Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Aplicación web para ayudar al alumno

Web application to support the student

Omar Patricio Pérez Znakar

La Laguna, 8 de junio de 2019

D. **Vicente José Blanco Pérez**, con N.I.F. 42171808C profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"*Aplicación web para ayudar al alumno*"

ha sido realizada bajo su dirección por D. **Omar Patricio Pérez Znakar**, con N.I.F. 79062976Q.

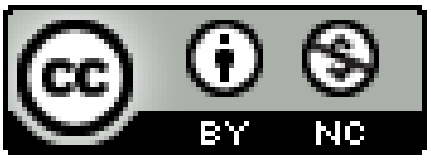
Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 8 de junio de 2019

Agradecimientos

Para realizar tanto este proyecto como otros a largo de la carrera me ha sido indispensable la ayuda y el apoyo de ciertas personas que me han acompañado a lo largo de este recorrido. Es por esto, que en este momento le dedico a esas personas mis logros durante el proceso.

En primer lugar, debo mencionar a mi tutor del trabajo de fin de grado, Don Vicente José Blanco Pérez, quién me guió durante el proyecto, aportando sus conocimientos y consejos de la mejor manera posible, junto con una gran paciencia y disponibilidad, lo cual resultó imprescindible en muchos momentos durante este y, al cual estoy totalmente agradecido por haberme permitido elaborar en su compañía este trabajo. Asimismo, agradecer a todos los integrantes del proceso de formación en el grado, así como todas aquellas personas que conforman parte de la comunidad universitaria. En segundo lugar, agradecer a mis padres, quienes me han brindado la oportunidad de poder estar en esta carrera y realizar este trabajo para labrar un futuro orientado hacia mis deseos y metas personales. Y, por último, a mi novia, que resultó un gran apoyo emocional para la ejecución del proyecto.

Licencia



Resumen

A lo largo del paso por la vida académica es necesario recurrir a ayuda externa de personas cualificada en determinadas áreas. Sin embargo, la obtención de esta ayuda a veces resulta bastante complicada. A fin de resolver esta problemática cuestión, se ha hecho la propuesta de creación de este proyecto.

El TFG propuesto consiste en el desarrollo de una aplicación web basada en la idea de ayudar al alumnado de cualquier ámbito académico, conseguido a través de la oferta y demanda de recursos destinados al estudio. Definimos estos recursos como:

- *Clases particulares: existencia de anuncios de carácter académico, en donde cualquier usuario puede ofertar clases particulares de ciertas asignaturas. No obstante, están enfocados a aquellos alumnos que hallan superado ese nivel y quieran obtener beneficio económico.*
- *Apuntes: anuncios sobre la venta de apuntes o libros de índole académica.*
- *Noticias: artículos sobre novedades en el mundo de la enseñanza.*

Asimismo, para su desarrollo se hará uso de las últimas tecnologías basándose en el stack MEVN, el cual se caracteriza, principalmente, por el uso de componentes. A su vez, está compuesto por:

- *Mongodb: una base de datos que almacena todos los anuncios (clases particulares, apuntes y noticias) y la sesión del proyecto (datos del usuario).*
- *Express: framework que nos permite crear la API REST (intermediario entre cliente y servidor), que es la que contendrá las funciones principales (insertar, recoger, actualizar y eliminar) del backend del proyecto, tanto en la sesión como en los anuncios.*
- *Vue: framework de desarrollo web que nos posibilita el uso de JavaScript en prácticamente todo el proyecto (ayudado de HTML, CSS, entre otras).*
- *Node: entorno de ejecución del lado del servidor que nos permitirá mostrar el proyecto en el navegador.*

Junto a este, se ha utilizado el sistema de control de versiones Git, en concreto Github. Unida a la herramienta anterior, hemos hecho uso de la integración continua de diversas herramientas, donde cada una de ellas presenta una función determinada, como por ejemplo, aquellas que comprueban la ejecución óptima de las funcionalidades importantes (test backend o frontend) o de aquellas destinadas a la comprobación del código (SonarQube). De esta manera, si los resultados de este procedimiento son los esperados se realizaría un despliegue automatizado en la nube respaldado por Heroku.

Palabras clave: noticias, profesores, particulares, apuntes, libros, MEVN, Mongodb. Express, Vue, Node.

Abstract

During the academic life, sometimes it is necessary to ask for help from professionals that are qualified in an specific topic, but this can be quite difficult. To solve this matter, I propose to develop a project addressing this problem.

This Final Degree Project consists of developing a web application which focuses on the help to students of any academic field. This objective can achieved through the exchange of learning resources. We can define this resources as:

- *Individual lessons: the existence of ads related with several academic subjects. Users of the web application can offer lessons for any subject. Nevertheless, the teaching classes can only be offered by users that have overcome that current level and want to get a economic profit.*
- *Notes: ads related to the sale of academic notes and books.*
- *News: articles and post related to recent news in the field of education.*

For the development of the web application, MEVN stack based technologies were used. This software stack relies in the following components:

- *Mongodb: a database to store ads related with individual lessons, notes, and news as well as the project session management with user data.*
- *Express: a framework for the creation of the API REST to communicate data the client (frontend) and the server (backend). The server code implements the main CRUD backend methods (insert, collect, update and delete) of the project for the session and ads components.*
- *Vue: a Javascript based framework for web development. The Vue framework cam handle all related web technologies in the frontend (HTML, CSS, ...).*
- *Node: a Javascript execution runtime on the server side that allows us the execution of the backend code.*

Aditionally, other tools were used to track the development of the project. As a control version system, I have been using Git with Github as the remote storage support. A TDD programming methodology were followed with the use of continuous intetrations tools: Travis-CI for testing and SoncarQube for quality code management. A continuous deployment strategy were also introduced, in order to deploy new releases of the project automatically to Heroku.

Keywords: news, teachers, particular, notes, books, MEVN, Mongodb. Express, Vue, Node.

Índice general

1. **Introducción a la aplicación** **1**
   - 1.1. Descripción y objetivos . . . 1
   - 1.2. Justificación del proyecto . . . 2
   - 1.3. Antecedentes y estado actual . . . 2
   - 1.4. Metodología de trabajo . . . 2
2. **Herramientas y Tecnología** **3**
   - 2.1. Stack de desarrollo con MEVN . . . 3
     - 2.1.1. Mongodb . . . 3
     - 2.1.2. Express . . . 4
     - 2.1.3. Vue.js . . . 4
     - 2.1.4. Node.js . . . 5
   - 2.2. Github . . . 5
   - 2.3. Integración continua . . . 6
     - 2.3.1. Entorno de integración continuo con Travis CI . . . 6
     - 2.3.2. Despliegue continuo con Heroku . . . 7
     - 2.3.3. Análisis de código con SonarQube . . . 8
     - 2.3.4. Empaquetador de módulos con Webpack . . . 8
     - 2.3.5. Optimización de ficheros CSS con Purifycss . . . 9
     - 2.3.6. Cloudinary . . . 9
3. **Desarrollo** **11**
   - 3.1. Backend . . . 11
     - 3.1.1. Backend Sesión . . . 12
     - 3.1.2. Backend Productos . . . 14
     - 3.1.3. Test Backend . . . 17
   - 3.2. Frontend . . . 20
     - 3.2.1. Boostrap . . . 20
     - 3.2.2. Header y footer . . . 21
     - 3.2.3. Paginación . . . 21
     - 3.2.4. Frontend Sesión . . . 24
     - 3.2.5. Frontend Productos . . . 29
     - 3.2.6. Página responsive . . . 34
     - 3.2.7. Test Frontend . . . 35
4. **Guía de uso** **38**
   - 4.1. Acceder a la aplicación . . . 38
   - 4.2. Datos generales . . . 38
   - 4.3. Página web . . . 39
     - 4.3.1. Registro de un nuevo usuario . . . 39

4.3.2. Inicio de Sesión 40
4.3.3. Cerrar sesión 40
4.3.4. Modificar cuenta de usuario 41
4.3.5. Subir anuncio 42
4.3.6. Editar/eliminar anuncio 43
4.3.7. Buscar anuncios 43

5. Conclusiones y líneas futuras **45**
5.1. Conclusiones 45
5.2. Líneas futuras 45

6. Summary and Conclusions **47**
6.1. Conclusions 47
6.2. Future Work 47

7. Presupuesto **49**
7.1. Justificación del presupuesto 49

A. Apéndice: Test **50**
A.1. Test Backend 50
A.2. Test Frontend 53

Índice de Figuras

2.1. Integración continua 6
2.2. Entorno gráfico con Webpack 10

3.1. Prueba sobre API REST 18
3.2. Prueba sobre sesión de usuario 19
3.3. Prueba sobre productos 19
3.4. Esquema de componentes del proyecto 20
3.5. Página responsive 21
3.6. Entorno gráfico paginación 24
3.7. Entorno gráfico registro 25
3.8. Entorno gráfico inicio de sesión 27
3.9. Entorno gráfico Modificación cuenta de usuario 28
3.10 Pagina que muestra los datos del usuario 29
3.11 Página para insertar anuncios 30
3.12 Entorno gráfico de muestra de anuncios en página de nuevo anuncio o administrador 30
3.13 Explicación proyecto 31
3.14 Carrusel anuncios 31
3.15 Estilo anuncios de noticias 32
3.16 Estilo anuncios de clases particulares o apuntes 32
3.17 Página de búsqueda de anuncios 33
3.18 Mapa empresa 33
3.19 Página de contacto 34
3.20 Página de modificación de cuenta de usuario 34
3.21 Página de anuncios sobre clases particulares 35
3.22 Salida test con Jest 36

4.1. Lugar donde clicar icono usuario 39
4.2. Lugar donde ir a página registro 39
4.3. Página de registro 40
4.4. Página de Inicio de sesión 40
4.5. Cerrar sesión 40
4.6. Como encontrar la pestaña de modificar usuario 41
4.7. Página de modificar Usuario 41
4.8. Como encontrar la pestaña de subir anuncio 42
4.9. Página de subir anuncio 42
4.10 Eliminar o editar anuncio 43
4.11 Como encontrar la pestaña del buscador 43
4.12 Página del buscador 44

Índice de Tablas

7.1. Presupuesto 49

Capítulo 1

Introducción a la aplicación

1.1. Descripción y objetivos

El TFG propuesto consiste en el desarrollo de una aplicación web basada en la idea de ayudar a los alumnos de cualquier ámbito académico en donde se podrán encontrar:

- Noticias: existencia de noticias importantes sobre la actualidad de la educación (becas existentes, apertura del curso académico, ...).
- Profesores particulares: anuncios de profesores particulares enfocados a todos los cursos académicos. No obstante, están enfocado a aquellos alumnos que pueden explicar una asignatura en aquellos campos donde existan pocos profesores (generalmente universidad).
- Apuntes: anuncios sobre la venta de apuntes o libros sobre un campo determinado.

Todo esto, desarrollado mediante el stack de desarrollo MEVN y, su posterior despliegue en la nube. El stack de desarrollo MEVN se basa en la idea de unificar 4 tecnologías en una para crear una página web. Añadido a la idea de crear componentes lo que provoca que no sea necesaria la recarga de la página para que estos se actualicen, es decir, resalta la importancia de los eventos asíncronos. La tecnología que usa este stack de desarrollo sería la siguiente:

- Mongodb: base de datos de tipo no SQL que nos posibilita el manejo de grandes volúmenes de datos gracias a la tecnología de JSON.
- Express: framework que nos permite facilitar la creación de API REST, puesto que contiene muchas de las funcionalidades de estas y otras aportaciones que nos benefician.
- Vue.js: framework de desarrollo de la una página web que implementa la idea de usar Javascript en prácticamente todo el proyecto.
- Node.js: entorno de programación del lado del servidor que nos proporciona la posibilidad de mostrar una página web en el navegador.

1.2. Justificación del proyecto

A lo largo del desarrollo de la carrera e incluso en enseñanzas previas, muchos alumnos han necesitado ayuda en asignaturas concretas, en donde es muy difícil encontrar un profesor particular que sirva de refuerzo, debido a que los graduados no suelen enfocarse a este mercado. No obstante, se ha planteado la creación de una herramienta que solvente el problema anterior. Esta herramienta, se basa en una aplicación en donde existan anuncios (tanto de ofertas como demandas), enfocados principalmente a que alumnos que aún este cursando dicho nivel académico, puedan proporcionar sus recursos a cambio de una compensación económica.

1.3. Antecedentes y estado actual

En la actualidad existen diversas páginas con esta idea, pero ninguna con el criterio de unificar todo en una único lugar. Algunas de las páginas que contienen alguna función de estas son:

- www.tusclasesparticulares.com: un portal donde buscar profesores particulares. Se ha desarrollado a través de las herramientas proporcionadas por Microsoft Azure.
- www.wuolah.com: un portal en donde poder ganar dinero con tus apuntes. Se ha desarrollado bajo jQuery, Materialize y su despliegue junto con la base de datos proporcionada por Amazon.
- www.educaciontrespuntocero.com/noticias: un portal en donde poder leer noticias de interés sobre la educación. Se ha desarrollado a través de las herramientas proporcionadas por Wordpress.

1.4. Metodología de trabajo

Durante el desarrollo de este proyecto se ha apostado por el uso de las metodologías ágiles, en concreto SCRUM [7], esta se basa en dividir el proyecto en partes más pequeñas para optimizar su realización. Estas fases son:

- Análisis: etapa inicial y más importante del proyecto, en donde se identifica la idea inicial del portal web de ayuda al estudiante, así como las herramientas y las tecnologías requeridas para su desarrollo.
- Desarrollo: etapa más prolongada del proyecto, en la cual se realiza todas actividades relacionadas con la elaboración de la aplicación. Del mismo modo, encontramos lo que se conoce como interacciones del proceso o Sprint, es decir, pequeñas entregas del producto final, en este caso particular se hace referencia a pequeñas tutorías con el supervisor del proyecto.
- Testing: etapa final que consiste en la realización de una series de pruebas unitarias sobre la aplicación web (tanto frontend como backend) para comprobar que todo se ejecuta según lo deseado.

El uso de este método posibilita la continua comprobación de errores para que no se acumulen en un futuro. A su vez, permite que el receptor (en este caso el tutor) se implique en el desarrollo del proyecto aportando cambios y propuestas mejoras.

Capítulo 2

Herramientas y Tecnología

2.1. Stack de desarrollo con MEVN

Para el desarrollo de una página web es muy importante seleccionar el conjunto de software necesario para tal fin. Tras analizar detenidamente la cantidad de software existente, he escogido el que nos proporciona MEVN [27, 14], esta elección se debe a que nos proporciona muchas ventajas, entre las que cabe destacar: presentar previa experiencia en esta y su sistema de componentes.

El stack de desarrollo MEVN [2], esta compuesto por Mongodb,Express,Vue y Node. Este software nos permitirá la creación de la página web con mucha facilidad, dado que nos proporcionarán bibliotecas, en donde muchas de las funcionalidades estarán hechas. Además, nos aportarán rapidez en la aplicación e incluso seguridad.

2.1.1. Mongodb

Durante varios años las bases de datos usadas para las aplicaciones web han sido las relacionales. Sin embargo, junto con la evolución de las aplicaciones web aparecen nuevas necesidades, principalmente, la escalabilidad en los proyectos, particularidad que las bases de datos relacionales no solventan adecuadamente. Por ello, han nacido las bases de datos no relacionales [13]„ las cuales solventan este problema y además, nos ofrecen un abanico amplio de ventajas, como por ejemplo:

- Menos requisitos de hardware para su uso.
- Capacidad de dividir la carga en diversas máquinas en vez de darle la carga tan sólo a una.
- Capacidad de cambiar esquemas de la base de datos, mientras que esta sigue utilizándose.

