ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE
TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Trabajo Fin de Grado

Diseño e implementación con Node.js de una aplicación web para el seguimiento y evaluación del aprendizaje

AUTOR: José Joaquín Tudela Peñarrubia
DIRECTOR: Dr. Fernando Losilla López

Autor	José Joaquín Tudela Peñarrubia
E-mail del Autor	**tjosejoaquin@gmail.com**
Director	Fernando Losilla López
E-mail del Director	Fernando.losilla@upct.es
Codirector(es)	
Título del PFC	Diseño e implementación con Node.js de una aplicación web para el seguimiento y evaluación del aprendizaje.
Descriptores	e-learning, evaluación del aprendizaje

Resumen

En este trabajo se describe el desarrollo e implementación de una aplicación web que permite que distintos profesores puedan crear cuestionarios on-line para los alumnos de sus asignaturas y llevar control de los resultados obtenidos por los alumnos. Uno de los principales objetivos del proyecto es que sea fácilmente ampliable y sirva de base para introducir mejoras en un futuro que permitan adaptar los cuestionarios a cada alumno en función de sus necesidades. El proyecto se ha realizado usando tecnologías como HTML5, javascript, Node.js, MongoDB, entre otras.

Titulación	Grado en Ingeniería Telemática
Intensificación	
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	Octubre 2017

Contenido

1. Introducción

1.1 Planteamiento del proyecto

La irrupción de las nuevas tecnologías en los entornos docentes, están haciendo necesaria una revisión de los principios en los que se apoya la educación, ya que las Tecnologías de la Información y Comunicación (TIC) proporcionan nuevos planteamientos para el aprendizaje. Para ello se ha desarrollado este proyecto, una plataforma virtual donde los docentes desarrollen sus propios métodos de evaluación y que puede como punto de partida para la aplicación de diversas técnicas que mejoran la calidad de la enseñanza.

Así pues éste es un proyecto de futuro, en el que la plataforma creada es el inicio de un método de evaluación diverso, sencillo, escalable, interactivo y muy atractivo tanto para el docente como para el estudiante. Docentes y desarrolladores podrán implementar sus propios tipos de cuestiones ya sean test, desarrollo de código o cuestiones innovadoras, todo ello en una misma plataforma.

Este proyecto también es la base para introducir en el futuro mecanismos que evalúen el conocimiento del alumno y seleccionen las preguntas más apropiadas, es decir, desarrollar un algoritmo que sea capaz de identificar las debilidades del estudiante en los cuestionarios desarrollados por los docentes, siendo el propio algoritmo el que haga hincapié en estas debilidades volviendo a repetir preguntas y conceptos para que el estudiante supere los objetivos marcados.

El proyecto va a estar separado en diferentes secciones, en primer lugar se documentará al usuario de la información básica para entender las tecnologías usadas, a continuación se detallará el desarrollo del proyecto dividido en tres secciones correspondientes a los tres paneles de usuarios que utilizaran la plataforma, y finalmente se explicará con detalle aspectos importantes de la plataforma.

1.2 Objetivos del proyecto

Los objetivos del proyecto son:

- Desarrollar una plataforma con una gran escalabilidad para su futuro crecimiento.
- Implementar una plataforma sencilla e intuitiva para los usuarios.

- Crear una comunidad de docentes que puedan intercambiar implementaciones y cuestionarios.
- Usar tecnologías de vanguardia tales como Node.js, MongoDB y ExpressJS, además de la utilización de la arquitectura REST.

2. Tecnologías usadas

2.1 HTML5

HTML es un lenguaje de marcado utilizado para definir la estructura y contenido de una página o documento web. La idea de utilizar un lenguaje para hacer referencia a otros documentos, como archivos, imágenes, video, audio, etc. Es un estándar a cargo del World Wide Web Consortium (W3C), organización dedicada a la estandarización de muchas de las tecnologías dedicadas a la web, sobre todo en lo referente a su escritura e interpretación.

Desde su creación, el lenguaje HTML ha pasado de ser un lenguaje utilizado exclusivamente para crear documentos electrónicos a ser un lenguaje que se utiliza en muchas aplicaciones electrónicas como buscadores, tiendas online y banca electrónica.

Ha ido evolucionando de acuerdo con diversas necesidades que se orientan sobre todo a mejorar el procesamiento de la información, y así fue como aparecieron varias revisiones (HTML 2, HTML 3.2, HTML 4, HTML 4.01) que ampliaron y depuraron este lenguaje.

HTML5 es la última versión de HTML. Contiene un conjunto más amplio de tecnologías que permite a los sitios Web y a las aplicaciones ser más diversas y de gran alcance.

En HTML 5 se introducen nuevas etiquetas que nos permiten hacer cosas que con HTML no eran posibles. Se ha pretendido con esta nueva revisión que la escritura de código sea más sencilla, lógica y comprensible para el programador. La idea que se buscaba con HTML5 era la correcta visualización de contenido multimedia en dispositivos de gama baja, intentando prescindir de la instalación de plugins u otras tecnologías que podrían ralentizar el dispositivo o incluso no permitir visualizar la web.

Algunas de las nuevas etiquetas son:

- **article**: esta etiqueta se usa para escribir un artículo, publicación, comentario, independiente del resto del contenido.
- **header, footer**: estas etiquetas evitar tener que definir un div con un id determinado, como se solía hacer anteriormente. Además, se pueden insertar en cada sección.
- **nav**: se utiliza para definir una sección que sólo contiene enlaces de navegación.
- **section**: se utiliza para separar contenido dentro de la página, definiendo. secciones. Tiene un funcionamiento similar al div, pero de esta forma se ve claramente las separaciones entre secciones.
- **audio y video**: estas etiquetas definen el tipo de contenido multimedia que estará en su interior, además puede ser reproducido por casi todos los dispositivos.
- **embed:** con esta etiqueta se puede marcar la presencia de un contenido interactivo o aplicación externa que por lo general no es HTML.
- **canvas**: Representa un área de mapa de bits en el que se pueden utilizar scripts para renderizar gráficos como gráficas, gráficas de juegos o cualquier imagen visual al vuelo, necesitará el uso de JavaScript.

2.2 CSS3

La W3C define CSS (Cascading Style Sheets) como un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo es pronunciada la información presente en ese documento a través de un dispositivo de lectura. Es forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y forma de sus documentos.

Las hojas de estilos aparecieron poco después del leguaje de etiquetas SGML, alrededor del año 1970. Desde la creación de SGML, se observó la necesidad de definir un mecanismo que permitiera aplicar de forma consistente diferentes estilos a los documentos electrónicos.

Con el boom de internet y de HTML, la W3C decidió estandarizar CSS así pues en 1996 se publicó el primer estándar llamado “CSS nivel 1”. Con los años fueron apareciendo nuevos estándares como “CSS 2”, “CSS 2.1” y finalmente CSS3.

A diferencia de CSS2, que fue una única especificación que define varias funcionalidades, CSS3 está dividida en varios documentos llamados módulos. Algunos de estos módulos ya son recomendaciones generales de la W3C como “Selectores”, “Espacios de nombre” y “Color”.

Sus características principales son:

- **Efectos de texto:** CSS3 incorpora nuevas propiedades para crear diseños de texto más

atractivos.

- **Bordes:** Ahora podemos diseñar de manera sencilla bordes redondeados, bordes formados por imágenes y sombras sobre las cajas, entre muchas otras propiedades.
- **Animaciones:** Otra importante novedad son las animaciones, para ello nos permite cambiar de un estilo a otra de forma muy sencilla, pudiendo cambiar las propiedades de elementos HTML como queramos y cuando queramos. Para esto se utilizan las reglas de @keyframes.
- **Tranformaciones 2D/3D:** Las transformaciones en CSS3 ofrecen la posibilidad de cambiar el tamaño y la posición de los elementos con la utilización de ciertos métodos y la propiedad transform.
- **Transiciones:** Este tipo de efecto permiten cambiar el estilo de un objeto de manera progresiva. Para ello se utiliza el elemento transition, al cual le asignamos las propiedades que queremos cambiar y el tiempo de la transición en segundos.

2.3 BOOTSTRAP

Bootstrap es un entorno de trabajo o conjunto de herramientas que facilitan el desarrollo y diseño de sitios web. Está formado por plantillas de diseño con tipografías, botones, cuadros, formularios y barras de navegación, menús desplegables y otros elementos basados en HTML, CSS y JS.

Entre su características principales podemos destacar la facilidad para implementar el diseño responsive, pudiendo mostrar la información del sitio web en diferentes dispositivos electrónicos independientemente del tamaño del mismo. Para utiliza un sistema de columnas, el cual divide la pantalla en 12 columnas, donde se puede indicar el tamaño de columna según el tamaño de pantalla del dispositivo.

Bootstraps también trae un conjunto de hojas de estilos que otorga uniformidad a la página y una apariencia moderna. Cabe destacar los plugins de Javascript, que son componentes basados en la librería jQuery en los cuales podemos encontrar Modal, Dropdown, Scrollspy, Tab, Tolltip, Popover, Alert, Collapse, Carousel y Typehead.

2.4 JAVASCRIPT

JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Es conocido principalmente por su utilización en el lado del cliente, ya que es interpretado por todos los navegadores, para conseguir páginas dinámicas. Entre otras funciones podemos modificar el árbol DOM, incorporar imágenes, animaciones y acciones que se activan al pulsar botones. También usado en muchos entornos sin navegador como Node.JS o ExpressJS.

2.5 JQUERY

jQuery es una librería de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y AJAX (Asynchronous JavaScript And XML) a páginas web.

Sus principales características son:

- Es flexible y rápido para el desarrollo web.
- Viene con licencia MIT y es Open Source.
- Tiene una excelente comunidad de soporte.
- Tiene plugins.
- Bugs son resueltos rápidamente.
- Excelente integración con AJAX.
- Compatible con todos los navegadores.

2.6 NODEJS

Node.js es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8

de Chrome. Usa un modelo de E/S sin bloqueo y orientado a eventos, que lo hace liviano y eficiente. El ecosistema de paquetes de Node.js, npm, es el ecosistema más grande de librerías de código abierto en el mundo.

Concebido como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node está diseñado para construir aplicaciones en red escalables. En la siguiente aplicación de ejemplo "hola mundo", se pueden manejar muchas conexiones concurrentes. Por cada conexión el *callback* será ejecutado, sin embargo si no hay trabajo que hacer Node estará durmiendo.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hola Mundo\n');
});

server.listen(port, hostname, () => {
  console.log(`El servidor se está ejecutando en http://${hostname}:${port}/`);
});
```

Esto contrasta con el modelo de concurrencia más común hoy en día, donde se usan hilos del Sistema Operativo. Las operaciones de redes basadas en hilos son relativamente ineficientes y son muy difíciles de usar. Además, los usuarios de Node están libres de preocupaciones sobre el bloqueo del proceso, ya que no existe. Casi ninguna función en Node realiza I/O directamente, así que el proceso nunca se bloquea. Debido a que no hay bloqueo es muy razonable desarrollar sistemas escalables en Node.

Node lleva el modelo de eventos un poco más allá, este presenta un bucle de eventos como un entorno en vez de una librería. En otros sistemas siempre existe una llamada que bloquea para iniciar el bucle de eventos. El comportamiento es típicamente definido a través de *callbacks* al inicio del script y al final se inicia el servidor mediante una llamada de bloqueo como “EventMachine::run()”. En Node no existe esta llamada. Node simplemente ingresa el bucle de eventos después de ejecutar el script de entrada. Node sale del bucle de eventos cuando

no hay más *callbacks* que ejecutar. Se comporta de una forma similar a JavaScript en el navegador - el bucle de eventos está oculto al usuario.

HTTP es ciudadano de primera clase en Node, diseñado con operaciones de streaming y baja latencia en mente. Esto hace a Node candidato para ser la base de una librería o un framework web.

Solo porque Node esté diseñado sin hilos, no significa que no se pueda aprovechar los múltiples cores de su sistema. Procesos hijos pueden ser lanzados usando la API "child_process.fork()", la cual está diseñada para comunicarse fácilmente con el proceso principal. Construida sobre la misma interfaz está el módulo cluster, el cual permite compartir sockets entre procesos para activar el balanceo de cargas en sus múltiples cores.

2.7 MONGODB

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

En una base de datos orientada a documentos en lugar de guardar los datos en registros, los datos se guardan en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON.

Una de las diferencias más importantes con respecto a las bases de datos relacionales, es que **no es necesario seguir un esquema**. Los documentos de una misma colección - concepto similar a una tabla de una base de datos relacional -, pueden tener esquemas diferentes.

Imaginemos que tenemos una colección a la que llamamos Personas. Un documento podría almacenarse de la siguiente manera:

