GRADO EN INGENIERÍA EN TECNOLOGÍAS DE LA TELECOMUNICACIÓN

Escuela de Ingeniería de Fuenlabrada

Curso académico 2024/2025

Trabajo Fin de Grado

Monitorización y gestión de usuarios y cursos en una plataforma web de programación de robots

Autora: Iraide Hoyas Puente
Tutor: Jose María Cañas Plaza
Cotutor: José Felipe Ortega Soto

Agradecimientos

Este Trabajo de Fin de Grado quiero dedicárselo a mi familia, en especial a mis padres, Manuel y Estrella, y a mi pareja, Alejandro. Gracias a su incondicional apoyo y ayuda este camino ha sido más llevadero.

En segundo lugar, a mis tutores, José María y José Felipe, y al equipo de Unibotics por la paciencia y ayuda en los momentos más complicados. Agradecerles también sus ánimos a lo largo de todo el proceso.

Por último, a mis amigos y a mis compañeros del grado por acompañarme durante esta etapa, haciendo que recuerde estos años como los mejores de mi vida académica por los buenos momentos vividos.

Resumen

En los grados orientados a la ciencia y la tecnología se ha vuelto de vital importancia el uso de la robótica. Debido a esto, las herramientas disponibles para la enseñanza de robótica a alumnos universitarios cada vez son más abundantes. Por esta razón, aparece Unibotics, una plataforma web sencilla que permite programar el funcionamiento de robots mediante una interfaz simple e intuitiva. Esta plataforma está orientada a la enseñanza de programación de robots para estudiantes universitarios de Ingeniería Robótica.

En este TFG se pretende dar un nuevo enfoque a la página web, incluyendo diversas funcionalidades que la permiten orientarse completamente hacia la educación. Se consigue incluyendo la opción de crear diferentes *Clases* y *Cursos* que posibilitan mejorar la gestión de los usuarios y los contenidos. Por un lado, con el concepto de *Clases* se puede agrupar a un conjunto de usuarios para incluir algunas funcionalidades para profesores. Por otro lado, con el concepto *Cursos* se permite agrupar los ejercicios por temática o dificultad. Además, se incluyen *Permisos* que facilitan el acceso a los usuarios a los contenidos de Unibotics.

Con el fin de adaptar aún más la plataforma web a la enseñanza, se incorporan dos funcionalidades para los docentes. La primera de ellas es la posibilidad de monitorizar la actividad de los alumnos y que pueda ser observada mediante gráficos. La segunda, es la opción de poder visualizar el código que desarrollan los estudiantes y poder probarlo en vivo.

Para la consecución de todos estos objetivos, se parte de la versión existente Unibotics 3.11. Además, se ha hecho uso de otras tecnologías como PostegreSQL para las novedades en la base de datos, de Dash para la visualización de información mediante gráficas o React para el diseño de la interfaz de usuario.

Índice general

Índice de figuras

Índice de códigos

Capítulo 1

Introducción

A lo largo de este capítulo se explica el contexto del proyecto, la robótica, así como sus posibles aplicaciones. Además, se añade una introducción a la robótica educativa y, en particular, a la plataforma de Unibotics. En ésta se van a introducir mejoras para la gestión de usuarios y contenidos e incluir algunas funcionalidades para profesores.

1.1. Robótica

La robótica se define como "la técnica que aplica la informática al diseño y empleo de aparatos que, en sustitución de personas, realizan operaciones o trabajos, por lo general en instalaciones industriales"[1]. Su principal objetivo es la construcción de máquinas que se encarguen de realizar muchas de las tareas que desarrolla el ser humano con mayor eficiencia y rapidez. Esta disciplina combina diferentes campos de conocimiento como la informática, la física, la electrónica y la ingeniería.

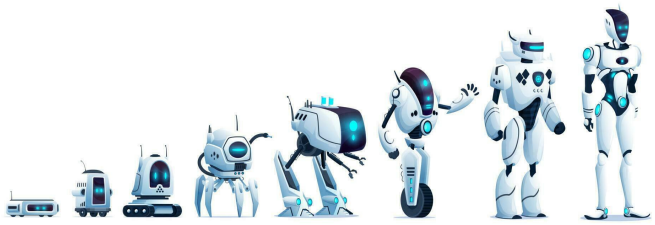

Figura 1.1: Evolución de la robótica

En cuanto a la morfología[2] que presentan los robots se puede hacer la siguiente clasificación.

[1] [1]https://www.rae.es/
[2] [2]https://concepto.de/robotica/

- **Poliarticulados:** utilizados en tareas industriales gracias a su capacidad de realizar movimientos complejos por incorporar gran cantidad de piezas móviles.

Figura 1.2: Robot poliarticulado

- **Móviles:** son de tipo automotor o con ruedas y se emplean en tareas de exploración, logística y transporte.

Figura 1.3: Robot móvil

- **Zoomórficos:** su aspecto es similar al de algunos animales. Son útiles para trabajos de rescate por su facilidad para adecuarse a entornos complejos o de difícil acceso.

Figura 1.4: Robot zoomórfico de perro

Figura 1.5: Robot zoomórfico de canguro

- **Antropomórficos:** posee las características corporales de las personas. Gracias a su estructura se utilizan en áreas como la medicina ayudando en cirugías, la industria manipulando materiales en cadenas de montaje, y la defensa realizando tareas para garantizar la seguridad de los ciudadanos.

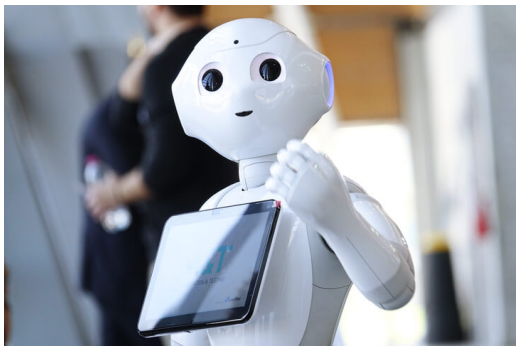

Figura 1.6: Robot antropomórfico con pantalla

Figura 1.7: Robot antropomórfico

Muchos de los robots de la actualidad combinan varias de las estructuras mencionadas anteriormente. A este tipo se les denomina robots híbridos.

La robótica presenta múltiples aplicaciones y ventajas para la sociedad, algunas de ellas se exponen a continuación.

- **Mayor productividad:** su uso en diferentes industrias permite realizar tareas repetitivas de forma más rápida y evita la realización de actividades peligrosas para los humanos.

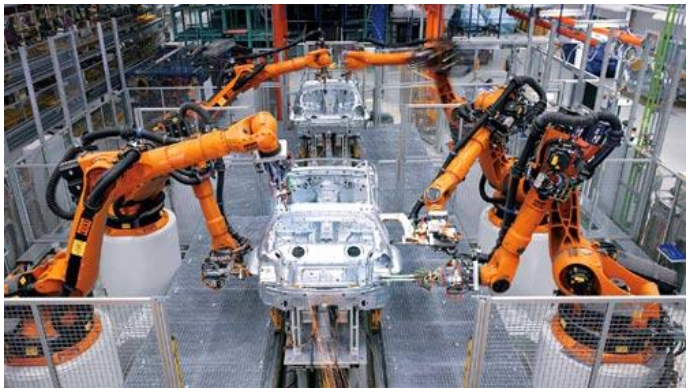

Figura 1.8: Robots en la industria automovilística

- **Automatizar tareas monótonas:** con la aparición de robots capaces de colaborar en tareas como la limpieza del hogar o la conducción autónoma de vehículos.

Figura 1.9: Aspirador autónomo

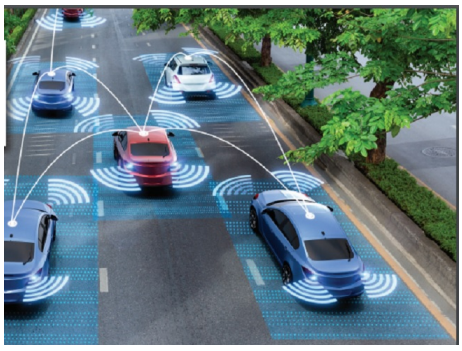

Figura 1.10: Coches autónomos

- **Ayudar en campos como la medicina:** actualmente existen diferentes robots que aportan mayor precisión y destreza en cirugías y proporcionan una visión clara de la anatomía del paciente.

Figura 1.11: Robot Da Vinci en medicina

- **Ingresar en lugares de difícil acceso y desempeño de actividades peligrosas:** los drones y robots militares permiten la inspección de zonas de acceso complicado para los humanos Además, realizan actividades que ponen en peligro la integridad de las personas como la desactivación de bombas o entrar en áreas contaminadas.

Figura 1.12: Dron de inspección

Figura 1.13: Robot de uso militar

1.1.1. Middleware robóticos

Los robots están formados por diferentes componentes tanto *hardware* como *software*. Por un lado, el *hardware* son los componentes físicos como sensores (cámaras o radares), actuadores (motores) o procesadores (GPUs o CPUs), que permiten realizar las diferentes tareas. Por otro lado, el *software* son los diferentes programas y sistemas que permiten controlar los componentes físicos. Con el fin de integrar todos estos componentes tanto *hardware* como *software* aparece el concepto de *middleware*.

El *middleware*[3] permite integrar los componentes físicos y lógicos en los sistemas robóticos facilitando la comunicación y coordinación entre estos componentes. En la actualidad, existen múltiples *middlewares* robóticos a pesar de que algunos de ellos han dejado de utilizarse. A continuación se hace referencia a algunos de ellos[4].

- **ROS[5] (Robot Operating System):** actualmente es el middleware robótico más conocido y utilizado, cuenta con un conjunto de herramientas para el desarrollo de software de robots. Sin ser un sistema operativo, ROS trabaja como si lo fuera, ya que permite abstracción del hardware, control de dispositivos de bajo nivel, paso de mensajes entre procesos y mantenimiento de paquetes. Se basa en una arquitectura de grafos, ya que los nodos se encargan del procesamiento, es decir, reciben, mandan y multiplexan los mensajes entre componentes.

[3] [3]`https://bambu-mobile.com/que-es-el-middleware-en-arquitectura-del-software/`
[4] [4]`https://es.wikibrief.org/wiki/Robotics_middleware`
[5] [5]`https://openwebinars.net/blog/que-es-ros/`

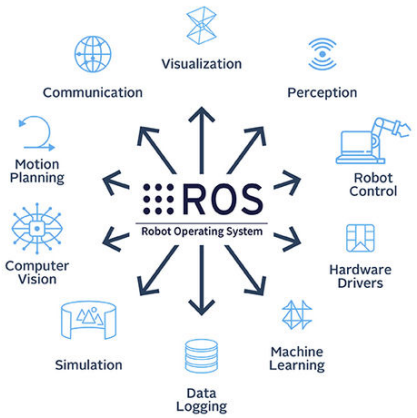

Figura 1.14: Robot Operating System

ROS[6] es extensible gracias a la variedad de herramientas con las que cuenta como RVIZ, Webiz, rqt-plot o rqt-graph. RVIZ permite visualizar en 3D información de sensores, modelos de robots y otros tipos de datos. Webiz es una herramienta que permite ver datos y archivos de ROS basada en navegador. Rqt-plot permite ver datos escalares publicados en temas de ROS. Rqt-graph es una herramienta que muestra un gráfico de los procesos que se ejecutan en ROS y sus conexiones.

- **YARP[7](Yet Another Robot Platform):** permitía la comunicación entre los diferentes módulos y destacaba por su flexibilidad a la hora de trabajar con diferentes dispositivos, ya que permitía operar con diversos protocolos de comunicación. Además, era de código abierto.

Figura 1.15: Yet Another Robot Platform

- **MRDS[8] (Microsoft Robotics Developer Studio):** al igual que los anteriores, permitía la comunicación y coordinación entre los componentes de los sistemas robóticos. También contaba con un entorno de simulación con una interfaz intuitiva para todo tipo de robots.

[6][6]https://www.ros.org/
[7][7]https://www.yarp.it/
[8][8]https://learn.microsoft.com/en-us/archive/msdn-magazine/2008/june/robotics-simulating-the-world-with-microsoft-robotics-studio

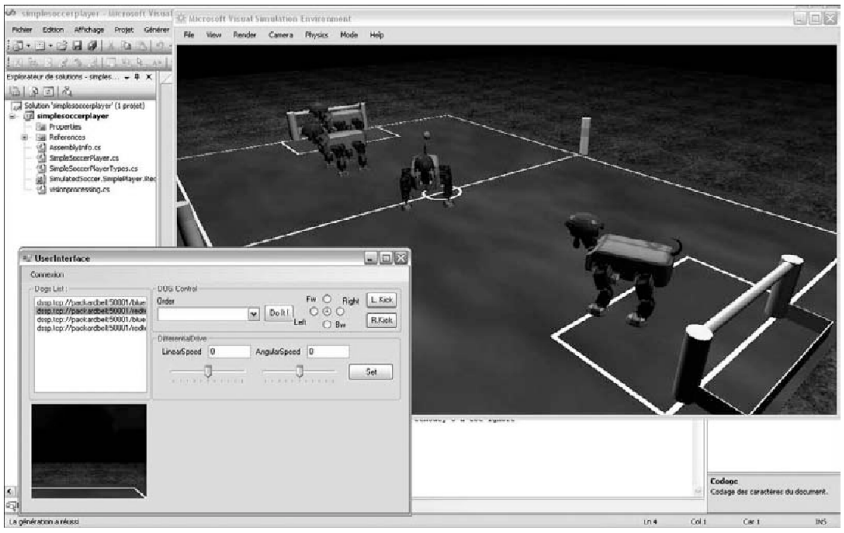

Figura 1.16: Microsoft Robotics Developer Studio

1.1.2. Simuladores robóticos

Para cualquier ámbito en el que se diseña y se hace uso de cualquier tipo de robot es fundamental el uso de herramientas que permitan la simulación del funcionamiento de los robots. Los simuladores son necesarios para poder experimentar y probar virtualmente los sistemas robóticos antes de ser utilizados. Algunos de los simuladores más populares son los siguientes.

- **Gazebo**[9]**:** es uno de los más conocidos junto a su funcionamiento con ROS. Permite una simulación dinámica gracias a sus avanzados gráficos en 3D, incluye numerosos tipos de robots con sensores y sonido con los que hacer la simulación y cuenta con distintas herramientas para personalizar el código que indica al robot lo que debe hacer.

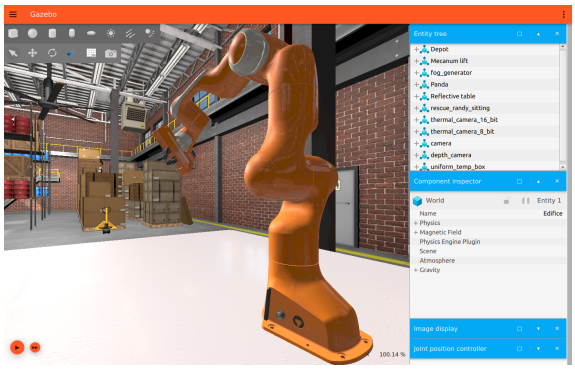

Figura 1.17: Simulador Gazebo

- **CoppeliaSim**[10]**:** su arquitectura permite controlar individualmente cada objeto mediante scripts, complementos o soluciones personalizadas. Incluye mo-

[9] [9]`https://gazebosim.org/`
[10] [10]`https://www.coppeliarobotics.com/`

tores dinámicos, planifica trayectorias, movimientos, colisiones y distancias, y permite la simulación con sensores. Este simulador permite la validación de sistemas robóticos complejos mediante prototipos y diferentes algoritmos.

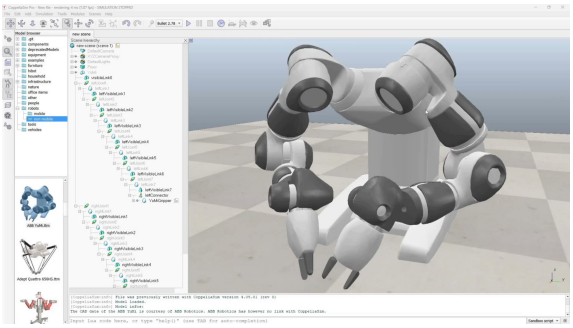

Figura 1.18: Simulador CoppeliaSim

- **WeBots**[11]**:** es una aplicación de escritorio de código abierto que permite modelar y programar robots para realizar una simulación. Se emplea especialmente en industria, educación e investigación. Posibilita crear prototipos de forma rápida y configurar entornos interactivos.

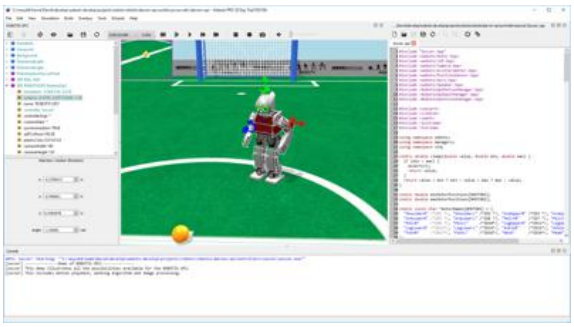

Figura 1.19: Simulador WeBots

- **MATLAB Robotics System Toolbox**[12]**:** al igual que las anteriores, ofrece herramientas y algoritmos para diseñar, simular y desplegar robots móviles y manipuladores así como escenarios de prueba. Además, permite crear robots basados en datos matemáticos y físicos de forma precisa y se puede integrar con ROS.

[11][11]`https://cyberbotics.com/`
[12][12]`https://es.mathworks.com/products/robotics.html`

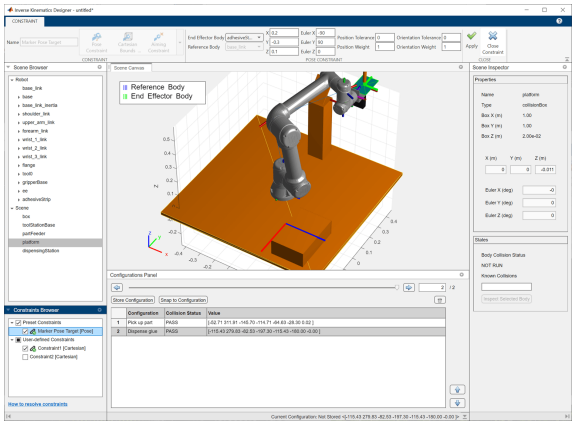

Figura 1.20: Simulador Robotics System Toolbox

1.2. Robótica educativa en ingeniería

La robótica educativa es una rama de la robótica que se integra en el ámbito académico, centrándose en el diseño, análisis, utilización y operación de robots con propósitos pedagógicos. Esta disciplina es aplicable en todos los niveles educativos, desde la educación infantil hasta los estudios de posgrado. Además, se emplea como una herramienta para enriquecer y facilitar el aprendizaje en diversas áreas, como la programación, la inteligencia artificial y la ingeniería robótica.

