GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Diseño de una infraestructura digital escalable para la gestión distribuida y análisis de datos de un proyecto de innovación docente

Autor: María Oliva Calero

Director: David Alfaya Sánchez

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Diseño de una infraestructura digital escalable para la gestión distribuida y análisis de datos de un proyecto de innovación docente

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2022/23 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

María Oliva Calero

Fdo.: María Oliva Calero Fecha: 26/ 06/ 2023

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: David Alfaya Sánchez Fecha: 26/ 06/ 2023

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Diseño de una infraestructura digital escalable para la gestión distribuida y análisis de datos de un proyecto de innovación docente

Autor: María Oliva Calero

Director: David Alfaya Sánchez

Madrid

Agradecimientos

A mi familia por acompañarme estos cuatro años de grado, a mis amigas de Milán por apoyarme durante todo el proyecto y a mi tutor, David por guiarme y aconsejarme.

DISEÑO DE UNA INFRAESTRUCTURA DIGITAL ESCALABLE PARA LA GESTIÓN DISTRIBUIDA Y ANÁLISIS DE DATOS DE UN PROYECTO DE INNOVACIÓN DOCENTE

Autor: Oliva Calero, María.
Director: Alfaya Sánchez, David.
Entidad Colaboradora: Proyecto "Maths Team Contest: Refinamiento y ampliación de metodologías docentes" (proyecto 2122-22 del programa 21-22 de la Universidad Pontificia Comillas para el desarrollo de proyectos de innovación docente), ICAI, Universidad Pontificia Comillas.

RESUMEN DEL PROYECTO

El principal objetivo del proyecto es el diseño e implementación de una plataforma escalable que de soporte a las actividades docentes desarrolladas en el Maths Team Contest (MTC). La arquitectura final interconecta varios subsistemas preexistentes en un único software de gestión.

Palabras clave: Sistema Software, Codocencia, Moodle, Django

1. Introducción

El Trabajo de Fin de Grado desarrollado se enmarca en la beca de colaboración en Innovación docente realizada en ICAI en el curso 2022-2023. La beca se basa el desarrollo del proyecto docente Maths Team Contest (MTC, proyecto 2122-22 del programa 21-22 de la Universidad Pontificia Comillas para el desarrollo de proyectos de innovación docente), iniciativa conjunta de las asignaturas de Álgebra y Geometría y Análisis Matemático y Cálculo Vectorial del grado de Ingeniería Matemática e Inteligencia Artificial (iMAT). El proyecto desarrollado consiste en el diseño y despliegue de una solución que permita la gestión del concurso MTC.

2. Definición del proyecto

El Trabajo desarrollado busca solventar las limitaciones que presentaba el sistema de gestión antiguo que se utilizaba en el MTC. Tras realizar un análisis de estos problemas y de los requisitos formales que debe cumplir la nueva plataforma, se ha diseñado una arquitectura que permite la gestión distribuida y escalable de la actividad del concurso. Esta solución se ha desarrollado e implementado siguiendo un ciclo iterativo e incremental de diseño de software.

El Sistema antiguo implementaba la figura de un coordinador que realizaba tareas de forma manual, la corrección de problemas directamente sobre Moodle, y la comunicación oral entre los profesores participantes en el concurso, todo ello limitando la escalabilidad y usabilidad del mismo.

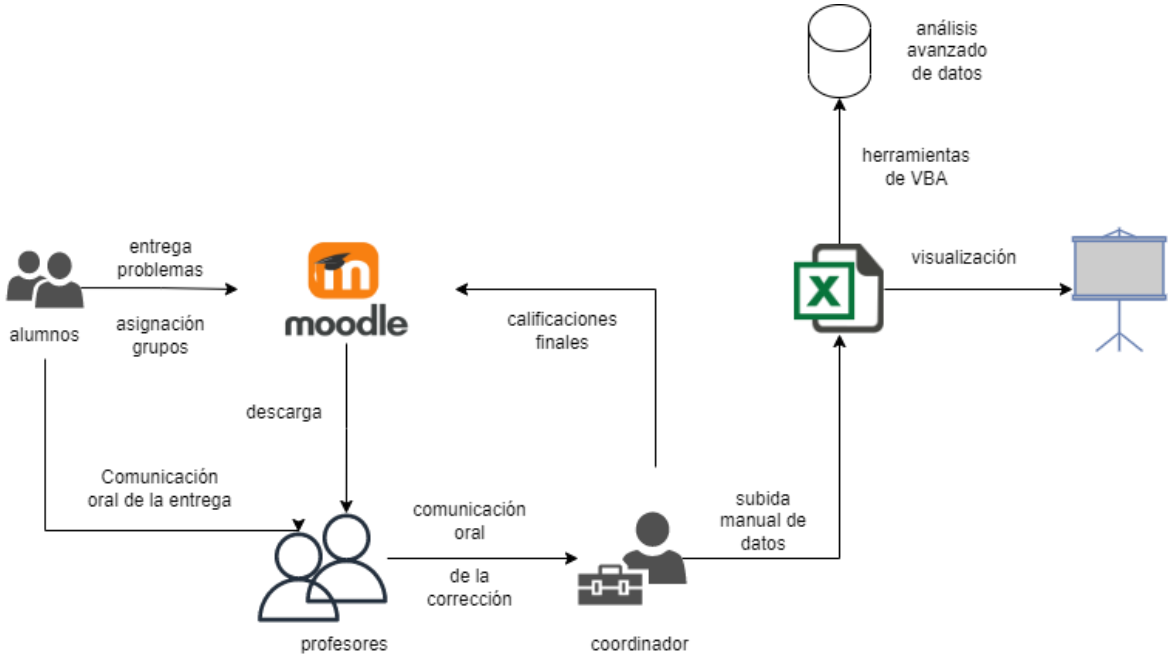


Ilustración 1: Sistema de gestión antiguo

3. Descripción del sistema

El nuevo sistema solventa estas limitaciones con una arquitectura basada en Django + SQLite3 desplegado en un servidor Apache de la Universidad. Mediante un Bot con credenciales limitados de profesor, se ha diseñado un sistema de crawling que sincroniza los datos de Moodle con los de la plataforma en tiempo real. El sistema central detecta las subidas de entregas que realizan los alumnos en Moodle a través del crawler y gestiona la prioridad de corrección, las puntuaciones/feedback de los profesores, la descarga de los archivos entregados, la adjudicación de puntuaciones temporales y la actualización de los rankings.

El sistema suministra información en tiempo real a las pantallas del Comillas Conecta Lab, permitiendo seguir la evolución de las pruebas y el feedback de los profesores simultáneamente en todas las pantallas. Además, se ha implementado una interfaz de datos que permite que sistemas preexistentes puedan crear automáticamente material complementario y realizar exportación de los datos del concurso para su análisis.

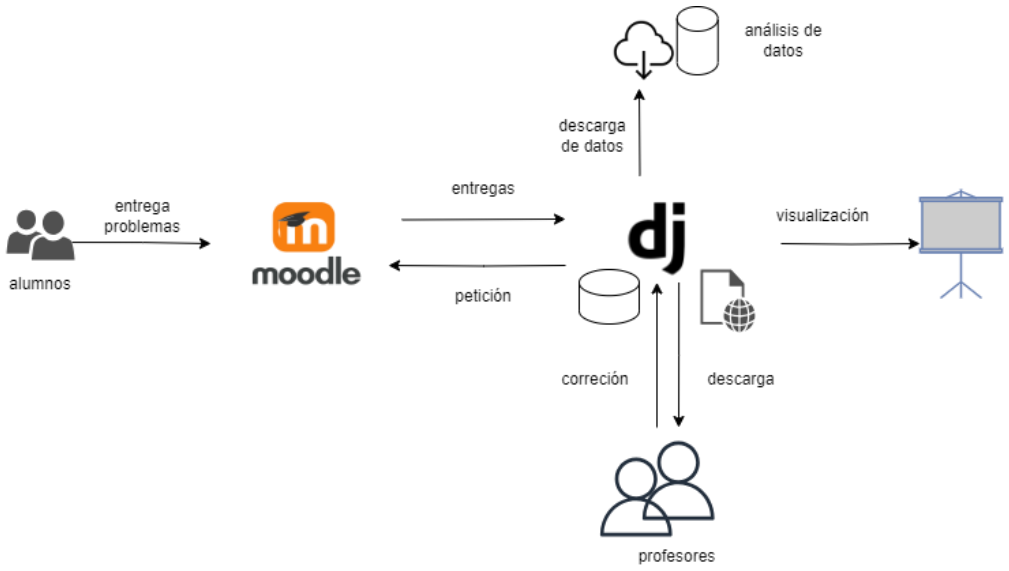


Ilustración 2: Sistema de gestión del MTC desarrollado

4. Resultados

El resultado es una arquitectura de alto nivel que permite el acceso a través de un cliente web y une la gestión de la actividad del concurso en una sola plataforma, manteniendo Moodle como interfaz de entrega de los problemas realizados por los alumnos.