```
{
  Nombre: "Pedro",
  Apellidos: "Martínez Campo",
  Edad: 22,
  Aficiones: ["fútbol","tenis","ciclismo"],
  Amigos: [
   {
    Nombre:"María",
    Edad:22
   },
   {
    Nombre:"Luis",
    Edad:28
   }
  ]
}
```

El documento anterior es un clásico documento JSON. Tiene strings, arrays, subdocumentos y números. En la misma colección podríamos guardar un documento como éste:

```
{
  Nombre: "Luis",
  Estudios: "Administración y Dirección de Empresas",
  Amigos:12
}
```

Este documento no sigue el mismo esquema que el primero. Tiene menos campos, algún campo nuevo que no existe en el documento anterior e incluso un campo de distinto tipo.

Esto, que es algo impensable en una base de datos relacional, es algo totalmente válido en MongoDB.

Las características principales de MongoDB son:

- **Consultas Ad hoc**: MongoDB soporta la búsqueda por campos, consultas de rangos y expresiones regulares. Las consultas pueden devolver un campo específico del documento pero también puede ser una función JavaScript definida por el usuario.
- **Indexación**: Cualquier campo en un documento de MongoDB puede ser indexado, al igual que es posible hacer índices secundarios. El concepto de índices en MongoDB es similar a los encontrados en base de datos relacionales.

- **Replicación**: MongoDB soporta el tipo de replicación primario-secundario. Cada grupo de primario y sus secundarios se denomina replica “set”. El primario puede ejecutar comandos de lectura y escritura. Los secundarios replican los datos del primario y sólo se pueden usar para lectura o para copia de seguridad, pero no se pueden realizar escrituras. Los secundarios tiene la habilidad de poder elegir un nuevo primario en caso de que el primario actual deje de responder.
- **Balanceo de cargar**: MongoDB se puede escalar de forma horizontal usando el concepto de “shard”. El desarrollador elige una clave de sharding, la cual determina cómo serán distribuidos los datos de una colección. Los datos son divididos en rangos (basado en la clave de sharding) y distribuidos a través de múltiples shard. Cada shard puede ser una replica “set”. MongoDB tiene la capacidad de ejecutarse en múltiple servidores, balanceando la carga y/o replicando los datos para poder mantener el sistema funcionando en caso que exista un fallo de hardware. La configuración automática es fácil de implementar bajo MongoDB y se pueden agregar nuevas servidores a MongoDB con el sistema de base de datos funcionando.
- **Almacenamiento de archivos**: MongoDB puede ser utilizado como un sistema de archivos, tomando la ventaja de la capacidad que tiene MongoDB para el balanceo de carga y la replicación de datos utilizando múltiples servidores para el almacenamiento de archivos. Esta función se llama GridFS y es más bien una implementación en los drivers, no en el servidor, por lo que está incluida en los drivers oficiales que la compañía de MongoDB desarrolla. Estos drivers exponen funciones y métodos para la manipulación de archivos y contenido a los desarrolladores. En un sistema con múltiple servidores, los archivos pueden ser distribuidos y replicados entre los mismos y de una forma transparente, de esta forma se crea un sistema eficiente que maneja fallos y balanceo de carga.
- **Agregación**: MongoDB proporciona un framework de agregación que permite realizar operaciones similares a las que se obtienen con el comando SQL "GROUP BY". El framework de agregación está construido como un pipeline en el que los datos van pasando a través de diferentes etapas en los cuales estos datos son modificados, agregados, filtrados y formateados hasta obtener el resultado deseado. Todo este procesado es capaz de utilizar índices si existieran y se produce en memoria. Asimismo, MongoDB proporciona una función MapReduce que puede ser utilizada para el procesamiento por lotes de datos y operaciones de agregación.

- **Ejecución de JavaScript en el lado del servidor**: MongoDB tiene la capacidad de realizar consultas utilizando JavaScript, haciendo que estas sean enviadas directamente a la base de datos para ser ejecutadas.

2.8 EXPRESSJS

Express es un framework de aplicaciones web minimalista y flexible para NodeJS que proporciona un conjunto sólido de características para las aplicaciones web y móviles. Siendo un framework para el "backend" de nuestra aplicación. La versión actual de la API Express es 4.x, la cual tiene un gran número de métodos y propiedades.

```
var express = require('express')
var app = express()

app.get('/', function (req, res) {
  res.send('Hello World')
})

app.listen(3000)
```

Aquí se puede ver un pequeño ejemplo del funcionamiento de Express, primero se importa el modulo express, después se crea la variable "app" y se define que cuando reciba la ruta "/" se mande un mensaje de vuelta, en este caso "hello world". Este es solo un simple ejemplo de lo que es capaz Express, también conviene destacar algunos métodos o propiedades como:

- **app.get()**: Responde a peticiones HTTP GET de una ruta específica con sus funciones específicas de "callback".
- **app.post()**: Responde a peticiones HTTP pero en este caso peticiones POST.

- **app.put()**: Responde a peticiones HTTP pero en este caso peticiones PUT.
- **app.delete()**: Responde a peticiones HTTP pero en este caso peticiones DELETE.
- **res.send()**: Envia una respuesta HTTP, se pueden enviar textos, código html y objetos json entre otros formatos.
- **res.render()**: Renderiza una vista y la envía al cliente como una cadena de HTML. Está respuesta pues llevar información en variables que se hayan definido previamente.

2.9 REST

REST (Representational State Transfer) es un estilo de arquitectura software de desarrollo web que se apoya totalmente en el estándar HTTP.

REST nos permite crear servicios y aplicaciones que pueden ser usadas por cualquier dispositivo o cliente que entienda HTTP, por lo que es increíblemente más simple y convencional que otras alternativas que se han usado en los últimos diez años como SOAP y XML-RPC.

Los sistemas que siguen los principios REST se llaman con frecuencia RESTful.

REST afirma que la web ha disfrutado de escalabilidad como resultado de una serie de diseños fundamentales clave:

- **Un protocolo cliente/servidor sin estado**: cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en HTTP utilizan cookies y otros mecanismos para mantener el estado de la sesión (algunas de estas prácticas, como la reescritura de URLs, no son permitidas por REST).

- Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son **POST**, **GET**, **PUT** y **DELETE**. Con frecuencia estas operaciones se equiparan a las operaciones CRUD en bases de datos (CLAB en castellano: crear,leer,actualizar,borrar) que se requieren para la persistencia de datos, aunque POST

no encaja exactamente en este esquema.

- Una **sintaxis universal** para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su URI.

- El uso de hipermedios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente HTML o XML. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

Un concepto importante en REST es la existencia de recursos (elementos de información), que pueden ser accedidos utilizando un identificador global (URI). Para manipular estos recursos, los componentes de la red (clientes y servidores) se comunican a través de una interfaz estándar (HTTP) e intercambian representaciones de estos recursos (los ficheros que se descargan y se envían).

La petición puede ser transmitida por cualquier número de conectores (por ejemplo clientes, servidores, cachés, túneles, etc.) pero cada uno lo hace sin "ver más allá" de su propia petición (lo que se conoce como stateless (sin estado), otra restricción de REST, que es un principio común con muchas otras partes de la arquitectura de redes y de la información). Así, una aplicación puede interactuar con un recurso conociendo el identificador del recurso y la acción requerida, no necesitando conocer si existen cachés, proxys, cortafuegos, túneles o cualquier otra cosa entre ella y el servidor que guarda la información. La aplicación, sin embargo, debe comprender el formato de la información devuelta (la representación), que es por lo general un documento HTML o XML, aunque también puede ser una imagen o cualquier otro contenido.

2.10 Paquetes NPM

- **Body-parse**: Es un middelware para Node.js, el cual actúa entre la petición del cliente y antes de ser recibida, analizando el cuerpo de la petición y extrayendo la información de las variables de los métodos GET o POST.
- **Ejs**: Motor para renderizar vistas o templates, además de generar contenido

dinamicamente.

- **Method-override**: Permite usar los métodos PUT y DELETE en sitios donde el cliente no lo soporta.
- **Passport, Passport-local y Passport-local-mongo**: Es un middleware de autenticación para Node.js. Es flexible y modular, puede ser utilizado en cualquier aplicación web Express. Una de sus características principales es que soporta la autenticación mediante Facebook o Twitter, entre otras.
- **Lodash**: Es una librería de Javascript que hace más sencillo trabajar con arrays, números, objetos, strings, etc.
- **Flash**: Permite mostrar mensajes a Express.

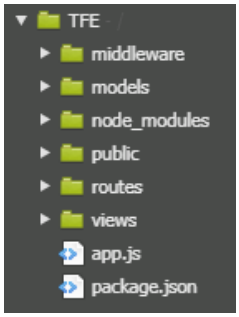
3. Desarrollo del proyecto

3.1 Objetivo

Los objetivos principales son crear una aplicación sencilla y amigable para los usuarios, además de un código legible y abierto para las futuras modificaciones o actualizaciones de la aplicación.

3.2 Estructura

Es importante describir la estructura de la aplicación web para poder entenderla, ya que en Node.js la aplicación se divide en diversos directorios donde separa vistas, módulos, rutas, middleware entre otros.



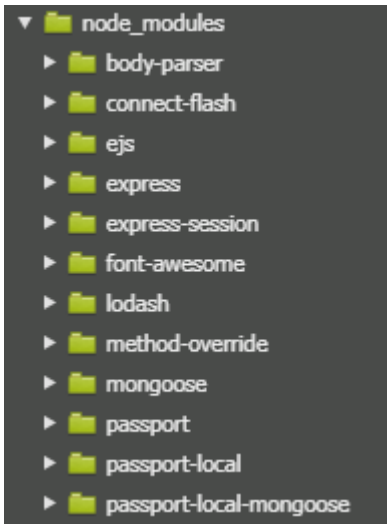
3.2.1 Package.json

Es un archivo JSON que contiene información sobre la aplicación web como el nombre, versión, archivo principal o las dependencias de la misma.

```
{
  "name": "tfe",
  "version": "1.0.0",
  "description": "Proyecto final de estudios",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Jose Joaquin Tudela Peñarrubia",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.15.2",
    "connect-flash": "^0.1.1",
    "ejs": "^2.5.2",
    "express": "^4.14.0",
    "express-session": "^1.14.1",
    "font-awesome": "^4.7.0",
    "lodash": "^4.17.2",
    "method-override": "^2.3.6",
    "mongoose": "^4.6.4",
    "passport": "^0.3.2",
    "passport-local": "^1.0.0",
    "passport-local-mongoose": "^4.0.0"
  }
}
```

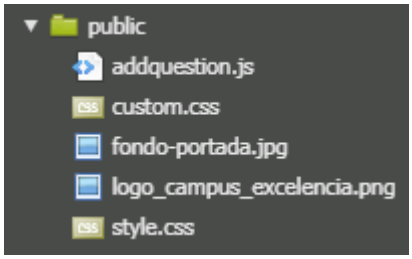
3.2.2 node_modules

En este directorio se encuentran todos los paquetes que se han instalado y que ya han sido explicados anteriormente.



3.2.3 Public

En el directorio public se encuentran todos los archivos a los cuales podemos acceder desde cualquier ubicación de la aplicación. En este caso contiene archivos de JS y CSS que se utilizan en múltiples archivos.



3.2.4 Middleware

Contiene un único documento, "index.js", el cual contiene diversas funciones middleware que se utilizan para detectar si el usuario que mantiene la sesión tiene permisos o no para entrar en dicha página.

3.2.5 App.js

Es el archivo principal de la aplicación, aquí se inicializan y configuran los paquetes. Primero se importan las librerías de los paquetes y se inicializa una instancia de Express.

