nos proporciona medio giga de almacenamiento gratis.

4.1 Capa de datos

Nuestra aplicación deberá guardar la información del usuario, esta información deberá ser persistente en memoria. Esto será esencial para identificarlo. De este modo determinaremos sus gustos y podremos dar buenas recomendaciones.

Para hacer esto, usaremos un identificador que será único para cada usuario, además de este identificador también guardaremos su email que también será único y no nulo.

La aplicación tendrá un mantenimiento, normalmente llamado back office. Por lo tanto, los usuarios tendrán un rol para determinar su grado de privilegios. Para identificarse se usará su mail y una contraseña. Los usuarios se podrán autentificar por medios externos como por ejemplo la cuenta de Google, para ellos crearemos un campo para saber el tipo de autentificación del usuario. Otros atributos que tendrá el usuario serán su nombre, edad, género, ciudad y país.

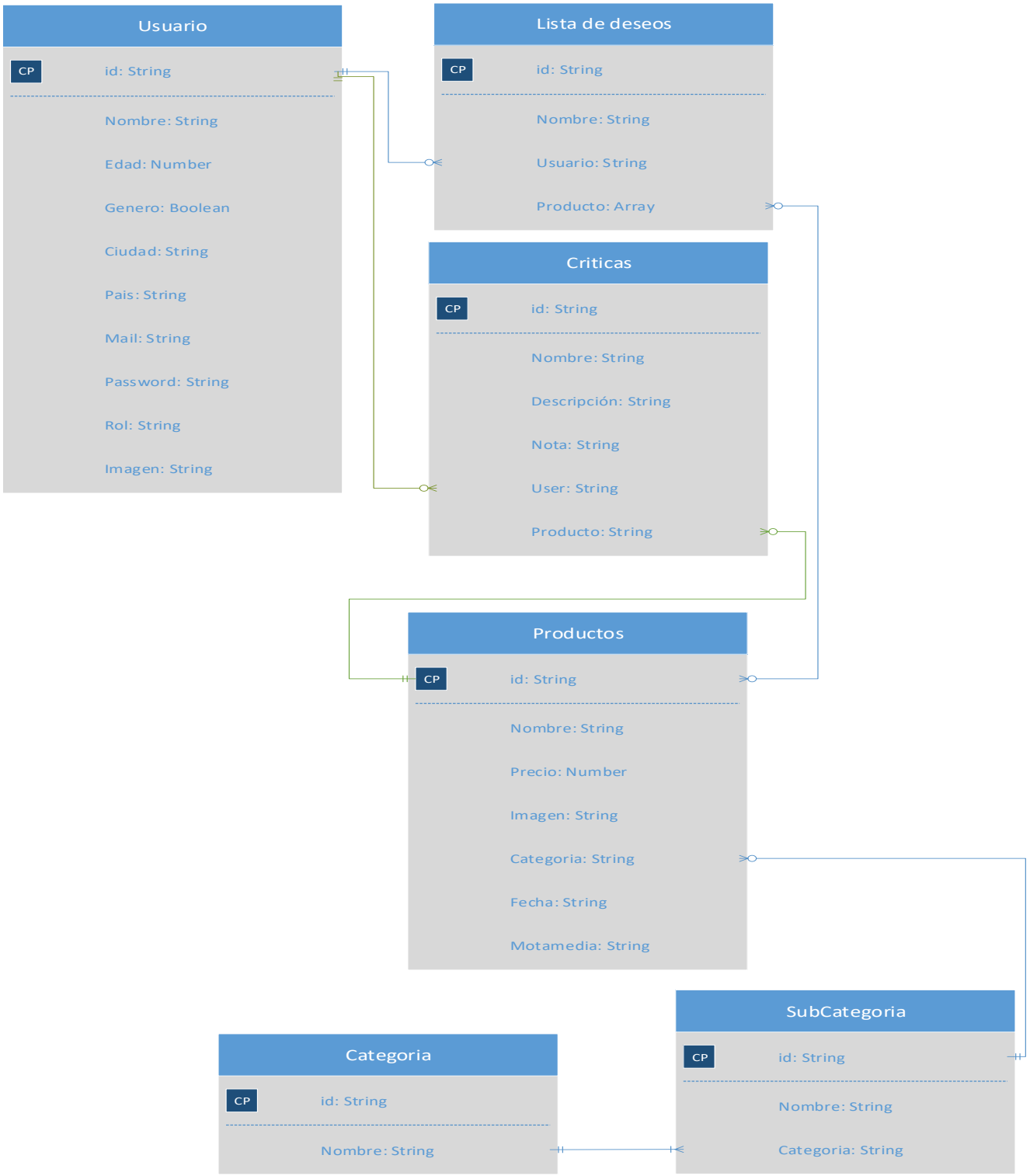
Para poder hacer unas buenas recomendaciones será necesario que guardemos las críticas de los usuarios. Por esto cada crítica que haga un usuario será registrada, estas críticas se identificarán en por el usuario que las creo y por un identificador único que tendrá cada crítica. Las críticas guardarán las notas de los productos y opcionalmente un título y una descripción.

Si un usuario lo desea también podrá crear listas de productos, estas listas se identificarán por el usuario y un identificador único. Tendrán un nombre y contendrán todos los productos que un usuario desee.

Aparte de los usuarios tendremos los productos. Estos también se identificarán por un identificador único, entre sus atributos encontraremos una descripción, su fecha de publicación, el precio y su nombre. Cada producto pertenecerá a una subcategoría. También se guardará la nota media aritmética de los votos de los usuarios. Otro campo importante es el de la popularidad del producto, que será el número de votos que haya obtenido el producto.

Para poder clasificar adecuadamente los productos se guardarán categorías que englobarán subcategorías. Tanto unas como las otras se identificarán por identificadores únicos, las dos dispondrán de nombre. Las subcategorías además de por su identificador único también deberán hacer referencia a una categoría.

Desde un punto de vista relacional la base de datos la podríamos modelar del siguiente modo:



Para implementar la base de datos se han creado 3 colecciones, la colección de usuarios, de productos y de categorías.

Las tablas de críticas y deseos se han embebido en la colección de usuarios y las subcategorías se han embebido en la colección de críticas.

A continuación, se mostrará un ejemplo en formato JSON de cada colección.


```
{
  "_id" : ObjectId("5bbd3035bdee6e299c826b76"),
  "role" : "USER_ROLE",
  "google" : false,
  "nombre" : "test2",
  "edad" : "33330303",
  "genero" : true,
  "ciudad" : "Al",
  "pais" : "Es",
  "email" : "test2@test.com",
  "password": "$2a$10$r8j7NGJtuwLqmtbSBb6tW.STPVDEdqiEwV72UNaX0oFbNQNA39YtG",
  "listasDeDeseos" : [
    {
      "producto" : [
        ObjectId("5bc767d6bd4596293c63c7d4")
      ],
      "_id" : ObjectId("5bc767d6bd4596293c63c7d3"),
      "nombre" : "Mi lista 1"
    }
  ],
  "criticas" : [
    {
      "_id" : ObjectId("5bbd3231bdee6e299c826b98"),
      "nota" : 1,
      "producto" : ObjectId("5bbd3132bdee6e299c826b80")
    },
    {
      "_id" : ObjectId("5bbd323abdee6e299c826b9a"),
      "nota" : 5,
      "producto" : ObjectId("5bbd3140bdee6e299c826b81")
    },
    {
      "_id" : ObjectId("5bbd325dbdee6e299c826b9c"),
      "nota" : 5,
      "producto" : ObjectId("5bbd3151bdee6e299c826b83")
    }
  ],
```

```
  "__v" : 0
}
```

Como ya se ha dicho, listasDeDeseos y críticas son los documentos embebidos, tanto las críticas como las listasDeDeseos irán variando en función de las críticas o los deseos del usuario.

También se puede observar como la contraseña está cifrada, esto se hará automáticamente en el la capa de negocio.

A continuación, se mostrará la colección de productos:

```
 {
  "_id" : ObjectId("5bbd3148bdee6e299c826b82"),
  "nombre" : "I5",
  "precio" : 32,
  "descripcion" : "23",
  "subcategoria" : ObjectId("5bb660caa6451317e0ea22d1"),
  "fecha" : ISODate("2018-10-09T22:52:56.807Z"),
  "__v" : 0,
  "notamedia" : 3.5,
  "popularidad" : 4
}
```

Y por último la colección de las categorías:

```
{
  "_id" : ObjectId("5bb78f876afdf22ab807da15"),
  "nombre" : "Ropa",
  "__v" : 0
  "subcategoria" : [
   {
    "_id" : ObjectId("5bb660caa6451317e0ea22d1"),
    "nombre" : "Vaqueros"
  }
  ]
}
```

En este caso subcategorías es la colección embebida.

4.2 Capa de negocio

En esta capa se ha implementado una API RESTFul, esta API será consumida por la parte del cliente.

El término REST (Representational State Transfer) se originó en el año 2000, descrito en la tesis de Roy Fielding, padre de la especificación HTTP. Un servicio REST no es una arquitectura software, sino un conjunto de restricciones con las que podemos crear un estilo de arquitectura software, la cual podremos usar para crear aplicaciones web respetando HTTP.

Hoy en día la mayoría de las empresas utilizan API REST para crear servicios. Esto se debe a que es un estándar lógico y eficiente para la creación de servicios web. Por poner algún ejemplo tenemos los sistemas de identificación de Facebook, la autenticación en los servicios de Google (hojas de cálculo, Google Analytics, …).

Por lo tanto, una API RESTful es una interfaz (API) que utiliza solicitudes HTTP para obtener datos, las peticiones más usadas son GET, PUT, POST y DELETE.

Hay que mencionar que tanto las peticiones POST como PUT necesitarán un JSON. Se usará x-www-form-urlencoded, que codificará los valores en tuplas llave-valor separadas por '&', con un '= ‘entre la llave y el valor.

La API nos devolverá datos en formato JSON e interactuará con la capa de datos mediante un driver de base de datos.

A continuación, se presentará una explicación de cada petición.

En primer lugar, explicaremos los usuarios, hay que recordar que tanto los deseos como las críticas van embebidas en los usuarios:

Ver el intervalo de usuarios deseado

GET [IP]:[PUERTO]/usuario?desde=0&hasta=5?token=[tokendevalidacion]

Mostrar usuario por id

GET [IP]:[PUERTO]/usuario/[idDeUsuario]?token=[tokendevalidacion]

Actualizar un usuario según su id, se debe proporcionar el body que será un JSON con un formato igual al ejemplo expuesto en la capa de datos.

PUT [IP]:[PUERTO]/usuario/[idDeUsuario]?token=[tokendevalidacion]

Insertar un usuario. También se debe proporcionar un JSON como en el ejemplo anterior. Esta es la única petición que no necesita de token para validarse, ya que, no se puede tener un token antes de estar registrado.

POST [IP]:[PUERTO]/usuario

Elimina un usuario por su id

DELETE [IP]:[PUERTO]/usuario/[idDeUsuario]?token=[tokendevalidacion]

En segundo lugar, se mostrarán las peticiones de los productos.

Ver el intervalo de productos deseado

GET [IP]:[PUERTO]/producto?desde=0&hasta=5?token=[tokendevalidacion]

Ver el intervalo de productos deseado por subcategoria

GET [IP]:[PUERTO]/producto/subcat/[idSubCategoria]?desde=0&hasta=999?token=[tokendevalidacion]

Ver un producto por su id

GET [IP]:[PUERTO]/producto/[idProducto]?token=[tokendevalidacion]

Insertar un nuevo producto

POST [IP]:[PUERTO]/producto]?token=[tokendevalidacion]

Actualizar un producto

PUT [IP]:[PUERTO]/producto/[idProducto]?token=[tokendevalidacion]

Actualizar la nota media de un producto

PUT [IP]:[PUERTO]/critica/[idProducto]token=[tokendevalidacion]

Eliminar un producto

DEL [IP]:[PUERTO]/producto/[idProducto]?token=[tokendevalidacion]

En tercer lugar, podremos ver las categorías.

