|  |  |
| --- | --- |
|  | **ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN**  **PROYECTO FIN DE GRADO**  **TÍTULO: ESTUDIO E IMPLEMENTACIÓN DE NUEVOS TIPOS DE EJERCICIOS EN LA PLATAFORMA MALL UP2B2**  **AUTOR: ROCIO MAGALÍ ROMERO PANERO**  **TITULACIÓN: GRADO EN INGENIERÍA TELEMÁTICA**  **TUTOR: ANTONIO DA SILVA FARIÑA**  **DEPARTAMENTO: DTE**    **Miembros del Tribunal Calificador:**  **PRESIDENTE: IRINA ARGÜELLES ALVAREZ**  **VOCAL:**  **SECRETARIO: ANA BELÉN GARCÍA HERNANDO**  **Fecha de lectura:**  **Calificación:**  **El Secretario,** |

Resumen

Este proyecto fin de carrera tiene como finalidad el estudio de las plataformas MALL para el aprendizaje de una lengua extranjera, así como la continuación del desarrollo del aplicativo Up2B2, añadiendo ejercicios con imágenes y audios. (1 a 2 páginas).

Abstract

The aim of this project is the design and implementation of ... (1 to 2 pages).

Índice de contenido

[Resumen i](#_Toc473644677)

[Abstract iii](#_Toc473644678)

[Índice de figuras vii](#_Toc473644679)

[Lista de acrónimos ix](#_Toc473644680)

[1 Introducción y objetivos 1](#_Toc473644681)

[1.1 Marco y motivación del proyecto 1](#_Toc473644682)

[1.2 Objetivos del proyecto 1](#_Toc473644683)

[1.3 Organización del resto de la memoria 1](#_Toc473644684)

[2 Marco tecnológico 3](#_Toc473644685)

[2.1 Introducción 3](#_Toc473644686)

[2.1.1 MALL 3](#_Toc473644687)

[2.1.2 Tecnologías empleadas 3](#_Toc473644688)

[2.2 Node.js 3](#_Toc473644689)

[2.2.1 Característica 1 3](#_Toc473644690)

[2.2.2 Característica 2 3](#_Toc473644691)

[2.3 Express.js 3](#_Toc473644692)

[2.4 MongoDB 5](#_Toc473644693)

[2.5 Ionic 5](#_Toc473644694)

[2.6 AngularJS 5](#_Toc473644695)

[2.7 Cordova 5](#_Toc473644696)

[3 Descripción del diseño y desarrollo realizados 9](#_Toc473644697)

[3.1 Modelos de datos en base de datos 9](#_Toc473644698)

[3.1.1 Modelo de usuario 9](#_Toc473644699)

[3.1.2 Modelo de pregunta 10](#_Toc473644700)

[3.2 Software utilizado para el desarrollo 12](#_Toc473644701)

[3.2.1 GITHUB y SourceTree 12](#_Toc473644702)

[3.2.2 Sublime Text 3 12](#_Toc473644703)

[3.2.3 Robomongo 12](#_Toc473644704)

[3.2.4 Postman 13](#_Toc473644705)

[3.3 Medidas de seguridad 13](#_Toc473644706)

[3.3.1 Archivo de configuración 14](#_Toc473644707)

[3.3.2 Registro en la aplicación 14](#_Toc473644708)

[3.3.3 Token de acceso 14](#_Toc473644709)

[3.3.4 Autenticación 15](#_Toc473644710)

[3.4 Funcionalidades de la aplicación 15](#_Toc473644711)

[3.4.1 Autenticación 15](#_Toc473644712)

[3.4.2 Administración 17](#_Toc473644713)

[3.4.3 Test 26](#_Toc473644714)

[3.4.4 Ranking 26](#_Toc473644715)

[3.4.5 Perfil 27](#_Toc473644716)

[3.4.6 Desconectar 28](#_Toc473644717)

[4 Resultados y discusión 31](#_Toc473644718)

[5 Conclusiones y trabajos futuros 33](#_Toc473644719)

[5.1 Conclusiones 33](#_Toc473644720)

[5.2 Trabajos futuros 33](#_Toc473644721)

[Bibliografía 35](#_Toc473644722)

[Anexo A. Manual de usuario 37](#_Toc473644723)

[A.1 Subcapítulo anexo 37](#_Toc473644724)

[A.1.1. Sub-sub capítulo anexo 37](#_Toc473644725)

[A.2 Otro subcapítulo anexo 37](#_Toc473644726)

Índice de figuras

[Figura 1. Esquema conceptual de Internet 1](#_Toc444017665)

Lista de acrónimos

# Introducción y objetivos

## Marco y motivación del proyecto

Bla bla bla ...



Figura 1. Esquema conceptual de Internet

## Objetivos del proyecto

Los objetivos de este proyecto fin de carrera son, desde el punto de vista técnico:

* Estudio de las metodologías en el aprendizaje de un idioma asistido a través de un dispositivo móvil (MALL).
* Continuación del desarrollo de la aplicación Up2B2 añadiendo ejercicios con imágenes y audio
* Aprendizaje de tecnologías basadas en Javascript como son AngularJS y NodeJS, además de bases de datos no SQL como es MongoDB.

## Organización del resto de la memoria

En los próximos capítulos de la memoria se ofrecen información sobre...

# Marco tecnológico

## Introducción

### MALL

Las siglas MALL significan “Mobile-assisted lenguaje learning”, es decir, aprendizaje de un lenguaje a través de una plataforma móvil. En la actualidad la mayoría de la población dispone de dispositivos móviles, y con este tipo de aplicaciones se permite el acceso a materiales, ejercicios y exámenes para el desarrollo del aprendizaje de un idioma.

### Tecnologías empleadas

Las tecnologías empleadas para el desarrollo de la aplicación las podemos separar en 2 tipos: las utilizadas para la parte del cliente, tales como Ionic, AngularJS y Cordova; y las utilizadas para la parte del servidor: NodeJS, ExpressJS y MongoDB.

## Node.js

Esta tecnología se utiliza en la parte del servidor. Nos ofrece un entorno de ejecución de Javascript orientado a eventos asíncronos.

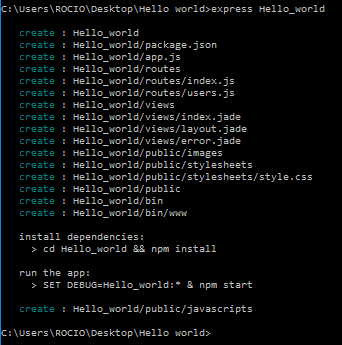
### Característica 1

### Característica 2

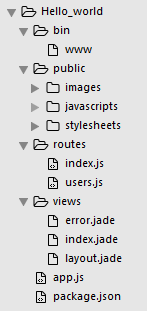
## Express.js

Además de Node.js, en la parte servidora hemos utilizado un framework llamado Express.js que nos ayuda a organizar la parte servidora con una estructura MVC, a la vez que nos facilita el manejo de rutas. Este framework permite utilizar muchas extensiones muy útiles y que nos permiten un desarrollo más rápido.

Comenzar a crear una API con Express es muy sencillo, basta con ejecutar el comando “express [nombre de la aplicación deseada]”. A continuación vemos un ejemplo para una aplicación llamada “Hello\_world”:



Como podemos observar, una vez ejecutado el comando se generan carpetas y archivos con todo lo necesario para comenzar a desarrollar nuestra aplicación, no teniendo que preocuparnos por la redirección ni la estructura de carpetas. Seguidamente instalaremos las dependencias con “npm install”.