```
var express = require("express");
var mongoose = require("mongoose");
var bodyparser = require("body-parser");
var passport = require("passport");
var LocalStrategy = require("passport-local");
var User = require("./models/user");
var Subject = require("./models/subject");
var methodOverride = require("method-override");
var flash = require("connect-flash");

//inicializate express
var app = express();
```

También importamos todas las rutas que se utilizan en la aplicación y seguidamente configuramos los paquetes en la instancia de Express. Algunas de las configuraciones que se ven a continuación:

```
//for no need write ejs in all views
app.set("view engine","ejs");

//Connection to databases
mongoose.connect("mongodb://localhost/tfe_v3");

//create a static route for public script
app.use(express.static(__dirname + "/public"));

//settings npm package bodyparse
app.use(bodyparser.urlencoded({extended:true}));
app.use(methodOverride("_method"));

//start flash npm
app.use(flash());

//================PASSPORT SETTINGS====================
app.use(require("express-session")({
    secret: "Diseno e implementacion con Node.js de una aplicacion web para seguimiento y evaluacion del aprendizaje",
    resave: false,
    saveUninitialized: false
}));
app.use(passport.initialize());
app.use(passport.session());

passport.use(new LocalStrategy(User.authenticate()));
passport.serializeUser(User.serializeUser());
passport.deserializeUser(User.deserializeUser());
//=========================================================
```

Cabe destacar la conexión a la base de datos y la configuración del middleware Passport para usar el sistema de autenticación.

El puerto de escucha de la aplicación también se configura aquí.

3.2.6 Views

En el directorio views se almacenan todas las vistas de la aplicación, son archivos EJS, ya que contiene código HTML y código JavaScript ejecutado desde el servidor mediante EJS. Se han creado un total de 27 vistas para la plataforma.

3.2.7 Models

Carpeta destinada a los archivos relacionados con base de datos, en este caso MongoDB, los cuales contienen la configuración de la estructura y sus relaciones.

3.2.8 Routes

Para facilitar el manejo de la aplicación es aconsejable dividir las rutas o direcciones en distintos archivos, beneficiando una futura ampliación de la aplicación. De modo que las rutas con elementos comunes estén en el mismo archivo.

3.3 Desarrollo y configuración de la base de datos en MongoDB

La base de datos de la aplicación se compone de varias colecciones que a su vez están

compuestas de documentos. Las colecciones creadas son necesarias para la plataforma, almacenando datos esenciales como preguntas, usuarios, asignaturas, etc. Por este motivo primero se explican los modelos de esquemas que se utilizan para crearlas se explicarán las colecciones y seguidamente.

3.3.1 Colecciones

Users

Contiene datos de tres tipos de usuarios alumnos, profesores y admin. Todos ellos comparten el mismo modelo de esquema, diferenciando su condición mediante el valor del campo "rol".

```
var usersSchema = mongoose.Schema({
    //common areas
    email: String,
    passport: String, //hash
    rol: String, //student/teacher/admin
    username: String,
    subject: [{type: mongoose.Schema.Types.ObjectId, ref: "Subject"}],
    //student areas
    teacher: [{type: mongoose.Schema.Types.ObjectId, ref: "User"}],
    score: [{type: mongoose.Schema.Types.ObjectId, ref: "Score"}],
    //teacher areas
    questionnaire: [{type: mongoose.Schema.Types.ObjectId, ref: "Questionnaire"}], //unsed
    active: Boolean

});
```

La principal característica de esta coleccione reside en codificar la contraseña para la privacidad de los usuarios, esto es posible al paquete npm Passport, que ya se ha explicado.

```
//npm passport
usersSchema.plugin(passportLocalMongoose);
```

De este modo se indica que cualquier inserción en esta colección tiene que ser codificada o *hasheada.*

El documento para el usuario *admin* quedaría:

```
{
  "_id" : ObjectId("5899ecdb4e890908d665698a"),
  "salt" : "65b2645d6011a2fd7a7491c9d2a46e612274cc7cc6ea35ba4537faf523d7b300",
  "hash" :   "a4692c43b0191dbcd3e79b993d4...3f162f9ee707571a086b707413d840052",
  "username" : "administrator",
  "email" : "admin@admin.com",
  "rol" : "admin",
  "active" : true,
  "questionnaire" : [ ],
  "score" : [ ],
  "teacher" : [ ],
  "subject" : [ ],
  "__v" : 0
}
```


Los campos “questionnaire”, “score”, “teacher” y “subject” quedan vacíos ya que el administrador de la plataforma no necesitara tales campos.

Para profesores:


```
{
  "_id" : ObjectId("589ca23f66c5670913176d60"),
  "salt" : "36e9f25b048ef99b96fc343a03c174a95065415f5874810667b2178e23a23566",
  "hash" : "e8eedf7e69b1019f72cdf...bf11239f550171c2caedabec545d528c9f799a",
  "username" : "profesor",
  "email" : "profesor@profesor.com",
  "rol" : "teacher",
  "active" : true,
  "questionnaire" : [ ],
  "score" : null,
  "teacher" : null,
  "subject" : [ ObjectId("58a1dc6d315bc808f32bbd78") ],
  "__v" : 1
}
```


Todos los profesores deberán tener el campo “Active” activado para poder iniciar sesión, mientras que los campos “score” y “teacher” quedan a *null*, ya que no hará uso de ellos. Los campos “questionnaire” y “subject” son listas de identificaciones que hacen referencia a otros documentos como cuestionarios creados por el profesor o asignaturas.

Para alumnos:

```
{
  "_id" : ObjectId("589cad6166c5670913176d61"),
  "salt" : "6f05cfa6114c87b5a044e01cec2b7a358550924afaa28e09f5fa1ff2cac25695",
  "hash" :"9b8738d13370431da80aa48ca55df22b0...9c4aab0b89face89f0fe5fe9521",
  "username" : "alumno",
  "email" : "alumno@alumno.com",
  "rol" : "student",
  "active" : true,
  "questionnaire" : null,
  "score" : [ ObjectId("58a57a9d5ff6ad087f977505") ],
  "teacher" : [ ObjectId("589ca23f66c5670913176d60") ],
  "subject" : [ ObjectId("58a1dc6d315bc808f32bbd78") ],
  "__v" : 2
}
```


Los alumnos en cambio solo tienen a *null* el campo "questionnaire", sin embargo si utilizan el campo "score" que servirá para su seguimiento.

Questions

Colección destinada a guardar las preguntas creadas por los profesores. El modelo de esquema queda:

```
var questionSchema = mongoose.Schema({
    type: String, //currently simple and complex
    wording: String,
    answers: Array,
    solution: Array,
    difficulty: String, //0 - 10
    tags: Array,
    topic:
        [{
            type: mongoose.Schema.Types.ObjectId,
            ref: "Topic"
        }],
    owner:
        {
            type: mongoose.Schema.Types.ObjectId,
            ref: "User"
        },
    privacy: String
});
```

En el campo "type" especificamos el tipo de pregunta, actualmente existen dos, simples y complejas, las primeras son preguntas tipo test con una única solución mientras que las segundas pueden tener una o varias soluciones, además de estos dos tipos se pueden implementar más tipos sin aplicar apenas cambios en la plataforma. Cada pregunta está asociada a un usuario o profesor mediante el campo "owner" y decidir si es una pregunta pública o privada por medio del campo "privacy", de modo que el docente puede decidir si los demás profesores pueden o no ver sus preguntas.

```
{
  "_id" : ObjectId("58a1d66f315bc808f32bbd77"),
  "type" : "complex",
  "wording" : "Cual es el resultado de la operación raiz(4)",
  "difficulty" : "Fácil",
  "privacy" : "public",
  "owner" : ObjectId("589ca23f66c5670913176d60"),
  "topic" : [ ],
  "tags" : [ "mates" ],
  "solution" : [ "0", "1" ],
  "answers" : [ "2", "-2", "1", "4" ],
  "__v" : 0
}
```

Questionnaires

Esta colección guarda un conjunto de preguntas para crear cuestionarios o un test. Como en la colección de preguntas también tiene asociado un profesor, puede ser pública o privada y además en este caso están asociados a asignaturas.

Score

Registra las calificaciones del alumno, en el documento se detallan el cuestionario al que pertenece, fecha, puntuación y número de intentos. Además se registran todos los intentos del alumno a parte del mejor, primer y último intento, todos ellos por referencia.

```
//score of a student
var scoreSchema = mongoose.Schema({
    username: String,
    date: String,
    score: String,
    questionnaire: {type: mongoose.Schema.Types.ObjectId, ref: "Questionnaire"},
    num_attemps: Number,
    best: { "type": mongoose.Schema.Types.ObjectId, "ref": 'Submission' }, //the best attempt
    last: { "type": mongoose.Schema.Types.ObjectId, "ref": 'Submission' }, //the last attempt
    first: { "type": mongoose.Schema.Types.ObjectId, "ref": 'Submission' }, //first attempt
    submissions: [{ "type": mongoose.Schema.Types.ObjectId, "ref": 'Submission'}], //all attempt
    ok_after_attemps: Number
});
```

Submission

Básicamente registra cada uno de los intentos del alumno, para ello guarda nombre de usuario, fecha, puntuación, cuestionario, número de intento y la respuesta del alumno, esta última se guarda como un objeto de Javascript.

```
var submissionSchema = mongoose.Schema({
    username: String,
    date: String,
    score: String,
    questionnaire: {type: mongoose.Schema.Types.ObjectId, ref: "Questionnaire"},
    answer: mongoose.Schema.Types.Mixed,
    num_submission: Number
});
```

Subject

Asignaturas creadas por los profesores y cuestionarios asociadas a ellas.

```
var subjectSchema = mongoose.Schema({
    name: String,
    description: String,
    questionnaire:
        [{
            type: mongoose.Schema.Types.ObjectId,
            ref: "Questionnaire"
        }],
    owner: {type: mongoose.Schema.Types.ObjectId, ref: "User"}
});
```

Topic

Registra los temas creados por profesores.

```
var topicSchema = mongoose.Schema({
    name: {type:String,unique:true}
});
```

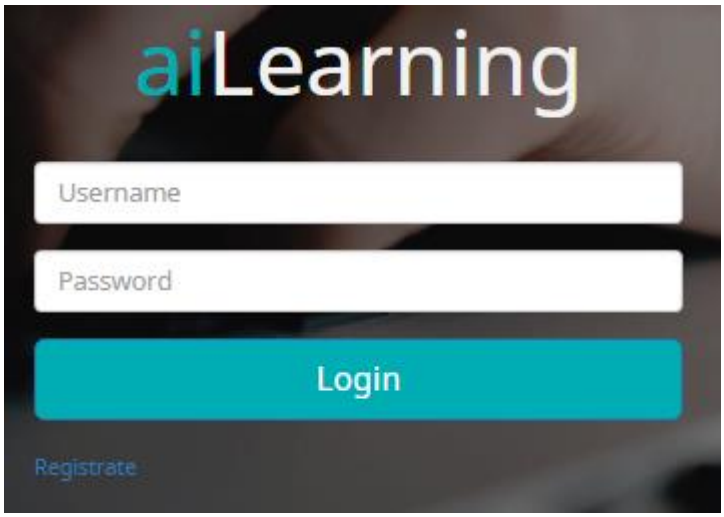
3.4 Registro e inicio de sesión de usuarios

3.4.1 Administrador

El administrador es creado cuando la aplicación web se inicia por primera vez, es decir, cuando se inicia se ejecuta automáticamente el archivo principal (app.js), el cual contiene un pequeño código que crea el perfil del administrador con unos datos preestablecidos.

```
User.findOne({username:'administrator',rol:'admin'},function(err,fuser){
    if(err){
        console.log("Error: "+err);
    }else{
        if(fuser == null){
            console.log(fuser);
            var newuser = new User({username: 'administrator', email:'admin@gadministrador.com',rol: 'admin',active:true});
            User.register(newuser, 'contraseña', function(err, user){
             if(err){
                console.log(err);
            }else{
                console.log(user);
            }
            });
        }
    }
});
```

En el formulario de la página de bienvenida de la plataforma se introducen las credenciales, las credenciales son verificadas y se accede al panel de administrador.



Para comprobar las credenciales de los usuarios el formulario manda una petición POST a la ruta "/" con los credenciales, estos son recogidos con el paquete npm "bodyparser". Utilizando el middleware Passport se comprueba si el nombre de usuario y contraseña son correctas, en caso de ser incorrectas la plataforma nos indica que no existe ningún usuario con esas credenciales impidiendo así acceder a la plataforma.

