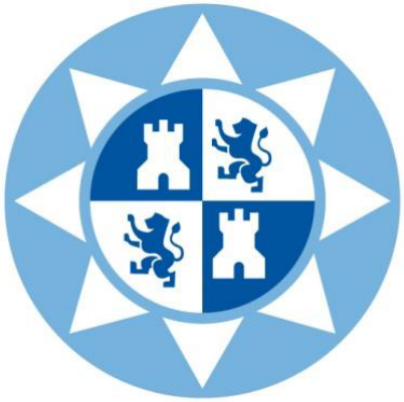
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA



Trabajo Fin de Grado

Desarrollo de una aplicación web con Django para evaluación automática de código Python

Autor: Juan Luis Navarro Rey

Director: Fernando Losilla López

Junio 2017

Autor	Juan Luis Navarro Rey
E-mail del autor	jl.navarro16@gmail.com
Director	Fernando Losilla López
E-mail del autor	fernando.losilla@upct.es
Codirector(es)	
Título del TFG	Desarrollo de una aplicación web con Django para evaluación automática de código Python
Descriptores	

Resumen:
En este TFE se ha desarrollado una aplicación web que permite a un profesor crear asignaciones en la que los alumnos deben enviar un código que es evaluado por un servidor. Se ha desarrollado un interfaz web que permite tanto a profesores como a alumnos subir, crear y cargar asignaciones, respectivamente. El interfaz permite la creación de asignaciones sencillas, generando código en el servidor para la comprobación de resultados. Asi mismo, se ha contemplado la posibilidad de que profesores con mayores conocimientos puedan crear asignaciones más complejas, modificando los archivos generados.

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

Agradecimientos

Agradezco a mi familia todo el apoyo que me ha dado y el esfuerzo realizado durante mi etapa como estudiante.

También agradezco a mi director Fernando Losilla, por haberme permitido tener la oportunidad de realizar este proyecto con él y principalmente por todo el tiempo y dedicación que ha invertido en mí y en que este proyecto saliese adelante.

Ya por último, agradezco a mis compañeros de clase los buenos ratos pasados durante la carrera.

Contenidos

Lista de figuras

Lista de tablas

1 Introducción

En este TFE se ha desarrollado una aplicación web que permite a un profesor crear asignaciones en la que los alumnos deben enviar un código que es evaluado por un servidor. Se ha desarrollado un interfaz web que permite tanto a profesores como a alumnos subir, crear y cargar asignaciones, respectivamente. El interfaz permite la creación de asignaciones sencillas, generando código en el servidor para la comprobación de resultados. Asi mismo, se ha contemplado la posibilidad de que profesores con mayores conocimientos puedan crear asignaciones más complejas, modificando los archivos generados.

El trabajo que se muestra en la memoria de este proyecto muestra el diseño y desarrollo de una aplicación web, haciendo uso de las tecnologías que están teniendo mejor acogida entre los desarrolladores web.

La aplicación consiste en un evaluador de código Python, dónde el profesor tendrá la potestad de crear ejercicios. La aplicación realizará la correspondiente plantilla para el alumno y el alumno se descargará la plantilla, la rellenará y la subirá al servidor para ser corregido por la aplicación web.

La aplicación web tendrá otras funcionalidades secundarias como: ver las notas de los ejercicios, ver la fecha limite de entrega del ejercicio, un estudio simple estadístico acerca de los ejercicios, etc.

1.1 Objetivos

Los objetivos de este trabajo son:

- Desarrollar una aplicación que permita la evaluación automática de código en Python.
- Desarrollo de un interfaz web que permita la gestión de perfiles de profesor y alumno.
- Posibilidad de crear asignaciones simples para los alumnos mediante el interfaz web y asignaciones más complejas mediante código.
- Desarrollo de una aplicación fácilmente ampliable en funcionalidad.

1.2 Metodología de trabajo

1.2.1 Scrum

Scrum [2] es una metodología ágil [3] basada en un proceso iterativo e incremental creada por Ikujiro Nonaka e Hirotaka en la década de 1980. En la metodología Scrum se definen

una serie de buenas prácticas, procesos y roles.

Antes de seguir hablando de Scrum es importante definir una palabra propia de la metodología que se usa constantemente. El Sprint.
Un Sprint es el periodo de tiempo entre la generación de un entregable y otro. Son periodos de corto tiempo de entre 2 y 3 semanas.

Entre las buenas prácticas y procesos que define Scrum, las reuniones son las más comunes. Unas de estas reuniones serían las reuniones diarias (DailyScrum). Son reuniones cortas de no más de 15 ó 20 minutos, en ellas el Scrum Master, pregunta a los miembros del equipo su trabajo realizado durante el día anterior, si han tenido algún problema y el trabajo a realizar en el día actual.

Al final del Sprint el equipo se vuelve a reunir, pero esta vez con el ProductOwner, para revisar el Sprint anterior y validarlo con el cliente. También se reúnen internamente para planificar el próximo Sprint.

Los roles en los que se agrupan los integrantes de un proyecto Scrum son los siguientes:

- ProductOwner: Supone la figura del cliente, define las historias de usuario (requisitos) y se asegura que el proyecto se realice de forma correcta, desde el punto de vista comercial.
- Scrum Master: Es el encargado de hacer que el equipo cumpla con las reglas que propone Scrum para la realización del Sprint, a diferencia de otras metodologías este "jefe de proyectos" no se encarga de gestionar al equipo de trabajo ya que estos se auto gestionan, el Scrum Master se limita a ofrecer una protección al equipo de trabajo ante una posible distracción.
- Equipo de desarrollo: Son los encargados de entregar el producto a tiempo al final del Sprint, lo forman equipos pequeños de no más de 10 personas, es importante que los integrantes del equipo tengan las competencias transversales necesarias para trabajar en equipo.

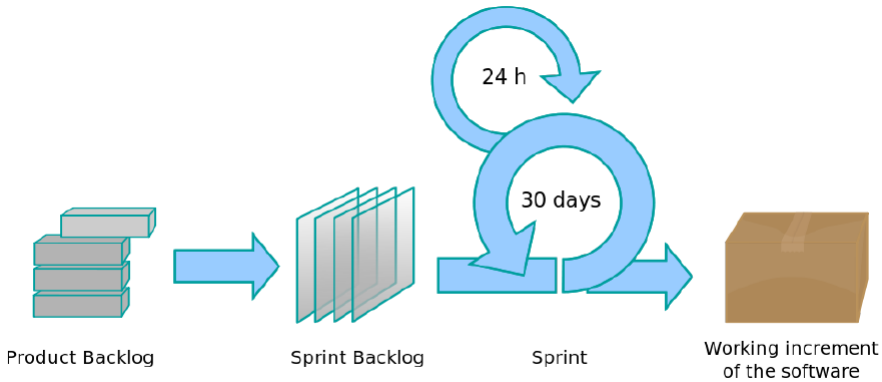


Figura 1.1: *Proceso Scrum*

En el desarrollo de la aplicación se ha elegido la metodología Scrum debido a que ayuda a definir los procesos del desarrollo y poder aprovechar todas las ventajas que ofrece una metodología ágil frente a una metodología clásica. El principal escollo a la hora de usar una metodología ágil es que el cliente debe de estar presente en el desarrollo del sistema

llegando a ser tratado como parte del equipo de desarrollo.

En este trabajo el rol ProductOwner lo ha tenido el director del trabajo simulando las exigencias del cliente.

1.2.2 Subversion

Subversión [4] es una herramienta de control de versiones basada en un repositorio (localizado en el servidor) que se asemeja a un sistema de ficheros. Utiliza el concepto de revisión para guardar los cambios producidos en el repositorio. Entre dos revisiones sólo guarda el conjunto de modificaciones, optimizando así al máximo el uso de espacio en disco.

Estructura [4]

Generalmente para cada proyecto se genera la siguiente estructura:

- Trunk: Rama de desarrollo principal.
- Tags: Rama de gestión de versiones. Reservado para versiones cerradas, por tanto no se desarrollará sobre esta rama.
- Branches: Rama con evoluciones paralelas al Trunk.

Implementación de Subversion en el proyecto

Subversion ha sido usado durante el desarrollo de la aplicación web. Aunque normalmente se suele crear una rama por cada tarea, en el proyecto se ha creado una única rama en el directorio Branches.
En Tags, se han ido guardando los distintas versiones que se han creado y se verá en secciones posteriores de este capítulo.
En Trunk, se ponía la última versión estable de la aplicación web.

1.3 Funcionalidad de la aplicación

A continuación se explica muy brevemente (se explicará con más detalle en el capítulo 3) la funcionalidad por roles de cada tarea. Como se puede observar en la imagen 1.2 las principales tareas desarrolladas en este trabajo son:

- SignUp: Registrarse en la aplicación.
- LogIn: Entrar en la aplicación.
- Editor: Un editor simple de código Python.
- Mis ejercicios: Ejercicios asociados a un alumno.
- Estadísticas: Estadísticas de los ejercicios.
- Nuevo ejercicio: Se crea un nuevo ejercicio.
- Ejercicios mandados: Se obtiene una lista con todos los ejercicios mandados.
- Ajustes BBDD: Gestión de la base de datos.

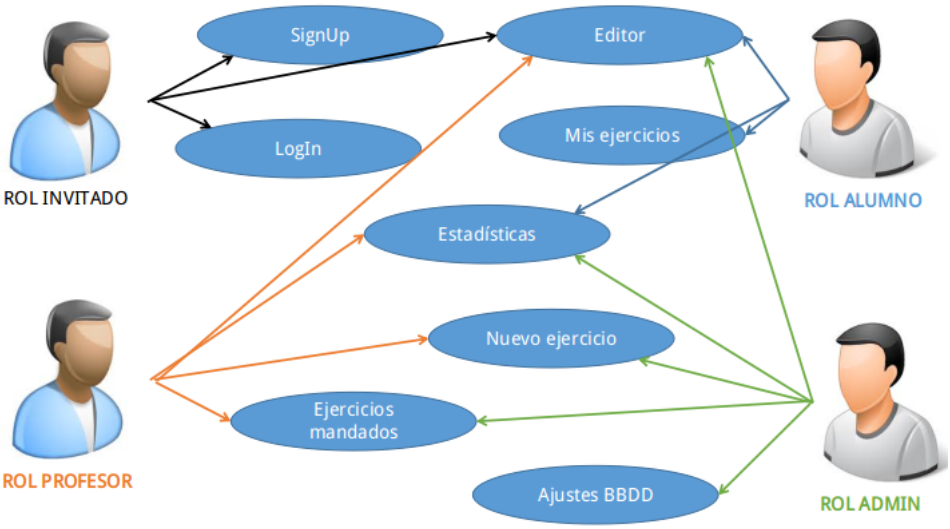


Figura 1.2: *Funcionalidad por roles*

TAREA	INVITADO	ALUMNO	PROFESOR	ADMIN
SignUp	x			
LogIn	x			
Editor	x	x	x	x
Mis Ejercicios		x		
Estadísticas		x	x	x
Nuevo ejercicio			x	x
Ejercicios mandados			x	x
Ajustes BBDD				x

Tabla 1.1: *Asociación de roles a cada tarea*

1.4 Evolución del proyecto

Como se ha dicho en secciones anteriores, se ha usado en el proyecto un control de versiones. En la tabla 1.2 se detalla la versión y las modificaciones correspondientes a dicha versión. En esta tabla se puede observar la evolución funcional del proyecto.

A continuación, en la figura 1.3, se puede observar la evolución temporal del proyecto. Esta evolución se ha medido mediante el diagrama de Gantt.

Diagrama de Gantt

El diagrama de Gantt es una herramienta gráfica cuyo objetivo es medir la evolución temporal del proyecto asociándole una fecha de inicio y una fecha de fin a cada tarea a lo largo de un período temporal.

Cuando dos tareas están relacionadas mediante una flecha en el diagrama, significa que la segunda tarea no puede empezar hasta que concluya la primera.

Versión	Modificaciones
1.1.0 (versión inicial)	Se implementa: El editor. Ver detalles del ejercicio. Ver lista de ejercicios mandados. Ver ejercicios del alumno. Crear nuevo ejercicio. Ver las notas del ejercicio.
1.2.0	Se implementa el registro en la web.
1.2.1	Se quitan los CDNs externos salvo los modales de documentación que sí tiene los CDNs.
1.2.2	La base de datos ya no apunta a mlab.com sino a localhost, de esta manera si no hay internet la aplicación sigue funcionando.
1.2.3	Se implementa el registro cifrado y se corrige el bug del login (se implementa el login cifrado).
1.2.4	Se cambia la ruta del fichero subido desde el editor ace.
1.2.5	El administrador tiene la potestad de cambiar los roles.
1.3.0	Se añade las estadísticas: Por ejercicio: Se calcula el número de personas que han aprobado y el número de personas que han suspendido el ejercicio. Se calcula por nota. Se calcula cuántos tienen un 5, un 6, un 7, etc Por alumnos: Se calcula el número de ejercicios que ha aprobado y el número de ejercicios que ha suspendido
1.4.0	Se amplia la opción de probar una función varias veces. En el campo se pone una lista que cada posición de la lista irá separada por el caracter de escape ';'.
1.4.1	Se corrige el bug del paginador de la tabla que pertenece al fichero misEjercicios.html. Se actualiza la versión del framework AngularJS a la versión 1.5.6. Se corrige el bug de la corrección del ejercicio. Cuando hay un error de compilación salta una excepción y el ejercicio en base de datos se queda como no corregido en vez de poner un 0.
1.5.0	Añadir los 'Sin corregir' al gráfico de 'Aprobados/suspensos' tanto de las estadísticas del ejercicio como de las estadísticas del alumno.
1.5.1 (versión final)	Al subir el fichero desde el editor, se asocia el fichero al alumno. Se corrige el bug que no comprobaba la obligatoriedad de poner el valor de retorno de una función.