En nuestra aplicación hemos utilizado los siguientes paquetes de NodeJS:

* Crypto: para hashear las contraseñas. En nuestro caso hemos utilizado sha256
* Jwt-simple: utilizado en el middleware para la gestión de los tokens.
* Mongoose: Mongoose se conecta a mongodb y nos permite especificar esquemas de datos, pudiendo añadirles métodos estáticos.
* Moment: Librería para parsear, validar, manipular y formatear fechas.
* Random-password-generator: Generador de cadenas de caracteres aleatorias.
* Nodemailer: Permite el envío de emails.
* fs: Node file system, usado para almacenar los archivos de imagen y audio en el servidor.
* Formidable: permite parsear información de formularios, usado para parsear la subida y descarga de los archivos de imagen y audio

## MongoDB

MongoDB es una base de datos NoSQL ágil en la que la estructura de los datos se basa en documentos.

El software utilizado para la gestión de la base de datos es Robomongo, pues nos ofrece una interfaz sencilla en la que poder visualizar fácilmente los documentos existentes en la base de datos y poder rápidamente modificarlos, eliminarlos o incluso añadir nuevos.

## Ionic

Es un framework para construir aplicaciones móviles. Sigue el patrón MVC (Modelo-Vista-Controlador) en el que se separa los datos, de la lógica y las interfaces de usuario. Permite el desarrollo de aplicaciones híbridas basadas en Javascript, HTML5 y CSS, optimizado con AngularJS y Sass.

Ionic nos ofrece un alto rendimiento, desarrollando una sola aplicación y pudiendo compilar tanto para Android como para IOS y web. Además disponemos de plantillas para no tener que comenzar el desarrollo desde cero, plugins de cordova y componentes desarrollados como directivas, tales como botones, formularios o tarjetas, todos con sus CSS asociados.

## AngularJS

Es un framework javascript y también sigue el patrón MVC. Permite “Data binding”, es decir, la actualización automática del contenido de la vista cuando el modelo cambia, o al revés, modificar el modelo cuando la vista cambia. Por otro lado, los controladores son los encargados de controlar el comportamiento de los elementos del DOM, como puede ser la acción a realizar al pulsar un botón. AngularJS también nos ofrece validación de formularios, comunicaciones asíncronas con el servidor, directivas o componentes reusables.

## Cordova

Es un framework para el desarrollo de aplicaciones móviles que permite usar HTML5, CSS3 y javascript para poder desarrollar aplicaciones en distintas plataformas. Permite utilizar una misma aplicación en distintas plataformas sin tener que implementarla para cada una.

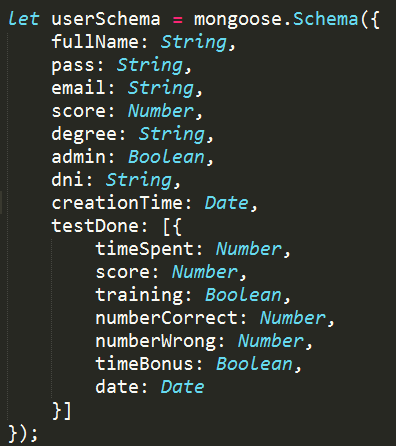
# Descripción del diseño y desarrollo realizados

## Modelos de datos en base de datos

Al haber utilizado el módulo de Mongoose en Express para el manejo de la base de datos, hemos podido establecer de antemano el modelo que seguirán las estructuras de datos de los documentos en MongoDB. Esto nos permite que en el caso de que se intente guardar contenido que no corresponde con el modelo, éste no se guarde.

### Modelo de usuario

El modelo del objeto de cada usuario se compone por: nombre completo, contraseña, email, puntuación actual, titulación, si es o no administrador, dni, fecha de creación del usuario, y una lista con información de cada test hecho, conteniendo el tiempo utilizado para hacer el test, la puntuación obtenida, si es de entrenamiento o no, el número de respuestas correctas, el número de respuestas incorrectas, si ha obtenido el bonus de tiempo, y la fecha en la que realizó el test. A continuación podemos ver cómo queda este esquema en código:

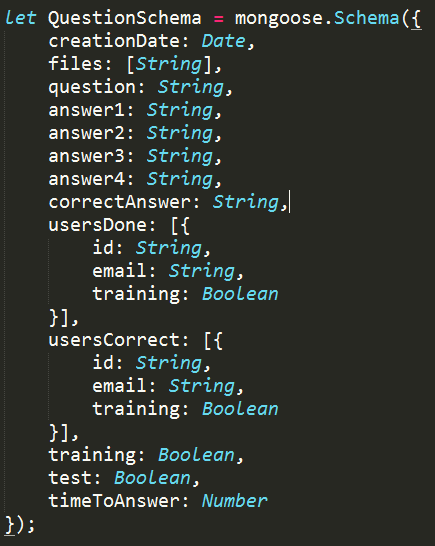


Si accedemos con Robomongo a la base de datos, para un usuario creado con anterioridad que ha realizado 1 test, su estructura sería la siguiente:

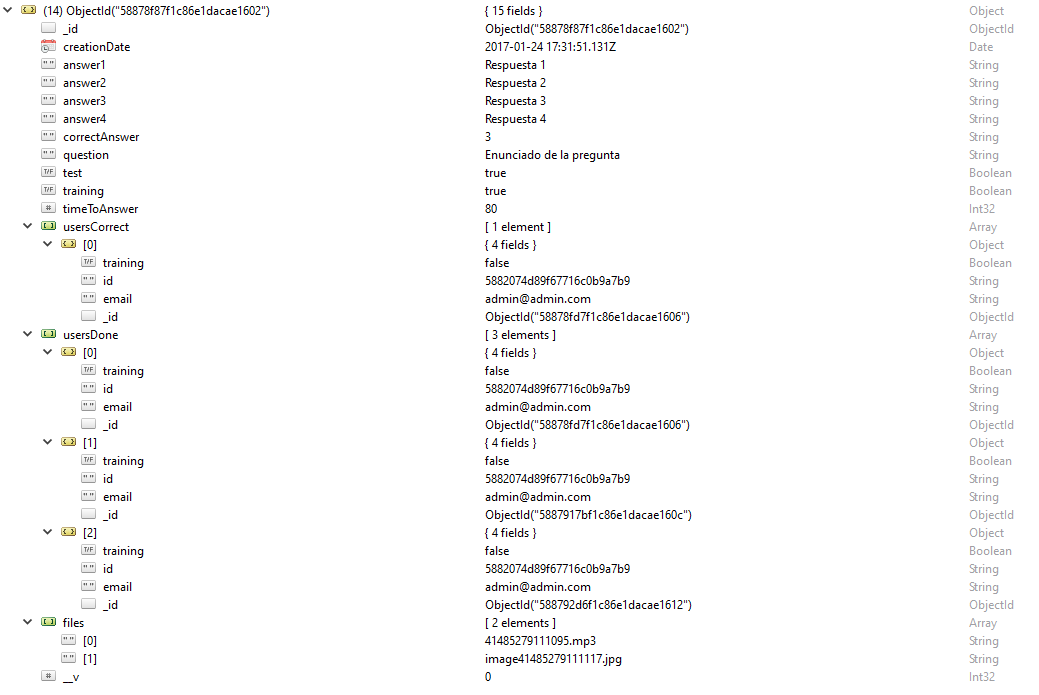


### Modelo de pregunta

El modelo de cada pregunta se compone por: fecha de creación, lista con la ruta a los archivos asociados, el enunciado de la pregunta, las 4 posibles respuestas, la respuesta correcta, si es de entrenamiento o no, si es de tipo test o no, el tiempo para contestar la pregunta, y 2 listas con los usuarios que han respondido a la pregunta y los que la han respondido correctamente. En código ésta estructura es la siguiente:



Por otro lado, en la base de datos accediendo desde Robomongo, podemos ver un ejemplo de una pregunta con 2 archivos asociados, 3 usuarios que han respondido a la pregunta, y 1 usuario que ha respondido a la pregunta correctamente:



## Software utilizado para el desarrollo

### GITHUB y SourceTree

Para el control se versiones se ha utilizado GitHub, a través del programa SourceTree que nos ofrece una interfaz gráfica. Con GitHub podemos crear ramas de trabajo para las distintas funcionalidades a desarrollar, y nos permite guardar cambios y documentar el desarrollo de la aplicación. Así, podemos ver las modificaciones, eliminaciones e inserciones de archivos en el proyecto, el porqué de los cambios, quién lo ha realizado y en qué momento.