```
router.post("/",passport.authenticate("local",{failureRedirect: "/"}),function(req, res) {
    var user = req.body.username.toLowerCase();
    User.findOne({username:user},function(err,fuser){
        if(err){
            console.log(err);
        }else{
          if(!fuser.active){
            return res.redirect("/");
          }
          if(fuser.username === 'administrator' && fuser.rol === 'admin'){
            return res.redirect("/administrator");
          }
          res.redirect("/user/"+fuser._id);
        }
    });
});
```

3.4.2 Profesores

Todos los profesores tienen que registrarse para poder acceder a la plataforma, para ello rellenaran los datos del formulario requerido, que se encuentra en la ruta "/usersingup" (la cual mediante una petición GET renderiza la vista).

Una vez introducido los datos, se envían mediante una petición POST a la ruta "/user", donde se crea el usuario con sus credenciales, este usuario solo será creado si no existe ya un perfil con el mismo nombre e e-mail. Debido a que cualquier persona podría seleccionar la opción "Teachers" el campo "active" se inicializa a *false*. Siendo el administrador el único con la capacidad para activarlo desde su panel.

```
router.post("/",function(req,res){
    if(!req.body.email || !req.body.rol){
        console.log("faltan los valores de email o rol");
        return res.redirect("back");
    }
    //if password is different return error
    if(req.body.password != req.body.cpassword){
        req.flash('info', 'Las contraseñas no son iguales');
        return res.redirect("back");
    }
    var newuser;
    //all users sign up with lowercase
    var name = req.body.username.toLowerCase();
    var mail = req.body.email.toLowerCase();

    //for teacher, inicializate "active" to false because admin must active
    if(req.body.rol === 'teacher'){
        newuser = new User({username: name, email:mail,rol: req.body.rol,teacher:null,score:null,active:false});
    }else{
        //for student
        newuser = new User({username: name, email:mail,rol: req.body.rol,questionnaire:null,active:true});
    }
    //register user
    User.register(newuser, req.body.password, function(err, user){
         if(err){
            console.log(err);
            req.flash('info', 'El nombre de usuario "'+name+'" ya esta en uso');
            res.redirect("back");
        }else{
            //return confirmation and redirect
            req.flash('info', 'El usuarui "'+name+'" se ha registrado correctamente');
            res.redirect("back");
        }
    });
});
```

Una vez creado el perfil de profesor y de ser activado por el administrador, el profesor podrá iniciar sesión al igual que el administrador. Sin embargo, el profesor accederá a su panel.

3.4.3 Alumnos

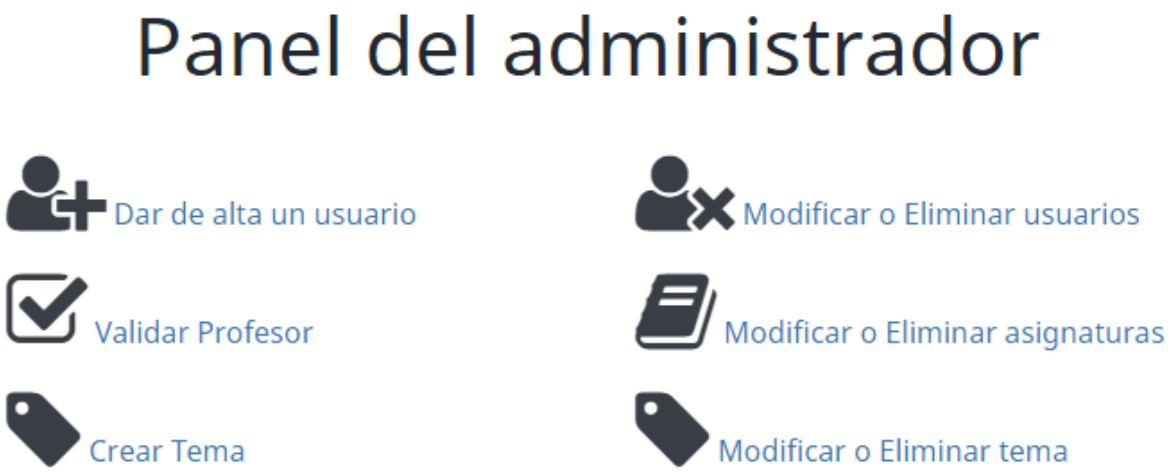
El alumno tiene que registrarse al igual que el profesor pero este tiene que elegir la opción "Student", al contrario que el profesor el alumno se inicializa activado por defecto. Iniciará sesión de la misma forma que administrador y profesor, aunque el alumno accederá a su panel.

3.5 Descripción de paneles de usuarios

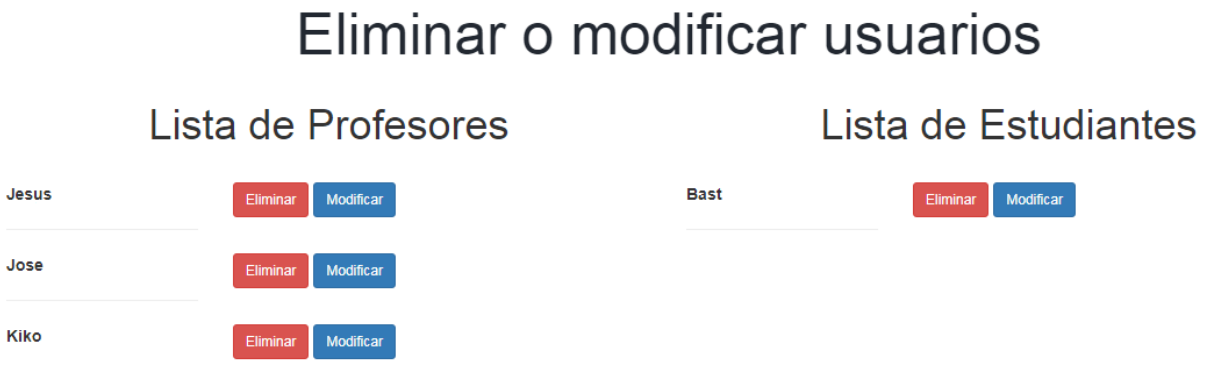
La plataforma se divide en tres paneles bien diferenciados, el panel del administrador, el panel del profesor y el panel de alumno.

3.5.1 Panel de Administrador

El administrador debe poder controlar la mayor parte de las funciones de la plataforma, para ello sea creado un panel donde este es capaz de añadir, modificar o eliminar usuarios, asignaturas, temas y activar profesores entre otras funciones.

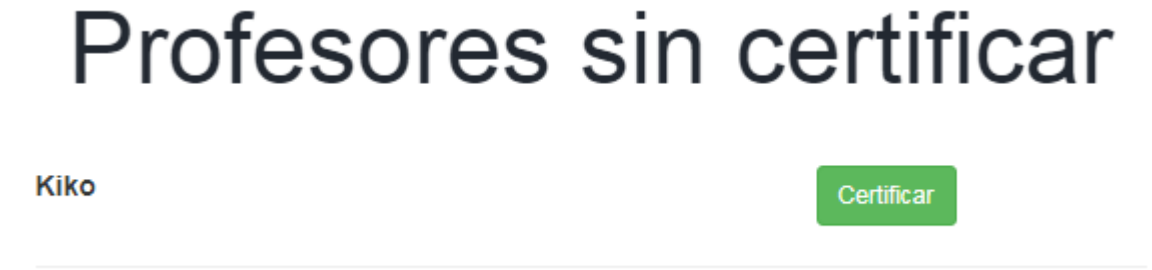


En el enlace "Dar de alta un usuario" lleva a la misma pestaña en la que profesores y alumnos se dan de alta en la plataforma. Mientras que "modificar o eliminar usuarios" nos lleva a una página donde se podrá modificar el nombre de usuarios o eliminar los mismos.



El administrador podrá crear, modificar y eliminar temas desde las pestañas "crear tema" y "modificar o eliminar tema".

Una de las funciones más importantes del administrador es validar a los profesores que emplear la plataforma, para ello se ha creado la pestaña "validar profesor". En la cual entramos en una página sencilla donde fácilmente se pueden activar a los profesores.



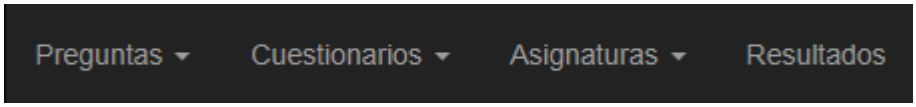
Rutas REST utilizadas

Recurso	Path	GET	PUT	POST	DELETE
Usuarios	/user	Devuelve lista de usuarios	_	Crea un nuevo usuario	_
Usuario	/user/{id}	_	Sobreescribe la información de usuario	_	Borra un usuario
Usuario	/user/{id}/edit	Devuelve información de un usuario	_	_	_
Usuario (profesor)	/user/{id}/validate	_	Sobreescribe el campo active de un profeosor		
Asignaturas	/subject	Devuelve lista de asignaturas	_	Crea una nueva asignatura	_
Asignatura	/subejct/{id}	_	Sobreescribe la información de una	_	Borra una asignatura
Asignatura	/subject/{id}/edit	Devuelve información de una asignatura	_	_	_
Temas	/topic	Devuelve lista de temas	_	Crea un nuevo tema	_
Tema	/topic/{id}	_	Sobreescribe la información de un tema	_	Borra un tema
Tema	/topic/{id}/edit	Devuelve la información de un tema	_	_	_

3.5.2 Panel de Profesor

Las funciones principales de un profesor son crear preguntas, cuestionarios y asignaturas, las cuales serán contestadas por los alumnos. Y se podrá seguir la evolución del alumno en una de las pestañas del panel llamada “Resultados”.

El panel está confeccionado de manera sencilla e intuitiva para que cualquier profesor pueda utilizarla sin problemas o sin muchas dificultades.



Se explicará inicialmente cómo se crean asignaturas, después cómo se introducen las preguntas, luego cómo se agrupan estas para crear un cuestionario y por ultimo ver los resultados de los alumnos.

Asignaturas

Esta pestaña del menú se divide a su vez en dos más, una es “Crear asignatura” y la otra es “Ver mis asignaturas”. En una se crearan las asignaturas introduciendo nombre y una breve descripción, y en la otra pestaña se verán las asignaturas creadas. Además en esta última también se podrán modificar y eliminar.

El formulario de registro de asignaturas es muy simple, como se ha dicho anteriormente es un campo para el nombre y otro para la descripción, estos campos se mandan mediante una petición POST a la ruta “/subject” en la cual se guarda. Y para eliminar una asignatura se manda una petición DELETE a la ruta “/subject/{id}”, mientras que para modificar envía una petición PUT a la misma ruta.

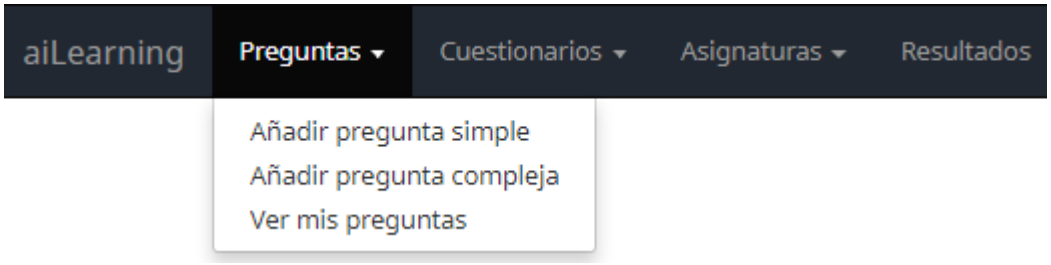
En la pestaña de “Ver mis asignaturas” además de eliminar y modificar las asignaturas también hay un botón llamado “enlace”. Este botón imprime un enlace por pantalla, el cual el profesor debe pasar a los alumnos para que se matriculen en su asignatura. El enlace cambiara según el dominio en el cual la plataforma este alojada, seguido de los identificadores del profesor y la

asignatura.