Uno de estas bases de datos no relacionales (comúnmente denominadas como no SQL) es Mongodb [20], que se basa en el uso de Javascript para este fin mediante JSON (JavaScript Simple Object Notation), el cuál permite representar objetos mediante esta tecnología de forma simple y concisa. Un ejemplo de esta tecnología sería:

```
_id: ObjectId("5c9767f0e7909732a48b888c")
anunciante: "omarperezznakar@gmail.com"
fecha: "2019-05-02"
titulo: "Aprende a crear tu blog escolar"
foto: "http://telmexeducacion.com/noticias/Imagenes%20Noticias/Notas/Blog_esc..."
descripcion: "El blog escolar es una herramienta cada vez más usada en los centros e..."
tipo: "noticias"
nivel: ""
provincia: "Tenerife"
localidad: ""
precio: null
__v: 0
```


En donde podemos ver cada uno de los argumentos de un dato específico (id, anunciante, fecha,...) y la información asociada a cada campo.

2.1.2. Express

Express [11, 4, 16] es el framework más extendido y usado de Node.js. Este framework nos aporta:

- Manejo de peticiones HTTP.
- Establecer puertos por donde se verá la aplicación.
- Creación de rutas de la API REST.
- Elementos indispensables para la conexión con la base de datos junto con Mongoose (lugar donde está la base de datos, funciones get,post,put,delete,...).

Gracias a esta herramienta, se ha podido conectar a la base de datos de forma más fácil, y a su vez, también ha sido indispensable para el trabajo en local, debido a que se le ha especificado que si no existe un puerto predefinido se le establezca el puerto 3000.

2.1.3. Vue.js

Vue [9, 25] es un framework progresivo (necesita pocas librerías para empezar a usarlo y de forma progresiva las vas incorporando) que se basa como muchos otros, en la existencia de componentes que se ejecutan por separado para formar una página web. Además, incorpora muchas ideas nuevas y muy sencillas de usar, como pueden ser:

- Todo el código de cada componente se pueden encontrar en un único fichero (CSS,HTML y Javascript). Pudiendo hacer que este lo contenga todo o separarlo. Esto provoca, mayor posibilidad de personalización y mayor adaptación a las necesidades y gustos de diferentes personas.
- Como muchos otros framework introduce el concepto de componentes, los cuales son necesarios para evitar el recargo continuo de página que se usaba en el pasado.
- Comunicación entre componentes mediante eventos asíncronos de formas sencilla. En el proyecto, esto fue indispensable para pasar información relevante entre componentes, como puede ser el JSON completo de un anuncio a la página de visualización de este.
- Facilidad de integración con otras herramientas (Webpack,Travis,...).

2.1.4. Node.js

Node [8, 4, 21] es un entorno de ejecución para Javascript, no solo para el cliente sino también para el lado del servidor. Esta idea es muy importante, debido que junto con el motor V8 desarrollado por Google, es capaz de compilar y ejecutar Javascript muy rápido. Además de esta ventaja, Node.js también aporta muchas otras, algunas de estas son:

- Una escalabilidad bastante buena, debido a que permite un gran número de conexiones a la base de datos sin prácticamente (salvo casos extremos) notar un deterioro en la velocidad de las páginas web.
- Operaciones asíncronas con el servidor, es decir, resuelve el problema que existía de conexiones síncronas de manera que si no se acababa una petición no resolvía la siguiente.
- Gran variedad de librerías gratuitas que se descargan rápidas.

Por todo lo comentado, se ha elegido esta tecnología. Además, de que se integra perfectamente con el resto de herramientas que se han mencionado en los apartados anteriores.

2.2. Github

A lo largo del desarrollo de un proyecto es muy interesante tener un historial de los cambios que se van realizando, principalmente por si ocurre algún fallo inoportuno poder volver a una versión anterior a la aplicación que estemos desarrollando. Por esto nació Github [17] , la cual nos permite con tan sólo unos simples comandos poder hacer un registro de todo lo que estemos haciendo.

Durante el desarrollo de este proyecto se ha hecho uso de esta tecnología desde el inicio hasta el final registrando todo, desde los cambios hechos hasta quien los realiza e incluso la fecha en la que los cambios son realizados. A su vez, se han creado diversas ramas en donde se han hecho partes concretas del proyecto, estas ramas son:

- Master: rama que recoge la totalidad del proyecto.
- Backend: esta rama se ha usado para toda la funcionalidad del backend (conexiones con servidor,..). Ver capítulo 3.1 para más información.
- Frontend: esta rama se ha usado para toda la funcionalidad del frontend (conexiones con el backend, funcionalidades, parte visual,..). Ver apartado 3.2 para más información.
- Test: esta rama se ha usado para toda la parte de comprobación de fallos de la aplicación.Ver capítulos 3.2.7 y 3.1.3 para más información.
- Errores: esta rama se ha usado para corregir los diversos fallos que van surgiendo durante el desarrollo.
- Optimización: como todo los proyectos también necesitan optimizarse y eliminar líneas de código para mejorar su velocidad y comprensión, para ello se ha usado esta rama.

A su vez, junto a esta aplicación se han realizado dos despliegues tanto de la página web (ver apartado 2.3.2) como de una pequeña explicación, tanto del TFG como de las tecnologías usadas, esto se podrá encontrar en:

- Aplicación: `https://cosasdeclase.herokuapp.com/`
- Explicación: `https://omar97perez.github.io/CosasDeClase-MEVN/`
- Github: `https://github.com/Omar97perez/CosasDeClase-MEVN`

2.3. Integración continua

La integración continua [5] consiste en comprobar automáticamente y de forma periódica el código con el fin de encontrar fallos que, posteriormente podrían generar muchos problemas. A su vez, también ayudan a obtener un código con mucha más calidad que si no se hiciera con esta práctica. En este proyecto, se ha usado la integración continua [6] de la siguiente manera:

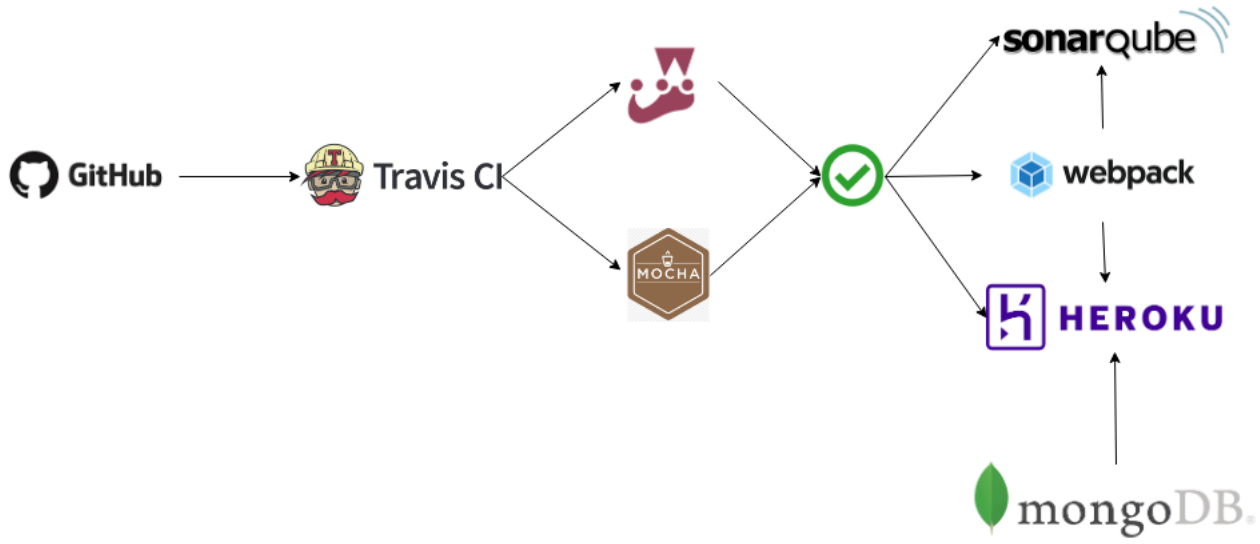


Figura 2.1: Integración continua

Como vemos en la imagen al realizar un commit Github enviará mediante su fichero de configuración la información a Travis, este realizará una serie de comandos (que hemos establecido antes) y, generará un archivo webpack (compilación de código en Javascript). Este código, tendrá que pasar el testing realizado con Mocha y Jest. Posteriormente, junto al resto del proyecto serán evaluados por Sonarqube en busca de errores y code smels. Y por último, si el resultado es el esperado, hará el despliegue continuo a la nube de los cambios realizados en la aplicación junto con la base de datos Mongodb Atlas [19].

Con todo lo anteriormente citado, podemos asegurarnos de que gran parte de los errores serán comprobados y beneficiará a la mejora de la calidad del código.

2.3.1. Entorno de integración continuo con Travis CI

Travis [10, 24] es el entorno, quien junta todas las plataformas de comprobación y despliegue de una aplicación.

Para usar este tecnología es necesario darle permisos a Travis en Github y realizar un archivo de configuración que consiste en incorporar todos los pasos que este debe de seguir para realizar lo citado. En este proyecto, el fichero de configuración es el siguiente:

```
language: node_js
node_js:
- '10'
addons:
  sonarqube:
    organization: "omar97perez-github"
    token:
        secure: 31fdac66d1953eb7683fc66b93f64f2b6f2ad586
sudo: false
script:
- cd app
- npm install
- npm run test
- npm run jest
- sonar-scanner
cache:
  directories:
  - node_modules
```

Como podemos ver en el fichero se le ha especificado:

- Versión de Node usada en el proyecto.
- Organización a la que pertenece (una propia) en SonarQube y código de seguridad de dicha organización.
- Pasos a ejecutar por Travis, en este caso debe de acceder a la carpeta (en donde esta el proyecto) e instalar las dependencias necesarias, ejecutar los test y, por último, realizar las comprobaciones de SonarQube.
- Se le especifica el directorio cache del proyecto (carpeta donde residen las dependencias instaladas previamente) para que no se compruebe dicho fichero.

2.3.2. Despliegue continuo con Heroku

Heroku [18], es una aplicación web que permite el despliegue de un proyecto en la nube. En la versión gratuita se dispone de recursos muy limitados para su prueba.

En este proyecto se ha realizado un despliegue continuo únicamente en la rama master, haciendo que sólo cuando exista un commit (o en este caso específico, un merge) suceda el despliegue. Esto es debido a que en la versión gratuita sólo deja un número limitado de despliegues quedando limitado esta herramienta en este aspecto. A su vez, el despliegue (mientras que se iba realizando el proyecto) se ha realizado mediante Node de forma local ligado con Webpack (ver apartado 2.3.4) mediante dos funcionalidades creadas en el Package.son, estas funcionalidades son:

- ”dev”: ”nodemon app/src/index.js –ignore ’src/public’”
- ”webpack”: ”webpack –watch –mode development”

Estas funcionalidades se ejecutan en dos terminales, permitiendo realizar cambios en el proyecto, al guardar se activa webpack que ejecuta el código y genera un fichero bundle.js que es ejecutado por el servidor de Node y, muestra los cambios en el navegador de forma automática en el localhost especificado en el backend.

2.3.3. Análisis de código con SonarQube

SonarQube [23] es una aplicación que te analiza el código, encontrando errores y code smel. Es una herramienta muy útil,ya que te ayuda a mejorar y optimizar el código mediante una interfaz sencilla. Algunos de los errores que me han surgido, han sido:

- Existencia de etiquetas de HTML4 que debían de actualizarse a las nuevas en HTML5..
- Código duplicado que debía de ser borrado.
- Variables declaradas que nunca se usaban.

2.3.4. Empaquetador de módulos con Webpack

A modo de resumen Webpack [28, 26] consiste en coger todo el código generado en una aplicación, traducirlo a Javascript y unificarlo todo en un único fichero el cúal será ejecutado por el servidor. Para realizar esto en necesario crear un archivo de configuración el cuál sería el siguiente:

```
const  {VueLoaderPlugin}  = require('vue-loader');

module.exports = {
  entry: './app/src/app/index.js',
  output: {
    path: __dirname + '/app/src/public/js',
    filename: 'bundle.js'
  },
  module: {
    rules: [
      {
        test: /\.js$/,
        exclude: /node_modules/,
        loader: 'babel-loader'
      },
      {
        test: /\.vue$/,
        loader: 'vue-loader'
      }
    ]
  },
  plugins: [
    new VueLoaderPlugin()
  ]
};
```

Dicho fichero consiste en:

- Especificarle donde se encuentran los archivos del proyecto, con MEVN es tan fácil como introducir la ruta de index,js (lugar en donde se encuentran la declaración de componentes del proyecto).
- Lugar y nombre desde donde se va a usar por el servidor.Además,este fichero será llamado desde un index.html en esta misma carpeta, el cual contiene todos los links y script necesarios para el proyecto (boostrap,jquery,...).
- Ficheros o carpetas que no son necesarios que sean compilados (nodeModules la carpeta donde residen las dependencias del proyecto).

2.3.5. Optimización de ficheros CSS con Purifycss

Purifycss [22] es una herramienta de optimización de CSS que consiste en el análisis del código y elimina aquellas líneas del CSS que no son utilizadas. En este proyecto, ha sido muy útil, debido a que los ficheros de Boostrap han sido descargados y no puestos mediante un link. Posibilitando que nos ahorremos miles de líneas, que en caso contrario, cargaríamos sin ningún fin. Además de un refuerzo para las nuevas líneas creadas.

En primera instancia, se realizó junto con Webpack, pero debido a la lentitud de este proceso se cambio para el uso manual de esta herramienta. Para facilitar su uso el comando se introdujo en el Packet.json y sería el siguiente:

- ”purifycss”: ”purifycss app/src/public/css/style.css app/src/app/components/*.vue –min –out app/src/public/css/final.css –info”

Dicho comando se le especifica el fichero CSS a analizar, los componentes existentes y el lugar en donde reside el fichero de salida.

2.3.6. Cloudinary

Cloudinary [1, 15] es un sitio web en el que se puede gestionar imágenes. Estas, se suben a través de la API de un proyecto web y se gestionan a través de su página con un entorno gráfico sencillo e intuitivo.

En este proyecto se ha hecho uso de esta tecnología para la subida de imágenes tanto de los usuarios como de los anuncios y, para este fin, se ha hecho:

- Crear una función Javascript, la cuál se encarga de la subida de la imagen. Esta función (a través del uso de los id de HTML) usa los siguientes argumentos:
  - La imagen.
  - El porcentaje de subida de la carga (inicial mente a 0).
  - El evento que al cambiar lo asocia con una nueva imagen.
  - La URL del repositorio de Cloudinary.
  - Contraseña generada por Cloudinary para permitir la subida.