Otro aspecto destacado es el enfoque en la inclusión y la accesibilidad, que ha llevado al desarrollo de programas adaptados para diversos niveles de habilidad y contextos educativos. La adopción de plataformas online y simuladores durante la pandemia ha reforzado aún más este campo, permitiendo que la robótica siga siendo una herramienta clave en la educación, incluso en entornos remotos.

1.2.1. Plataformas de educación en robótica para ingeniería

Actualmente existen algunas plataformas que facilitan la enseñanza de robótica a los estudiantes en niveles superiores. Para su uso es necesario tener conocimientos previos de robótica y programación. A continuación se exponen algunas de ellas.

- **The Construct**[13]**:** es una plataforma que utiliza ROS, diseñada para el aprendizaje de robótica y ofrece un plan de estudios junto a una gran variedad de cursos. No requiere de instalaciones previas y permite experimentar a los usuarios en entornos virtuales sin la necesidad de tener un componente físico.

[13][13]`https://www.theconstruct.ai/home/`

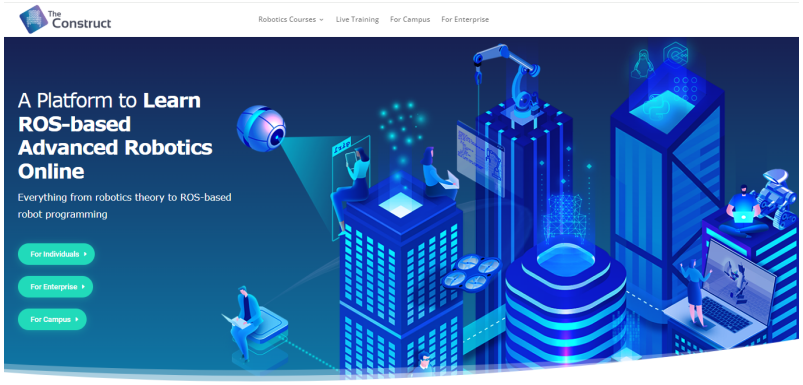

Figura 1.21: Página oficial de The Construct

- **Riders.ai**[14]**:** esta plataforma se centra en la enseñanza de robótica e inteligencia artificial a través de simulaciones y competiciones online. Permite la creación, programación y control de robots mediante simulaciones realistas. Se diferencia de otras plataformas por organizar competiciones de simulaciones entre los usuarios fomentando la motivación y la aplicación de los conocimientos. En relación a la enseñanza proporciona tutoriales y ejemplos a los usuarios.

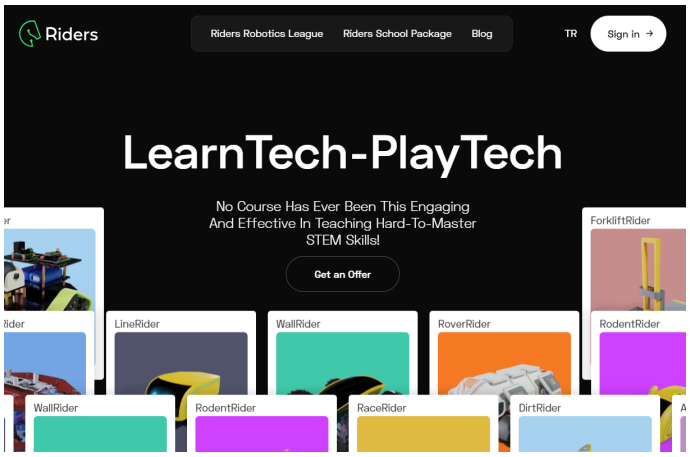

Figura 1.22: Página oficial de Riders.ai

- **Asimovo**[15]**:** es una plataforma enfocada en la enseñanza de robótica mediante la programación de robots y su construcción. Sus kits de construcción contienen sensores, controladores y motores que permiten un aprendizaje práctico. También integra herramientas para programar el comportamiento de los robots y probar su funcionamiento.

[14][14]`https://riders.ai/`
[15][15]`https://asimovo.com/`

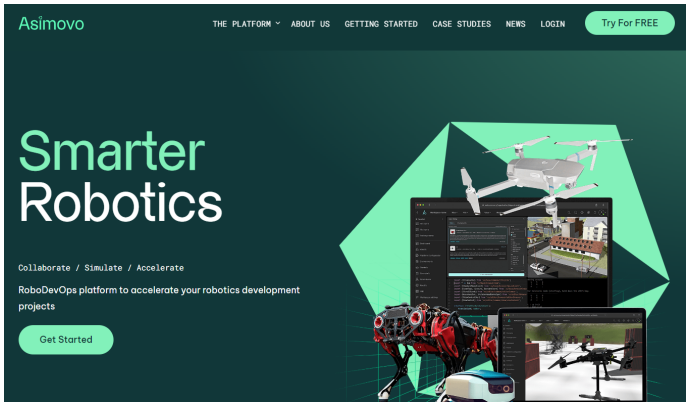

Figura 1.23: Página oficial de Asimovo

- **Unibotics**[16] **:** este TFG se enfoca en esta plataforma web que se explorará en detalle en la siguiente sección y a lo largo del proyecto.

1.2.2. Tecnologías web

La mayoría de plataformas de robótica educativa mencionadas hacen uso de las tecnologías web más recientes. Esto se debe a su compatibilidad con cualquier sistema operativo y navegador facilitando el acceso a todos los contenidos educativos disponibles.

El concepto de tecnología web[17] se define como el conjunto de herramientas que permiten la comunicación entre computadoras mediante el uso de Internet. Esto hace que el usuario únicamente necesite un dispositivo con acceso a Internet, sin la necesidad de realizar instalaciones en el ordenador del usuario. Las tecnologías web han sufrido una gran transformación a lo largo de los años, desde páginas web con un contenido básico y limitado, hasta ofrecer funcionalidades avanzadas hoy en día.

Para el uso de las tecnologías web son necesarios dos componentes imprescindibles, el cliente y el servidor. Desde el lado del servidor, se gestionan las solicitudes enviadas por parte del cliente. Cuando el servidor recibe una solicitud, accede a bases de datos, genera páginas dinámicas o ejecuta acciones específicas para enviar una respuesta al cliente. Desde el lado del cliente, cuando un usuario hace una petición, el navegador recibe los archivos del servidor y los procesa para presentar la respuesta en la interfaz de usuario. Existen tanto tecnologías *backend* que trabajan del lado del servidor como tecnologías *frontend* que trabajan del lado del cliente[18].

- **Tecncologías *backend*:** Django, Python, PHP, bases de datos...
- **Tecncologías *frontend*:** HTML, CSS, JavaScript...

[16][16]https://unibotics.org/
[17][17]https://estudyando.com/que-es-la-tecnologia-web-definicion-y-tendencias/
[18][18]https://www.proun.es/blog/tecnologias-web-actuales/

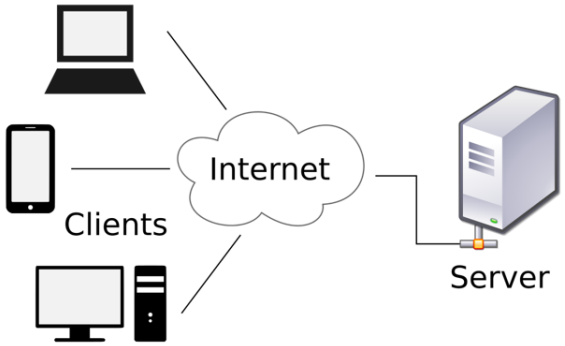

Figura 1.24: Cliente-Servidor

Las características más relevantes de las tecnologías web son las siguientes.

- **Alcance global:** es posible acceder a cualquier tecnología web desde cualquier dispositivo que cuente con conexión a internet.
- **Escalabilidad:** las aplicaciones web tienen la capacidad de crecer o decrecer en función de los usuarios que haya sin que afecte al rendimiento de la infraestructura.
- **Compatibilidad:** la tecnologías web pueden usarse en diversos navegadores y sistemas operativos.
- **Facilidad de uso:** los usuarios pueden acceder fácilmente a las aplicaciones web a través de un navegador sin necesidad de instalaciones ni configuraciones complejas.

En la actualidad, el uso de tecnologías web es muy común y se pueden encontrar ejemplos de su uso en diversos ámbitos como por ejemplo en los medios de comunicación , que además de contar con sus versiones tradicionales como la televisión, la radio o la prensa escrita, cuentan todos ellos con página web como es el caso de *El Mundo*.

Figura 1.25: Página web de *El Mundo*

Por otro lado, se puede ver la utilización de la tecnología web en páginas que brindan al usuario la posibilidad de acceder a contenido multimedia en cualquier dispositivo y en cualquier momento como es el caso de *Netflix*.

Figura 1.26: Página web de *Netflix*

Además, las tecnologías web aparecen en *websales*, diseñados para la compra y venta de productos y servicios online como es el caso de *Amazon*.

Figura 1.27: Página web de *Amazon*

1.3. Robótica educativa con Unibotics

La aplicación sobre la que trabajaremos en este TFG es la plataforma web Unibotics[19]. Unibotics es una web orientada a la enseñanza de robótica para alumnos universitarios, que permite programar robots en Python desde el propio navegador y ejecutar el código realizado para probar su funcionamiento. Todas las dependencias necesarias para hacer uso de Unibotics ya están preinstaladas, lo que evita tener que hacer instalaciones en el ordenador local.

Esta plataforma hace uso de ROS y del simulador Gazebo. Además, ofrece una gran variedad de ejercicios que permiten el aprendizaje de diversos robots y simular su funcionamiento. Unibotics también cuenta con un foro donde preguntar dudas o hablar de algunos temas en relación a la aplicación con otros usuarios.

Actualmente, esta plataforma se está usando en la Universidad Rey Juan Carlos, en la Universidad Complutense de Madrid, en el Politecnico Colombiano Jaime Isaza Cadavid y en la Baden-Wuerttemberg Cooperative State University Karlsruhe (Alemania).

[19][16]`https://unibotics.org/`

Figura 1.28: Página oficial de Unibotics

Unibotics presenta una gran variedad de ventajas frente a otros softwares robóticos.

- Es abierto, lo que permite su uso a cualquier persona que se registre en la plataforma de forma gratuita.
- Está disponible para cualquier sistema operativo.
- Sigue en expansión por lo que sigue habiendo nuevas actualizaciones con nuevos ejercicios y funcionalidades.
- Es escalable, es decir, puede aumentar el número de usuarios utilizando la aplicación sin afectar el rendimiento de Unibotics.

Capítulo 2

Objetivos y Metodología

En este capítulo se desglosan tanto los objetivos marcados al inicio del proyecto, como la metodología y el plan de trabajo desarrollado para la consecución de dichos objetivos.

2.1. Objetivos

El trabajo de fin grado consta de dos objetivos principales. Por un lado, mejorar la gestión de usuarios y contenidos de Unibotics y, por otro, implementar nuevas funcionalidades exclusivamente para profesores.

En cuanto al primer objetivo pueden diferenciarse subobjetivos más específicos. Algunos de ellos son los siguientes:

- **Crear nuevos roles:** implementar nuevos roles permite la diferenciación de usuarios entre alumnos y profesores. Además, facilita la agrupación de los usuarios en clases que estarán compuestas tanto por alumnos como por profesores y en función del rol tendrán diferentes funcionalidades dentro de la aplicación.
- **Creación de clases y cursos:** implementar clases y cursos en la plataforma permite agrupar tanto a los usuarios como a los ejercicios. Las clases están formadas por usuarios, la mayoría tendrán rol de alumno y habrá uno o varios con rol de profesor. Los ejercicios pueden pertenecer a un curso en función de la temática o dificultad.
- **Inclusión de permisos:** permite dar acceso a unos cursos o ejercicios en función de las clases a las que pertenece un usuario. De esta forma, no todas las clases y usuarios tienen permiso para ver todos los contenidos de la plataforma.
- **Crear una nueva página de entrada para los usuarios:** con la aparición de los cursos es necesario crear una nueva página para los usuarios en la que se les muestren los cursos y unidades a los que tiene acceso según sus permisos. Además, se necesita crear las páginas de presentación de cada curso con los ejercicios que lo componen, una breve descripción o un vídeo descriptivo.

Con respecto al segundo objetivo también encontramos subobjetivos específicos, como por ejemplo:

- **Visualización de los datos de actividad de los alumnos:** añadir una opción en la que los profesores puedan visualizar la actividad de sus clases y alumnos. Esto es útil para que puedan sacar estadísticas tanto a nivel grupal como individual y observar el tiempo dedicado a la asignatura.
- **Visualización y ejecución del código de los alumnos:** desarrollar las extensiones de software que permitan que los profesores puedan ver y ejecutar el código de sus alumnos. De esta forma, pueden calificar el progreso y funcionamiento de los ejercicios que han realizado sus alumnos.

2.2. Metodología

Para la realización de este TFG se ha empleado la metodología *scrum con sprints* que se caracteriza por dividir el proyecto en tareas sencillas de corta duración con el fin de centrarse en el objetivo fijado para cada *sprint*. Se trata de una metodología ágil y que permite recibir realimentación continuamente. Para llevar a cabo esta metodología se ha seguido la siguiente planificación.

En primer lugar, se han realizado reuniones semanales para revisar el trabajo realizado, repasar los objetivos y adaptarlos según se iba avanzando y resolver las dudas que podían surgir. También disponía de una plataforma, Slack, para mantener contacto directo tanto con los tutores como con otros desarrolladores y así poder resolver los problemas de forma más rápida. Además, se ha escrito un blog[1] que se iba actualizando cada semana para que quedase reflejado el trabajo realizado de inicio a fin.

En segundo lugar, la metodología empleada para alcanzar los objetivos fue desarrollar cada funcionalidad individualmente y empezar nuevas funcionalidades una vez acabada la anterior. Cada funcionalidad se llevaba a cabo primeramente en un entorno local D1. D1 es un entorno de desarrollo en el que se instala la versión más reciente de la plataforma en el ordenador local. En este entorno cada desarrollador puede realizar todos los cambios necesarios y así poder probarlos sin que afecte a la plataforma web real. Una vez que funcionaban mis cambios en este entorno local, comenzaba la siguiente funcionalidad.

Tras tener todas las nuevas funcionalidades implementadas en mi entorno local D1 funcionando correctamente, abrí varias incidencias en GitHub para cada parte nueva con sus respectivas ramas, a las que subí los cambios que afectaban a cada parte. Posteriormente, los desarrolladores con más experiencia revisan las incidencias y las ramas. Una vez revisadas, integran los cambios en un entorno de test D2. D2 es otro entorno en el que se añaden los cambios que se van subiendo a GitHub.

[1] [19] `https://roboticslaburjc.github.io/2023-tfg-iraide-hoyas/`

Este entorno se utiliza para realizar las pruebas necesarias sin que se vea afectada la plataforma real e introducir modificaciones en el caso de que algo falle.

Finalmente, una vez pasadas las pruebas y funcionar todo correctamente, mis nuevas funcionalidades se integraron en el entorno D3. D3 es el entorno de producción, es decir, los cambios que se añaden a este entorno son los que aparecen en la plataforma real donde los usuarios pueden visualizar y disfrutar de las nuevas actualizaciones.

2.3. Plan de trabajo

El plan de trabajo que se ha seguido durante el desarrollo del Trabajo de Fin de Grado se ha dividido en varias fases ordenadas cronológicamente desde el inicio.

- **Investigación (primera fase):** la duración de esta primera fase fue de dos semanas y consistió en un primer contacto con la documentación y la plataforma de Unibotics para conocer mejor la aplicación, accediendo a ella a nivel usuario con el fin de probar su funcionamiento. Además, se hizo una pequeña investigación de las primeras herramientas que se iban a utilizar posteriormente. Algunas de éstas fueron Django, PostgreSQL y la librería Dash de Python. Por otro lado, se creó el blog semanal con sus primeras publicaciones.

- **Prototipo (segunda fase):** durante aproximadamente tres meses se desarrolló un prototipo antes de comenzar a manejar el código real de la aplicación. Este prototipo consistió en crear un servidor Django con una base de datos PostgreSQL, en la que hubo que crear tablas para usuarios con diferentes roles, ejercicios y registros de actividad. Una vez hecho esto, había que crear gráficos con Dash que permitieran ver la actividad de los usuarios en función de su rol. El objetivo del prototipo era conocer en profundidad las diferentes tecnologías.

- **Desarrollo (tercera fase):** esta fase duró alrededor de tres meses, primeramente se montó un entorno local para hacer el despliegue D1. Se trata de una copia de la aplicación real para poder hacer y probar los cambios sin que se vean afectados otros desarrolladores ni la aplicación en producción. En este despliegue se fueron desarrollando secuencialmente cada una de las funcionalidades para alcanzar los objetivos marcados.

- **Pruebas (cuarta fase):** esta última fase se realizó una vez que todas las nuevas funciones estaban terminadas y probadas correctamente. Hubo que subir los cambios a nuevas ramas del repositorio de GitHub para que otros desarrolladores con experiencia pudieran subir los cambios a un entorno de pruebas D2. En este entorno se prueba durante un tiempo el funcionamiento de la aplicación con los cambios para detectar posibles errores y modificarlos. Una vez pasadas las pruebas, los cambios se integran en el entorno D3 de producción.

Capítulo 3

Herramientas y tecnologías utilizadas

Durante la realización del proyecto se han utilizado múltiples herramientas y tecnologías que se explican en detalle en las siguientes secciones.