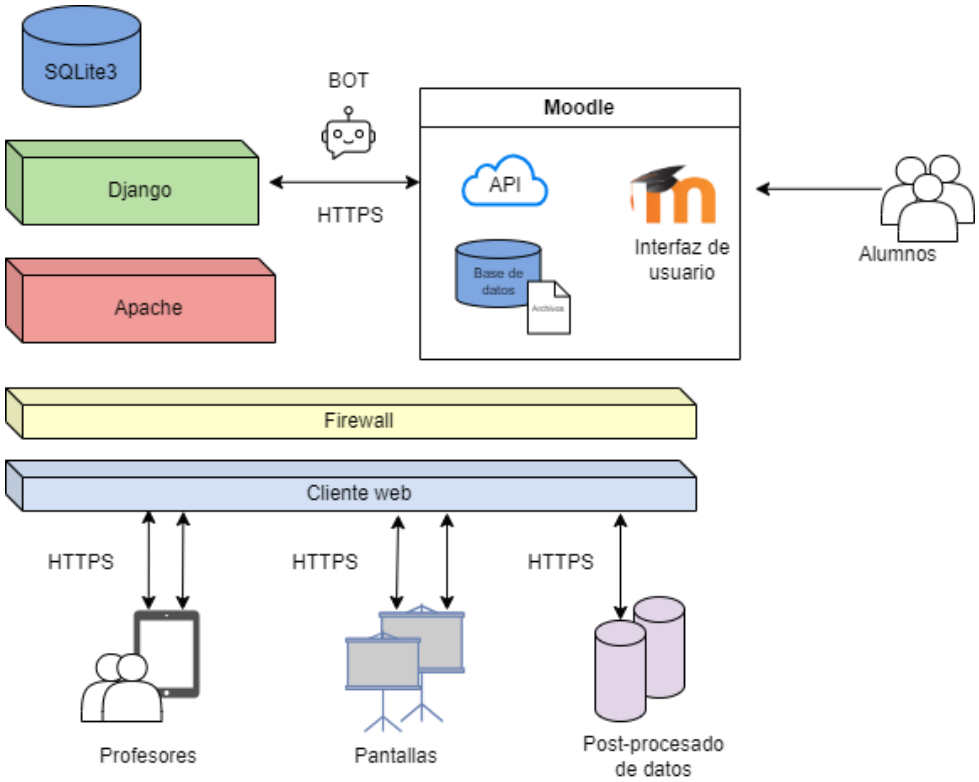


Ilustración 3: Arquitectura de alto nivel

5. Conclusiones

En conclusión, se ha conseguido diseñar una arquitectura que solventa las limitaciones que planteaba el sistema antiguo de gestión del Maths Team Contest, implementando con éxito la plataforma desarrollada.

Este sistema se ha utilizado para la realización de todas las pruebas del MTC en el curso académico 2022-23, evaluando su funcionamiento en un entorno real con casi 100 alumnos y 5 profesores, gestionando un total de 1680 problemas entregados.

Los resultados del proyecto se han presentado, además, en la Jornada de Buenas Prácticas en Innovación Docente 2023 de la Universidad Pontificia de Comillas.

DESIGN OF A SCALABLE DIGITAL INFRASTRUCTURE FOR THE DISTRIBUTED MANAGEMENT AND DATA ANALYSIS OF A TEACHING INNOVATION PROJECT

Author: Oliva Calero, María.
Supervisor: Alfaya Sánchez, David.
Collaborating Entity: Project "Maths Team Contest: Refinement and extension of teaching methodologies" (project 2122-22 of the program 21-22 of the Universidad Pontificia Comillas for the development of teaching innovation projects), ICAI, Universidad Pontificia Comillas.

ABSTRACT

The main objective of the project is the design and implementation of a scalable platform to support the teaching activities developed in the Maths Team Contest (MTC). The final architecture interconnects several pre-existing subsystems into a single management software.

Keywords: Software System, Co-teaching, Moodle, Django

1. Introduction

The Final Degree Project developed is part of the collaboration grant in Teaching Innovation carried out in ICAI in the academic year 2022-2023. The grant is based on the development of the teaching project Maths Team Contest (MTC, project 2122-22 of the program 21-22 of the Universidad Pontificia Comillas for the development of teaching innovation projects), a joint initiative of the subjects of Algebra and Geometry and Mathematical Analysis and Vector Calculus of the degree in Mathematical Engineering and Artificial Intelligence (iMAT). The project consists of the design and deployment of a solution that allows the management of the MTC contest.

2. Project definition

The project developed seeks to solve the limitations of the old management system used in the MTC. After analysing these problems and the formal requirements to be met by the new platform, an architecture has been designed that allows the distributed and scalable management of the contest activity. This solution has been developed and implemented following an iterative and incremental software design cycle.

The old system implemented the figure of a coordinator who performed tasks manually, the correction of problems directly on Moodle, and the oral communication between the teachers participating in the contest, all of which limited the scalability and usability of the system.

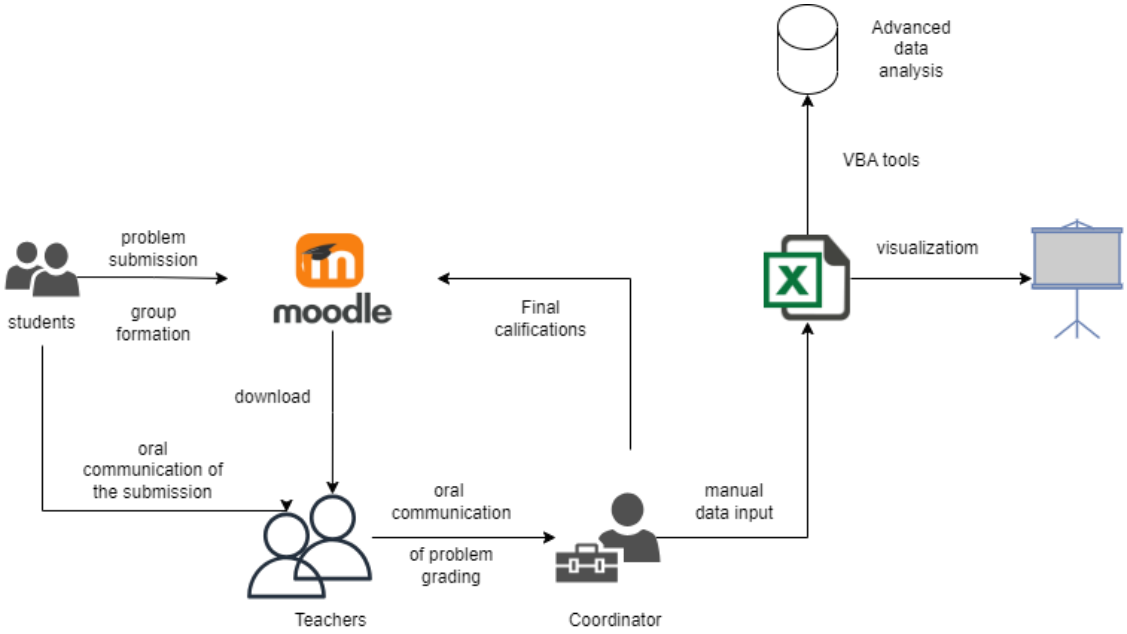


Figure 4: Deprecated management system

3. System description

The new system overcomes these limitations with an architecture based on Django + SQLite3 deployed on an Apache server at the University. Using a Bot with limited teacher credentials, a crawling system has been designed that synchronizes Moodle data with those of the platform in real time. The central system detects student uploads of submissions to Moodle through the crawler and manages the correction priority, teacher scores/feedback, downloading of submitted files, awarding of temporary scores and updating of rankings.