- Crear el entorno gráfico que vería el usuario al subir la foto. Dicho entorno sería:

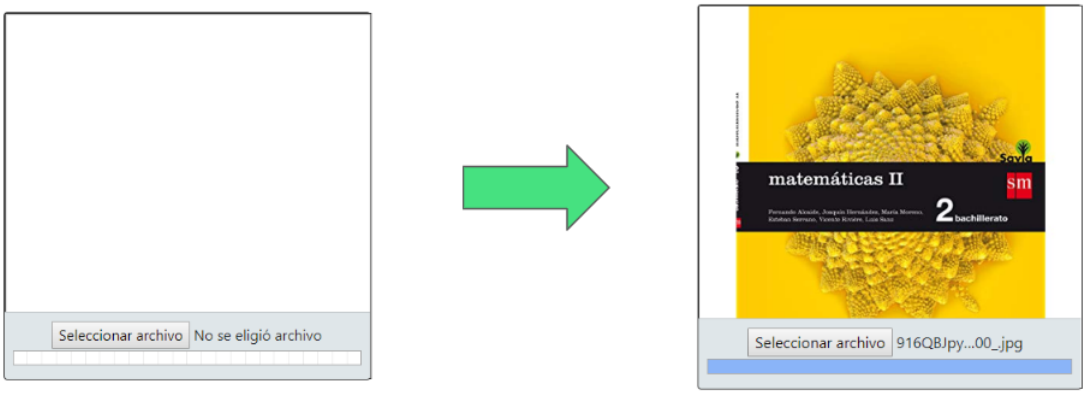


Figura 2.2: Entorno gráfico con Webpack

Capítulo 3

Desarrollo

3.1. Backend

El backend [3, 12] es toda la lógica que pasa en una pagina web y no podemos ver (conexión con la base de datos, puertos, funciones de inserción,eliminación,...).En este proyecto, como en todos, se ha tenido que crear una estructura general para el backend. Para ello se ha usado:

- Requisitos indispensables no sólo para el backend sino para la aplicación. Estos serían:

```
require('rootpath')()
const express = require('express')
const morgan = require('morgan')
const mongoose = require('mongoose')
const cors = require('cors')
const bodyParser = require('body-parser')
const jwt = require('./authentication/backend/_services/jwt')
const errorHandler = require('./authentication/backend/_services/error-handler')


const app = express();
app.use(bodyParser.urlencoded({ extended: false }))
app.use(bodyParser.json())
app.use(cors())
app.use('/users', require('./authentication/backend/users/users-controller'))
app.use(errorHandler)
```

- Conexión con la base de datos. En este caso, al hacer un despliegue en la nube se ha introducido el link hacía el cluster proporcionada por MongodbAtlas.
- Despliegue: para este apartado, se ha hecho que el despliegue se genere en el puerto 3000, siempre y cuando no exista un puerto establecido por la máquina (esto último para el despliegue en Heroku). Para ello se ha hecho:

```
//Ajustes
  //Si el servidor tiene puerto lo coge sino pone el puerto 3000
  app.set('port', process.env.PORT || 3000  );

//Sever escucha en el puerto x te lo muestra por pantalla
app.listen(app.get('port'), () =>{
    console.log('Server on port', app.get('port'));
});
```

3.1.1. Backend Sesión

A la hora de implementar la API para controlar el uso de la sesión de los usuarios se ha hecho uso de Mongoose, para crear un modelo que permita el correcto uso de este tipo de dato. Para ello, se ha hecho uso de JSON junto con la tecnología anteriormente citada. Dicho modelo sería el siguiente:

```
const userSchema = new Schema({
    // username: { type: String, unique: true, required: true },
    password: { type: String, required: true },
    name: { type: String, required: true },
    surname: { type: String, required: true },
    email: { type: String, unique: true, required: true },
    paragraph: { type: String, required: true },
    image: { type: String, required: true },
    telephone: { type: String, required: true },
    birthdate: String,
    genre: { type: String },
    createdDate: { type: Date, default: Date.now }
});
```

En donde:

- id: número único que dispone cada cuenta de usuario.
- name: nombre del usuario.
- surname: apellidos del usuario.
- email: correo electrónico del usuario. Este apartado, es único debido a que sólo existe un email en el mundo. Además, nos ayuda a unir las tablas de usuarios y productos por este campo, con esto logramos encontrar todos los productos que oferta un usuario.
- paragraph: pequeño párrafo descriptivo de un usuario específico. Este párrafo, será mostrado en el perfil creado en el Frontend.
- image: foto personal del usuario.
- telephone: número de teléfono de contacto del usuario.

- birthdate: fecha de nacimiento del usuario. Con este apartado, vemos la edad del usuario.
- createdDate: fecha de creación de la cuenta. Este apartado, se rellena solo y recoge la fecha actual.

Junto con lo anteriormente comentado, es necesario funcionalidades para operar con la base de datos. Estas funcionalidades serían:

- Crear un nuevo usuario: se ha implementado una función, la cual tiene que comprobar si un email existe en la base datos, para posteriormente, si no existe crear el usuario. Dicha función sería:

```
async function create(userParam, res) {

    if (await User.findOne({ email: userParam.email })) {
        throw Error(`El email ${userParam.email} ya está registrado`)
    }
    else {
        const user = new User(userParam)

        if (userParam.password) {
            user.password = bcrypt.hashSync(userParam.password)
        }
        await user.save()
    }
}
```

- Autentificar a un usuario: se ha implementado una función, la cual permite autentificar a un usuario tras pasarle un correo y. Además, si dicho usuario se loguea se crea un token con el usuario creado. Esta función sería:

```
async function authenticate({ email, password }) {
    const user = await User.findOne({ email })

    if (user && bcrypt.compareSync(password, user.password)) {
        const token = jwt.createToken(user)
        return {
            token
        };
    }

}
```

- Recoger usuarios: se ha implementado una función que posibilita recoger un usuario de la base de datos, siempre sin recoger la contraseña. Esta funcionalidad permite crear un perfil de usuarios, el cual permite al cliente conocer un poco más del usuario que oferta un producto o servicio. Estas funcionalidades son:

```
async function getAll() {
    return await User.find().select('-password')
}

async function getById(id) {
    return await User.findById(id).select('-password')
}
```

- Actualizar usuario: se ha implementado una función, que permite actualizar los campos de un usuario. Esta funcionalidad permite que tras pasarle el id de un usuario y un objeto con los parámetros modificados, permiten actualizar los datos a los nuevos. Esta función sería:

```
async function update(id, userParam) {
    const user = await User.findById(id)

    if (!user) throw 'User not found'
    if (user.email !== userParam.email && await User.findOne({ email: userParam.email })) {
        throw Error(`email ${userParam.email} is already taken`)
    }

    if (userParam.password) {
        user.password = bcrypt.hashSync(userParam.password, 10)
    }

    .
    .
    .
   if(userParam.genre) {
       user.genre = userParam.genre
   }

   await user.save()
}
```

- Eliminar usuario: se ha implementado una función que tras pasarle el id (elemento único del modelo) se elimina un usuario. Esta función sería:

```
async function _delete(id) {
    await User.findByIdAndRemove(id)
}
```

3.1.2. Backend Productos

En este apartado se hablará únicamente de la parte referida al backend de los productos del proyecto, es decir, a lo referido con los anuncios de apuntes,noticias y clases particulares.

A la hora de implementar la API para controlar el uso de los productos en este proyecto, se ha hecho uso de Mongoose para crear un modelo que permita el correcto almacenamiento en la base de datos mediante JSON. Dicho modelo sería el siguiente:

```
var mongoose = require('mongoose');
var Schema = mongoose.Schema;

//Esquema para Productos
const Productos = new Schema({
  id: Number,
  anunciante: String,
  fecha: String,
  titulo: String ,
  foto: String ,
  descripcion: String,
  tipo: String ,
  nivel: String ,
  provincia: String ,
  localidad: String ,
  precio: Number  ,

});

module.exports = mongoose.model('Productos', Productos);
```

En donde:

- Id: número único que dispone cada producto.
- Anunciante: correo electrónico único de la persona que sube el anuncio. Este campo, será rellenado de forma automática por el sistema a través del token de autentificación de la persona que quiera subir dicho anuncio.
- Fecha: fecha en la que el usuario ha subido el anuncio.
- Título: título del anuncio que se requiere subir.
- Foto: url de la imagen que se requiere subir.
- Descripción: texto descriptivo del producto.
- Tipo: tipo de producto que es, es decir, se requerirá una elección entre apuntes, noticias y clases particulares.
- Nivel: en caso de referirse a apuntes o clases particulares se requerirá una especificación del nivel académico al que se refiere el anuncio (universidad, bachillerato,...).
- Provincia: en caso de referirse a apuntes o clases particulares se requerirá una especificación de la provincia, en donde se recoge o se imparte dicho anuncio.
- Localidad: en caso de referirse a apuntes o clases particulares se requerirá una especificación de la localidad, en donde reside el propietario del anuncio.

- Precio: precio del producto. En caso de referirse a clases particulares se refiere al importe por hora del profesor. Y en caso, de referirse a apuntes el precio final. Al igual que varios apartados anteriores sólo se refiere en los casos especificados.

Junto con lo anteriormente citado, también se han implementado diversas funcionalidades. La cuáles serían:

- Introducir: se ha implementado una función para insertar elementos en la base de datos, siempre y cuando estén acordes con el modelo anteriormente citado. A su vez, para mejorar la calidad de la aplicación se ha implementado un pequeño mensaje para verificar su introducción. Esta función sería:

```
//Introducir datos
router.post('/',async (req,res) =>{

    const producto = new Productos(req.body);  //>Similar a Select * From x (SQL)
    //console.log(producto); //Imprime por consola lo contenido en la variable productos
    //res.json(producto); //Imprime en el navegador lo contenido en la variable producto
    await producto.save();
    res.json({
        status:'Producto guardado'
    });
});
```

- Recoger: se ha implementado una función para recoger todos los productos de la base de datos. A su vez, también se ha implementado una función para recoger un único elemento de la base de datos, para hacer uso de esta función hace falta que reciba el id (elemento único para este modelo). Estas funciones son:

```
//Recoger lo que tenga la base de datos
router.get('/', async (req,res) => {
    const producto = await Productos.find()  //>Similar a Select * From x (SQL)
    res.json(producto);
    //console.log(producto); //Imprime por consola lo contenido en la variable productos
    //res.json(producto); //Imprime en el navegador lo contenido en la variable producto
});

router.get('/:id', async (req,res) => {
    const producto = await Productos.findById(req.params.id)  //>Similar a Select * From x (SQL)
    res.json(producto);
});
```

- Eliminar: se ha implementado una función la cual al pasarle como parámetro el id (elemento único para este modelo) es capaz de eliminar dicho producto. A su vez, también se ha incorporado un pequeño mensaje para verificar su eliminación. Dicha función sería:

```
router.delete('/:id', async (req,res) =>{
  await Productos.findByIdAndRemove(req.params.id);
  res.json({
      status:'Producto eliminado '
  });
});
```

- Actualizar: se ha implementado una función que con pasarle un id (elemento único para este modelo) y un elemento modificado, es capaz de sobrescribir el último sobre el primero. Esta función es muy útil para actualizar la información de un elemento siempre que se desee hacerlo. Esta función sería:

```
//Actualizar datos
router.put('/:id', async (req,res) =>{
  await Productos.findByIdAndUpdate(req.params.id, req.body);
  res.json({
      status:'Producto actualizada'
  });
});
```

3.1.3. Test Backend

Los test son un parte fundamental del desarrollo de una aplicación. Cuando desarrollamos una API REST buscamos que haga exactamente lo que queremos hacer y para ello existe esta práctica..

En este proyecto, se ha efectuado los test del backend con Mocha. Mocha, es un framework de test para Node.js el cual permite ejecutarse de forma manual mediante un comando y genera una salida, que te permite saber si todo se efectúa exactamente como se requiere. Para conseguir tal fin, se han realizado 3 ficheros, los cuales se relacionan con la API REST, productos y sesión del usuario. Además, un ejemplo de estos test se encuentra en A.2. A su vez, se ha incorporado el comando de ejecución del test en el fichero de configuración de package.json. Esta línea sería la siguiente:

- ”test”: ”mocha ./app/test/integration –exit”,

En estos test realizados con Mocha se han realizado pruebas como:

- Pruebas sobre el correcto funcionamiento de la API REST, así como pruebas de que si no existe una máquina se conecte a localhost:3000 (entre otras).
    - La salida que genera Travis en este aspecto sería:

```
Pruebas sobre get
  ✓ prueba de estado
  ✓ prueba de puerto
  ✓ prueba de permiso de escritura
  ✓ prueba de tipo
```

Figura 3.1: Prueba sobre API REST

- Este test se puede encontrar en este este enlace[1].

- En cuanto a la sesión de usuario se han realizado pruebas como:

  - Pruebas sobre el correcto registro de un nuevo usuario. Para este fin, se ha creado un objeto según el modelo descrito (ver aparatado 3.1.1) y se ha comprobado su correcta inserción en la base de datos.
  - Pruebas sobre la correcta validación de un usuario actual sobre la base datos.
  - Pruebas en el que tras validarse el token que adquiere el usuario se adhiere a su sesión.
  - Pruebas que verifiquen que los anteriores test descritos sigan el patrón de diseño que se usará en el Frontend.
  - Pruebas que verifiquen que el modelo de usuario se rige al modelo descrito (ver apartado 3.1.1), debido a que si se cambia puede verse comprometida la seguridad de la página.
  - Este test se puede encontrar en este este enlace[2].
  - Salida de Travis ante este apartado de Test:

[1] https://github.com/Omar97perez/TFG-ULL-CosasDeClase-MEVN/blob/master/app/test/integration/api_rest_test.js

[2] https://github.com/Omar97perez/TFG-ULL-CosasDeClase-MEVN/blob/master/app/test/integration/usuarios_test.js

Figura 3.2: Prueba sobre sesión de usuario

- En cuanto a los productos (anuncios,clases particulares,noticias) se ha realizado:
  - Pruebas sobre la correcta inserción a la base de datos de un objeto que sigue el modelo descrito 3.1.1
  - Pruebas de obtención del objeto anteriormente insertado.
  - Pruebas sobre la correcta modificación del objeto anterior.
  - Pruebas de borrado del objeto anteriormente citado.
  - Pruebas que verifiquen que las anteriores funcionalidades descritas sigan el modelo que se realizará en el Frontend, es decir, si para modificar un objeto es necesario pasar únicamente el id o si para eliminar un objeto sigue este mismo funcionamiento.
  - Pruebas sobre el modelo del objeto (ver apartado 3.1.1. Esta prueba consiste, básicamente en asegurarnos que cada campo del modelo siga con el mismo tipo de dato descrito, debido a que si se cambia podrían fallar funcionalidades de la página web.
  - Este test se puede encontrar en este enlace[3].
  - Salida de Travis ante este apartado de Test:

Figura 3.3: Prueba sobre productos

[3] https://github.com/Omar97perez/TFG-ULL-CosasDeClase-MEVN/blob/master/app/test/integration/producto_test.js

3.2. Frontend

El Frontend [12] está enfocado al usuario, es decir, en un página web es todo aquello que podemos observar al acceder a esta. Es todo elemento gráfico diseñado con HTML, CSS y Javascript. En este proyecto, al hacer uso de Vue.js hemos creado 16 componentes que conforman nuestra página web y, de esto, se hablará en este aparatado. Estos componentes se dividen en la página a través del siguiente esquema:

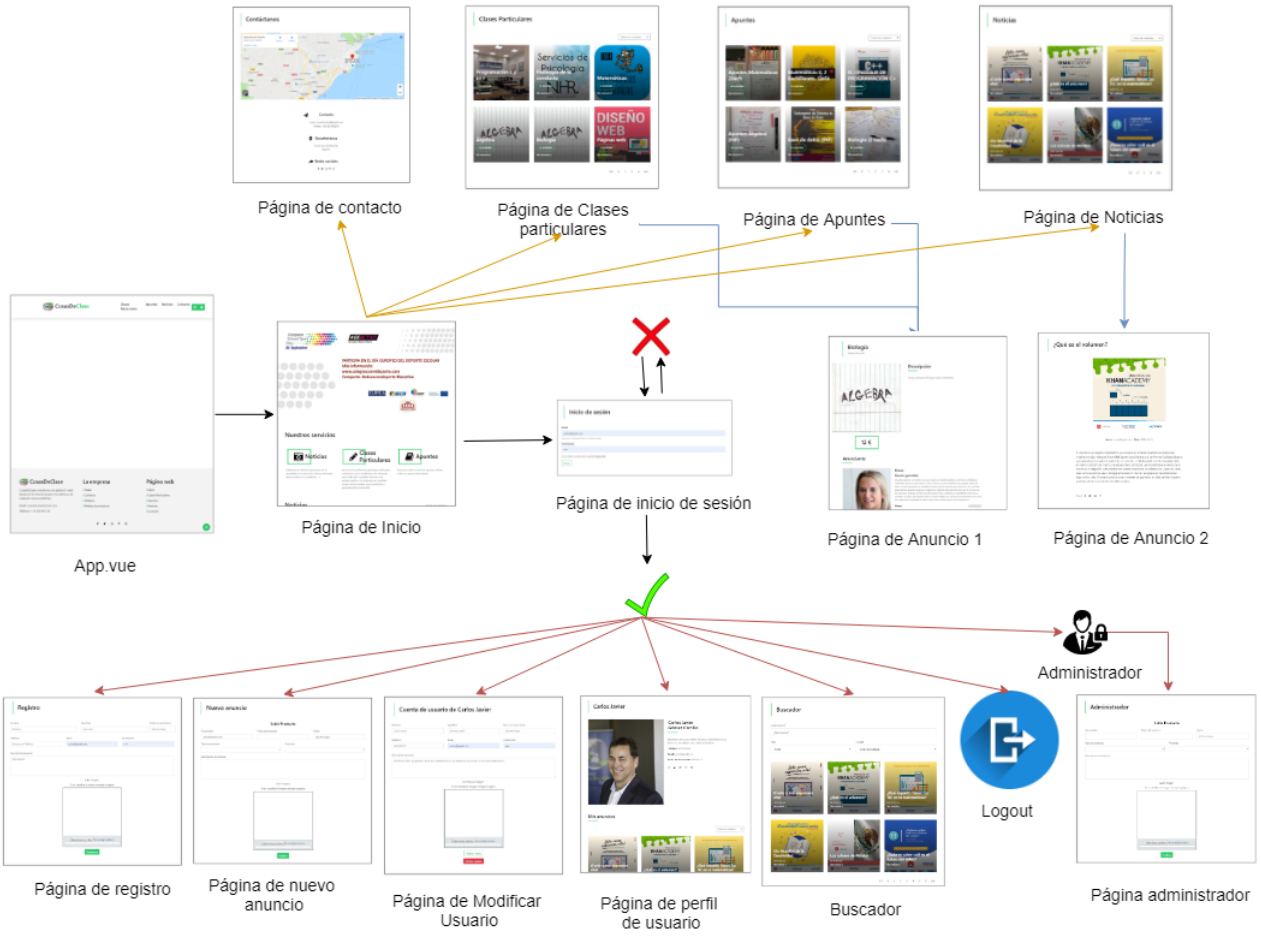


Figura 3.4: Esquema de componentes del proyecto

3.2.1. Boostrap

En este proyecto se ha hecho uso de Boostrap para la estética. Boostrap es un framework cuyo objetivo principal es facilitar la vida al programador en el diseño de una página web. Es de código abierto y nos da muchas ventajas entre las que destacan:

- Permite el acoplamiento de tu página web a cualquier dispositivo mediante el uso de filas y columnas de formas simple y atractiva.
- Permite el uso de muchos elementos web, que ya están hechos y adaptados a todos los dispositivos.
- Es open source.

- Frente a sus competidores es el framework de diseño web más extendido y, por tanto,esto significa que hay muchas funcionalidades hechas por la comunidad. A su vez, si es muy demandado dispone un mantenimiento óptimo sacando mejores versiones cada transcurso de tiempo corto.

3.2.2. Header y footer

Para el Header y footer del proyecto, se ha hecho uso de Boostrap (ver apartado 3.2.1). Esto quiere decir que, es totalmente adaptable a todos los dispositivos.

En el caso del header, se han creado dos menús separados uno para dispositivos más grandes (ordenador y tablet) y otro para móviles. He preferido crear dos menús separados, ya que, aunque sea verdad de que Boostrap te lo de hecho, no te da mucho juego de personalización y, de la segunda manera, lo adaptas a las tus necesidades, obtienes un resultado mucho más sencillo y visualmente, bajo mi punto de vista, mucho más bonito. El Header y footer implementados serían:

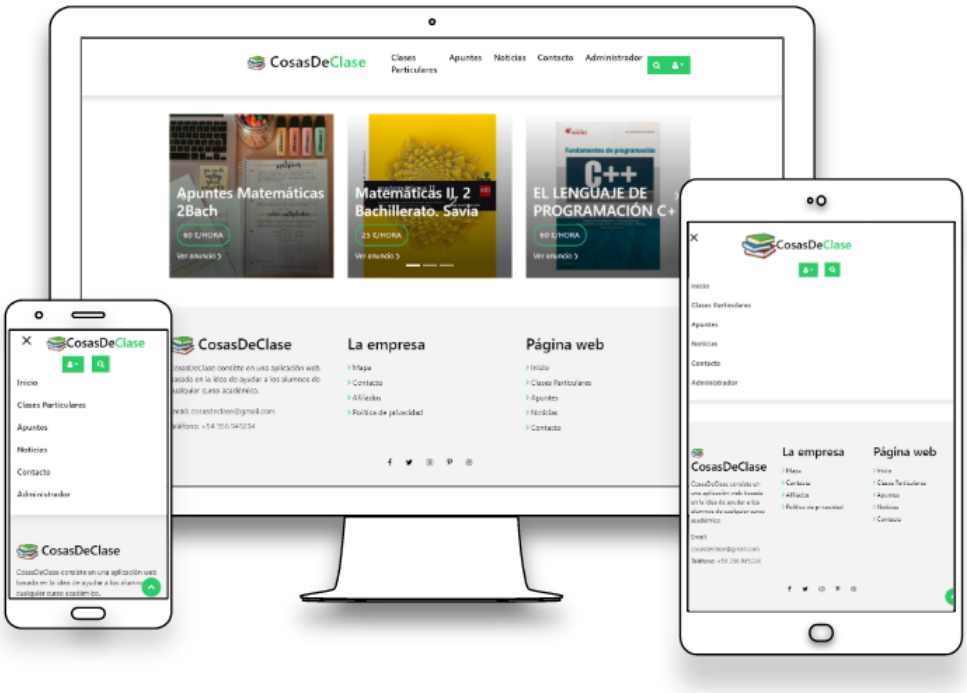


Figura 3.5: Página responsive

3.2.3. Paginación

La paginación es un método muy usado en páginas web para no sobrecargar el Frontend de datos y, que este, tarde mucho en poder imprimirlos por pantalla. Básicamente, consiste en recoger un número determinado de productos (en la mayoría del proyecto 6 elementos) de la base de datos y, que al cambiar de página estos productos se cambien por los siguientes (siguiendo la cadena de elementos), y así, sucesivamente, hasta acabar con todos los productos a imprimir.

En la actualidad Vue.js ya dispone de una paginación ya diseñada. Sin embargo, en este proyecto se ha preferido hacer una paginación propia, debido a que la estructura del proyecto provocaría que esta no fuera del todo óptima, esto se debe a que la mayoría de las páginas contienen no sólo la paginación, sino también, algún tipo de filtro o búsqueda (ver apartado 3.2.5). En concreto, la paginación implementada sería la siguiente:

- Se crean una serie de variables para que esto sea factible. Estas variables son:
  - Un vector para los productos totales.
  - Un vector para los productos que se están paginando.
  - Un vector para los productos que se van a páginas cumpliendo los requisitos del buscador.
  - Número de páginas necesarias para la paginación.
  - Número de página por el que va la paginación.
  - Tamaño de la paginación (6 elementos mostrados de una vez en la mayoría de los casos).
- En el created (método que inicializa funciones en Vue) inicializamos la función de recogida de datos y la función que te calcula el número de páginas totales necesarias. Estas funciones son:
  - Función que consiste en recoger los elementos de la base de datos desde la posición 0 hasta el tamaño de elementos de la paginación establecido. La función sería:

```
getProductos() {
  fetch('/api/CosasDeClase/Producto/')
    .then(res => res.json())
    .then(data => {
      this.Paginacion = data;
      this.Productos = this.Paginacion.slice(0,this.tampagina);
    });
```

  - Función que calcula cuantas páginas son necesarias. Para este cálculo, se coge el vector que contiene todas los productos y se divide por el tamaño de elementos de cada página (este valor ha de ser redondeado al alza, debido a que si existen valores decimales es necesario crear una página más para estos últimos valores). Esta función sería:

```
NumPaginas() {
  this.numero = Math.ceil(this.ProductosPaginacion.length/this.tampagina);
  return this.numero;
},
```

- Al cambiar cualquier elemento del buscador (usando la función @change de vue que avisa si un elemento cambia de valor) se hace llama las siguientes funciones:

- Se llama a una función que comprueba si la paginación actual necesita hacer un filtro sobre alguno de los campos necesarios en el buscador. Esta función sería:

```
buscarProducto() {
  this.ProductosPaginacion = this.Paginacion.filter(
    Producto => Producto.tipo.includes(this.tipo)
                && Producto.provincia.includes(this.ciudad)
                && Producto.titulo.includes(this.busqueda) );
  this.buscador_pagination(this.ProductosPaginacion);
}
```

- Posteriormente, es necesario calcular en que posición se encuentran los elementos que se van a imprimir, para ello, es necesario multiplicar la página anterior por el número de elementos de la misma (elemento mínimo) y, multiplicar la página que nos piden por el mismo valor (elemento máximo). Por último, es necesario filtrar desde la posición inicial a la posición final calculada del vector de productos. Esta función sería:

```
buscador_pagination(vector) {
  var numpag,x;
  numpag = this.numeropagina
  x = this.tampagina * numpag;
  numpag = numpag - 1;
  numpag = numpag * this.tampagina;
  this.Productos = vector.slice(numpag,x);
}
```

- A su vez, se imprimen todas las páginas existentes junto con otras funcionalidades añadidas. Estas funciones consisten en moverse a una página específica, a la siguiente o anterior y moverse a la primera o última página. Llaman a los mismos métodos que la búsqueda del punto anterior y, por tanto, solo es necesario modificar la página que se quiere cambiar. Por ello, estas funciones serían:

  - En caso de cambiar a una página especifica es imprescindible cambiar la página actual a la que nos piden. La función sería:

```
pagination(numpag) {
  this.numeropagina = numpag
},
```

  - En caso de cambiar a la anterior es necesario comprobar que este no sea 0 y posteriormente restar. La función sería:

```
cambioanterior() {
  if(this.numeropagina > 1 ){
      this.numeropagina = this.numeropagina - 1;
  }
},
```

- En caso de cambiar a la siguiente se comprueba que no se ha llegado al final y se suma. La función sería:

```
cambiosiguiente() {
  if(this.numeropagina < this.numero ){
      this.numeropagina = this.numeropagina + 1;
  }
},
```

- En caso de ir a la primera es necesario igualar la página actual a 1. La función sería:

```
cambioprimera() {
      this.numeropagina = 1;
},
```

- En caso de ir a la última es necesario igualar al tamaño de páginas calculada en el created. La función sería:

```
cambioultima() {
      this.numeropagina = this.numero;
},
```

- De forma estética quedaría:

Figura 3.6: Entorno gráfico paginación

3.2.4. Frontend Sesión

Página de control sesión (login,registro y logout)

Como en la mayoría de las páginas web se ha implementado la vista gráfica de todo lo referido a la sesión del usuario. Todo esto ayudado con las funciones creadas en el backend (ver capitulo 3.1). La páginas web básicas implementadas han sido:

- Página de registro: en esta, se ha implementado un formulario en el que se recoge la información básica del usuario y se manda al servidor. Para ello, se ha realizado:
  - Se ha creado una variable para cada datos necesario. Estas variables son rellenadas por el usuario a través de los input creados gracias a la ayuda de Vue con su función v-model (permite rellenar una variable a través de la entrada por teclado).
  - Posteriormente, tras pulsar el botón de registrarse se comprobará si el email introducido no está en la base de datos (elemento único de este modelo), y en caso de que no este, se mandará la información a la base datos. Por último, el usuario será redirigido a la página de login. Esta función sería:

```
register () {
  this.$store.dispatch('register', {
    name: this.name,
    surname: this.surname,
    paragraph: this.paragraph,
    image: document.getElementById('img-preview').src,
    telephone: this.telephone,
    email: this.email,
    password: this.password,
    birthdate: this.birthdate,
    genre: this.genre
  })
  .then(response => {
    this.$router.push({ name: 'Login' })
  })
  .catch(error => {
    $('#m_error_r').empty()
    $('#m_error_r').append(`
        <br>
        <div class="alert alert-danger" role="alert">
          El email introducido está en uso
        </div>
      `)
    //resolve(error)
  })
 }
}
```

- El entorno gráfico de esta implementación se vería así:

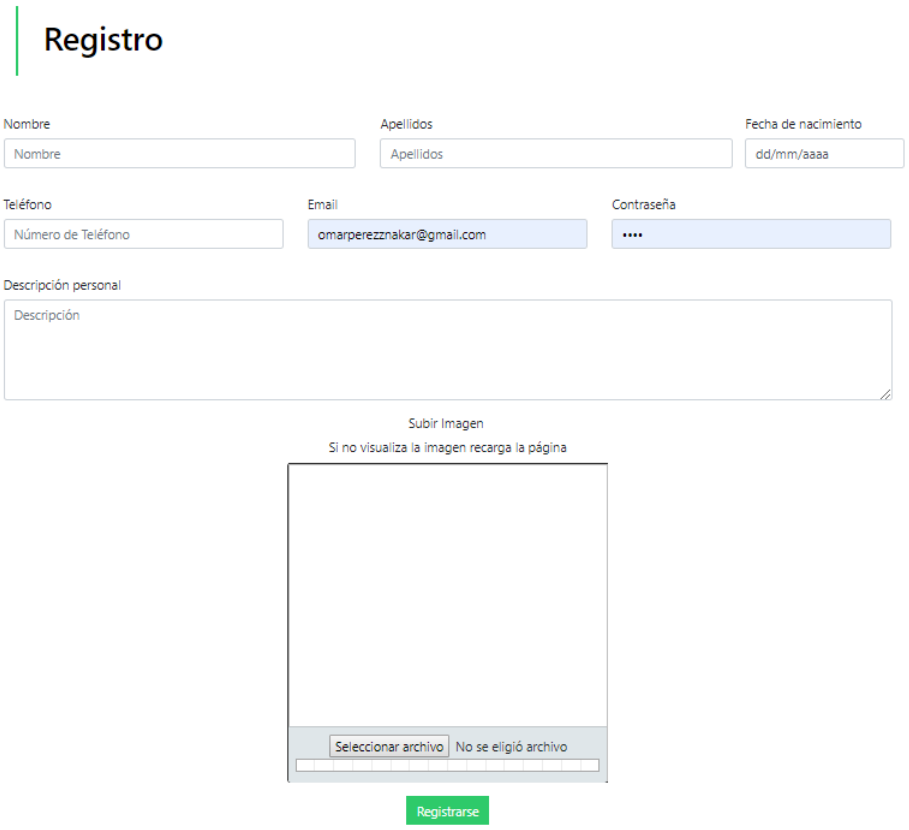


Figura 3.7: Entorno gráfico registro

- Página de login: en esta página se ha implementado un formulario básico de que recoge email y contraseña. Además, comprueba si estas credenciales son válidas. Para ello, se ha implementado:
  - Se han creado dos variables para recoger los datos necesarios que el usuario introduce por teclado (ayudado de la funvión v-model implementada por vue).
  - Tras pulsar el botón de loguearse es necesario comprobar que los datos son correctos, para ello, se ha hecho uso de una función de Vuex, denominada retrieveToken, la cual hace uso de Axios y de las funciones del backend para realizar el login. A su vez, se hace uso de la función de vue "get user data", la cual nos permite obtener información del usuario logueado para el resto de la página. Por último, tras loguearse se redirige a la página de inicio. Estas implementación en código, sería:

```
export default {
  name: 'login',
  data () {
    return {
      email: '',
      password: ''
    }
  },
  methods: {
    login() {
      this.$store.dispatch('retrieveToken',{
        email: this.email,
        password: this.password
      })
      .then(response => {
        this.$store.dispatch('get_user_data')
        this.$router.push({ name: 'index' })
      })
      .catch(error => {
        $('#m_error_1').empty()
        $('#m_error_1').append(`
            <br>
            <div class="alert alert-danger" role="alert">
              Email o contraseña incorrectos.
            </div>
          `)
      })
    }
  }
}
```

- El entorno gráfico de esta implementación se vería así:

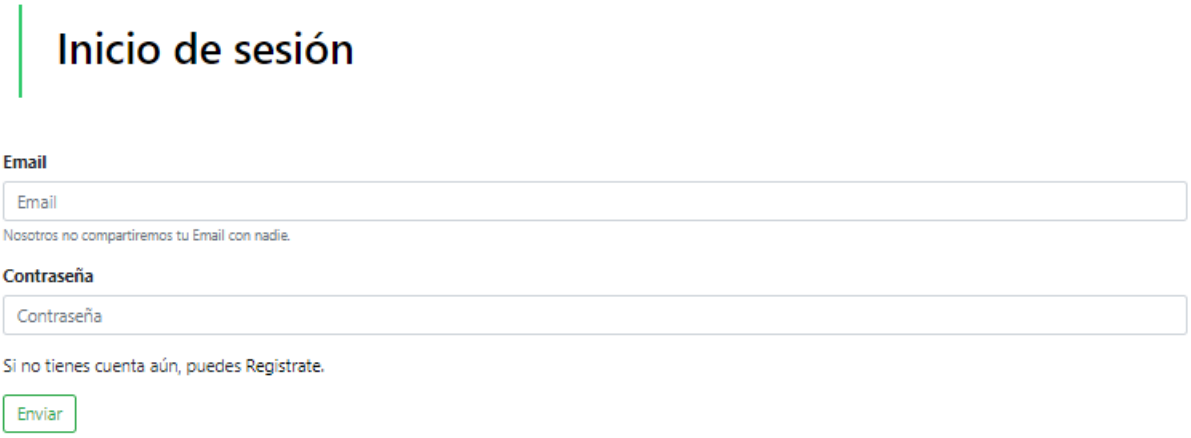


Figura 3.8: Entorno gráfico inicio de sesión

- Logout: Cuando un usuario quiere cerrar la sesión lo único que se hace es eliminar el token y volver a la página inicial. Esto último, es debido a por si el usuario cierra la sesión en algún momento, en donde sea necesario un token (modificar el usuario,...). Esta implementación sería:

```
export default {
  created(){
    this.$store.dispatch('destroyToken')
    this.$router.push({ name: 'index'})
  }
```

A su vez, se han añadido nuevas funcionalidades mejorando el control de sesión, estas son:

- Menú preparado para la sesión: el menú cambia en función de si estas iniciado o no. Esto es muy útil para poder mostrar todas las funcionalidades que nos aporta el poder tener información del usuario y, que veremos en los siguientes puntos. Para conseguir esto, se ha usado un 'if', el cual comprueba el valor que retorna una función la cual devuelve true o false. Esta funcionalidad en código sería:

```
loggedIn() {
  return this.$store.getters.loggedIn
},
```

- Menú especial para el administrador: se ha implementado una pestaña especial para aquel usuario que es el administrador de la página (esta pestaña se oculta en caso de que no este logueado como tal). Para ello, se ha hecho uso de la store de Vue, en la que sea especificado el correo del administrador. A su vez, se ha implementado una función, la cual devuelve si el usuario actual es el administrador o no. Estas funciones que se han comentado serían:

Página de Header

```
isAdmin() {
  return this.$store.getters.isAdmin
}
```

Store de Vue

```
cambioultima() {
      this.numeropagina = this.numero;
},
```

Página de modificación de usuario

Como ya sabemos, un usuario suele cambiar su información por algún motivo (cambio de descripción, equivocación,...). Para ello, se ha implementado una página web que permita cambiar la información de uno a más campos. Para conseguir esto, solo es necesario retocar el o los campos requeridos y, posteriormente, darle a guardar. Esta información será presentada a través de un formulario simple y limpio. Este formulario sería:

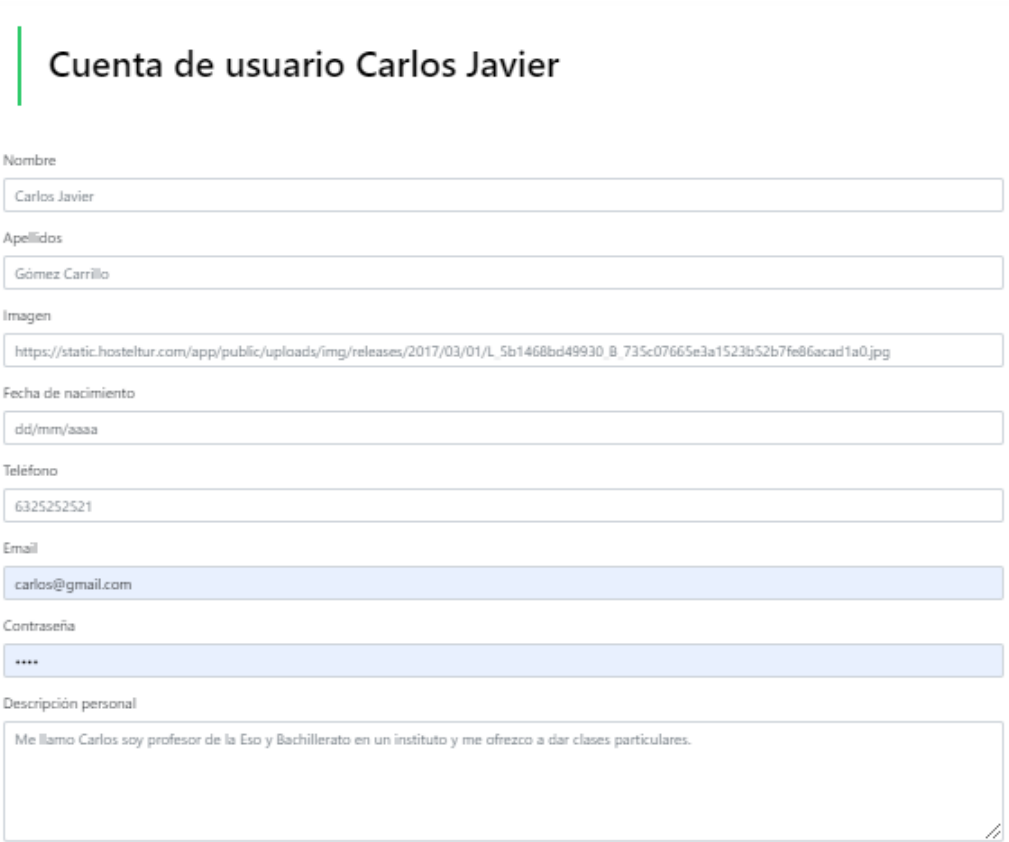


Figura 3.9: Entorno gráfico Modificación cuenta de usuario

Página de Perfil

Se ha creado una página web a modo de descripción del perfil de un usuario. Con esta página web se pretende que el cliente conozca más al anunciante. A su vez, en cada anuncio de una persona podremos encontrar un pequeño resumen sobre el usuario, esta información será más detallada en su perfil. Este perfil contiene:

- Información importante del usuario para el cliente (nombre, apellidos,email,...). Esta sección sería:

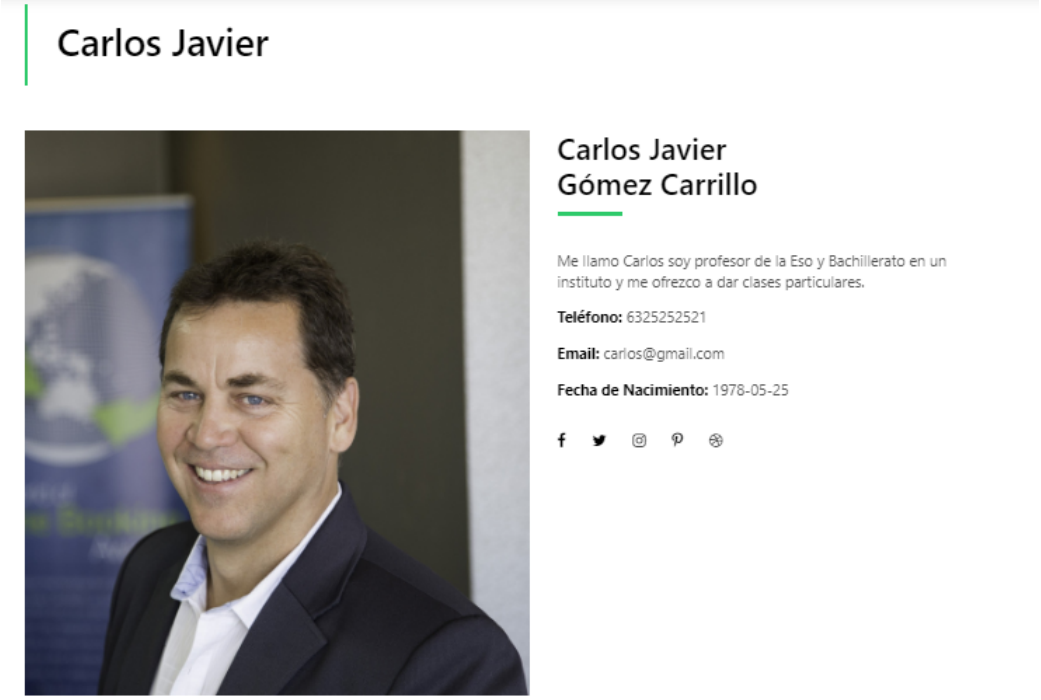


Figura 3.10: Pagina que muestra los datos del usuario

- Anuncios que el usuario tiene subidos a la página web.

3.2.5. Frontend Productos

Página de inserción de anuncios

En la parte referida a los productos es necesario poder incorporar nuevos anuncios para que los clientes puedan verlos. Para ello, se han pensado diversas opciones para conseguir tal fin, llegando a las conclusiones siguientes:

- Aunque muchos portales permiten subir anuncios sin loguearse se ha llegado a la conclusión de que es mejor estar logueado en el sistema para poder vender o dar servicio. Esto se debido a que, el usuario podría ver el perfil del anunciante y con ello, la página se haría más fiable y más atractiva a nuevos clientes.
- Cada anuncio estará ligado a una cuenta y, por tanto, el propietario de la cuenta podrá insertar, modificar o eliminar sus anuncios.
- Es necesario una página de administrador, debido a que hay que revisar periódicamente los anuncios, ya que podrían ser falsos o estar mal redactados. Para este fin, ha de crear una nueva pestaña en el menú, que sólo se despliegue si eres el administrador de la página web.

Para conseguir los objetivos plateados se han creado dos páginas idénticas (administrador y usuario), la única diferencia es que una de ellas sólo mostrará los anuncios del usuario logueado y, la otra, mostrará todos los anuncios del sistema. Este concepto siempre visto desde la escalabilidad y sencillez de la página al usuario, es decir, para

mejorar la escalabilidad se ha hecho uso de la paginación (ver apartado 3.2.3) y para ayudar al usuario dispondremos de un pequeño buscador (ver apartado 3.2.5).Para conseguir introducir un nuevo anuncio, se ha hecho uso de un pequeño formulario que cambia en función del tipo de anuncio que se requiera subir. Este formulario sería:

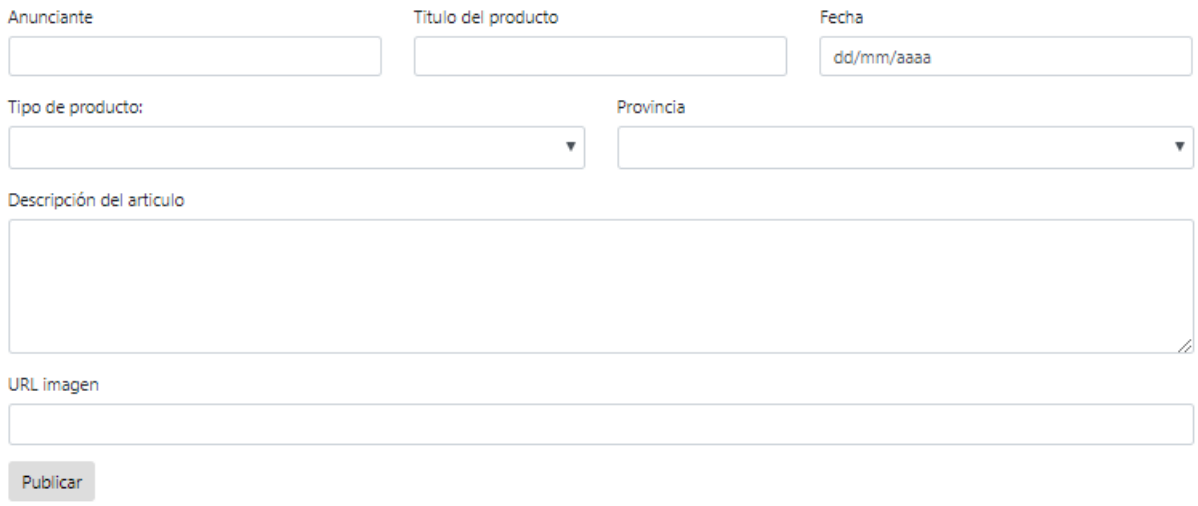


Figura 3.11: Página para insertar anuncios

A su vez se ha creado una tabla para verificar si todos los datos han sido introducidos correctamente e incorporar la posibilidad de eliminar o modificar un anuncio. Lo primero, se consigue clicando sobre el botón 'eliminar' y, lo segundo, es tan fácil como clicar 'modificar' y este anuncio se incorporará sobre el mismo formulario descrito anteriormente, donde podremos modificar todos los campos bajo nuestra criterio.Esta tabla sería:

Figura 3.12: Entorno gráfico de muestra de anuncios en página de nuevo anuncio o administrador

Página inicial

En cuanto a la página inicial del proyecto, se ha intentado crear una página limpia y atractiva, dado que es lo primero que verá el usuario al entrar en esta. Para ello, he realizado:

- Un carrusel con las noticias más relevantes de la fecha en la educación. Para ello, se ha hecho uso de un carrusel de Boostrap.
- Una pequeña y breve explicación de qué es la página web y cuáles son sus principales funcionalidades. Esta explicación sería:

Figura 3.13: Explicación proyecto

- Tres carruseles sobre los anuncios de la página web (uno para noticias, otro para apuntes y otro para clases particulares) implementados con Boostrap y modificados para lograr incorporar 3 anuncios por cada una de las páginas. Uno de los carruseles que se han implementado sería:

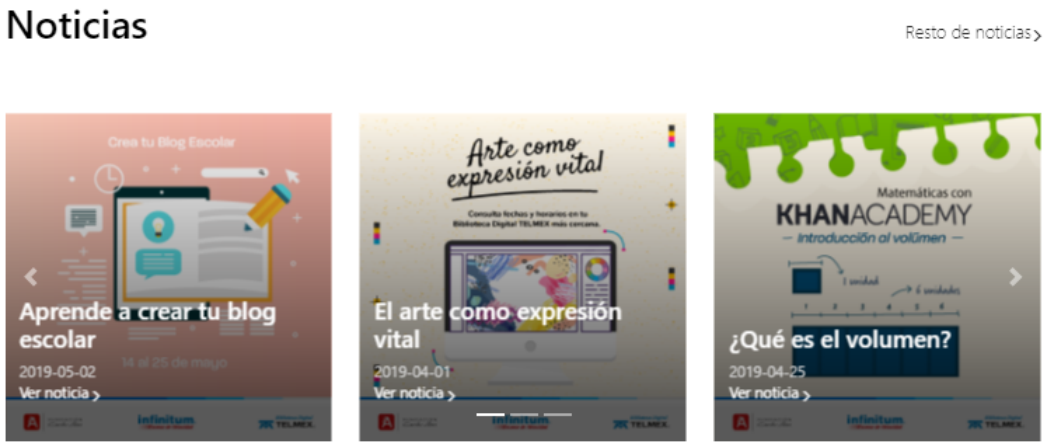


Figura 3.14: Carrusel anuncios

Página de productos (apuntes,noticias y clases particulares)

Se han creado tres páginas web referidas a cada anuncio. Sin embargo, los estilos de la web de apuntes y clases particulares son exactamente iguales. Esto es debido, a que son anuncios en los que si bien por vender o por ofrecer un servicio se requiere de centrarse en la venta. A su vez, como pasa en muchas otras páginas del proyecto se ha buscado la escalabilidad y la ayuda al usuario, por lo que se han implementado una paginación (ver apartado 3.2.3) de 6 elementos cada vez por pantalla y, a su vez, se ha implementado un pequeño filtro (ver apartado 3.2.5) para saber si el anuncio está en Tenerife o Gran Canaria.