### Sublime Text 3

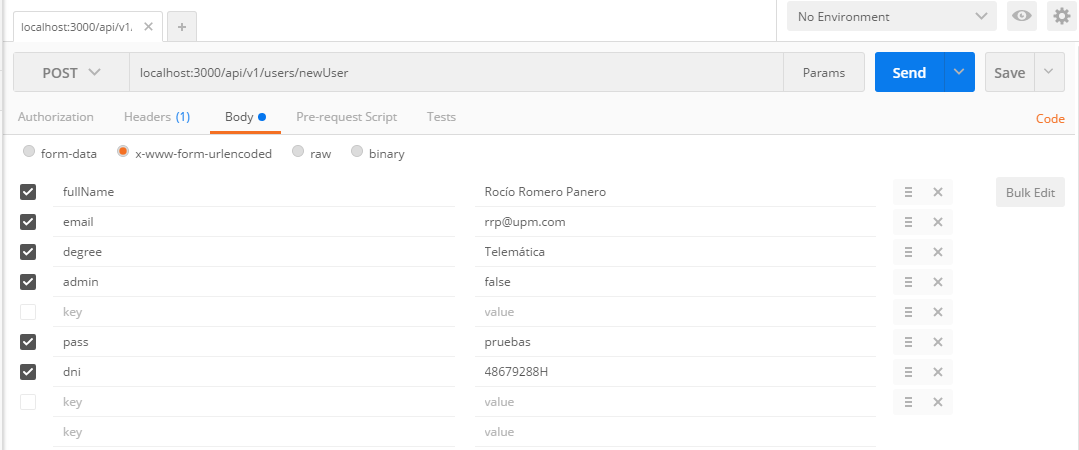
Para el desarrollo del código se ha utilizado el software Sublime Text 3, que incluye paquetes que se pueden instalar que facilitan el desarrollo y ahorran tiempo. Por ejemplo, incluye paquetes de AngularJS con snipets de creación rápida de controladores o servicios, o paquetes para CSS en que al escribir el color en hexadecimal, el color se pinta en la pantalla, o autocompletados de funciones y corrección de cierres de funciones con paréntesis.

### Robomongo

Software para la gestión de la base de datos de MongoDB. Incluye un aspecto visual intuitivo y fácil, facilitando la gestión de la base de datos y permitiendo visualizar, modificar, crear y eliminar los documentos de manera sencilla.

### Postman

Software útil al principio del desarrollo de la parte servidora. Permite realizar llamadas al servidor y probar las rutas para comprobar el correcto funcionamiento del servidor sin necesidad de tener operativo un cliente que llame a dichas rutas. A continuación se muestra una captura de pantalla del programa con un ejemplo de llamada para crear un nuevo usuario:



Como vemos, configuramos que la llamada sea de tipo “POST”, a la ruta api/v1/users/newUser, e introducimos en Body los campos que tendría la petición si la hiciéramos desde un cliente real. Además, podemos visualizar la respuesta recibida del servidor, que es la información que recibirá la parte cliente:



## Medidas de seguridad

Como medidas de seguridad en la aplicación, se ha optado por un lado del registro en la plataforma como control de acceso, además de la necesidad de autenticarse para poder acceder a la aplicación. Además, la contraseña en la base de datos se encuentra hasheada. Por otro lado, para el acceso a las rutas de la parte servidora es necesario tener un token que se genera una vez el usuario se ha autenticado en el sistema.

### Archivo de configuración

Para almacenar las variables de configuración del servidor, se ha creado un archivo excluido del sistema de control de versiones. De esta manera, nos aseguramos que el contenido del archivo, que contiene información sensible no sea accesibles para el exterior del servidor, además de permitir una configuración de la aplicación centralizada en un solo documento, sin necesidad de modificar el código. En este archivo tenemos las variables de configuración siguientes:

* Usuario y contraseña de la cuenta para la gestión del envío de emails para la plataforma.
* Palabra secreta con la cual se codificará y decodificará el token de autenticación
* Dirección de la base de datos
* Dirección del servidor
* Puerto del servidor
* Asunto de los emails de registro y cambio de contraseña.

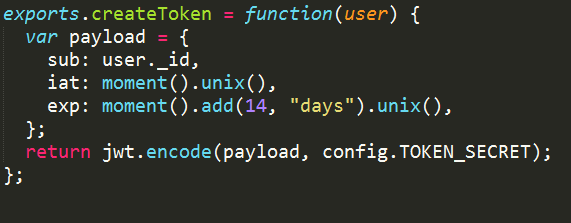
### Registro en la aplicación

La responsabilidad del registro en la aplicación de los usuarios será por parte de un usuario de tipo Admin. Por tanto, el apartado de administración donde se encuentra la opción de registro sólo se puede acceder si el usuario dispone de esta modalidad.

### Token de acceso

Una vez autenticado el usuario, se generará un token de acceso al servidor, y todas las peticiones realizadas al servidor desde el cliente tendrán que tener en la cabecera de la petición el token de acceso adjunto. De este modo, en el servidor antes de procesar cada petición, se recogerá dicho token y se comprobará si es válido.

Al crear un tóken de acceso, primero se establece su contenido, y después se codifica con una palabra secreta previamente elegida al configurar el servidor. En nuestro caso, el token se conforma por un payload con el id del usuario, el momento de la creación del token, y el momento de expiración del token.



Así, para comprobar la validez del token, se siguen los pasos siguientes:

1. Se decodifica el token con la clave secreta, y en caso de que se produzca una excepción al decodificar, significará que el token no ha sido codificado con la misma clave secreta del servidor y por tanto no es válido.
2. En caso de que se decodifique correctamente, se comprobará si el token ha expirado su periodo de validez, que en nuestro caso lo hemos configurado como 14 días.

En caso de que el token se haya codificado con la misma clave secreta y que no haya expirado, se permitirá el acceso a la ruta del servidor.

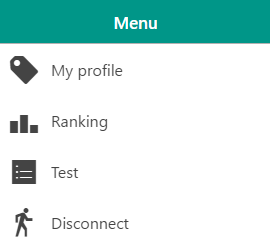
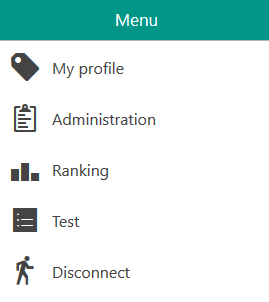
### Autenticación

Para poder pasar de la pantalla de Login de la aplicación, es necesario autenticarse con email y contraseña válidos presentes en la base de datos. Una vez el usuario cumplimenta el formulario de autenticación, se realiza una petición al servidor de tipo POST a la ruta /api/v1/users/login, en el que se siguen los siguientes pasos:

1. Se comprueba si el usuario con dicho email existe en la base de datos, y en caso de que no exista el servidor responderá con un código de estatus 404 de usuario no encontrado
2. En caso de que exista dicho usuario, se hasheará la contraseña con sha256 y se comprobará si coincide con la contraseña almacenada en la base de datos. Si las contraseñas no coinciden, el servidor responderá con un código de estatus 401.
3. Si la contraseña coincide, se generará un token de acceso y se responderá con un código de estatus 200 de éxito y el contenido del id, email, nombre completo, dni, titulación, si es o no admin, y la puntuación del usuario autenticado.