Ver una categoría

GET [IP]:[PUERTO]/categoria?desde=0&hasta=5?token=[tokendevalidacion]

Insertar una categoría

POST [IP]:[PUERTO]/categoria?token=[tokendevalidacion]

Actualizar una categoría

PUT [IP]:[PUERTO]/categoria/[idCategoria]?token=[tokendevalidacion]

Eliminar una categoría

DEL [IP]:[PUERTO]/categoria/[idCategoria]?token=[tokendevalidacion]

Y con esto ya tendremos un CRUD básico de todas las colecciones, pero necesitaremos un token de autentificación.

Insertar un usuario, si dicho usuario esta en la base de datos nos devolverá el token.

POST [IP]:[PUERTO]/login

El token expira cuando pasa cierto tiempo, si el usuario que sigue navegando tiene un token valido puede pedir un nuevo token

GET [IP]:[PUERTO]/login/renuevatoken?token=[tokendevalidacion]

En nuestra aplicación también se manejan imágenes, por lo tanto, debemos tener herramientas para poder manejarlas.

Ver una imagen por su ruta

GET [IP]:[PUERTO]/img/usuarios/[idImagen] ?token=[tokendevalidacion]

Actualizar o Insertar una imagen por el id del usuario

PUT [IP]:[PUERTO]/upload/usuarios/[idUsuario] ?token=[tokendevalidacion]

Actualizar o Insertar una imagen por el id del producto

PUT [IP]:[PUERTO]/upload/productos/[idProducto] ?token=[tokendevalidacion]

También será necesario un buscador.

Búsqueda por colección

GET [IP]:[PUERTO]/busqueda/coleccion/[Colección]/[CampoDeBusqueda]?desde=0&hasta=5?token=[tokendevalidacion]

Búsqueda en toda la base de datos

GET [IP]:[PUERTO]/busqueda/bd/[CampoDeBusqueda]?desde=0&hasta=5?token=[tokendevalidacion]

Indicadores útiles.

Se muestran los productos más viejos o más nuevos

GET [IP]:[PUERTO]/indicadores/[AscendenteODescendente]/fecha?token=[tokendevalidacion]

Se muestran productos ordenados por más nuevos con muchas y buenas valoraciones

GET [IP]:[PUERTO]/indicadores/tendencias?token=[tokendevalidacion]

Se muestran productos ordenados por popularidad o impopularidad.

GET
[IP]:[PUERTO]/indicadores/[AscendenteODescendente]/popularidad?token=[tokendevalidacion]

Se muestran productos ordenados por nota.

GET
[IP]:[PUERTO]/indicadores/[AscendenteODescendente]/notamedia?token=[tokendevalidacion]

Se muestran productos recomendados para un usuario.

GET [IP]:[PUERTO]/kvecionos/[idUsuario]?token=[tokendevalidacion]

4.2.1 Web Scraping

Al inicio, la aplicación estará vacía de productos, esto es un problema ya que nadie va a usar nuestra aplicación sin productos. Las marcas de productos tampoco añadirán sus productos a una aplicación que no es conocida. Por lo tanto, al principio tenemos que idear un modo de añadir todos los productos de las grandes tiendas de un modo automático. Ya que, manualmente sería demasiado costoso por la gran cantidad de productos que existen. De esto se deberá encargar nuestra capa de negocio.

El Web scraping es una forma automática de extraer información de sitios web de forma automática y esto es justo lo que necesitamos.

Para realizar esto usaremos NodeJS, ExpressJS, Request y Cheerio. Con Request se hacen peticiones HTTPS desde el servidor y se captura todo el cuerpo de la página web, seguidamente con Cheerio (implementación de JQuery para el servidor que nos permite manejar fácilmente el DOM) se capturar todos los datos deseados para luego ser insertados en la base de datos, para ello marcaremos los elementos del DOM que deseemos y luego simplemente crearemos bucles para recorrer todas las rutas de la aplicación y que se extraigan automáticamente todos los productos de cada tienda que deseemos. Esto nos servirá para todas las páginas que tengan la misma estructura, pero fácilmente se puede adaptar para cada tipo de página.
Hay que tener en cuenta que cada tienda tiene una estructura diferente y deberemos crear un programa particular para cada tienda diferente.

Cabe mencionar que esto no es ilegal directamente, pero si podríamos tener algún problema legal por alguna cuestión indirecta.
En un principio es como hacer un copiar y pegar, o un recorte de pantalla a una imagen.

Especialmente deberemos ir con cuidado con el Real Decreto Legislativo 1/1996, de 12 de abril, que habla sobre los derechos de propiedad intelectual. La base de datos según como se mire (esto depende de su originalidad) podría considerarse propiedad intelectual.
También podríamos violar los términos de uso del sitio al que realicemos la extracción de datos, por esta cuestión tienen derecho a bloquearnos. Y si por ejemplo hiciéramos una imitación de un sitio web nos podrían acusar de conducta desleal.

A las tiendas no les agrada y se suelen proteger contra los bots scrapers mediante captchas o bloqueos de IP, por lo cual sería recomendable usar la red Tor o ir variando las IP mediante un Proxy.

Hay empresas que se dedican a esto. Como por ejemplo Visual Web Ripper, que además nos proporciona un software para hacer scraping web. Esto sería una opción si no queremos desarrollar el software nosotros mismos.

4.2.2 Algoritmos usados

Para poderles recomendar a los usuarios será necesario usar un algoritmo. En nuestro caso se ha implementado el algoritmo k-vecinos (k-nearest neighbors).

Una vez el usuario ha votado un número mínimo de productos se usará este algoritmo para predecir qué productos le pueden gustar, el algoritmo se activará cada vez que se llame a la petición:
HTTPS GET [IP]:[PUERTO]/kvecionos/[idUsuario]?token=[tokendevalidacion]
Seguidamente devolverá un número de productos que le puedan interesar al usuario.

En primer lugar, se calculará la similitud de este usuario con los demás, para ello se usarán sus votos, estos votos serán tratados como un vector en un espacio multidimensional.

La similitud se puede calcular de diferentes modos, usaremos la métrica de similitud del coseno, este método calcula el ángulo entre el vector de votos coincidentes del usuario respecto a los demás usuarios.

$$sim(x,y) = \frac{\sum_{i \in B_{x,y}} r_{x,i} \cdot r_{y,i}}{\sqrt{\sum_{i \in B_{x,y}} r_{x,i}^2} \cdot \sqrt{\sum_{i \in B_{x,y}} r_{y,i}^2}} \in [0,1]$$

Una vez calculada la similitud calcularemos los K vecinos más cercanos, para ello haremos uso de la métrica de similitud para obtener los k usuarios más afines.

Seguidamente usaremos estos k usuarios más afines para calcular la media ponderada de los productos coincidentes que el usuario al que se le quiere recomendar no ha votado. Se ponderará usando la métrica de similitud.

$$p_{u,i} = \frac{\sum_{n \in G_{u,i}} sim(u,n) \cdot r_{n,i}}{\sum_{n \in G_{u,i}} sim(u,n)}$$

Y finalmente de todos estos productos se le recomendarán al usuario de mayor nota a peor.

Para entender mejor el algoritmo pondremos un ejemplo.

Digamos que tenemos la siguiente matriz y queremos saber los productos que le gustarán al usuario 1. Las filas serán Usuarios y las columnas los productos.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
User1	1	2	-	-	2	-	3	4	-	4	1	-
User2	-	-	1	5	-	5	3	1	-	5	2	1
User3	1	-	-	2	-	1	-	3	4	-	-	-
User4	-	1	4	4	-	-	3	-	5	4	-	1
User5	2	-	5	-	1	-	1	-	-	-	2	1
User6	-	-	5	2	1	-	-	4	-	1	-	2

En primer lugar, calculamos el coseno del User1 con todos los demás. Con ello obtenemos todas las similitudes.

	User2	User3	User4	User5	User6
User1	SimA	SimB	SimC	SimD	SimE

Seguidamente las ordenaremos de mayor a menor, si ponemos k=2, cogeremos los 2 usuarios con mayores valores, en nuestro caso por ejemplo el User4 y el User3.

Una vez tenemos a los k usuarios más afines pasamos a calcular la media ponderada, usando para ponderar la métrica de similitud.
En el caso del ejemplo se ha usado la media aritmética para simplificar el ejemplo, pero en el caso real si se ha usado la media ponderada.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
User1	1	2	2	3	2	0,5	3	4	4,5	4	1	0,5
User3	1	-	-	2	-	1	-	3	4	-	-	-
User4	-	1	4	4	-	-	3	-	5	4	-	1

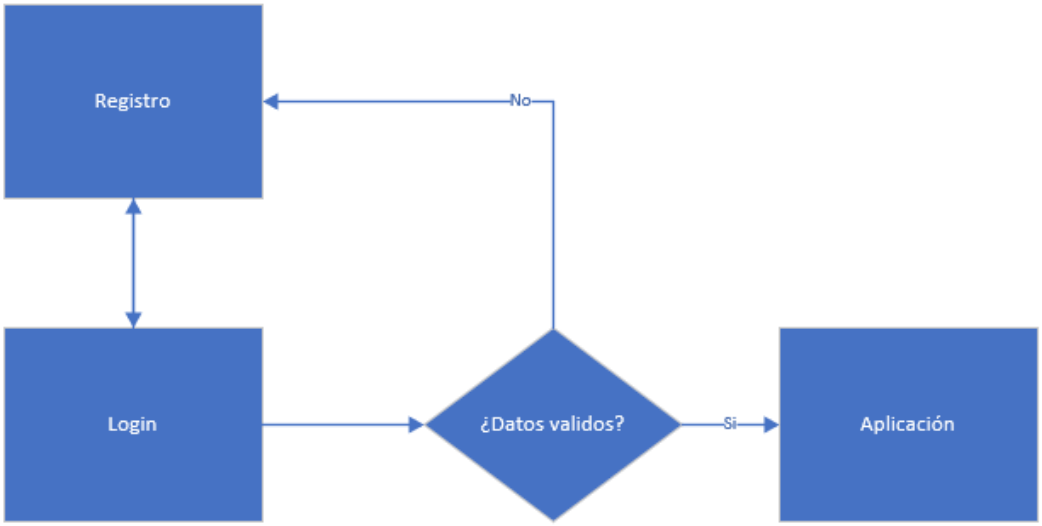
Por lo tanto, al Usuario 1 el producto que más le puede interesar es el P9, ya que tiene un 4,5, en segundo lugar, el P4 y así hasta que deseemos o nos quedemos sin productos.

Este algoritmo ha sido elegido principalmente porque da buenos resultados para su sencillez. Pero el alumno es consciente de que hay algoritmos que pueden mostrar mejores recomendaciones.

En nuestro caso se ha implementado el algoritmo desde cero, pero en un caso real lo más práctico será usar algo que ya este hecho mediante una API, de este modo nos ahorraremos mucho tiempo.

4.3 GUI

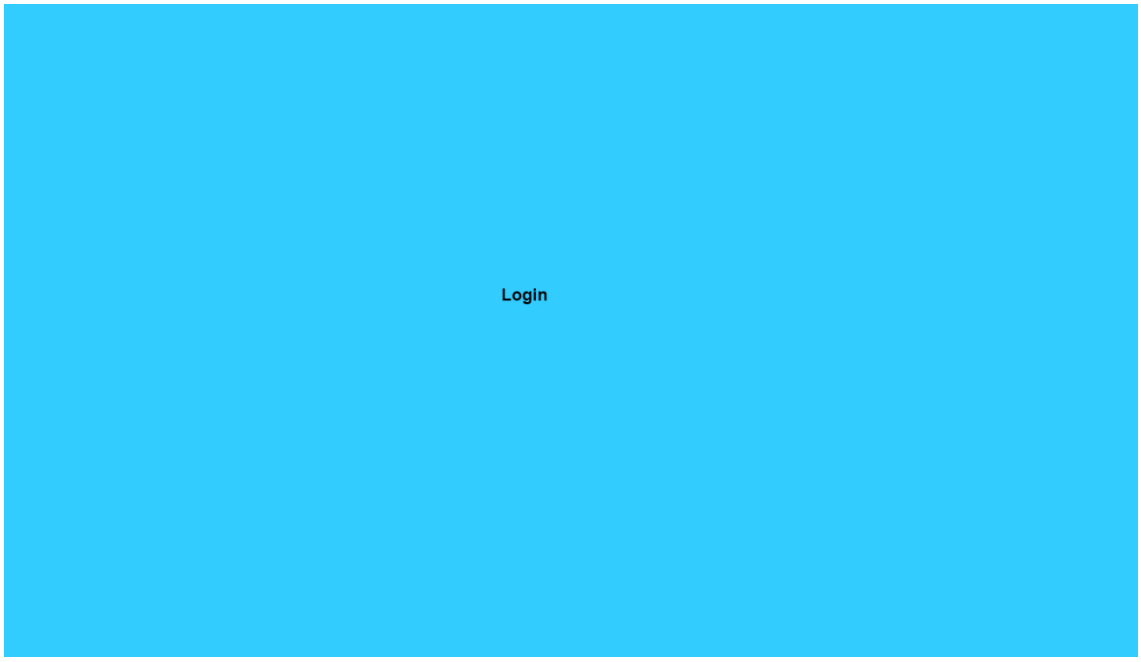
En esta parte explicaremos sobre la capa de presentación. El login y el registro serán las únicas rutas a las que se podrá acceder si autenticación.