3.1. Python

Python[1] es un lenguaje de programación de alto nivel, interpretado, orientado a objetos y de fácil aprendizaje gracias a su sintaxis simple y clara. Sus usos más comunes son el desarrollo web empleando algunos de sus frameworks, la automatización de tareas o creación de prototipos por su rapidez y sencillez. Algunas de las características mas relevantes de este lenguaje son:

- **Multiparadigma:** admite diferentes métodos o estilos de programación en función de las necesidades del programador para resolver cada problema.
- **Interpretado:** no requiere compilación previa, ejecuta cada línea. Esto facilita la prueba y depuración del código.
- **Biblioteca estándar:** consta de una gran biblioteca que proporciona diferentes paquetes y módulos.
- **Extensible:** se puede combinar con otros lenguajes de programación para aumentar las capacidades de Python.
- **Multiplataforma:** es compatible con los sistemas operativos más comunes.

[1] [20]`https://docs.python.org/`

Figura 3.1: Python

En este TFG se ha empleado esta tecnología para el desarrollo del backend. Dicho desarrollo ha consistido en extraer datos de la base de datos y su posterior filtrado para ser devueltos al frontend, donde se maneja la visualización de los mismos. La versión utilizada ha sido Python 3.8.

3.2. Django

Django[2] es un framework de alto nivel para Python que permite el desarrollo de aplicaciones web de forma rápida y eficaz. Además, ofrece plena seguridad a los usuarios y una gran versatilidad y escalabilidad. Su estructura se basa en el patrón Model-View-Template[3]:

- **Model:** se encarga de la base de datos, creando las tablas, su estructuración y la relación que existe entre ellas.
- **View:** es la parte lógica de la aplicación, se encarga de determinar los datos que recibe el usuario y la forma de procesarlos.
- **Template:** son las plantillas que reciben los datos enviados por la vista y se encarga de mostrarlos visualmente al usuario.

Figura 3.2: Django

Figura 3.3: MVT

En este proyecto se ha utilizado Django como servidor web de la aplicación, ya que es el servidor de la plataforma Unibotics. Se hace uso de la estructura explicada anteriormente para el manejo y visualización de los datos. La versión utilizada ha sido Django 4.2.

[2] [21] https://www.djangoproject.com/

[3] [22] https://www.adrformacion.com/knowledge/programacion/_como_es_la_estructura_de_django_.html

3.3. Dash

Dash[4] es un framework de Python para construir aplicaciones de visualización y análisis de datos mediante la creación de diferentes tipos de paneles y gráficos. Una de sus ventajas es la integración con la biblioteca Plotly, ya que ofrece gráficos dinámicos e interactivos que se actualizan automáticamente al hacer uso de botones o menús desplegables donde el usuario puede seleccionar el filtrado de los datos.

Figura 3.4: Dash

Esta herramienta ha sido muy útil en el desarrollo de este TFG para mostrar de forma gráfica e interactiva la actividad de los diferentes usuarios en la plataforma mediante gráficos de puntos e histogramas. La versión utilizada ha sido Dash 2.15.0.

3.4. Javascript

JavaScript[5] es un lenguaje de programación interpretado orientado a objetos basado en prototipos. También se conoce por ser un lenguaje de scripting para páginas web orientado a la interacción con el usuario para que su uso sea más dinámico. Algunas de sus características mas destacables son:

- **Dinamismo y flexibilidad:** el tipado es débil, lo que permite no asignar un tipo específico a una variable, ya que el tipo va ligado al valor asignado a dicha variable. Esto hace que sea muy flexible aunque puede dar lugar a errores.
- **Asincronía:** permite manejar operaciones asíncronas y gracias a ello se evitan bloqueos en la aplicación.
- **Variedad de bibliotecas:** consta de un gran número de bibliotecas y frameworks que facilitan el desarrollo de aplicaciones con complejidad.

[4] [23] `https://plotly.com/dash/`
[5] [24] `https://developer.mozilla.org/es/docs/Web/JavaScript`

Figura 3.5: JavaScript

La versión utilizada para JavaScript ha sido Node.js 17.

3.4.1. React

React[6] es una librería de JavaScript que permite crear interfaces de usuario a partir de componentes, es decir, piezas individuales que se van anidando y combinando. Además, permite la reutilización de los componentes en diferentes partes. De esta forma, se consigue facilitar la construcción de interfaces complejas. Algunas de sus características, además de las ya mencionadas, son:

- **Unidireccional:** los datos siguen una única dirección, de los componentes padres a los componentes hijos.
- **Virtual DOM:** hace uso de DOM virtual para actualizar de forma eficiente cuando un componente cambia.
- **JSX:** es una extensión que permite incluir HTML en el script.
- **Universal:** es posible usarlo tanto en el cliente como en el servidor. Además, tiene una versión para el desarrollo de aplicaciones móviles.

Figura 3.6: React

La versión utilizada ha sido React 18.2.0.

En este proyecto se ha hecho uso de JavaScript junto a la librería React para crear la parte dinámica de la interfaz de Unibotics, con la que interactúa el usuario a la hora de visualizar las diferentes aplicaciones y ejercicios de la plataforma.

[6] [25] `https://es.react.dev/`

3.5. CSS

CSS[7] (hojas de estilo en cascada) es un lenguaje de estilos para documentos HTML y XML. Se utiliza para el diseño visual de las páginas web consiguiendo un aspecto más estético y atractivo. De esta forma, se pretende separar el contenido de la presentación, siendo mas fácil modificar en un futuro el diseño, colores o formas de las páginas. Las características principales son:

- **Estilos en cascada:** los estilos en los diferentes elementos se aplican en función de una jerarquía, las reglas que son mas específicas tienen prioridad frente a aquellas que son genéricas.
- **Modelos de caja:** se aplica un modelo en el que cada elemento de la página es tratado como una caja y posee unas propiedades como el contenido, relleno, margen...
- **Posicionamiento:** existen diferentes técnicas para controlar las disposición de los elementos en la página.

Figura 3.7: CSS

Las hojas de estilo en cascada se han empleado en este TFG para el diseño, estilo y disposición de los elementos que componen la plataforma de Unibotics.

3.6. HTML

HTML[8] es un lenguaje de marcado para organizar el contenido de una página mediante el uso de etiquetas. Estas etiquetas están predefinidas y cada una de ellas tiene unas características y una finalidad específica. Además, consta de dos secciones claramente definidas. Por un lado la cabecera, que contiene metadatos, scripts y hojas de estilo. Por otro lado el cuerpo del documento, que incluye el contenido visible. Este lenguaje es universal, es decir, es compatible con todos los dispositivos y navegadores lo que permite plena accesibilidad desde cualquier explorador.

[7] [26] `https://developer.mozilla.org/es/docs/Web/CSS`
[8] [27] `https://developer.mozilla.org/es/docs/Web/HTML`

Figura 3.8: HTML5

En este proyecto se hace uso de HTML para integrar los componentes de React explicados anteriormente. La versión utilizada ha sido HTML5.

3.7. PostgreSQL

PostegreSQL[9] es un sistema de base de datos relacional, orientada a objetos y de código abierto. Su función principal es almacenar y recuperar datos de forma segura. Sus características más relevantes son:

- **Código abierto:** es gratuito y puede ser usado, modificado y distribuido por cualquiera. Además, existe una comunidad activa que se encarga de su mantenimiento.
- **Confiabilidad:** gracias a sus mecanismos es posible recuperar datos ante posibles desastres.
- **Seguridad:** incluye sistemas de autenticación, un robusto control de acceso y cifrado de datos.
- **Extensibilidad:** permite personalizar el tipo de datos y funciones así como el uso de diferentes lenguajes mediantes diferentes extensiones.
- **Concurrencia:** permite que diferentes procesos accedan y modifiquen los datos de forma simultánea.
- **Alta compatibilidad:** es compatible con múltiples sistemas operativos a diferencia de otros sistemas de base de datos.

Figura 3.9: PostgreSQL

[9][28]https://www.postgresql.org/about/

El uso de esta herramienta en el proyecto es de gran importancia, ya que la base de datos de la plataforma Unibotics es PostgreSQL. Gracias a ella se almacenan todos los datos de la plataforma como usuarios, ejercicios o registros de actividad. La versión utilizada ha sido PostgreSQL 13.11.

3.8. Docker

Docker[10] es una tecnología de contenedores que permite crear, desplegar y gestionar aplicaciones. Un contenedor es una unidad estándar de software que encapsula el código y las dependencias de una aplicación para ejecutarla de forma confiable y rápida. Además, existen las imágenes de Docker, que son paquetes independientes que incluyen todo lo necesario para ejecutar una aplicación y en tiempo de ejecución se convierten en contenedores.

Figura 3.10: Docker

Los contenedores tienen ventajas similares a las máquinas virtuales, sin embargo, su funcionamiento es distinto. Por un lado, los contenedores pueden ejecutarse varios simultáneamente en una misma máquina compartiendo el sistema operativo y ejecutándose de forma independiente. Además, ocupan menos espacio. Por otro lado, las máquinas virtuales permiten ejecutar varias simultáneamente con un solo servidor físico pero cada una de ellas tiene su propio sistema operativo, aplicaciones, binarios y otras bibliotecas lo que repercute en una necesidad de tener disponible mucho espacio. Su uso conjunto aporta flexibilidad en la implementación y administración de los programas.

[10] [29] `https://www.docker.com/resources/what-container/`

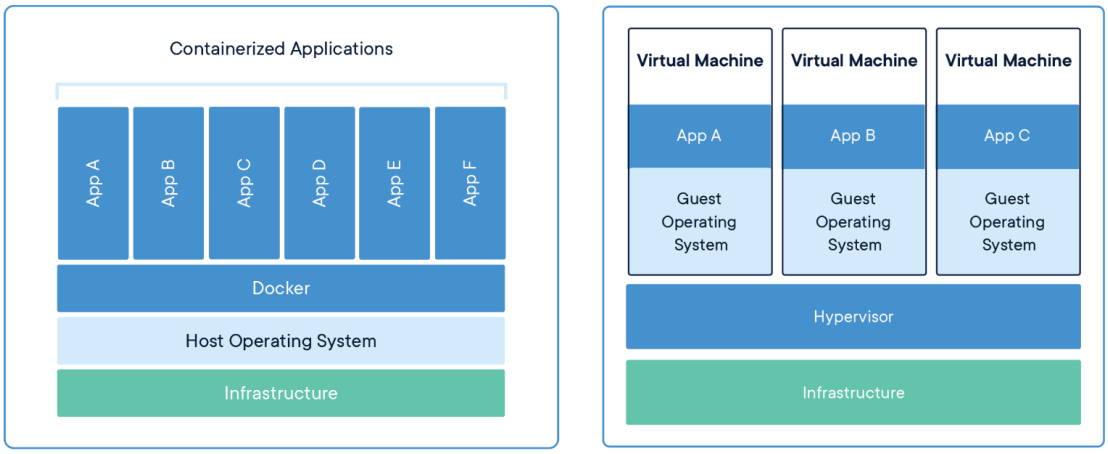

Figura 3.11: Contenedores vs Máquinas virtuales

Las imágenes de Docker se utilizan en Unibotics para crear RADI (Robotics Academy Docker Image) que incluye todas las dependencias necesarias para los ejercicios que proporciona la plataforma y se puede instalar fácilmente en cualquier sistema operativo habitual. También se usan los contenedores de Docker para ROS (Robotics Operative System). La versión utilizada ha sido Docker 5.0.2.

3.9. GitHub

GitHub[11] es una herramienta que permite trabajar conjuntamente con otros usuarios, ya que sirve para almacenar y compartir código en repositorios compartidos mediante el sistema de control de versiones de Git. Además, es útil para hacer un seguimiento del código a lo largo del tiempo y para que los usuarios puedan revisar el código de otros y proponer mejoras antes de integrar los cambios. A continuación se explican las funcionalidades principales de Git y GitHub:

- **Repositorios:** lugar en el que se almacena todo el contenido de un proyecto. Pueden ser públicos o privados en función de si es accesible para todo el mundo o solo para usuarios autorizados.
- **Commits:** sirve para guardar los cambios y queda reflejado qué se ha cambiado, cúando y quién lo ha hecho.
- **Branches:** es posible crear ramas para realizar cambios sin que se vea afectada la rama principal o master.
- **Merge:** se emplea para fusionar los cambios de las diferentes ramas creadas con la rama principal.
- **Forks:** permite crear copias de los repositorios y se guarda en la cuenta de un usuario como un nuevo repositorio.

[11] [30] `https://docs.github.com/es`

- **Pull requests:** es utilizado para solicitar que se revisen e integren los cambios realizados en una rama o tras haber hecho un fork.
- **Issues:** son propuestas que realizan los usuarios para reportar errores que han encontrado en el código o para proponer posibles mejoras. Es posible enlazar issues con Pull requests.

Figura 3.12: GitHub

La plataforma GitHub se ha usado a lo largo de todo el desarrollo de este proyecto para guardar los cambios en el repositorio a través de commits. A su vez, para cada funcionalidad, se han creado ramas asociadas a una issue de mejoras y su respectivo pull request para que otros desarrolladores la revisaran e hicieran la integración de los nuevos cambios. Además, se ha utilizado su herramienta GitHub Pages para alojar y publicar el blog semanal.

3.10. Unibotics

Unibotics[12] es una plataforma web para uso educativo en Universidades como ya se ha mencionado en capítulos anteriores. Unibotics 3.11 es la versión de la página sobre la que se ha desarrollado este proyecto, es decir, la base desde la que se parte para implementar las nuevas funcionalidades.

Figura 3.13: Logo de Unibotics

Esta versión de Unibotics no diferencia entre distintos tipos de usuarios a excepción de aquellos que tienen rol de administrador. No existen *Roles*, *Clases*, *Cursos* ni *Permisos*, si no que se tiene una tabla rasa de usuarios. Todos los usuarios tienen las mismas funcionalidades dentro de la plataforma y pueden acceder a todos sus contenidos.

En Unibotics 3.11, cuando un usuario accede con su cuenta a la página web, entra en la sección de *Academy* y ve un listado con todos los ejercicios pudiendo

[12][16]`https://unibotics.org/`

acceder a cualquiera sin ningún tipo de restricción, ya que la base de datos consta de una tabla rasa de ejercicios. Cuando el usuario accede a uno de los ejercicios, puede comenzar a escribir su propio código en el caso de que no tuviera código almacenado y, en caso contrario, se le muestra el código que tuviera guardado. En la propia página de cada ejercicio hay varias partes diferenciadas:

- En la parte superior aparece la información más relevante del ejercicio.
- Debajo de la información hay varios botones para guardar el código, ejecutarlo, parar la ejecución o comenzar a ejecutar desde el principio.
- A continuación aparecen cuatro paneles, un editor donde escribir el código, una consola donde se puede visualizar los errores o trazas del código, una simulación gráfica que muestra lo que haría el robot con el código que se ha escrito y una imagen característica del ejercicio.

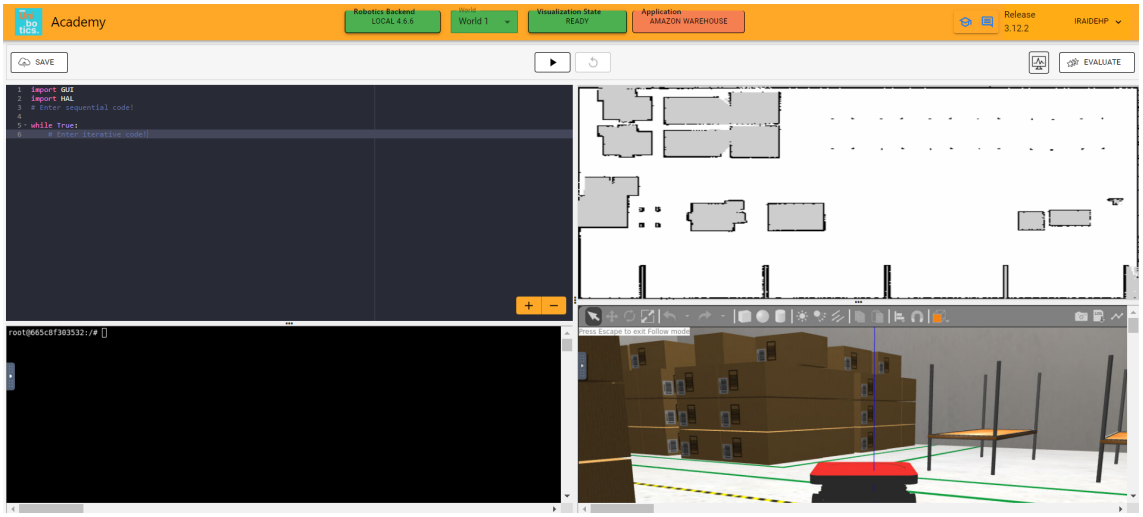

Figura 3.14: Página de un ejercicio

Tanto el simulador del robot como el código del alumno corre en RADI[13] (Robotics Academy Docker Image), una imagen con todas las dependencias necesarias para ejecutar los ejercicios. Con RADI el usuario únicamente ha de ejecutar el contenedor Docker sin necesidad de instalar las dependencias específicas de cada robot en los ordenadores propios. RADI recibe el código editado por el usuario y lo devuelve al navegador formateado. Posteriormente, el navegador muestra visualmente toda la información recibida.

Esta plataforma web hace uso de Django como servidor web, de PostgreSQL como base de datos para información estructural y de Amazon S3 para almacenar el código de los usuarios. Adicionalmente, el middlware utilizado es ROS junto al simulador Gazebo mencionados en el Capítulo 1.

[13] [31] https://doi.org/10.1007/s11042-023-17514-z

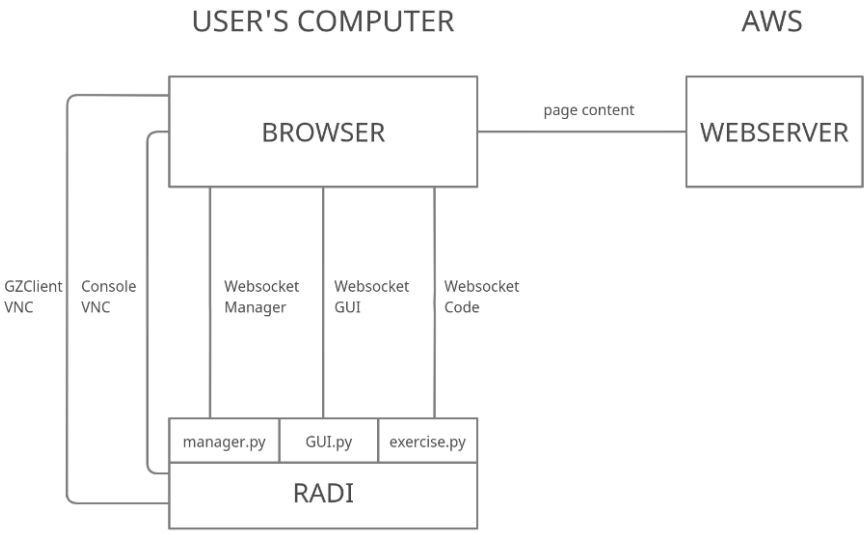

Figura 3.15: Arquitectura de Unibotics

Capítulo 4

Gestión de usuarios y contenidos

En este capítulo se explican los cambios que se han realizado sobre Unibotics 3.11, tanto para mejorar la gestión de los usuarios y contenidos de la plataforma, como para implementar posteriormente las funcionalidades para profesores.