The system provides real-time information to the Comillas Conecta Lab screens, allowing to follow the evolution of the tests and the teachers' feedback simultaneously on all screens. In addition, a data interface has been implemented that allows pre-existing systems to automatically create supplementary material and export the contest data for advanced analysis.

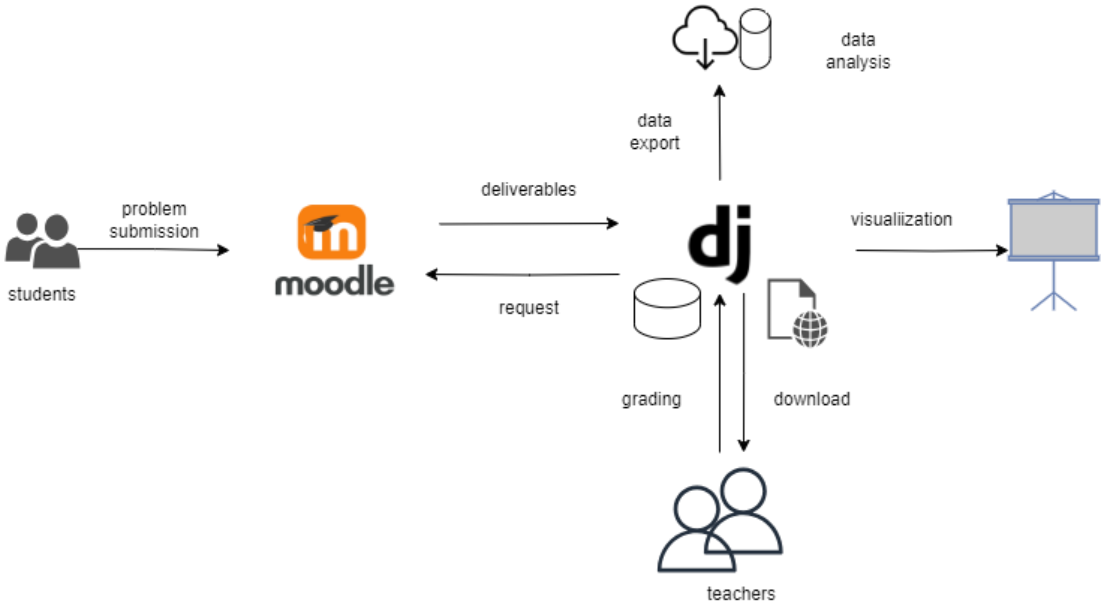


Figure 5: MTC Management System developed

4. Results

The result is a high-level architecture that allows access through a web client and unites the management of the contest activity in a single platform, keeping Moodle as the delivery interface for the problems done by the students.

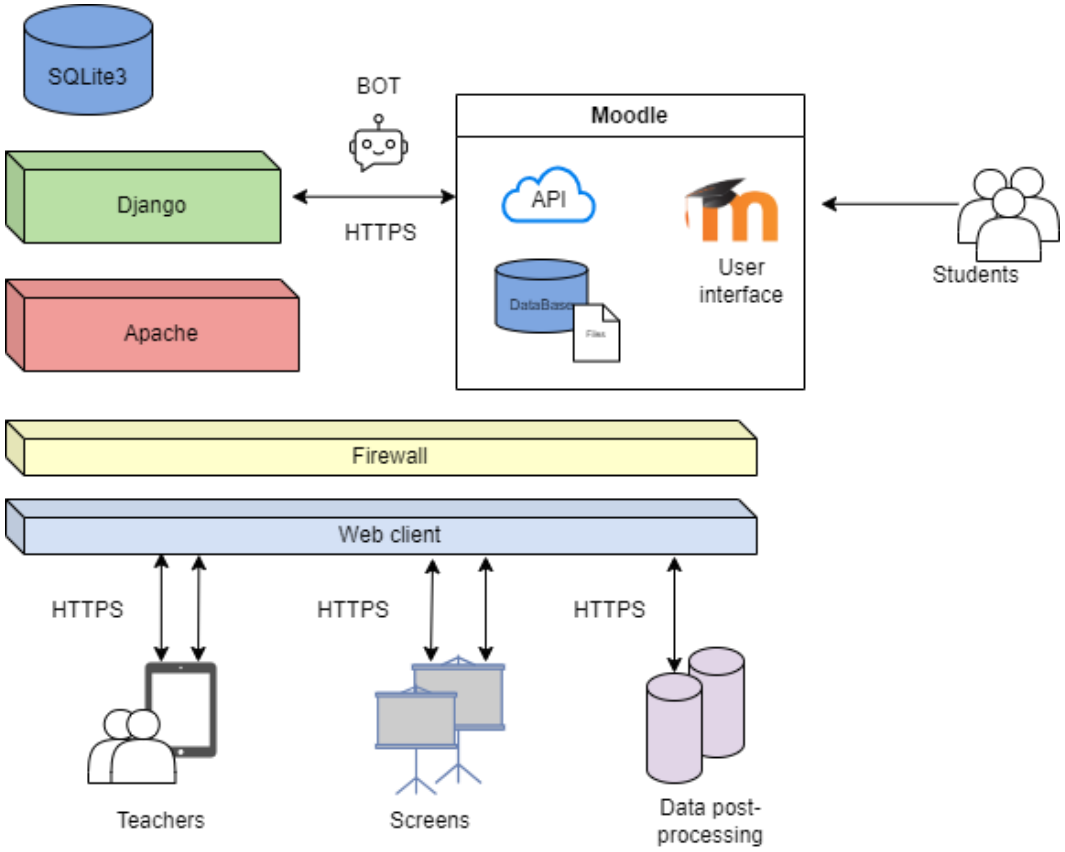


Figure 6: High-level architecture

5. Conclusions

In conclusion, it has been possible to design an architecture that overcomes the limitations posed by the old Maths Team Contest management system, successfully implementing the developed platform.

This system has been used to carry out all the MTC tests in the 2022-23 academic year, evaluating its performance in a real environment with almost 100 students and 5 teachers, managing a total of 1680 problems submitted.

The results of the project were also presented at the Conference on Best Practices in Teaching Innovation 2023 at the Universidad Pontificia de Comillas.

Índice de la memoria

Índice de figuras

Índice de tablas

1. INTRODUCCIÓN

El Trabajo de Fin de Grado desarrollado se enmarca en la beca de colaboración en Innovación docente realizada en ICAI en el curso 2022-2023. La beca se basa el desarrollo del proyecto docente Maths Team Contest (MTC, proyecto 2122-22 del programa 21-22 de la Universidad Pontificia Comillas para el desarrollo de proyectos de innovación docente), iniciativa conjunta de las asignaturas de Álgebra y Geometría y Análisis Matemático y Cálculo Vectorial del grado de Ingeniería Matemática e Inteligencia Artificial (iMAT).

El proyecto MTC organiza un concurso matemático por equipos para las dos asignaturas mencionadas. Su objetivo principal es incrementar el rendimiento en competencias matemáticas y de pensamiento abstracto de los alumnos de grado, mejorando sus habilidades interpersonales, como gestión de tiempo, trabajo en equipo o comunicación. En este contexto, el MTC fomenta la colaboración, organización y comunicación entre docentes, explorando nuevas formas de innovación docente.

El concurso se estructura en 4 pruebas para cada una de las asignaturas, que se realizan en el Comillas Conecta Lab a lo largo del año escolar. En cada prueba, los alumnos compiten en grupo, resolviendo problemas que los profesores corrigen en tiempo real. Éstos dan feedback a los equipos, lo que supone una gran cantidad de datos a gestionar en tiempo real.

