



Escuela
Politécnica
Superior

Aplicación Web: Qualiteacher



Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:

Cristóbal Jesús González

Tutor/es:

Otto Colomina Pardo

Septiembre 2017



Universitat d'Alacant
Universidad de Alicante

Justificación personal

Qualiteacher surge en 2014 en pleno periodo de matriculación. Debía matricularme en segundo año de este grado, y habiendo escogido las asignaturas que quería cursar, me costaba escoger un turno, ya no por el horario, sino porque quería elegir correctamente a los profesores que me iban a enseñar. Embarcado en esta situación (al igual que en el resto de matriculaciones) he ido preguntando a compañeros que ya habían cursado las asignaturas para que me aconsejaran qué profesor escoger, aunque no siempre hice la elección correcta.

Ésta idea intenta suplir algo que, si no dispones de los contactos adecuados, es algo difícil de escoger a priori. Seleccionar de qué asignaturas matricularse, con qué profesor hacerlo... o incluso qué grado/universidad es la mejor en el campo en el que quieres aventurarte, y qué mejor fuente para aconsejarte, que los alumnos que han asistido a dichos cursos.

Además, establece un nuevo incentivo para el personal docente, aparecer más arriba en el ranking o mejorar su nota en ciertos campos es algo que, pese a parecer un tanto agresivo, puede ser bueno para el sistema docente actual.

Agradecimientos

Antes de entrar en materia me gustaría agradecer a mis padres todo lo que han hecho por mí, todo el empeño que han tenido tanto a nivel formativo como a nivel personal, regalándome unos valores que para mí son fundamentales.

A Yaiza González Valer por apoyarme cada momento de estos largos 4 años de mi vida, en tantos momentos de desesperación que ella ha hecho un poco más llevaderos, y por seguir haciéndolo una vez acabada esta etapa.

A mi tío Antonio González, por sacrificar su disfrute tecnológico para que yo tuviera un portátil con el que poder trabajar los largos días que he pasado en el campus.

A Germán y a Jose, por el apoyo, el cariño y los ratos en los que han conseguido rebajar mi nivel de estrés entre risas.

Al equipo de Viajesta por entender (casi siempre) mis ausencias en las que he dedicado tiempo a este trabajo.

Y en definitiva, a toda mi familia y amigos, por estar ahí siempre que les he necesitado.

Índice

Justificación personal	1
Agradecimientos	2
Índice	3
Índice de ilustraciones	6
Índice de tablas	7
1. Introducción	8
2. Estado del arte	9
2.1. patatabrava.com	9
2.2. misprofesores.com	9
2.3. Qualiteacher	9
3. Objetivos	10
4. Metodología	11
5. Arquitectura	12
5.1. Backend	12
5.2. Frontend	12
5.3. Base de datos	13
5.3.1. Versión inicial	13
5.3.2. Versión final	14
5.4. Schemas Mongoose	14
6. Herramientas	17
6.1. Trello	17
6.2. Git + GitHub	18
6.3. Heroku + mLab	18
6.4. QualiteacherDbGenerator + FakerJS	18
6.5. WebStorm	19
6.6. npm	19
6.7. Robo3T	20

7.	Tecnologías	21
7.1.	NodeJs	21
7.2.	MongoDB.....	21
7.3.	AngularJs.....	21
7.4.	Librerías	22
7.4.1.	Mongoosejs	22
7.4.2.	Jwt-simple	22
7.4.3.	Bcrypt.....	22
7.4.4.	Nodemailer	23
7.4.5.	Node-cron.....	23
7.4.6.	Chalk	23
7.4.7.	Faker.js	23
7.4.8.	Observe.....	23
8.	API	24
8.1.	Rutas generales.....	24
8.2.	Rutas del modelo usuarios.....	24
8.3.	Rutas del modelo profesores	25
8.4.	Rutas del modelo asignaturas.....	25
8.5.	Rutas del modelo carreras	26
8.6.	Rutas del modelo universidades	26
9.	Cliente	27
9.1.	Inicio	27
9.2.	Vista universidad.....	28
9.3.	Vista carrera	29
9.4.	Vista asignatura	29
9.5.	Vista profesor	31
9.6.	Encuesta de calidad de un profesor	32
9.7.	Registro	34
10.	Seguridad.....	35

10.1.	JWT	35
10.1.1.	Header	35
10.1.2.	Payload.....	35
10.1.3.	Signature.....	35
10.2.	Permisos	35
11.	Código e instalación.....	36
12.	Conclusiones	37
13.	Referencias.....	39

Índice de ilustraciones

Ilustración 1 - Tablero Kanban en Trello	17
Ilustración 2 - Tarea con checklist en Trello	17
Ilustración 3 - Página de GitHub del proyecto	18
Ilustración 4 - Interfaz WebStorm.....	19
Ilustración 5 - Interfaz Robo 3T	20
Ilustración 6 - Interfaz edición Robo 3T	21
Ilustración 7 - Ejemplo JWT	22
Ilustración 8 - Tabla de coste de rondas sal Bcrypt.....	23
Ilustración 9 - Esquema db inicial.....	13
Ilustración 10 - Esquema db final	14
Ilustración 11 - Página inicio Qualiteacher	27
Ilustración 12 - Ranking inicio Qualiteacher	27
Ilustración 13 - Buscador Qualiteacher	28
Ilustración 14 - Tooltip de resultados del buscador	28
Ilustración 15 - Vista universidad	28
Ilustración 16 - Vista carrera.....	29
Ilustración 17 - Seleccionada una asignatura en la vista de carrera	29
Ilustración 18 - Estadísticas de una asignatura.....	30
Ilustración 19 - Estadísticas de un profesor que imparte la asignatura	31
Ilustración 20 -Vista profesor.....	31
Ilustración 21 - Asignaturas que imparte el profesor	32
Ilustración 22 - Encuesta de calidad actual, en papel.....	33
Ilustración 23 - Encuesta de calidad en Qualiteacher.....	33
Ilustración 24 - Mensaje de error al enviar calificación.....	34
Ilustración 25 - Formulario de registro.....	34
Ilustración 26 - Header del token	35
Ilustración 27 - Payload del token.....	35

Índice de tablas

Tabla 1 - Descripción y entradas/salidas rutas generales	24
Tabla 2 - Descripción y entradas/salidas rutas modelo usuarios.....	25
Tabla 3 - Descripción y entradas/salidas rutas modelo profesores.....	25
Tabla 4 - Descripción y entradas/salidas rutas modelo asignaturas	25
Tabla 5 - Descripción y entradas/salidas rutas modelo carreras.....	26
Tabla 6 - Descripción y entradas/salidas rutas modelo universidades	26
Tabla 7 - Iconos de tipo de resultado del buscador	28

1. Introducción

Qualiteacher es una aplicación que permite realizar online las encuestas que ahora se realizan en papel sobre la labor del personal docente, y en base a estas encuestas calcula la nota de cada profesor en cada asignatura que imparte, así como una nota global (teniendo en cuenta todos los votos de sus asignaturas). Además, en base a las notas de los profesores que imparten una asignatura, se calculan las mismas métricas para las asignaturas, las carreras, y las universidades, permitiendo así a los usuarios de la aplicación conocer, por ejemplo, en qué universidad tiene una mayor valoración el grado/master que quieren estudiar.

Se ha desarrollado siguiendo el modelo MVC. Para el backend se ha utilizado Nodejs+Expressjs para realizar un API de donde obtener los datos desde el cliente, el cual está escrito en HTML5 + CSS3 + Javascript usando el framework Angularjs. Los datos se almacenan en una base de datos de MongoDB, que en la primera fase del desarrollo se encontraba en local, pero al desplegar la aplicación en heroku se migró a mLab (Database-as-a-Service).

Como sistema de control de versiones se ha usado Git y GitHub ha sido el hosting, haciendo uso de su paquete para estudiantes he podido ir desarrollando el sistema en un repositorio privado, que se hará público tras la defensa de este trabajo.

2. Estado del arte

Dado que los organismos de calidad de las universidades otorgan sus resultados exclusivamente a los docentes evaluados, el resto de personas implicadas en el ecosistema educativo no puede disponer de dichos resultados. Con el fin de tener unos resultados públicos y que ayuden al resto de la comunidad, se pretende desarrollar Qualiteacher.

En estos momentos es bastante complicado encontrar en la red un lugar donde poder obtener información sobre valoraciones de profesores en un aspecto más reglado, ya que en los foros de discusión sobre este aspecto la mayoría de comentarios son cuanto menos constructivos. Durante el análisis de los requisitos del proyecto, se han buscado en internet aplicaciones web con los requisitos que se quieren desarrollar. De entre esta búsqueda cabe destacar las siguientes páginas web.

2.1. patatabrava.com

En ella hay un ranking de profesores, pero la valoración se limita a un sistema de estrellas de 1 a 5, el cual se promedia para mostrar una nota de profesor (no hay encuesta sobre aspectos sobre su labor). En base a esas notas muestra rankings de universidades, facultades y carreras.

Además, dispone de un sistema de comentarios para que los usuarios puedan dejar valoraciones sobre los profesores, pero por lo general son anécdotas acaecidas en clase.

2.2. misprofesores.com

Misprofesores ofrece una encuesta más rica que patatabrava, pero se queda a medio camino ya que no se centra en aspectos de la calidad de la enseñanza como tal, sino que trata aspectos más subjetivos como la *facilidad*, *claridad*, y *ayuda*. Estos aspectos se pueden valorar del 1 al 5 y se calcula el promedio para cada uno de ellos, así como un *promedio general* que es la media aritmética de estos.

También dispone de comentarios y muestra los valores otorgados por el alumno en cada aspecto. Como aspecto negativo, la traducción no es muy buena y está llena de botones para compartir en redes sociales o invitar a nuevos usuarios.

2.3. Qualiteacher

Como se ha podido observar, ninguna ofrece una fuente fiable de información ni valora realmente la función del docente. Además, ninguna de ellas limita las encuestas a usuarios de la universidad, permitiendo que usuarios de otras universidades, o incluso personas o bots que no han tenido relación con el profesor puedan emitir una valoración sobre él, generando inseguridad y poca credibilidad en la información que ofrecen.

Estos aspectos pretenden ser cubiertos con Qualiteacher como se mostrará más adelante.