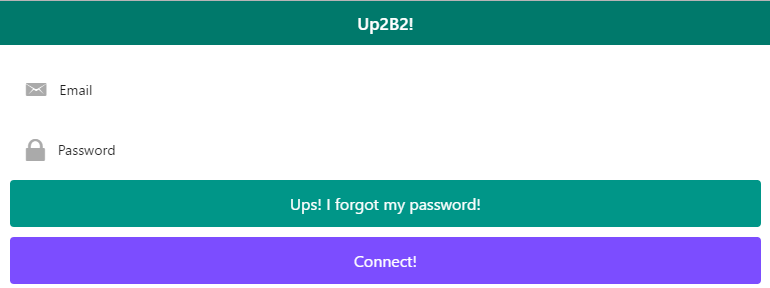
## Funcionalidades de la aplicación

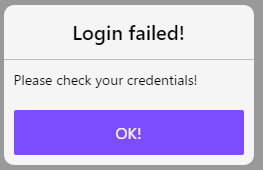
La aplicación dispone por un lado de una sección de administración al que sólo puede acceder el usuario que es de tipo administrador, y por otro lado, todos los usuarios pueden acceder a los distintos apartados que se encuentran en el Menú. Según el tipo de usuario, éste tendrá acceso desde el menú al apartado de Administración o no.



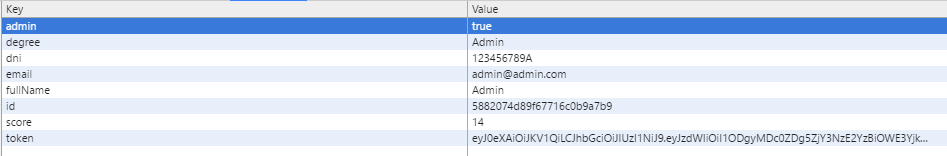
### Autenticación

Al entrar a la aplicación, la primera pantalla que se visualiza es la de autenticación. En esta pantalla tendremos que completar el email y la contraseña del usuario. En caso de que surja un error en el proceso de autenticación como puede ser que el usuario con dicho email no exista, o que las contraseñas no coincidan, se abrirá un popup de información.





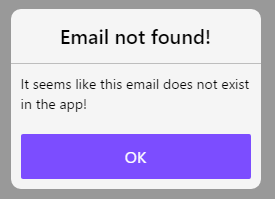
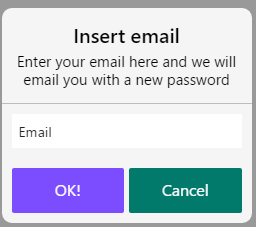
En caso de autenticación existosa, se guardará la información del usuario en el local storage, y si el usuario es de tipo administrador se redirigirá a la pantalla de administración, y si no a la pantalla de perfil.

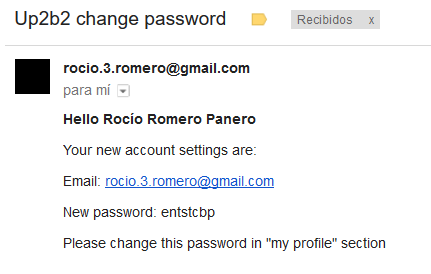
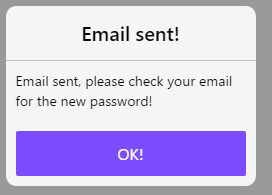


#### Recordar contraseña

En la pantalla de autenticación tenemos la opción de recibir un email en caso de que no nos acordemos de la contraseña. Al pulsar el botón “Ups! I forgot my password!” se abrirá un popup en el que rellenaremos el email el cual no nos acordamos de la contraseña. Una vez pulsado en el botón de “OK!” se enviará una petición al servidor de tipo POST a la ruta /api/v1/password, donde se seguirán los siguientes pasos:

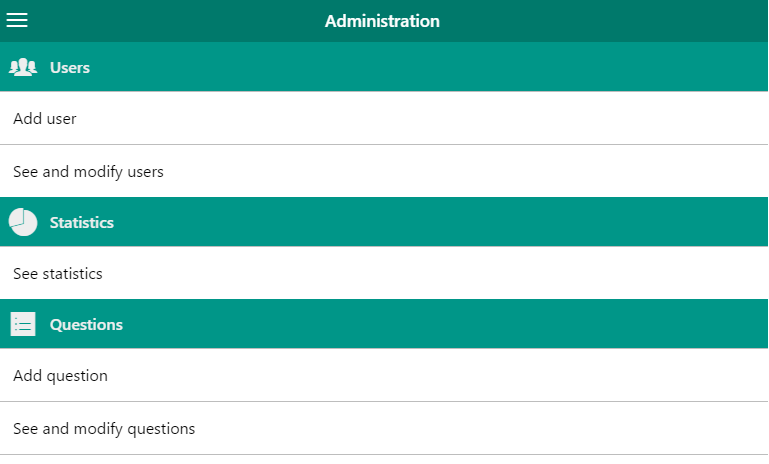
1. Buscar en la base de datos el usuario con el email recibido. En caso de que no exista, se responderá con un código de estado 499 al cliente.
2. Si el usuario existe en la base de datos, se generará una contraseña nueva aleatoria y se enviará un email al correo especificado, con la nueva contraseña, y se responderá con un código de estado 200 al cliente.





### Administración

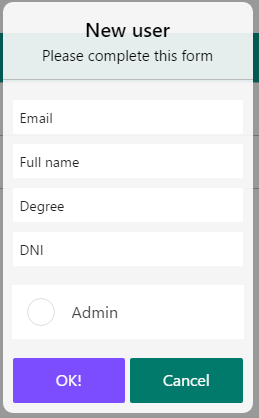
En la pantalla de administración disponemos de distintos apartados tanto para administración de usuarios, como de preguntas y de visualización de estadísticas.