El MTC surgió en el curso académico 2021-2022. Durante este primer año de actividad, se emplearon varias herramientas, que conjuntamente y con la ayuda de un coordinador, que realizaba tareas de forma manual, servían como sistema de gestión. Ver Capítulo 0: Estado de la cuestión para más información sobre esta arquitectura.

A partir de la experiencia del año 21-22 y ante el crecimiento del número de alumnos y de profesores participantes, se identifican nuevas necesidades de gestión.

De estas necesidades de gestión surge el proyecto actual, creando una infraestructura digital y escalable que de soporte a la actividad del concurso.

Los resultados de este proyecto se han presentado en la Jornada de Buenas Prácticas en Innovación Docente 2023 de la Universidad Pontificia de Comillas, bajo el título: *Maths Team Contest: Refinamiento y Ampliación de Metodologías Docentes.*

MOTIVACIÓN

La motivación del proyecto es la automatización del sistema de gestión que da soporte al MTC que, hasta ahora, era realizado de forma manual. Ver Ilustración 9: Sistema de gestión durante el curso académico 2021-22.

Debido al volumen creciente de alumnos y datos que tiene el concurso, a la necesidad de coordinación entre un mayor número de profesores y a la utilización de entornos innovadores como es el Comillas Conecta Lab (2.1.2), en el que los alumnos se reparten en una sala amplia y donde es necesario retransmitir la información a varios juegos de pantallas independientes, es necesaria una plataforma que permita mejorar la comunicación profesor-profesor y profesor-alumno y la gestión descentralizada y concurrente de los datos en tiempo real por parte de los profesores a través de terminales móviles (tabletas).

OBJETIVOS

El objetivo principal de este trabajo de fin de grado es el diseño, desarrollo y despliegue de un sistema escalable de gestión distribuida y concurrente de datos académicos que de soporte a las actividades desarrolladas en el proyecto de innovación docente Maths Team Contest así como al análisis de los datos recopilados durante dichas actividades.

Dentro de este objetivo, se pueden distinguir los siguientes:

- Análisis de las limitaciones de la solución anterior y de los requisitos funcionales y no funcionales que debe cumplir el nuevo sistema.
- Estudio de diferentes infraestructuras que permitan dar solución al problema planteado.

- Diseño de una arquitectura que interconecte múltiples infraestructuras preexistentes en un único software de gestión que permita automatizar y distribuir el flujo de datos de la actividad.
- Creación de subsistemas que permitan la sincronización de datos con una plataforma de e-learning (Moodle).
- Diseño de un sistema de visualización que permita mostrar los resultados de la actividad en un entorno con pantallas múltiple (Comillas Conecta Lab).
- Implementación de un sistema que garantice la seguridad del dato
- Despliegue de la solución desarrollada para su uso en las pruebas del MTC del curso académico 2022-23.

ESTRUCTURA DEL DOCUMENTO

En el siguiente capítulo, *Planteamiento del problema,* se introduce el MTC y se realiza un estudio de las tecnologías utilizadas en la solución que daba soporte al concurso en el curso 2021-22. Tras analizar sus limitaciones y puntos de mejora, se ha realizado un análisis de requisitos del nuevo sistema.

En el 3. se establece el plan de trabajo seguido en este proyecto, que se basa en un ciclo iterativo e incremental de desarrollo de software, marcado por la celebración de las pruebas del concurso.

En el 4. se hace un análisis general de la solución implementada, comenzando con un Diseño general de la arquitectura de alto nivel y haciendo hincapié en los puntos importantes de esta como la Conexión con Moodle y la Ciberseguridad. Al final del capítulo se hace un análisis de las Arquitecturas descartadas y los motivos de descarte de cada una.

El Diseño de la Solución Software profundiza en cada uno de los submódulos de la solución propuesta y desarrollada, haciendo hincapié en la construcción de cada uno de ellos y en el flujo de datos de la plataforma.

El Capítulo 6 de este documento está dedicado al Desarrollo, Despliegue y Pruebas de la solución desarrollada, elaborando en los entornos de trabajo creados, la implementación y las Pruebas llevadas a cabo.

Finalmente, en el Conclusiones y trabajo Futuro se exponen las conclusiones de este proyecto y el futuro del mismo.

En el ANEXO II: Código Fuente puede consultarse el código de la solución propuesta.

NOTA: Con el fin de garantizar el anonimato de los alumnos y profesores participantes en el concurso y la protección de sus datos, en el presente documento se emplearán datos ficticios, desarrollados en un entorno acotado de pruebas.

2. PLANTEAMIENTO DEL PROBLEMA

MTC

El proyecto Maths Team Contest surge en el curso académico 2021-2022 de una iniciativa conjunta de las asignaturas de Álgebra y Geometría y Análisis Matemático y Cálculo Vectorial del grado de Ingeniería Matemática e Inteligencia Artificial (iMAT). El MTC tiene como principal objetivo incrementar el rendimiento en competencias matemáticas y de pensamiento abstracto de los alumnos de grado, así como mejorar algunas de sus habilidades interpersonales, como trabajo en equipo, comunicación, gestión de tiempo… mediante la organización de un concurso matemático por equipos coordinado entre las dos asignaturas. Como parte de ello, el proyecto fomenta la colaboración, organización y comunicación entre docentes, explorando nuevas formas de innovación docente.

Para la realización del concurso es necesaria un sistema de gestión que permita las actividades descritas anteriormente. De esta necesidad de gestión surge el proyecto actual.

Ilustración 7: Logo Maths Team Contest

2.1.1 FUNCIONAMIENTO

El Maths Team Contest se estructura en una serie de pruebas en las que los alumnos tienen una hora para resolver 10 problemas de la asignatura correspondiente, de diferente índole y dificultad.

Las pruebas del MTC se realizan cuatro veces al año, en el innovador espacio Comillas Conecta Lab, en el cual los alumnos se organizan en equipos, resolviendo problemas y acumulando puntos en distintas sesiones y retos colaborativos.

El desarrollo de las pruebas del concurso es el siguiente: [2]

Antes del comienzo de la primera prueba, todos los alumnos se registran en su equipo MTC tanto en la asignatura de Álgebra y Geometría como en el de Cálculo y Análisis Vectorial a través de Moodle.

1. Al comenzar una prueba, todos los grupos reciben 10 problemas, cada uno con su tarea asociada de Moodle (una tarea para el P1, otra para el P2…).
2. Tras completar un problema, un alumno sube la solución a la tarea de Moodle correspondiente, incluyendo los archivos necesarios para su corrección (.py, .mlx, .pdf, imágenes…). Estas tareas se sincronizan entre todos los integrantes del grupo: cualquier miembro puede añadir, modificar o eliminar archivos y todo lo que una persona modifique afecta a la entrega del equipo entero.
3. Al realizar la entrega de un problema, el sistema de puntuación añade, de forma temporal, 10 puntos de ese problema como “posibles” y actualizará el ranking general y las tablas de puntuaciones en directo.
4. Si el equipo ha sido uno de los dos primeros en realizar ese problema, se le adjudica, pendiente de verificación, un premio de velocidad de 5 puntos adicionales.
5. Durante la prueba, los profesores corrigen estas entregas y se actualizan los rankings y tablas para mostrar las calificaciones corregidas y, en su caso, adjudicar definitivamente los premios de velocidad.

6. Los alumnos reciben feedback durante la prueba a través de las pantallas que muestran diferentes estadísticas sobre su evolución en la prueba y el ranking. Además, los profesores pueden acercarse a la mesa de un grupo a resolver dudas o si ellos mismos necesitan una aclaración sobre los problemas subidos.
7. Tras la prueba, los profesores corrigen los problemas restantes presentados por los equipos que no se han corregido durante la prueba y asignan las puntuaciones definitivas.
8. Los alumnos pueden consultar los rankings actualizados y definitivos al finalizar la corrección de la prueba.
9. Se entregan premios a los campeones y subcampeones de cada prueba. Estos, junto con los premios por la clasificación final en el concurso, conforman la nota MTC de los alumnos. La nota MTC es parte de la nota final de la asignatura correspondiente, Álgebra y Geometría o Análisis Matemático y Cálculo Vectorial.