3. Objetivos

Debido a las carencias anteriormente descritas en el mundo de la educación, se pretende desarrollar una aplicación que cumpla los siguientes objetivos:

- Registro y autenticación de usuarios, almacenando los datos sensibles de forma segura.
- Realizar encuestas de profesores en base a asignaturas. Sólo se permitirá realizar dichas encuestas si se pertenece a la universidad del profesor.
- Ofrecer anonimato a los usuarios que realicen encuestas.
- Ver los resultados de las encuestas para todo el sistema académico, es decir, profesores, asignaturas, carreras y universidades.
- Desplegar la aplicación en un servidor de internet y utilizar un servicio de database-as-a-service para almacenar los datos.

Como objetivo secundario se plantea crear la aplicación utilizando la tecnología Full Stack Javascript MEAN (MongoDB, ExpressJs, AngularJs y Nodejs) y desplegarla en internet. Se ha escogido dicha tecnología por el auge que está teniendo en el mundo laboral y por las ganas de probar sus ventajas e inconvenientes.

4. Metodología

Debido a que el proyecto está desarrollado por una única persona, se ha seguido la metodología Kanban. Mediante un tablero en Trello se han ido estableciendo las tareas en tarjetas. También se ha limitado el flujo de la columna de desarrollo a 3 tarjetas simultáneas, de manera que se han tenido que terminar tareas ya empezadas antes de empezar tareas nuevas.

5. Arquitectura

5.1. Backend

Se ha seguido el patrón MVC, por lo tanto, tenemos tres capas:

- **Modelo:** Nos la proporciona Mongoose y es la capa de acceso a datos que nos permite interactuar con su persistencia.
- **Vista:** Son las pantallas que el usuario va a ver o los datos (en formato json) que devuelve el backend al realizar una petición. Para ciertas rutas el backend sirve un HTML con plantillas de AngularJs. Son las mismas que se usan en el frontend, pero debido a que no hemos usado el componente `$urlServiceProvider` de Angular, el backend manda las plantillas al cliente. Para otras rutas, el servidor simplemente responde con los datos necesarios en json como un API. Además, hemos usado el sistema de plantillas `ejs` en algunas de estas vistas para pasar los datos iniciales.
- **Controlador:** Es la lógica que se ejecuta tras una petición HTTP. Ya sea para recuperar y enviar datos o para llamar al modelo y actualizarlos.

5.2. Frontend

Aquí también se ha usado el patrón MVC:

- **Modelo:** Son los datos que se quieren mostrar en la vista. Se definen dentro del ámbito del controlador de angular.
- **Vista:** Son archivos HTML que contienen la interfaz, así como plantillas para poder mostrar los datos dinámicos necesarios.
- **Controlador:** Es la capa que gestiona la lógica de las acciones que el usuario realiza en la vista, así como la carga de datos vía Ajax.

5.3. Base de datos

5.3.1. Versión inicial

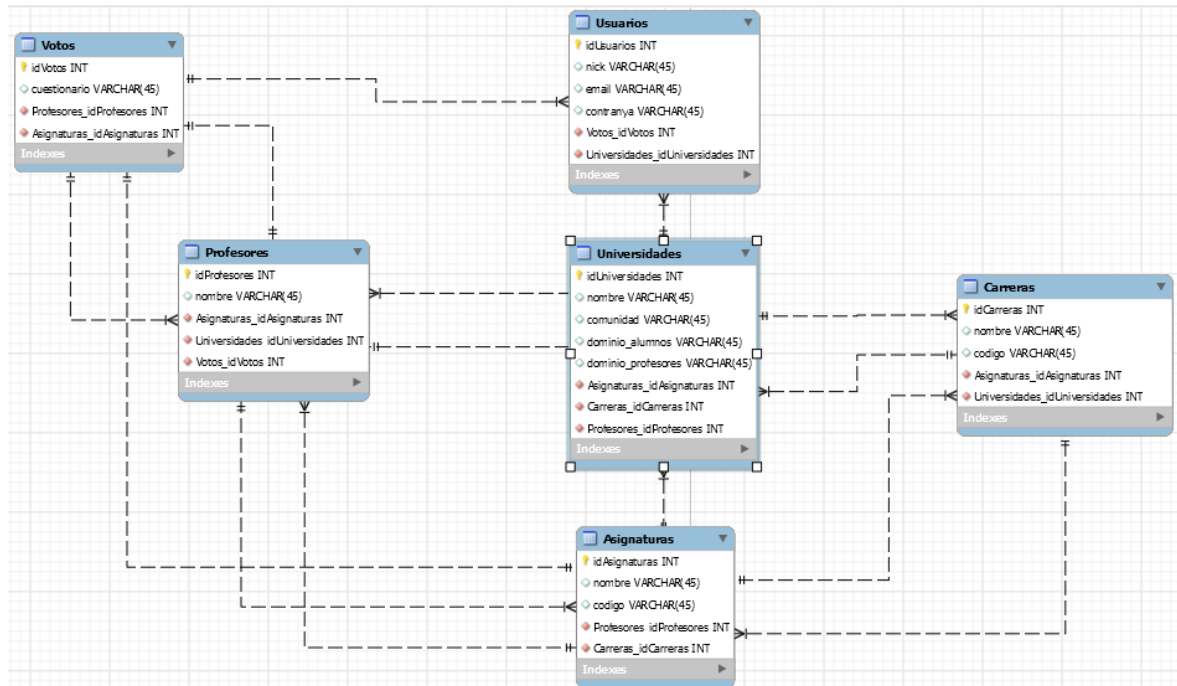


Ilustración 1 - Esquema db inicial

Sin embargo, ya avanzado el desarrollo y debido al sistema de cálculo de votos que se implementaba a en base a este modelo, tuvimos problemas de rendimiento con grandes volúmenes de datos. Tras hablarlo con el tutor se implementó un sistema de votos distinto y se cambió este modelo.

5.3.2. Versión final

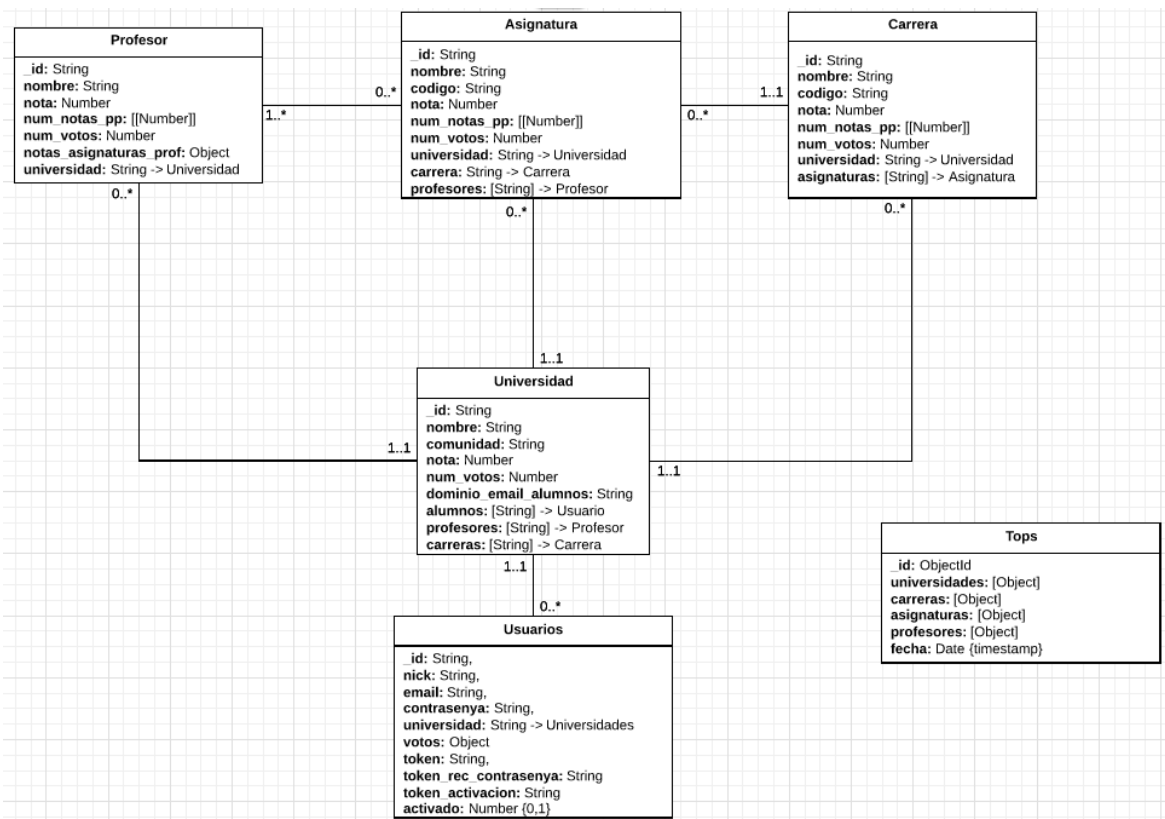


Ilustración 2 - Esquema db final

Gracias a la flexibilidad de mongo, la entidad de votos ha desaparecido. Ahora cada entidad tiene el número de votos y una estructura que almacena el resultado de las encuestas. Éste apartado se explicará en el siguiente punto.