```
https://webdev-josej-
1.c9users.io/link/589ca23f66c5670913176d60/58a1dc6d315bc808f32bbd78
```

Preguntas

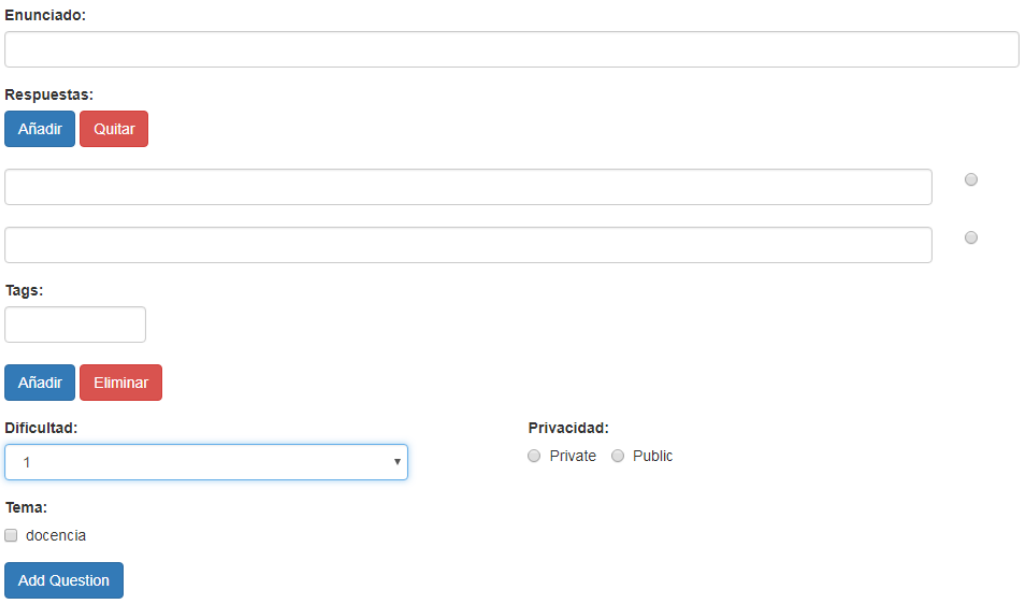
Dentro de la pestaña preguntas distinguimos cuatro subsecciones, una es para introducir preguntas simples, otra para preguntas complejas, otra para ver las preguntas del profesor y otra para ver las preguntas que son públicas (que pueden ver y utilizar los demás profesores de la plataforma).



Pregunta simple

Esta página sirve para crear preguntas simples, que son preguntas tipo test con una única solución, estas preguntas podrán tener múltiples respuestas pero solo una de ellas será la solución a la pregunta dada.

El formulario para crear preguntas consta de un campo para el enunciado, otro para añadir las repuestas, también un campo para añadir "tags" que el profesor crea convenientes, así como la dificultad de la pregunta (de 1 a 10), elegir temas relacionados y por último seleccionar si la pregunta será pública o privada.

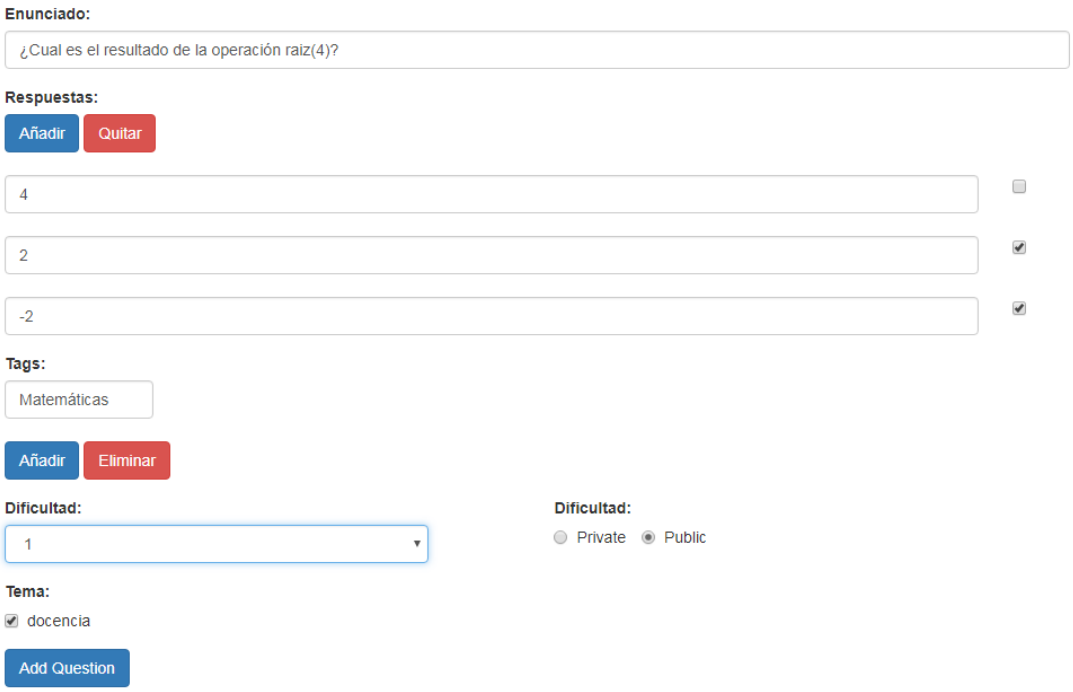


Una vez rellenado el formulario la pregunta se guardara enviando una petición POST con los datos de la pregunta a la ruta "/question", donde se extraen los datos y se guardan con la función de "create" del middleware Mongoose.

```
Question.create({type:type,wording:wording,answers:answers,solution:solution,tags:tags,difficulty:difficulty,privacy:privacy,owner:owner},function
    if(err){
        console.log(err);
        req.flash('info', 'NO se ha añadido correctamente');
        res.redirect("back");
    }else{
        //look for a topic by id
        Topic.find({'_id':{$in:topic}
        },function(err,ftopics){
            if(err){
                console.log(err);
                req.flash('info', 'NO se ha añadido correctamente');
                res.redirect("back");
            }else{
                ftopics.forEach(function(topic){
                    question.topic.push(topic);
                });
                question.save();
                console.log("despuesde save: "+question);
                req.flash('info', 'Se ha añadido correctamente');
                res.redirect("back");
            }
        }
        );
    }
});
```

Pregunta compleja

Es igual que la pestaña de preguntas simples excepto que esta puede tener varias soluciones. De modo que en el formulario se tienen que marcar las respuestas correctas.



Las preguntas se registran exactamente igual que las preguntas simples.

Ver mis preguntas

Aquí se muestran todas las preguntas creadas por el profesor, se pueden ver enunciados, respuestas con las soluciones marcadas, dificultad y tags.

Cuestionarios

En esta sección del panel el profesor podrá crear o eliminar cuestionarios, además de activarlos o desactivarlos para que sus alumnos puedan contestarlos.

Tras crear una asignatura y varias preguntas ya se podrá crear cuestionarios, que no son más que un conjunto de preguntas asociadas a una asignatura.

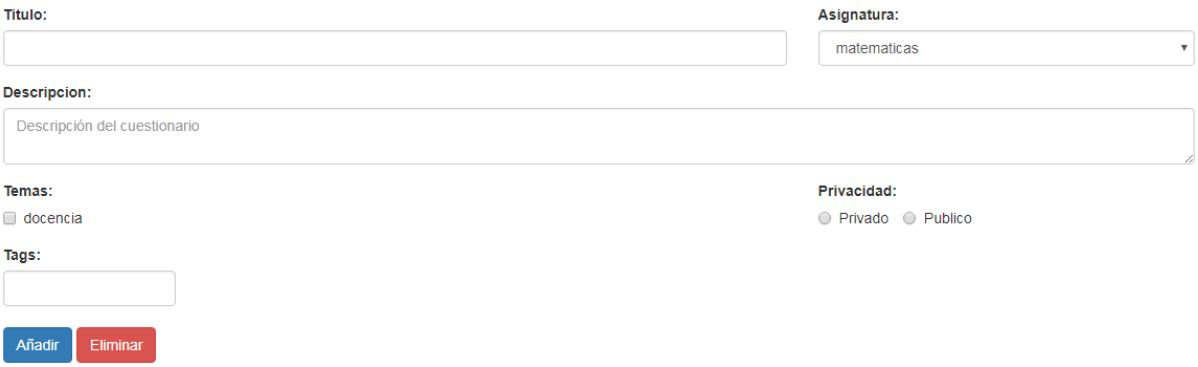
Esta sección se divide a su vez en dos subsecciones, una para crear los cuestionarios llamada “Crear cuestionario” y en la otra se llama “Ver mis cuestionarios” en la cual cada profesor podrá ver sus cuestionarios, eliminarlos o activar y desactivar los mismos.

Crear cuestionarios

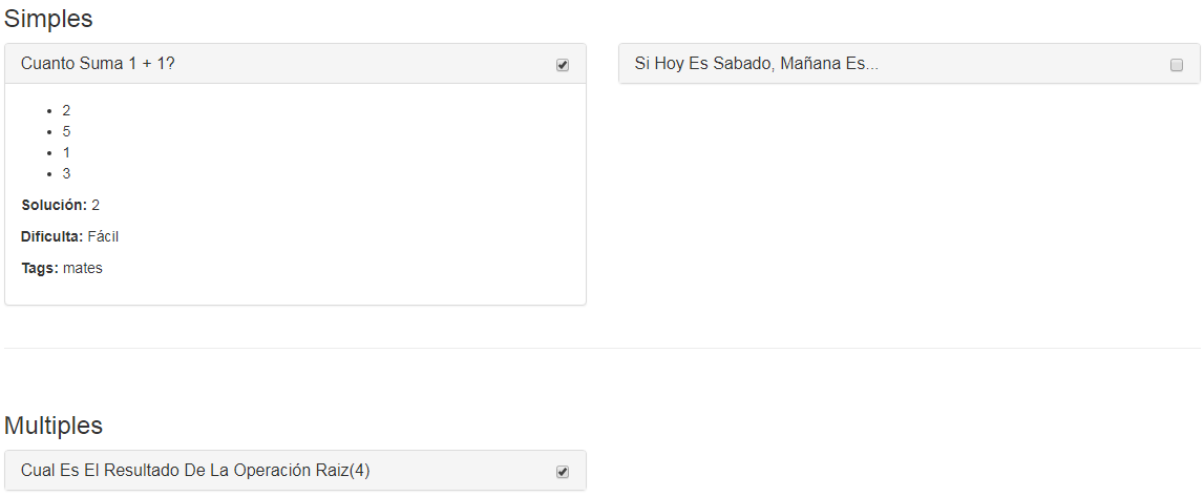
Sección donde los profesores a través de un formulario crean sus cuestionarios, estos estarán

asociados a una asignatura y contendrán preguntas que haya creado previamente el profesor u otros profesores siempre y cuando estas preguntas se registrarán como públicas. Se podrán seleccionar preguntas de cualquier tipo para formar parte del cuestionario.

La primera parte del formulario se introduce datos como el título, asignatura, descripción, temas, privacidad o tags.



Y en la segunda parte del formulario se seleccionan las preguntas que el profesor deseé, para facilitar la elección se pueden pre visualizar y están divididas por tipo.



Una vez rellenado el formulario se enviará mediante un petición POST a la ruta "/questionnaire" en la cual se registrara principalmente con el método "Create" del middleware Mongoose.