4.1. Diseño

Para la consecución de los objetivos marcados, se han diseñado los cambios que hay que introducir en la plataforma. En la versión previa a este trabajo, Unibotics 3.11, la base de datos constaba de tablas para usuarios y ejercicios como ya se ha explicado en el capítulo anterior. Esto hacía que la página web estuviera orientada completamente para alumnos, ya que cualquier usuario que tuviera una cuenta o se registrara podía acceder a todos los ejercicios disponibles y poder aprender a programar diferentes robots.

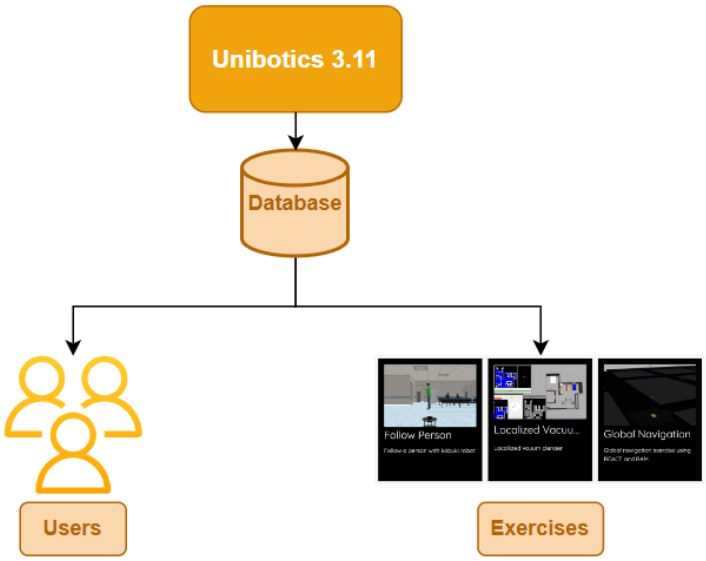

Figura 4.1: Estructura de usuarios y contenidos en Unibotics 3.11

En la nueva versión sobre la que se trabaja en este TFG, Unibotics 3.12, se pretende adaptar la plataforma para poder gestionar a los usuarios mediante clases y a los ejercicios mediante cursos. La aparición de clases facilita adaptar la plataforma para profesores pudiendo llevar un control de sus alumnos y clases. La inclusión de cursos permite una mejor gestión de los ejercicios.

Adicionalmente se diseñan diferentes tipos de permisos, para aceptar o denegar el acceso a los ejercicios y cursos. Dos de los cuatro permisos contemplados permiten dar acceso a cursos y ejercicios de forma individual a cada usuario. Los otros dos permisos existentes funcionan de manera grupal, es decir, se acepta o deniega el acceso a los contenidos por clases. Todos los usuarios que pertenecen a una clase que tiene permisos podrán entrar en los contenidos para los que la clase tiene permiso.

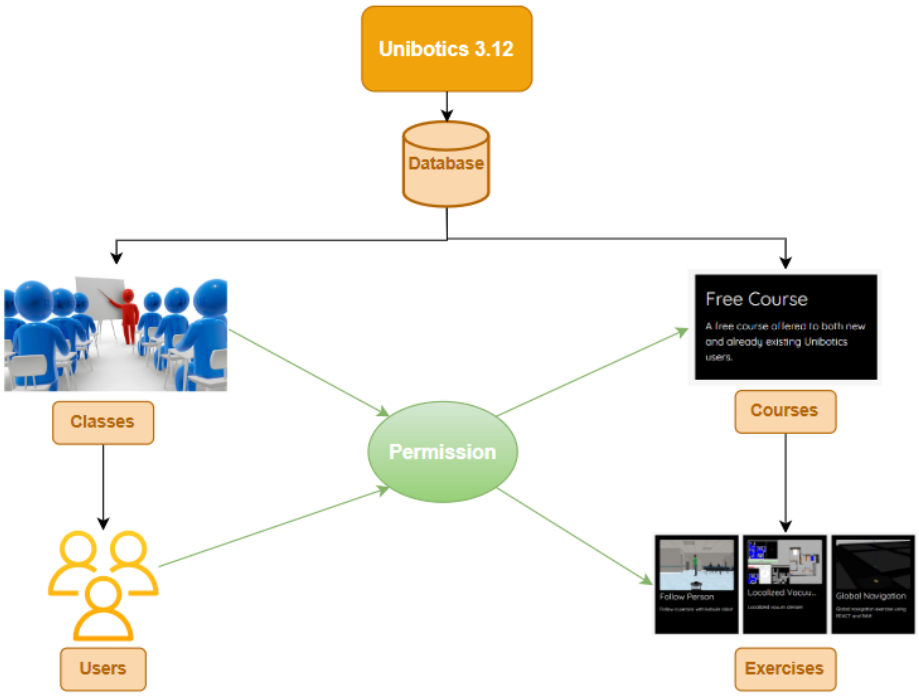

Figura 4.2: Estructura de usuarios y contenidos en Unibotics 3.12

A lo largo del capítulo se explica en profundidad cómo se ha conseguido implementar los cambios contemplados en este diseño. Además, se expone la nueva interfaz de usuario diseñada para incluir los nuevos cursos y ejercicios.

4.2. Cambios en la base de datos

En primer lugar, ha sido necesario añadir nuevas tablas en la base de datos de Unibotics para la gestión de usuarios y ejercicios, así como incluir roles para los usuarios.

4.2.1. Roles

El primer cambio es la inclusión de roles para los usuarios, ya que anteriormente todos los usuarios eran iguales exceptuando aquellos que tenían permisos de administrador. Por esta razón, es necesario incluir roles para poder distinguir entre diferentes tipos de usuarios. Existen dos tipos de roles:

- **Profesores:** tienen acceso a la información académica de todas sus clases y alumnos. Al acceder con este rol a la aplicación, el usuario tendrá algunas funcionalidades diferentes con respecto al resto de usuarios.
- **Alumnos:** los usuarios con este rol no tendrán gran diferencia con respecto a los usuarios de antes. Sin embargo, gracias a este rol, el alumno puede ser asignado a diferentes clases para ver los contenidos que pertenezcan a cada una de ellas.

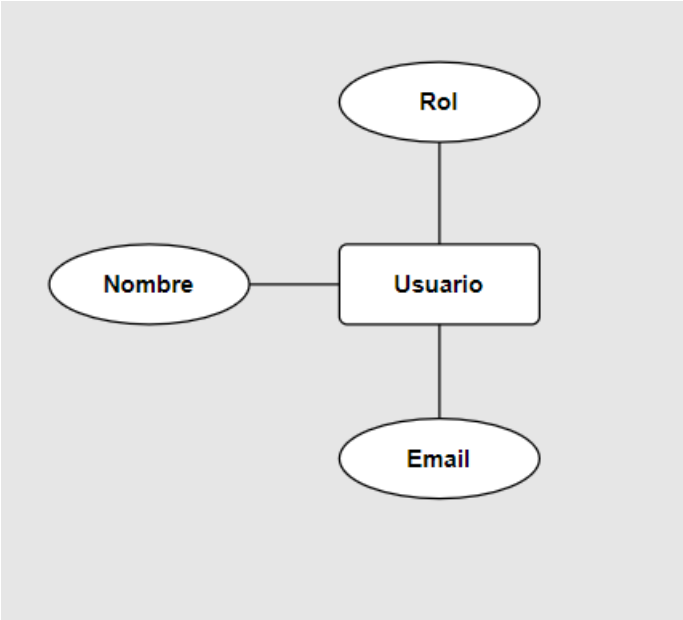

Figura 4.3: Nuevo campo rol en la tabla de Usuarios junto algunos de sus campos

Con la incorporación de estos roles, se facilita la labor de definir qué partes de la aplicación son accesibles para cada usuario y las opciones que tienen disponibles.

4.2.2. Clases

La primera tabla a crear es la relativa a las *Clases* para poder agrupar a los usuarios en diferentes clases. Las clases están formadas por usuarios que pueden tener rol tanto de profesor, como de alumno. Los usuarios pueden pertenecer a tantas clases como se quiera.

En cuanto al código desarrollado para esta tabla tenemos diferentes campos. Éstos son el identificador y nombre para cada clase y los usuarios que pertenecen a ella. A continuación, se muestra el fragmento de código correspondiente.

```
class Classes(models.Model):
    class Meta:
        db_table = 'clase'

    id_clase = models.AutoField(primary_key=True)
    usuarios = models.ManyToManyField(User,
                db_table='clase_usuario', db_column='usuarios')
    nombre_clase = models.CharField(max_length=100)

    def __str__(self):
        return self.nombre_clase

```

Listing 4.1: Tabla para las clases

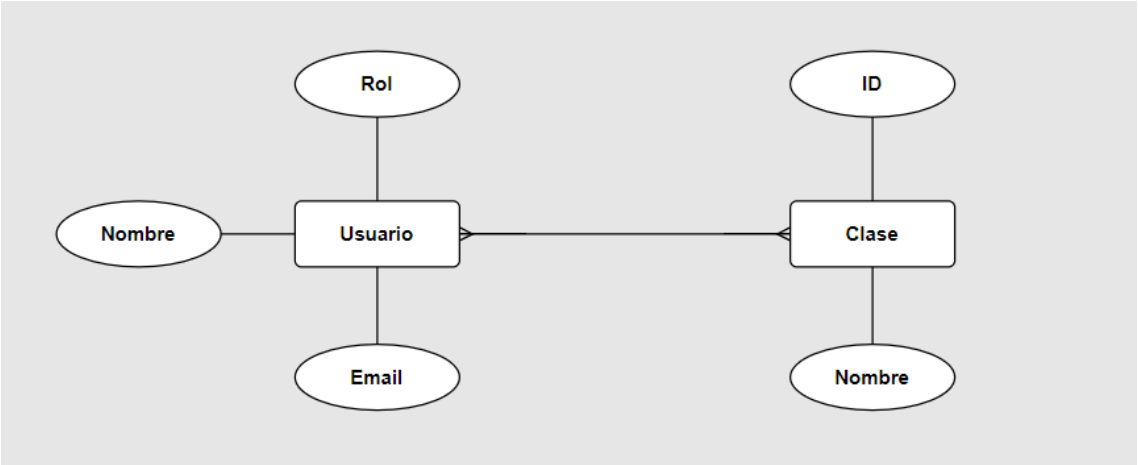

Figura 4.4: Relación entre Clases y Usuarios

La relación entre *Clase* y *Usuario* es una relación *Many To Many*, ya que un usuario puede pertenecer a múltiples clases y, a su vez, una clase contiene muchos usuarios.

4.2.3. Cursos

La siguiente tabla a crear es la de *Cursos*, que permitirá crear diferentes cursos para la plataforma. Hasta ahora, solo existían ejercicios a los que podía acceder cualquier usuario registrado. Sin embargo, con la aparición de los cursos podemos agrupar los ejercicios de Unibotics en función de la dificultad, temática u objetivos a alcanzar.

El código desarrollado para la tabla de cursos sólo incluye la información relativa a un curso como su identificador, nombre y descripción. La forma en que se relaciona un curso con el conjunto de ejercicios que pertenecen a él se explica más adelante. El código para la tabla *Cursos* es el siguiente.

```
class Courses(models.Model):
    class Meta:
        db_table = 'curso'
```

```
4
5           id_curso = models.AutoField(primary_key=True)
6           nombre_curso = models.CharField(max_length=100)
7           descripcion = models.TextField()
8
9           def __str__(self):
10              return self.nombre_curso
11
```

Listing 4.2: Tabla para los cursos

En cuanto a los ejercicios que pertenecen a un curso, ha sido necesario crear una nueva tabla en la base de datos que contiene los campos necesarios para seleccionar el ejercicio que queremos añadir a un curso, el curso al que pertenecerá y la posición que ocupará en el curso. Además, se añade una opción *ordering* que indica que cuando se quiera extraer la información de la base de datos, la información sea dada en orden de menor a mayor según el campo de la posición.

```
1       class OrderExerciseCourse(models.Model):
2           course =
                        models.ForeignKey(Courses,on_delete=models.CASCADE)
3           exercise = models.ForeignKey(Exercise,
                        on_delete=models.CASCADE)
4           order = models.PositiveIntegerField()
5
6           class Meta:
7               ordering = ['order']
8
9           def __str__(self):
10              return '%s, %s, %s'%(self.course.nombre_curso,
                        self.exercise.name, self.order)
11
```

Listing 4.3: Tabla para seleccionar el orden de los ejercicios dentro de un curso

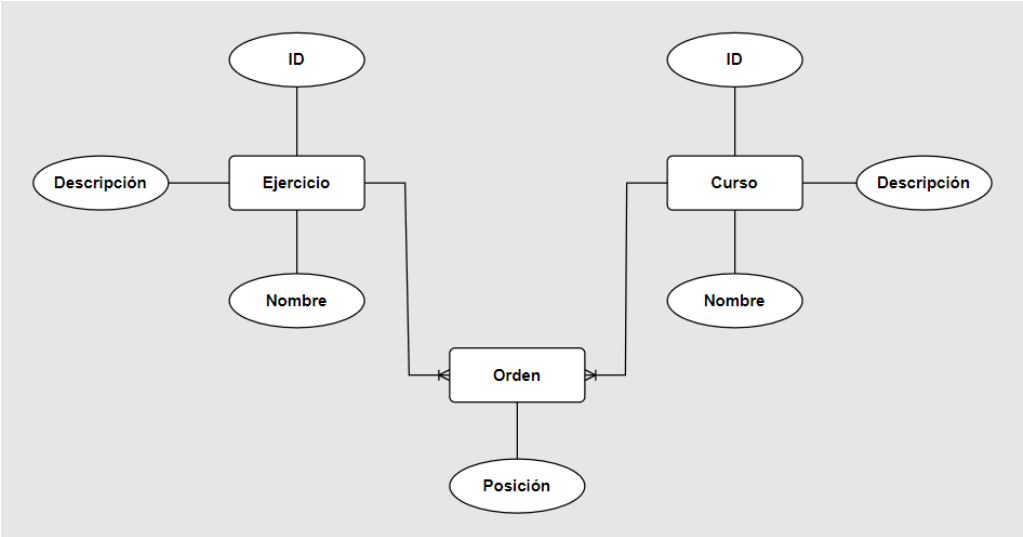

Figura 4.5: Relación entre Cursos y el orden de los Ejercicios dentro de él

4.3. Permisos

Con el fin de mostrar a cada usuario únicamente lo que necesita ver, ha sido necesario añadir cuatro tablas nuevas en la base de datos para controlar los permisos que tienen los usuarios a la hora de visualizar y acceder a los diferentes contenidos. Cada tabla se corresponde con uno de los cuatro permisos mencionados en la sección de diseño. Las tablas creadas para ello son:

- **Permisos de usuarios a ejercicios:** esta tabla permite dar los permisos necesarios para filtrar qué alumnos pueden visualizar cada ejercicio. De esta forma, un usuario al entrar en su cuenta verá unos ejercicios que posiblemente no pueden ver otros y viceversa. Los campos que contiene esta tabla son el usuario y el ejercicio en cuestión, y un campo permiso que determina si lo tiene o no mediante un booleano.

```
1          class PermissionUserExercise(models.Model):
2              usuario = models.ForeignKey(User,
                       on_delete=models.CASCADE)
3              exercise = models.ForeignKey(Exercise,
                       on_delete=models.CASCADE)
4              permission = models.BooleanField(default=False,
                       help_text="True if user has permission to
                       view the exercise. False otherwise")
5
6              def __str__(self):
7                  return '%s, %s, %s'%(self.usuario.username,
                       self.exercise.name, self.permission)
8
```

Listing 4.4: Permiso para el acceso de los usuarios a cada ejercicio

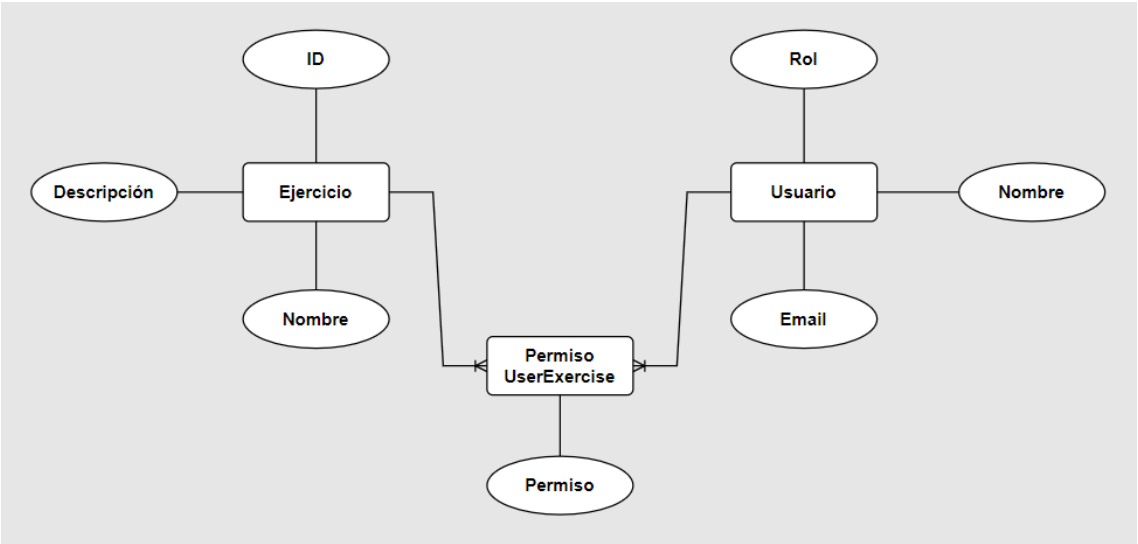

Figura 4.6: Permiso para el acceso de un usuario a un ejercicio

- **Permisos de clases a ejercicios:** al igual que ocurre con la anterior, esta tabla concede los permisos necesarios para visualizar y acceder a cada ejercicio

pero en relación a una clase (entendida como un grupo de usuarios). Este permiso es útil, ya que da la posibilidad de acceso a un ejercicio a múltiples usuarios con una sola instrucción. Aunque un usuario no tenga permisos para acceder a un ejercicio, si pertenece a una clase que si los tiene podrá acceder igualmente. La estructura de esta tabla es similar a la anterior, solo cambia el campo usuario por uno de tipo clase.

```
1         class PermissionClassExercise(models.Model):
2             clase = models.ForeignKey(Clase,
                      on_delete=models.CASCADE)
3             exercise = models.ForeignKey(Exercise,
                      on_delete=models.CASCADE)
4             permission = models.BooleanField(default=False,
                      help_text="True if class has permission to
                      view the exercise. False otherwise")
5
6             def __str__(self):
7                 return '%s, %s, %s'%(self.clase.nombre_clase,
                      self.exercise.name, self.permission)
8
```