5.4. Schemas Mongoose

Los esquemas de Mongoose son definiciones de clases que se mapearán a objetos en la base de datos de MongoDB. A continuación, se muestran los esquemas de cada entidad:

```

var ProfesoresSchema = new Schema({
  _id: String,
  nombre: String,
  nota: Number,
  num_notas_pp: [[Number]],
  num_votos: Number,
  notas_asignaturas_prof: [{
    asignatura: { type: String, ref: 'Asignaturas' },
    nota_asignatura: Number,
    num_notas_pp: [[Number]],
    num_votos: Number
  }],
  universidad: { type: String, ref: 'Universidades' }
});
  
```

```

var AsignaturasSchema = new Schema({
  _id: String,
  nombre: String,
  codigo: String,
  descripcion: String,
  nota: Number,
  num_notas_pp: [[Number]],
  num_votos: Number,
  universidad: { type: String, ref: 'Universidades' },
  carrera: { type: String, ref: 'Carreras' },
  profesores: [{ type: String, ref: 'Profesores' }]
});

```

```

var CarrerasSchema = new Schema({
  _id: String,
  nombre: String,
  codigo: String,
  nota: Number,
  num_notas_pp: [[Number]],
  num_votos: Number,
  asignaturas: [{ type: String, ref: 'Asignaturas' }],
  universidad: { type: String, ref: 'Universidades' }
});

```

```

var UniversidadesSchema = new Schema({
  _id: String,
  nombre: String,
  comunidad: String,
  nota: Number,
  num_votos: Number,
  dominio_email_alumnos: String,
  alumnos: [{ type: String, ref: 'Usuarios' }],
  profesores: [{ type: String, ref: 'Profesores' }],
  carreras: [{ type: String, ref: 'Carreras' }]
});

```

```

var UsuariosSchema = new Schema({
  _id: String,
  nick: String,
  email: String,
  contrasenya: String,
  token: String,
  universidad: { type: String, ref: 'Universidades' },
  votos: [{
    profesor: { type: String, ref: 'Profesores' },
    asignatura: { type: String, ref: 'Asignaturas' }
  }],
  token_activacion: String,
  token_rec_contrasenya: String,
});

```



```
    activado: Number
  });
```

```
var TopsSchema = new Schema({
  universidades : [{
    _id: { type: String, ref: 'Universidades'},
    nombre: String,
    nota: Number
  }],
  carreras : [{
    _id: { type: String, ref: 'Carreras'},
    nombre: String,
    nombre_universidad: String,
    nota: Number
  }],
  asignaturas : [{
    _id: { type: String, ref: 'Asignaturas'},
    nombre: String,
    nombre_universidad: String,
    nombre_carrera: String,
    nota: Number
  }],
  profesores : [{
    _id: { type: String, ref: 'Profesores'},
    nombre: String,
    nombre_universidad: String,
    nota: Number
  }],
  fecha: { type: Date, default: Date.now }
});
```

MongoDB almacena todo lo que queramos, de manera que en una misma colección podemos almacenar desde objetos con **distintos atributos** hasta archivos binarios como pdf (por hacer una mala analogía, en SQL lo que aquí llamamos colección sería una tabla). Valiéndose de esta cualidad, Mongoose añade automáticamente un atributo `_v` a cada objeto que crea. Este `_v` hace referencia a la versión del schema, y aumenta cada vez que almacenamos un objeto con diferentes atributos. De esta manera, podemos gestionar las distintas versiones de objetos.

Gracias a ésta funcionalidad y al mapeo que hace Mongoose de los Schemas, añadir/quitar atributos y entidades ha sido tan sencillo como modificar los modelos.

6. Herramientas

6.1. Trello

Trello es un gestor de tareas que imita las pizarras utilizadas en las metodologías LEAN para organizar el trabajo.

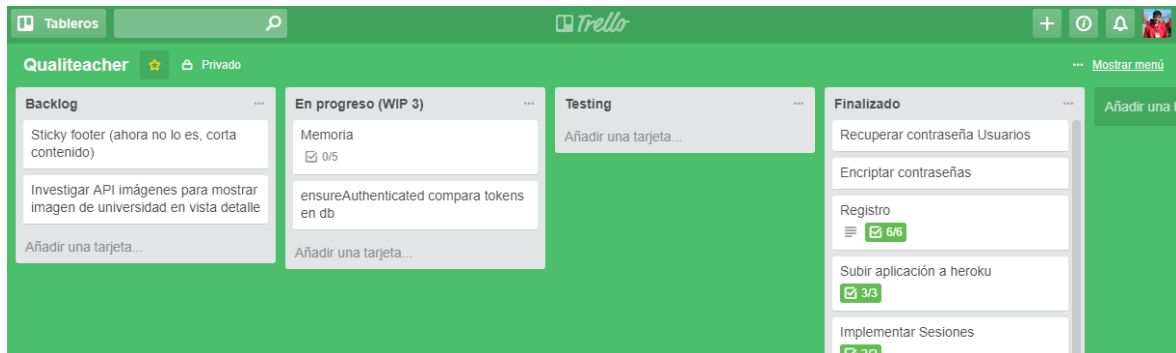


Ilustración 3 - Tablero Kanban en Trello

Cada tarjeta de Trello es una tarea por realizar. Éstas se pueden mover entre columnas para indicar el estado en el que se encuentran, de manera que echando un vistazo rápido a este tablero podemos ver las tareas que quedan por realizar, las ya terminadas, las que están en progreso o en fase de testing. Además, en ellas se pueden establecer descripciones, crear listas de objetivos, añadir comentarios y ver la actividad de cada tarjeta:

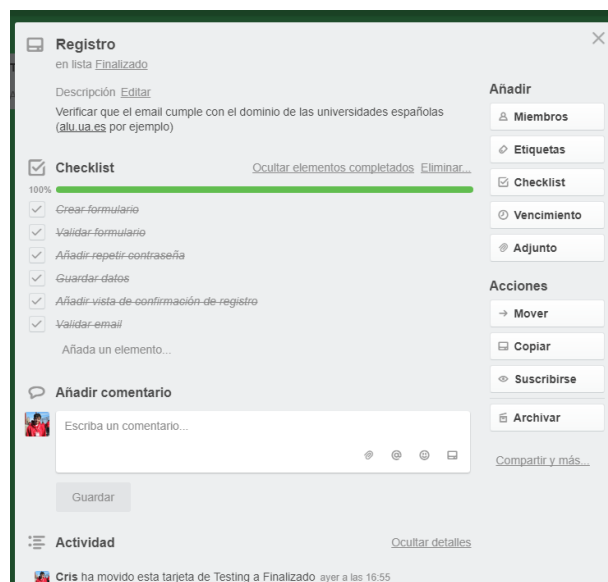


Ilustración 4 - Tarea con checklist en Trello

En las listas de objetivos podemos ir marcando los que vamos consiguiendo y el indicador que aparece debajo del título se va actualizando el progreso.

6.2. Git + GitHub

Git es el software de control de versiones más famoso, y en conjunto con el hosting de código GitHub, nos permiten realizar desarrollos utilizando un control de versiones distribuido, con todas las ventajas que ello ofrece. Además, GitHub provee una interfaz bonita, sencilla e intuitiva y que nos ayuda a gestionar el proyecto, ya sea añadiendo/restringiendo colaboradores, ver el histórico del proyecto o incluso ver las diferencias tras un commit.

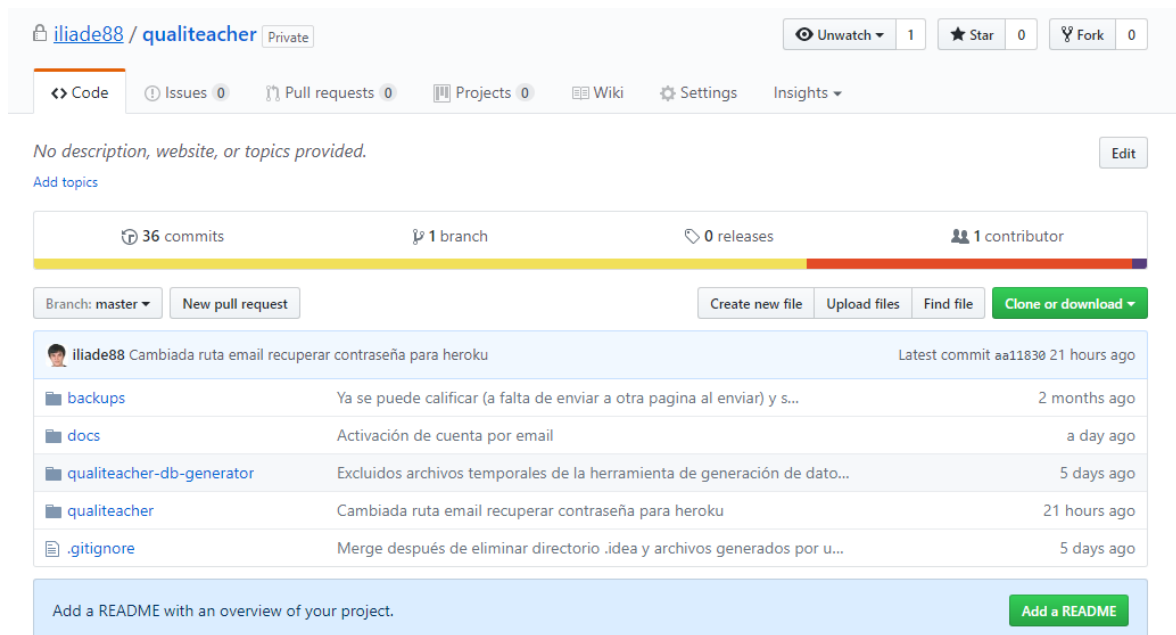


Ilustración 5 - Página de GitHub del proyecto

6.3 Heroku + mLab

Heroku es un platform-as-a-service de computación en la nube que nos permite desplegar aplicaciones de distintos lenguajes de programación en internet. Entre ellos se encuentra NodeJs, que es el que necesitábamos para desplegar la aplicación desarrollada. A pesar de que existen otras plataformas donde podríamos haber desplegado el proyecto como nodejitsu o digitalocean, se ha decidido utilizar heroku porque ya se hizo en la asignatura de “Aplicaciones Distribuidas en Internet” y ofrece un plan gratuito que es suficiente para este proyecto.

Heroku ha dejado de ofrecer soporte de bases de datos de MongoDB, pero recomienda usar mLab, un database-as-a-service el cual también tiene un plan gratuito. Mediante la herramienta mongodump y mongorestore se han migrado los datos del servidor local a la nube.

6.4 QualiteacherDbGenerator + FakerJS

Se pretendía utilizar MULE para consumir WebServices que poblaran la base de datos con información real de las carreras y profesores docentes actuales, pero tras horas de investigación se

ha determinado que no es posible obtener dichos datos. Quizá en un futuro gracias al avance del de los datos abiertos esto se pueda llegar a conseguir, pero en estos momentos es imposible.

Para solventar este problema, se ha desarrollado una utilidad paralela al proyecto, que teniendo las universidades existentes cargadas en la colección de MongoDB genera, para cada una de ellas, carreras (con nombres de un listado encontrado en internet), asignaturas y profesores. Además, también genera votos para que de ésta manera podamos ver realmente cuál sería el resultado tras un medio-largo plazo donde los usuarios hubieran tenido tiempo para realizar las encuestas de sus profesores.