La página web referida los apuntes y clases consta a primera vista de la información importante de un anuncio (titulo, precio, nivel, localidad y provincia). Asimismo,se ha querido implementar un estilo más moderno haciendo que la información aparezca tras pasar el cursor por encima. No obstante, en la versión de móvil y tablet esta opción se ha eliminado apareciendo la información a primera instancia.Esta estructura sería:

Figura 3.15: Estilo anuncios de noticias

A su vez, se ha incorporado un estilo propio para las noticias. Este estilo, igualmente muestra la información más importantes de estas. Esta estructura sería:

Figura 3.16: Estilo anuncios de clases particulares o apuntes

Buscador

El buscador consiste en el uso de los filtros para mejorar las experiencia del usuario, con esto conseguimos que el usuario tarde menos buscando en una serie de objetos. Su funcionalidad es básica, se incorporarán una serie de opciones en el que pueda introducir lo que se desea y la página mostrará lo más cercano a lo que el usuario pide.

En este proyecto se ha implementado un buscador adaptado a este tipo de tecnología. Esta, nos permite implementar un buscador en el que no es necesario recargar la página para mostrar los resultados, sino que, mientras que el usuario escoge el filtro deseado para la página va filtrando e intentando buscar el resultado más parecido. Todo esto, gracias a la función que viene incorporada por Vue.js, denominada onchange. Esta permite, que cuando exista un cambio en una variable se llame a una función y que pueda actualizar los valores de la página a lo que el usuario pide. En particular, se han implementado tres filtros estos son:

- Texto introducido por el usuario, el cual busca dentro del nombre del producto.
- El tipo de anuncio que quiere buscar (apuntes, clases particulares o noticias).
- Ciudad en la que se busca el anuncio.
- De forma gráfica sería:

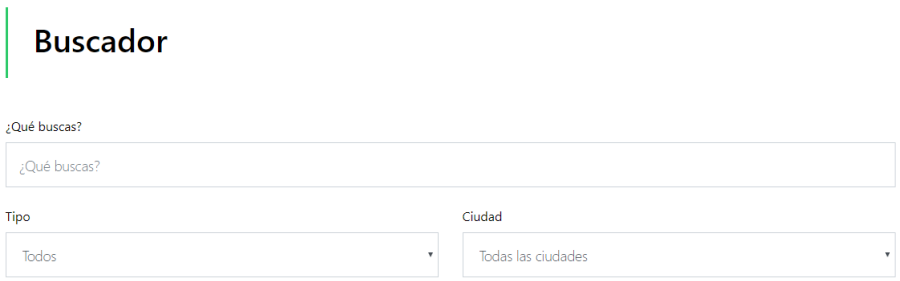


Figura 3.17: Página de búsqueda de anuncios

Página de contacto

Se ha implementado un página web, en donde se ha incorporado información ficticia sobre la empresa. En donde podremos encontrar:

- Un mapa del lugar de localización de la empresa.El mapa sería:

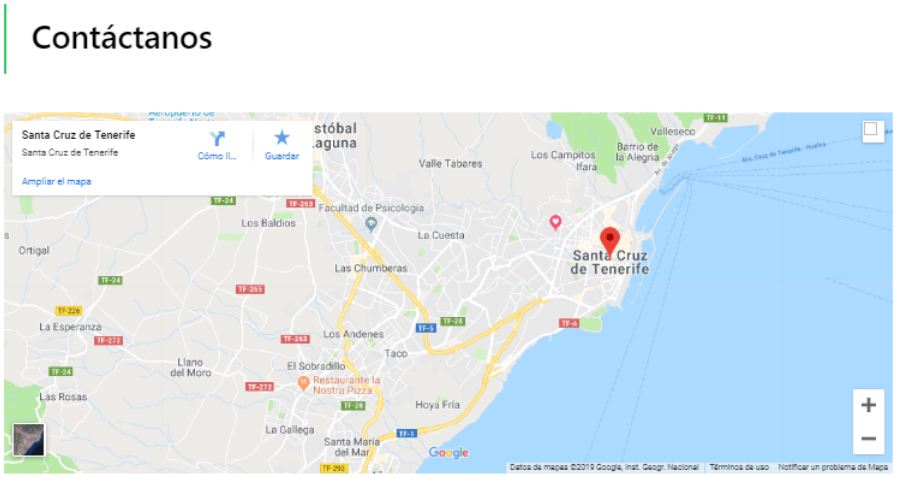


Figura 3.18: Mapa empresa

- Forma de contacto de la empresa (email y teléfono).
- Redes sociales de la empresa.

3.2.6. Página responsive

Una página resposive, quiere decir que está adaptada a el uso de cualquier dispositivo que se conecte a ella.

Como hemos comentado en el apartado 3.2.1 se ha usado Boostrap en la totalidad del proyecto. Por ello, se ha conseguido que el proyecto se vea como si su página hubiera sido diseñada para tal fin. Algunos ejemplos serían:

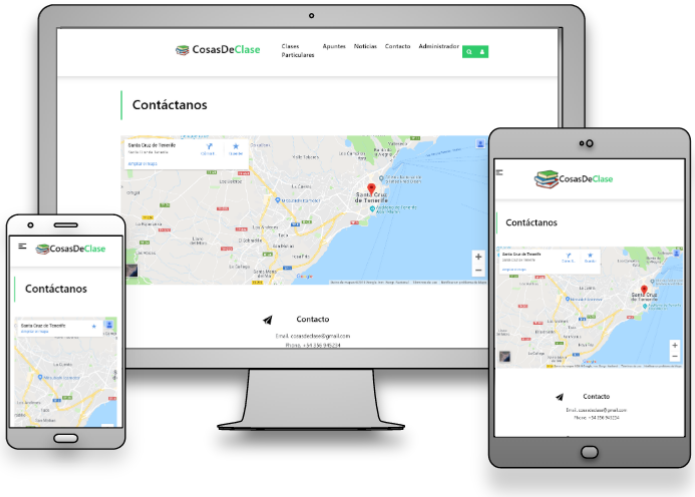


Figura 3.19: Página de contacto

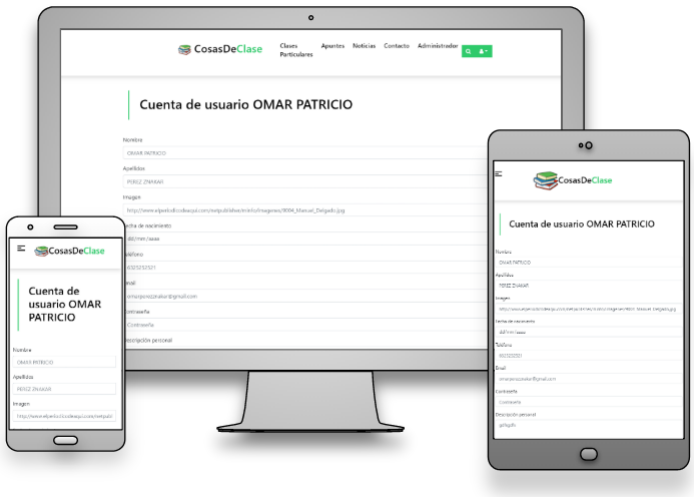


Figura 3.20: Página de modificación de cuenta de usuario

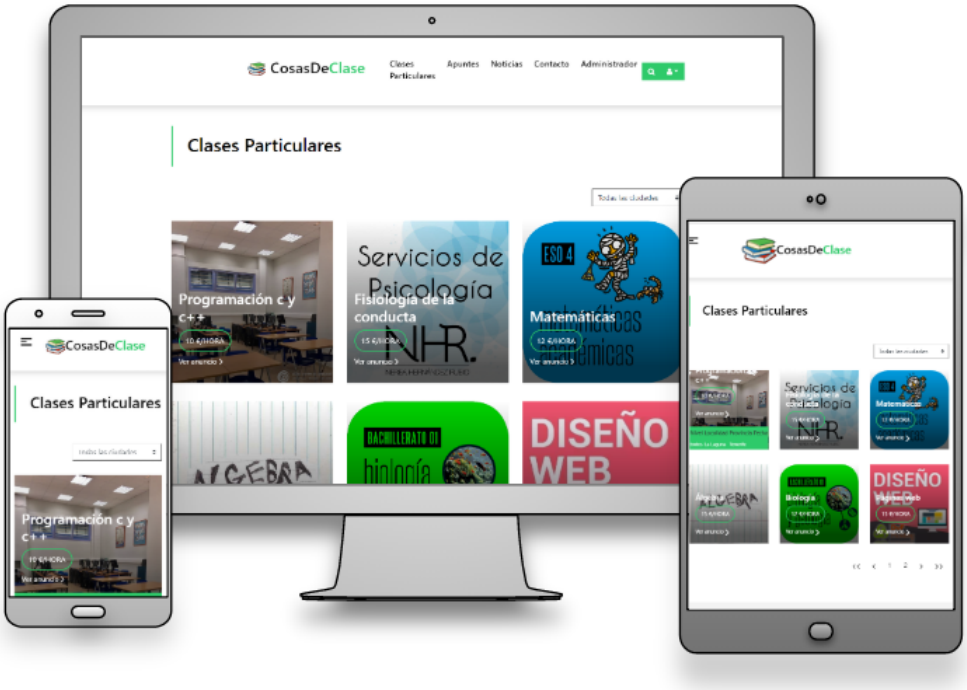


Figura 3.21: Página de anuncios sobre clases particulares

3.2.7. Test Frontend

Los test son un parte fundamental del desarrollo de una aplicación. Cuando desarrollamos los test para la parte del frontend, buscamos simular el uso de la página web, para así garantizarnos que lo que hemos hecho se realiza tal y como queremos.

En este proyecto se han realizado los test con Jest. Esta tecnología, se basa en comprobar el correcto funcionamiento del Javascript de una página a través de ünit test"(test unitarios). Esta idea, consiste en realizar primero el código y, posteriormente, con todo hecho realizar las pruebas que se necesiten. A partir de las ideas anteriormente citadas se han realizado los test para este proyecto. En concreto, se han dividido en dos partes los test de los usuarios y los test de los productos. Estos se comentarán en los apartados siguientes. Además, un ejemplo de estos test se encuentra en A.2.

A su vez, se ha creado un nuevo apartado en el package.json para recordar mejor su uso, Este apartado sería:

- "jest": "jest ./app/test/vueTests"

Gracias a lo anterior, tan sólo es necesario ejecutar "npm run jest". Este comando generaría la siguiente salida:

Figura 3.22: Salida test con Jest

Test Frontend Usuario

En este apartado se han realizado los test correspondiente a los usuarios siguiendo la filosofía de realizar lo que un usuario haría. Esto, nos ayuda a saber si todo funciona correctamente. Para conseguir este propósito, se ha hecho un registro de un usuario creado, posteriormente, se ha realizado el login y, por último, se han modificado los datos del usuario. Para realizar esto, se han implementado los siguientes pasos:

- Registro: se ha comprobado que los componentes de HTML más importante existen. Posteriormente, se han rellenado los valores de la sesión y, se ha comprobado que al clicar se envíen correctamente. Este test, se puede encontrar en este enlace[4].

[4] https://github.com/Omar97perez/TFG-ULL-CosasDeClase-MEVN/blob/master/app/test/vue_tests/Usuarios/Registro.test.js

- Login: se ha comprobado que los apartados más importantes existen. A su vez, se han introducido los datos del apartado anterior, para simular un inicio de sesión. Este test se puede encontrar en este enlace[5].

- Modificar usuario: se ha comprobado que las etiquetas más relevantes existen. A su vez, se han modificado los datos de la sesión y, se ha comprobado, que se han cambiado correctamente.Este test se puede encontrar en este enlace[6].

Test Frontend Anuncios

En este aparatado se han realizado los test referidos a los anuncios siguiendo la filosofía de la simulación de uso de este tipo de dato. Para conseguir este propósito, se han realizado los test en el apartado de incorporación, seguidamente los de visión de los anuncios y, por último, del buscador. Para realizar esto, se han implementado los siguientes pasos:

- Incorporación de datos: se ha constatado que todos los elementos relevantes existen. A su vez, se ha comprobado que la paginación funciona correctamente (tamaño de página, página inicial,..). Y, por último, se han realizado las pruebas correspondientes a la introducción de un objeto a la base de datos. Este test se puede encontrar en este enlace[7].

- Ver anuncios: se ha constatado que la paginación funciona correctamente (tamaño de página, página inicial,..). A su vez, se ha verificado que el filtrado de las ciudades funciona adecuadamente. Y, por último, se ha comprobado que los anuncios mostrados son del tipo que es necesario (clases particulares,apuntes o noticias). Este test se puede encontrar en este enlace[8].

- Buscador: se ha verificado que las etiquetas más relevantes existen. A su vez, se ha comprobado que la paginación actúa como se desea (tamaño de páginas, pagina inicial,...). Y, por último, se ha constatado el funcionamiento de los apartados del buscador (texto de filtrado, tipo y ciudad del anuncio). Este test se puede encontrar en este enlace[9]

[5] https://github.com/Omar97perez/TFG-ULL-CosasDeClase-MEVN/blob/master/app/test/vue_tests/Usuarios/Login.test.js

[6] https://github.com/Omar97perez/TFG-ULL-CosasDeClase-MEVN/blob/master/app/test/vue_tests/Usuarios/ModUsuario.test.js

[7] https://github.com/Omar97perez/TFG-ULL-CosasDeClase-MEVN/blob/master/app/test/vue_tests/Productos/Administrador.test.js

[8] https://github.com/Omar97perez/TFG-ULL-CosasDeClase-MEVN/blob/master/app/test/vue_tests/Productos/Apuntes.test.js

[9] https://github.com/Omar97perez/TFG-ULL-CosasDeClase-MEVN/blob/master/app/test/vue_tests/Productos/Buscador.test.js

Capítulo 4

Guía de uso

En este capítulo se intentará hacer una pequeña guía de uso de la aplicación para intentar resolver la mayor parte de los problemas o dudas que le surja al usuario a la hora de usarla.

4.1. Acceder a la aplicación

Para acceder a la aplicación es tan fácil como clicar en los siguientes links:

- Página web: `https://cosasdeclase.herokuapp.com/`
- Código fuente: `https://github.com/Omar97perez/TFG-ULL-CosasDeClase-MEVN`
- Explicación proyecto: `https://omar97perez.github.io/TFG-ULL-CosasDeClase-MEVN/`

4.2. Datos generales

En primera instancia al acceder a la página web nos encontraremos con la página de inicio, la cual contiene:

- Explicación página web (para mayor explicación ver 3.21).
- Carrusel de anuncios (para mayor explicación ver 3.14).

A su vez, cabe destacar que existe una pequeña restricción a ciertas características no esenciales para un usuario que esta interesado en la adquisición de un producto o servicio. No obstante, para eliminar esta restricción es muy sencillo, tan solo, es necesario registrarse e iniciar sesión en el sistema (ver apartado 4.3.2). A su vez, para mejorar la intimidad del usuario, el perfil de este sólo será visualizado en caso de que suba un anuncio. Las páginas restringidas bajo las condiciones anteriores son:

- Buscador.
- Perfil del anunciante. No obstante, el cliente puede ver los datos de interés del anunciante, la única función restringida es la de ver todos los demás anuncios del anunciante.
- Página para subir anuncios.
- Página para modificar datos de la sesión.
- Perfil de usuario Propio.