Tabla 1.2: *Detalle de cada versión de la aplicación web*

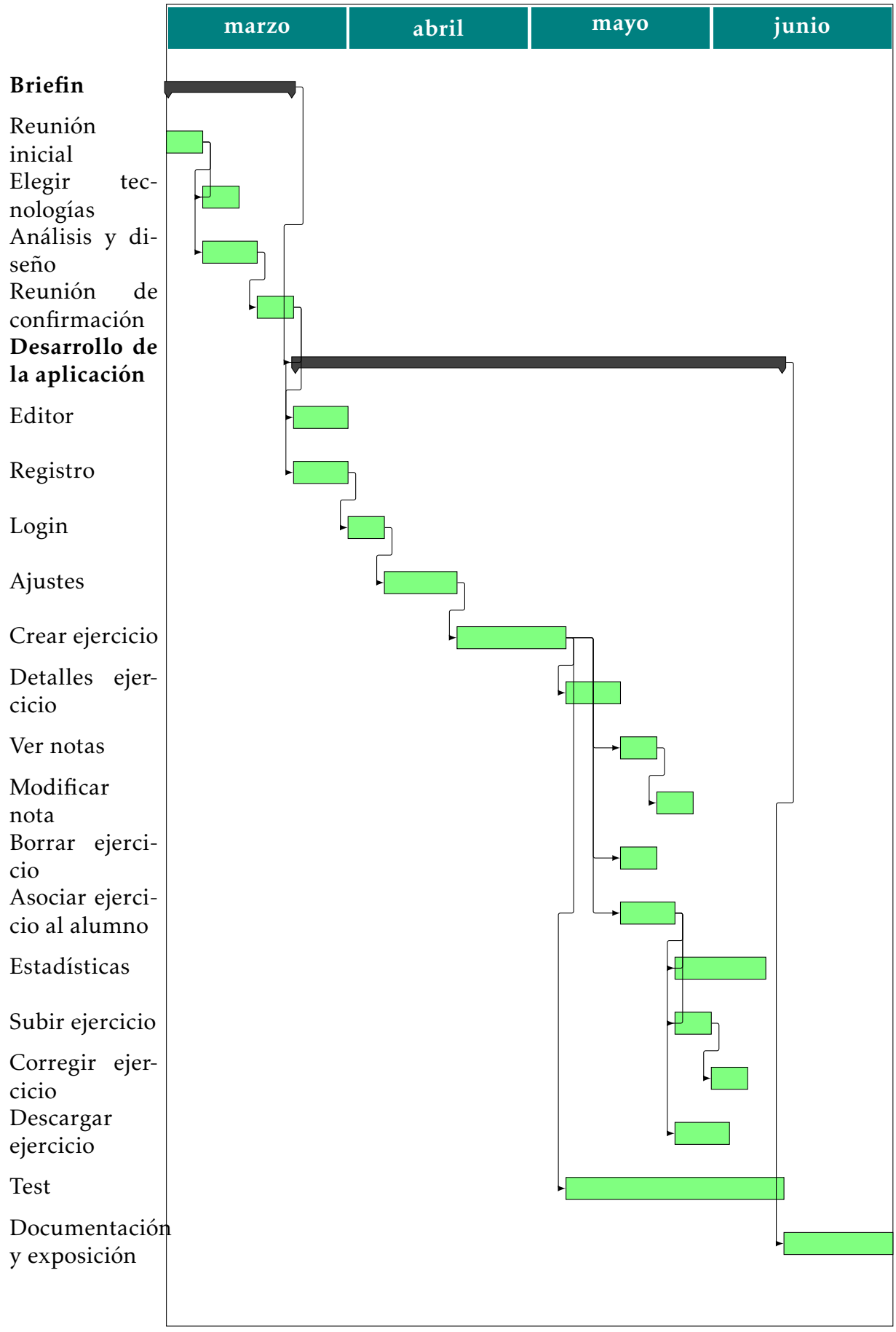


Figura 1.3: *Diagrama de Gantt*

2 Tecnologías

2.1 Introducción

(a) (b) (c) (d) (e)

Figura 2.1: *Principales tecnologías*

Como se puede observar en la figura 2.1, las principales tecnologías que se han usado como base en este proyecto son: HTML5, CSS3, javascript, jQuery y python.
A continuación, se procederá explicar brevemente cada tecnología.

2.1.1 HTML5

HTML5 [5] es un lenguaje de marcas interpretado por el navegador usado para estructurar y presentar el contenido para la web. Se trata de un sistema para formatear el layout de nuestras páginas, así como hacer algunos ajustes a su aspecto.

HTML5 introduce nuevas etiquetas multimedia para incluir dichos elementos de tal manera que permite al programador reducir el número de plug-ins a usar en la aplicación web. Entre estos plug-ins se encuentra el Adobe Flash, que con las nuevas etiquetas HTML5 permite no tener que usarlo.

HTML5 también permitió avanzar en las aplicaciones móviles ya que permite acceder a sitios web de manera offline implementada mediante una sección llamada “Local storage”. “Local storage” es un pequeño almacén de datos que se guardará en el navegador. También contiene una sección “Session storage” que es muy similar al “Local storage”. La diferencia entre ambos es que los datos en “Local storage” no se borran automáticamente mientras que en “Session storage” se borran al cerrar el navegador. Por último, implementa una base de datos no relacional llamada “IndexedDB”.

En la tabla 2.1 se pueden ver las nuevas etiquetas introducidas en HTML5.

2.1.2 CSS3

Hojas de estilo en cascada (CSS) [6] es un lenguaje para el diseño gráfico de una aplicación web. Aunque se puede usar en diversos tipos de ficheros, los ficheros más frecuentes son los

Etiqueta	Descripción
article	Esta etiqueta sirve para definir un artículo, un comentario de usuario o una publicación independiente dentro del sitio.
header, footer	Estas etiquetas individuales ahorran tener que insertar IDs para cada uno, como se solía hacer anteriormente. Además, se pueden insertar headers y footers para cada sección, en lugar de tener que hacerlo únicamente en general.
nav	Etiqueta que define una lista de navegación.
section	Etiqueta para ordenar el código en secciones. Crea una sección.
audio y video	Permiten acceder de forma más simple a contenido multimedia.
embed	Permite marcar la presencia de un contenido interactivo o aplicación externa.
canvas	Crea un lienzo para dibujar líneas o formas o para insertar imágenes. Simplifica la implementación de los juegos HTML.

Tabla 2.1: *Etiquetas introducidas en HTML5*

ficheros de tipo HTML o XML.

CSS trata de dar flexibilidad a la aplicación web separando el código HTML del diseño gráfico en ficheros diferentes. Esta separación del permite presentar el mismo documento con diferentes diseños para diferentes métodos de renderizado, como en pantalla, en tabletas, en móviles, etc.

La especificación CSS describe un esquema prioritario para determinar qué reglas de estilo se aplican si más de una regla coincide para un elemento en particular. En caso de repetir una regla, el navegador se queda con la última definición.

Novedades

- Varias imágenes de fondo: Cada contenedor tiene su fondo. En la versión anterior, esto no era así y para hacerlo se necesita superponer las imágenes.
- Esquinas redondeadas.
- Bordes con imágenes: En esta versión se permite poner una imagen o repetición de imágenes de fondo en un borde.
- Sombras: Se permite poner sombras tanto a los elementos como a los textos.
- Texto en varias columnas.
- Instalación de fuentes: Se permite instalar otras fuentes ya estén en el ordenador o en un servidor en internet.
- Animaciones: Se permite transiciones y transformaciones [7].

2.1.3 JavaScript

JavaScript (JS) [8] es un lenguaje de programación interpretado. El uso más común de javascript es en el lado del cliente para realizar páginas webs dinámicas. Aunque hoy en

día, también se puede usar en el lado del servidor usando NodeJS[1]. Actualmente, todos los navegadores tienen un intérprete de javascript.

Las principales funciones de javascript para realizar una web dinámica son:

- Dar mensajes de alerta.
- Mostrar u ocultar elementos.
- Cambiar colores, estilos, etc. de los elementos.
- Comprobar los datos de un formulario antes de enviarlo.

Características [9]

- Dinámico
  - Tipado dinámico: En la declaración de la variable no se especifica el tipo de dicha variable. El intérprete es capaz de saber el tipo de variable según su contenido. Por tanto si se hace `var x = "str"`, el intérprete le dará a la variable un tipo String, si por el contario se hace `var x = 2`, el intérprete le dará a la variable el tipo number.
  - Objetual: Está formado casi en su totalidad por objetos, aunque éstos se definan como funciones en la mayoría de los casos. Los objetos son arrays asociativos, es decir, si tenemos un objeto persona cuyo nombre es Juan, se accederá al atributo nombre de las siguientes maneras: `persona["nombre"]` o `persona.nombre`.
- Funcional: Las funciones son objetos en sí mismos. Las funciones tienen propiedades y métodos. Las funciones pueden tener otras funciones incorparadas. Estas funciones anidadas se crean cuando se crea el objeto. Cada función interna recibe el nombre de closure.

2.1.4 jQuery

jQuery [11] es una biblioteca multiplataforma de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

Características

- Selección de elementos DOM.
- Interactividad y modificaciones del árbol DOM, incluyendo soporte para CSS3.
- Efectos y animaciones.
- AJAX.

La gran ventaja [10] de usar esta librería es que no importa el navegador que se esté utilizando, ya que es la propia librería la encargada de saber el navegador que se está usando para aplicar las correspondientes líneas de código para dicho navegador.

[1] Esta tecnología no se usa en este proyecto. Más información en https://nodejs.org/es/

2.1.5 AJAX

AJAX (Asynchronous JavaScript And XML) [12], es una técnica de desarrollo web para crear aplicaciones dinámicas. Estas aplicaciones se ejecutan en el cliente manteniendo una comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas.

Existen varias librerías JavaScript que implementan AJAX y simplifican su funcionalidad. En este trabajo se ha usado mediante la librería jQuery. Esta librería se usa mucho hoy en día porque simplifica la realización del código. No sólo permite una comunicación asíncrona (implementada mediante promesas que tiene JavaScript[2]) sino que también simplifica la realización de la petición (ya se haga mediante GET o POST) así como la recepción de los datos.

2.1.6 Python

Python [13] es un lenguaje de programación interpretado cuya filosofía refuerza una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos y programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Características

- Lenguaje de programación de alto nivel. Cada fichero python se considera un script.
- Diseñado para ser fácil de leer y simple de implementar.
- Es multiplataforma.
- Es multiparadigma (orientación a objetos y programación funcional).
- Tipado dinámico: En la declaración de la variable no se especifica el tipo de dicha variable. El intérprete es capaz de saber el tipo de variable según su contenido. Por tanto si se hace `x = "str"`, el intérprete le dará a la variable un tipo string, si por el contario se hace `x = 2`, el intérprete le dará a la variable el tipo number.

2.2 Bootstrap

Twitter Bootstrap [14] o Bootstrap es un framework o conjunto de herramientas de código abierto para diseño gráfico de aplicaciones web. Contiene plantillas CSS de diseño con tipografía, formularios, botones, etc. Este framework necesita de un fichero javaScript para dar la funcionalidad a los elementos más complejos de HTML como pueden ser los paneles o los modales.

[2] ¿Qué es una promesa?, https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Promise

Figura 2.2: Logo Bootstrap

2.2.1 Características

Las principales características [15] son:

- Responsive design: Permite adaptar el diseño de la aplicación al tamaño de la pantalla.
- Es totalmente compatible con otros frameworks como jQuery y AngularJS.
- Dispone de distintos layouts.
- Es compatible con todos los navegadores.

2.2.2 Componentes webs

Los principales componentes [15] son:

- Botones: Permite crear botones usando la clase btn. Esta clase se adapta perfectamente a las etiquetas `<a>`, `<input>` y `<button>`.
- Dropdown: Menús desplegables agrupando varias opciones.
- Formularios: Contiene diversos layouts para los formularios. Existen los formularios en línea (inline).

2.3 Django

Figura 2.3: Logo django

Django [16] es un framework[3] de desarrollo web de código abierto (escrito en Python) que implementa el patrón de Modelo-Vista-Controlador (MVC). Django [17] fomenta un desarrollo rápido, un diseño limpio y pragmático.

[3] Un framework es un conjunto de librerías para simplificar la programación.

2.3.1 Características

Las principales características son:

- Escrito en Python: es capaz de heredar todas las características y facilidades del lenguaje, permitiendo desarrollar aplicaciones rápidas y potentes.
- DRY (Don't Repeat Yourself): filosofía de no repetir código y reutilizarlo.
- Admin: Viene activado por defecto las funcionalidades típicas del administrador.

2.3.2 Estructura

Como se puede observar en la figura 2.4, la estructura de un proyecto está compuesto por: [19]

- mysite: Es un contenedor para el proyecto. El nombre es irrelevante.
  - manage.py: Utilidad de línea de comandos que le permite interactuar con este proyecto de Django de varias maneras.
  - mysite: El directorio interno de mysite es el paquete real de python para el proyecto.
    - __init__.py: Un archivo vacío que indica a python que este directorio debe considerarse un paquete de python.
    - settings.py: Configuración para este proyecto django.
    - urls.py: Las declaraciones de URL para este proyecto de django.
    - wgsi.py: Punto de entrada para servidores web compatibles con WSGI para servir a su proyecto.

```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        wsgi.py
```

Figura 2.4: *Estructura de un proyecto django*

2.3.3 Patrón Modelo Vista Controlador

El patrón de diseño MVC [1] separa la lógica y los datos de una aplicación a partir de la presentación y la interacción con la interfaz de usuario, manteniendo un bajo acoplamiento entre los componentes, por lo que la aplicación queda dividida en tres capas.

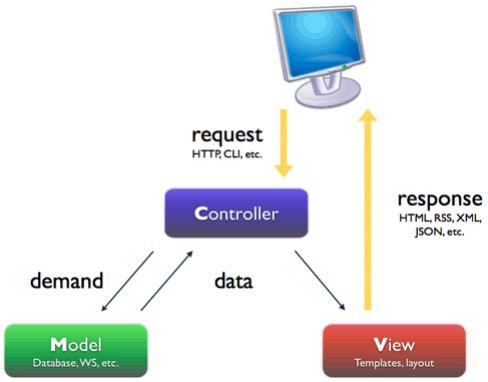


Figura 2.5: *Patrón MVC*

- Modelo: Contiene la capa de datos y la lógica de dominio específico de la aplicación, no tiene ninguna dependencia, o conocimiento, de la capa de interfaz de usuario.