Listing 4.5: Permiso para el acceso de las clases a cada ejercicio

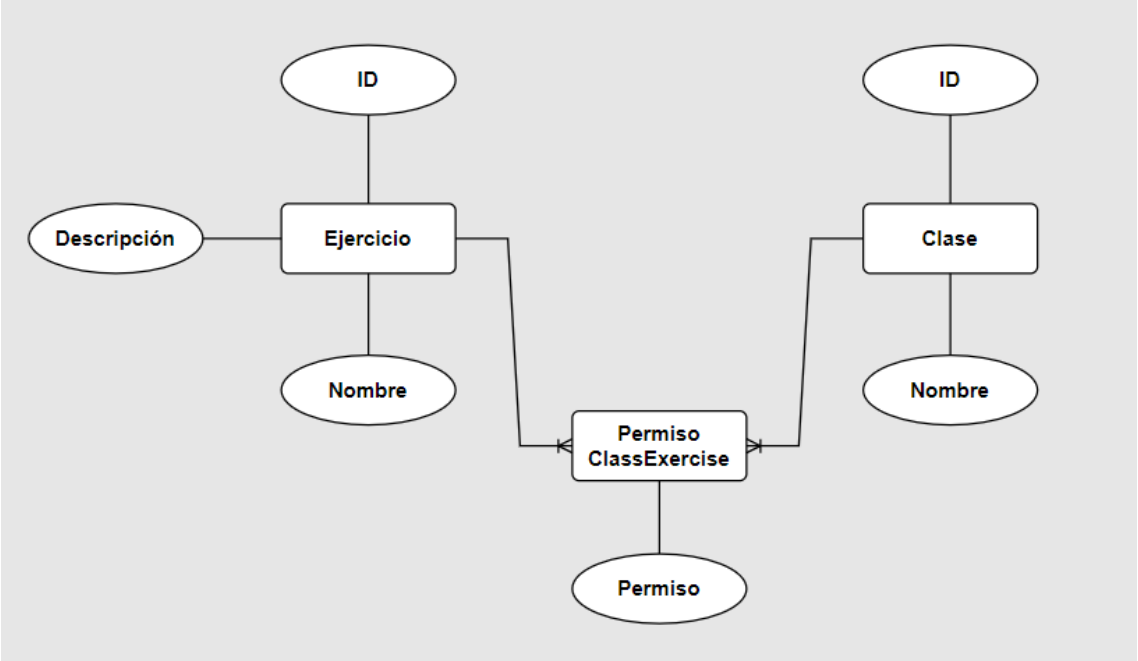

Figura 4.7: Permiso para el acceso de una clase a un ejercicio

- **Permisos de usuarios a cursos:** esta tabla es muy similar a la primera que se ha explicado en esta sección, concede permisos de visualización y acceso de los usuarios a cada curso. Su estructura sigue el mismo esqueleto que las anteriores, contiene los campos para seleccionar el usuario y curso correspondiente, y el campo que determina si se tiene el permiso.

```
class PermissionUserCourse(models.Model):
    usuario = models.ForeignKey(User,
            on_delete=models.CASCADE)
    course = models.ForeignKey(Curso,
            on_delete=models.CASCADE)
    permission = models.BooleanField(default=False,
            help_text="True if user has permission to
            view the course. False otherwise")

    def __str__(self):
        return '%s, %s, %s'%(self.usuario.username,
            self.course.nombre_curso, self.permission)

```

Listing 4.6: Permiso para el acceso de los usuarios a cada curso

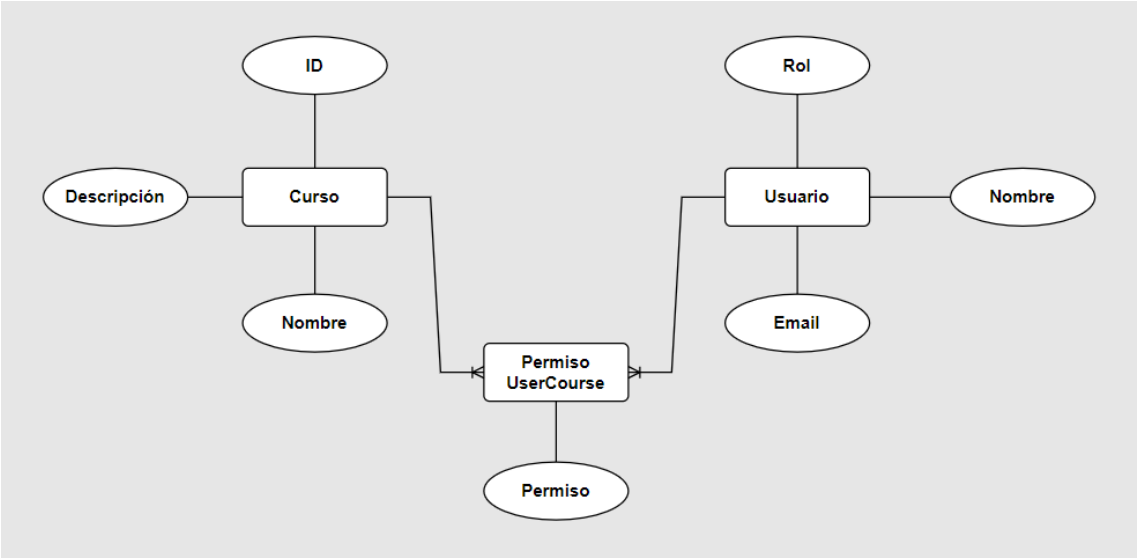

Figura 4.8: Permiso para el acceso de un usuario a un curso

- **Permisos de clases a cursos:** la última tabla permite conceder el acceso de un grupo de usuarios a una agrupación de ejercicios, o lo que es lo mismo, de las clases a los diferentes cursos. Puede ser útil para tener cursos con diferentes dificultades y asignárselos a cada clase en función del nivel educativo en el que se encuentren. Sigue la misma estructura mencionada en los anteriores apartados.

```
class PermissionClassCourse(models.Model):
    clase = models.ForeignKey(Clase,
            on_delete=models.CASCADE)
    course = models.ForeignKey(Curso,
            on_delete=models.CASCADE)
    permission = models.BooleanField(default=False,
            help_text="True if class has permission to
            view the course. False otherwise")

```

```
6             def __str__(self):
7                 return '%s, %s, %s'%(self.clase.nombre_clase,
                      self.course.nombre_curso, self.permission)
8
```

Listing 4.7: Permiso para el acceso de las clases a cada curso

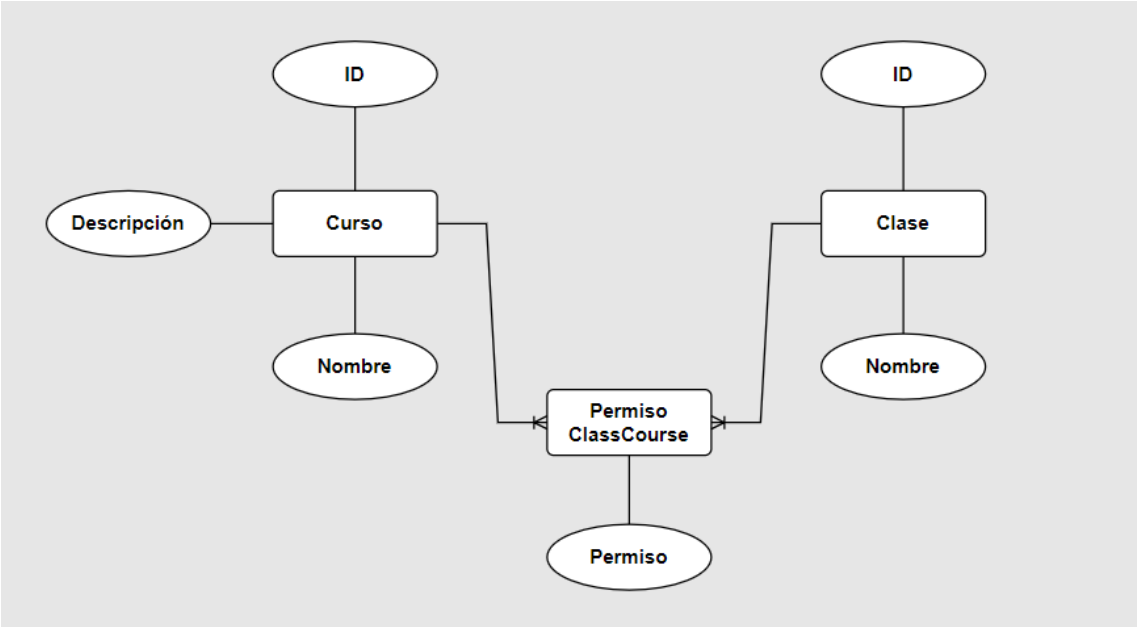

Figura 4.9: Permiso para el acceso de una clase a un curso

4.4. Nueva página web de aterrizaje de usuarios

Durante el desarrollo de este proyecto se ha implantado una nueva interfaz de usuario en la plataforma web de Unibotics. Previamente, al acceder a la sección *Academy* de la plataforma con un usuario, se mostraban directamente todos los ejercicios de la plataforma, como se muestra a continuación.

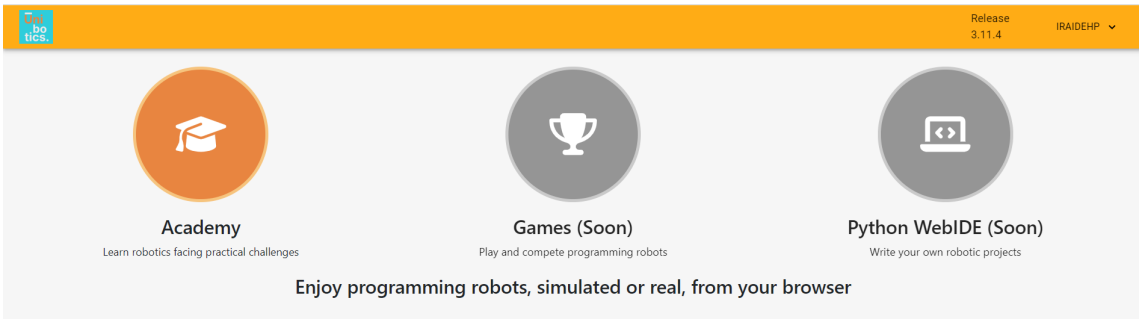

Figura 4.10: Página principal de Unibotics

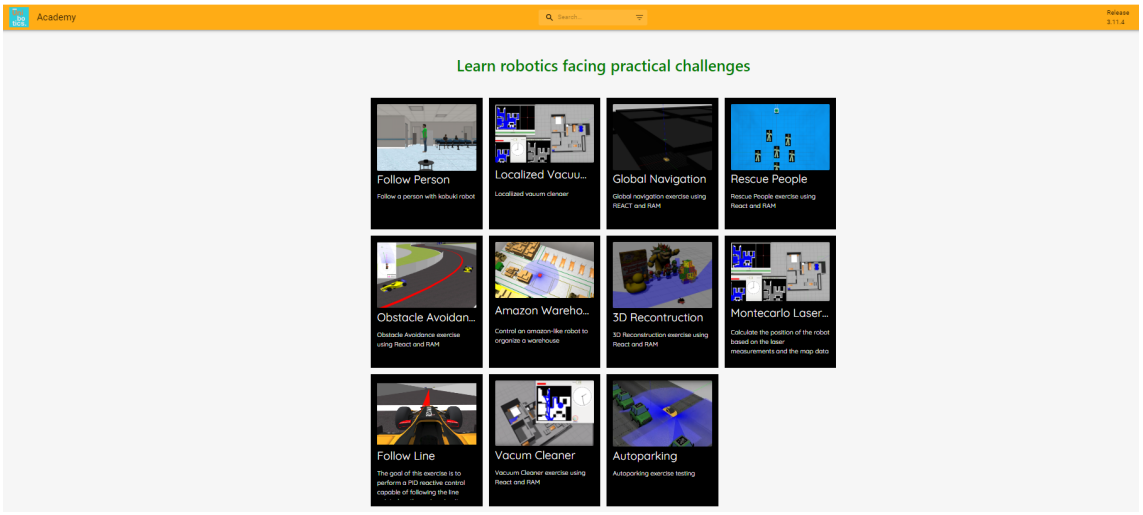

Figura 4.11: Página al acceder a la sección de *Academy*

Con el objetivo de tener una plataforma con mayor flexibilidad, se introducen algunos cambios al acceder a esta sección. Actualmente, se ven dos partes diferenciadas, por un lado los cursos disponibles y, por otro lado, los ejercicios disponibles para cada usuario. Cuando un usuario entra en esta página, se comprueban los permisos que tiene el usuario para acceder a todos los contenidos y sólo se le mostrarán aquellos para los cuáles tiene concedido el permiso. La comprobación de estos permisos se realiza desde dos funciones (una para cursos y otra para ejercicios) creadas en las vistas de Django en las que se siguen los siguientes pasos.

1. Se extrae el nombre del usuario que realiza la petición.
2. Se extraen de la base de datos todas las clases a las que pertenece dicho usuario.
3. Se extraen todos los cursos y ejercicios a los que tiene permiso el usuario.
4. Se extraen todos los cursos y ejercicios a los que tiene permiso el usuario por pertenecer a determinadas clases
5. Por último, se procesan los datos de cursos y ejercicios, obteniendo un listado de éstos con los valores relevantes, y se envían en formato JSON.

Este fragmento de código se corresponde con la extracción de los *Cursos*, para los ejercicios es similar cambiando los objetos a los que se hace referencia en la base de datos, es decir, se usan las tablas *PermissionUserExercise*, *PermissionClassExercise* y *Exercise*.

```
1      #Fragmento correspondiente al paso 1
2      user = request.user
3
4      #Fragmento correspondiente al paso 2
5      user_class = Classes.objects.filter(usuarios=user)
6      user_class = list(user_class.values('id_clase'))
7
```

```
8       #Fragmento correspondiente al paso 3
9       course_user =
                    PermissionUserCourse.objects.filter(usuario=user,
                    permission=True)
10      course = list(course_user.values('course'))
11
12      #Fragmento correspondiente al paso 4
13      for c in user_class:
14          course_class =
                    PermissionClassCourse.objects.filter(clase=c['id_clase'],
                    permission=True)
15          course_class = list(course_class.values('course'))
16          for cur in course_class:
17              if cur not in course:
18                  course.append(cur)
19
20      #Fragmento correspondiente al paso 5
21      courses_list =[]
22      for c in course:
23          courses = Courses.objects.filter(id_curso=c['course'])
24          courses = list(courses.values())
25          courses_list.append(courses[0])
26      return JsonResponse({"courses": courses_list})
27
```

Listing 4.8: Extracción de datos de los cursos que un usuario puede ver

Una vez se tiene esta página, se puede elegir entrar a un ejercicio o a un curso. En el caso de querer entrar a un ejercicio no hay diferencias con respecto al funcionamiento previo de la aplicación. Sin embargo, en el caso de querer entrar a un curso se ha añadido una nueva página en la que se muestra el nombre y la descripción del curso, y los ejercicios que pertenecen a él en el orden seleccionado.

Para extraer la información relativa a un curso se han seguido los siguientes pasos.

1. Se extrae de la base de datos el objeto relativo al curso que se corresponde con el identificador de curso que viene en la petición.

2. Se obtiene el nombre y la descripción del objeto extraído en el punto anterior y se envía esta información en formato JSON.

3. Se extraen todos los objetos de la tabla *OrderExerciseCourse* que se corresponden con el curso en cuestión.

4. Obtenemos los datos de cada ejercicio obtenido en el punto anterior, se introducen en una lista y se envían en formato JSON.

```
1       def course_information(request, courseid):
2           #Fragmento correspondiente al paso 1
3           course = Courses.objects.get(id_curso=courseid)
4           #Fragmento correspondiente al paso 2
5           name = course.nombre_curso
6           description = course.descripcion
```

```
7           return JsonResponse({"name": name, "description":
                    description})
8
9
10      def get_exercise_list(request, courseid):
11          #Fragmento correspondiente al paso 1
12          course = Courses.objects.get(id_curso=courseid)
13          #Fragmento correspondiente al paso 3
14          order = OrderExerciseCourse.objects.filter(course=course)
15          #Fragmento correspondiente al paso 4
16          exercises_list = []
17          for i in order:
18              exercise =
                        Exercise.objects.filter(exercise_id=i.exercise.exercise_id).v
19              exercise = list(exercise)
20              exercises_list.append(exercise[0])
21          return JsonResponse(exercises_list, safe=False)
22
```

Listing 4.9: Extracción de la información de cada curso

Cada página de la plataforma es un HTML que extiende de una base e incluye el componente padre de React, y cada componente padre incluye a los componentes hijo necesarios para cada página. Los datos, extraídos y enviados en formato JSON, a los que se hace referencia anteriormente son devueltos al componente hijo que realizó la petición. El flujo que siguen los componentes hijo para enviar y recibir los datos necesarios es el siguiente:

1. El componente hijo hace una petición mediante una URL que esta contemplada en el fichero *urls.py* de *Django*.

2. La URL en cuestión redirige a la función correspondiente del fichero *views.py*. Las funciones son las explicadas previamente que devuelven un JSON.