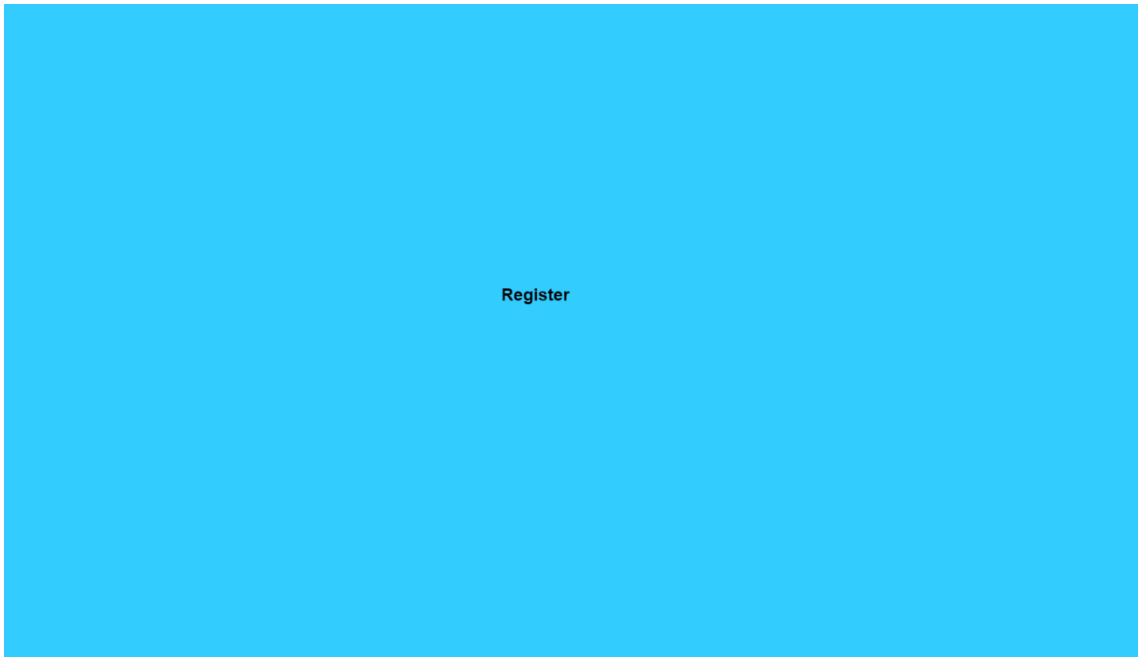


La ruta del login y del registro serán las siguientes:

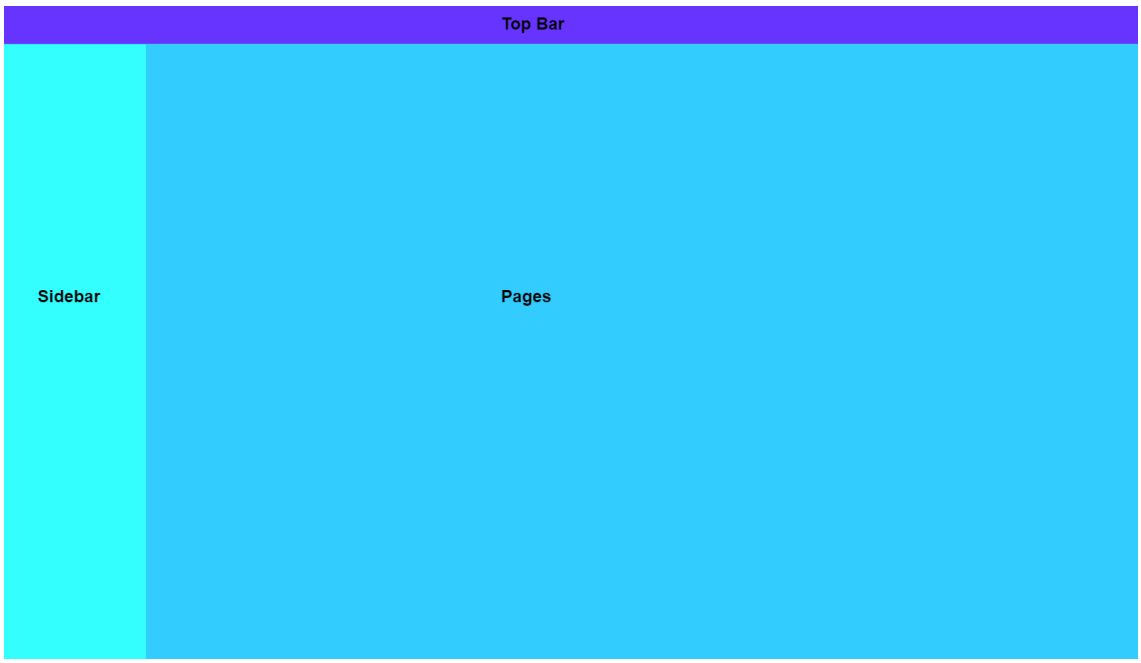
https://[Dominio/IP]:[Puerto]/#/login

https://[Dominio/IP]:[Puerto]/#/register





Al autenticarse se podrá acceder a la aplicación. Una vez dentro toda la aplicación tendrá la misma estructura. La barra superior y lateral serán compartidas por toda la aplicación (menos por el login y el registro). Donde pone pages en la siguiente imagen será la parte variable que irá cambiando dependiendo de la ruta donde estemos.



En la pagina principal accederemos a la ruta

https://[Dominio/IP]:[Puerto]/#/dashboard

Y desde esta ruta ya podremos acceder al resto de la aplicación con un solo click, esto será posible al sidebar y al topbar, que están fijas en todas las rutas hijas, sólo cambiará el apartado de pages.

A continuación, se mostrarán los componentes con sus respectivas rutas.

https://[Dominio/IP]:[Puerto]/#/ verporcat/[idCategoria]
Ver por Categoria

https://[Dominio/IP]:[Puerto]/#/productoUser/[idProducto]
Ver producto

https://[Dominio/IP]:[Puerto]/#/ busqueda/[campoDeBusqueda]
Buscar productos

https://[Dominio/IP]:[Puerto]/#/ usuarios
Ver usuarios

https://[Dominio/IP]:[Puerto]/#/productos
Ver productos

https://[Dominio/IP]:[Puerto]/#/ categorias
Ver categorias

https://[Dominio/IP]:[Puerto]/ #/ producto/[idProducto]
Editar productos

https://[Dominio/IP]:[Puerto]/ #/
Dashboard

En el siguiente apartado, se explicará más detalladamente que hace cada ruta. Desde la siguiente ruta podemos ver modificar el perfil y ver estadísticas sobre nuestros gustos.

https://[Dominio/IP]:[Puerto]/#/perfil

Podremos ver posibles productos deseados

https://[Dominio/IP]:[Puerto]/#/recomendaciones

Gestionar nuestras criticas.

https://[Dominio/IP]:[Puerto]/#/criticas

Gestionar nuestros deseos

https://[Dominio/IP]:[Puerto]/#/deseos

Cambiar el estilo de la aplicación

https://[Dominio/IP]:[Puerto]/#/account-settings

Ver productos por categoría

https://[Dominio/IP]:[Puerto]/#/ verporcat/[idCategoria]

Ver un producto

https://[Dominio/IP]:[Puerto]/#/productoUser/[idProducto]

Busquedas

https://[Dominio/IP]:[Puerto]/#/ busqueda/[campoDeBusqueda]

A esta parte solo se podrá acceder si se tienen suficientes privilegios y nos servirá para gestionar toda la aplicación

Gestionar usuarios

https://[Dominio/IP]:[Puerto]/#/ usuarios

Gestionar productos

https://[Dominio/IP]:[Puerto]/#/productos

Gestionar categorías

https://[Dominio/IP]:[Puerto]/#/categorias

Editar un producto

https://[Dominio/IP]:[Puerto]/#/ producto/[idProducto]

Un prototipo de baja fidelidad de nuestra página principal podría ser el siguiente:

Una vez ya desarrolladas las tres capas pasaremos a probar la seguridad.

4.3.1 Seguridad

La seguridad es algo fundamental en toda aplicación, este trabajo no es sobre ciberseguridad, pero si es necesario comentar algunas de las medidas básicas que se han tomado para asegurar que los usuarios hagan un uso adecuado de la aplicación.

En primer lugar, nuestra aplicación dispondrá de un login, cuando un usuario desee acceder, se enviará una petición HTTPS al servidor, este consultara si el usuario está en base de datos. Si el usuario esta guardado el servidor devolverá un token válido para que pueda acceder a la aplicación.

Sin este token, todas las rutas de la aplicación quedarán desactivadas, esto será posible gracias a un sistema de seguridad de Angular llamado guards.
Además, todas las peticiones HTTPS al servidor necesitarán el token para funcionar, sin el token nada funciona y el único modo de obtenerlo es estando registrando en la base de datos para luego autentificarse mediante el login.

Por si no fuera poco, el token irá cifrado con un algoritmo SHA256 para que no pueda ser manipulado. Este algoritmo cifrará mediante una semilla que estará en la parte del código del servidor donde nadie puede ver el código.
De este modo tanto la parte del cliente como la del servidor llevan una capa de seguridad para que los usuarios no puedan hacer un uso incorrecto de la aplicación.
La autentificación por Google registra los datos en la base de datos y una vez el usuario registrado el servidor nos da el token de acceso.

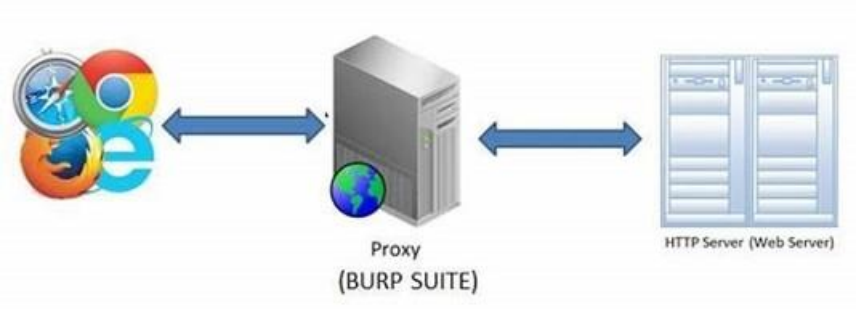
Sin embargo, el servidor ha sido configurado para que solo las peticiones autorizadas sean aceptadas, esto ha sido posible gracias a un mecanismo llamado Cors(Intercambio de Recursos de Origen Cruzado).

Para realizar el proyecto se han intentado usar las buenas prácticas que nos proporciona OWASP, que es una comunidad que nos provee información sobre seguridad en aplicaciones.

A pesar de todos los esfuerzos en el tema de la seguridad nunca hay suficiente.

Por lo tanto, se ha usado una herramienta para realizar el análisis acerca de la seguridad de las aplicaciones, dicha herramienta se llama Burp Suite.

La principal funcionalidad de esta herramienta es la de ponerse entre la parte del cliente y el servidor para manipular las peticiones.



Esta herramienta también es capaz de escanear toda nuestra web en búsqueda de posibles vulnerabilidades. Para ello, hace todas las peticiones posibles al servidor y recorre todas las páginas disponibles intentando hacer ataques conocidos.