- Vista: Es la capa que muestra el estado actual de los objetos de dominio de la aplicación y proporciona la interacción del sistema con el exterior.

- Controlador: Contiene los mecanismos que gestionan los cambios sobre el modelo y selecciona la vista correspondiente para cada acción.

2.4 AngularJS

Figura 2.6: *Logo AngularJS*

AngularJS [20] es un framework de JavaScript de código abierto creado por Google que se utiliza para crear y mantener aplicaciones web de una sola página. Es un framework que usa el patrón MVC.

AngularJS es un framework que genera aplicaciones web de página única. Es decir, en AngularJS sólo se crea un único HTML y es el propio framework quién modifica el árbol DOM para pintar las vistas. Aunque las vistas si pueden estar separadas en varios ficheros HTML.

2.4.1 Componentes básicos

Scopes [20]
Los scopes son los distintos ámbitos[4] sobre los que trabajan las expresiones de AngularJS, por ejemplo, cuando referenciamos un atributo del modelo mediante la directiva ng-model. En los scopes se guarda la información de los modelos que se representan en la vista y también atributos que se utilizan para manejar la lógica de la misma.

Controladores [22]
Los controladores son los encargados de inicializar y modificar la información que contienen los scopes en función de las necesidades de la aplicación. También podemos declarar funciones en el scope que se podrán utilizar más tarde o ser llamadas desde la vista.

Filtros [20]
Los filtros nos permiten modificar el modo en el que se va a presentar la información al usuario. La sintaxis de un filtro es la siguiente: `{{expresion | filtro}}`

Servicios [20]
Los servicios son los encargados de comunicarse con el servidor para enviar y obtener información que después será tratada por los controladores para mostrarla en las vistas.

Factorías [23]
Las factorías son un tipo de servicios que contiene angularJS y son declaradas mediante funciones. La gran ventaja de las factorías es que al usar el patrón singleton[5] sólo se obtiene una única instancia de dicho objeto cuyo ámbito es global (es toda la aplicación angularJS).

```
var app = angular.module('application',[]);
app.factory('nombre', function() {
   var datos = {
      nombre : "",
      apellidos : ""
   }

   return {
      getDatos : function(){
         return datos;
      },
      setDatos : function(nuevosDatos){
         datos = nuevosDatos;
      }
   };
});
```

[4] Tiempo de vida por el cual la variable retiene su valor. Al salirse el programa del ámbito de la variable, ésta pierde su existencia o su valor.
[5] Es un patrón que permite instanciar una única vez un elemento.

2.4.2 Principales directivas

En la tabla 2.2 se explican las principales directivas [24] que tiene el framework angularJS.

Directiva	Descripción
ng-app	Directiva obligatoria que define una aplicación angularJS
ng-model	Directiva que crea una variable como modelo dentro del patrón MVC
ng-controller	Directiva que define un controlador
ng-click	Directiva que captura el evento al hacer un click de ratón y llama a la función asociada
ng-include	Directiva que se usa para incluir el contenido de un fichero HTML dentro del fichero HTML en el que se pone dicha directiva
ng-repeat	Directiva que sirve para crear código repetido según se recorre una lista
ng-show	Directiva que sirve para mostar un elemento del DOM
ng-hide	Directiva que sirve para ocultar un elemento del DOM
ng-if	Directiva que sirve para ejecutar código si se cumple la condición

Tabla 2.2: *Principales directivas de angularJS*

2.5 MongoDB

Figura 2.7: *Logo MongoDB*

MongoDB [25] es una base de datos no relacional (NoSQL). Es una base de datos ideal para aplicar a las aplicaciones del BigData. MongoDB al ser no relacional, la estructura de mongoDB es totalmente distinta a las bases de datos SQL.
Una base de datos mongoDB está compuesta por colecciones y documentos:

- Colecciones: Es una lista que contiene uno o varios documentos.
- Documentos: Un documento tiene la forma de un JSON. Un documento contiene la información a guardar en la base de datos.

MongoDB permite crear documentos de diferentes formas y tamaños, es decir, los documentos de una misma colección pueden tener atributos distintos.

2.5.1 Manejo de la base de datos

A continuación se pone un ejemplo de cómo se crea una base datos, una colección, un documento y como se inserta, se recupera y se modifica la información de la base de datos. Para crear una base de datos:

```
use nombreBBDD
```

Para crear una colección se usa el siguiente comando:

```
crear una colección
db.createCollection("nombreColeccion")
```

Para añadir un documento a la colección se usa:

```
db.nombreColeccion.insert({
   "x" : 5,
   "y" : 10
})
```

Recuperación de todos los elementos de una colección:

```
db.nombreColeccion.find()
Ejemplo: recuperar todos los elementos de una colección cuyo campo x sea 5
db.nombreColeccion.find({"x":5})
```

Para borrar un elemento se usa el siguiente comando:

```
db.nombreColeccion.remove()
Ejemplo: borrar todos los elementos de una colección cuyo campo x sea 5
db.nombreColeccion.remove({"x":5})
```

Para actualizar un documento se utiliza el siguiente comando:

```
db.nombreColeccion.update(filtro,{$set:documento_modificado})
Ejemplo: actualizar todos los elementos de una colección cuyo campo x sea 5
 ↪  por el campo x=3
db.nombreColeccion.update({"x":5},{$set:{"x":3}})
```

2.6 Google Charts

Google Charts [31] es una librería diseñada por Google para realizar gráficos. Esta librería se ha usado para realizar los gráficos estadísticos de la aplicación web.

Principales tipos de gráficos:

- Gráfico de línea
- Gráfico circular
- Gráfico de barras
- Diagrama de Venn

Figura 2.8: *Logo Google Charts*

- Gráfico de puntos
- Mapa
- Gráfico de radar

2.7 JQuery validation

Figura 2.9: *Logo jQuery validation*

JQuery validation es una librería de javaScript que requiere tener instalado la librería jQuery cuya finalidad es validar los datos de un formulario.

Las principales validaciones [32] son:

- Required: Verifica que se ha completado el campo del formulario.
- Minlength: Verifica que el texto de dicho campo tenga al menos un mínimo de caracteres.
- Maxlength: Verifica que el texto de dicho campo no tenga más de un máximo de caracteres.
- Min: Verifica que el campo numérico no sea inferior de lo desado.
- Max: Verifica que el campo numérico no sea superior de lo desado.
- Email: Verifica que el campo email esté bien formado.
- Number: Verifica que el campo sea de tipo numérico.
- Regex: Regex viene de Regular Expresion. Permite poner una expresión regular para comprobar que el dato introducido tiene un formato correcto.

En caso de incumplir las condiciones, la librería añade un mensaje de error junto al campo que incumple la condición.

2.8 JQuery confirm

JQuery confirm [33] es una librería de javaScript que requiere tener instalado la librería jQuery y el framework Bootstrap, cuya finalidad es programar alertas, confirmaciones, animaciones, etc de manera sencilla para el desarrollador.

Figura 2.10: *Logo jQuery confirm*

```
$.confirm({
   title: 'Titulo de la alerta',
   content: 'Mensaje de la alerta',
   buttons: {
      aceptar:{
         btnClass: 'btn-success',
         action : function(){
            console.log("Se ha confirmado la acción");
         }
      },
      cancelar:{
         btnClass: 'btn-danger',
         action : function(){
            console.log("Se ha rechazado la acción");
         }
      }
   }
});
```

3 Desarrollo del cliente

3.1 Introducción

En este capítulo se va a explicar los distintos roles existentes. También se explicará las funcionalidades de cada rol. A continuación se explicará la estructura de directorios del cliente y se finalizará el capítulo explicando las vistas de la aplicación.

3.2 Funcionalidad de los roles

Como se puede observar en la figura 3.1, se han creado varios roles. Estos roles se han creado para limitar al usuario el acceso a ciertas funciones. Es importante que un alumno no pueda modificar su propia nota. De ahí que haya sido necesario catalogar al usuario en alumno, profesor o administrador.

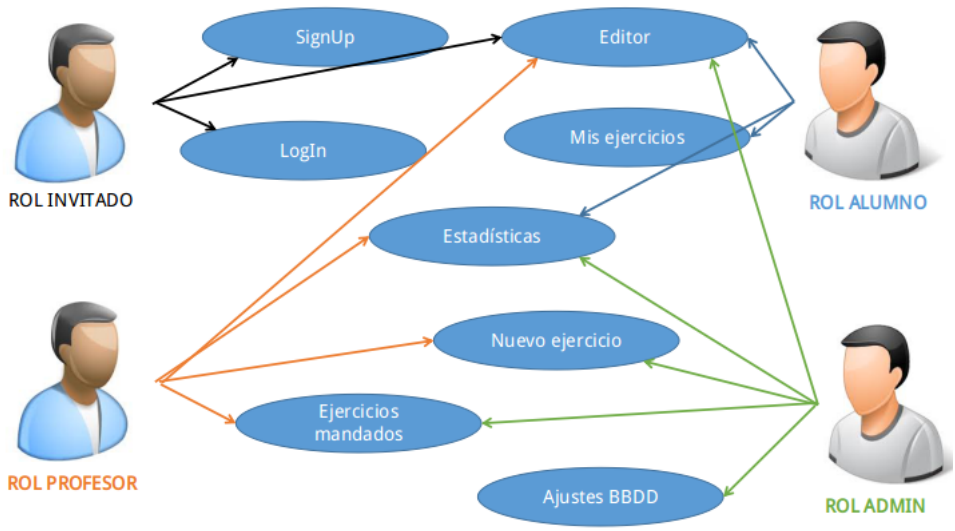


Figura 3.1: *Explicación de los roles*

- Rol invitado: Es la forma de catalogar a un usuario que aún no ha entrado en la aplicación. Al no estar dentro, no se le debe permitir acceder a los ejercicios, crear ejercicios, modificar la base de datos, etc.

- Rol alumno: Se identifica al usuario como un alumno, por lo tanto no debería de tener acceso a crear ejercicios, modificar la nota de los ejercicios, modificar la base de datos, etc. A este rol sí se le permitirá acceder a los ejercicios asociados a él. Podrá descargar la plantilla del ejercicio, podrá subir el ejercicio resuelto y corregirlo. También tendrá acceso a las estadísticas de un ejercicio determinado obteniendo los datos de dicho ejercicio de forma anónima y a las estadísticas de sus propios ejercicios.

- Rol profesor: Se identifica al usuario como un profesor, por lo tanto el profesor no debe de tener acceso a la base de datos aunque sí debe de tener la potestad de poder modificar la nota de un ejercicio. Como es lógico, el profesor no tendrá ejercicios asociados a él y por tanto no debe tener acceso a esa funcionalidad. Sin embargo si puede crear ejercicios y las plantillas. También tendrá acceso a las estadísticas de cada ejercicio, obteniendo los datos de forma anónima y acceso a las estadísticas de cada alumno. También tendrá acceso a los datos de cada ejercicio siendo capaz de ver las notas de cada ejercicio de todos los alumnos.
- Rol administrador: Tendrá acceso a todas las funciones de la aplicación web salvo a la función de ver los ejercicios asociados a él mismo, ya que el administrador será un profesor. Al administrador se le permite acceder a la base de datos, pudiendo modificar a los usuarios registrados en la misma. Esta funcionalidad es la diferencia entre el rol administrador y el rol profesor.

3.3 Estructura

En la figura 3.2 se muestra la estructura del proyecto, pero en esta sección se va a explicar la estructura que le corresponde al cliente. A continuación se explica el contenido de la

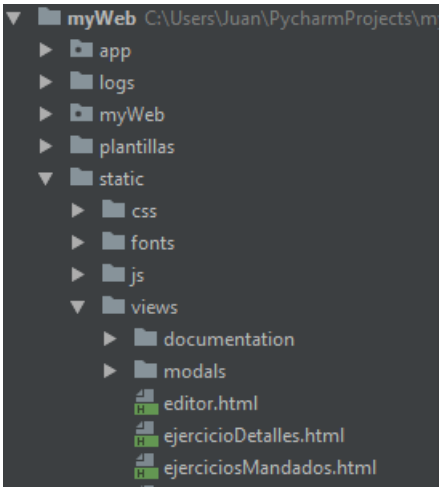


Figura 3.2: *Estructura del cliente*

carpeta static:

- css: Es un directorio que contiene todas las hojas de estilo que se han usado.
- fonts: Este directorio contiene las fuentes que se han usado para los iconos de las aplicación.
- js: Contiene otros directorios que son las librerías usadas en la aplicación. También contiene los ficheros JavaScript propios.
  - ace: Librería que permite realizar el editor. Más adelante se explicará dicha vista.
  - bootstrap: Contiene los ficheros JavaScript necesarios para instalar el framework.
  - bower_components: Contiene el fichero JavaScript para instalar el framework AngularJS y todos los módulos de AngularJS usados.

- controllers: Contiene los controladores de AngularJS que se encargarán de dar dinamismo a la aplicación.
- google_charts: Contiene los ficheros JavaScript para poder instalar esta librería de Google.
- jquery: Contiene los ficheros necesarios para poder instalar jQuery, jQuery validation y jQuery confirm.
- modelos: Contiene parte de los modelos usados en AngularJS.

■ views: Contiene las vistas de la aplicación.

- documentation: Ficheros HTML que servirán de ayuda al usuario.
- modals: Contiene los modales de la aplicación web.

3.4 Explicación de las vistas

3.4.1 Código común

Barra de navegación

Como se puede observar en la figura 3.3, la barra de navegación es distinta según el rol del

(a) Rol invitado

(b) Rol alumno, rol profesor

(c) Rol admin

Figura 3.3: *Barra de navegación*

usuario.

Para el rol invitado se muestra el enlace al editor, el acceso a la aplicación mediante el botón entrar y la posiblilidad de registrarse en el caso de no estar registrado ya.

La barra de navegación es muy parecida para los roles alumno y profesor. Ambos roles tienen el acceso al editor, a las estadísticas y a las propiedades de los ejercicios.
La diferencia está en las propiedades de los ejercicios. Para el alumno el desplegable tendrá un enlace para acceder a los ejercicios asociados al alumno, mientras que el desplegable para el rol profesor contendrá dos funciones. Por un lado, se podrá acceder a crear un nuevo ejercicio y por otro lado, el profesor podrá ver todos los ejercicios que ha mandado.