2.1.2 CONECTA LAB

El Comillas Conecta Lab es el espacio elegido para realizar las pruebas del Maths Team Contest, además de muchas otras actividades docentes de la Universidad. El Conecta Lab nace en abril de 2021 como parte del nuevo Plan Estratégico 2019-23 de la Universidad Pontificia de Comillas promover el espíritu innovador de la universidad para avanzar en la calidad de su formación. [7]

Así, el nuevo plan de la Universidad se centra en formar estudiantes con un amplio conjunto de habilidades, utilizando nuevas metodologías de enseñanza que ponen al alumno en el centro y estableciendo el denominado aprendizaje activo.

De este deseo de posicionar al alumno como protagonista de su propio aprendizaje, surge el Comillas Conecta Lab, un espacio pionero en el panorama universitario español en el que los docentes pueden desarrollar nuevas metodologías de enseñanza.

El Conecta Lab deja atrás las aulas tradicionales de escuelas y universidades para dar paso a espacios flexibles, que permitan adaptar el lugar de acuerdo con los proyectos o actividades que se desarrollen en él.

El espacio está compuesto por mesas y sillas con ruedas, paneles separadores, pizarras móviles y pantallas independientes permitiendo crear espacios de aprendizaje diferentes con los mismos recursos.

Por ello, el concurso MTC tiene lugar en este innovador "laboratorio", que da cabida a los casi 100 alumnos que participan en él. Además, durante el desarrollo de las pruebas del concurso, se utilizan las 5 pantallas independientes con las que cuenta el espacio como marcador para que, tanto alumnos como profesores, puedan ver la evolución de cada grupo en tiempo real.

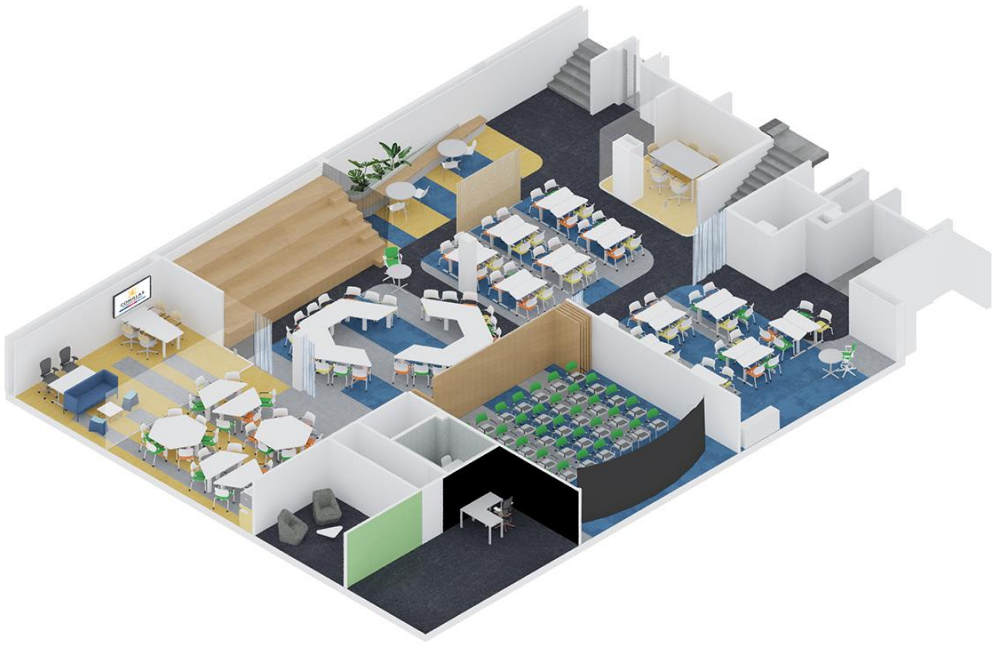


Ilustración 8: Comillas Conecta Lab [4]

ESTADO DE LA CUESTIÓN

Durante el curso 2021/22, se realizaron un total de ocho pruebas del MTC, divididas en cuatro fechas para las asignaturas de Álgebra y Análisis Matemático y Cálculo Vectorial.

En este apartado, revisaré la solución que existía para la gestión del concurso durante el curso 2021/22. Además, a partir de la experiencia del año 21-22 y ante el crecimiento del número de alumnos y datos, se identifican nuevas necesidades de gestión.

Una vez detallado el funcionamiento del concurso, podemos extraer un funcionamiento básico que nos ayude a entender el estado de la cuestión: los alumnos forman equipos y en cada prueba tratan de resolver problemas en el menor tiempo posible. Las entregas de estos problemas se realizan a través de Moodle, plataforma de enseñanza de la Universidad Pontificia de Comillas. Un grupo de profesores designado para el concurso corrige los problemas en tiempo real y da feedback a los alumnos. El feedback toma dos formas: por una parte, existe una comunicación verbal profesor-alumno para aclaraciones, resolución de dudas… Por otro lado, existe una visualización del progreso de cada grupo en la prueba a través de las pantallas del Conecta Lab.

Al tener unas características muy específicas, marcadas por la necesidad de utilizar Moodle como plataforma de entrega de los problemas, y del espacio multi-pantalla en el que se realizan las pruebas, no existía en el mercado un sistema que permitiese la completa gestión del concurso en una sola plataforma.

Es por esto que, durante el primer año de la realización del concurso, se emplearon varias herramientas, que conjuntamente y con la ayuda de un coordinador, que realizaba tareas de forma manual, servían como sistema de gestión.

Las herramientas de software utilizadas durante este primer año del MTC fueron Moodle, como plataforma de subida de entregas por parte de alumnos y de corrección por parte de los profesores, Excel como base de datos y para la visualización del progreso de cada prueba

y una serie de herramientas- Excel y VBA principalmente- para la gestión del dato post-prueba que beben del sistema centralizado.

Como podemos ver en la Ilustración 9, gran parte de las actividades se realizan de forma manual, surgiendo la necesidad de automatizar estos procesos. Cada uno de los sistemas empleados y sus funciones se detallan a continuación.

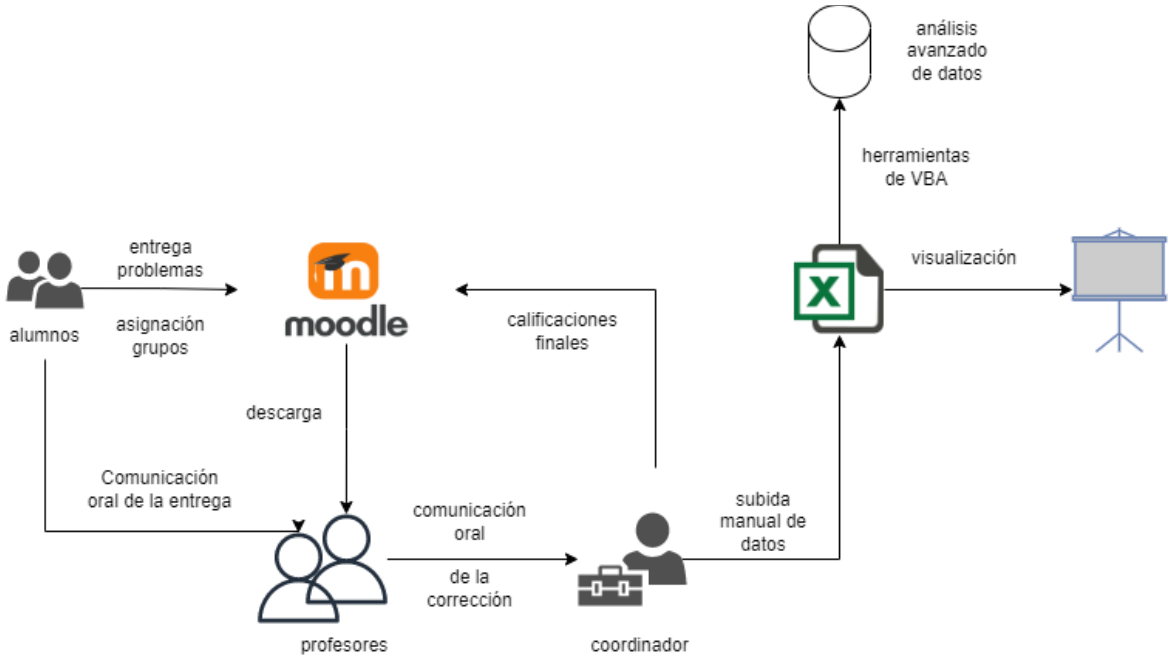


Ilustración 9: Sistema de gestión durante el curso académico 2021-22

Cabe destacar que la comunicación de los profesores con Moodle era totalmente asíncrona y bajo demanda. Una vez un alumno comunicaba la entrega de un problema, gritando "¡el equipo X ha hecho el problema Y!", el profesor accedía a la tarea del problema Y en Moodle, buscaba el problema del grupo X y descargaba los archivos entregados uno a uno para su corrección. Todo este proceso suponía una gran lentitud en las tareas de corrección, que sumado a la tardanza de la interfaz de Moodle cuando hay una gran carga de trabajo, hacían necesario un sistema de corrección más ágil.