3. El componente que realizó la petición, recibe la respuesta JSON y utiliza estos datos para introducirlos en un fragmento HTML que será enviado al componente padre.

```
1           #fragmento que realiza una peticion a una URL y el valor
                    obtenido se asigna a una variable
2           const [coursesList, setCoursesList] = useState([]);
3           useEffect(() => {
4               const apiURL = `/courses`;
5               fetch(apiURL)
6                 .then((res) => res.json())
7                 .then((data) => {
8                         setCoursesList(data.courses);
9                     });
10            }, [setCoursesList]);
11
12          #fragmento que redirige la URL a una de la funciones que
                    devulven un JSON
13          path("courses/", views.get_courses, name="get_courses")
14
```

```
15          #fragmento que devuelve un HTML al componente padre con
                los valores recibidos en el JSON
16          <div className={"title"}>Available courses</div>
17          <div className={"course-list"}>
18            {coursesList.map(course => {
19              return (
20                  <CourseCard
21                      key={course.id_curso}
22                      courseid={course.id_curso}
23                      name={course.nombre_curso}
24                      description={course.descripcion}
25                  />
26              );
27            })}
28          </div>
29
```

Listing 4.10: Código de petición a una URL y retorno de HTML

El ejemplo anterior es un pequeño fragmento que muestra cómo obtener el listado de cursos a los que un usuario puede acceder, para obtener los ejercicios o la información de un curso concreto se sigue el mismo patrón.

4.5. Validación experimental

Los resultados obtenidos en este capítulo suponen una mejora en la gestión de usuarios y contenidos de la plataforma web Unibotics. Por un lado, se consigue manejar el papel que desempeña cada usuario en la plataforma mediante la asignación de *Roles* y adaptar la aplicación para profesores mediante la agrupación de diferentes usuarios en *Clases*. Por otro lado, se consigue una mayor flexibilidad a la hora de mostrar los contenidos de la plataforma limitando el acceso a cada uno de ellos mediante la asignación de *Permisos*. Además, aparece el concepto de *Curso* que permite agrupar diferentes ejercicios en el orden deseado.

Para validar experimentalmente los cambios realizados, se van a describir algunos casos de uso tanto desde el punto de vista de un administrador como de el de un usuario con otro rol.

CASO DE USO 1: Crear clases

- **Objetivo:** Intentar crear una nueva clase desde el panel de administrador de Django.
- **Autor:** un usuario administrador que quiere realizar la acción.
- **Acciones:** descripción del flujo que seguiría la acción.

  1. Un usuario con rol de administrador inicia sesión en la página web.
  2. Una vez dentro, despliega el menú superior derecho y hace click en el panel de administrador.

3. El sistema comprueba su permiso de administrador y se le permite acceder.
4. Cuando consigue entrar, hace click en añadir una nueva clase.
5. Se redirige al usuario a una página para añadir la clase y tiene que seleccionar los usuarios que pertenecerán a ella y poner un nombre para dicha clase.
6. Tras completar la información de una clase, se guarda y al entrar en la tabla de las clases aparece reflejada la nueva clase creada.

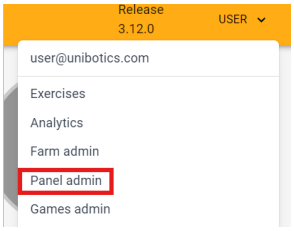

Figura 4.12: *Panel Admin*

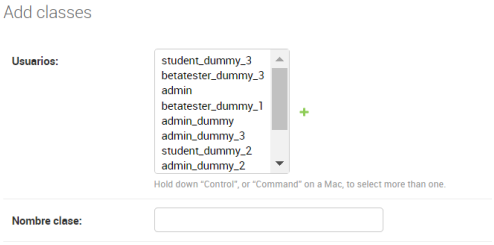

Figura 4.13: Añadir una clase

CASO DE USO 2: Crear curso

- **Objetivo:** Intentar crear un nuevo curso con un orden de ejercicios específico desde el panel de administrador de Django.
- **Autor:** un usuario administrador que quiere realizar la acción.
- **Acciones:** descripción del flujo que seguiría la acción.

1. Un usuario con rol de administrador inicia sesión en la página web.
2. Una vez dentro, despliega el menú superior derecho y hace click en el panel de administrador.
3. El sistema comprueba su permiso de administrador y se le permite acceder.
4. Cuando consigue entrar, hace click en añadir un nuevo curso.
5. Se redirige al usuario a una página para añadir el nuevo curso y debe elegir su nombre y descripción.
6. Cuando termina de completar la información del curso, se guarda y al entrar en la tabla de cursos aparece reflejada el nuevo curso.
7. A continuación, el usuario accede a la tabla para añadir los ejercicios al curso en el orden deseado.
8. Una vez dentro, selecciona el curso creado anteriormente, el ejercicio que quiere añadir y su posición dentro del propio curso, y le da a guardar. Tras esta acción, ve reflejado el ejercicio dentro del curso en la posición marcada. Esta acción la repite con cada ejercicio que quiere añadir al curso.

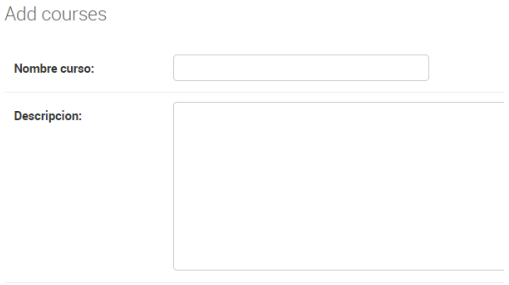

Figura 4.14: *Panel Admin*

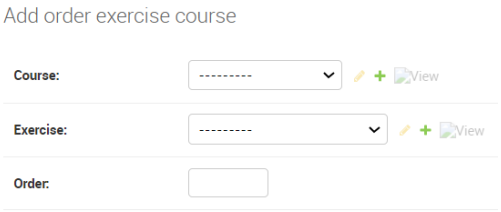

Figura 4.15: Añadir una clase

CASO DE USO 3: Asignar permisos a un usuario

- **Objetivo:** añadir un permiso para que un usuario pueda ver y acceder a un ejercicio.
- **Autor:** un usuario administrador que quiere realizar la acción.
- **Acciones:** descripción del flujo que seguiría la acción.

  1. Un usuario con rol de administrador inicia sesión en la página web.
  2. Una vez dentro, despliega el menú superior derecho y hace click en el panel de administrador.
  3. El sistema comprueba su permiso de administrador y se le permite acceder.
  4. Cuando consigue entrar, hace click en añadir en la tabla *PermissionUserExercise*.
  5. Se redirige al usuario a la página para añadir un nuevo permiso y debe seleccionar el usuario, el ejercicio al que se quiere dar permiso y dar el valor *True* al campo *Permission*, y guarda.
  6. Una vez guardado el permiso, debe verse reflejado en la tabla correspondiente y el usuario en cuestión puede ver el ejercicio en su apartado *Available exercises*.

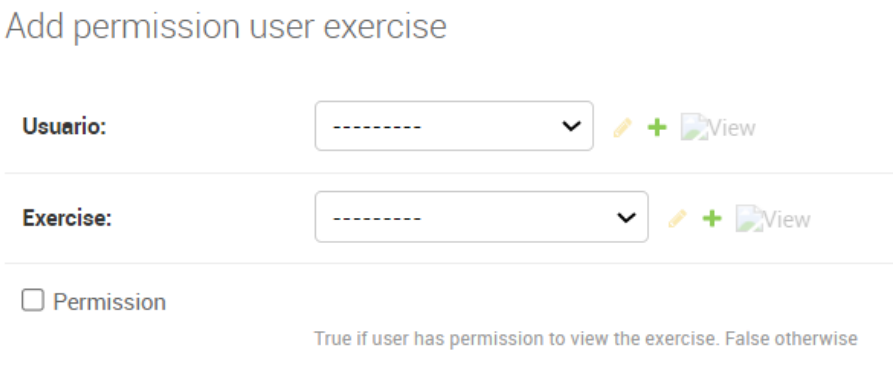

Figura 4.16: Dar permiso a un usuario para acceder a un ejercicio

Para dar permisos a una clase para acceder a un ejercicio o curso y a un usuario a un curso, el caso de uso sería idéntico a excepción de cambiar la tabla en la que se añade el permiso.

CASO DE USO 4: Modificar un permiso

- **Objetivo:** modificar un permiso para una clase que tenía acceso a un curso y ahora se quiere quitar el permiso.
- **Autor:** un usuario administrador que quiere realizar la acción.
- **Acciones:** descripción del flujo que seguiría la acción.
  1. Un usuario con rol de administrador inicia sesión en la página web.
  2. Una vez dentro, despliega el menú superior derecho y hace click en el panel de administrador.
  3. El sistema comprueba su permiso de administrador y se le permite acceder.
  4. Una vez dentro, accede a la tabla *PermissionClassCourse* y busca el permiso en cuestión. Cuando lo encuentra, entra en el objeto correspondiente y se cambia el valor del campo *Permission* de *True* a *False* y se guarda.
  5. Con el permiso cambiado, debe visualizarse en la tabla que el permiso se encuentra en *False* y que un usuario de dicha clase ya no puede visualizar el curso.

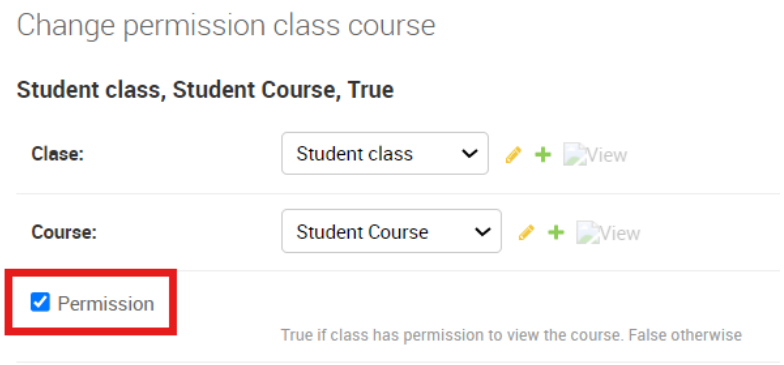

Figura 4.17: Modificar un permiso existente de una clase a un curso

Para modificar clases, cursos y otros permisos, el caso de uso sería similar. Hay que acceder al objeto que se quiere modificar, se hace la modificación y se guarda.

CASO DE USO 5: Acceder a la plataforma y visualizar la página principal

- **Objetivo:** visualizar y acceder a los contenidos disponibles de la plataforma para un usuario.
- **Autor:** un usuario registrado tanto con rol de alumno, profesor o administrador.

- **Acciones:** descripción del flujo que seguiría la acción.

  1. Un usuario registrado con cualquier rol ingresa en la web de Unibotics y accede a la página principal.
  2. Se le muestran las diferentes secciones de la página web y entra en la sección *Academy*.
  3. Al entrar en esta sección, el servidor comprueba todos los permisos que tiene el usuario por sí mismo o por pertenecer a una clase para mostrarle la página con los ejercicios y cursos que tiene disponibles el usuario en cuestión.
  4. A continuación, el usuario puede ver dos secciones. En la parte superior puede ver los cursos a los que puede acceder y en la parte inferior los ejercicios.
  5. En este punto, el usuario puede hacer click en cualquiera de los contenidos que se le muestran y entrar en ellos.

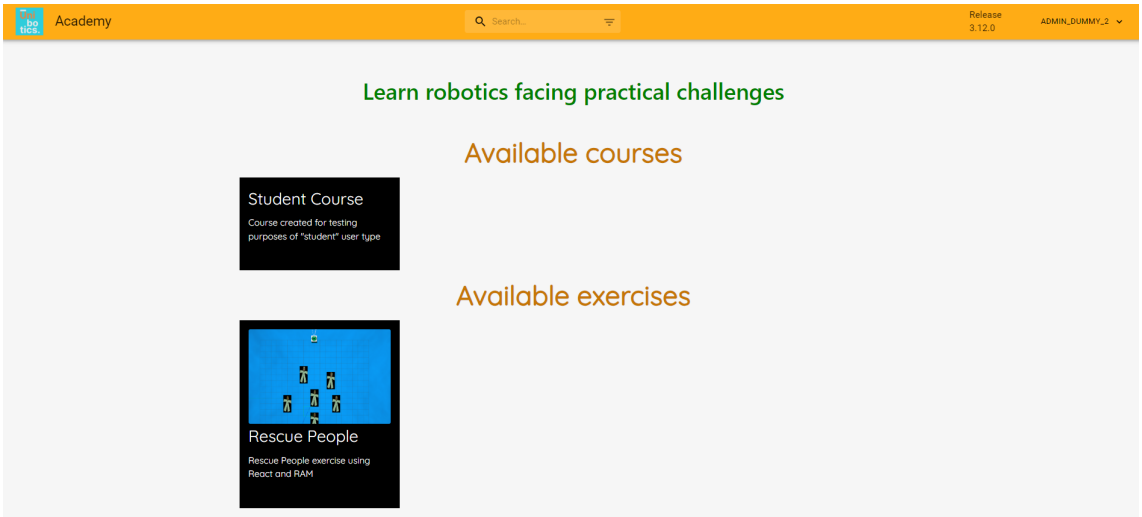

Figura 4.18: Página al acceder a la sección de *Academy*

CASO DE USO 6: Acceder a la página de un curso

- **Objetivo:** acceder a la página de un curso concreto al que el usuario tiene acceso.
- **Autor:** un usuario registrado tanto con rol de alumno, profesor o administrador.
- **Acciones:** descripción del flujo que seguiría la acción.

  1. Un usuario registrado con cualquier rol ingresa en la web de Unibotics y accede a la página principal.
  2. Se le muestran las diferentes secciones de la página web y entra en la sección *Academy*.

3. Al entrar en esta sección, el servidor comprueba todos los permisos que tiene el usuario por sí mismo o por pertenecer a una clase para mostrarle la página con los ejercicios y cursos que tiene disponibles el usuario en cuestión.
4. El usuario quiere acceder a un curso al que tiene acceso y hace click en él.
5. El servidor carga toda la información relativa a ese curso, es decir, su nombre, descripción y los ejercicios en orden. A continuación, muestra la página del curso al usuario con todo su contenido.
6. Finalmente, el usuario ya ve la página del curso y puede hacer click en cualquiera de los ejercicios del curso para acceder y comenzar a practicarlos.

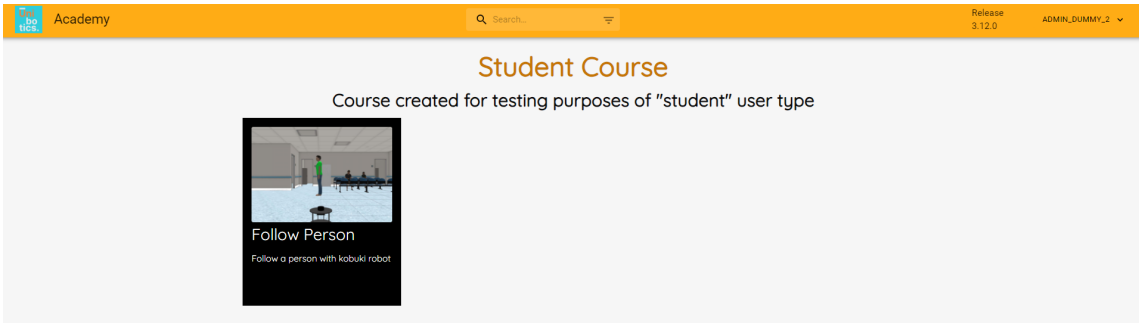

Figura 4.19: Página al acceder a un curso

ACADEMY		
Classess	+ Add	Change
Coursess	+ Add	Change
Order exercise courses	+ Add	Change
Permission class courses	+ Add	Change
Permission class exercises	+ Add	Change
Permission user courses	+ Add	Change
Permission user exercises	+ Add	Change

Figura 4.20: Tablas de la base datos a las que se hace referencia en los casos de uso

Capítulo 5

Funcionalidad para profesores

En este capítulo se explican algunas funcionalidades que se han incluido en la web de Unibotics para mejorar la experiencia de los usuarios que tienen rol de profesor. Las modificaciones del capítulo anterior no incluyen ninguna diferencia para los usuarios con rol de profesor con respecto a aquellos que tienen rol de alumno ya que el acceder a la web pueden ver la misma información.

El propósito de este capítulo es implementar algunas funcionalidades que permitan un mayor control de los profesores sobre las clases que tienen y los alumnos que pertenecen a ellas. De esta forma, se facilita la evaluación del trabajo que realizan los alumnos en Unibotics.

5.1. Monitorización de la actividad de sus alumnos

La sección *Analytics* es la parte de la aplicación en la que se pueden ver diferentes datos de la plataforma en forma de numerosos gráficos mediante el uso de Dash. En esta sección se ha incluido un apartado para profesores para que pueda visualizar la actividad que tienen sus clases y alumnos en Unibotics.

El apartado que se ha creado para profesores consta de dos gráficos, uno para clases y otro para alumnos. Para cada profesor solo se muestran los datos de sus clases y alumnos, evitando poder acceder a datos de otras clases y usuarios.

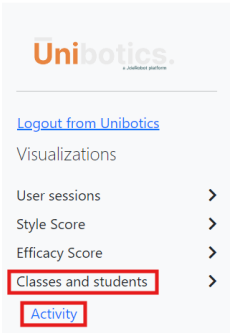

Figura 5.1: Menú al acceder a *Analytics*

Cuando el profesor accede a la actividad de sus clases y alumnos, nada más entrar observa la gráfica general de todos los datos. El profesor puede filtrar por clases y alumnos para ver la información de actividad de sus alumnos de forma colectiva o individual. Además, puede seleccionar el rango de fecha deseado.

Figura 5.2: Gráficos de actividad de clases y alumnos

Para la obtención del menú y los gráficos de actividad de los alumnos se han implementado algunos cambios. En primer lugar, cuando un usuario quiere acceder a esta sección se debe comprobar si su rol se corresponde con *profesor*, ya que en caso de no serlo se le denegará el acceso. Al comprobar el rol, si se verifica que el usuario es profesor se le asigna una cookie específica para profesores.