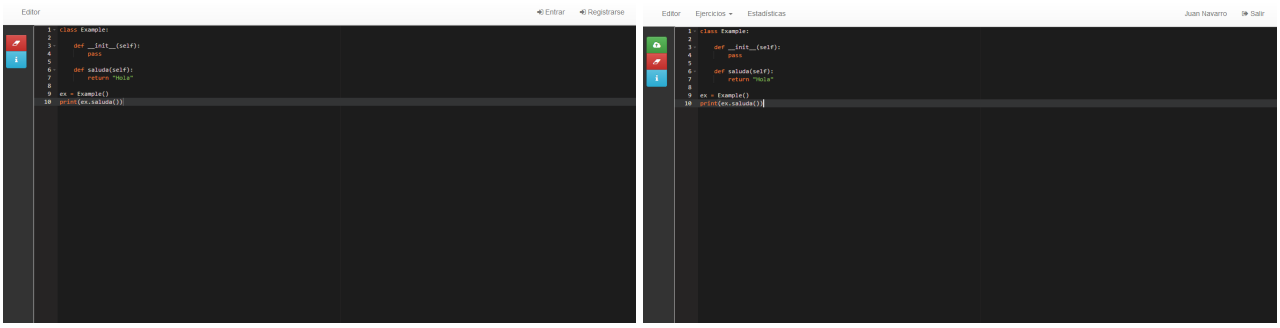
La barra de navegación del rol administrador es la misma que la barra del rol profesor, añadiéndole los ajustes que dará acceso a la base de datos.

Alertas

- Alerta de éxito: esta alerta se mostrará cuando las peticiones se ejecuten correctamente.
- Alerta de error: esta alerta se mostrará cuando se produzca un error al ejecutar una petición.
- Alerta de carga: esta alerta se mostrará cuando se esté cargando una funcionalidad.

3.4.2 Editor

Esta vista muestra distintas funciones según el rol que tenga el usuario (ver figura 3.4) Esta



(a) Rol invitado (b) Resto de roles

Figura 3.4: *Vista del editor*

vista tiene dos funciones comunes a todos los roles.

La primera de ellas (botón rojo) borra todas las líneas escritas en el editor. La segunda, muestra un modal de ayuda. Este modal contiene una ayuda muy básica de programación en Python. También se puede ver en la figura 3.4 que el editor resalta con colores las palabras claves el lenguaje Python.

Al entrar en la aplicación se activa una nueva funcionalidad. Esta funcionalidad consiste en subir el código (al pulsar el botón verde) que se ha escrito en el editor. Al pulsar el botón se pedirá el nombre del fichero.

3.4.3 Mis ejercicios

Esta vista muestra la funcionalidad según el estado del ejercicio. Esta vista sólo está disponible para el rol alumno. En la figura 3.5 se muestran tres ejercicios, donde cada ejercicio se muestra en un estado diferente. El primer ejercicio se encuentra en el estado de corregido por eso en la columna “Acciones” no hay ninguna funcionalidad. Y se puede ver la nota del ejercicio en su correspondiente columna.

El segundo ejercicio se encuentra en estado entregado. Al estar en este estado, en la columna “Acciones” se habilita la funcionalidad de corregir el ejercicio.

Y el tercer y último ejercicio se encuentra en el estado no entregado. Al no estar entregado,

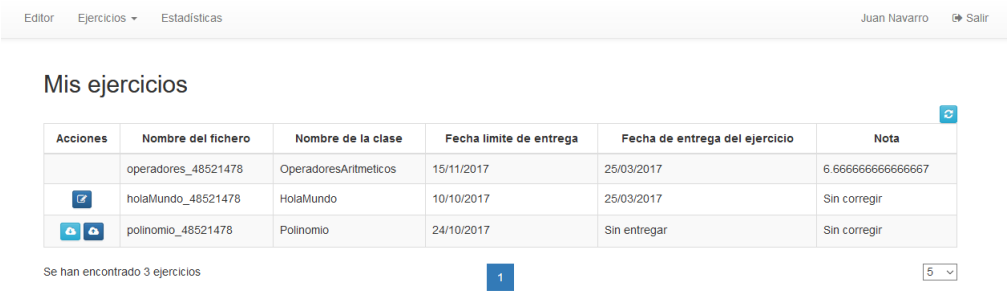


Figura 3.5: *Vista de los ejercicios asociados al alumno*

se habilita la funcionalidad de descargar la plantilla del alumno. También está habilitada la funcionalidad de subir el ejercicio. Al subirlo, el ejercicio pasaría al estado entregado.

3.4.4 Estadísticas

Esta vista muestra las estadísticas según el ejercicio y según el alumno. Como se puede ob-

(a) Rol alumno (b) Rol profesor, rol admin

Figura 3.6: *Vista de las estadísticas*

servar en la figura 3.6 el layout de esta vista es de dos columnas.

En la primera columna hay un desplegable con todos los ejercicios. En el botón buscar se obtienen las estadísticas del ejercicio marcado.

Estas estadísticas consiste en calcular el número de alumnos que han aprobado el ejercicio, el número de alumnos que lo han suspendido y el número de alumnos que aún no han entregado el ejercicio. Una vez obtenidos estos datos, se calcula el porcentaje de aprobados, suspensos y no entregados y se pinta en un gráfico circular.

También se pensó en calcular el número de alumnos que habían obtenido un cero, un uno, un dos, un tres, etc calculando el porcentaje de cada nota y pintándolo en otro gráfico circular.

En la segunda columna, se mostrará un desplegable con todos los alumnos en el caso de que el usuario tenga el rol profesor o el rol admin. Si el usuario tiene el rol alumno, entonces en la vista habrá un campo con el alumno bloqueado para no poder modificarse y un alumno no acceda a los datos de otros alumnos.

Una vez seleccionado el alumno, se obtendrá las estadísticas de dicho alumno al hacer click en el botón de buscar.

Cuando se obtengan todos los ejercicios de dicho alumno, se calculará el número de ejercicios aprobados, suspensos y sin corregir (estén entregados o no). Una vez obtenidos estos datos, se calcula el porcentaje de los ejercicios entregados, suspensos y sin corregir y se pintan en un gráfico circular.

También se pensó en poner en un gráfico de barras todas las notas de los ejercicios entregados.

3.4.5 Nuevo ejercicio

Esta vista sólo está disponible para aquel usuario que tenga rol profesor o rol administrador. Esta vista permite crear un nuevo ejercicio. Esta vista tiene dos partes bien diferenciadas. La

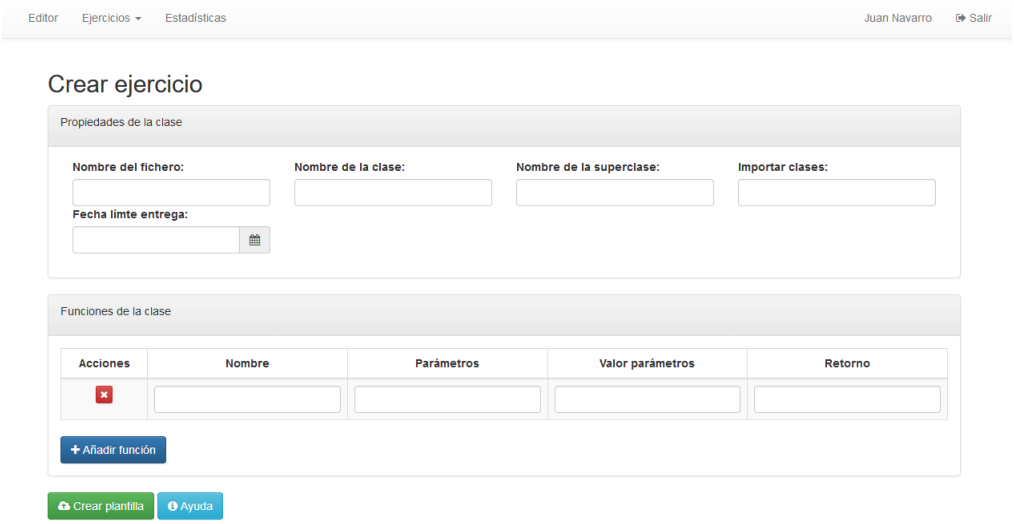


Figura 3.7: *Vista para crear ejercicios*

primera parte hace referencia a las propiedades de la clase del ejercicio y la segunda parte contiene las funciones que tiene que implementar el alumno. Al final de la vista hay dos botones, uno crea las plantillas y el otro muestra un modal de ayuda para rellenar correctamente la vista.

Propiedades de la clase

- Nombre del fichero: Este campo es obligatorio. El nombre del fichero no puede contener el caracter '_'.
- Nombre de la clase: Este campo es obligatorio.
- Nombre de la superclase: Nombre de la clase padre en caso de haber herencia. Si no la hay, se deja en blanco.
- Importar clases: En este campo se pone las líneas de código para importar las clases. Si hay más de una, se separarán mediante comas. Si no se importa nada, entonces se dejará en blanco.

- Fecha límite de entrega: Es la fecha límite que tiene el alumno para entregar el ejercicio.

Funciones de la clase

- Nombre: Nombre de la función.
- Parámetros: Parámetros de la función. Los parámetros van separados mediante comas. Si no tiene parámetros, entonces se dejará en blanco.
- Valor parámetros: Valor de los parámetros de la función para hacer las pruebas. Los valores van separados mediante comas. Se permite hacer varias pruebas de la misma función. Para ello, los valores irán separados por el caracter de escape ';'. Si no tiene parámetros, entonces se dejará en blanco.
- Retorno: Valor o fórmula que irá en el return de la función. Se permite introducir polinomios de cualquier grado y cualquier otra función matemática que esté en la librería math de python.

Al hacer click en el botón "Crear plantilla", se mandará una petición al servidor para que genere las plantillas del alumno, del profesor y la plantilla correctora.

En esta aplicación también se permite al profesor modificar la plantilla del profesor accediendo al directorio que las contiene. El profesor podrá abrir la plantilla y modificarla para añadirle una funcionalidad más compleja. Bastará con acceder a la función que desee e implementarla con la nueva funcionalidad. Es obligatorio devolver un valor en esta función ya que la plantilla correctora usará este valor devuelto para evaluar el ejercicio. En el caso de usar herencia (superclase), el profesor tendrá que añadir el fichero que contenga la superclase en el directorio.

3.4.6 Ejercicios mandados

Esta vista sólo está disponible para aquel usuario que tenga un rol profesor o un rol administrador. Esta vista tiene tres funcionalidades.

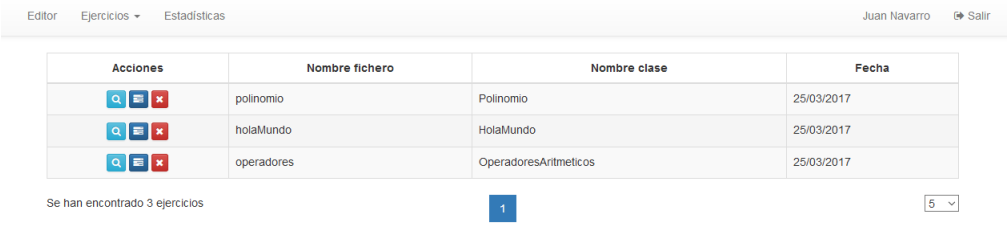


Figura 3.8: *Vista para mostrar los ejercicios mandados*

La primera funcionalidad es mostrar los detalles de cada ejercicio. Como se puede observar en la figura 3.9, esta funcionalidad muestra las propiedades de la clase del ejercicio y las funciones de la clase. Esta vista no permite modificar nada, sólo sirve para ver en qué consiste el ejercicio mandado.

La segunda funcionalidad es ver las notas de cada ejercicio de los alumnos. Como se puede

Figura 3.9: *Detalles de los ejercicios mandados*

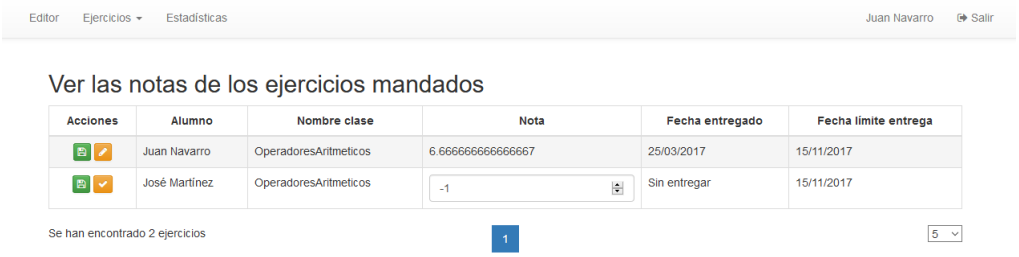


Figura 3.10: *Ver las notas de los alumnos*

observar en la figura 3.10, esta nueva vista tiene una única funcionalidad que es la de modificar la nota del alumno. Esta vista tiene dos botones:

- Botón de guardado: Al hacer click en este botón se manda una petición para actualizar la nota de ese ejercicio de dicho alumno en la base de datos.
- Botón modificar: Al hacer click aquí se habilita el campo de edición de la nota del ejercicio.

Si se modifica la nota pero no se hace click en el botón de guardar, no se actualizará la nota y se perderá el cambio.

Por último, la tercera funcionalidad consiste en borrar el ejercicio. Al borrarlo, se borrarán las plantillas que haya en el servidor y todos los datos que haya en la base de datos asociados a dicho ejercicio.

3.4.7 Ajustes

Como ya se ha explicado anteriormente, esta vista da acceso completo a la base de datos aunque solo se accede a las propiedades del alumno. Como se puede observar en la figura 3.11, sólo se accede al nombre, al usuario y al rol del usuario.

Esta vista tiene dos partes bien diferenciadas.