Por ejemplo, nuestra aplicación no dispone de un captcha en el login, por lo tanto, nos aparecería una vulnerabilidad acerca del formulario, diciendo que puede ser explotado con fuerza bruta.

Además de esto, también nos permite hacer diversos ataques como XSS, inyecciones SQL, ataques de fuerza bruta o de diccionario, entre muchos otros. Esto es perfecto para hacer Pentesting a nuestra aplicación.

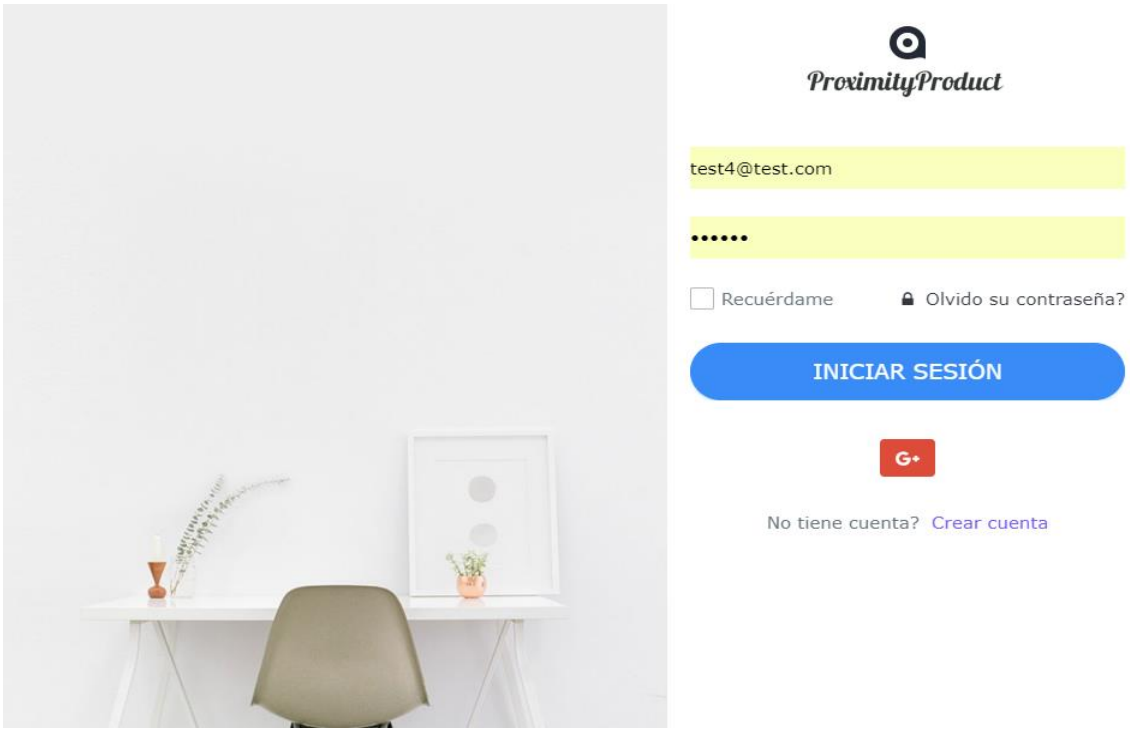
Esta aplicación también nos permite generar informes automáticamente, esto nunca va mal cuando terminas un desarrollo, así nos podemos hacer una idea de posibles fallas en nuestra aplicación.

Lo más correcto para proteger nuestra aplicación será la contratación de una empresa externa para que realice una auditoría, así tendremos una perspectiva diferente, ya que un experto habrá revisado la seguridad de nuestra aplicación.

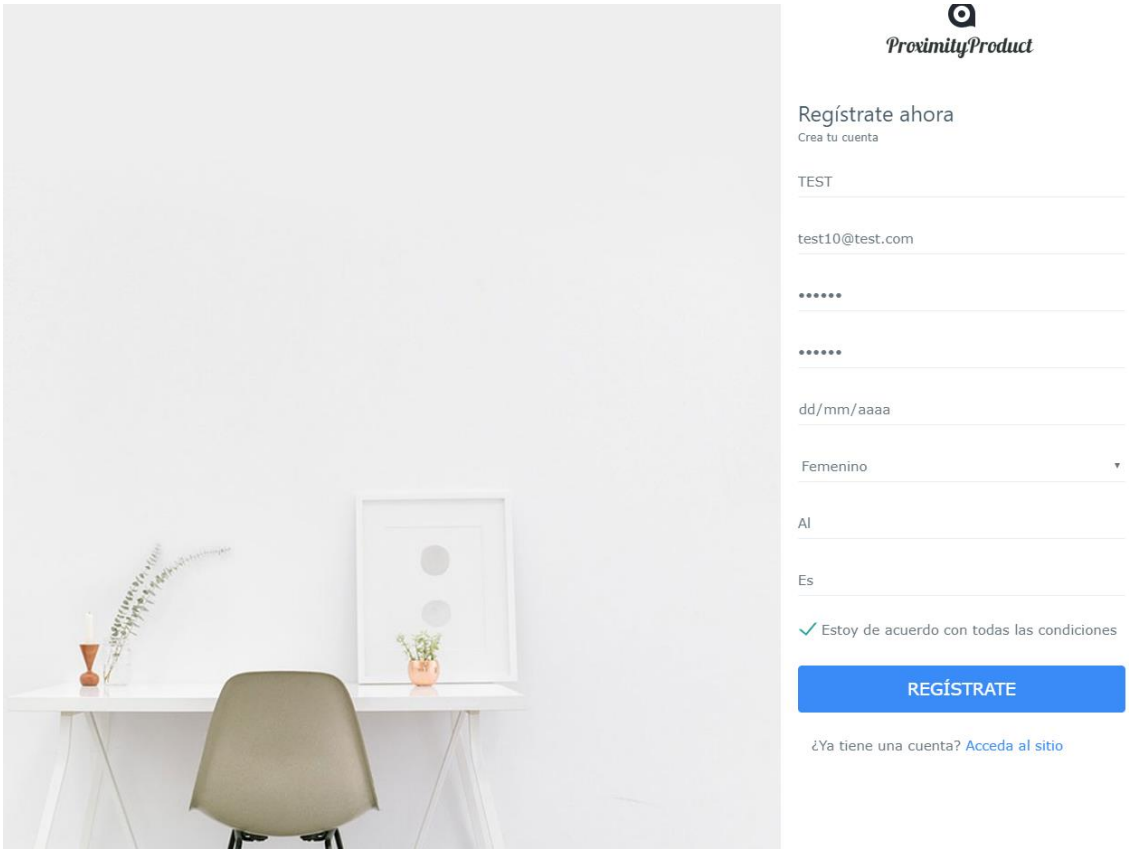
4. Resultado

Seguidamente se hará un tour por la aplicación para mostrar su contenido.

En la siguiente imagen se puede apreciar el login de la aplicación, cuyo funcionamiento ya se ha explicado en el apartado de seguridad.



Los usuarios deben tener la opción de poder registrarse en la aplicación.
Si un usuario no está registrado, se puede registrar desde esta pantalla. Todos los campos tienen validaciones para comprobar que los datos sean correctos desde la parte del cliente, de este modo no sobrecargar el servidor con peticiones de datos incorrectos, si por algún modo se manipularan los datos desde la parte del cliente el servidor también los validaría para estar seguros que son correctos.
Cuando se registren los datos de un usuario, estos quedarán guardados en base de datos y ya se podrá loguear.



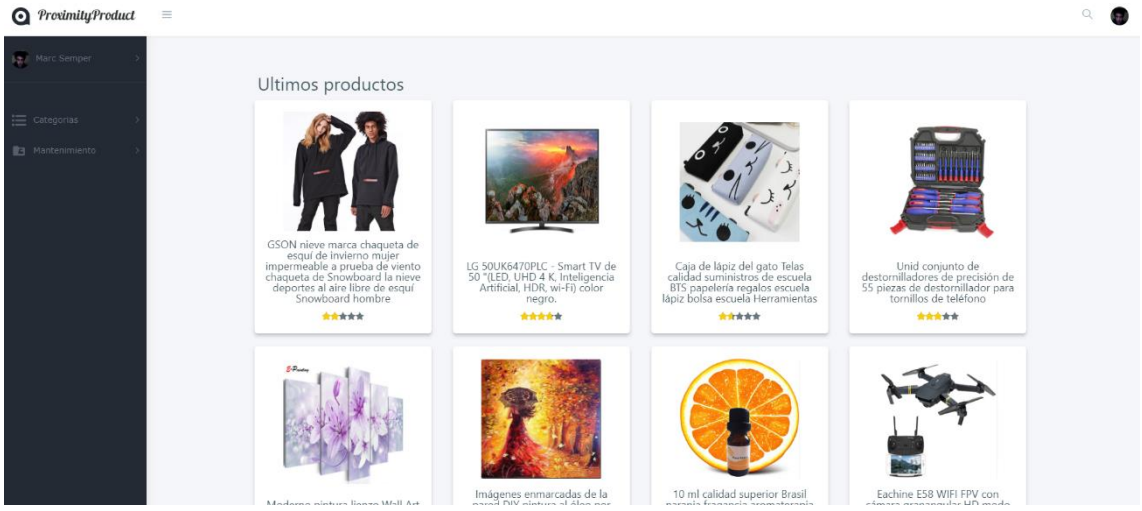
Cuando una aplicación de angular se inicia se cargan todos los componentes, esto puede hacer que la aplicación pierda rendimiento, ya que la aplicación se descargaría al completo, aunque el usuario no pase el login. Por lo tanto, se ha usado una técnica llamada Lazy Loading, esta técnica carga solo los componentes que necesitemos al inicio de nuestra aplicación.

Por lo tanto, las páginas del registro como la del login se descargan en primer lugar y sólo cuando un usuario se loguee se descarga el resto de la aplicación.

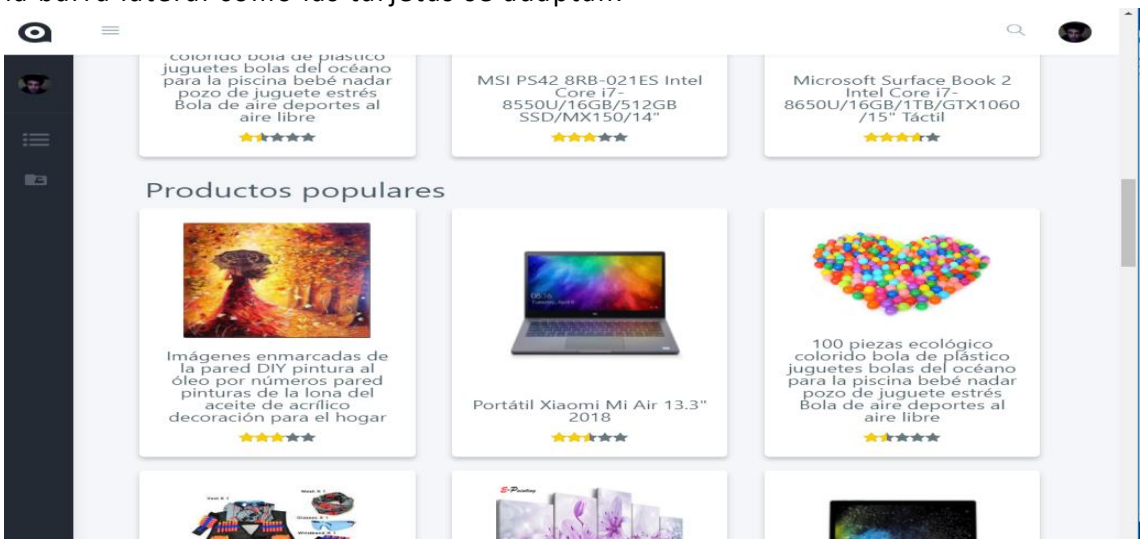
Una vez el usuario se haya logueado, ya tendrá acceso a la aplicación. En la primera pantalla tendremos indicadores útiles, como por ejemplo los últimos productos, los productos más populares o los mejores valorados.

Estos indicadores mostrarán productos desde la base de datos, el servidor está programado para que sepa que productos son los más nuevos, los mejores valorados… etc.
La parte del cliente le enviará al servidor una petición HTTPS y el servidor le responderá con un JSON con los valores deseados.
Esta dinámica será igual para toda la aplicación.