6.5. WebStorm

Webstorm es un complejo IDE para el desarrollo de aplicaciones Javascript desarrollado por JetBrains. Tiene integración con Git y otros sistemas de control de versiones, detectando los cambios realizados desde el último commit y marcándolos. Además, tiene un analizador de código muy potente que ayuda al desarrollo, así como soporte para Angular y NodeJs, entendiendo el código y dándole estilo.

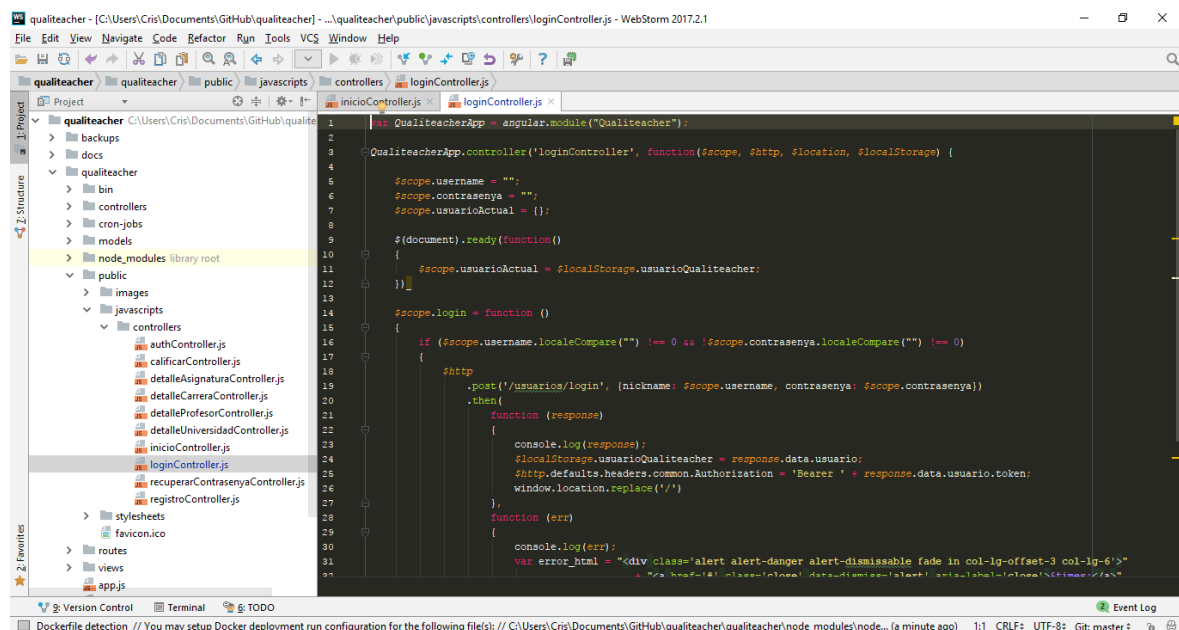


Ilustración 6 - Interfaz de WebStorm

6.6. npm

Npm es el gestor de paquetes para javascript más grande del mundo. Nos permite importar de forma muy sencilla librerías externas en nuestro proyecto, compartir nuevas librerías y gestiona las dependencias de forma que no es necesario, por ejemplo, subir las dependencias externas del

proyecto a GitHub, ya que cualquiera que clone el proyecto, gracias al archivo package.json puede instalar todas las dependencias necesarias con una sola instrucción.

6.7. Robo3T

También conocido como Robomongo, es una ligera y sencilla interfaz gráfica para gestionar bases de datos de MongoDB. Es gratuito y de código abierto, aunque en su web también ofrecen Studio 3T, que es un IDE para uso profesional de MongoDB (y es de pago).

Aquí podemos ver la interfaz de Robot 3T:

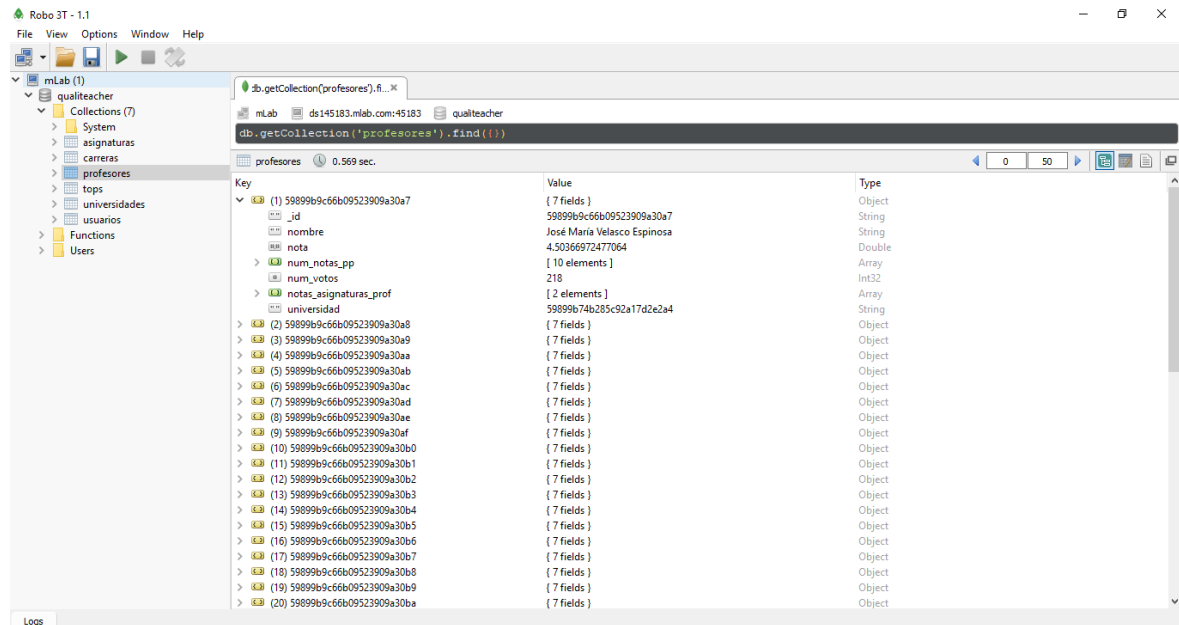


Ilustración 7 - Interfaz de Robo 3T

Y también nos permite editar los documentos y validar que los cambios efectuados son un json válido:

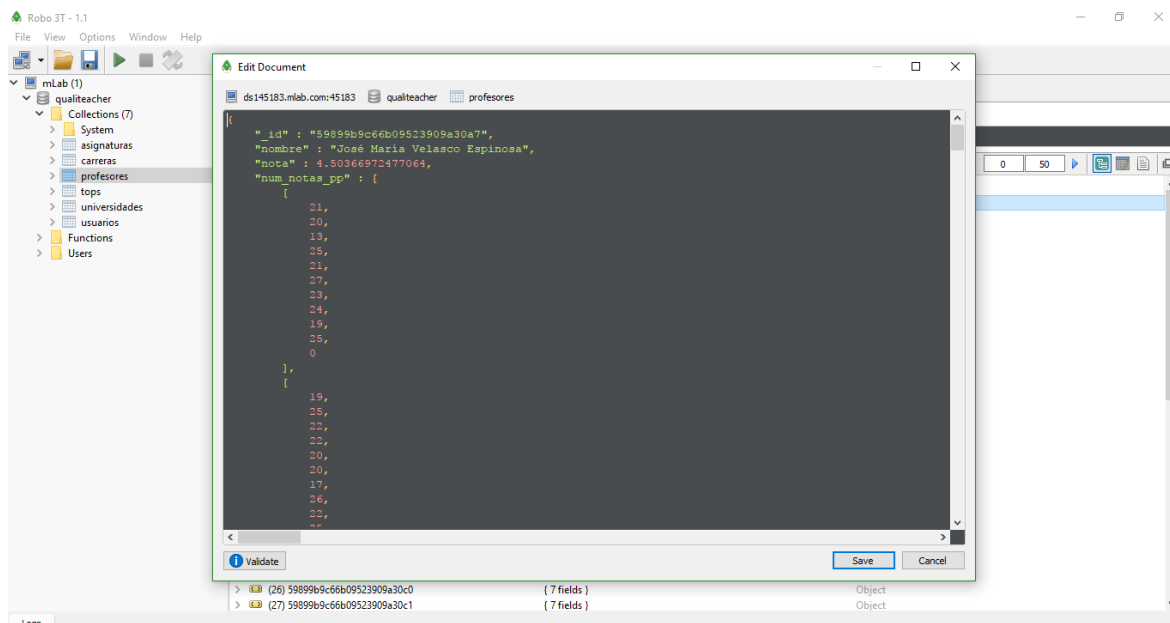


Ilustración 8 - Interfaz edición Robo 3T

7. Tecnologías

7.1. NodeJs

Tradicionalmente javascript se ejecutaba en el navegador, y por lo tanto, era un lenguaje de entorno cliente. Gracias a NodeJs, que funciona sobre el motor v8 desarrollado para Google Chrome, podemos ejecutar javascript en el lado del servidor y a muy rápido, ya que compila javascript en código máquina nativo en lugar de interpretarlo o ejecutarlo como bytecode.

Además, gracias a su bucle de eventos y a la programación asíncrona, un servidor corriendo sobre NodeJs es mucho más escalable que un Apache, por ejemplo, ya que no genera un nuevo hilo para cada cliente, simplemente dispara un evento.

7.2. MongoDB

Es un sistema de base de datos *No SQL* orientado a documentos. Esto significa que no almacena los datos en tablas relacionales, si no que guarda estructuras BSON con cualquier cosa (se pueden guardar documentos pdf, por ejemplo) y no existen las relaciones como tal, por lo que si necesitamos relaciones tendremos que simularlas nosotros mediante el diseño de los modelos y la programación.

7.3. AngularJs

Es un framework MV* para el entorno cliente. MV* significa que sigue el patrón Modelo-Vista-Cualquier cosa, de modo que sirve para implementar patrones MVC, MVP, MVVM, etc. Es muy famoso y gracias a sus controladores y su *data binding* de doble sentido hace que el desarrollo de las interfaces de aplicaciones web sea sencillo y rápido.

7.4. Librerías

7.4.1. Mongoosejs

Mongoose es un object data mapper para MongoDB. Gracias a él podemos usar instancias de objetos como modelos de MongoDB, y nos permite crear los esquemas que definen los modelos, hacer casting a dichos modelos y en definitiva cualquier cosa que se pueda hacer directamente con mongo, pero haciéndolo más sencillo, rápido y entendible.