La primera es un filtro que permite buscar por nombre, apellidos, usuario o rol. También está permitido combinar varios, por ejemplo por nombre y por apellidos. Para mostrar todos

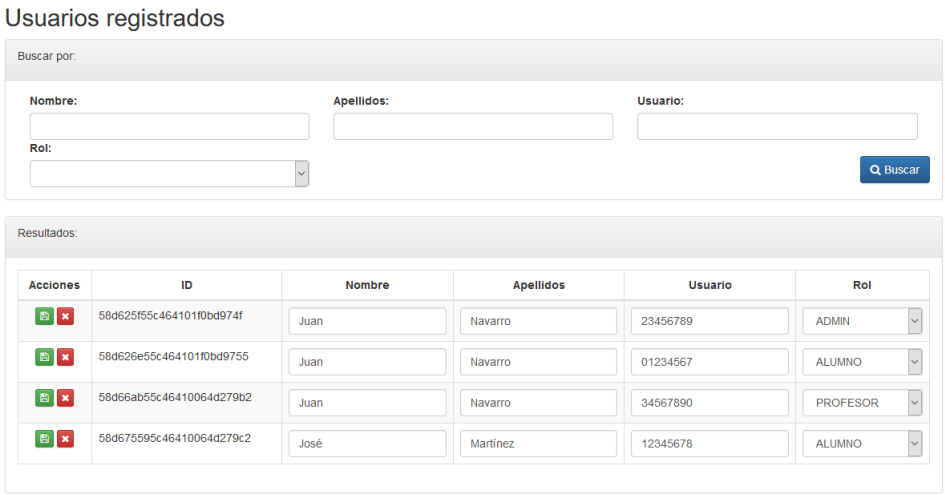


Figura 3.11: *Ajustes*

los usuarios, se dejarán en blanco los campos del filtro.

La segunda parte muestra una tabla con los datos. Esta tabla permitirá ver el identificador del alumno, el nombre, los apellidos, el usuario y el rol.

Esta vista tiene dos funcionalidades:

- Actualizar usuario: Al hacer click en el botón verde se mandará una petición al servidor con los nuevos datos del usuario para actualizarlo.
- Borrar usuario: Al hacer click en el botón rojo se mandará una petición al servidor los datos del usuario para borrarlo de la base de datos.

4 Desarrollo del servidor

4.1 Introducción

En este capítulo se va a explicar la estructura del proyecto que le corresponde al servidor. También se explicará el funcionamiento del servidor. Es decir, se explicarán las principales funcionales: login, signup, crear plantillas, obtener los ejercicios, etc.

4.2 Estructura

En la figura 4.1 se muestra la estructura del proyecto, pero en esta sección se va a explicar la estructura que le corresponde al servidor.

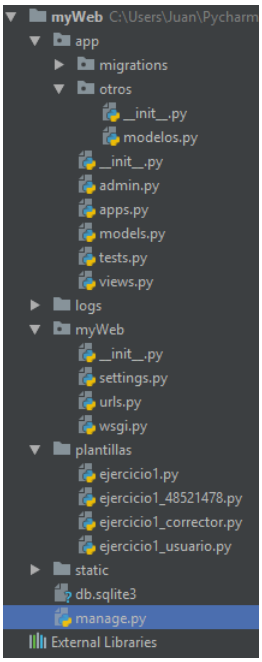


Figura 4.1: *Estructura del servidor*

A continuación se explica la estructura:

- app: Carpeta que pertenece a un proyecto Django. Esta carpeta se ha explicado en el tema 2.
  - otros: Carpeta creada para añadir la funcionalidad del servidor mediante clases.
- logs: Carpeta que contendrá ficheros .log que contendrán mensajes informativos que generará la aplicación.

- myWeb: Carpeta que pertenece a un proyecto Django. Esta carpeta se ha explicado en el tema 2.
- plantillas: Esta carpeta contendrá las plantillas generadas por la aplicación. Los ejercicios subidos por el alumno se almacenarán en este directorio.
- static: Carpeta que contiene los ficheros del cliente. Esta carpeta ya se ha explicado en el tema 3.

4.3 Composición de la base de datos

Una base de datos MongoDB está compuesta por colecciones y las colecciones están compuestas por documentos. A continuación se explican las colecciones usadas en la base de datos.

- alumnos: Esta colección contendrá todos los datos relacionados con el usuario. El documento está compuesto por:
  - _id: Identificador del usuario.
  - nombre: Nombre del usuario.
  - apellidos: Apellidos del usuario.
  - rol: Rol del usuario.
  - usuario: DNI del usuario.
  - passw: Contraseña cifrada del usuario.
  - ejerciciosMandados: En esta lista se guardarán los ejercicios que se asocien al alumno. Si el usuario no tiene rol 'ALUMNO' esta lista estará vacía. Esta lista contendrá el siguiente documento.
    - fechaEntregado: Fecha en la que se entrega el ejercicio.
    - nombreFichero: Nombre del fichero del ejercicio.
    - nombreClase: Nombre de la clase del ejercicio.
    - fechaLimite: Fecha límite de entrega del ejercicio.
    - nota: Nota del ejercicio.
    - idPlantilla: Identificador de la plantilla del profesor de dicho ejercicio.
- plantillas_ejercicios: Esta colección contendrá todos los ejercicios creados por el profesor. El documento estará compuesto por:
  - _id: Identificador de la plantilla del ejercicio del profesor.
  - propClase: Contiene las propiedades de la clase. El documento contendrá los siguientes valores:
    - nombreFichero: Nombre del fichero del ejercicio.
    - importarClases: Se guardan todas las clases que se tienen que importar.
    - nombreClase: Nombre de la clase del ejercicio.
    - nombreSuperclase: Se guarda el nombre de la superclase en caso de haber herencia.
    - fechaLimite: Fecha límite de entrega del ejercicio.

- funciones: Contiene una lista con todas las funciones que deberá rellenar el alumno. El documento contendrá:
  - retorno: Contiene el valor de retorno de la función.
  - nombre: Nombre de la función del ejercicio.
  - parametros: Parametros que recibe la función.

- roles: Contiene los todos los roles de la aplicación. El documento contendrá:
  - _id: Id del rol
  - roles: Una lista con todos los roles de la aplicación.

4.4 Funcionalidad

Como ya se ha explicado en el tema 2, el framework Django usa el patrón MVC (explicado en el tema 1). Como en esta aplicación se han usado varios frameworks compatibles (Django, AngularJS, Bootstrap...), las vistas no son servidas por Django pero si los controladores para la funcionalidad del servidor.

4.4.1 Inicio

Este controlador se encarga de servir la vista principal. Después será AngularJS quién se encargue de navegar entre las diferentes vistas.

4.4.2 Login

Este controlador realiza la funcionalidad del login. Recibe dos parámetros:

- usuario: Recibe el usuario de la persona que quiere entrar en el sistema.
- passw: Recibe la contraseña sin cifrar de la persona.

Este controlador se conecta a la base de datos, cifra la contraseña y le solicita a la base de datos todos los usuarios que cumpla la siguiente condición: devolver todos los usuarios cuyo usuario sea el pasado por parámetro y cuya contraseña sea la pasada por parámetro cifrada.

Si el usuario está registrado, la consulta devolverá un único resultado y por tanto el controlador devolverá el usuario. Si por el contrario el usuario no está registrado, la consulta no devolverá resultados y por tanto el controlador devolverá un mensaje de error.

4.4.3 Signup

Este controlador se encarga de registrar a una persona. Recibe los siguientes parámetros:

- nombre: Nombre de la persona

- apellidos: Apellidos de la persona
- usuario: Usuario de la persona que será el DNI
- passw: Contraseña del usuario sin cifrar

A continuación, se conecta a la base de datos y se crea el documento que se va a guardar en la base de datos. El documento contiene los siguientes campos: nombre, apellidos, usuario, rol (por defecto 'ALUMNO') y ejerciciosMandados (una lista vacía).

4.4.4 Crear plantillas

En esta funcionalidad se generan las plantillas para el profesor, para el alumno y la plantilla encargada de corregir el ejercicio del profesor.

Esta funcionalidad recibe los siguientes parámetros:

- propClase: Es un JSON con todas las propiedades de la clase del ejercicio.
- funciones: Es un JSON con todas las funciones que tiene el ejercicio.

Una vez recogidos los parámetros, se generan las plantillas del profesor, del alumno y la plantilla del corrector.

El siguiente es conectarse a la base de datos para añadir el nuevo ejercicio que se ha creado.

Por último, sólo queda asociar el nuevo ejercicio a cada alumno registrado.

Generación de la plantilla del profesor

Esta clase recibe las propiedades de la clase y las propias funciones. En el constructor de la clase se inicializa la variable con la ruta y el nombre del ejercicio y la variable con el nombre de la clase del Ejercicio.

A continuación, se comprueba el fichero existe, porque si el fichero existe no se puede crear el ejercicio puesto que ya está creado y para modificarlo, se haría de otra manera (accediendo directamente al fichero).

Una vez comprobado que el ejercicio no existe, se crea una variable de tipo 'File' para poder escrbir en un fichero. Se configurará este fichero para convertirlo en un fichero de tipo python.

El siguiente paso es recuperar los datos de todas las clases que se quieren importar. Estos datos vienen en forma de lista, por lo tanto, se recorrerá dicha lista y se irán añadiendo los imports a la plantilla.

A continuación, se comprueba si se usa herencia. En caso afirmativo, se añadirá a la plantilla la superclase. Llegado a este punto, la clase ya contiene todas sus propiedades: el nombre, la herencia y los imports.

El siguiente paso es recorrer la lista de las funciones. Para cada función, lo primero que se hace es comprobar si dicha función contiene parámetros ya que si tiene parámetros, es necesario adapatar la lista de parámetros a la definición de la función en python.

Una vez realizado esto, dentro de la función se le añade el return con la expresión que ha generado el profesor.

Generación de la plantilla del alumno

La generación de la plantilla de alumno es casi idéntica a la plantilla del profesor, con la diferencia del return.

En esta plantilla se crea el constructor de igual manera que en la plantilla del profesor. En esta plantilla también se crea un constructor vacío que no recibe parámetros.

La definición de las funciones es muy semejante a la plantilla del profesor. En esta plantilla, se añade un comentario en la función para indicar que hay que rellenarla. Y a diferencia de la plantilla del profesor, en esta plantilla se rellena la función con un return -1.

Generación de la plantilla correctora

Esta plantilla recibe dos parámetros: el ejercicio del alumno a corregir y el ejercicio creado por el profesor.

En el constructor se obtiene el nombre de la clase del ejercicio del alumno y se obtiene el nombre de la clase de la plantilla del profesor. También se obtiene los nombres de los ficheros.

Una vez que se ha recuperado esta información se realiza el import de la clase del ejercicio del alumno y el import de la clase de la plantilla del profesor. Realizados los imports, se asocian las clases a variables de clase. También se inicializa la variable nota con el valor 10.0 y se calcula el número de pruebas y se almacena en otra variable.

A continuación, se crea una función de prueba por cada función que creó el profesor para el ejercicio. En la función se crea un if que será el encargado de restar nota en el caso de que la función esté mal. Si está mal, el if llamará a la función error() que contendrá la clase.

Esta función error() actualiza la nota de la siguiente manera:

$$nota = nota - \frac{10.0}{númeroDePruebas}$$

Por último, en esta plantilla se crea una función principal que es la encargada de realizar las pruebas. Esta función devuelve la nota final del alumno.

4.4.5 Obtener ejercicios

Esta funcionalidad recupera de la base de datos todos los ejercicios creados por el profesor. Lo primero que se hace es conectarse a la base de datos. A continuación, se pide a la base de datos recuperar todos los ejercicios que existan en la colección correspondiente. Por último la funcionalidad devuelve la lista con todos los ejercicios y el número de ejercicios que ha encontrado.

Esta funcionalidad se ha usado para mostrarle a un usuario con rol 'PROFESOR' o rol 'ADMIN' todos los ejercicios mandados.

4.4.6 Borrar ejercicios

Esta funcionalidad se encarga de borrar un ejercicio de la base de datos. También borrará las plantillas del alumno, del profesor y la plantilla correctora.

Esta funcionalidad recibe el identificador del ejercicio como parámetro. A continuación, se conecta con la base de datos y se recuperan todos los alumnos cuyo rol sea 'ALUMNO'.

Se recorre la lista de los alumnos verificando que dicho alumno tiene asociado el ejercicio. De ser así, se borra el ejercicio de la lista "ejerciciosMandados" que contiene todos los ejercicios asociados al alumno y se borra el ejercicio entregado en caso de estarlo.

El ejercicio también será borrado en la colección "plantillas_ejercicios" de la base de datos.

4.4.7 Obtener ejercicios del alumno

Esta funcionalidad recupera de la base de datos todos los ejercicios asociados al alumno. Lo primero que se hace es conectarse a la base de datos. A continuación, se pide a la base de datos recuperar todos los ejercicios del alumno cuyo id se pasa como parámetro. Por último la funcionalidad devuelve la lista con todos los ejercicios y el número de ejercicios que ha encontrado.

4.4.8 Corregir ejercicio

Esta funcionalidad recibe dos parámetros:

- idUsuario: Es el id del alumno que quiere corregir el ejercicio.
- ejercicio: Es un JSON que contiene los datos del ejercicio que se quiere corregir.

Lo primero que se hace en esta funcionalidad es conectarse a la base de datos y recuperar el ejercicio del profesor.

A continuación, se llama a la clase encargada de realizar las pruebas. Esta llamada se encapsula en un bloque `try - except` para cubrir el caso de que exista un error de sintaxis y

el ejercicio lance una excepción. De esta manera se captura la excepción y se le pone un cero como nota del ejercicio.

Ya por último, se modifica la nota obtenida en la base de datos. Se recuperan todos los ejercicios del alumno, buscando el ejercicio correspondiente para actualizarle la nota.

4.4.9 Actualizar nota

Esta funcionalidad sólo está disponible para aquellos usuarios que tengan el rol 'PROFESOR' o el rol 'ADMIN'.

Este controlador recibe el identificador del alumno, el nombre del fichero, el nombre de la clase del ejercicio y la nueva nota del ejercicio como parámetros.

Se hace una conexión a la baes de datos y se recupera al usuario filtrando por su identificador. Una vez recuperado el alumno, se busca el ejercicio, se accede a la propiedad de la nota y se modifica.