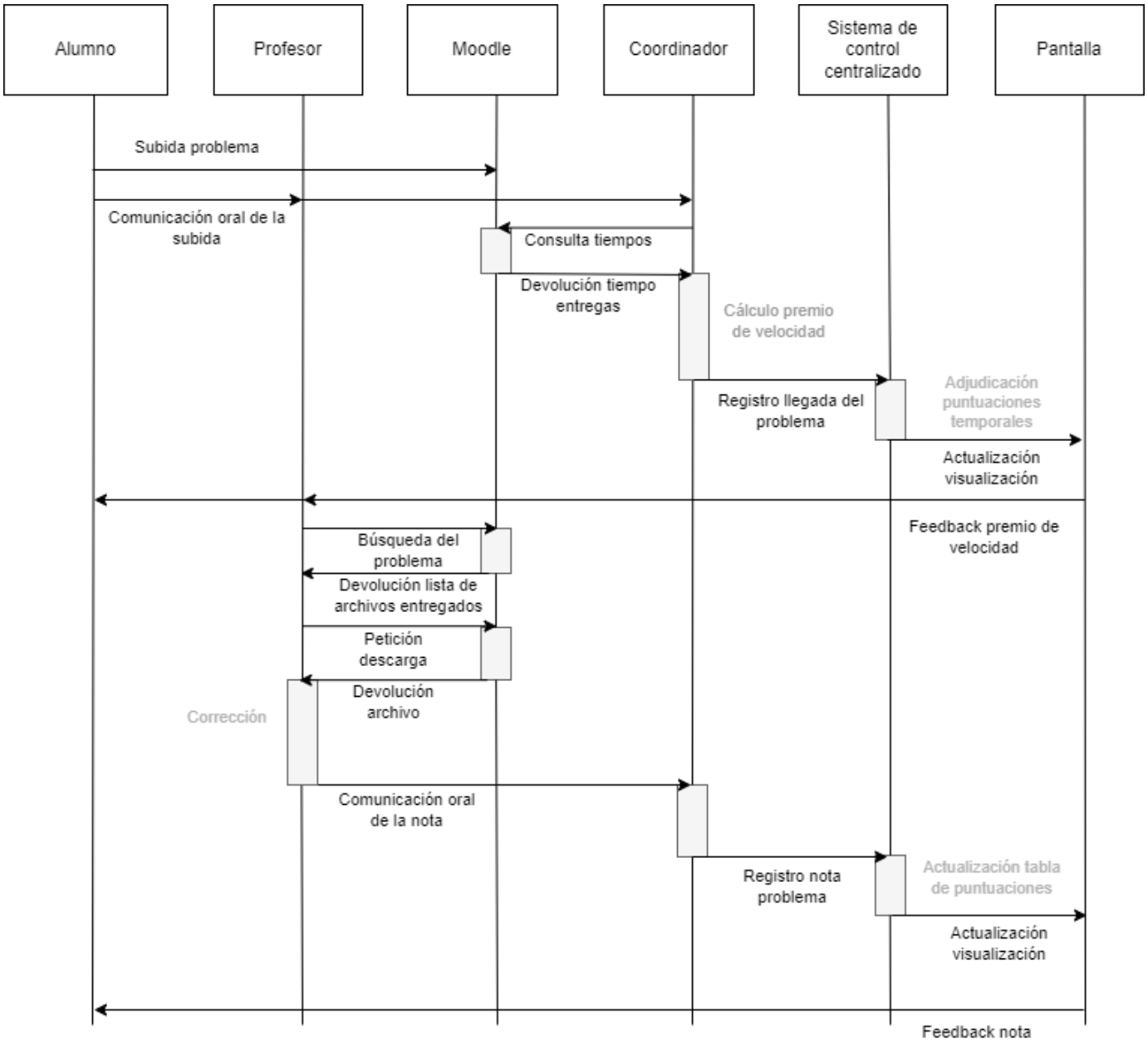


Ilustración 10: Diagrama interacción de la corrección de un problema en el sistema antiguo de gestión

En la Ilustración 10, podemos observar el flujo de interacción de todos los agentes participantes en el concurso cuando se produce la entrega de un problema por parte de un alumno. Analizando el diagrama de interacción, podemos observar como la comunicación entre los agentes humanos es completamente oral.

2.1.3 MOODLE

Ilustración 11: Logo Moodle

Moodle es una plataforma de enseñanza que proporciona a estudiantes, profesores, y coordinadores un sistema integrado único, robusto y seguro para crear espacios de enseñanza y aprendizaje totalmente personalizados.[1] La plataforma está estructurada en cursos, en los cuales los profesores y administradores pueden crear y desplegar recursos para el aprendizaje de los alumnos. Moodle es un sistema que admite la codocencia, pues permite que varios profesores pertenezcan al mismo curso y compartan tareas de enseñanza, planificación y evaluación.

Moodle es la plataforma empleada por la Universidad Pontificia de Comillas para el desarrollo de su actividad docente y, por lo tanto, fue la herramienta utilizada para la creación de los problemas a entregar por los alumnos, la gestión de los equipos que formaban parte del concurso y el intercambio de información alumno-profesor.

Ilustración 12: Uso de Moodle en el Sistema de gestión antiguo

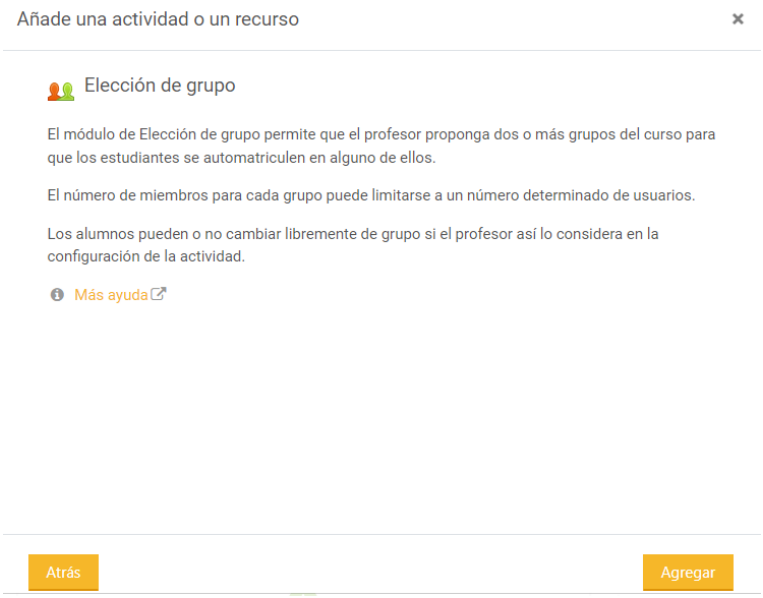


Ilustración 13: Módulo gestión de grupos Moodle

La creación de los equipos y gestión de los mismos se realizó a través de Moodle, con el módulo de la plataforma que permite la gestión de grupos que puede observarse en la Ilustración 13 13. Una vez divididos los alumnos en equipos, la actividad de un integrante del equipo está asociada a la del grupo completo. Es decir, si un alumno entrega un archivo, este archivo aparece entregado para todos los integrantes del grupo. Si otro integrante decide borrar un archivo o modificar la entrega, estos cambios también sobrescribirán la actividad de todo el equipo. Así, un grupo de Moodle y, por tanto, un equipo del concurso, funcionará como una única unidad de cara a la corrección de problemas.