```
Questionnaire.create({title:req.body.title,subject:req.body.subject,description:req.body.description,tags:req.body.tags,owner:req.body.owner,privacy:req.
body.privacy,active:false},function(err,questionnaire){
    if(err){ req.flash('info', 'No se ha podido añadir'); res.redirect("back");
    }else{
        Topic.find({'_id':{$in:topic}},function(err,ftopics){
            if(err){ req.flash('info', 'No se ha podido añadir'); res.redirect("back");
            }else{
                ftopics.forEach(function(topic){
                    questionnaire.topic.push(topic);
                });
                Question.find({'_id':{$in:question}},function(err,fquestion){
                    if(err){ req.flash('info', 'No se ha podido añadir'); res.redirect("back");
                    }else{
                        fquestion.forEach(function(question){
                            questionnaire.questions.push(question);
                        });
                        questionnaire.save();
                        req.flash('info', 'El cuestionario '+questionnaire.title+'se ha añadido correctamente, puedes activarlo en la pestaña del menu "mis
                        cuestionarios"'); res.redirect("back");
                    }
                });
            }
        });
    }
});
```

Ver mis cuestionarios

Aquí se podrán visualizar los cuestionarios creados con sus preguntas, además de eliminarlos y activarlos o desactivarlos.

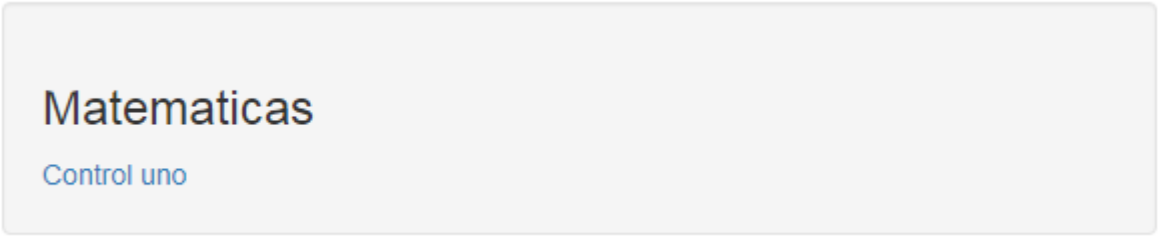
Para poder ver los cuestionarios de forma cómoda aparecerán en forma de acordeón, es decir, se verán los títulos de los cuestionarios y al pulsar en ellos se desplegaran y se podrán ver las preguntas.

Es importante la función del botón “activado/desactivado”, cuando éste está activado el alumno verá el cuestionario y podrá responder a las preguntas. Mientras que si esta desactivado el cuestionario no estará y ningún alumno podrá realizar el cuestionario. La función “activado/desactivado” cambia de forma asíncrona mediante una llamada AJAX, la cual es una llamada HTTP PUT actualizando así el estado.

Resultados

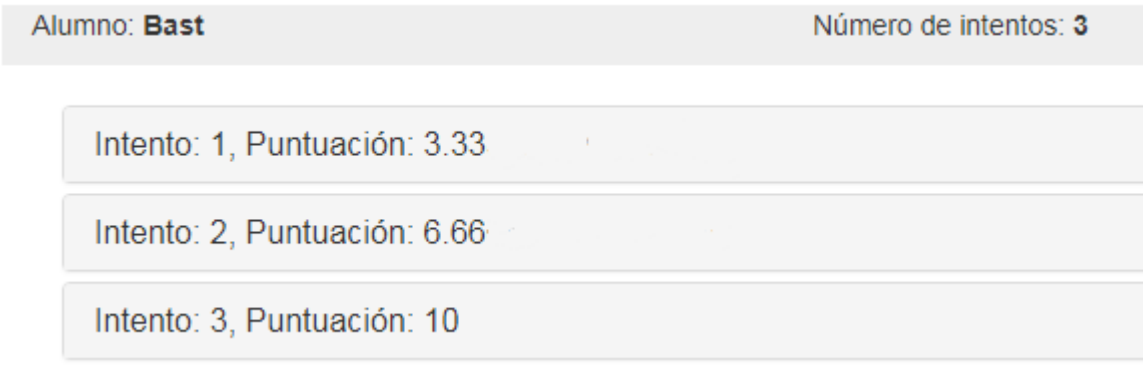
La idea principal de esta sección mostrar la mayor cantidad de información sobre los resultados del alumno, de modo que los profesores puedan realizar análisis profundos de estos datos. Por ello se registran los resultados de cada intento de los cuestionarios, el tiempo, el número intentos antes de acertar y cada una de los errores o aciertos del alumno.

Al entrar en esta sección se verán todas las asignaturas del profesor, y por cada asignatura sus cuestionarios. Así pues con una primera visión se pueden ver la mayoría de asignaturas y cuestionarios realizados.

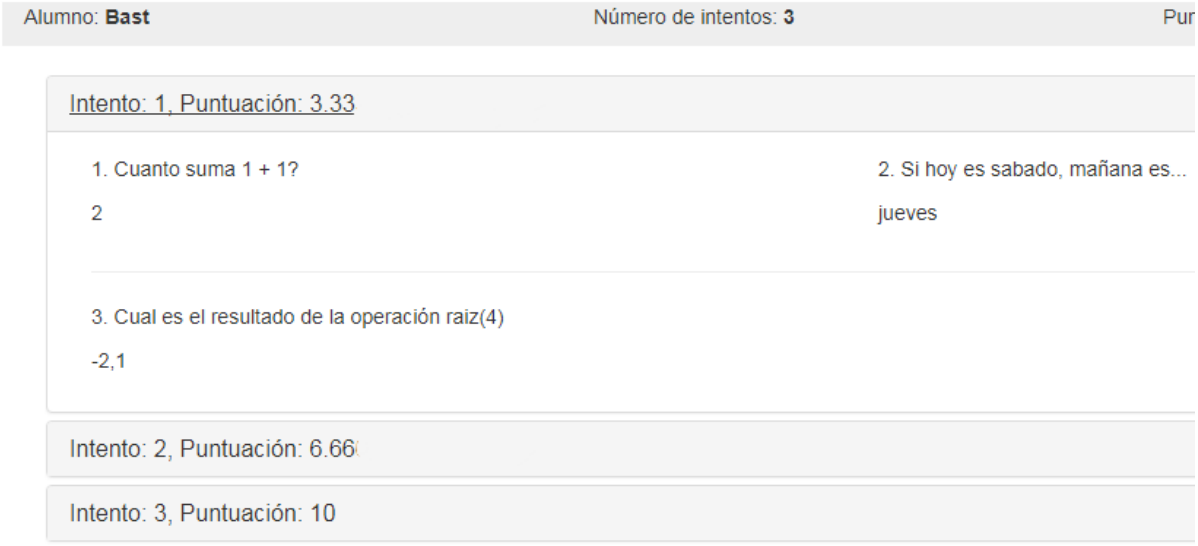


Para ver los resultados de un cuestionario el profesor pulsara en el enlace del cuestionario y le llevara a una página interna con todos los datos del mismo. Una vez dentro se observaran todos los alumnos que han realizado el cuestionario, viéndose el nombre, el número de intentos y su puntuación.

Así se ve cuantos alumnos han participado y que nota tienen pero para saber más sobre sus repuestas hay que pulsar sobre el alumno, de modo que se desplegara una lista con cada uno de los intentos del alumno.



A su vez, se puede pulsar en cada uno de los intentos y se verán todas la preguntas con las respuestas que el alumno ha seleccionado.

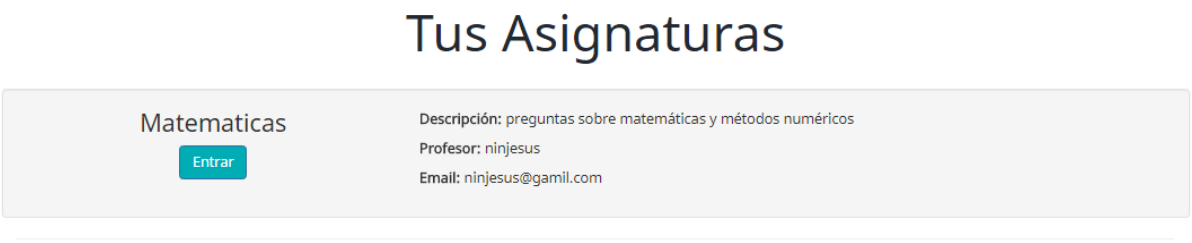


Rutas REST utilizadas

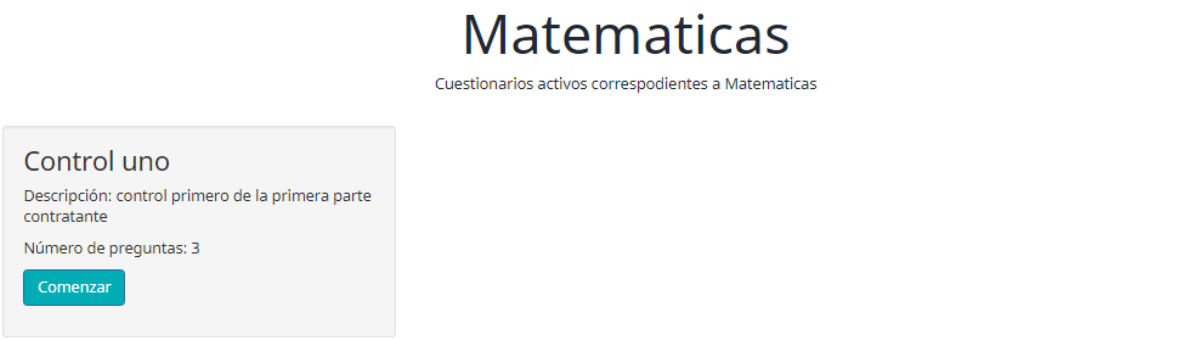
Recurso	Path	GET	PUT	POST	DELETE
Usuario	/user/{id}	Devuelve información de un usuario	–	–	–
Asignatura	/subject/{id.profesor}	–	–	Crea una nueva asignatura para un profesor	–
Asignaturas	/user/{id.profeosor}/subjects	Devuelve todas las asignaturas de un profeosr	–	–	–
Asignatura	/subject/{id}	–	Sobreescribe una asignatura	–	Borra una asignatura
Pregunta	/question	Devuelve preguntas de todos los profesores	–	Crea una nueva pregunta	
Preguntas	/user/{id}/questions	Devuelve todas las preguntas de un profesor	–	–	–
Cuestionario	/questionnaire	–	–	Crea un nuevo cuestionario	
Cuestionarios	/user/{id}/questionnaires	Devuelve todos los cuestionarios de un profesor	–	–	–
Cuestionario	/questionnaire/{id}	–	Sobreescribe el campo active de un cuestionario	–	Borra un cuestionario
Resultados	/score/{id.cuestionario}	Devuelve todo los resultados de un cuestionario	–	–	–

3.5.3 Panel de Alumno

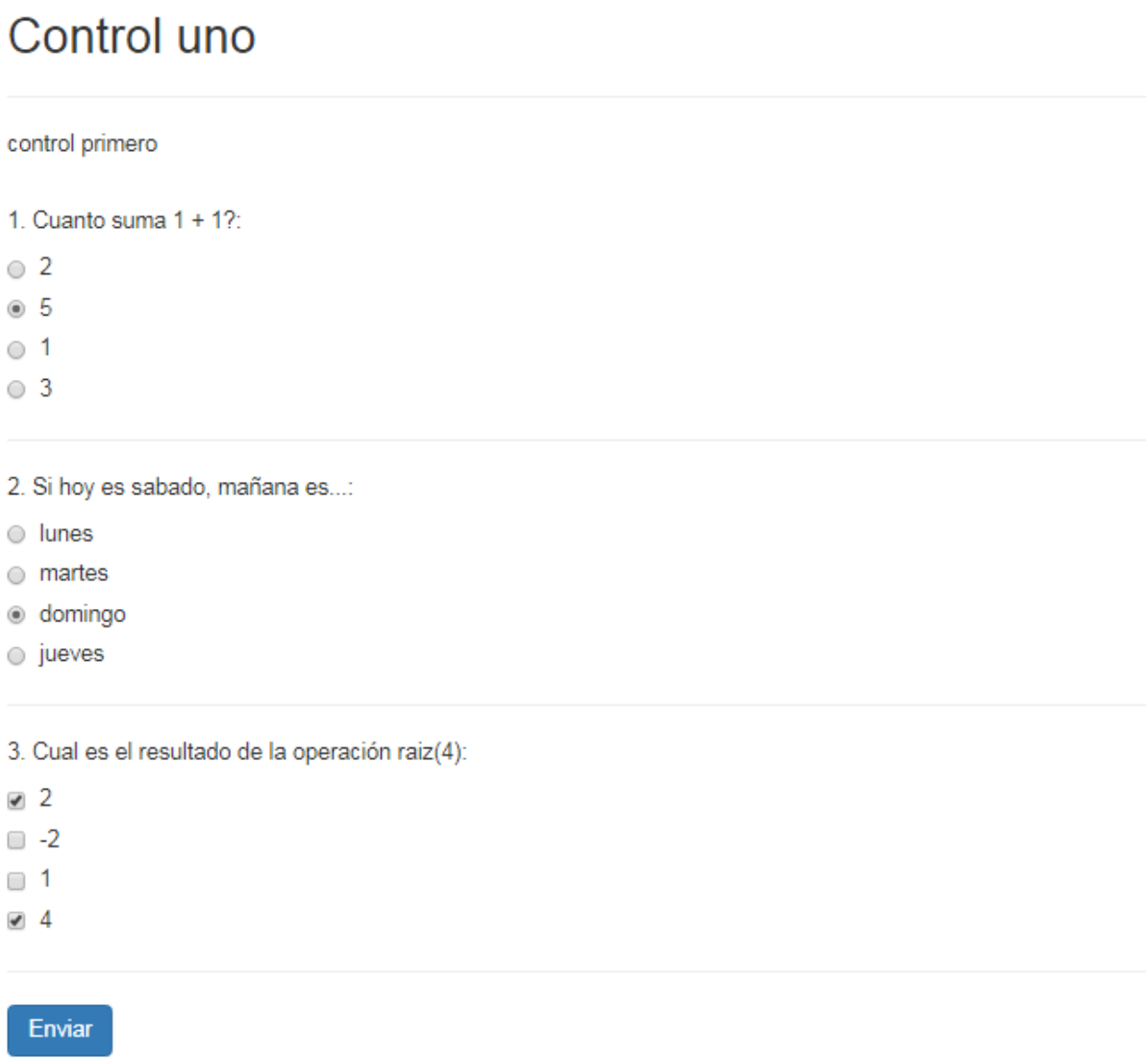
Apartado de la plataforma destinado a los alumnos, siendo la resolución de cuestionarios su principal uso. El alumno tendrá acceso a todas sus asignaturas a las cuales este registrado y sus respectivos cuestionarios siempre y cuando estos estén activados.



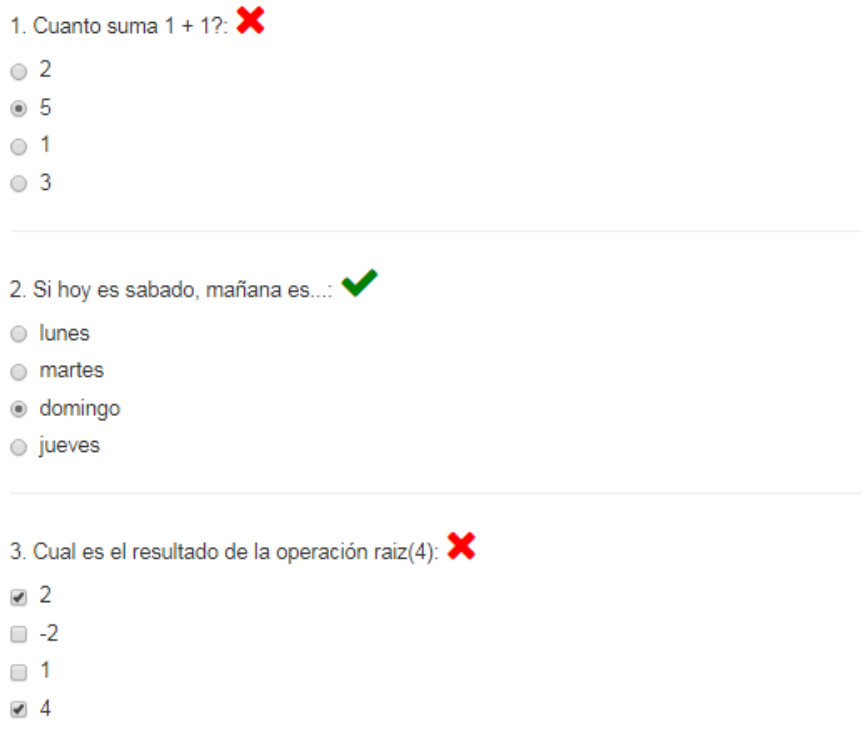
En la página principal del alumno se mostrara un listado de las asignaturas del alumno, pulsando en ellas llegamos a una página interior en la cual se muestras todos los cuestionarios activos de esa asignatura.



Para realizar el cuestionario simplemente el alumno pulsara en el botón “comenzar”, así entrara en el panel del cuestionario y vera las preguntas. Internamente esta llamada se hace mediante GET con la ruta “user/id.cuestionario/id.alumno”, indicando la id del cuestionario a realizar y la id del alumno que va a realizarlo, obteniendo así el recurso que se desea.



Una vez contestadas las preguntas el alumno las enviara y podrá ver cuántas de ellas son correctas y cuantas incorrectas, después de esta información podremos seguir contestando y volviendo a enviar. Este proceso de envío y respuesta es totalmente asíncrono ya que utilizamos AJAX, enviando las preguntas y sus respuestas con el método POST a la ruta "/submission/id.cuestionario". De este modo, el servidor es el encargado de procesar la respuesta del alumno, registrar el intento y devolver la respuesta. Se debe destacar que este tráfico de información se hace utilizando los objetos JSON.



Así pues, se crea un sistema de intercambio de información sencillo y transparente para el alumno, donde toda la información se registra y no satura el sistema.

Asociación de asignaturas

Es necesario explicar cómo el alumno puede ver las asignaturas de un profesor en su panel, o dicho de una forma más técnica, cómo la entidad del alumno se asocia a las múltiples asignaturas que crean los profesores.

El proceso es sencillo, primero el profesor debe pasar un enlace a sus alumnos, este enlace se obtiene en el panel del profesor en el apartado "mis asignaturas" que ya se explicó anteriormente. Una vez que el profesor comparte el enlace con los alumnos, estos aceden a ese

enlace y automáticamente quedan registrados. Si el alumno aún no está registrado en la plataforma primero se tendrá que registrar y de esta forma también se asociará a esta asignatura.

Rutas REST utilizadas

Recurso	Path	GET	PUT	POST	DELETE
Usuario	/user/{id}	Devuelve información de un usuario	_	_	_
Cuestionario	/questionnaire/{id.cuestionario}/{id.alumno}	Devuelve información de un cuestionario asociado a un alumno	_	_	_
Envio respuesta	/submission/{id}	_	_	Resgritro de respuesta	_

3.5.4 Aspectos Representativos de la Plataforma

Se deben destacar algunos aspectos de la plataforma debido a su importancia o que necesitan ser mencionados para el entendimiento correcto de la plataforma.

Registro de respuestas

Una de las funcionalidades más importante de la plataforma es el registro de la respuesta de un cuestionario realizado por un alumno, sobre todo para la escalabilidad de la plataforma ya que en un futuro se podrán añadir nuevos tipos de preguntas. Cuando el alumno envía su respuesta el servidor analiza y registra la respuesta, dependiendo de los resultados se devolverá una respuesta u otra.

Dado que se podrán añadir distintos tipos de preguntas en el futuro, la plataforma tiene que seguir un estándar a la hora de enviar la información de las respuestas de los alumnos. La información se enviara mediante AJAX y como un objeto de JavaScript, así pues el formato que sigue la plataforma es la id de la pregunta como clave y la respuesta como valor, este valor principalmente es string, array o entero pero admite más tipos de valores. Un ejemplo de este formato seria así:

{id.pregunta: “respuesta”,id.pregunta2: [respuesta2.1,respuesta2.2]}

Este es el estándar que debe seguirse en la plataforma, de modo que para futuros tipos de preguntas los formularios de estas preguntas tienen que contener en el campo “name” de las etiquetas “<inputs>” la id de la pregunta, ya que de ahí se extrae.