4.4.10 Generación de las estadísticas

Este controlador calcula las estadísticas de un ejercicio. El controlador recibe el identificador del ejercicio.

Accede a la base de datos y recupera todos los alumnos. Por cada alumno, se guarda en un array asociativo el número de alumnos aprobados, suspensos y el número de alumnos que aún no han entregado el ejercicio.

También en el array se calcular el número de alumnos cuya nota va desde cero hasta uno, desde uno hasta dos, desde dos hasta tres, etc.

Estos datos se devuelven al cliente y se pintan en la vista a través de la librería Google Charts.

5 Pruebas de la aplicación

Se han hecho diversas pruebas para verificar el correcto funcionamiento. Se trata de conseguir estar en el papel de un usuario que desconoce el funcionamiento de la aplicación. Así como conseguir estar en el papel del usuario que trata de dañar la aplicación haciéndola fallar a propósito. También se simulará un usuario que conoce perfectamente la aplicación y la usará correctamente.

5.1 Registro

Se ha probado la aplicación poniendo el formulario del registro en blanco (no cumplimentado). También se ha probado a poner el campo DNI errónea. Obteniendo varios supuestos:

- El DNI no es un número si no un texto.
- El DNI es más corto.
- El DNI es más largo.

Se comprueba el correcto funcionamiento de la funcionalidad con el campo contraseña con los siguientes supuestos:

- El campo contraseña tiene menos de cuatro caracteres.
- El campo contraseña y el campo repita la contraseña no coinciden.

También se ha probado la aplicación rellenando el formulario correctamente. En la tabla 5.1 se detallan las pruebas realizadas a la funcionalidad del registro de la aplicación web.

5.2 Login

Se ha probado la aplicación poniendo el formulario del registro en blanco (no cumplimentado). También se ha probado poniendo las credenciales incorrectamente y poniendo las credenciales correctamente. Se comprueba que el campo contraseña tenga al menos cuatro caracteres como se pedía en el registro.

En la tabla 5.2 se detallan las pruebas realizadas a la funcionalidad del login de la aplicación web.

5.3 Ajustes

Como ya se ha explicado en el tema "Desarrollo del cliente", esta vista contiene un formulario que hace de filtro. Por tanto se probará que el resultado de la petición al servidor sea correcto según el criterio del filtro. Para ello se probará:

Prueba	Requerimientos	Acción	Resultados esperados
Rellenar el formulario con todos los datos en blanco	Tener el servidor MongoDB arrancado. Tener la base de datos creada	Hacer click en el botón de registrarse	Ver mensajes de error en el formulario y no hacer el registro en la base de datos
Rellenar el campo DNI como texto	Tener el servidor MongoDB arrancado. Tener la base de datos creada	Hacer click en el botón de registrarse	Ver mensajes de error en el formulario y no hacer el registro en la base de datos
Rellenar el campo DNI con menos de 8 números	Tener el servidor MongoDB arrancado. Tener la base de datos creada	Hacer click en el botón de registrarse	Ver mensajes de error en el formulario y no hacer el registro en la base de datos
Rellenar el campo DNI con más de 8 números	Tener el servidor MongoDB arrancado. Tener la base de datos creada	Hacer click en el botón de registrarse	Ver mensajes de error en el formulario y no hacer el registro en la base de datos
Rellenar el campo contraseña con menos de 4 caracteres	Tener el servidor MongoDB arrancado. Tener la base de datos creada	Hacer click en el botón de registrarse	Ver mensajes de error en el formulario y no hacer el registro en la base de datos
Rellenar el campo repetir contraseña con valor distinto al campo contraseña	Tener el servidor MongoDB arrancado. Tener la base de datos creada	Hacer click en el botón de registrarse	Ver mensajes de error en el formulario y no hacer el registro en la base de datos
Rellenar todos los campos correctamente	Tener el servidor MongoDB arrancado. Tener la base de datos creada	Hacer click en el botón de registrarse	Hacer el registro en la base de datos

Tabla 5.1: *Pruebas de la funcionalidad del registro*

- Todos los campos del filtro en blanco.
- Se probará la funcionalidad filtrando por nombre.
- Se probará la funcionalidad filtrando por apellidos.
- Se probará la funcionalidad filtrando por usuario.
- Se probará la funcionalidad filtrando por rol.

En la tabla 5.3 se detallan las pruebas realizadas a la funcionalidad de los ajustes de la aplicación web.

5.4 Crear ejercicio

Se han probado todos los campos obligatorios. En el campo nombre fichero, se comprobará que el nombre no tenga caracteres prohibidos. También se pondrán todos los campos obli-

Prueba	Requerimientos	Acción	Resultados esperados
Rellenar el formulario con todos los datos en blanco	Tener el servidor MongoDB arrancado. Tener la base de datos creada. Tener usuarios en la base de datos	Hacer click en el botón entrar	Ver mensajes de error en el formulario y no entrar en el sistema
Rellenar el campo contraseña con menos de 4 caracteres	Tener el servidor MongoDB arrancado. Tener la base de datos creada. Tener usuarios en la base de datos	Hacer click en el botón de registrarse	Ver mensajes de error en el formulario y entrar en el sistema
Rellenar el formulario con las credenciales incorrectas	Tener el servidor MongoDB arrancado. Tener la base de datos creada. Tener usuarios en la base de datos	Hacer click en el botón entrar	Ver mensajes de error en el formulario y no entrar en el sistema
Rellenar el formulario con las credenciales correctamente	Tener el servidor MongoDB arrancado. Tener la base de datos creada. Tener usuarios en la base de datos	Hacer click en el botón entrar	Entrar en el sistema

Tabla 5.2: *Pruebas de la funcionalidad del login*

gatorios correctamente.

En la tabla 5.4 se detallan las pruebas realizadas.

5.5 Modificar nota

En esta sección se prueba la funcionalidad modificar nota. Se han probado los siguientes casos:

- La nota es superior a 10
- La nota es inferior a 0 y distinta a -1
- La nota es igual a -1
- La nota está en el rango de 0 a 10

En la tabla 5.5 se detallan las pruebas realizadas.

Prueba	Requerimientos	Acción	Resultados esperados
Rellenar el formulario con todos los datos en blanco	Tener un usuario con rol ADMIN	Hacer click en el botón buscar	Se devuelven todos los usuarios que haya en la base de datos
Rellenar el campo nombre con un nombre (por ejemplo “Juan”)	Tener un usuario con rol ADMIN	Hacer click en el botón buscar	Se devuelven todos los usuarios cuyo nombre sea “Juan” que haya en la base de datos
Rellenar el campo apellidos con unos apellidos (por ejemplo “Navarro Rey”)	Tener un usuario con rol ADMIN	Hacer click en el botón buscar	Se devuelven todos los usuarios cuyo apellidos sea “Navarro Rey” que haya en la base de datos
Rellenar el campo usuario con un usuario (por ejemplo “12345678”)	Tener un usuario con rol ADMIN	Hacer click en el botón buscar	Se devuelven todos los usuarios cuyo usuario sea “12345678” que haya en la base de datos
Rellenar el campo rol con un rol (por ejemplo “ALUMNO”)	Tener un usuario con rol ADMIN	Hacer click en el botón buscar	Se devuelven todos los usuarios cuyo rol sea “ALUMNO” que haya en la base de datos

Tabla 5.3: *Pruebas de la funcionalidad de los ajustes*

5.6 Subir ejercicio

En esta sección se probará subir el ejercicio al servidor cuando el ejercicio aún no ha sido entregado y cuando ya se haya entregado.
En la tabla 5.6 se detallan las pruebas realizadas.

5.7 Corregir ejercicio

En esta sección se prueban las siguientes acciones:

- Mostrar el botón de corregir el ejercicio cuando el ejercicio esté entregado
- Poner un cero al ejercicio cuando haya un error de sintaxis
- Añadir la nota en la base de datos.

En la tabla 5.7 se detallan las pruebas realizadas.

Prueba	Requerimientos	Acción	Resultados esperados
No se rellena el campo del nombre del fichero	Tener un usuario con rol ADMIN o con rol PROFESOR	Hacer click en el botón crear plantilla	Mostrar mensaje de error que indique que el nombre del fichero es obligatorio
Se rellena el campo del nombre del fichero con el caracter '_'	Tener un usuario con rol ADMIN o con rol PROFESOR	Hacer click en el botón crear plantilla	Mostrar mensaje de error que indique que el nombre del fichero no puede contener el caracter '_'
No se rellena el campo del nombre de la clase	Tener un usuario con rol ADMIN o con rol PROFESOR	Hacer click en el botón crear plantilla	Mostrar mensaje de error que indique que el nombre de la clase es obligatorio
No se pone ninguna función en el ejercicio	Tener un usuario con rol ADMIN o con rol PROFESOR	Hacer click en el botón crear plantilla	Mostrar mensaje de error que indique que al menos haya que poner una función en el ejercicio
No se pone nombre a la función	Tener un usuario con rol ADMIN o con rol PROFESOR	Hacer click en el botón crear plantilla	Mostrar mensaje de error que indique que es obligatorio poner un nombre a la función del ejercicio
No se pone valor de retorno a la función	Tener un usuario con rol ADMIN o con rol PROFESOR	Hacer click en el botón crear plantilla	Mostrar mensaje de error que indique que es obligatorio poner un valor de retorno a la función del ejercicio
Se ponen todos los campos correctamente	Tener un usuario con rol ADMIN o con rol PROFESOR	Hacer click en el botón crear plantilla	Se crean las plantillas y se asocia el ejercicio a todos los usuarios cuyo rol sea ALUMNO

Tabla 5.4: *Pruebas de la funcionalidad nuevo ejercicio*

Prueba	Requerimientos	Acción	Resultados esperados
Se le asocia al campo nota un valor superior a 10	Tener un usuario con rol ADMIN o con rol PROFESOR	Hacer click en el botón de guardar	Mostrar mensaje de error y no actualizar la nota
Se le asocia al campo nota un valor inferior a 0 y distinto a -1	Tener un usuario con rol ADMIN o con rol PROFESOR	Hacer click en el botón de guardar	Mostrar mensaje de error y no actualizar la nota
Se le asocia al campo nota un valor igual a -1	Tener un usuario con rol ADMIN o con rol PROFESOR	Hacer click en el botón de guardar	Se actualiza la nota. La nota -1 significa que el ejercicio no está corregido.
La nota está en el rango de 0 a 10	Tener un usuario con rol ADMIN o con rol PROFESOR	Hacer click en el botón de guardar	Se actualiza la nota.

Tabla 5.5: *Pruebas de la funcionalidad modificar nota*

Prueba	Requerimientos	Acción	Resultados esperados
Subir un ejercicio no entregado	Tener un usuario con rol ALUMNO	Hacer click en el botón subir ejercicio	Se sube el ejercicio y se añade la fecha en la que se ha subido el ejercicio
Subir un ejercicio ya entregado	Tener un usuario con rol ALUMNO	Hacer click en el botón subir ejercicio	Se muestra un mensaje de error y no se sube el nuevo ejercicio

Tabla 5.6: *Pruebas de la funcionalidad subir ejercicio*

Prueba	Requerimientos	Acción	Resultados esperados
Mostar el botón de corregir el ejercicio	Tener un usuario con rol ALUMNO. Tener un ejercicio entregado asociado a dicho alumno.	Mostar los ejercicios del alumno	Se deberá mostar un botón de color azul que permita corregir el ejercicio
Corregir un ejercicio con errores de sintaxis	Tener un usuario con rol ALUMNO. Tener un ejercicio entregado asociado a dicho alumno.	Hacer click en el botón corregir	Se pondrá un cero al ejercicio
Corregir un ejercicio sin errores de sintaxis	Tener un usuario con rol ALUMNO. Tener un ejercicio entregado asociado a dicho alumno.	Hacer click en el botón corregir	Se pondrá la nota correspondiente al ejercicio según el correcto funcionamiento del ejercicio

Tabla 5.7: *Pruebas de la funcionalidad corregir ejercicio*

6 Conclusiones y líneas futuras

Para finalizar la documentación del proyecto se muestran las conclusiones sobre el desarrollo del mismo y las posibles mejoras o nuevas funcionalidades que se le podrían añadir al sistema.

6.1 Conclusiones

Una vez finalizado el proyecto se puede decir que se ha cumplido con todos los objetivos propuestos.

Se ha desarrollado una aplicación web de evaluación de código automático, la cual evalúa los ejercicios resueltos por el alumno.

En el aspecto técnico, el hecho de trabajar con el framework AngularJS ha supuesto un punto muy relevante en el desarrollo el proyecto, ya que es un framework con el que estoy altamente familiarizado. La utilización del framework Django también ha sido muy útil para simplificar algunas funciones de la programación web.

En el aspecto personal, gracias a la realización de este proyecto he podido asentar las bases aprendidas durante la carrera. A parte de aprender a desarrollar una aplicación web, en este trabajo se ha aprendido a usar las principales tecnologías webs (HTML5, CSS3, JavaScript) y los principales frameworks existentes (AngularJS, Django, Bootstrap...).

6.2 Líneas futuras

A continuación se detallarán una serie de funcionalidades que sería interesante añadir al proyecto en versiones futuras.

Un punto interesante es permitir al profesor crear plantillas de ejercicios más completos mediante una interfaz web. Actualmente, esta funcionalidad está presente en la aplicación aunque es incómodo ya que es el profesor quién tiene que buscar la plantilla generada automáticamente. Modificarla sabiendo qué aspectos de la plantilla no se pueden modificar y tener que avisar a los alumnos.

También sería útil añadir un sistema de avisos. Como por ejemplo:

- La fecha límite de entrega del ejercicio está cerca.
- Hay nuevos ejercicios para realizar.
- La plantilla del ejercicio ha sido modificada por el profesor.
- Se ha borrado el ejercicio.