7.4.2. Jwt-simple

Es una librería que implementa la tecnología de jwt mediante dos sencillos métodos, encode y decode, para convertir el payload a token y el proceso inverso.

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJfMTIzNDU2aWQiLCJuYW11IjoiaXVhbG10ZWFjaGVyIn0.0eDjq5i9-qNTz8H4mXwCC52ZM9czDeasx1KGytXwJs0
```

Decoded EDIT THE PAYLOAD AND SECRET (ONLY HS256 SUPPORTED)

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "_123456id",
  "name": "Qualiteacher"
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
) ☐ secret base64 encoded
```

Ilustración 9 - Ejemplo JWT

Además, firma y verifica que el token no haya sido manipulado.

7.4.3. Bcrypt

Para almacenar las contraseñas de los usuarios se ha utilizado esta librería que implementa el cifrado del mismo nombre. Mediante dos funciones sencillas, hash y compare, nos permite hashear la contraseña del usuario para que sea ilegible y comparar el texto plano con dicho hash. Además, es asíncrona (aunque también modo síncrono) y nos permite establecer el número de rondas que queramos. Hemos escogido hacer 8 rondas, que en realidad la librería internamente convierte a 2^8 , debido a al rendimiento y a que no es una aplicación que almacene datos especialmente sensibles. Aquí se puede ver un benchmark que ofrecen en la página de librería:

```
rounds=8 : ~40 hashes/sec  
rounds=9 : ~20 hashes/sec  
rounds=10: ~10 hashes/sec  
rounds=11: ~5 hashes/sec  
rounds=12: 2-3 hashes/sec  
rounds=13: ~1 sec/hash  
rounds=14: ~1.5 sec/hash  
rounds=15: ~3 sec/hash  
rounds=25: ~1 hour/hash  
rounds=31: 2-3 days/hash
```

Ilustración 10 - Tabla de coste de rondas sal Bcrypt

7.4.4. Nodemailer

Nodemailer nos permite de manera sencilla enviar correos desde nuestro backend. Se ha utilizado para la funcionalidad de registro y recuperación de contraseñas, mandando un enlace al usuario para que active o restablezca su contraseña según necesite.

7.4.5. Node-cron

Es un paquete que nos permite programar tareas para que se realicen periódicamente. Se ha utilizado para recalcular el ranking de la página principal una vez al día. Debido al gran coste de esta operación no es un proceso que se pueda realizar en cada petición ya que tardaría demasiado.

7.4.6. Chalk

Gracias a chalk podemos dar estilo a las salidas por consola de las aplicaciones. Debido a que la utilidad de generación de datos para la aplicación se ejecuta en terminal, se ha dado color a los mensajes que se muestran para que sea más amigable leer la salida.

7.4.7. Faker.js

Faker se ha usado también para la utilidad de generación de datos, generando nombres e imágenes de profesores y asignaturas.

7.4.8. Observe

Es una librería que implementa el patrón observer sobre objetos y arrays de javascript. Se ha usado en la utilidad de generación de datos para terminar el proceso una vez finalizado, ya que, debido al diseño de la utilidad, la escritura de las distintas colecciones de mongo se realiza en paralelo.

8. API

Aquí se van a describir las rutas expuestas en el servidor, sus parámetros, respuestas y funcionalidad.

8.1. Rutas generales

Tipo	Ruta	Entrada	Salida
GET	/	N/A	Muestra la vista inicial de la aplicación.
GET	/buscar/:cadena	:cadena → cadena a buscar	Envía los datos en formato json que se muestran en el buscador de la página principal.
GET	/login	N/A	Muestra el formulario para iniciar sesión.
GET	/registro	N/A	Muestra la vista de registro
GET	/registro-completado	N/A	Muestra la vista de registro completado

Tabla 1 - Descripción y entradas/salidas rutas generales

8.2. Rutas del modelo usuarios

Tipo	Ruta	Entrada	Salida
POST	/usuarios	{ Nickname, Email, Contrasenia, Universidad (El id de la universidad a la que pertenece) }	Status: <ul style="list-style-type: none">• 200: Si el registro ha sido correcto• 400: Si el Nick ya existe.
POST	/usuarios/login	{ NickName, Contrasenia }	Status: <ul style="list-style-type: none">• 200: Si el login ha sido correcto• 401: Si las credenciales proporcionadas no son válidas o el usuario no ha activado su cuenta. Objeto: Si el login es correcto, se envía al cliente un objeto con los datos del usuario y el token de sesión para que lo almacene en el local storage del navegador.
GET	/usuarios/activar/:token	:token: El token que se ha enviado por email tras el registro.	Muestra la vista activacionCompletada si todo ha ido bien, y la vista activacionError si ha habido algún problema activando la cuenta.
GET	/usuarios/recuperar	N/A	Muestra el formulario para recupera la contraseña
GET	/usuarios/recuperar/:token	:token: El token generado por el email tras solicitar la recuperación.	Si el token es válido muestra el formulario para establecer una nueva contraseña y si no es válido una vista de error.
GET	/usuarios/contrasenia-cambiada	N/A	Muestras la vista de finalización del proceso de recuperación de contraseña.

POST	/usuarios/nueva-contrasena	{ token: El token generado por el email tras solicitar la recuperación. Contrasena }	Status: <ul style="list-style-type: none"> • 200: Si se ha establecido correctamente la nueva contraseña. • 400: Si el token no es válido.
POST	/usuarios/email-recuperar-contrasena	{ Email }	Status: <ul style="list-style-type: none"> • 200: Si el email se ha enviado correctamente. • 400: Si no existe el email en la base de datos. <p>Manda un email al correo de entrada con un enlace de recuperación de contraseña.</p>

Tabla 2 - Descripción y entradas/salidas rutas modelo usuarios

8.3. Rutas del modelo profesores

Tipo	Ruta	Entrada	Salida
GET	/profesores/:id	:id: El id del profesor	Muestra la vista de detalle del profesor, con su nota y estadísticas.
GET	/profesores/id/:id	:id: El id del profesor	Obtiene los datos en json de un profesor.
GET	/profesores/:id/calificar	:id: El id del profesor	Muestra la encuesta del profesor.
POST	/profesores/:id/:asignatura/calificar	{ :id: El id del profesor :asignatura: El id de la asignatura para la que se ha rellenado la encuesta. Token: La cabecera de autenticación para JWT con el token de sesión del usuario }	Status: <ul style="list-style-type: none"> • 200: Si se ha registrado la encuesta correctamente. • 400: Si el id de profesor no existe o ya se ha votado al profesor para esa asignatura.

Tabla 3 - Descripción y entradas/salidas rutas modelo profesores

8.4. Rutas del modelo asignaturas

Tipo	Ruta	Entrada	Salida
GET	/asignaturas/:id	:id: El id de la asignatura	Muestra la vista de detalle de la asignatura.

Tabla 4 - Descripción y entradas/salidas rutas modelo asignaturas

8.5. Rutas del modelo carreras

Tipo	Ruta	Entrada	Salida
GET	/carreras/:id	:id: El id de la carrera	Muestra la vista de detalle de la carrera.
GET	/carreras/:id/datos	:id: El id de la carrera	Devuelve los datos de la carrera en formato json.

Tabla 5 - Descripción y entradas/salidas rutas modelo carreras

8.6. Rutas del modelo universidades

Tipo	Ruta	Entrada	Salida
GET	/universidades	N/A	Devuelve todas las universidades en formato json.
GET	/universidades/:id	:id: El id de la universidad	Muestra la vista de detalle de la universidad.

Tabla 6 - Descripción y entradas/salidas rutas modelo universidades

9. Cliente

Aquí se van a mostrar las vistas de las que dispone la aplicación. Se van a obviar las vistas de login y recuperación de contraseña ya que no tienen nada especial.

9.1. Inicio

La página de inicio es la siguiente:



Ilustración 11 - Página inicio Qualiteacher

En ella se puede ver el top 5 diario de cada modelo, así como clicar sobre cualquiera de ellos para ver en detalle cada modelo:

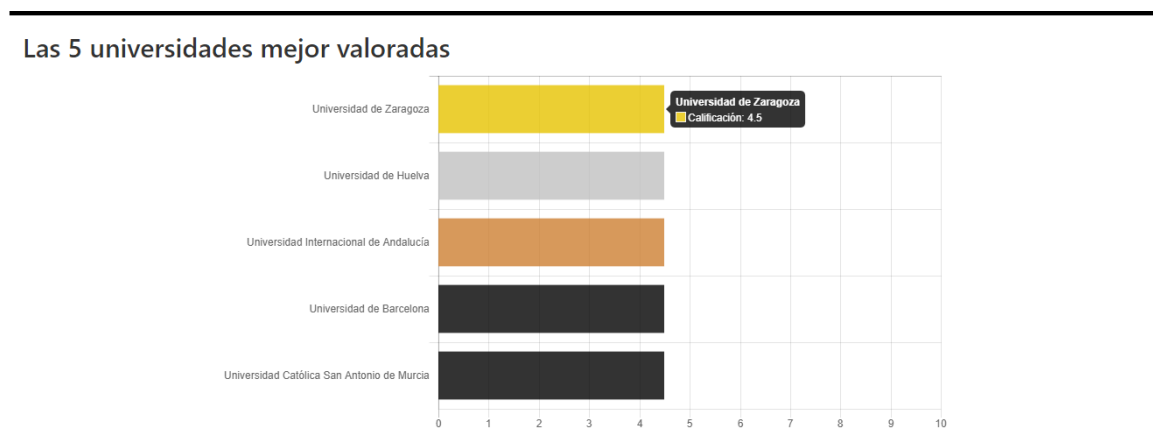


Ilustración 12 - Ranking inicio Qualiteacher

También se puede buscar y a medida que vamos tecleando aparecen resultados, con una imagen que describe el tipo de resultado que estamos viendo y con la posibilidad de hacer clic para ir a la vista de detalle del resultado:

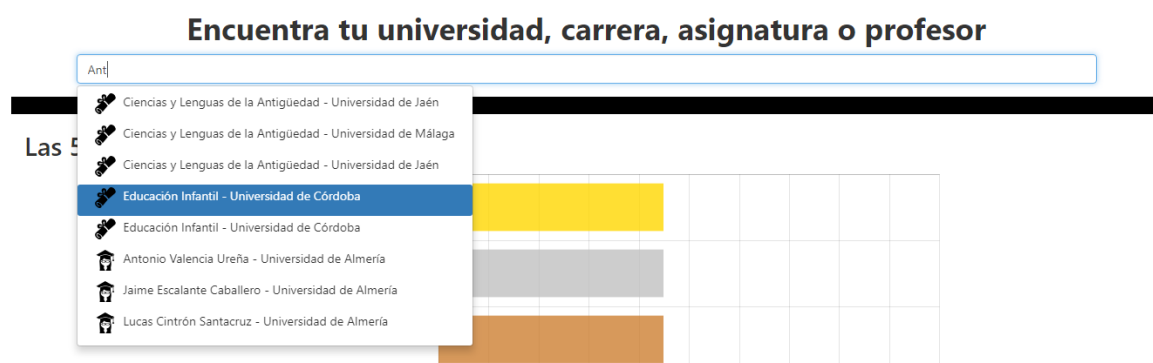


Ilustración 13 - Buscador Qualiteacher

Los iconos de tipo poseen un tooltip para describir a qué tipo pertenecen:

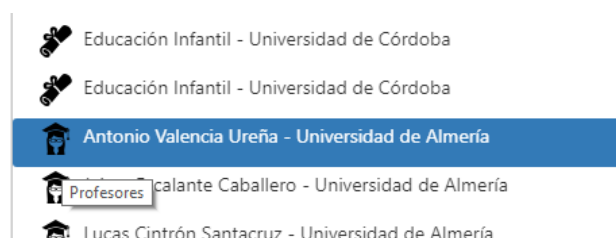


Ilustración 14 - Tooltip de resultados del buscador

Y éstos son todos los iconos:





Universidades	Carreras	Asignaturas	Profesores
			

Tabla 7 - Iconos de tipo de resultado del buscador

9.2. Vista universidad

En ella podemos ver los rankings de las carreras y profesores mejor valorados, así como su nota y número de votos.

Universidad de Zaragoza

Las carreras mejor valoradas de ésta universidad

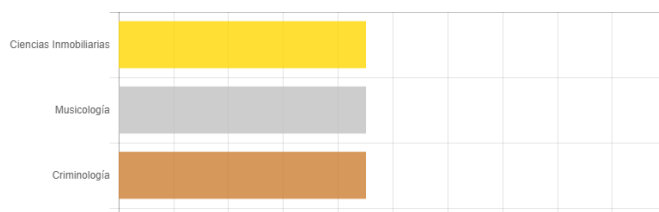


Ilustración 15 - Vista universidad

9.3. Vista carrera

Aquí podemos ver la nota de la carrera en cada aspecto de la encuesta, así como el número de veces que ha sido votada:

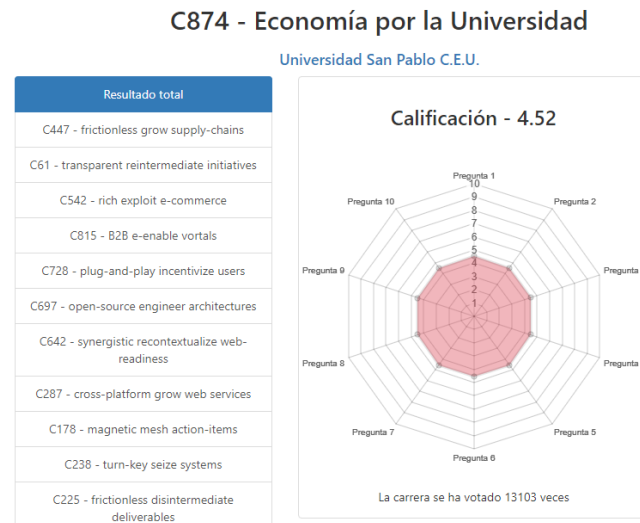


Ilustración 16 - Vista carrera

También podemos pinchar sobre el nombre de la universidad a la que pertenece para verla en detalle. Además, podemos seleccionar las asignaturas que pertenecen a dicha carrera para ver sus resultados, así como pinchar en el enlace debajo de la gráfica para ir a la vista de detalle de la asignatura:

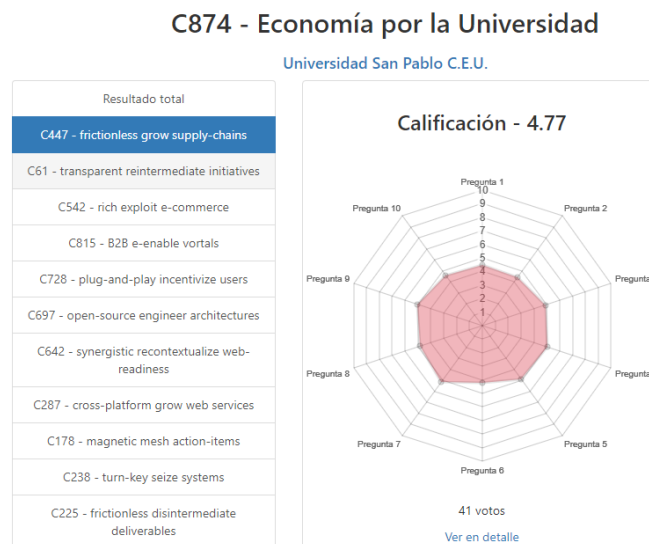


Ilustración 17 - Seleccionada una asignatura en la vista de carrera

9.4. Vista asignatura

Esta vista dispone de enlaces para ir a la universidad y carrera a las que pertenece la asignatura. Además, podemos ver los resultados de la asignatura:

C443 - world-class whiteboard markets

Química - Universidad de Almería

Resultado total
Alfredo Acevedo Urías
Samuel Nazario Espinosa
Vicente Aguilera Granado

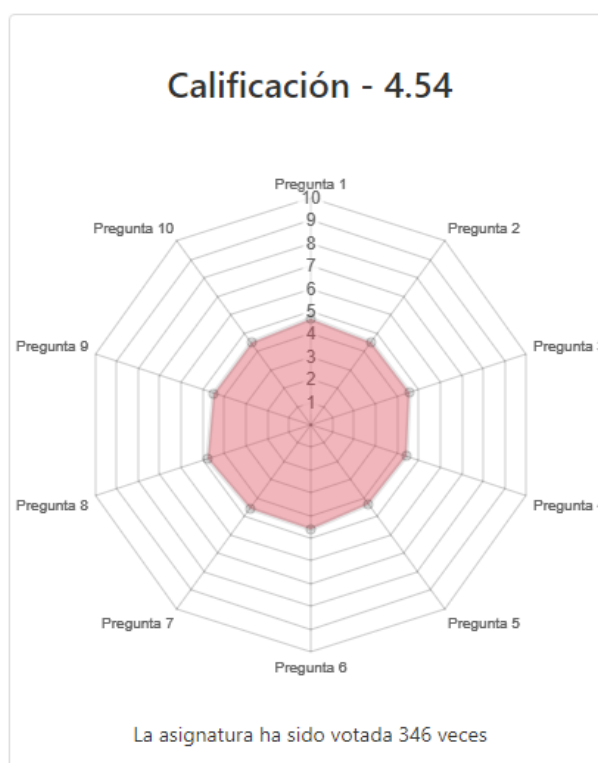


Ilustración 18 - Estadísticas de una asignatura

También aparecen los profesores que la imparten y las estadísticas obtenidas para dicha asignatura:

C443 - world-class whiteboard markets

Química - Universidad de Almería

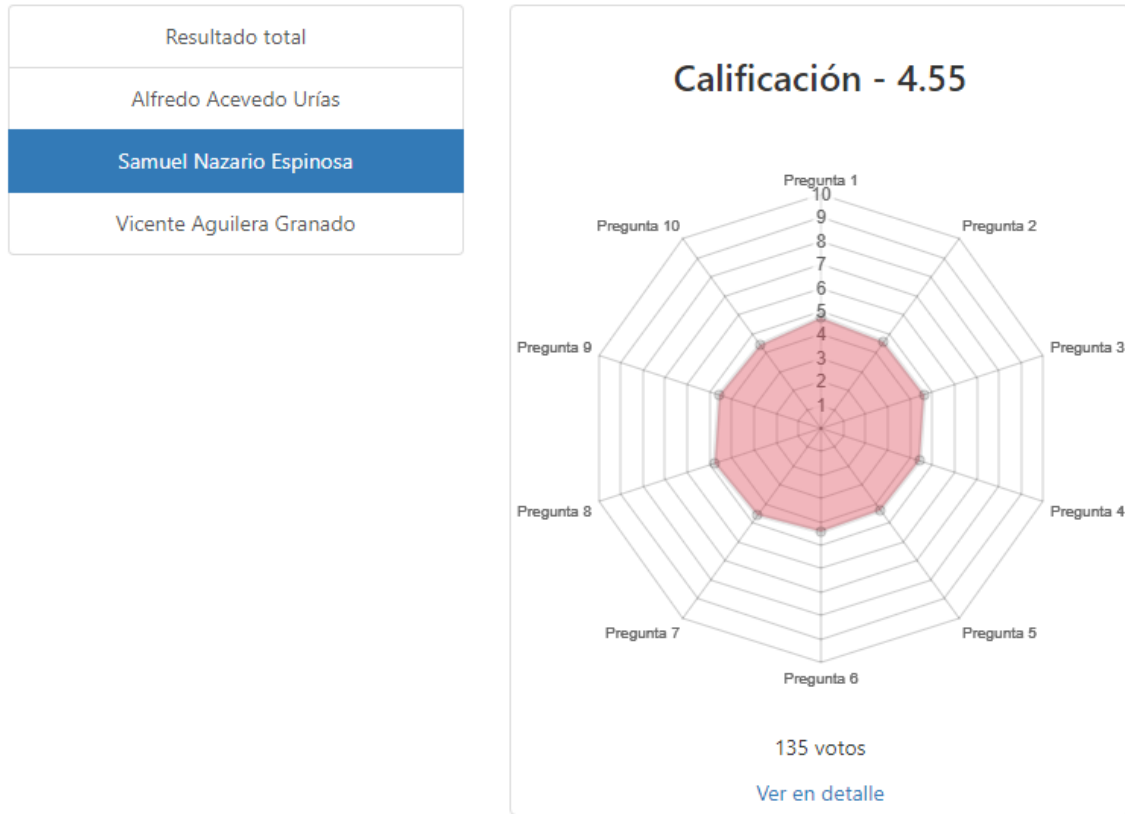
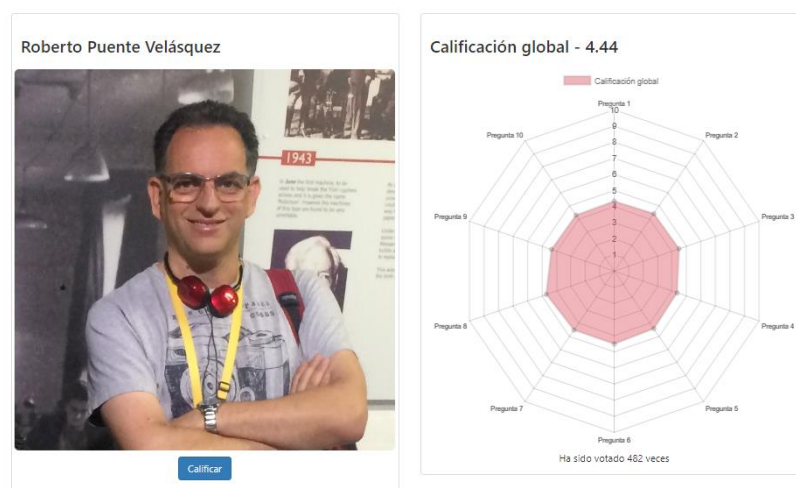