En la Ilustración 14 podemos observar un ejemplo de esto. El Usuario 3 y Usuario 4 pertenecen al grupo MTC04, estando su actividad vinculada. Al realizar el Usuario 3 una entrega, esta misma aparece también al Usuario 4. Sus calificaciones también están vinculadas. No obstante, los Usuarios 1 y 2, que pertenecen a grupos diferentes entre ellos y distintos al MTC04 tienen una actividad en esta tarea independiente de los otros dos usuarios.

Ilustración 14: Funcionamiento de las entregas grupales

Tras la asignación de los alumnos a los grupos, dentro de los cursos de las asignaturas que forman parte del MTC, se crearon secciones propias del concurso, en cada una de la cual se crearon los problemas pertenecientes a cada prueba. En estas tareas que pueden observarse en la Ilustración 15, los alumnos subían los diferentes archivos de los problemas que realizaban para su corrección.

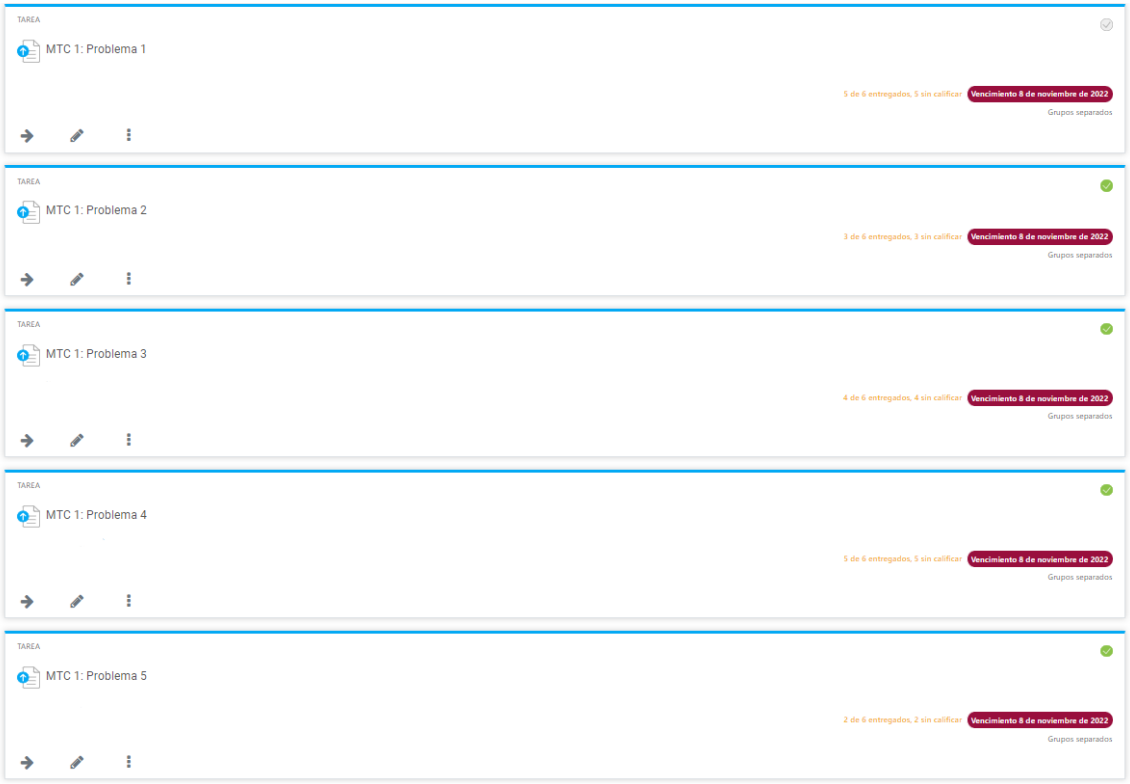


Ilustración 15: Tareas de Moodle para el MTC 1

Como ya he mencionado anteriormente, Moodle es una plataforma que favorece la codocencia, permitiendo la participación de varios profesores en un mismo grupo. Así, durante la realización de las pruebas del MTC, los diferentes profesores del concurso podían acceder a las entregas de los alumnos y corregirlas.

La principal limitación de Moodle es la lentitud de su interfaz de usuario. Su base de datos y backend es muy potente, sin embargo, la interfaz de usuario de la plataforma no cumple con las necesidades del concurso, en el que comunicación profesor-profesor debe ser casi inmediata para que la corrección sea eficiente. Desde que un profesor empieza a corregir una entrega hasta que este califica la misma, Moodle no alerta al resto de profesores del hecho de que uno de sus compañeros está encargado de esa entrega en particular, lo que en casos normales puede no ser relevante. No obstante, para la realización de un concurso en el que 20 equipos deben resolver 10 problemas en una hora, cada segundo cuenta, y es necesario

que exista una comunicación fluida profesor-profesor para que no se den casos de que varios profesores corrijan la misma entrega o se pisen el trabajo de evaluación unos a otros.

Como métrica de usabilidad de la interfaz de Moodle hemos usado el "click count" de la acción de corrección. Suponiendo que el profesor A se encuentra corrigiendo un problema cualquiera y el grupo X le informa de que ha realizado la entrega del problema Y, son necesarios 7 clicks para pasar a ese problema y descargarlo, incluyendo la necesidad de filtrar por las tareas entregas y ordenando por "más recientes" para ver los más recientes arriba y así ver los tiempos para los premios de velocidad. Como medida de éxito del nuevo sistema compararemos el "click count" de corrección de los dos sistemas. Ver Tabla 4: Comparativa Métricas de Usabilidad.

Además, cabe remarcar la problemática que supone para los profesores tener que estar pendientes de dos sistemas no interconectados y estar pasando de uno a otro en los dispositivos móviles para tomar decisiones de qué problemas corregir después.

A pesar de sus limitaciones, la utilización de Moodle como plataforma de entrega de los problemas por parte de los alumnos es un requisito fundamental del diseño del nuevo sistema, pues, al tratarse el MTC de una actividad que se utiliza para evaluar las capacidades de los alumnos y que influye en su nota final de las asignaturas, debe tener el carácter oficial que se sigue en la Universidad, donde toda entrega o subida de exámenes, actividades, proyectos, memorias de laboratorios… se realiza en Moodle, quedando un registro de todos los archivos que permanece aún después de que los alumnos finalicen la asignatura y su actividad docente en la Universidad.

Por otra parte, una de las características de Moodle es la trazabilidad plena de las acciones realizadas y los usuarios que realizan cada acción, además del acceso restringido y securizado a los cursos y pruebas de evaluación.

2.1.4 EXCEL

Excel es la herramienta por excelencia utilizada para procesamiento de números y datos. Excel es un programa desarrollado por Microsoft, que facilita la manipulación de datos numéricos y de texto, el análisis información, la generación reportes, etc.

Esta herramienta se utilizó para el almacenamiento de las notas de las correcciones de cada problema y para la visualización de los datos, que se realizaba en las pantallas inmersivas del aula Comillas Conecta Lab.

El registro de las calificaciones se realizaba de forma manual por parte del coordinador del concurso de tal forma que, una vez un profesor corregía una entrega comunicaba verbalmente al coordinador la siguiente información: Grupo + Problema + Nota.

Esto supone varias limitaciones. En primer lugar, la necesidad de tener una persona dedicada casi exclusivamente al registro de las calificaciones. La realización de esta tarea de forma manual puede llevar además a errores humanos, por equivocación a la hora de introducir una calificación o por un error entre coordinador y profesor a la hora de comunicar una nota. En segundo lugar, la carga de datos durante la realización de las pruebas del concurso es muy alta, lo que dificulta una correcta gestión de la base de datos. Por último, la necesidad de los profesores de localizarse cerca del coordinador para poder comunicar las notas de forma efectiva limita la movilidad de los mismos, que no pueden moverse por la sala del concurso para resolver dudas de alumnos o aclarar dudas propias sobre los documentos entregados por los alumnos.