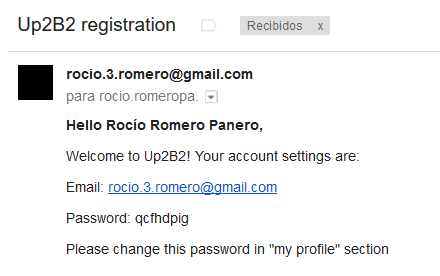


#### Registro de usuario

Una de las opciones para la gestión de los usuarios es la de registrar nuevos usuarios en la aplicación. Al pulsar en la opción, se abrirá una ventana popup en la que tendremos que rellenar los datos del nuevo usuario: email, nombre completo, titulación, dni y si será de tipo admin o no. Al pulsar en el botón “OK!” primero en la parte cliente se comprobará si todos los campos están completados, y en caso contrario se abrirá otro popup pidiendo que se rellenen todos los datos. Si todos los datos con correctos, se enviará una petición de tipo POST a la ruta /api/v1/users/newUser y se seguirán los siguientes pasos:

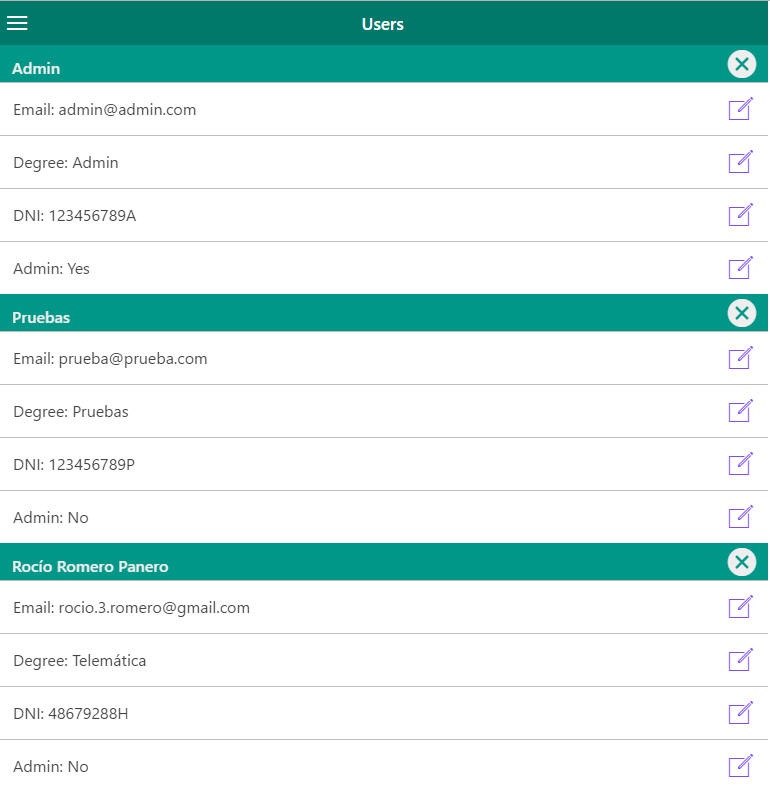
1. Se comprobará que el email del usuario no existe actualmente en la aplicación. En caso de que ya exista un usuario registrado con este email, se responderá con un código de estado 499.
2. En caso de que no exista un usuario con este email, se generará una contraseña aleatoria y se hasheará con sha256 para guardarla en la base de datos. Seguidamente se salvará el nuevo usuario en la base de datos y se enviará un email al usuario nuevo registrado y al usuario de administración con sus credenciales de acceso a la aplicación. Una vez se han enviado los emails, se responde al cliente con un código de estado 200.



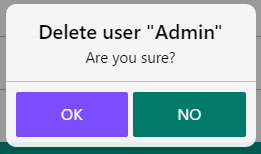


#### Gestión de usuarios

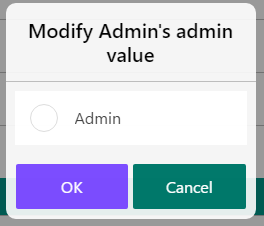
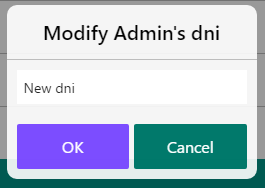
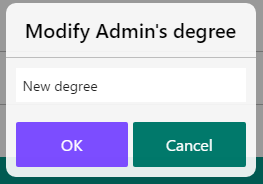
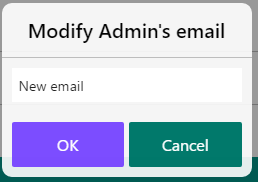
En este apartado podremos ver los usuarios registrados en la aplicación, modificar sus datos o eliminarlos. Al entrar en la pantalla primero se realizará una llamada al servidor de tipo GET a la ruta /api/v1/users/ para obtener todos los usuarios registrados en la aplicación y así poder mostrarlos en la vista.



Al pulsar en el botón de arriba a la derecha de cada usuario, nos aparecerá un popup en que tendremos que confirmar la eliminación del usuario. En caso de que pulsemos en “OK”, se realizará una petición al servidor de tipo DELETE a la ruta /api/v1/users/:id en la que se eliminará al usuario de la base de datos y se responderá al cliente con el código de estado 200.



Por otro lado, si pulsamos en los iconos de la derecha de email, degree, dni o admin, nos aparecerá un popup donde podremos poner el nuevo valor a modificar. Una vez introduzcamos el nuevo valor y pulsemos en el botón “OK”, se hará una petición al servidor de tipo PUT a la ruta /api/v1/users/userData/:id en la que se modificará el valor del usuario en la base de datos y en caso de que se modifique correctamente, se responderá con un código de estado 200.



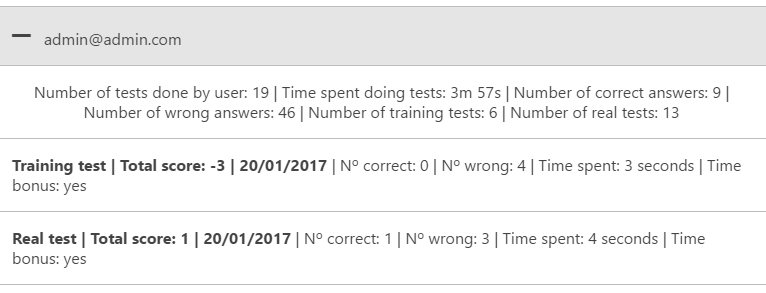
#### Estadísticas

En el apartado de estadísticas podremos obtener estadísticas de los usuarios y de las preguntas de la aplicación.

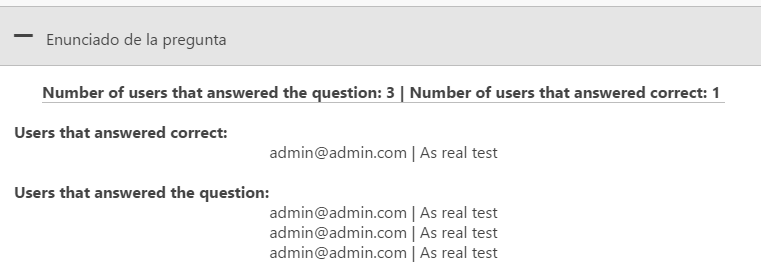


En el caso de estadísticas por usuarios, tendremos una lista con los usuarios de la aplicación, que al desplegarlo tendremos información del usuario: número de tests realizados, el tiempo total haciendo tests, número de respuestas correctas totales, número de respuestas incorrectas totales, el número de tests de tipo entrenamiento y el número de tests de tipo real. Además, tendremos una lista de los tests realizados por el usuario, especificando el tipo de test, la puntuación total obtenida, la fecha de realización del test, número de respuestas correctas, número de respuestas incorrectas, el tiempo para realizar la pregunta y si ha recibido el bonus de tiempo.

Para obtener esta información, una vez pulsado en “By User” se hace una llamada al servidor de tipo GET a la ruta /api/v1/users para obtener los usuarios de la aplicación y así poder mostrarlos en la vista.