Ilustración 19 - Estadísticas de un profesor que imparte la asignatura

9.5. Vista profesor

En ella podemos ver los datos del profesor, así como sus estadísticas:



Asignaturas que imparte

Ilustración 20 -Vista profesor

También podemos realizar una encuesta sobre este profesor haciendo clic en el botón *calificar*. Además, podemos ver los resultados que ha obtenido en las asignaturas que imparte y clicar en el enlace para ver el detalle de la asignatura:

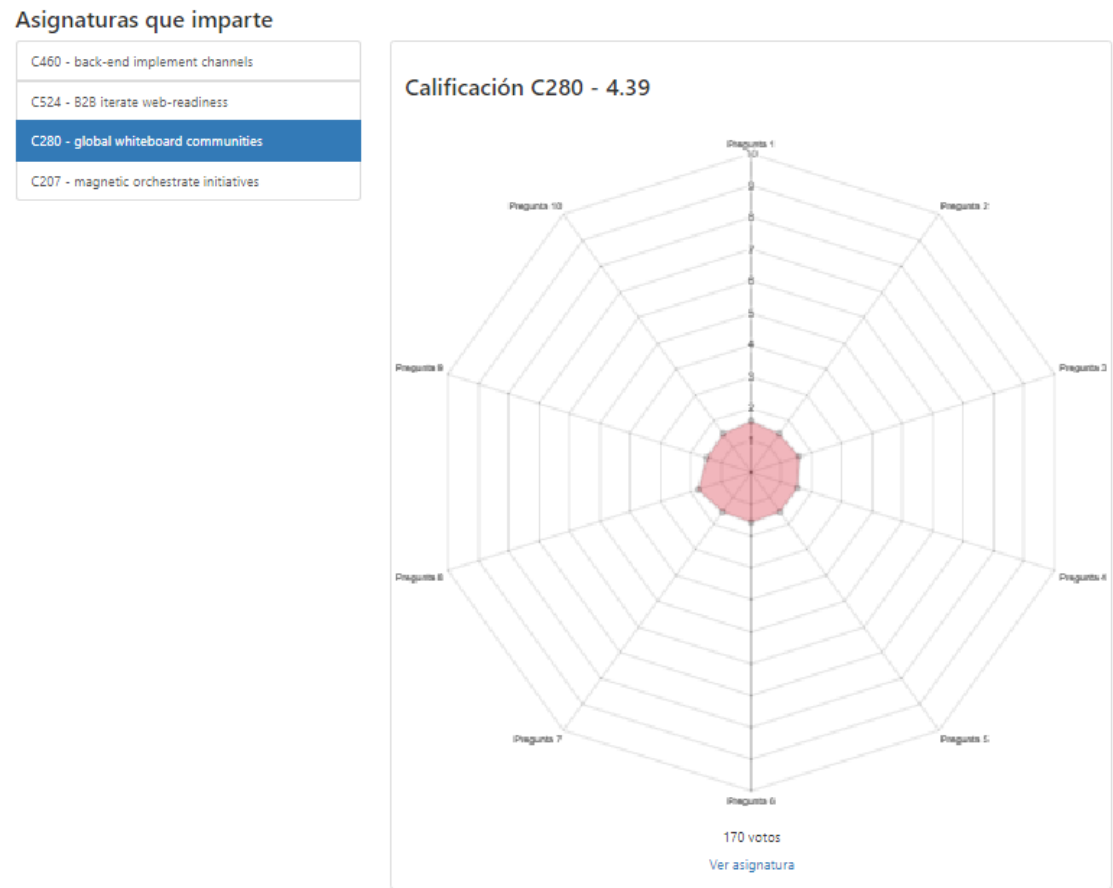


Ilustración 21 - Asignaturas que imparte el profesor

9.6. Encuesta de calidad de un profesor

Ésta es la vista de la encuesta de calidad. En ella, aparece un formulario igual que los que ahora se realizan en papel:

Valora de **0** (totalmente en desacuerdo) a **10** (totalmente de acuerdo) las siguientes afirmaciones:

	SIN OPINIÓN	0	1	2	3	4	5	6	7	8	9	10
1 - La información que me ha proporcionado el/la profesor/a sobre la actividad docente al comienzo del curso (objetivos, planificación, actividades y sistema de evaluación) ha sido adecuada.	<input type="radio"/>	0	1	2	3	4	5	6	7	8	9	10
2 - El/La profesor/a tiene la capacidad de enseñar.	<input type="radio"/>	0	1	2	3	4	5	6	7	8	9	10
3 - El/La profesor/a es accesible en sus tutorías, ya sea personal o virtualmente.	<input type="radio"/>	0	1	2	3	4	5	6	7	8	9	10
4 - El/La profesor/a me despierta el interés por la materia que imparte.	<input type="radio"/>	0	1	2	3	4	5	6	7	8	9	10
5 - El/La profesor/a muestra un conocimiento y formación adecuados de la materia.	<input type="radio"/>	0	1	2	3	4	5	6	7	8	9	10
6 - El/La profesor/a mantiene un buen clima de comunicación con los estudiantes.	<input type="radio"/>	0	1	2	3	4	5	6	7	8	9	10
7 - Los materiales y recursos docentes recomendados y utilizados por el/la profesor/a me han facilitado el aprendizaje.	<input type="radio"/>	0	1	2	3	4	5	6	7	8	9	10
8 - El desarrollo de la actividad docente del/de la profesor/a se adecua a los planes y objetivos establecidos	<input type="radio"/>	0	1	2	3	4	5	6	7	8	9	10
9 - El/La profesor/a ha facilitado mi aprendizaje, gracias a su ayuda he logrado mejorar mis conocimientos, habilidades o modo de afrontar determinados temas.	<input type="radio"/>	0	1	2	3	4	5	6	7	8	9	10
10 - En general, estoy satisfecho con la labor de este/a profesor/a.	<input type="radio"/>	0	1	2	3	4	5	6	7	8	9	10

Ilustración 22 - Encuesta de calidad actual, en papel

Podemos ver que son las mismas preguntas, con las mismas posibles respuestas. La única diferencia es que el usuario debe seleccionar la asignatura para la que quiere votar a este profesor.

Encuesta de calidad de Roberto Puente Velásquez

Asignatura a calificar:

C460 - back-end implement channels

	Sin opinión	0	1	2	3	4	5	6	7	8	9	10
La información que me ha proporcionado el/la profesor/a sobre la actividad docente al comienzo del curso (objetivos, planificación, actividades y sistema de evaluación) ha sido adecuada	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
El/la profesor/a tiene la capacidad de enseñar	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
El/la profesor/a es accesible en sus tutorías, ya sea personal o virtualmente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
El/la profesor/a me despierta el interés por la materia que imparte	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
El/la profesor/a muestra un conocimiento y formación adecuados de la materia	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
El/la profesor/a mantiene un buen clima de comunicación con los estudiantes	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Los materiales y recursos docentes recomendados y utilizados por el/la profesor/a me han facilitado el aprendizaje	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
El desarrollo de la actividad docente del/de la profesor/a se adecua a los planes y objetivos establecidos	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
El/La profesor/a ha facilitado mi aprendizaje, gracias a su ayuda he logrado mejorar mis conocimientos, habilidades, o modo de afrontar determinados temas	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
En general, estoy satisfecho con la labor de este/a profesor/a	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Ilustración 23 - Encuesta de calidad en Qualiteacher

Si el usuario no está registrado/logueado, o no pertenece a la universidad de éste profesor, no se le permite votar y se muestra un mensaje de error:

The screenshot shows a survey form titled "Asignatura a calificar:" with the value "C460 - back-end implement channels". A modal window from "qualiteacher.herokuapp.com" is displayed over the form, stating: "Debes acceder con tu cuenta para poder votar." with an "Aceptar" button. The survey form contains several statements with a rating scale from 0 to 10. At the bottom, there are two buttons: "Enviar calificación" (green) and "Cancelar" (red).

Statement	0	1	2	3	4	5	6	7	8	9	10
La información que me ha proporcionado el/la profesor/a sobre la actividad docente al comienzo del curso (objetivos, planificación, actividades y sistema de evaluación) ha sido adecuada											
El/la profesor/a tiene la capacidad de enseñar											
El/la profesor/a es accesible en sus tutorías, ya sea personal o virtualmente											
El/la profesor/a me despierta el interés por la materia que imparte											
El/la profesor/a muestra un conocimiento y formación adecuados de la materia											
El/la profesor/a mantiene un buen clima de comunicación con los estudiantes											
Los materiales y recursos docentes recomendados y utilizados por el/la profesor/a me han facilitado el aprendizaje											
El desarrollo de la actividad docente del/de la profesor/a se adecua a los planes y objetivos establecidos											
El/La profesor/a ha facilitado mi aprendizaje, gracias a su ayuda he logrado mejorar mis conocimientos, habilidades, o modo de afrontar determinados temas											
En general, estoy satisfecho con la labor de este/a profesor/a											

Ilustración 24 - Mensaje de error al enviar calificación

Además, cabe destacar que la encuesta es anónima, ya que sólo se almacena que el usuario ha realizado la encuesta para el profesor y la asignatura seleccionada (para sólo permitir 1 voto por usuario).