4.3. Página web

4.3.1. Registro de un nuevo usuario

Para poder hacer uso de la sesión es necesario registrarse previamente. Para conseguir este paso hay que seguir las siguientes indicaciones:

1. Clicar en el icono de usuario. Este se encuentra en el menú en el extremo derecho.

Figura 4.1: Lugar donde clicar icono usuario

2. Clicar en el apartado de registrase.

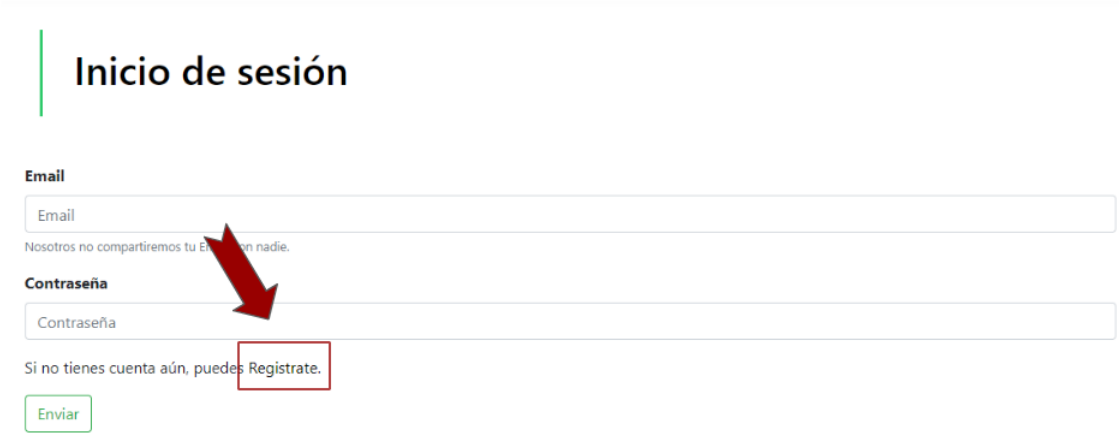


Figura 4.2: Lugar donde ir a página registro

3. Rellenar los datos y clicamos en el botón de registrase.

Figura 4.3: Página de registro

4.3.2. Inicio de Sesión

Tras realizar el paso 4.3.1 seguiremos los siguientes pasos:

1. Clicar en el icono de usuario. Este se encuentra en el menú en el extremo derecho. Ver figura 4.1.
2. Rellenamos los datos y clicamos en enviar.

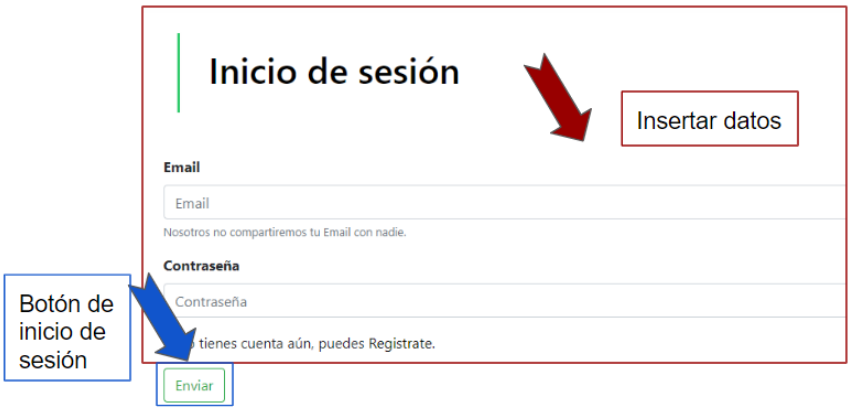


Figura 4.4: Página de Inicio de sesión

4.3.3. Cerrar sesión

Tras realizar el paso 4.3.2 iremos al menú al icono de usuario y en el desplegable clicaremos en "Logout".

Figura 4.5: Cerrar sesión

4.3.4. Modificar cuenta de usuario

Tras realizar el paso 4.3.2 seguiremos los siguientes pasos:

1. Clicar en el icono de usuario. Este se encuentra en el menú en el extremo derecho. Y este, se desplegara con varias opciones. Entre estas, elegimos la opción de "Modificar usuario".

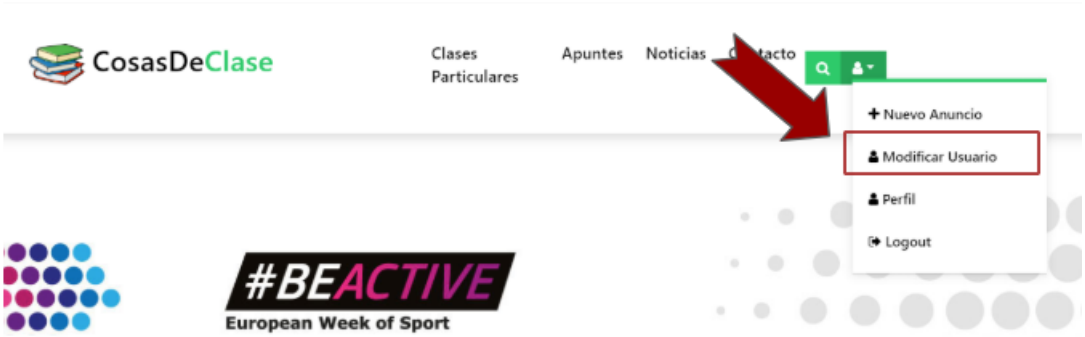


Figura 4.6: Como encontrar la pestaña de modificar usuario

2. Rellenamos los datos y clicamos en "guardar datos". Cabe destacar, que solo hay que modificar el campo que se requiere no todos. Sin embargo, en caso de querer eliminar la cuenta le damos a eliminar usuario.

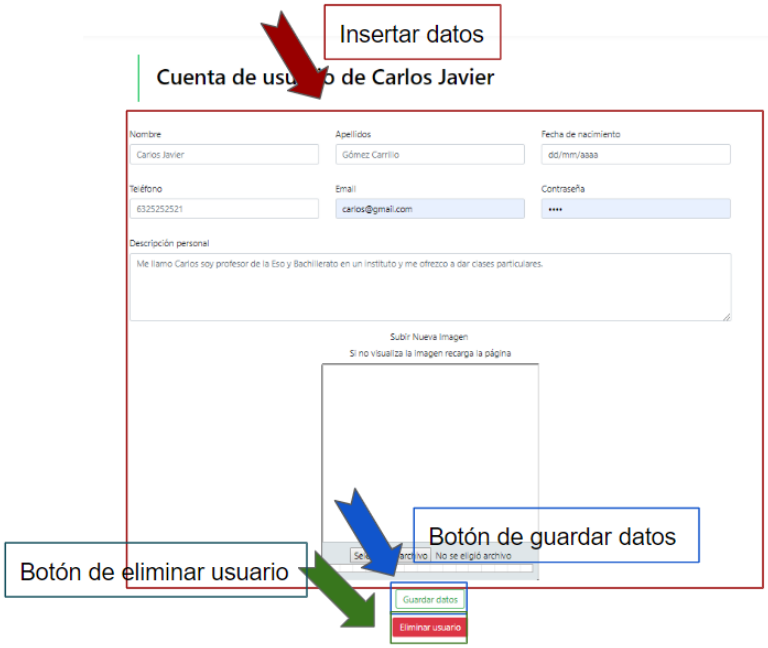


Figura 4.7: Página de modificar Usuario

4.3.5. Subir anuncio

Tras realizar el paso 4.3.2 seguiremos los siguientes pasos:

1. Clicar en el icono de usuario. Este se encuentra en el menú en el extremo derecho. Y este, se desplegará con varias opciones. Entre estas, elegimos la opción de "Nuevo anuncio".

Figura 4.8: Como encontrar la pestaña de subir anuncio

2. Rellenamos los datos del anuncio y clicamos en publicar anuncio.

Figura 4.9: Página de subir anuncio

4.3.6. Editar/eliminar anuncio

Tras realizar el paso 4.3.2 seguiremos los siguientes pasos:

1. Acceder a la pestaña correspondiente. Ver figura 4.8.
2. Buscamos el anuncio en cuestión y clicamos en eliminar o editar(según se desee). Cabe destacar, que si decidimos editar el anuncio los valores de este aparecerán en el formulario de la figura 4.9.

Figura 4.10: Eliminar o editar anuncio

4.3.7. Buscar anuncios

Tras realizar el paso 4.3.2 seguiremos los siguientes pasos:

1. Clicar en el icono del buscador. Este se encuentra en el menú en el extremo derecho.

Figura 4.11: Como encontrar la pestaña del buscador

2. En el formulario que nos sale tendremos que incorporar lo que deseemos buscar. Cabe destacar, que no es necesario rellenar todos los temas y, que no existe botón de buscar, dado que automáticamente el filtro se aplica a los anuncios.

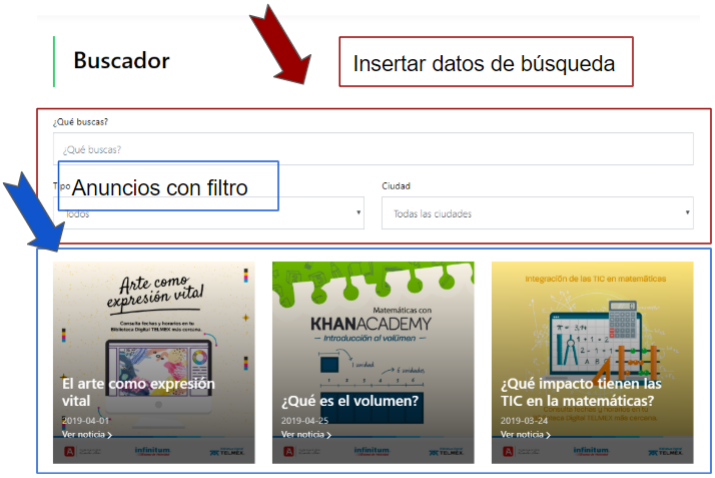


Figura 4.12: Página del buscador

Capítulo 5

Conclusiones y líneas futuras

5.1. Conclusiones

Esta aplicación surgió tras analizar uno de los problemas que presentan los estudiantes durante su vida académica. Este inconveniente se fundamenta principalmente en la necesidad de una persona al prestar sus servicios como profesor sobre un área de conocimiento, en donde la oferta está muy limitada. Además, de lo comentado, se amplió el atractivo de la página para no centrarse sólo en la búsqueda de un profesor sino también de recursos académicos de interés (apuntes, libros,...) e incluso noticias de índole académica.

Para resolver esta cuestión planteada, se intento usar las últimas tecnologías existentes en el campo de las aplicaciones web (sin olvidar la búsqueda de la sencillez e intuitividad característica de este tipo de proyecto). Además, se ha hecho uso de un control de versiones que proporciona seguridad a la hora de poder seguir desarrollando, ya que nos da la garantía de poder regresar a un punto anterior en caso de que sea necesario. A su vez, se intentó realizar una herramienta útil siguiendo una metodología correcta de trabajo. Para conseguir este fin, se han usando test (tanto en el backend como en el frontend) junto con Webpack, uniendo estas dos tecnologías se ha podido abarcar una gran cantidad de posibles fallos que se han podido solucionar rápidamente. Además, gracias a PurifiCss hemos podido optimizar el código CSS en gran cantidad dejándolo únicamente en aquello que se necesita.

A modo de cierre, cabría resaltar que se han podido resolver todos los problemas planteados siguiendo el plazo de tiempo propuesto. Además, se ha podido avanzar mucho más de lo que se esperaba, creando funcionalidades que en un principio no estaban impuestas.

5.2. Líneas futuras

Este proyecto ha seguido la idea de ayudar a las personas en su paso por la educación. El proyecto, tiene su primera versión totalmente acabada y, con esto, lo único que haría falta es que se divulgue por la rama académica de diversas formas, que podrían ser:

- Medios tradicionales: una forma de publicitarla sería a través de carteles colocados en los sitios permitido.
- Medios digitales: una forma de publicitarla sería mediante publicidad en medios digitales de uso por los jóvenes (mayoría de colectivos de interés). Algunos de estos sitios serían: Instagram, Twitter, Youtube,...

- Medio oral: divulgación a través de distintas fuentes que se trasmiten de persona a persona (charlas,...).

No obstante, aunque tenga su primera versión siempre hay propuestas de mejora. Algunas de estas opciones son:

- Corrección de errores: nada está totalmente correcto hasta que una masa de usuarios pruebe el proyecto. Por ello, en un futuro es necesario corregir los errores que vayan apareciendo.
- Registro e inicio de sesión con redes sociales: gracias a Google cabe la posibilidad de que un usuario no necesite escribir sus datos, debido a que estos pueden extraerse de alguna de las redes sociales que este disponga.
- Mejorar la subida de imagen: se podría hacer que no sólo el usuario deba clicar para seleccionar el archivo y moverse a la carpeta, sino que incluso pueda arrastrar la imagen en cuestión y, esta realice los mismos procesos que hasta ahora.
- Implementar un sistema de markdown: una forma de mejorar la página web actual consistiría implementar un markdown (editor de texto). Esto, sería muy favorable para el usuario, ya que este podría modificar su anuncio a su gusto haciéndolo más atractivo para el cliente.
- Mejorar el sistema de filtrado: la búsqueda en la actualidad sigue un diseño tradicional (únicamente buscando en los títulos de los anuncios). A pesar de que es muy eficaz se podría mejorar e implementar una búsqueda que a su vez, también busque en etiquetas generadas por los anunciantes (tal y como los hace Youtube) o incluso que pueda buscar dentro del texto del anuncio.
- Mejorar los tipos de búsqueda: actualmente solo se filtra por anuncios. Sin embargo, también sería recomendable que pueda buscar a los anunciantes que dispongan de uno o más anuncios.

A modo de inciso se destaca que, aunque existan mejoras no serían imprescindibles, ya que a pesar de ayudar al usuario no son totalmente necesarias para que los clientes puedan realizar la función de este proyecto.

Capítulo 6

Summary and Conclusions

6.1. Conclusions

This application arises after the analysis of one of the problems presented by the students during their academic life. This drawback is mainly based on the need of a person to provide services as a teacher over an area of interest, where the offer is very limited. Additionally, the attractiveness of the site was expanded not only to focus on the search of a teacher but also other academic resources (notes, books, ...) and even academic related news.

To resolve this issue, I will try to use the latest technologies in the field of web applications (not forgetting the pursuit of simplicity and intuitiveness feature of this type of project). It has also made use of a control version that provides security as the program is being developed, as it gives us the guarantee of returning to an earlier point if necessary. In turn, it is essential to create a useful tool following a correct methodology. To achieve this I have been using TDD programming with tests (both at the backend and the frontend) using Webpack. The combination of these two technologies allows me to cover a large number of errors, which have been solved quickly.

To conclude, all the problems have been solved following the proposed plan. In addition, it has also been possible to achieve even more progress than expected, thus creating features that were not initially planned.

6.2. Future Work

The project has followed the idea of helping people on their way through education. The project has been fully implemented in its first version. All we need is to promote the use by the academic community. We can advertise the application with the following strategies:

- Traditional media: a way to advertise it would be through posters in the allowed sites.
- Digital media: a way to advertise it be through the digital media used mostly by young people (most collective interest). Some of these sites are: Instagram, Twitter, Youtube, etc.
- Oral means: dissemination through various sources that are transmitted from person to person (Lectures, conferences, etc).

However, even if the first version is a success, there are always suggestions for improvement. Some of these options are:

- Bugfix: nothing is quite correct until a mass of users test the project. Therefore, in the future it is necessary to correct errors that appear.
- Registration and login with social networks through Google auth2 protocol, which makes possible for users to use of all the features Google provide and all social networks related to Google, such as not writing the user and password in each login.
- Improve image dragging: it could make not only the user must click to select the file and move to the folder, but can even drag the image in question and this make the same processes so far.
- Markdown editor: a way to improve the current website would implement a markdown (text editor). This would be very suitable for the user, since he might change his ad to make it more attractive for the customer.
- Improve the filtering system: the search now follows a traditional design (only looking at the titles of the ads). Although it is very effective, it could improve and implement a search which in turn also looks for labels generated by advertisers (such as Youtube does) or you can search within the text of the ad.
- Improve search types: currently only filtered by ads. However, it would be advisable to look for advertisers that have one or more ads.