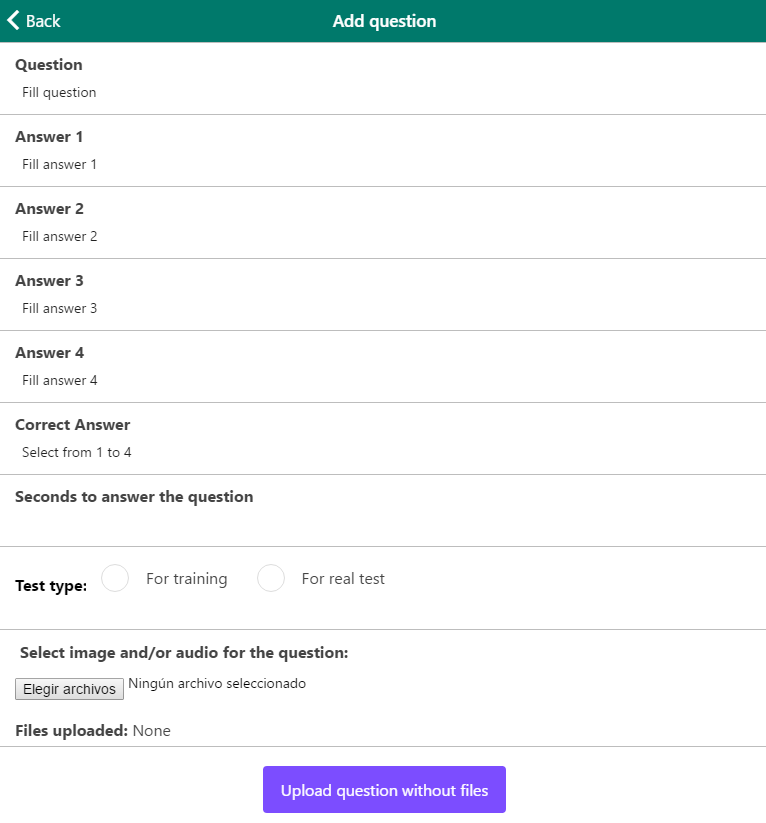


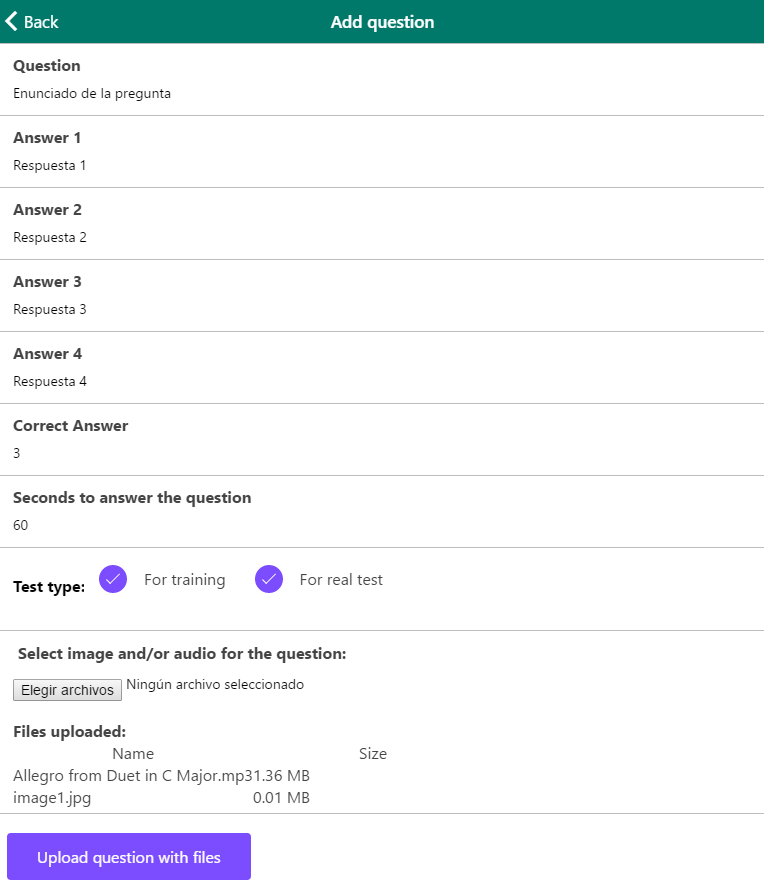
Por otro lado, en el caso de estadísticas por preguntas, tendremos una lista con las preguntas añadidas a la aplicación, con información del número de usuarios que han respondido la pregunta, el número de usuarios que la han respondido correctamente, y el email de los usuarios para ambos casos. Para obtener esta información, una vez pulsado en “By question” se hace una llamada al servidor de tipo GET a la ruta /api/v1/questions.



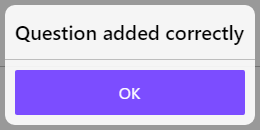
#### Creación de preguntas

Al crear una pregunta tendremos que completar los campos de enunciado, respuestas 1, respuesta 2, respuesta 3, respuesta 4, respuesta correcta, segundos para contestar a la pregunta, el tipo de test (entrenamiento y/o test real) y los archivos de imagen y/o audio asociados a la pregunta.





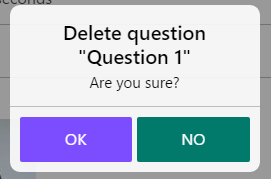
Una vez pulsamos en “Upload question with files”, primero los archivos se hace una petición de tipo POST a la ruta /api/v1/upload en la que se parseará la información del formulario recibido con los archivos, se les añadirá al nombre del archivo la fecha de creación para evitar duplicados, y se guardarán en la ruta de carpetas public/files. Una vez recibida la respuesta del servidor al cliente, se hace una petición de tipo POST a la ruta /api/v1/questions/newQuestion en la que se guarda en el campo files de la pregunta el nombre de los archivos asociados y seguidamente se guarda la pregunta en la base de datos.

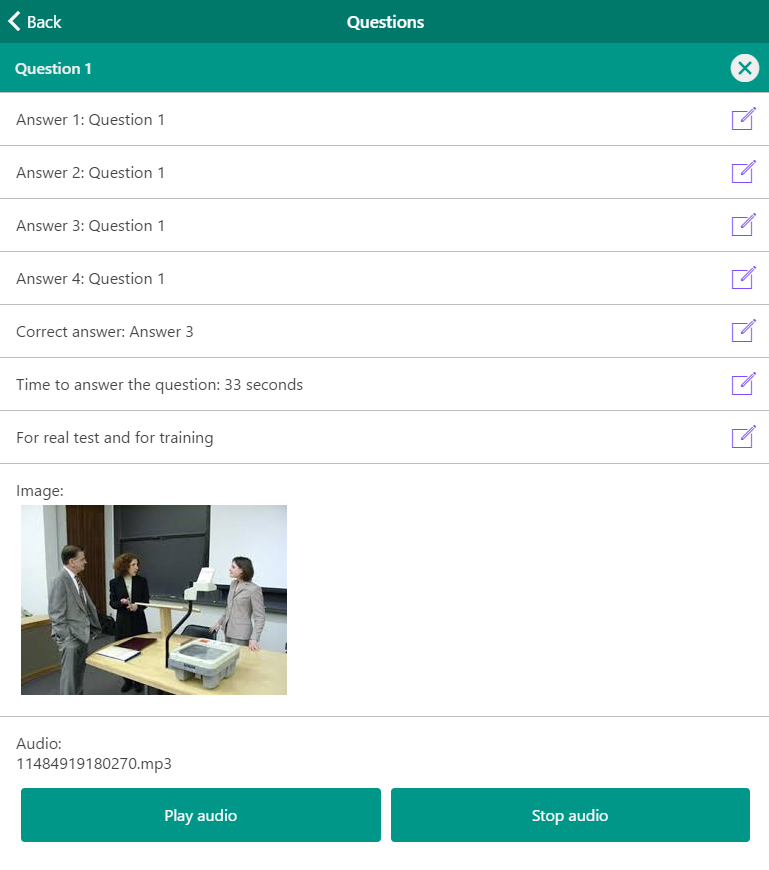


#### Gestión de preguntas

En esta pantalla tendremos la información de todas las preguntas añadidas a la aplicación, pudiendo modificar sus campos o eliminarlas. Al entrar a la pantalla se hace una petición de tipo GET a la ruta /api/v1/questions que devuelve todas las preguntas creadas en el sistema.