Bibliografía

[1] Poncier F. (n.d). El tutorial Jobeet, http://librosweb.es/libro/jobeet_1_4/

[2] Sims C., Jhonson H. L., Scrum: a Breathtakingly Brief And Agile Introduction

[3] Fowler M., Beck K., Beedle M., Cockburn A., Thomas D. Manifesto for Agile Software Development, http://agilemanifesto.org/

[4] Pilato C. Michael, Collins Sussman Ben, W. Fitzpatrick Brian, Version Control with Subversion

[5] Schmitt Christopher, Simpson Kyle, HTML5 Cookbook

[6] Eguiluz Javier, Introducción a CSS, https://librosweb.es/libro/css/

[7] Animaciones de CSS3, https://www.w3schools.com/css/css3_animations.asp

[8] Eguiluz Javier, Introducción a JavaScript, https://librosweb.es/libro/javascript/

[9] Crockford Douglas, Javascript: The Good Parts

[10] Documentación oficial de jQuery, https://api.jquery.com/

[11] Murphey Rebecca, Fundamentos de jQuery, https://librosweb.es/libro/fundamentos_jquery/

[12] Eguiluz Javier, Introducción a AJAX, https://librosweb.es/libro/ajax/

[13] Documentación oficial de python, https://docs.python.org/3/tutorial/index.html

[14] Otto Mark, Thornton Jacob, Bootstrap 3, el manual oficial, https://librosweb.es/libro/bootstrap_3/

[15] Documentación oficial de Bootstrap, http://getbootstrap.com/getting-started/

[16] García M. Saúl, La guía definitiva de django. Desarrolla aplicaciones Web de forma rápida y sencilla.

[17] Concepto de django, https://www.djangoproject.com/

[18] Instalación de django, https://www.djangoproject.com/download/

[19] Crear proyecto django, https://docs.djangoproject.com/en/1.10/intro/tutorial01/

[20] Green Brad, Seshadri Shyam, AngularJS

[21] Instalación de angularJS, https://angularjs.org/

[22] Crear un controlador angularJS, https://docs.angularjs.org/guide/controller

[23] Crear una factoría angularJS, https://docs.angularjs.org/guide/providers

[24] Directivas angularJS, https://docs.angularjs.org/api/ng#directive

[25] Concepto de mongoDB, https://www.mongodb.com/es

[26] Crear colección en mongoDB, https://docs.mongodb.com/manual/reference/method/db.createCollection/

[27] Añadir datos en mongoDB, https://docs.mongodb.com/manual/reference/method/db.collection.insert/

[28] Recuperar datos en mongoDB, https://docs.mongodb.com/manual/reference/method/db.collection.find/

[29] Borrar datos en mongoDB, https://docs.mongodb.com/manual/reference/method/db.collection.remove/

[30] Actualizar datos en mongoDB, https://docs.mongodb.com/manual/reference/method/db.collection.update/

[31] Documentación ofical de Google Charts, https://developers.google.com/chart/

[32] Documentación ofical de jQuery validation, https://jqueryvalidation.org/documentation/

[33] Documentación de jQuery confirm, https://craftpip.github.io/jquery-confirm/

[34] Documentación oficial de pymongo, https://api.mongodb.com/python/current/

A Manual del usuario

A.1 Introducción

En este anexo se va a explicar cómo se usa la aplicación web según el rol que le corresponda al usuario. También se explicará el rol 'INVITADO' que es el rol que tiene todo usuario antes de entrar en el sistema.

A.2 Rol invitado

Como ya se ha mencionado en el la introducción de este anexo, este rol le corresponde a cualquier usuario que no esté en el sistema ya sea porque no se haya registrado, no hay entrado en la aplicación o bien haya cerrado la sesión.

Al acceder a la aplicación web se muestra el contenido que se puede observar en la figura A.1.
Al acceder a la aplicación se accede al editor (explicado en la sección 3.4.2). En la parte

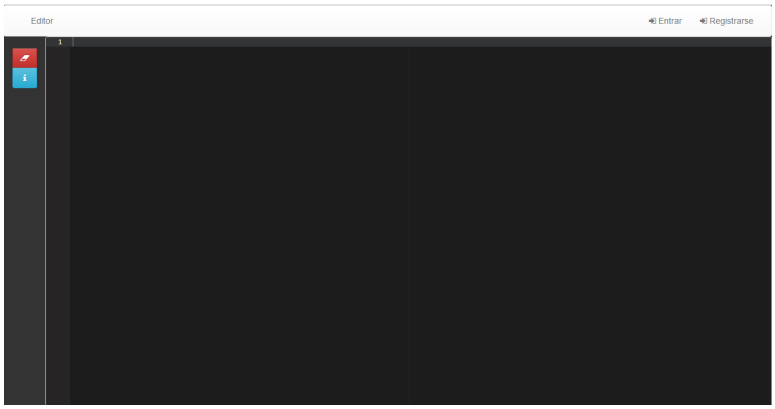


Figura A.1: *Vista inicial*

derecha de la barra de navegación, existen dos botones: entrar y registrarse.

Al hacer click en el primero se mostrará un formulario (ver figura A.2). Rellenando este formulario correctamente y haciendo click en el botón entrar, se mandará una petición al servidor para autenticarse y así poder entrar en el sistema.

El segundo botón es para registrarse en el sistema, creando un nuevo usuario. Este botón también muestra un formulario, como se puede observar en la figura A.3.

Una vez rellenado correctamente el formulario, el usuario se registrará haciendo click en el botón enviar. Por defecto, al usuario recién registrado en el sistema se le asocia el rol 'ALUMNO'. En caso de ser un profesor, éste tendrá que ponerse en contacto con el administrador para que le asocie el rol 'PROFESOR'.

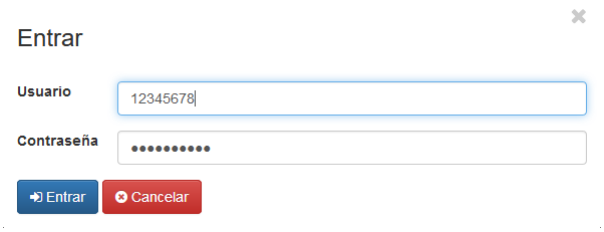


Figura A.2: *Formulario de LogIn*

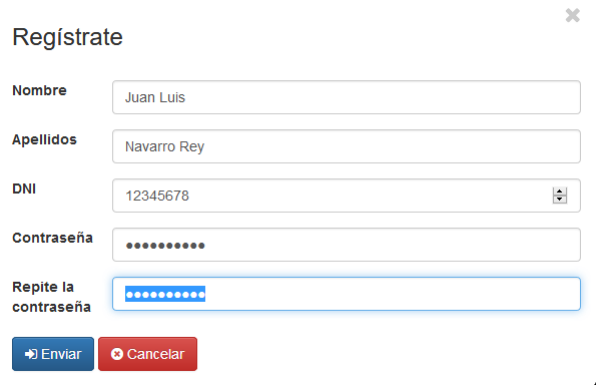


Figura A.3: *Formulario de registro*

A.3 Rol administrador

Como ya se ha explicado en la sección 3.2, el rol 'ADMIN' y el rol 'PROFESOR' son muy parecidos, de hecho la única diferencia que existe entre ellos es que el rol 'ADMIN' tiene la funcionalidad de modificar los datos de la base de datos.

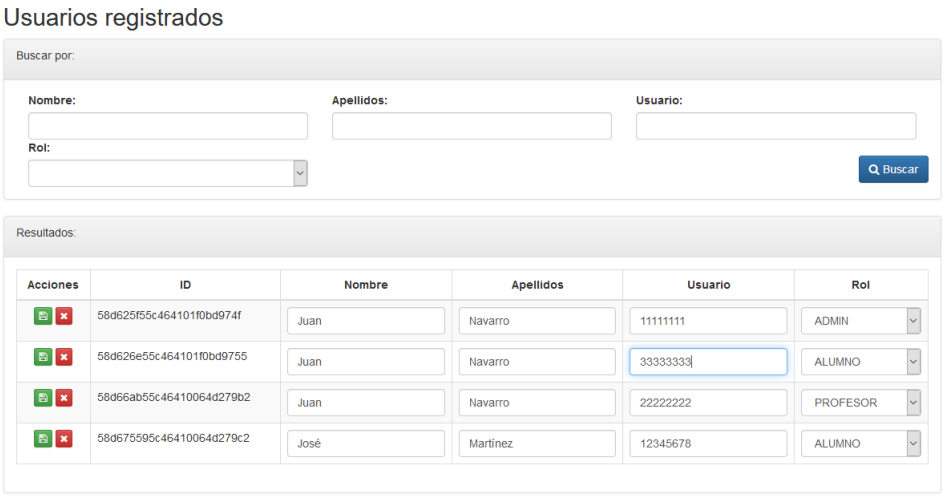
Por eso, en esta sección sólo se va a explicar dicha funcionalidad ya que el resto de las funcionalidades serán explicadas en la siguiente sección (A.4).

Al acceder a la aplicación como administrador, se habilitará un botón en la parte derecha de

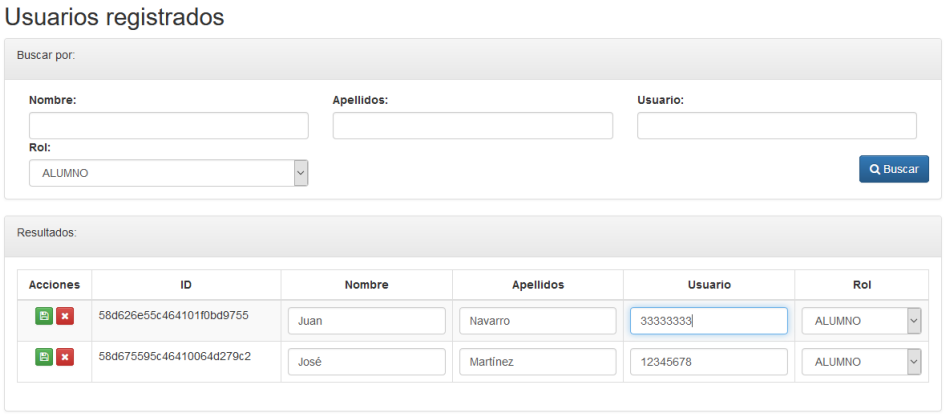
Figura A.4: *Barra de navegación*

la barra de navegación, llamado ajustes que dará acceso a la base de datos (ver figura A.4). Al hacer click en dicho botón se navega a otra vista (ver figura A.5).

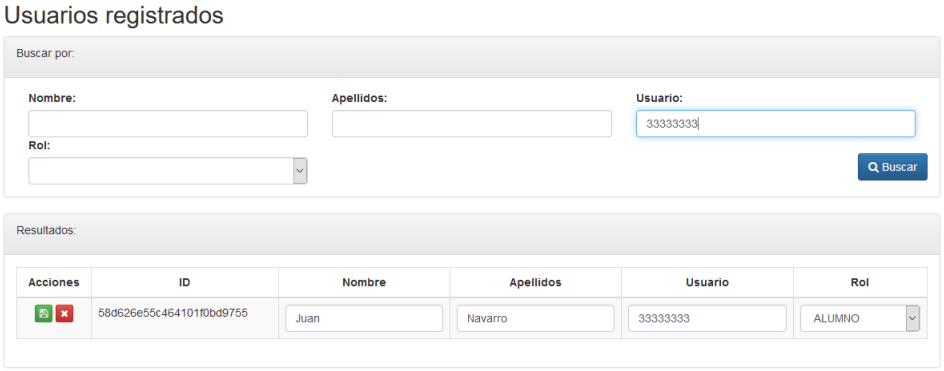
En esta vista se recuperan los usuarios de la base de datos permitiendo realizar búsquedas filtrando por cualquier propiedad asociado a un usuario. Esta vista ya ha sido explicada en la sección 3.4.7.



(a) Búsqueda sin filtro



(b) Búsqueda por rol



(c) Búsqueda por usuario

Figura A.5: *Editando la base de datos*

A.4 Rol profesor

En esta sección se va a explicar la aplicación web desde el punto de vista de un profesor. Como ya se ha explicado en la sección 3.2, el profesor podrá crear ejercicios, ver los ejercicios

mandados y ver las estadísticas de los alumnos.

A.4.1 Crear ejercicio

Aunque esta vista ya ha sido explicada en la sección 3.4.5, a continuación se pone un ejemplo de uso. Para ello se va a usar de base la imagen A.6.

En este ejemplo se va a crear una plantilla para el alumno con el nombre "areas_usuario.py".

Figura A.6: *Creando un ejercicio*

Este fichero contendrá una clase llamada "Areas" y a su vez esta clase tendrá dos funciones: círculo y rectángulo.

La función círculo recibirá un parámetro r que será el radio del círculo. En la plantilla del alumno la función devolverá un -1, mientras que en la plantilla del profesor la función devolverá $\pi \cdot r^2$ que es el contenido del campo de retorno. El campo "Valor parámetros" será usado por la plantilla evaluadora para comparar el resultado de la plantilla del profesor con el resultado de la plantilla del alumno de la siguiente forma:

$$\underbrace{circulo(2)}_{Plantilla\ profesor} == \underbrace{circulo(2)}_{Plantilla\ alumno}$$

$$\underbrace{circulo(3)}_{Plantilla\ profesor} == \underbrace{circulo(3)}_{Plantilla\ alumno}$$

$$\underbrace{circulo(4)}_{Plantilla\ profesor} == \underbrace{circulo(4)}_{Plantilla\ alumno}$$

$$\underbrace{circulo(5)}_{Plantilla\ profesor} == \underbrace{circulo(5)}_{Plantilla\ alumno}$$

La segunda función, la función rectángulo recibe dos parámetros: base del rectángulo b y la altura h. En la plantilla del alumno la función devolverá un -1, mientras que en la plantilla del profesor la función devolverá $b \cdot h$ que es el contenido del campo de retorno. El campo "Valor parámetros" será usado por la plantilla evaluadora para comparar el resultado de la plantilla del profesor con el resultado de la plantilla del alumno de la siguiente forma:

$$\underbrace{rectangulo(2,3)}_{Plantilla\ profesor} == \underbrace{rectangulo(2,3)}_{Plantilla\ alumno}$$

$$\underbrace{rectangulo(1,2)}_{Plantilla\ profesor} == \underbrace{rectangulo(1,2)}_{Plantilla\ alumno}$$

$$\underbrace{rectangulo(6,2)}_{Plantilla\ profesor} == \underbrace{rectangulo(6,2)}_{Plantilla\ alumno}$$

A.4.2 Ejercicios mandados

En esta vista se recuperan de la base de datos todos los ejercicios mandados y se muestran en una tabla como se puede ver en la figura A.7.