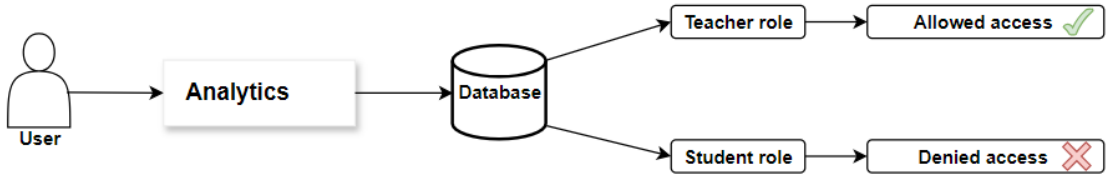

Figura 5.3: Comprobación del acceso a *Analytics*

En segundo lugar, se crea el submenú correspondiente para la actividad de clases alumnos. Dentro del apartado *Classes and students* podrían añadirse tantas secciones como se quiera, en esta versión únicamente se monitoriza la *Actividad*.

```
        submenu_4 = [
        dash.html.Li(
            dbc.Row(
                [
                    dbc.Col("Classes and students"),
                    dbc.Col(
                        dash.html.I(className="fas
                        fa-chevron-right mr-3"), width="auto"
                    ),
                ],
                className="my-1",
            ),
            style={"cursor": "pointer"},
            id="submenu-4",
        ),
        dbc.Collapse(
            [dbc.NavLink("Activity", href="/information")],
            id = "submenu-4-collapse"
        ),
    ]

```

Listing 5.1: Código para crear el menú de actividad

Tras esto, la aplicación de Dash comprueba la cookie que tiene el usuario, ya que los usuarios que tienen acceso a Dash pueden ser profesores o administradores. En el caso de ser administrador se le mostrarán todos los menús existentes, pero en el caso de ser profesor solo verá el menú de clases y alumnos que se ha explicado anteriormente. Cuando el profesor decide entrar en *Actividad*, se redirige a la URL */information*, como se puede ver en el código anterior. Al acceder a esta URL, se hace uso de tres funciones:

- **get classes and students name:** se le pasa el identificador del usuario profesor y se obtiene el nombre de todas las clases que tiene y el nombre de todos sus alumnos.
- **activity:** se le pasa una lista con el nombre de todos los alumnos del usuario profesor y se obtienen todos los datos de actividad de estos usuarios.
- **get temporal figure session time:** función a la que se le pasan los datos de actividad de los alumnos y se encarga de crear la figura relativa a estos datos, con la fecha en el eje X y el tiempo por sesión en el eje Y.

Con estas funciones, se recopilan los datos iniciales que se muestran al profesor nada más entrar, que incluye el de todos los usuarios, ya que aún el profesor no ha filtrado la información específica que quiere ver.

Los nombres de clases y alumnos, obtenidos con la primera de las funciones, también se usan para crear los desplegables con los nombres que se pueden elegir a la hora de filtrar. Para cada apartado, clases y alumnos, se crea el desplegable para las fechas y para el nombre de alumnos o clases. Estos desplegables se incluyen en el HTML que se enviará junto con los gráficos extraídos.

```
            date_picker = dash.dcc.DatePickerRange(
                id='actividad_clases',
                clearable=True
            ),
            drop_down = dash.dcc.Dropdown(
                id='classes',
                placeholder='Select class...',
                options=[{'value': x, 'label': x}
                         for x in name_classes],
                multi=True,
            ),
            date_picker_unique = dash.dcc.DatePickerRange(
                id='actividad_alumnos',
                clearable=True
            ),
            drop_down_unique = dash.dcc.Dropdown(
                id='students',
                placeholder='Select student...',
                options=[{'value': x, 'label': x}
                         for x in students_name],
                multi=True,
            ),

```

Listing 5.2: Código de los desplegables de fechas y nombres

Por último, cabe señalar la forma en que se comporta la aplicación cuando el profesor quiere actualizar la información mediante alguno de los filtros. Para actualizarlo se utiliza el decorador *@app.callback* de Dash que permite vincular una salida a una o varias entradas. Existen tres casos posibles para que se actualicen los datos:

- **Seleccionar rango de fechas**

  Es posible seleccionar la fecha de inicio, la fecha de fin o ambas. Cuando se selecciona alguna fecha concreta, el gráfico cambia automáticamente buscando únicamente los datos de actividad de los alumnos dentro de los valores seleccionados. Existen dos entradas, una para la fecha de inicio y otra para la de fin vinculadas a la salida, tanto para el gráfico de *Actividad de clases* como para el de *Actividad de alumnos*.

```
df_sessions = dash_utils.activity(conn, start_date, end_date, students_name, "Total",'total_time')
```

Figura 5.4: Datos de actividad con la fecha de inicio y fin seleccionada

- **Seleccionar una o varias clases**

  Es posible seleccionar una o varias clases para poder visualizar la actividad de los alumnos de forma conjunta o separada. Cada vez que se selecciona una clase, se obtiene una salida que responde a las tres entradas que tiene vinculadas.

  Para la salida se obtienen de la base de datos, los datos del usuario que realiza la petición y se comprueba el valor de la entrada Clases. Si es nulo se devuelve una gráfica con los datos de todas las clases. Sin embargo, si se ha seleccionado una o varias clases se hace uso de una nueva función que devuelve únicamente el nombre de los usuarios que pertenecen a las clases seleccionadas. Con estos nombres de usuario se llama a la función que devuelve la actividad de estos alumnos para enviarla a la función que nos devuelve la nueva figura.

```
1         @app.callback(
2         dash.dependencies.Output('activity_class', 'figure'),
3         [dash.dependencies.Input('actividad_clases',
                    'start_date'),
4          dash.dependencies.Input('actividad_clases',
                    'end_date'),
5          dash.dependencies.Input('classes', 'value')])
6
```

Listing 5.3: Código de las entradas que tiene vinculadas la salida Actividad de Clases

- **Seleccionar uno o varios alumnos**

  Al igual que ocurre con las clases, se pueden seleccionar uno o varios alumnos. Esto permite hacer comparativas entre diferentes alumnos o clases. También consta de tres entradas vinculadas a la salida, similar al ejemplo anterior pero la última entrada se corresponde con los alumnos seleccionados.

  Para la salida se comprueba si es nulo el valor de alumno. En caso afirmativo, igual que con las clases, se retorna un gráfico con la actividad de todos los usuarios. En caso negativo, se obtienen los valores de actividad de los alumnos que se hayan seleccionado para ser enviados a la figura que se le mostrará al profesor.

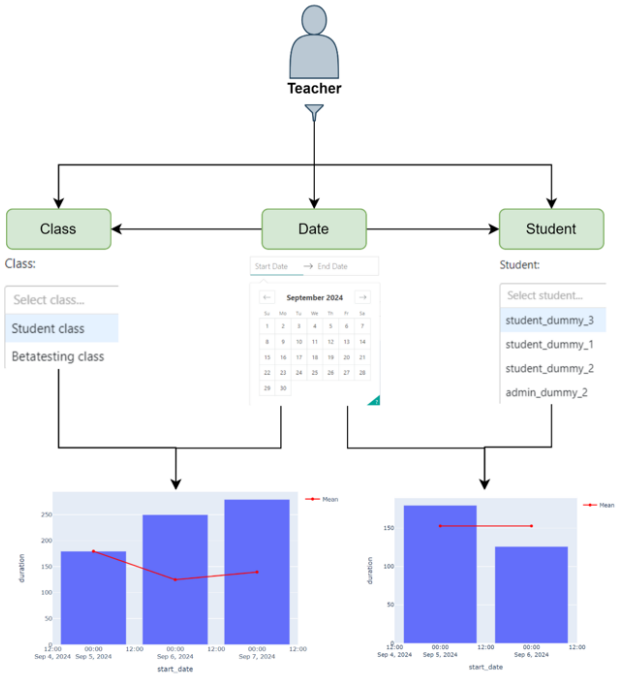

Figura 5.5: Funcionamiento de los filtros en Dash

5.2. Visualización y ejecución del código de sus alumnos

La segunda funcionalidad que se ha implementado para profesores es la posibilidad de elegir un alumno de sus clases, para poder visualizar el código que ha desarrollado y ejecutarlo con el fin de probar su funcionamiento. Esto facilita la misión de corregir y evaluar para el profesor, eliminando la necesidad de que el alumno tenga que entregar su código al profesor y a su vez el profesor copiar este código y probarlo en su cuenta.

Cuando un usuario accede a un ejercicio cualquiera se comprueba su rol, si tiene rol de profesor se le mostrarán los desplegables para poder elegir el alumno y la clase a la que pertenece el alumno para facilitar la búsqueda. En cualquier otro caso, no se mostrarán los desplegables y se visualizará la página del ejercicio como antes sin ningún cambio.

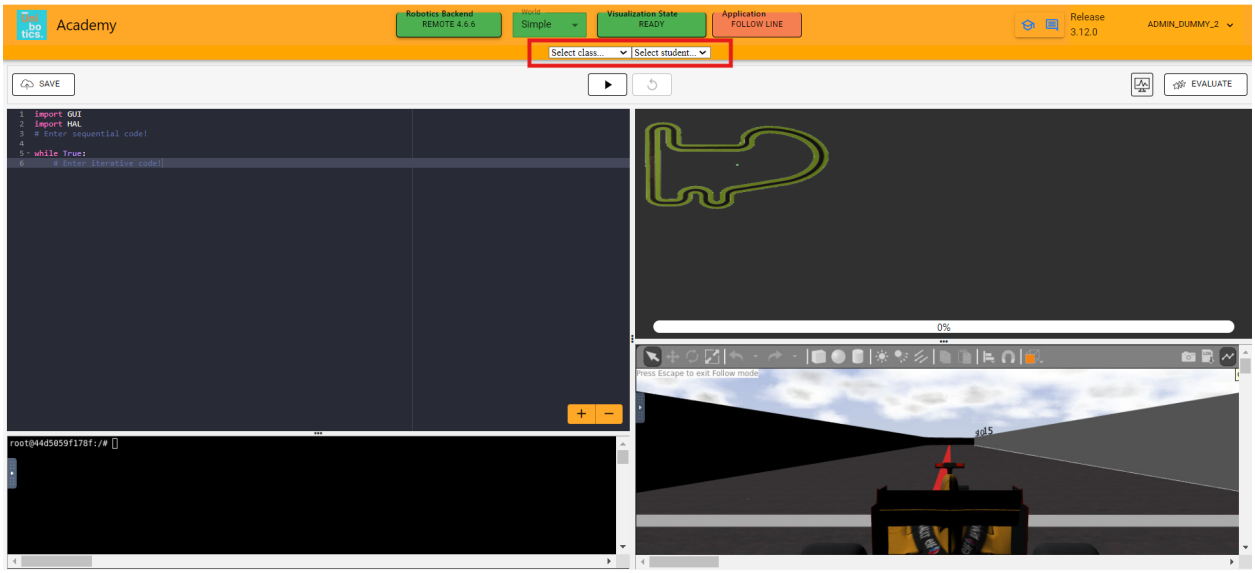

Figura 5.6: Página de un ejercicio para profesores

Para crear esta funcionalidad para profesores, en primer lugar se ha creado un nuevo componente de React para incluirlo junto a los demás componentes, en la página del ejercicio. Este componente devuelve un fragmento HTML con los dos desplegables:

- **Desplegable para seleccionar una clase:** se hace una petición al servidor mediante una URL para obtener el nombre de todas las clases que tiene el profesor en cuestión. Una vez tenemos el nombre de las clases, se muestran en el despleglable para seleccionar una.

```
1          useEffect(() => {
2              #URL para la peticion
3              fetch('/classes/')
4                  .then(response => {
5                      #si la respuesta no es buena se lanza un
                       mensaje de error
6                      if (!response.ok) {
7                          throw new Error('Network response was
                       not ok ' + response.statusText);
8                      }
9                      return response.json();
10                 })
11                 #si la respuesta es buena, se asigna el nombre
                       de las clases a una variable
12                 .then(data => {
13                     setclasses(data.classes);
14                 })
15                 .catch(error => {
16                     console.error('Hubo un error:', error);
17                 });
18         }, []);
19
```

Listing 5.4: Petición del nombre de las clases al servidor y se asigna a una variable

La URL redirige a una función del fichero *views.py* y en esta función se extrae de la base de datos el nombre de todas las clases a las que pertenece el profesor para ser enviadas en formato JSON al componente de React.

```
def get_classes(request):
    #se obtiene el identificador del profesor que realiza la peticion
    user_id = request.user.id
    #se obtienen todas las clases a las que pertenece el profesor
    classes_query = Classes.objects.filter(usuarios=user_id)
    #se crea una lista a la que se van anadiendo todos lo nombres de clas clases
    classes = []
    for c in classes_query:
        classes.append(c.nombre_clase)
    #se devuelve la lista de nombres en formato JSON
    return JsonResponse({'classes': classes})

```

Listing 5.5: Extraer las clases de la base de datos

- **Desplegable para seleccionar un alumno:** para poder elegir el alumno es necesario haber elegido la clase a la que pertenece previamente para facilitar la búsqueda. Cuando se ha seleccionado la clase a la que pertenece el alumno, se realiza otra petición al servidor mediante una URL similar a la anterior pero se envía el nombre de la clase de la que se quiere saber el nombre de los alumnos que pertenecen a ella.

  El proceso es similar al caso anterior, la URL redirige a una función del fichero *views.py*, la cual extrae todos los usuarios que pertenecen a la clase y envía en formato JSON el nombre de los usuarios.

```
def get_students_class(request, nameclass):
    students = []
    #si se ha seleccionado una clase entra en el if
    if nameclass != "sinseleccion":
        #se obtiene el objeto de la clase que se ha seleccionado
        c = Classes.objects.get(nombre_clase=nameclass)
        #se sacan todos los usuarios que pertenecen a la clase
        users = c.usuarios.all()
        #para cada usuario se extrae su nombre de usuario y se anade a la lista
        for u in users:
            username = u.username
            if username not in students:
                students.append(username)
```

```
14                  #se devuelve una lista con todos los nombre en
                        formato JSON
15                  return JsonResponse({'students': students})
16
```

Listing 5.6: Extraer las clases de la base de datos

Cuando un alumno es seleccionado se repite el mismo procedimiento pero esta vez para adquirir el código del alumno. El código de los alumnos se encuentra almacenado en unos servidores de Amazon Web Service. El procedimiento es el siguiente:

1. Se realiza una petición mediante una URL, contemplada en el fichero *urls.py* que incluye el alumno seleccionado y el ejercicio en el que se encuentra el profesor

2. La URL redirige a la función correspondiente del fichero *views.py*. La función extrae los datos del ejercicio y del usuario del que se quiere el código. Posteriomente, se hace una petición a los servidores de AWS con la información del usuario y ejercicio en cuestión para que devuelva el código correspodiente. La petición a AWS se realiza mediante una función ya existente para ello.

3. La función retorna a React el código en formato JSON que ha recibido.

4. El componente de React muestra el código en el editor mediante la función *showUserCode*. Esta función obtiene el editor en el que se muestra el código y fija el valor del editor con el enviado al componente.

```
1           function showUserCode(code){
2               const editorele = ace.edit("code");
3               editorele.setValue(code)
4           }
5
```

Listing 5.7: Función para mostrar el código en el editor

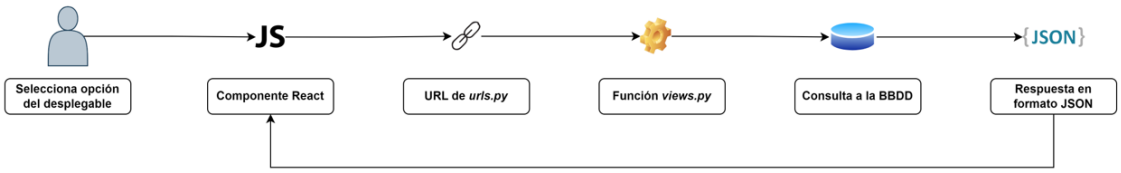

Figura 5.7: Flujo para mostrar la información de los desplegables y el código

Una vez realizado todo este flujo para obtener todos los datos necesarios, el profesor puede visualizar el código del alumno o ejecutarlo para comprobar su funcionamiento. Esto permite una evaluación de forma rápida y eficaz, sin la necesidad de tener que recurrir a otras herramientas para adquirir el código del alumnado.

5.3. Validación experimental

Las modificiones mencionadas en el presente capítulo, desarrolladas e implementadas en Unibotics, suponen una gran mejora en la web al agregar nuevas opciones para aquellos usuarios que sean profesores. Hasta ahora, un profesor solo podía desarrollar y ejecutar su propio código al igual que el resto de usuarios al no existir diferenciación de roles. Con estas opciones para profesores se consigue tener un mayor control sobre sus clases y alumnos.

Para validar experimentalmente estas funcionalidades, se van a enumerar algunos casos de uso de reales a los que se enfrentaría un profesor que accede a la plataforma.

CASO DE USO 1: Visualización de actividad monitorizada

- **Objetivo:** Un usuario tiene varias clases con diferentes alumnos y quiere visualizar la actividad de una de ellas y de un alumno en el último mes.
- **Autor:** Un usuario con rol de profesor que tiene dos clases.
- **Acciones:** descripción del flujo que seguiría la acción.

  1. El profesor en cuestión accede a su cuenta de Unibotics.
  2. Una vez dentro de su cuenta, despliega el menú superior derecho y accede a la sección *Analytics*.
  3. El sistema comprueba su rol para aceptar o denegar el acceso al apartado de analíticas. Además en función de su rol, se le muestran unos menús u otros.
  4. El usuario al tener rol de profesor se acepta su acceso y se le muestra únicamente el menú correspondiente a *Clases y alumnos*.
  5. Cuando hace click en dicho menú, se despliegan las subsecciones. En este caso solo existe la subsección *Actividad*.
  6. Al acceder a la actividad, el sistema recopila todos los datos de actividad de sus clases y alumnos, y los nombres de las clases y los alumnos para mostrarlos en los desplegables. Tras esto, se le muestran al profesor.
  7. Para ver los datos de la clase, el profesor selecciona el rango de fechas del último mes y la clase que quiere analizar en el apartado para *Clases*.
  8. Cuando selecciona las fechas, el sistema recarga los datos llamando a la función *Activity* pasándole los datos de fecha de inicio y fin. De igual forma, cuando se selecciona una clase, el sistema extrae los alumnos que pertenecen a la clase, recopila la actividad de todos ellos y la muestra de forma conjunta.