Al pulsar en el botón de arriba a la derecha de cada pregunta, podremos eliminarla confirmando en el popup. Entonces, se hará una petición de tipo DELETE a la ruta /api/v1/questions/:id donde primero se eliminarán los archivos del servidor asociados a la pregunta y seguidamente se eliminará la pregunta de la base de datos.

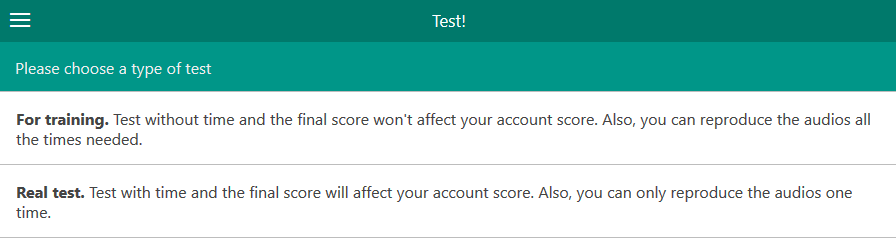




### Test

En la funcionalidad de test tenemos 2 modos:

* Entrenamiento: Tests sin límite de tiempo y la puntuación obtenida en el test no afecta a la puntuación de la cuenta del usuario. Además, los audios se pueden reproducir sin límite.
* Test real: Tests con límite de tiempo, y la puntuación obtenida sí que afectará a la puntuación final del usuario. Por otro lado, los audios sólo se pueden reproducir una vez.



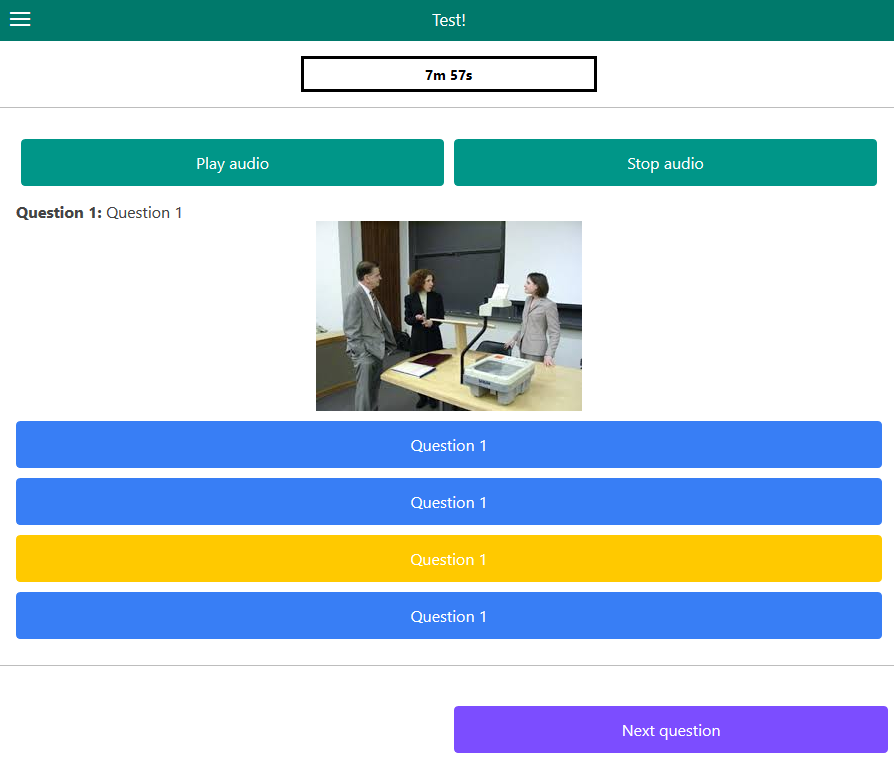
Una vez elegida la modalidad de test a realizar, se realiza una petición GET al servidor a la ruta /api/v1/questions/test en la que se seleccionará 10 preguntas del tipo seleccionado que el usuario no haya respondido correctamente con anterioridad. Al recibir las preguntas, en el cliente se mostrará la pantalla con una pregunta por pantalla, con su respectivo enunciado, respuestas, control del audio e imagen. Además se dispondrá de botones de adelante y atrás para poder navegar por las preguntas, y un botón de finalizar test en la última pregunta.

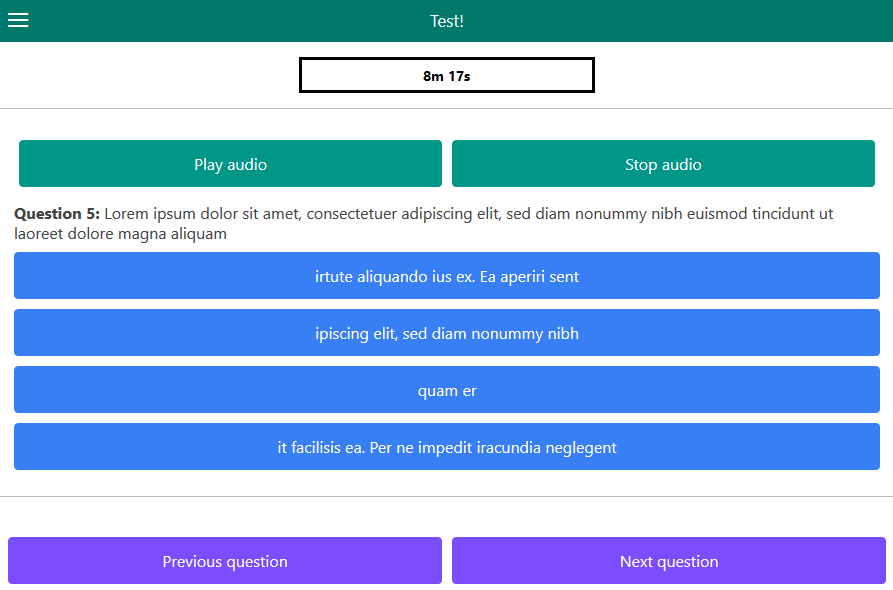
En el caso del tipo de test real, además dispondremos de un contador en la parte de arriba de la pregunta para poder visualizar el tiempo restante, además de que una vez pulsado el botón de iniciar el audio de cada pregunta, éste se bloqueará y no se permitirá volver a reproducirlo. En el caso de que el tiempo del contador llegue a 0 segundos, el test finalizará.