También se puede apreciar en la imagen que tendremos una barra superior y otra lateral, estas serán compartidas para toda la aplicación.

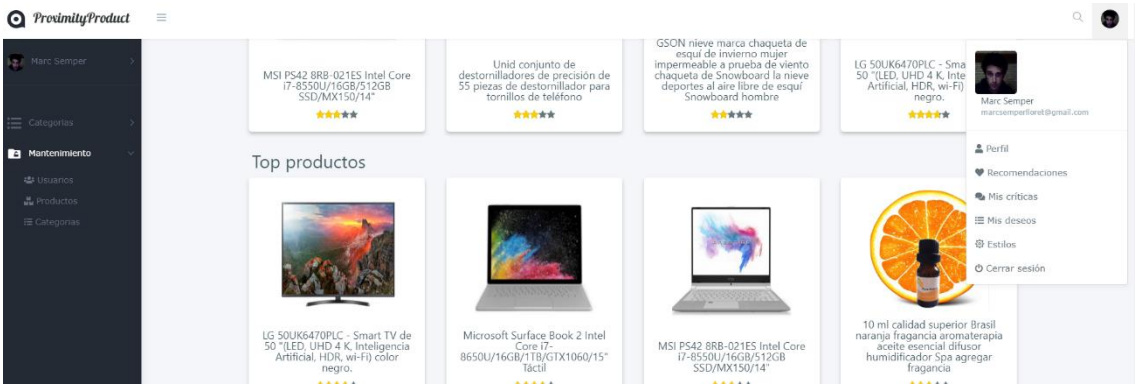


En la siguiente imagen se puede observar como la aplicación es responsive, tanto la barra lateral como las tarjetas se adaptan.

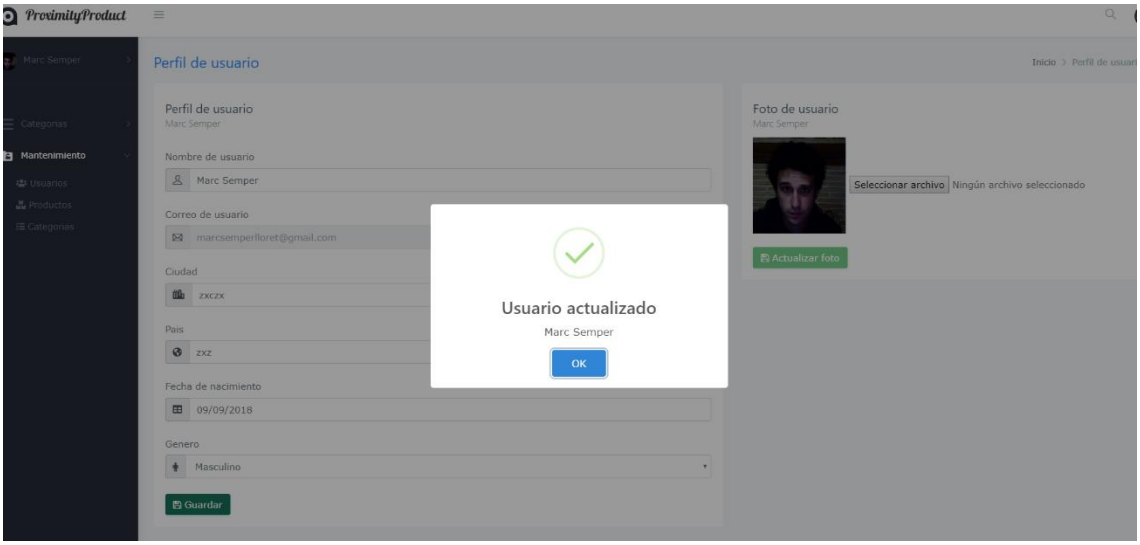


En la barra lateral se pueden desplegar las opciones del menú, este menú es cargado dinámicamente desde el servidor, por lo tanto, puede ser modificado fácilmente sin modificar código. En la parte superior derecha tenemos un desplegable donde se muestran las opciones del usuario.

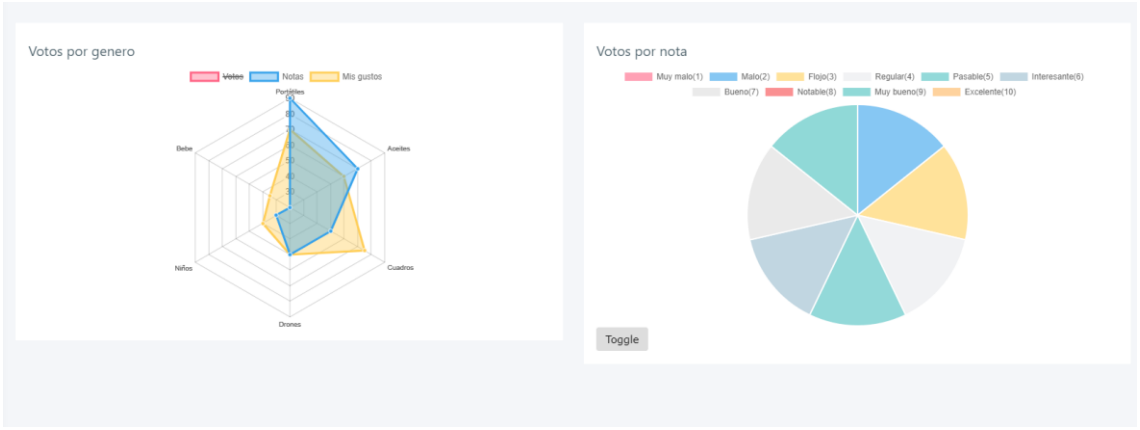
En la imagen se puede observar como en la barra lateral izquierda tenemos desplegado un menú de mantenimiento, este menú sólo estará disponible según los privilegios del usuario, el servidor no lo enviará a la parte del cliente si el usuario no es administrador. Tampoco se pondrá poner la ruta directamente en la barra de navegación ya que los guards de angular las desactivaran si no se tienen suficientes privilegios.



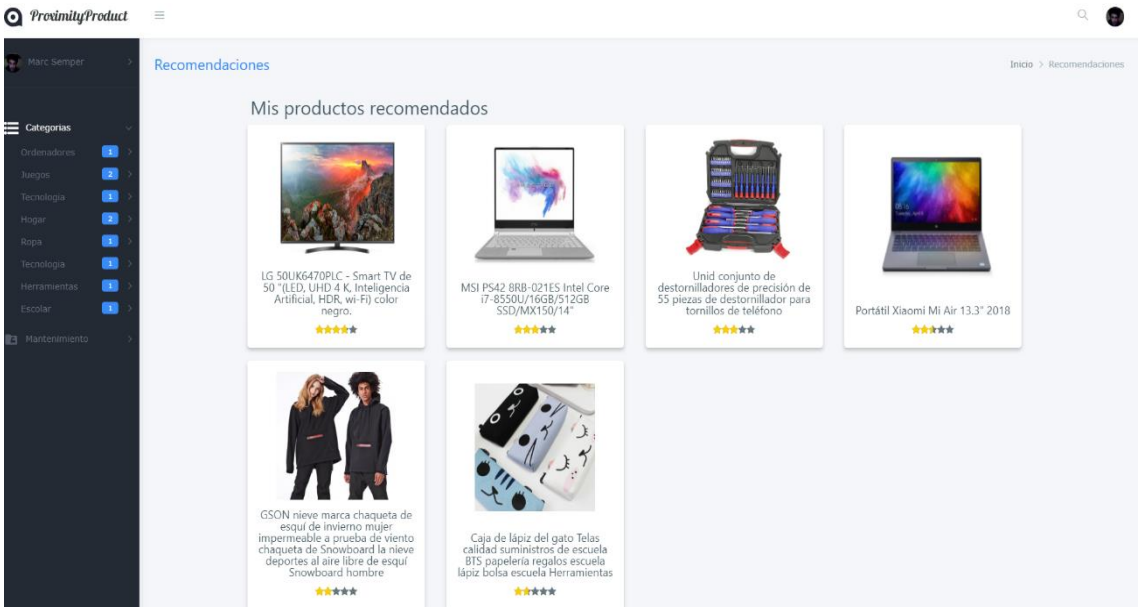
Si el usuario accede a su perfil podrá cambiar sus datos personales. En la imagen se aprecia el feedback que recibe cuando sus datos son modificados.



Además, podrá ver gráficos con algunos indicadores de interés, las gráficas son totalmente dinámicas. Se modifican en tiempo real conforme el usuario hace nuevas cítricas.

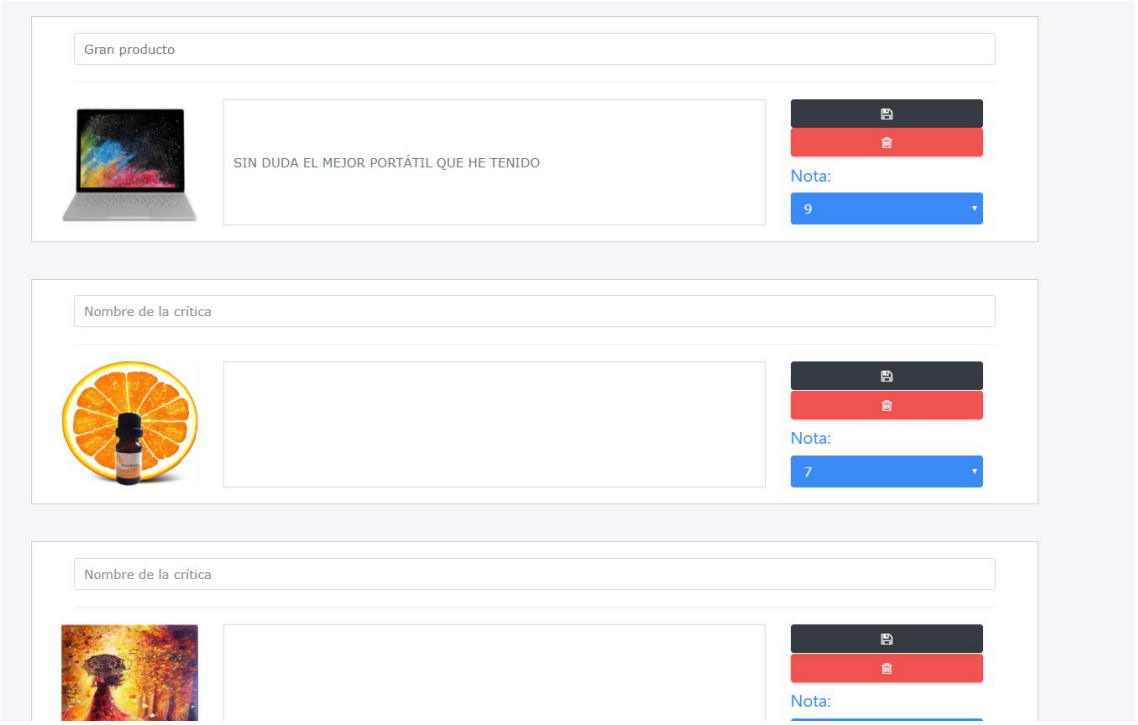


En la imagen que aparece a continuación se puede apreciar en el menú izquierdo las categorías, estas se irán modificando dinámicamente para todos los usuarios dinámicamente si un administrador las elimina o añade nuevas.