Finally, all the improvements mentioned above are just non relevant features that focus on making the use of the application more user friendly; all of these are just extra features that are not necessary for the performance of the program.

Capítulo 7

Presupuesto

7.1. Justificación del presupuesto

Llegados a este punto se hace una baremación del coste que supone el desarrollo de este proyecto. Además, se ha calculado el precio de creación y mantenimiento de un año de la aplicación web. Para llevarlo a cabo, se ha tenido en cuenta las tarifas de un programador junior. A continuación, se presenta una tabla con el presupuesto:

TIPO	CANTIDAD	COSTE UNIDAD	COSTE TOTAL
Creando Infraestructura para el desarrollo	40 horas	15 €/h	600 €
Prototipo del proyecto	120 horas	12 €/h	1440 €
Despliegue	120 horas	12 €/h	1440 €
Backend	80 horas	15 €/h	1200 €
Frontend	160 horas	12 €/h	1920 €
Mongodb Atlas	8928 h/año	0,08 €/h	714,24 €/año
Despliegue con Heroku	12 meses	22,24 €/mes	266.88 €/año
Gestos de imágenes Cloudinary	12 meses	0 €/mes	0 €/año
TOTAL			6141,12 €

Cuadro 7.1: Presupuesto

Como podemos observar en la tabla descrita anteriormente, se hace una relación directa entre las horas dedicadas a cada fase de creación del proyecto y la tarifa estándar aplicada a ese tiempo. A su vez, se refleja un coste provisional que correspondería con el uso de las aplicaciones web externas *Mongodb Atlas, Heroku y Cloudinary*, las cuales resultan imprescindibles para su uso. Asimismo, debe aclararse que las tarifas de las herramientas externas son básicas, debido que sólo sustentan un número reducido de usuarios (Heroku) o espacio de almacenamiento (Mongodb Atlas y Cloudinary). Por tanto, en caso de ampliarse a tarifas más amplias, se incrementaría también el presupuesto expuesto.

Apéndice A

Apéndice: Test

A.1. Test Backend

```
/*****************************************************************************
*Productos_test.js
*
*****************************************************************************
*
* AUTOR: Omar Patricio Pérez Znakar
*
* FECHA: 21/04/2019
*
*
* DESCRIPCION: Test para comprobar que los productos se usas correctamente en el backend.
*
*
*****************************************************************************/

describe("Pruebas sobre un Producto de la página web", function(){
    describe("Prueba de creación de un producto", function(){
        it("comprobando que se crea correctamente", function(){
            let producto_test = {
                id: 34567,
                anunciante: "nombre",
                fecha: "06/03/2019",
                titulo: "test_name",
                foto: "test_foto",
                descripcion: "test_descripcion",
                tipo: "test_tipo",
                nivel: "Todos",
                provincia: "Tenerife" ,
                localidad: "Candelaria",
                precio: 40,
            }
            request.post("/").send(producto_test).expect("Producto guardado");
        });
    });
    describe("Prueba de modificación de un producto", function(){
        it("comprobando que se modifica correctamente", function(){
            let producto_test = {
              id: 34567,
              anunciante: "nombre",
              fecha: "06/03/2019",
              titulo: "test_name",
              foto: "test_foto",
```

```
          descripcion: "test_descripcion",
          tipo: "test_tipo",
          nivel: "Todos",
          provincia: "Tenerife" ,
          localidad: "Candelaria",
          precio: 40,
        }
        request.post("/").send(producto_test).expect("Producto guardado");
        request.put("/:id").send(producto_test).expect("Producto actualizada");
    });
});
describe("Prueba de borrado de un producto", function(){
    it("comprobando que se borra correctamente", function(){
        let producto_test = {
          id: 34567,
          anunciante: "nombre",
          fecha: "06/03/2019",
          titulo: "test_name",
          foto: "test_foto",
          descripcion: "test_descripcion",
          tipo: "test_tipo",
          nivel: "Todos",
          provincia: "Tenerife" ,
          localidad: "Candelaria",
          precio: 40,
        }
        request.post("/").send(producto_test).expect("Producto guardado");
        request.delete("/:id").send(producto_test).expect("Producto eliminado ");
    });
});
describe("Prueba de obtención de un producto", function(){
    it("comprobando que se obtiene correctamente", function(){
        let producto_test = {
          id: 34567,
          anunciante: "nombre",
          fecha: "06/03/2019",
          titulo: "test_name",
          foto: "test_foto",
          descripcion: "test_descripcion",
          tipo: "test_tipo",
          nivel: "Todos",
          provincia: "Tenerife" ,
          localidad: "Candelaria",
          precio: 40,
        }
        request.post("/").send(producto_test).expect("Producto guardado");
        request.get("/:id").send(producto_test).expect('Content-Type', /json/);
    });
});
describe("Prueba de obtención de productos", function(){
    it("comprobando que se obtienen correctamente", function(){
        let producto_test = {
          id: 34567,
          anunciante: "nombre",
          fecha: "06/03/2019",
          titulo: "test_name",
          foto: "test_foto",
          descripcion: "test_descripcion",
          tipo: "test_tipo",
          nivel: "Todos",
```

```
          provincia: "Tenerife" ,
          localidad: "Candelaria",
          precio: 40,
        }
        request.get("/").expect('Content-Type', /json/);
    });
});
describe("Pruebas sobre Productos.js", function(){
    it("comprobando existe objeto Productos", function(){
        assert.typeOf(productos_js.schema.obj, "object");
    });
    it("comprobando que campo id es de tipo Number", function(){
        assert.equal(productos_js.schema.paths.id.instance, "Number");
    });
    it("comprobando que campo anunciante es de tipo String", function(){
        assert.equal(productos_js.schema.paths.anunciante.instance, "String");
    });
    it("comprobando que campo fecha es de tipo Number", function(){
        assert.equal(productos_js.schema.paths.fecha.instance, "String");
    });
    it("comprobando que campo titulo es de tipo String", function(){
        assert.equal(productos_js.schema.paths.titulo.instance, "String");
    });
    it("comprobando que campo foto es de tipo String", function(){
        assert.equal(productos_js.schema.paths.foto.instance, "String");
    });
    it("comprobando que campo descripcion es de tipo String", function(){
        assert.equal(productos_js.schema.paths.descripcion.instance, "String");
    });
    it("comprobando que campo tipo es de tipo String", function(){
        assert.equal(productos_js.schema.paths.tipo.instance, "String");
    });
    it("comprobando que campo nivel es de tipo String", function(){
        assert.equal(productos_js.schema.paths.nivel.instance, "String");
    });
    it("comprobando que campo provincia es de tipo String", function(){
        assert.equal(productos_js.schema.paths.provincia.instance, "String");
    });
    it("comprobando que campo localidad es de tipo Number", function(){
        assert.equal(productos_js.schema.paths.localidad.instance, "String");
    });
    it("comprobando que campo precio es de tipo Number", function(){
        assert.equal(productos_js.schema.paths.precio.instance, "Number");
    });
});

it("comprobando que existe método get sobre /", function(){
    assert.equal(producto_js.stack[0].route.path, "/");
    assert.equal(producto_js.stack[0].route.methods.get, true);
});
it("comprobando que existe método post sobre /", function(){
    assert.equal(producto_js.stack[2].route.path, "/");
    assert.equal(producto_js.stack[2].route.methods.post, true);
});
it("comprobando que existe método get sobre /:id", function(){
    assert.equal(producto_js.stack[1].route.path, "/:id");
    assert.equal(producto_js.stack[1].route.methods.get, true);
});
it("comprobando que existe método put sobre /:id", function(){
    assert.equal(producto_js.stack[3].route.path, "/:id");
```

```
        assert.equal(producto_js.stack[3].route.methods.put, true);
    });
    it("comprobando que existe método delete sobre /:id", function(){
        assert.equal(producto_js.stack[4].route.path, "/:id");
        assert.equal(producto_js.stack[4].route.methods.delete, true);
    });
});
```

A.2. Test Frontend

```
/*****************************************************************************
*
 * Registro.test.js
*
*****************************************************************************
*
* AUTOR: Omar Patricio Pérez Znakar
*
* FECHA: 21/04/2019
*
*
* DESCRIPCION: Test para comprobar que el registro del usuario funciona perfectamente.
*
*
*****************************************************************************/

describe("Registro.vue", () => {

  const wrapper = mount(Registro);

  it("Comprobando que se encuentra campo nombre de formulario", () => {
    expect(wrapper.html()).toContain('id="name"')
  });

  it("Comprobando que se encuentra campo apellidos de formulario", () => {
    expect(wrapper.html()).toContain('id="surname"')
  });

  it("Comprobando que se encuentra campo fecha de nacimiento de formulario", () => {
    expect(wrapper.html()).toContain('id="birthdate"')
  });

  it("Comprobando que se encuentra campo email de formulario", () => {
    expect(wrapper.html()).toContain('id="email"')
  });

  it("Comprobando que se encuentra campo contraseña de formulario", () => {
    expect(wrapper.html()).toContain('id="password"')
  });

  it("Comprobando que se encuentra botón de registrarse", () => {
    expect(wrapper.contains('button')).toBe(true)
  });

  it("Comprobando que hay etiqueta de formulario", () => {
    expect(wrapper.contains('form')).toBe(true)
  });

  it("Comprobando introduccion de nombre y que no tenga errores", () => {
```

```
  wrapper.setProps({ name: 'test' })
  expect(wrapper.vm.name).toBe('test')
  expect(wrapper.find('.error').exists()).toBe(false)
});

it("Comprobando la introducción de apellidos y que no tenga errores", () => {
  wrapper.setProps({ surname: 'test_1 test_2' })
  expect(wrapper.vm.surname).toBe('test_1 test_2')
  expect(wrapper.find('.error').exists()).toBe(false)
});

it("Comprobando la introducción de email y que no tenga errores", () => {
  wrapper.setProps({ email: 'test@test.com' })
  expect(wrapper.vm.email).toBe('test@test.com')
  expect(wrapper.find('.error').exists()).toBe(false)
});

it("Comprobando la introducción de contraseña y que no tenga errores", () => {
  wrapper.setProps({ telephone: '6546546544' })
  expect(wrapper.vm.telephone).toBe('6546546544')
  expect(wrapper.find('.error').exists()).toBe(false)
});

it("Comprobando la introducción de contraseña y que no tenga errores", () => {
  wrapper.setProps({ password: '1234567' })
  expect(wrapper.vm.password).toBe('1234567')
  expect(wrapper.find('.error').exists()).toBe(false)
});

it("Comprobando la introducción del párrafo y que no tenga errores", () => {
  wrapper.setProps({ paragraph: 'Descripción test' })
  expect(wrapper.vm.paragraph).toBe('Descripción test')
  expect(wrapper.find('.error').exists()).toBe(false)
});

it("Comprobando la introducción de la imágen y que no tenga errores", () => {
  wrapper.setProps({ image: 'Descripcióntest' })
  expect(wrapper.vm.image).toBe('Descripcióntest')
  expect(wrapper.find('.error').exists()).toBe(false)
});


it("Comprobando la introducción de fecha de nacimiento y que no tenga errores", () => {
  wrapper.setProps({ birthdate: new Date("1997-03-25") })
  expect(wrapper.vm.birthdate).toEqual(new Date("1997-03-25"))
  expect(wrapper.find('.error').exists()).toBe(false)
});

it("Comprobando la introducción de genero de nacimiento y que no tenga errores", () => {
  new Date
  wrapper.setProps({ genre: 'otro' })
  expect(wrapper.vm.genre).toBe('otro')
  expect(wrapper.find('.error').exists()).toBe(false)
});

it("Comprobando que al registrarse no existan errores", () => {
  const button = wrapper.find('button')
  button.trigger('click')
  expect(wrapper.find('.error').exists()).toBe(false)
});
```

```
  it("Comprobando la introducción de genero de nacimiento y que no tenga errores", () => {
    expect(wrapper.vm.genre).toBe('otro')
  });
});
```

Bibliografía

[1] Breve explicación de cloudinary y sus funcionalidades. http://www.nerdilandia.com/cloudinary-una-forma-inteligente-de-gestionar-y-manipular-imagenes-en-linea/. Accessed: 2019-04-12.

[2] Desarrollo web con stacks de software. https://www.ionos.es/digitalguide/servidores/know-how/desarrollo-web-con-stacks-de-software/. Accessed: 2019-04-18.

[3] Descripción de backend. https://rafarjonilla.com/que-es/backend/. Accessed: 2019-04-19.

[4] Descripción de express y node.js. https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction. Accessed: 2019-04-19.

[5] Descripción de integración continua. https://aws.amazon.com/es/devops/continuous-integration/. Accessed: 2019-04-18.

[6] Descripción de integración continua. http://www.robertocrespo.net/kaizen/aprende-a-montar-un-entorno-de-integracion-continua-i/. Accessed: 2019-04-18.

[7] Descripción de metodologías ágiles (SCRUM). https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/. Accessed: 2019-04-18.

[8] Descripción de nodejs. https://www.netconsulting.es/blog/nodejs/. Accessed: 2019-04-18.

[9] Descripción de stack de desarrollo. https://www.beeva.com/beeva-view/desarrollo/introduccion-al-stack-mean-o-como-hacer-un-desarrollo-end-end/. Accessed: 2019-04-18.

[10] Descripción sistema distribuido travis. https://www.genbeta.com/desarrollo/travis-ci-sistema-distribuido-de-integracion-continua-libre-integrado-con-github. Accessed: 2019-04-19.

[11] Descripción y como usar express. http://www.nodehispano.com/2012/01/express-el-framework-web-para-nodejs/. Accessed: 2019-04-19.

[12] Descripción y comparación entre frontend y backend. https://devcode.la/blog/frontend-y-backend/. Accessed: 2019-04-19.

[13] Principales diferencias entre sql y nosql. https://blog.pandorafms.org/es/nosql-vs-sql-diferencias-y-cuando-elegir-cada-una/. Accessed: 2019-04-18.

[14] Puntos a favor del stack MEVN. https://www.genbeta.com. Accessed: 2019-04-19.

[15] Página oficial de cloudinary. https://cloudinary.com/. Accessed: 2019-06-05.

[16] Página oficial de express. https://expressjs.com/es/. Accessed: 2019-06-06.

[17] Página oficial de github. https://github.com/. Accessed: 2019-06-05.

[18] Página oficial de heroku. https://www.heroku.com/. Accessed: 2019-06-05.

[19] Página oficial de mongo atlas. https://www.mongodb.com/cloud/atlas. Accessed: 2019-06-05.

[20] Página oficial de mongodb. https://www.mongodb.com/es. Accessed: 2019-06-06.

[21] Página oficial de node.js. https://nodejs.org/es/. Accessed: 2019-06-06.

[22] Página oficial de purifycss. https://purifycss.online/. Accessed: 2019-06-05.

[23] Página oficial de sonar qube. https://www.sonarqube.org/. Accessed: 2019-06-05.

[24] Página oficial de travis. https://travis-ci.org/. Accessed: 2019-06-05.

[25] Página oficial de vue. https://vuejs.org/. Accessed: 2019-06-06.

[26] Página oficial de webpack. https://webpack.js.org/. Accessed: 2019-06-05.

[27] Qué es el stack MEAN y cómo escoger el mejor para ti. https://www.campusmvp.es/recursos/post/Que-es-el-stack-MEAN-y-como-escoger-el-mejor-para-ti.aspx. Accessed: 2019-04-10.

[28] Qué es Webpack, cómo funciona y forma de usarlo. https://www.arsys.es/blog/programacion/webpack-instalacion-funcionamiento/. Accessed: 2019-04-19.