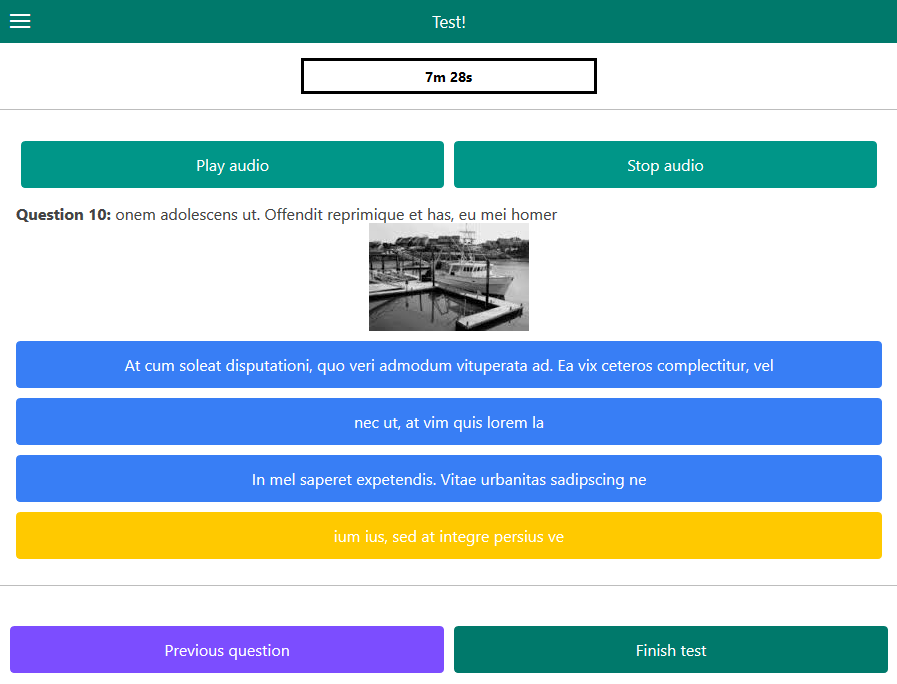
Cuando el test finalice, ya sea porque el tiempo disponible para responder el test se ha agotado o que el usuario ha pulsado en el botón de finalizar el test, en el cliente se recogerán las respuestas contestadas y se hará una petición de tipo POST a la ruta /api/v1/questions/test/resolve donde se seguirán los siguientes pasos:

1. Actualización de la puntuación total del usuario si el test era de tipo test real, teniendo en cuenta que el rango de valores a tomar es de -100 a 100 puntos. Además, se actualizará la lista de tests realizados del usuario, con el tiempo total empleado, si era de entrenamiento o test real, puntuación obtenida, número de respuestas correctas, número de respuestas incorrectas, si ha obtenido el bonus de tiempo y la fecha de realización.
2. Actualización de cada pregunta, añadiendo a la lista de usuarios que han realizado la pregunta y usuarios que han respondido a la pregunta, con la información de si era de entrenamiento o no, el id del usuario y el email del usuario.

Una vez recibido el estado 200 del servidor, en el cliente tendremos por un lado el resultado obtenido, el tiempo empleado, las respuestas correctas e incorrectas, y seguidamente el enunciado de cada pregunta junto con sus respuestas, indicando en rojo si la respuesta es incorrecta y en verde la correcta. En caso de no haber respondido a la pregunta se ha optado por no especificar la respuesta correcta.



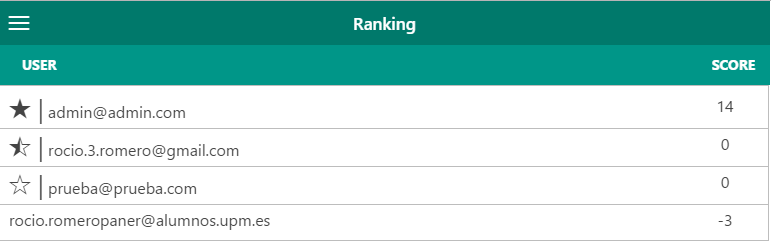






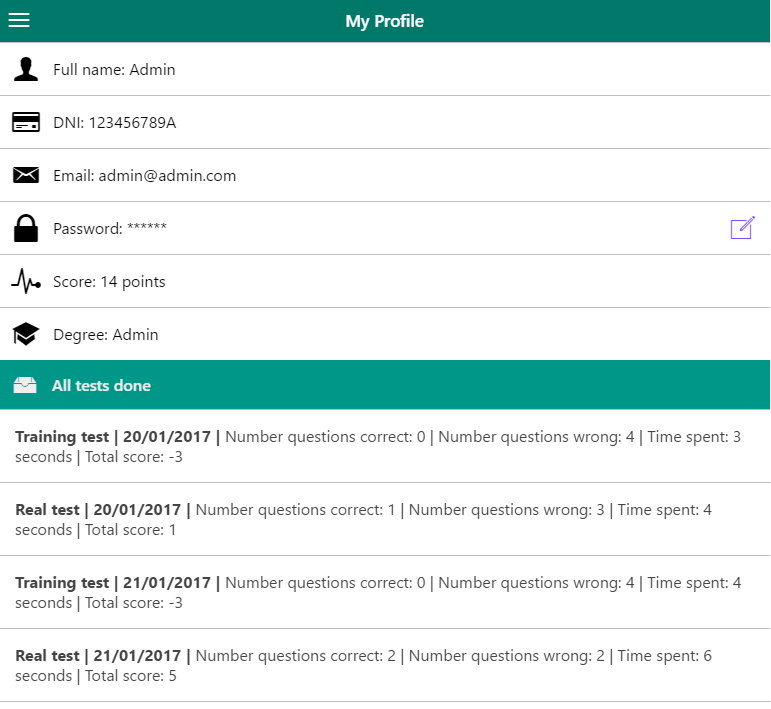
### Ranking

En la pantalla de ranking podremos ver la posición de cada usuario dependiendo de los puntos que tenga, ordenados de mayor a menor puntuación. Al entrar en la pantalla se hace una petición GET a la ruta /api/v1/users/ranking en la que se obtiene los usuarios ordenados por el campo score. El rango de la puntuación es de -100 a 100 puntos.



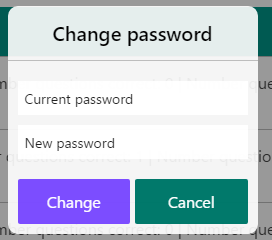
### Perfil

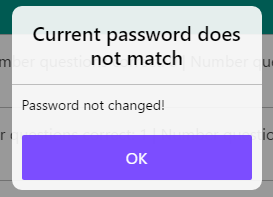
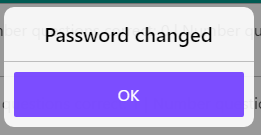
En la pantalla de mi perfil el usuario podrá visualizar toda la información relativa a su cuenta, cambiar su contraseña y ver un registro de los tests que ha hecho. Al entrar en la pantalla se hará una petición de tipo GET a la ruta /api/v1/users/:id para obtener la información específica del usuario.



Si se selecciona el botón de cambiar de contraseña, aparecerá un popup en el que tendremos que rellenar el campo de contraseña actual y nueva contraseña. Una vez pulsado en el botón “Change”, se realizará una petición al servidor de tipo PUT a la ruta /api/v1/password/:id en la que se seguirán los siguientes pasos:

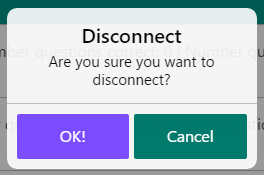
1. Comprobación de existencia del usuario en la base de datos, en caso contrario se responderá con un código de estado 404.
2. Si el usuario existe en la base de datos, se comprobará que la contraseña actual coincide con la existente, y en caso de que no coincidan se enviará un código de estado 401.
3. Por otro lado, si las contraseñas coinciden, se hasheará la nueva contraseña y se modificará en la base de datos, respondiendo con un código de estado 200.





### Desconectar

Al pulsar es desconectar aparecerá un popup de confirmación que, en caso de confirmar la desconexión, se limpiará el contenido del local storage y se redirigirá a la pantalla de login.



# Resultados y discusión

# Conclusiones y trabajos futuros

## Conclusiones

Tras la realización de este proyecto fin de carrera ...

## Trabajos futuros

Bla bla bla...

Bibliografía

1. Manual de usuario
   1. Subcapítulo anexo
      1. Sub-sub capítulo anexo
   2. Otro subcapítulo anexo