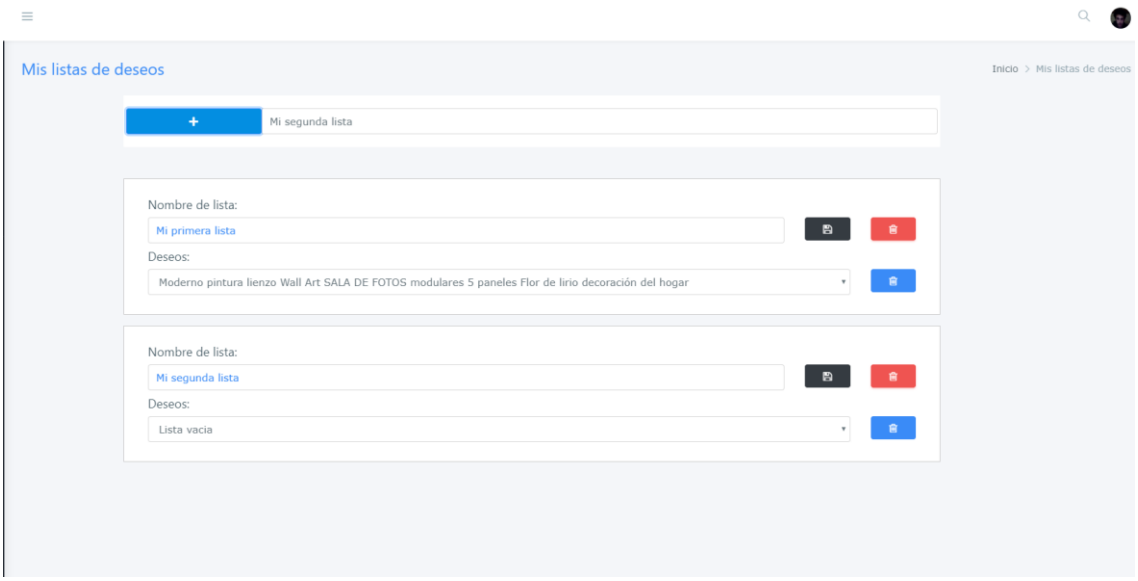


Para las recomendaciones se ha usado el algoritmo k-nearest neighbors, este ya ha sido explicado anteriormente.

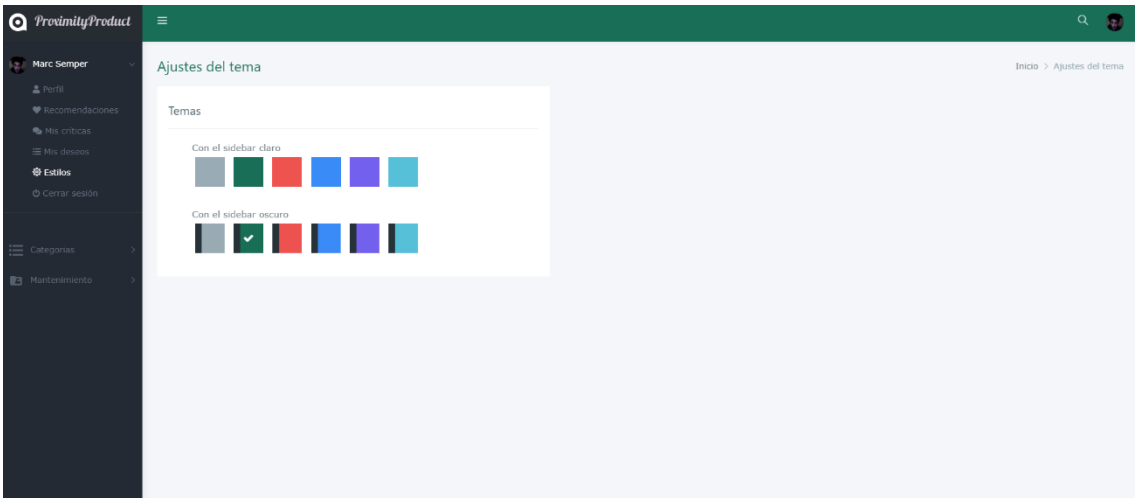
Los usuarios podrán gestionar sus criticas desde el apartado Mis críticas, donde las podrán modificar o eliminar, además desde aquí podrán acceder al producto por si quieren modificarlo.



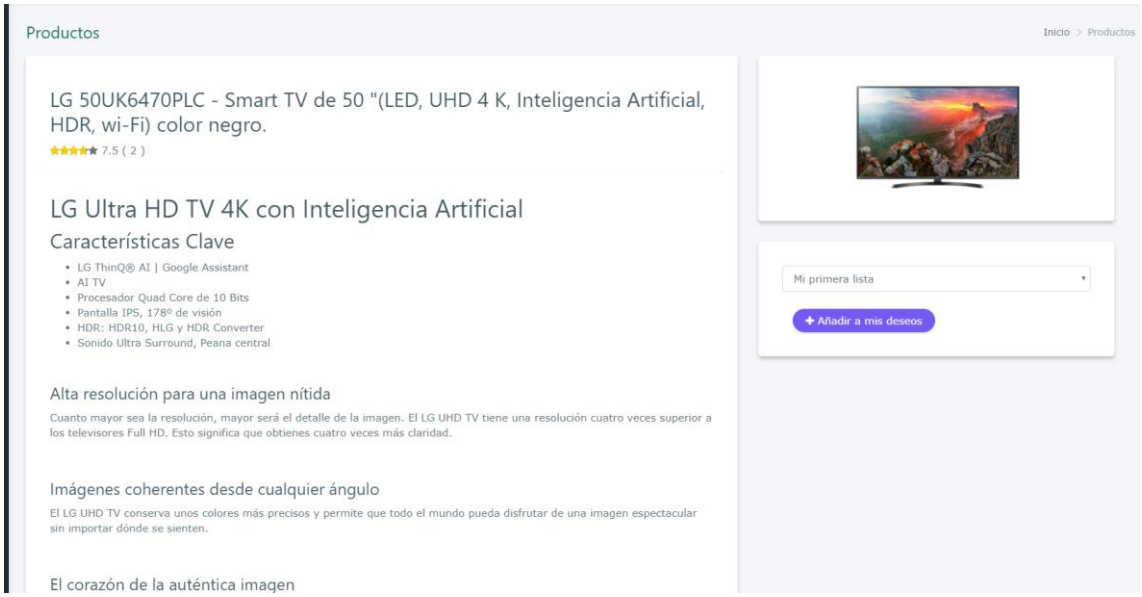
Otra opción disponible para los usuarios es la de crear sus propias listas de deseos, donde podrán añadir los productos que les gusten.



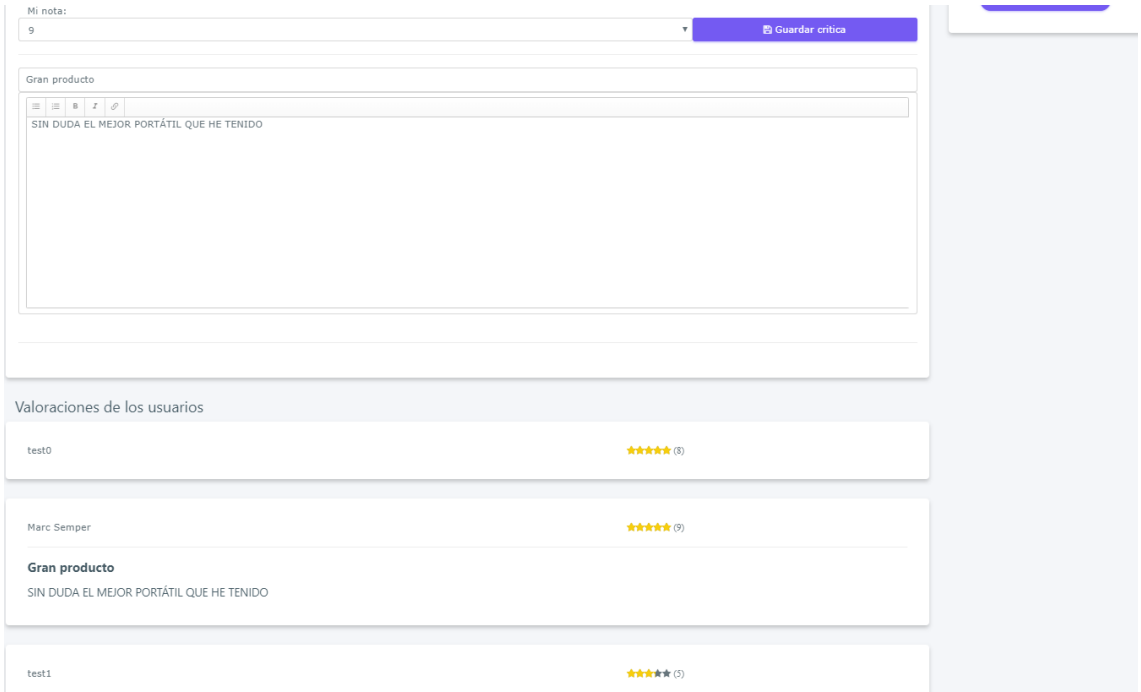
También tendrán la opción de cambiar el estilo de la aplicación, si nos fijamos la barra superior ha cambiado de color al verde.



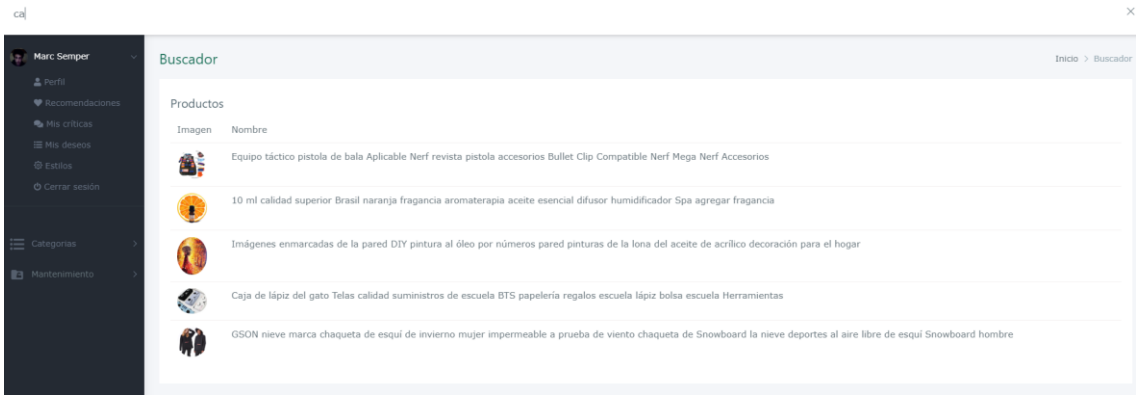
Si entramos a un producto podremos ver el título, su nota, número de votos y la descripción del producto.



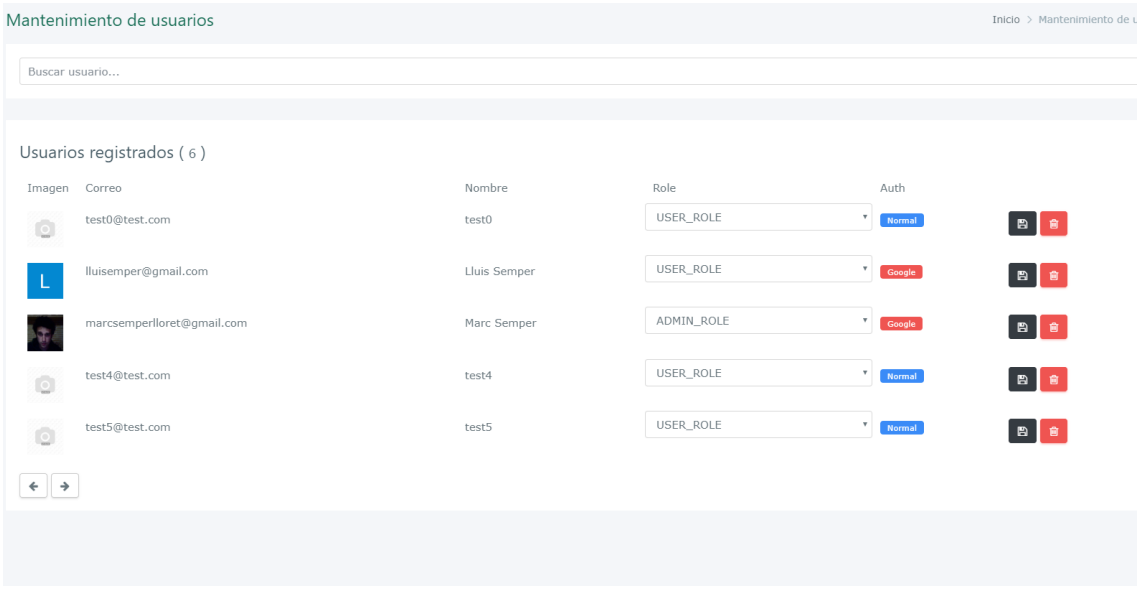
Y cómo es obvio también tendremos la opción de puntuarlo y añadir una crítica. En la parte inferior nos irán apareciendo dinámicamente las criticas de otros usuarios. Si una crítica no tiene nombre o descripción solo aparecerá el nombre del usuario que ha votado y la nota.