Figura 5.8: Datos de actividad de una clase seleccionada en el último mes

9. Para ver los datos del alumno, el profesor selecciona el rango de fechas del último mes y el alumno al que quiere analizar su rendimiento en el apartado para *Alumnos*.
10. De forma similar a las clases, el sistema recopila los datos de actividad del alumno enviando la fecha de inicio y fin seleccionada a la función *Activity*. Cuando selecciona un alumno sigue el mismo proceso enviando el nombre del alumno para filtrar únicamente los datos de este.

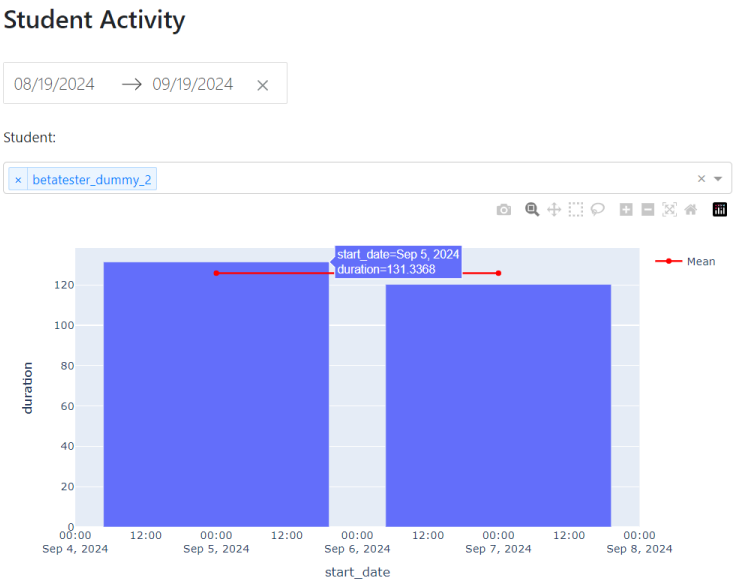

Figura 5.9: Datos de actividad de un alumno seleccionado en el último mes

Casos de uso con un procedimiento idéntico serían comparar la actividad de dos o más clases o alumnos. En estos casos, los pasos serían los mismos y el profesor podría hacer una comparativa de las medias de actividad o de los dias que hacen uso de la plataforma las diferentes clases y alumnos.

CASO DE USO 2: Visualizar y ejecutar el código de un alumno

- **Objetivo:** Un usuario que tiene varias clases quiere probar el código de un ejercicio realizado por dos alumnos de una misma clase.
- **Autor:** Un usuario con rol de profesor que tiene varias clases.
- **Acciones:** descripción del flujo que seguiría la acción.

  1. El profesor inicia sesión en la página web y accede a la página.
  2. El sistema comprueba los permisos que tiene el usuario para mostrarle únicamente los cursos y ejercicios que tiene permitidos.
  3. El usuario accede a la página donde se muestran los ejercicios y cursos disponibles.
  4. El usuario entra en un curso y hace click en el ejercicio en el que quiere probar el código de dos de sus alumnos.
  5. Al entrar en el ejercicio, el sistema comprueba el rol que tiene el usuario. Al comprobar su rol de profesor, se le muestran los dos desplegables para seleccionar clase y alumno en la página del ejercicio. Además, comprueba qué clases tiene dicho profesor para rellenar los datos del primer desplegable.
  6. El profesor selecciona la clase en la que se encuentran los alumnos correspondientes.
  7. El sistema extrae de la base de datos todos los alumnos pertenecientes a la clase seleccionada para rellenar los datos del segundo desplegable.
  8. El profesor selecciona el primer alumno del listado de alumnos.
  9. El sistema obtiene el código relativo al estudiante seleccionado y se le muestra en el editor del ejercicio.
  10. El profesor ya puede observar el código del alumno para analizarlo y da al botón de ejecutar para que comience a funcionar.
  11. Para el segundo alumno, al ser de la misma clase, seleccionaría de nuevo el alumno y seguiría el mismo proceso. En el caso de que el segundo alumno fuera de una clase distinta, se volvería a seleccionar la clase repitiendo el procedimiento desde el punto 6.

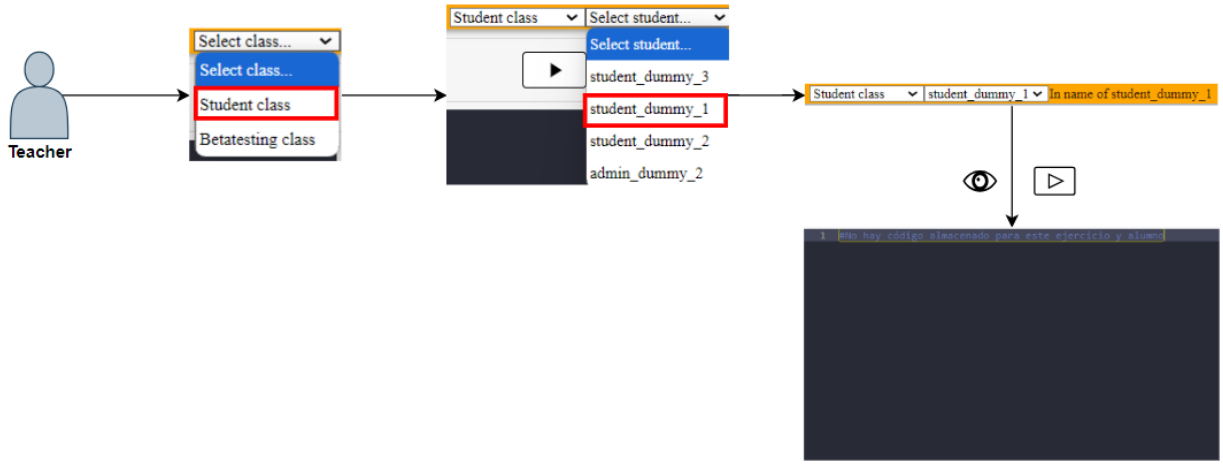

Figura 5.10: Flujo para visualizar y ejecutar el código

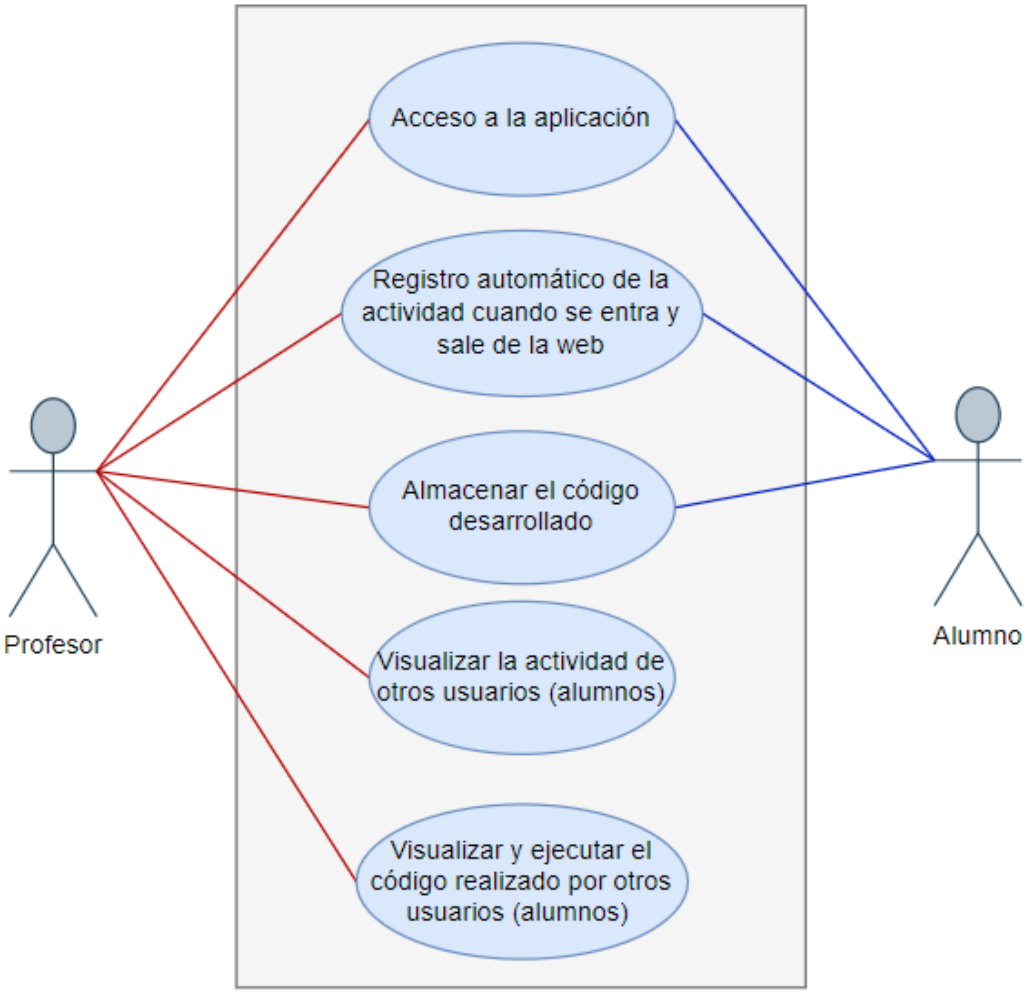

Figura 5.11: Diagrama distintivo de casos de uso de alumnos y profesores

Capítulo 6

Conclusiones

En este último capítulo se hace referencia a las conclusiones finales de este Trabajo de Fin de Grado tales como las mejoras que ha supuesto para Unibotics, los conocimientos adquiridos gracias a la realización del presente proyecto y posibles líneas de mejora en un futuro.

6.1. Conclusiones

Haciendo recapitulación, la base del proyecto se centra en la plataforma web Unibotics diseñada para la enseñanza de robótica a universitarios. La versión sobre la que se ha trabajado ha sido Unibotics 3.11, la cual no tenía diferenciación de usuarios exceptuando aquellos que eran administradores. Además, los contenidos de la plataforma eran ejercicios a los que todos los usuarios registrados tenían acceso. El objetivo de este TFG ha sido dotar a la platafroma web de algunas mejoras para la gestión de los usuarios y los contenidos, así como crear algunas funcionalidades específicas para profesores que facilitan la labor del docente.

En cuanto a las mejoras en la gestión de usuarios y contenidos, han aparecido algunos conceptos que antes no estaban presentes en Unibotics tales como Roles, Clases, Cursos y Permisos. Los roles han permitido hacer una diferenciación entre los diferentes usuarios, alumnos y profesores. Las clases y los cursos han facilitado la agrupación de usuarios y ejercicios respectivamente para facilitar la gestión. Los permisos han permitido señalar a qué contenidos tiene acceso cada usuario y cada grupo de usuarios, pudiendo asignarlos tanto de forma individual como colectiva. Todos estos conceptos han sido incluidos en nuevas tablas de la base de datos.

Con la aparición de los cursos ha sido necesario crear un nuevo formato de la página web. Por un lado, la página principal antes mostraba todos los ejercicios disponibles pero actualmente, con los nuevos desarrollos, se muestran los ejercicios y los cursos disponibles para el usuario tras hacer una comprobación de los permisos. Por otro lado, ha sido necesario crear una nueva página para cada curso que incluya el título, la descripción y los ejercicios del curso, ya que antes no existía este concepto en Unibotics. Además, se ha añadido la posibilidad de elegir el orden de los ejercicios dentro de un curso con el objetivo de indicar a los usuarios de forma orientativa en

qué orden deben realizar los ejercicios de un curso.

En relación a las funcionalidades diseñadas específicamente para profesores, permiten un mayor control de los docentes sobre el trabajo realizado y las horas que dedican sus alumnos. El trabajo realizado pueden comprobarlo accediendo al código de sus alumnos para ejecutarlo y pueden observar las horas dedicadas gracias a la sección para profesores creada en Dash. Estas herramientas suponen una mejora, ya que facilitan el trabajo de los profesores que usen la plataforma a la hora de evaluar a los estudiantes haciendo un balance entre el tiempo dedicado y el resultado obtenido.

Como conclusión, en este Trabajo de Fin de Grado se han cumplido tanto los objetivos generales como los más específicos que se marcaron en el capítulo 2. Se han incluido dos funcionalidades para profesores, han aparecidos los roles y se ha mejorado la gestión de los contenidos. Estas contribuciones aportan flexibilidad a la página web de Unibotics, haciendo de ésta una plataforma personalizable, ya que cada usuario ve únicamente lo que necesita o lo que tiene permitido, pudiendo habilitar y restringir el acceso a los contenidos y las diferentes funcionalidades.

6.2. Competencias adquiridas

En el desarrollo de este TFG se han adquirido múltiples conocimientos que antes eran desconocidos y, a su vez, se han mejorado los conocimientos existentes.

- En cuanto a Python y Django, previamente tenía algunos conocimientos básicos. Sin embargo, en el transcurso del proyecto he ampliado estos conocimientos orientados al desarrollo web.
- Dash ha sido completamente nuevo para mí, me ha permitido conocer cómo se manejan y recopilan los datos y su posterior integración en las diferentes gráficas que ofrece este framework.
- Con PostgreSQL he ampliado las habilidades que ya tenía de bases de datos, ya que he comprendido mejor su funcionamiento, la forma en que se relacionan unas tablas con otras y su extracción de forma adecuada para trabajar con los datos.
- En relación al frontend, React también era desconocido para mí. Gracias a esta tecnología he aprendido cómo se comportan los diferentes componentes, cómo se transfiere la información entre ellos y cómo se hacen las peticiones al servidor mediante el uso de URLs.
- En cuanto a HTML5, conocía en profundidad esta tecnología gracias a su uso en diversas asignturas del grado. Sin embargo, he aprendido cómo incluir los diferentes scripts de REACT en los diferentes templates HTML.
- Los contenedores e imágenes de Docker nunca los había utilizado. En el proyecto me he familiarizado con ellos por la necesidad de descargar imágenes y ejecutarlas en un contenedor para poder acceder a la página de los ejercicios.

- Acerca de GitHub y el uso de repositorios tenía algunas nociones básicas. En este proyecto he aprendido a crear incidencias y diferentes ramas para subir mis cambios al repositorio donde se encontraba todo el código de Unibotics.
- Por último, Unibotics era absolutamente nuevo para mí, nunca había utilizado Unibotics ni conocía su existencia. Me ha permitido conocer cómo funciona una aplicación real con sus diferentes entornos de desarrollo y cómo se trabaja con otros desarrolladores.

6.3. Trabajos futuros

Por último, se proponen posibles trabajos futuros o mejoras que parten de lo desarrollado en este proyecto.

- Añadir más gráficas con nuevos datos para profesores. De esta forma los profesores pueden adquirir datos más específicos de sus alumnos como el tiempo en cada ejercicio o curso.
- La página de cada curso podría mejorarse añadiendo vídeos explicativos o recursos necesarios, creando módulos de ejercicios o introduciendo un foro de discusión para cada curso
- En la página de cada ejercicio se puede enlazar una solución de referencia visible sólo si el rol del usuario es profesor.
- Cuando un profesor accede a los ejercicios para revisar y ejecutar el código de sus alumnos, sería posible añadir una sección en la que el profesor deje comentarios al alumno sobre su trabajo.

Bibliografía

[1] *Real Academia Española*. URL: https://www.rae.es/.

[2] *Concepto de robótica*. URL: https://concepto.de/robotica/.

[3] *¿Qué es el middleware?* URL: https://bambu-mobile.com/que-es-el-middleware-en-arquitectura-del-software/.

[4] *Tipos de Middleware*. URL: https://es.wikibrief.org/wiki/Robotics_middleware.

[5] *¿Qué es ROS?* URL: https://openwebinars.net/blog/que-es-ros/.

[6] *Página oficial de ROS*. URL: https://www.ros.org/.

[7] *Página y documentación de YARP*. URL: https://www.yarp.it/.

[8] *Página y documentación de MRDS*. URL: https://learn.microsoft.com/en-us/archive/msdn-magazine/2008/june/robotics-simulating-the-world-with-microsoft-robotics-studio.

[9] *Página y documentación de Gazebo*. URL: https://gazebosim.org/.

[10] *Página y documentación de CoppeliaSim*. URL: https://www.coppeliarobotics.com/.

[11] *Página y documentación de WeBots*. URL: https://cyberbotics.com/.

[12] *Página y documentación de MATLAB Robotics System Toolbox*. URL: https://es.mathworks.com/products/robotics.html.

[13] *Página oficial de The Construct*. URL: https://www.theconstruct.ai/home/.

[14] *Página oficial de Riders.ai*. URL: https://riders.ai/.

[15] *Página oficial de Asimovo*. URL: https://asimovo.com/.

[16] *Página oficial Unibotics*. URL: https://unibotics.org/.

[17] *Tecnologías web*. URL: https://estudyando.com/que-es-la-tecnologia-web-definicion-y-tendencias/.

[18] *Tecnologías del cliente y del servidor*. URL: https://www.proun.es/blog/tecnologias-web-actuales/.

[19] *Blog desarrollado a lo largo del trabajo de fin de grado*. URL: https://roboticslaburjc.github.io/2023-tfg-iraide-hoyas/.

[20] *Documentación de Python*. URL: https://docs.python.org/.

[21] *Documentación de Django*. URL: https://www.djangoproject.com/.

[22] *Información acerca de la estructura de Django, MVT*. URL: https://www.adrformacion.com/knowledge/programacion/_como_es_la_estructura_de_django_.html.

[23] *Documentación de Dash*. URL: https://plotly.com/dash/.

[24] *Documentación de JavaScript*. URL: https://developer.mozilla.org/es/docs/Web/JavaScript.

[25] *Documentación de React*. URL: https://es.react.dev/.

[26] *Documentación de CSS*. URL: https://developer.mozilla.org/es/docs/Web/CSS.

[27] *Documentación de HTML5*. URL: https://developer.mozilla.org/es/docs/Web/HTML.

[28] *Documentación de PostgreSQL*. URL: https://www.postgresql.org/about/.

[29] *Documentación de Docker*. URL: https://www.docker.com/resources/what-container/.

[30] *Documentación de GitHub*. URL: https://docs.github.com/es.

[31] David Roldán-Álvarez, José M Cañas, David Valladares, Pedro Arias-Perez y Sakshay Mahna. «Unibotics: open ROS-based online framework for practical

learning of robotics in higher education». En: *Multimedia Tools and Applications* 83.17 (2024), págs. 52841-52866. URL: https://doi.org/10.1007/s11042-023-17514-z.