```
<!-- questions type simple -->
<% if(que.type === 'simple'){ %>
    <% que.answers.forEach(function(ans){ %>
        <div class="radio">
            <label><input type="radio" name="<%= que._id %>" value="<%= ans %>"><%= ans %></label>
        </div>
    <% }); %>
<!-- question type complex -->
<% }else if(que.type === 'complex'){ %>
    <% que.answers.forEach(function(ans){ %>
        <div class="radio">
            <label class="checkbox"><input type="checkbox" name="<%= que._id %>" value="<%= ans %>"><%= ans %></label>
        </div>
    <% }); %>
<% }else{ %>
<!-- ###################################
if you want add new type questions, here is the place
################################### -->
<% } %>
```

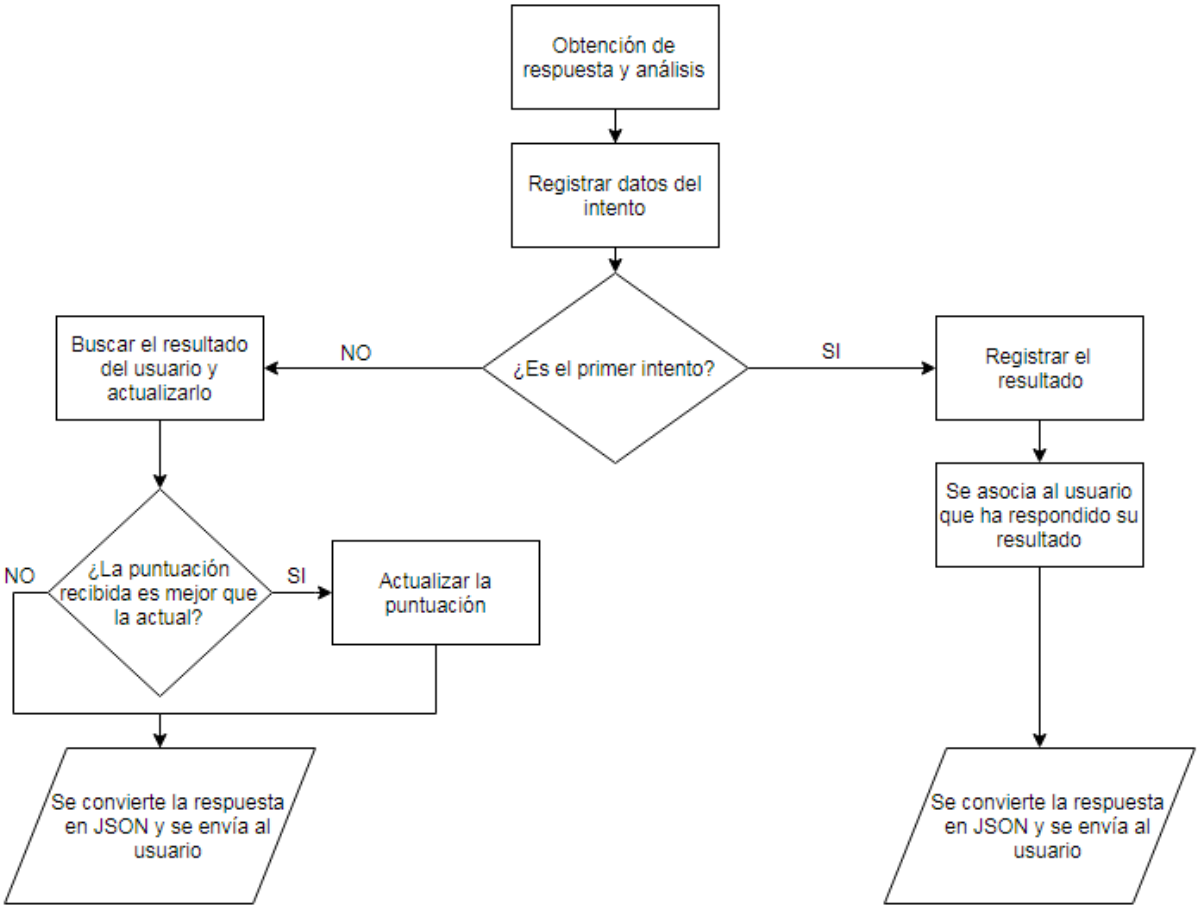
Para añadir un nuevo tipo de pregunta en el cuestionario se tiene que implementar después de los dos tipos ya creados y siendo identificados por su tipo.

Una vez enviada la información mediante AJAX a la ruta "/submission/{id.cuestionario}" es el servidor el encargado de analizar las respuestas del alumno.

Primero con la id de la ruta se busca el cuestionario en la base de datos y con la respuesta recibida mediante el método POST ya se pueden comparar las respuestas para comprobar cuáles son correctas y cuáles no. Se han de comparar según el tipo de pregunta, de modo que si se añaden nuevos tipos de preguntas se tendrá que implementar un apartado nuevo donde indica el código.

```
fqnr.questions.forEach(function(que,qindex){
    if(que.type == 'simple'){
        //console.log(req.body[que._id]+"·····"+que.answers[que.solution]);
        //if question is correct push true in array else push false
        if(answersUser[que._id]==que.answers[que.solution]){
            scoreResult.push(true);
        }else{
            scoreResult.push(false);
        }
    }else if(que.type == 'complex'){
        var aux = [];
        que.solution.forEach(function(com,index){
            if(req.body[que._id]){
                if(answersUser[que._id][index]==que.answers[com]){
                    aux.push(true);
                }else{
                    aux.push(false);
                }
            }else{
                aux.push(false);
            }
        });
        //if all element are true, return true
        scoreResult.push(aux.every(function(k){ return k === true;}));
    }else{
        //#######################################
        //for others type of questions
        //we must check this answers is correct, in this case push true in scoreResult or false in the others case.
        //#######################################
        console.log("others");
    }
    //console.log(scoreResult);
    //create object for return info to questionnaire.ejs
    infoBack[que._id] = scoreResult[qindex];
});
```

Seguidamente se registra el intento y el resultado, para explicar este proceso se utilizara el siguiente flujo grama.



En el flujo grama se observa el registro de puntuación, ésta es sobre diez y con dos decimales. También se debe destacar la conversión a JSON para devolver la información, para ello se utiliza el método “JSON.stringify()”.

Una vez que la llamada a AJAX devuelve los resultados, en el lado del cliente se convierte el objeto JSON a objeto de JavaScript con el método “JSON.parse()”, de tal modo que se pueda analizar y mostrar al alumno cuales son las repuestas correctas e incorrectas. Si todas las repuestas son correctas se mostrara un modal indicando dicha información.

Sistema de sesiones

Para que el sistema de sesiones funcione se necesitan dos paquetes npm, unos es Passport y el

otro es Express-session. El primero es para registrar e iniciar las sesiones de usuarios y con el segundo se mantiene la sesión del usuario mientras utiliza la plataforma. Así pues los primero es configurar Passport y Express-session en el documento app.js.