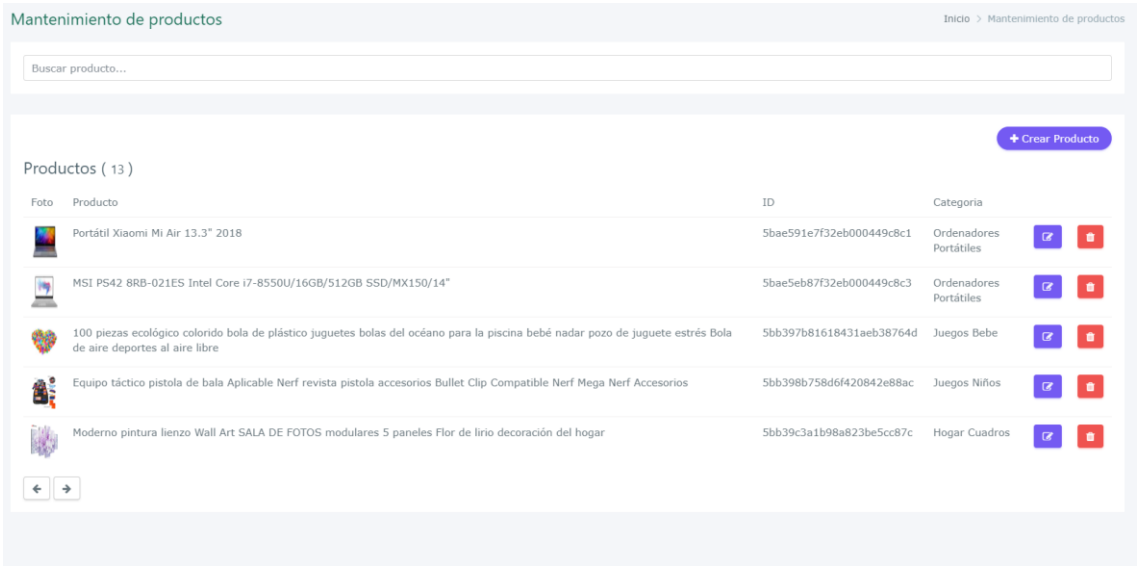
En la parte superior dispondremos de un buscador por si queremos buscar algún producto específico.



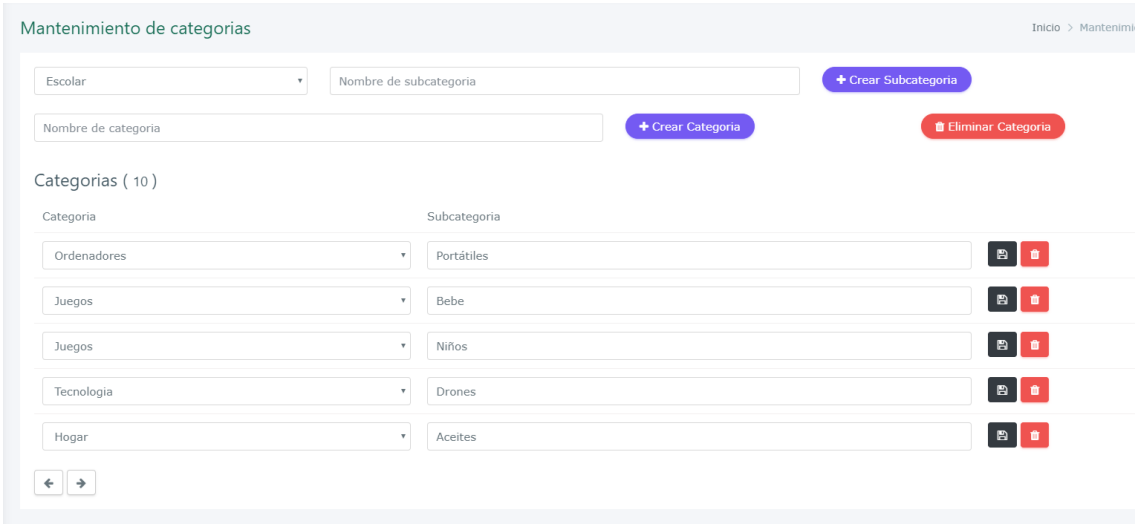
Los administradores tendrán la opción de modificar o borrar usuarios, podrán hasta modificar sus privilegios. También dispondrán la opción de buscar un usuario por mail.



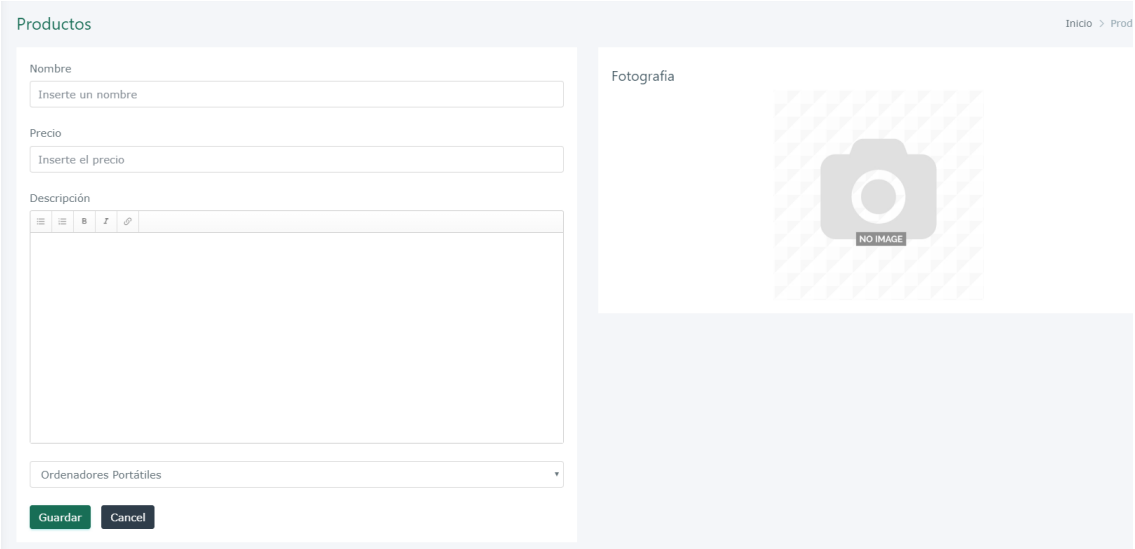
Además de los usuarios podrán hacer lo mismo con los productos, aquí también se podrán crear nuevos productos.



Y lo mismo con las categorías y las subcategorías.



Cuando se desee crear un nuevo producto, la descripción aparecerá tal cual en la visualización de productos gracias al uso de un editor ngxwig que nos traduce nuestro texto a código HTML. Este se guarda en base de datos para que luego se muestre la descripción del producto adecuadamente.



Y esto es todo lo que se ha hecho, para comprobar errores se han pasado algunas pruebas automáticas.

Las primeras que se han realizado son las pruebas unitarias, estas comprueban que el funcionamiento de los componentes de Angular sea el esperado. Es decir, comprobamos que cada parte aislada de la aplicación funcione bien.
Seguidamente se han pasado las pruebas de integración, estas últimas comprueban que los componentes aislados tengan el comportamiento esperado al juntarse.

Si se quiere comprobar su funcionamiento con más detalle, la aplicación está desplegada en el siguiente enlace:

https://marsemll.upv.edu.es

5. Conclusiones y trabajos futuros

Muchos alumnos sentimos que cuando terminamos la universidad no podemos aportar nada nuevo al mercado laboral, no nos sentimos capaces de crear nada y por ello tenemos la autoestima bastante baja.

Este trabajo ha sido importante para mejorar mis conocimientos como desarrollador de aplicaciones Web y subir mi autoestima en el ámbito profesional. Personalmente me ha aportado confianza en mí mismo, ya que, me he demostrado a mí mismo que soy capaz de desarrollar aplicaciones Web bastante grandes con cierta facilidad, hace unos meses pensé que sería algo imposible.

Pienso que he creado algo real que me puede dar proyección en el mercado laboral. Cuando realizas una entrevista de trabajo siempre preguntan si les puedes enseñar algo que hayas hecho tú. Y yo ya tengo algo que pueda enseñar.

Realmente antes de empezar este proyecto, sólo tenía nociones básicas sobre desarrollar aplicaciones Web. En la universidad había visto algunas cosas de diferentes tecnologías, muy genéricas y teóricas. Pienso que la universidad no me ha otorgado conocimientos específicos, pero si ha moldeado mi forma de pensar y gracias a ello tengo cierta facilidad para aprender cualquier cosa.

Las tecnologías que he usado me han parecido increíbles, después de haber trabajado principalmente con java todo el entorno JavaScript me gusta mucho más, no quiero decir que java sea algo malo, pero los desarrollos con java me parecen más lentos y ceremoniosos. Con JavaScript todo me resulta fácil, práctico y rápido.

El tema del Scraping no lo he realizado porque hacer un programa que recoja los productos de cada tienda es mucho trabajo, perfectamente cada programa de estos podría ser otro TFG. Por este motivo no lo he realizado, creo que hubiera sido demasiado. Eso si se ha probado con éxito y funciona bien, así que queda pendiente.

En cuanto a los trabajos futuros aún queda mucho por hacer, la aplicación funciona, pero aún tiene bugs y temas de usabilidad, seguridad…etc. por resolver. Por ejemplo, en algunos lugares cuando el usuario realiza una acción no recibe feedback y por esto no puede saber si su acción ha tenido éxito o no. Por poner otro ejemplo, un bot podría registrar automáticamente un montón de usuarios y llenar la base de datos en poco tiempo, ya que no hay confirmación por email ni captcha.

Bibliografía

DAFO, H. (25 de Noviembre de 2018). Obtenido de https://dafo.ipyme.org/

Documentación oficial de Angular 6. (s.f.). Obtenido de https://angular.io/docs

Documentación oficial de AngularCLI. (s.f.). Obtenido de https://github.com/angular/angular-cli/wiki

Documentación oficial de Boostrap. (s.f.). Obtenido de https://getbootstrap.com/docs/4.1/getting-started/introduction/

Documentación oficial de Express. (s.f.). Obtenido de https://expressjs.com/es/guide/routing.html

Documentación oficial de Github. (s.f.). Obtenido de https://github.com/features#documentation

Documentación oficial de jQuery. (s.f.). Obtenido de https://api.jquery.com/

Documentación oficial de JWT. (s.f.). Obtenido de https://jwt.io/

Documentación oficial de MongoDB. (s.f.). Obtenido de https://docs.mongodb.com/?_ga=2.176098914.1140191173.1543172475-1363611262.1543172475

Documentación oficial de Mongoose. (s.f.). Obtenido de https://mongoosejs.com/docs/index.html

Documentación oficial de NodeJS. (s.f.). Obtenido de https://nodejs.org/en/docs/

Documentación oficial de Postman. (s.f.). Obtenido de https://learning.getpostman.com/docs/postman/launching_postman/installation_and_updates/

Documentación oficial de TypeScript. (s.f.). Obtenido de https://www.typescriptlang.org/docs/home.html

FERREIRA-HERRERA, D. (25 de Noviembre de 2018). *El modelo CANVAS en la formulación de proyectos*. Obtenido de https://revistas.ucc.edu.co/index.php/co/article/view/1252

Christian Mainka, V. M. (2015). *Automatic recognition, processing and attacking of single sign-on protocols with burp suite*. Obtenido de https://dl.gi.de/handle/20.500.12116/1977

Jorge Enrique Rodríguez Rodríguez, E. A. (2007). *Clasifi cación de datos usando el método k-nn*. Obtenido de

http://revistas.udistrital.edu.co/ojs/index.php/vinculos/article/view/4111/5778

Juan José Mora Flórez, G. M. (2008). *Evaluación del clasificador basado en los "k" vecinos más cercanos para la localización de la zona en falla en los sistemas de potencia*. Obtenido de https://dialnet.unirioja.es/servlet/articulo?codigo=2945845

Parra, D. (9 de Febrero de 2015). *Filtrado Basado en Contenido Filtrado Basado en Contenido*. Obtenido de http://dparra.sitios.ing.uc.cl/classes/recsys-2015-2/clase6_contentBased_1.pdf

7. Anexos

Hay que mencionar que se han usado componentes gráficos de internet creados por la comunidad, actualmente se pueden encontrar en git-hub. Algunos de ellos son los siguientes:

- ng2-charts para las gráficas.
- Font Awesome para los iconos.
- Animate.css para las animaciones de desplazamientos.
- SweetAlert 2 para los modales de feedback.
- ngx-wig para el editor de texto.