A cada ejercicio ya creado se le asocian tres funcionalidades:

Figura A.7: *Creando un ejercicio*

- Detalles: Permite ver los detalles del ejercicio. Es decir, se muestran las propiedades del ejercicio y sus funciones.
- Notas: Permite ver las notas de todos los alumnos de dicho ejercicio.
- Borrar ejercicio: Borra el ejercicio de la base de datos.

A.4.3 Estadísticas

En esta sección se explican las estadísticas que se han metido en la aplicación. Como se puede observar en la figura A.8, existen dos partes.

Por un lado están las estadísticas asociadas al ejercicio seleccionado. En la parte de la izquierda, hay un desplegable con todos los ejercicios de la base de datos y se selecciona uno. Al hacer click en el botón de buscar de debajo del desplegable se realiza una petición al servidor para calcular las estadísticas. Estas estadísticas consisten en calcular el número

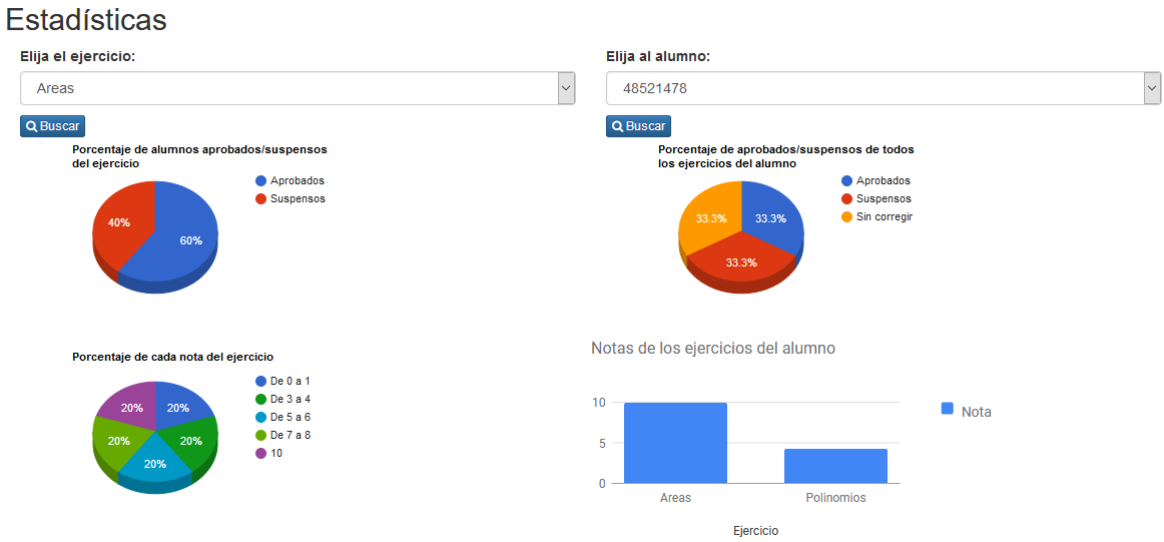


Figura A.8: *Interpretando las estadísticas*

de alumnos que han aprobado el ejercicio, el número de alumnos que lo han suspendido y el número de alumnos que aún no han entregado el ejercicio. En el segundo gráfico se calcula el número de alumnos que han obtenido un cero, un uno, un dos, un tres, etc calculando el porcentaje de cada nota.

La segunda parte hace referencia a las estadísticas de un alumno determinado. Para elegir al alumno hay un desplegable con todos los alumnos que hay en la base de datos. Se selecciona uno, al hacer click en el botón de buscar se envía una petición al servidor para recuperar las estadísticas de dicho alumno. Cuando se obtengan todos los ejercicios de dicho alumno, se calculará el número de ejercicios aprobados, suspensos y sin corregir (estén entregados o no). Una vez obtenidos estos datos, se calcula el porcentaje de los ejercicios entregados, suspensos y sin corregir y se pintan en un gráfico circular. También se pensó en poner en un gráfico de barras todas las notas de los ejercicios entregados.

A.5 Rol alumno

En esta sección se va explicar la aplicación web desde el punto de vista de un alumno. Como ya se ha explicado en la sección 3.2, el alumno podrá descargar y subir ejercicios, corregir los ejercicios mandados y ver las estadísticas de sus ejercicios.

A.5.1 Mis ejercicios

En esta sección se va a explicar la vista que contiene la funcionalidad de descargar un ejercicio, subir un ejercicio y corregir el ejercicio. A continuación se explica un ejemplo dónde todos los ejercicios aún no han sido entregados y se va a realizar el proceso completo de descargar el ejercicio, subirlo y corregirlo.

En esta vista hay una tabla con todos los ejercicios asociados al alumno (ver figura A.9).

Figura A.9: *Descargar y subir un ejercicio*

Entre las diferentes columnas de la tabla, en esta sección se centrará en la columna "Acciones". En la figura A.9, en la columna "Acciones" se muestran dos botones. Uno azul claro que permite descargar el ejercicio y otro azul oscuro que permite subir el ejercicio.

A continuación, se hará click en el botón azul claro y se descargará el ejercicio. Una vez completado el ejercicio por el alumno, éste procederá a subirlo haciendo click en el botón azul oscuro. Al hacer click se muestra un popup y pinchando en la imagen se seleccionará el fichero del ordenador.

Una vez seleccionado, se mostrará un mensaje de confirmación. Al hacer click en aceptar, se enviará una petición al servidor para subir el ejercicio. De esta manera, se ha descargado el ejercicio, se ha realizado y se ha subido.

Al subirse el ejercicio, se actualiza la tabla de la vista y ahora la columna "Acciones" sólo tiene un botón de color azul oscuro que será el encargado de evaluar el ejercicio (ver figura A.10).

Al hacer click en dicho botón, se enviará una petición al servidor para evaluar el código.

Figura A.10: *Evaluando un ejercicio*

Una vez evaluado, se devolverá la nota al cliente y la nota será almacenada en la base de datos. Evaluado el ejercicio, se actualizará de nuevo la tabla de la vista dejando ahora la columna "Acciones" vacía ya que el ejercicio ya ha sido descargado, subido y evaluado (ver figura A.11).

A.5.2 Estadísticas

En esta sección se explican las estadísticas que se han metido en la aplicación. Como se puede observar en la figura A.12, existen dos partes.

Figura A.11: *Ejercicio evaluado*

Por un lado están las estadísticas asociadas al ejercicio seleccionado. En la parte de la

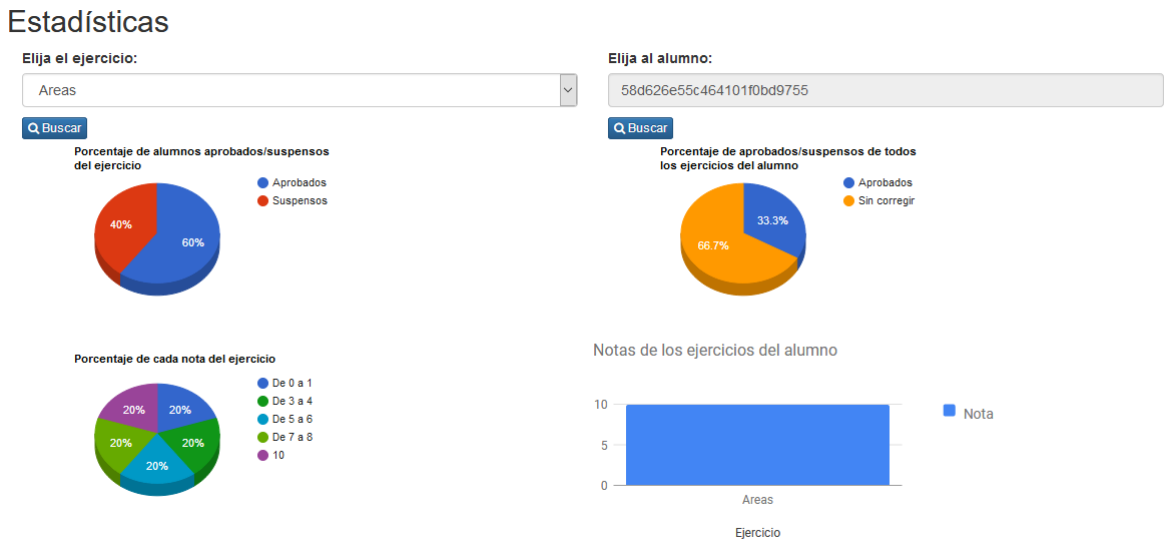


Figura A.12: *Interpretando las estadísticas*

izquierda, hay un desplegable con todos los ejercicios de la base de datos y se selecciona uno. Al hacer click en el botón de buscar de debajo del desplegable se realiza una petición al servidor para calcular las estadísticas. Estas estadísticas consisten en calcular el número de alumnos que han aprobado el ejercicio, el número de alumnos que lo han suspendido y el número de alumnos que aún no han entregado el ejercicio. En el segundo gráfico se calcula el número de alumnos que han obtenido un cero, un uno, un dos, un tres, etc calculando el porcentaje de cada nota.

La segunda parte hace referencia a las estadísticas del alumno correspondiente. Al hacer click en el botón de buscar se envía una petición al servidor para recuperar las estadísticas del alumno. Cuando se obtengan todos los ejercicios del alumno, se calculará el número de ejercicios aprobados, suspensos y sin corregir (estén entregados o no). Una vez obtenidos estos datos, se calcula el porcentaje de los ejercicios entregados, suspensos y sin corregir y se pintan en un gráfico circular. También se pensó en poner en un gráfico de barras todas las notas de los ejercicios entregados.

B Manual de despliegue

En este anexo se va a explicar cómo desplegar la aplicación web. Se va a desplegar la aplicación web en un servidor cuyo sistema operativo será Ubuntu Server.

B.1 Instalación de python

Como la aplicación web se ha desarrollado en python, es necesario instalar python. Actualmente, hay dos versiones de python diferentes que se usan con mucha frecuencia. Estas versiones son python2 y python3 que tienen algunas funciones incompatibles. De ahí que sea muy importante instalar la versión adecuada. Para ello, se ejecutará el siguiente comando:

```
~$ sudo apt-get install python3
```

Una vez instalado python3 se comprobará que se ha instalado correctamente con el siguiente comando:

```
~$ python3 --version
```

Si se ha instalado correctamente entoncés se verá en la consola la versión de python instalada. Si no se ha instalado correctamente, en la consola aparecerá un mensaje de error.

A continuación se instala pip:

```
~$ sudo apt-get install python3-pip
```

B.1.1 Instalación de Django

El siguiente paso es instalar el framework de python para simplificar la programación. Para ello se ejecuta el siguiente comando:

```
~$ pip3 install Django==1.10
```

Una vez instalado el framework django, se instala la librería pymongo que es la que se ha usado en la aplicación web para conectar con la base de datos MongoDB.

```
~$ pip3 install pymongo
```

B.2 Instalación de MongoDB

A continuación se instala la base de datos no SQL. Para instalar MongoDB es necesario poner los siguientes comandos:

```
~$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
 ↪  0C49F3730359A14518585931BC711F9BA15703C6
~$ echo "deb [ arch=amd64,arm64 ] http://repo.mongodb.org/apt/ubuntu
 ↪  xenial/mongodb-org/3.4 multiverse" | sudo tee
 ↪  /etc/apt/sources.list.d/mongodb-org-3.4.list
~$ sudo apt-get update
~$ sudo apt-get install -y mongodb-org
```

B.3 Instalación de los frameworks

En esta sección se va a instalar los frameworks AngularJS y Bootstrap. Para instalar angularJS y Bootstrap en una aplicación web se ponen las siguientes líneas en el fichero HTML principal:

```
<!DOCTYPE>
<html>
   <head>
      <meta charset="UTF-8">
      <title>Aplicación web</title>
      <!-- Instalación de AngularJS -->
      <script src =
 ↪  "https://ajax.googleapis.com/ajax/libs/angularjs/1.5.6/angular.min.js">
      </script>
      <!-- Instalación de Bootstrap -->
      <link href =
 ↪  "https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
      <script src =
 ↪  "https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js">
      </script>
   </head>
   <body>
   </body>
</html>
```

B.4 Instalación de las librerías JavaScript

Todas las librerías de javaScript se instalan de la misma manera. En las siguientes secciones se van a ir explicando cómo se instala cada libería en la aplicación web.

B.4.1 Instalación de jQuery

El siguiente código se pone en la etiqueta <head> del documento principal de la aplicación web:

```
<script src="js/jquery/jquery.js"></script>
```

B.4.2 Editor ACE

El siguiente código se pone en la etiqueta <head> del documento principal de la aplicación web:

```
<script src="js/ace/src-noconflict/ace.js" charset="utf-8"></script>
<script>
   var editor = ace.edit("editor");
   editor.setTheme("ace/theme/merbivore_soft");
   editor.session.setMode("ace/mode/python");
</script>
```

B.4.3 Google Charts

El siguiente código se pone en la etiqueta <head> del documento principal de la aplicación web:

```
<script src="js/google_charts/loader.js"></script>
```

B.4.4 jQuery validation

El siguiente código se pone en la etiqueta <head> del documento principal de la aplicación web:

```
<script src="js/jquery/jquery-validation.js"></script>
<script src="js/jquery/jquery-validation-additional-methods.js"></script>
```

B.4.5 jQuery confirm

El siguiente código se pone en la etiqueta <head> del documento principal de la aplicación web:

```
<script src="js/jquery/jquery-confirm.js"></script>
```

B.5 Desplegando la aplicación

Lo primero es arrancar la base de datos. Para ello en la consola se pondrá el siguiente comando:

```
~$ sudo service mongod start
```

En este punto, la base de datos queda accesible. Ahora se navega al directorio en otra consola que contiene los fuentes de la aplicación web y se ejecuta el siguiente comando:

```
~$ python3 manage.py runserver
```

Al ejecutar el comando anterior, se arranca el servidor de la aplicación web. La aplicación web quedará accesible desde la URL `http://localhost:8000/`