```
//================PASSPORT AND EXPRESS-SESSION SETTINGS====================
app.use(require("express-session")({
    secret: "Diseno e implementacion con Node.js de una aplicacion web para seguimiento y evaluacion del aprendizaje",
    resave: false,
    saveUninitialized: false
}));
app.use(passport.initialize());
app.use(passport.session());

passport.use(new LocalStrategy(User.authenticate()));
passport.serializeUser(User.serializeUser());
passport.deserializeUser(User.deserializeUser());
//==========================================================
```

En este documento también se ha implementado un pequeño código para que la sesión permanezca abierta independientemente de la ruta de la plataforma en la que este el usuario.

```
//maintaining a session on between routes
app.use(function(req, res, next){
   res.locals.currentUser = req.user;
   next();
});
```

De modo que la sesión se mantendrá en todos los puntos de la plataforma, el problema reside en que tenemos distintos tipos de usuarios que conviven en una misma plataforma, por tanto se han restringido las áreas en función de los privilegios de cada usuario. Esto es por motivos de seguridad además de asegurar el buen funcionamiento de la plataforma. Dado que hay tres tipos de usuarios, administradores, profesores y alumnos, se ha restringido el acceso a ciertas rutas mediante un middleware en la llamada a esas rutas, así pues cada vez que haya una llamada GET, POST, PUT o DELETE primero comprobara que usuario es y si tiene permisos para entrar, si los tiene entrara y realizara sus funciones y si nos los tiene será re direccionado a otra página anterior.

```
//show a form for the teachers add simple questions (only teachers)
router.get("/:id/addsimplequestions",middleware.isLoggedInTeacher,function(req, res) {
  User.findById(req.params.id, function(err,fteach){
      if(err){
          console.log(err);
      }else{
          Topic.find({},function(err,ftopic){
              if(err){
                  console.log(err);
              }else{
                  res.render("addsimplequestions",{teacher:fteach,topics:ftopic,message: req.flash('info')});
              }
          });
      }
    });
});
```

Como existen distintos tipos de usuarios con diferentes permisos se han creado varias funciones dentro del archivo middleware que comprobará los permisos de acceso a todas las páginas de la aplicación, de modo que restringirá el acceso dependiendo del tipo usuario que quiera entrar a una determinada página, un ejemplo de ellas:

```
//check session admin
middlewareObj.isLoggedInAdmin = function(req, res, next){
    if(req.isAuthenticated() && req.user.rol === "admin"){
        return next();
    }
    res.redirect("/");
};

//check session teacher
middlewareObj.isLoggedInTeacher = function(req, res, next){
    if(req.isAuthenticated() && req.user.rol === "teacher"){
        return next();
    }
    res.redirect("/");
};

//check session for teacher or admin
middlewareObj.isLoggedInDualTA = function(req, res, next){
    if(req.isAuthenticated() && ((req.user.rol === "teacher") || (req.user.rol === "admin"))){
        return next();
    }
    res.redirect("/");
};

//check session for student
middlewareObj.isLoggedInStudent = function(req, res, next){
    if(req.isAuthenticated() && req.user.rol === "student"){
        return next();
    }
    res.redirect("/");
};
```

Implementaciones en JS para formularios

Una particularidad que tienen los formularios que se utilizan para crear preguntas de distintos tipos o crear cuestionarios es que son asíncronos, esto es posible ya que están implementados en JavaScript. De modo que podemos añadir o eliminar elementos de los formularios de forma dinámica sin necesidad refrescar la página, consiguiendo así un formulario cómodo, intuitivo y sencillo para el usuario. Para que el documento sea utilizado por todos los formularios tiene que estar en la carpeta “public”, ya que todos los documentos de la plataforma pueden acceder a ella sin tener que importar nada.

Las funciones más comunes de este documento JavaScript son añadir o eliminar elementos del árbol DOM de manera asíncrona. Así pues añadimos o eliminamos respuestas, tags, temas y otros tipos de elementos, para esto se utiliza por lo general el evento “click”, el cual está asociado a un botón y ejecuta una secuencia de instrucciones.

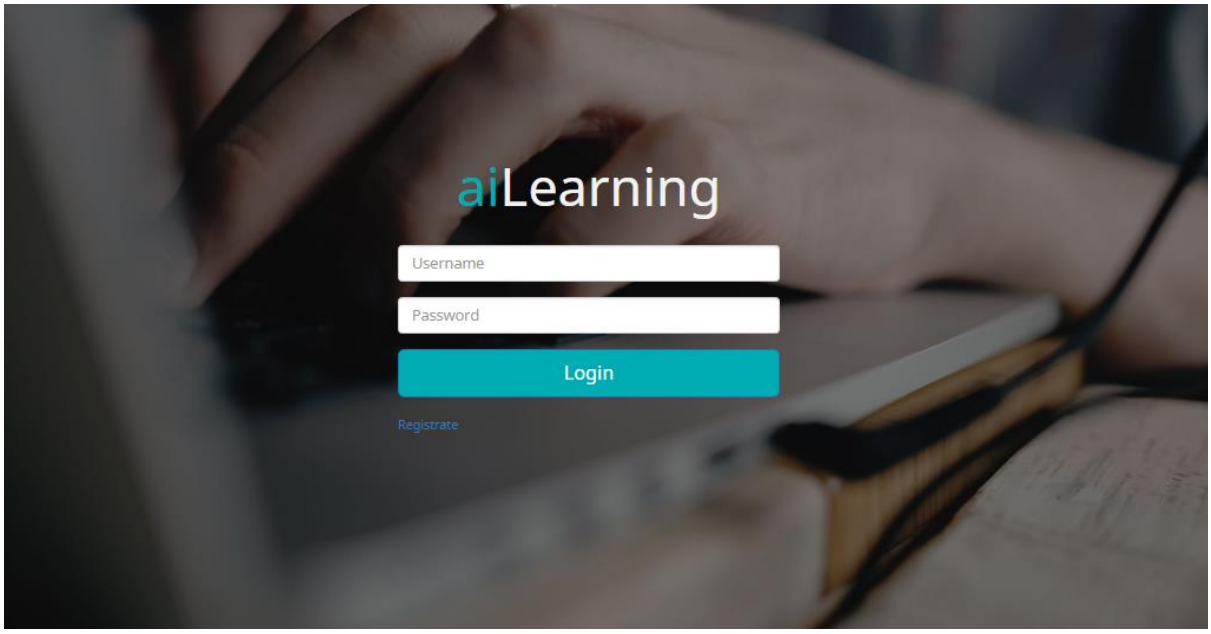
```
//select button to listener
var button = document.getElementById("add");
var delet = document.getElementById("delete");
var addtag = document.getElementById("addtag");
var deltag = document.getElementById("deltag");

//this event use for add "gaps answers", radio button to simple questions and checkbox to complex questions
button.addEventListener("click",function(){
    var sol = document.getElementById("solutions");
    var input = document.getElementsByName("solution[]")[0];
    var solution;
    if(input.type === 'radio'){
        solution = '<br><div class="col-md-11"><input type="text" class="form-control" name="answers[]" id="
    }else{
        solution = '<br><div class="col-md-11"><input type="text" class="form-control" name="answers[]" id="
    }
    var block = document.createElement('div');
    block.setAttribute("class","row");
    block.innerHTML =  solution;
    sol.appendChild(block);
});

//delete "gaps answer"
delet.addEventListener("click",function(){
    var sol = document.getElementById("solutions");
    if(document.getElementsByName("solution").length !== 2){
    sol.lastChild.remove();
    }
});
```

Diseño

La plataforma esta diseña en función del público objetivo, es decir, estudiantes. De modo que es un diseño actual, joven y atractivo, con colores vivos y elementos sencillos. También se ha diseñado páginas sencillas e intuitivas para todos los usuarios donde crear asignaturas y cuestionarios o añadir preguntas sea fácil sin tener que recurrir a guías de uso.



Los elementos de la plataforma son creados a través de la librería Bootstrap confiriendo uniformidad y fluidez al navegar por toda la plataforma. Una de las características principales

de esta librería es crear páginas “resposives” capaces de adaptarse a la mayoría de dispositivos electrónicos.

Por último indicar los colores y fuente utilizada en la plataforma:

- Fuente: Noto Sans.
- Colores (html): #222831, #393E46, #00ADB5 y #EEEEEE.

En futuras líneas de trabajo que requieran crear nuevas páginas se deberá utilizar esta tipografía y estos colores.

4. Conclusiones y líneas futuras

4.1 Conclusiones

Como resultado final se ha obtenido una plataforma abierta y totalmente escalable, todo esto con las tecnologías NodeJS, ExpressJS, MongoDB y JavaScript entre otras, además de usar la arquitectura REST. De modo que queda preparada para futuros desarrollos e implementaciones de docentes o desarrolladores que pretendan introducir nuevos métodos de evaluación o introducir mecanismos que permitan controlar qué preguntas se hacen a los estudiantes en función de sus respuestas.

La plataforma está preparada para que docentes creen asignaturas y preguntas de distintos tipos de forma que al agruparlas creen cuestionarios, los cuales van a poder ser contestados por los estudiantes cuando el docente lo permita. Posteriormente el docente obtendrá los resultados del estudiante pero no solo el resultado final sino cada uno de los intentos del estudiante y así tener más información para saber cuáles son los puntos conflictivos del cuestionario o temas de la asignatura. De modo que es una plataforma autónoma donde el docente y el estuante interactúan de forma directa, mientras que el administrador puede observar, modificar y limitar su uso en la plataforma.

También es una plataforma amigable y sencilla para administradores, docentes y alumnos, los cuales podrán utilizar la plataforma cuando quieran y desde donde quieran, ya que al uso de la librería Bootstrap se ha conseguido una plataforma adaptativa para todos los dispositivos electrónicos

4.2 Líneas futuras

Una de las futuras líneas de trabajo más obvias seria implementar mecanismos que evalúen el conocimiento de los estudiantes y seleccionen las preguntas más apropiadas. También nuevos tipos de preguntas, se podrá desarrollar cualquier tipo de preguntas siempre y cuando siga el estándar indicado a la hora de enviar las respuestas de los cuestionarios.

Otra línea futura seria traducir la plataforma a inglés y así poder llegar a muchos más docentes y estudiantes, creando así una comunidad mucho mayor.

5. Bibliografía

- HTML5:
    - https://es.wikipedia.org/wiki/HTML5

- CSS3:
    - https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada

- Bootstrap:
    - https://es.wikipedia.org/wiki/Bootstrap_(framework)
    - https://www.w3schools.com/bootstrap/default.asp
    - https://getbootstrap.com/docs/3.3/css/

- JavaScript:
    - https://developer.mozilla.org/es/docs/Web/JavaScript
    - https://es.wikipedia.org/wiki/JavaScript
    - https://www.w3schools.com/js/default.asp

- JQuery:
    - https://jquery.com/
    - https://es.wikipedia.org/wiki/JQuery

- NodeJS:
    - https://nodejs.org/es/

- http://chito.io/que-es-nodejs-para-que-sirve-ventajas-y-desventajas/
- https://es.wikipedia.org/wiki/Node.js

- MongoDB:
  - https://www.mongodb.com/
  - https://es.wikipedia.org/wiki/MongoDB
  - https://www.genbetadev.com/bases-de-datos/mongodb-que-es-como-funciona-y-cuando-podemos-usarlo-o-no

- ExpressJS:
  - https://en.wikipedia.org/wiki/Express.js
  - http://expressjs.com/

- REST:
  - http://www.restapitutorial.com/
  - https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional
  - https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos

- Paquetes NPM:
  - https://www.npmjs.com/
  - https://es.wikipedia.org/wiki/Npm

- Librería Lodash:
  - https://lodash.com/