7.1 Entorno de trabajo

7.1.1 Visual Studio Code

Es un famoso editor de código que nos brinda soporte para JavaScript, TypeScript, Node, depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

Plugins y snippets de Visual Studio Code usados:
• Angular 2 TypeScript Emmet
• Angular 6 Snippets – TypeScript, Html, Angular Material…
• Angular Language Service
• Angular v5 Snippets
• Angular2-inline
• Bootstrap 4& FontAwesome snippets
• HTML CSS Support
• JavaScript(ES6) code snippets
• JS-CSS-HTML Formatter
• JSHint
• Prettier – Code Formatter
• Terminal
• TSLint
• TypeScript Hero
• TypeScript Importer

7.1.2 Robo 3T

Es la GUI ligera para MongoDB. Desde aquí podremos gestionar nuestra base de datos de un modo sencillo.

7.1.3 Postman

Es una herramienta para gestionar nuestras peticiones HHTPS a nuestra API REST, además nos genera la documentación instantáneamente por si trabajamos en equipo y los encargados de la GUI necesitan saber cómo funciona la API REST.

7.2 Plan de negocio

En este apartado se realizará un análisis más extendido, para ello abordaremos cuestiones de un plan de negocio, en primer lugar, crearemos un modelo Canvas.

<table>
<tr>
<td rowspan="2">Problemas

Crear marca será un problema ya que hay muchas aplicaciones que hacen algo parecido y el usuario puede creer que lo que ofrecemos no es algo nuevo.

Recopilar, almacenar y gestionar tantos datos, puede resultar complicado y caro.</td>
<td>Solución

La actividad clave será la de darnos a conocer y hacer marca. Si consultar en nuestra aplicación fuera una costumbre para los consumidores, estos comprarían dependiendo de las valoraciones en nuestra aplicación.</td>
<td rowspan="2">Proposición de valor

Nuestro servicio otorgará una opinión imparcial sobre cualquier producto.

Otorgará comodidad al cliente por centralizar todos los productos de Internet en un sólo sito</td>
<td>Ventaja competitiva

Nuestra principal ventaja será la gran cantidad de productos que dispondremos.</td>
<td rowspan="2">Clientes

Nuestros clientes serán todos aquellos consumidores que quieran tener una opinión imparcial sobre un producto, o aquellos que no tengan claro que comprar.

Principalmente será gente joven que se maneje con las nuevas tecnologías.</td>
</tr>
<tr>
<td>Métricas

Los recursos claves serán disponer de una aplicación que pueda almacenar grandes cantidades de datos con todo lo que esto conlleva.</td>
<td>Canales

Nuestro servicio llegara al cliente mediante una aplicación informática</td>
</tr>
<tr>
<td colspan="3">Costes

Infraestructura para desplegar nuestra aplicación y guardar los datos.</td>
<td colspan="2">Ingresos

Basados en la publicidad.</td>
</tr>
</table>

En segundo lugar, trataremos todo lo que podría afectar al proyecto.

Algo bastante importante son los potenciales usuarios, vamos a plantear diversos escenarios acerca de porque estos podrían estar interesados en nuestra aplicación:

- Usuarios que estén comprando y quieran una segunda opinión porque no saben si un producto es bueno o malo. Podrían recurrir a nosotros por nuestra imparcialidad o por tener un número mayor de críticas.

- Usuarios que no saben que comprar. Estos podrían ir a nuestra aplicación a echar un vistazo, ya que en ella habrá más productos que en todas las demás tiendas, por lo tanto, se podrán hacer mejores recomendaciones.

- Un producto que no tenga opiniones suele producir dudas sobre su calidad, en nuestra aplicación habrá más críticas que en cualquier tienda, los usuarios al comprar un producto sin críticas en una tienda especifican puede que recurran a nosotros para consultar sobre su calidad.

- Cuando la aplicación ya tenga cierta imagen de marca los usuarios pueden buscar la valoración acerca de un producto antes de comprar cualquier cosa.

Otra parte importante antes de pensar en un negocio es estudiar la competencia, esto nos dará una idea si nuestro proyecto puede ser viable o no.

En cuanto a los competidores no tendremos un competidor real, todas las tiendas serán competidoras en cierto modo, todas tienen críticas y recomiendan productos. Pero para ellas no seremos su competencia, ya que no venderemos nada. Por lo único que nos deberemos de preocupar es por tener más productos y críticas que cualquier otra aplicación.

Una vez que tenemos una idea sobre que clientes y competidores pasaremos a plantear la situación económica.

La viabilidad económica del producto dependerá del tiempo en que se consiga una buena imagen de marca, esto es debido a que los costes de almacenamiento serán elevados.
Solamente en Amazon hay sobre 70.000.000 millones de productos (Esto se puede saber haciendo una búsqueda que no coincida con ningún resultado). Debemos tener en cuenta que no sólo estarán disponibles los productos de Amazon sino de todas las tiendas.

Actualmente un producto ocupa aproximadamente 12kB en base de datos y 50kB en el servidor (La imagen). Si multiplicamos los productos de Amazon por (12+50)72KB tendremos aproximadamente 5 Terabytes. Y esto sólo de Amazon.

Por lo tanto, los costes en cuanto almacenamiento serán bastante altos. Los hostings no suelen tener soluciones estándar para tantas necesidades de almacenamiento y las opciones personalizadas son muy caras. Otra opción será montar nuestro propio servidor. Esto supondrá una mayor inversión inicial, conocimiento técnico y algunos riesgos, pero con el tiempo podremos recuperar nuestra inversión. Obviamente esto se debe estudiar al detalle.

Aunque los costes de almacenamiento sean elevados, si conseguimos hacer que todos los consumidores consulten nuestra aplicación antes de que vayan a comprar cualquier cosa, podríamos disponer de mucho tráfico. Esto solamente ya nos daría beneficios, tanto por tener la opción de poner publicidad, o poner una cuota a las marcas que quieran destacar sus nuevos productos, entre otros. Hay que tener en cuenta, que todo esto no es posible en etapas iniciales, por lo tanto, crear marca y darnos a conocer será muy importante para hacer que el proyecto sea viable.

Una aplicación de este modo podría ser muy útil para el motor de búsqueda de Google, ya que, al buscar un producto, automáticamente podrían salir indicadores e incluso criticas relevantes sobre el producto directamente en el buscador, de este modo un usuario solamente buscando un producto en Google tendría una opinión imparcial sobre un producto. También se podría añadir al asistente de Google(interfaz de voz de Google) entre otros.
Actualmente Google ya hace esto con páginas de críticas de películas como Filmaffinity ¿Si ya se hace con películas, por qué no se podría hacer con productos? Que una compañía como Google pudiera comprar también sería una opción.

Los temas demográficos y culturales no supondrían ningún problema. Hoy en día casi todo el mundo usa la tecnología diariamente.
Con las nuevas generaciones la tecnología cada vez es más usada y tampoco depende demasiada de la cultura de cada país. Es algo que se está imponiendo a nivel mundial. Por lo tanto, esto será perfecto para nosotros.

En cuanto a lo tecnológico es un proyecto muy grande en el que se requieren conocimientos específicos sobre muchas cosas. Además, es una tarea con retos técnicos y se necesita desarrollar muchas partes diferentes.
Por ejemplo, extraer todos los productos automáticamente de todas las tiendas más grandes de internet es mucho trabajo, ya que, hay muchas tiendas y cada una tiene una estructura diferente, por lo cual deberemos hacer un programa personalizado por cada una que extraiga sus productos automáticamente.

También hay que pensar que al extraer automáticamente productos de tiendas diferentes se capturarán productos duplicados, es decir, productos que estén en las dos tiendas, obviamente si capturamos un duplicado se deberá fusionar. Es muy probable que su duplicado tenga un nombre, imagen y descripción diferente, por lo tanto, no dispondremos de ninguna referencia para saber que es un duplicado, además la diversidad de los datos será enorme, resolver esto parece trivial, pero no lo es. Y estos solo son algunos ejemplos de problemas técnicos a resolver, posiblemente todo esto sea demasiado trabajo para solo un estudiante.

Los temas políticos no nos afectan demasiado. En cuanto a temas legales si podríamos tener algún problema, normalmente a las tiendas no les gusta que un bot extraiga todos sus productos, cabe decir que esto no es ilegal, lo que haremos será como copiar y pegar texto del nombre, la descripción del producto y las críticas de sus usuarios o hacer una captura de pantalla de una imagen, pero claro, automáticamente. De este modo el proceso de rellenar la aplicación será más rápido. Aunque no sea ilegal, las compañías tienen derecho a bloquearnos si nos detectan, esto supondría un problema técnico más. Claramente es un tema que se debe estudiar al detalle.

Para organizar ideas y planificar la manera más adecuada de proceder crearemos un análisis DAFO.

Un análisis DAFO es estudio de la situación de un proyecto (u otro, empresa... etc.) donde se analizan sus propiedades internas y externas.

Debilidades	Amenazas
Falta de mano de obra, ya que es un proyecto muy grande para una sola persona. Falta de capital. Falta de infraestructura.	El tiempo, cualquier otro podría hacer algo similar y adelantarse. Los costes de almacenamiento hacen que la viabilidad del proyecto sea muy mala en las etapas tempranas. Los usuarios pueden pensar que nuestro producto no es algo nuevo.
Fortalezas	Oportunidades
Habilidades de desarrollo y conocimientos técnicos sobre cómo hacerlo.	No hay competencia directa. No hace falta invertir mientras se esté en desarrollo.

La Misión deberá ser definida para tener una idea clara y sencilla de que queremos ser, la principal finalidad de nuestro proyecto es el de proporcionar seguridad y confianza al comprar un producto, a la vez que guiamos a los usuarios entre la inmensa diversidad que existe entre dichos productos.

Por lo tanto, nuestra visión podría ser la siguiente:
Proporcionamos seguridad y confianza en la compra de su producto a la vez que los guiamos entre la inmensa diversidad que existe entre dichos productos.

La visión también deberá ser definida, de este modo tendremos claro a donde queremos llegar, en un futuro lo ideal sería que consultar en nuestra aplicación fuera una rutina para todos los consumidores del mundo.

Seguidamente pasaremos formular las estrategias que nos permitan aprovechar las oportunidades y nos conviertan las amenazas en oportunidades.

Estrategias de Supervivencia (debilidades + amenazas)	*Estrategias Adaptativas (debilidades + oportunidades)*
Buscar otros alumnos que quieran colaborar en el proyecto, esto reduciría el tiempo de desarrollo y el riesgo de que alguien se adelante en el desarrollo. Estos alumnos deben ser útiles, como programadores que ayuden con el web scraping, administradores de sistemas para crear una infraestructura óptima o administrativos con conocimientos de marketing que nos puedan ayudar a conseguir una buena imagen de marca, entre otros. Buscar inversores, nos proporcionaría capital y con esto podríamos conseguir la infraestructura necesaria.	Si conseguimos más mano de obra y capital podemos crear un producto excelente, además de esto si aprovechamos que no hay competencia puede que obtengamos un buen número de usuarios. El no tener dinero no supondrá un problema mientras no estemos en producción ya que en desarrollo no tendremos costes de hosting por trabajar en local.
Estrategias Defensivas (fortalezas + amenazas)	*Estrategias Ofensivas (fortalezas + oportunidades)*
Deberemos usar nuestras habilidades para realizar el proyecto lo más rápido posible y que otra empresa no se nos adelante.	Si con nuestras habilidades técnicas desarrollamos un buen proyecto podríamos tener mucho éxito por la falta de competencia directa.

Una vez hecho esto tenemos un análisis general de cómo está la situación y que decisiones debemos tomar para mejorar nuestro propósito.