9.7. Registro

Éste es el formulario de registro:

The screenshot shows a registration form titled "Registro". It contains the following fields: "Nombre de usuario" (text input with placeholder "Introduce tu nombre de usuario..."), "Universidad" (dropdown menu with "Universidad de Almería" selected), "Email universitario" (text input with placeholder "Ej. chf10@alu.ua.es"), "Contraseña" (text input with placeholder "Mínimo 6 caracteres"), and a second text input for "Repite la contraseña...". At the bottom, there are two buttons: "Crear cuenta" (green) and "Cancelar" (red).

Ilustración 25 - Formulario de registro

Es el típico formulario de registro, la única peculiaridad es que se debe seleccionar la universidad a la que pertenece el usuario y se valida tanto que el dominio del email coincida con el de dicha universidad, como que el email sea real mediante un correo de activación de cuenta que se envía tras el registro.

10. Seguridad

En este punto se va a describir las medidas de seguridad que se han tomado en el desarrollo de la aplicación.

10.1. JWT

JSON Web Token (JWT) es un estándar abierto (RFC-7519) basado en json para crear un token que sirve para enviar datos entre aplicaciones o servicios y garantizar que sean válidos y seguros. Cuando el usuario se autentica se genera un token para él y se almacena en el localStorage del navegador para las siguientes peticiones al API. Dicho token consta de tres partes en forma de cadena:

10.1.1. Header

Contiene el algoritmo utilizado para codificar el token y el tipo:

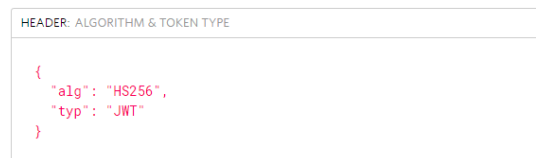


Ilustración 26 - Header del token

10.1.2. Payload

Aquí se definen los atributos que definen el token. Pese a la existencia de más atributos, nosotros sólo hemos utilizado los tres siguiente:

```
var payload = {
  sub: usuario._id,
  iat: moment().unix(),
  exp: moment().add(dias, "days").unix(),
};
return jwt.encode(payload, config.TOKEN_SECRET);
```

Ilustración 27 - Payload del token

- sub: Identifica al usuario que envía el token (es el id de usuario de MongoDB)
- iat: Contiene la fecha de expedición del token. Nos sirve para validarlo.
- exp: Contiene la fecha de expiración del token.

10.1.3. Signature

Es la firma del token y está formada por las dos partes anteriores, codificados en Base64. El resultado de la codificación se convierte en hash mediante un algoritmo a nuestra elección, en este caso HMAC-SHA256.

10.2. Permisos

Gracias a JWT, podemos crear una parte privada a la que sólo tengan acceso los usuarios autenticados. En nuestro caso, los permisos son los siguientes:

- Todos los usuarios pueden buscar cualquier tipo (universidades, carreras, asignaturas y profesores) o ver sus estadísticas.
- Como usuario autenticado se puede votar un profesor de la universidad a la que se pertenece.
- Sólo se puede votar a un profesor para una asignatura una vez.

11. Código e instalación

El código de la aplicación se encuentra alojado en GitHub, en la siguiente dirección:

- <https://github.com/iliade88/qualiteacher>

Además, se ha desplegado en heroku y se puede acceder desde aquí:

- <http://qualiteacher.herokuapp.com/>

Para instalar la aplicación en local, basta con instalar nodejs, MongoDB. Tras esto, se descarga el repositorio, se configura la ruta de conexión a mongodb y se escriben las siguientes instrucciones en un terminal:

```
npm install
```

```
node app.js
```

Con esto, ya se podrá ver la web en <http://localhost:3000>

12. Conclusiones

Una vez finalizado el desarrollo del proyecto, cabe destacar ciertos aspectos respecto a él.

En cuanto a la motivación de aprender la tecnología MEAN, ha sido una decisión agri dulce. Para empezar por las desventajas encontradas, el aspecto de la programación asíncrona y la disparidad en cuanto a cómo hacer las cosas supone un problema a la hora de aprender a usarla. Por ejemplo, para el buscador, he encontrado en internet más de tres tutoriales, además de la documentación oficial, que hacen lo mismo, con la misma librería (TwitterBootstrap Typeahead) pero cada uno de una forma distinta y usando conceptos o atributos distintos.

Por otra parte, el *callback-hell* es un problema casi imposible de evadir, haciendo que para cada tarea haya que escribir dos callback, la función de éxito y la de error, lo que es un tanto engorroso a la hora de organizar y leer el código. Además, cuando se requiere del resultado de una tarea asíncrona para realizar la siguiente tarea asíncrona, acaba resultando en un código bastante sucio desde mi punto de vista.

No obstante, una vez superada la barrera de entrada se pueden apreciar las ventajas que esta tecnología ofrece y el porqué del auge que está teniendo. Por ejemplo, el cambio en la estructura de datos antes mencionado, la facilidad que otorga para crear un endpoint de la API o la edición y muestra de datos dinámicos en las vistas gracias al data-binding de doble sentido. Además, tiene una gran comunidad detrás dispuesta a resolver las dudas que puedan aparecer y explicar conceptos por básicos que sean.

En definitiva, creo que para el desarrollo de aplicaciones de propósito general como es Qualiteacher, usar la tecnología MEAN es una opción excelente. Nos permite crear un API de manera muy sencilla y hacer lo que en internet llaman *dogfooding*, o hacer uso de nuestra propia API, para proveer al usuario de distintas interfaces, desacoplando la vista de la fuente de datos.

Otro problema que ha surgido es el método de obtención de datos. Sería genial poder tener un portal de datos abiertos de donde obtener toda la información de manera sencilla respecto a las carreras ofertadas, las asignaturas, los profesores que las imparten... pero sin embargo esto no es así, y creo que se debe trabajar en favor de ofrecer este servicio ya sea para los desarrolladores, o por el simple concepto de la transparencia en los organismos públicos.

En cuanto al desarrollo, los conocimientos obtenidos en las asignaturas de *Aplicaciones Distribuidas en Internet*, *Metodologías Ágiles de Diseño Software e Ingeniería Web* han sido de gran ayuda, ya que se han aplicado muchos conceptos de los aprendidos en ellas, como por ejemplo la autenticación basada en tokens con JWT o la herramienta Trello y la metodología Kanban para la gestión de tareas.

Por falta de tiempo, se han tenido ideas que no han podido ser implementadas, por ejemplo, un backoffice privado para que los organismos de calidad de las universidades introduzcan los datos

necesarios para que los usuarios puedan realizar las encuestas o el registro y autenticación mediante redes sociales (verificando que en ellas el usuario haya marcado la universidad a la que pertenece).

Para acabar, se ha desarrollado una aplicación web que gestiona y permite las estadísticas de manera masiva, así como realizar encuestas de calidad online, ahorrando papel y permitiendo a los alumnos obtener información sobre qué/donde estudiar.

13. Referencias

A continuación, aparece una lista de todos los recursos utilizados durante el desarrollo de este proyecto:

- Fuente para solucionar diversos problemas: <https://stackoverflow.com>
- W3Schools: <https://www.w3schools.com/>
- AngularJs: <https://angularjs.org/>
- Git: <https://git-scm.com/>
- Ejs: <http://ejs.co/>
- Robo3T: <https://robomongo.org/>
- Trello: <https://trello.com/>
- Observe: <https://www.npmjs.com/package/observe>
- Momentjs: <https://momentjs.com>
- ChartJs: <http://www.chartjs.org/docs/latest/>
- Creando herramientas en línea de comandos con node:
<https://developer.atlassian.com/blog/2015/11/scripting-with-node/>
- Chalk: <https://github.com/chalk/chalk>
- Fakerjs: <https://github.com/marak/faker.js>
- Node-cron: <https://www.npmjs.com/package/node-cron>
- Regexpr: <http://regexpr.com/>
- Bcrypt: <https://www.npmjs.com/package/bcrypt>
- Jwt-simple: <https://www.npmjs.com/package/jwt-simple>
- Referencias de ayuda sobre autenticación JWT:
 - <https://jwt.io/>
 - <http://jasonwatmore.com/post/2016/04/05/angularjs-jwt-authentication-example-tutorial>
 - <https://carlosazaustre.es/autenticacion-con-token-en-angularjs/>
 - <https://code.tutsplus.com/es/tutorials/token-based-authentication-with-angularjs-nodejs--cms-22543>
- Mongoose:
 - <https://www.npmjs.com/package/mongoose>
 - <http://mongoosejs.com/docs/>
- Cómo relacionar modelos MongoDB:
<https://carlosazaustre.es/como-relacionar-tus-modelos-en-mongodb/>
- MongoDB docs: <https://docs.mongodb.com/>
- Cómo buscar por más de un campo en mongodb:

<https://medium.com/@jeanjacquesbagui/in-mongoose-sort-by-date-node-js-4dfcba254110>

- Manejar objetos con Twitter Bootstrap Typeahead:
<http://tatiyants.com/how-to-use-json-objects-with-twitter-bootstrap-typeahead/>
- Flaticon: <https://www.flaticon.es>
- Cómo migrar datos a mLab: <http://docs.mlab.com/migrating/>
- Cómo desplegar en heroku con git: <https://devcenter.heroku.com/articles/git>
- Desplegar en heroku una aplicación que se encuentra en un subdirectorio del repositorio:
<https://coderwall.com/p/ssxp5q/heroku-deployment-without-the-app-being-at-the-repo-root-in-a-subfolder>
- Enviar emails con node: https://www.w3schools.com/nodejs/nodejs_email.asp
- Referencia sobre activación de usuarios vía email:
<http://www.forosdelweb.com/f18/aporte-activacion-cuenta-usuarios-via-e-mail-564126/>
- Generación clave secreta: <https://randomkeygen.com/>
- Mostrar tooltips cuando el ratón pasa sobre la imagen:
<https://stackoverflow.com/questions/2992018/how-can-i-display-tooltip-or-item-information-on-mouse-over>
- TwitterBootstrap Typeahead: <https://twitter.github.io/typeahead.js/examples/>