La visualización de los marcadores también se realizaba en Excel. Con diversas fórmulas de agregación y relaciones de datos, en una Hoja de Cálculo, se mostraban los marcadores que tenían la forma que se ve en la siguiente figura.

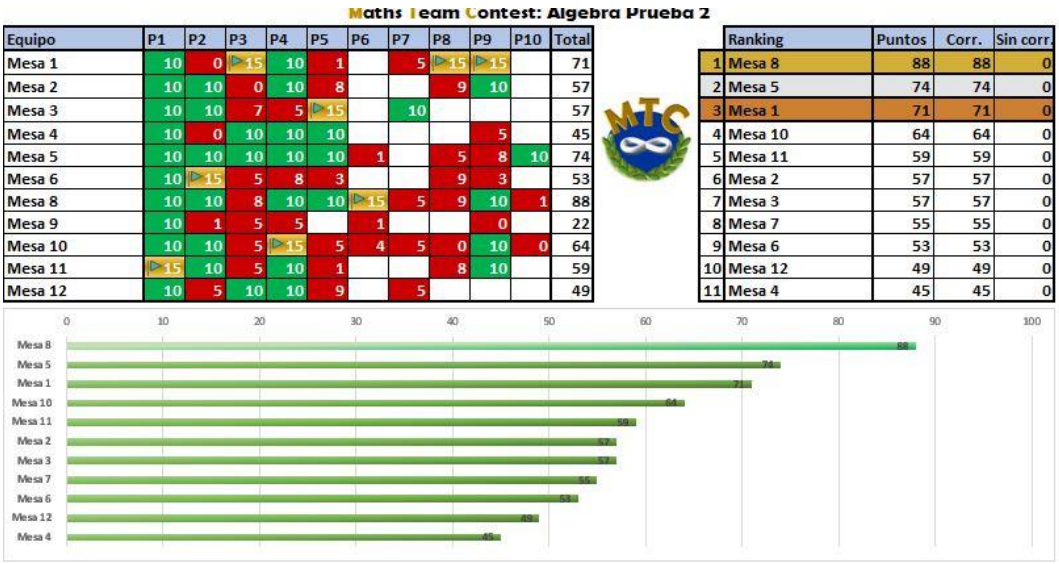


Ilustración 16: Marcador MTC durante el curso 21-22

Podemos ver tres visualizaciones diferentes en el marcador de cada prueba:

- Una tabla general con los puntos de cada equipo en cada problema.
- Un ranking con el dato agregado de cada equipo en la prueba, ordenado de mayor a menor puntuación.
- Un gráfico que muestra la evolución de cada equipo con respecto al equipo con mejor puntuación en ese momento.
- Adicionalmente, se colocaba un temporizador en la parte inferior derecha de la pantalla para controlar el tiempo transcurrido de la prueba.

Esta hoja era proyectada en las distintas pantallas del Comillas Conecta Lab durante la realización de las pruebas para que los alumnos pudiesen ver su evolución, corregir los problemas erróneos y volver a entregarlos o comparar su progreso con el del resto de equipos.

El principal problema de la hoja de cálculo de Excel como herramienta de visualización es la peculiaridad del Comillas Conecta Lab, que cuenta con 5 pantallas independientes, por lo que era necesario compartir la hoja de ranking a cada una de ellas. Esto se realizaba compartiendo pantalla desde un portátil conectado por cable a una de las pantallas y

retransmitiendo esta pantalla a través de Collaborate de unas a otras. Previamente, era necesario crear una sala de Collaborate y acceder a ella en todas las pantallas, entrando en la principal con permisos de presentador, para poder compartir las pantallas de ranking. Así en el ordenador conectado a la pantalla se tenían los sistemas de control, mientras en la pantalla extendida, los de "visualización", que eran compartidos a través de Collaborate.

2.1.5 POST PROCESADO DEL DATO

Para el post- procesado de los datos se utilizaron herramientas de VBA. VBA (Visual Basic) es un lenguaje de programación utilizado para automatizar tareas en aplicaciones de Microsoft tales como Word, Power Point y Excel. Este lenguaje se utiliza para la creación de scripts y macros que permiten realizar tareas recurrentes de manera eficiente.

Al utilizarse Excel como base de datos y punto central de la arquitectura, y ser Microsoft 365 la suite de servicios ofimáticos utilizada en la Universidad, se utilizaron scripts y macros que conectaban la base de datos con el resto de aplicaciones de Microsoft para el post-procesado de los datos del MTC.

Estas herramientas colgaban directamente del sistema centralizado de tal forma que los datos de la base de datos de Excel eran volcados para su análisis y generación de material auxiliar para el concurso, tal como diplomas o estadísticas de la evolución de los equipos a lo largo de la evolución de los equipos a lo largo de la duración del concurso.

Los sistemas de post-procesado de datos que se incluían durante el primer año del concurso son los siguientes:

- Estadísticas
- Ranking
- Evolución de grupos
- Generación de diplomas*
- Cálculo final de notas por alumnos*

* Para el cálculo de notas por alumno y la generación de diplomas individuales se utilizaba además el iPDI de la Universidad para recuperar nombres, apellidos y códigos

El sistema de post-procesado de datos que se utilizó durante el primer año de vida del concurso es óptimo pues al conectarse mediante enlace de datos a los datos exportados en Excel del IPDI de la Universidad y de Moodle para obtener los datos de los alumnos, evita tener que almacenar estos datos en el sistema central de gestión, añadiendo una capa extra de seguridad y evitando el almacenamiento de datos sensibles. Por lo tanto, uno de los requisitos de la nueva plataforma será la integración de estos subsistemas en la arquitectura final.

No obstante, aunque la utilización de VBA como herramienta central de gestión de datos post-prueba es correcta, los datos recogidos no permiten un análisis profundo sobre el rendimiento del MTC en técnicas de innovación docente, que es uno de los objetivos principales del Proyecto 2122-22 y futuro Proyecto 2223-14.

La recogida de datos de forma manual supone una pérdida de información valiosa para este estudio en Innovación Docente, como el número de veces que se ha re entregado un problema por un equipo, los tiempos en los que se ha entregado cada problema para estudiar las franjas temporales de mayor productividad…

ANÁLISIS Y DEFINICIÓN FORMAL DE REQUISITOS

De estas limitaciones que presenta la arquitectura utilizada durante el curso 2021/22 para el Maths Team Contest surge la necesidad de un nuevo sistema de gestión que de soporte a las actividades del MTC. Podemos dividir los requisitos del nuevo sistema en funcionales y no funcionales.

El análisis de requisitos se ha realizado siguiendo el ciclo iterativo de vida de la plataforma, como se explica en el Capítulo 3. Tras un análisis inicial de requisitos, se han realizado revisiones de los mismos a lo largo del ciclo de vida incremental del software.

A continuación, se muestra la definición formal final de los requisitos funcionales y no funcionales del sistema.

2.1.6 REQUISITOS FUNCIONALES

Perfiles de usuario

1. Deberán existir perfiles de usuario para acceder a la plataforma con un nombre de usuario y una contraseña.
2. Existirán cuatro tipos de usuarios principales: pantalla, investigador, profesor y administrador, con diferentes niveles de acceso y permisos. Debe existir la posibilidad de añadir tipos de usuario nuevos en el futuro.

Actividades asociadas a las pantallas

3. Las pantallas podrán acceder a las vistas de visualización de los rankings, tanto en tiempo real como ranking de prueba estático y ranking global de asignatura.

Actividades asociadas a los investigadores

4. Los investigadores podrán acceder a la interfaz de exportación de datos para su posterior análisis.

Actividades asociadas a los profesores

5. Los profesores serán los encargados de corregir los problemas de las pruebas.
6. Los profesores podrán acceder a la herramienta de corrección desde la cuál podrán descargar y corregir las entregas.
7. Los profesores también podrán acceder a las vistas de visualización para poder ver la evolución de la prueba en todo momento desde sus dispositivos móviles.
8. Los profesores podrán ver la actividad que realizan sus compañeros sin necesidad de comunicación verbal entre ellos.

Actividades asociadas a los administradores

9. Los administradores podrán acceder a las vistas de visualización de los rankings y a la herramienta de corrección.
10. Además, podrán acceder a una interfaz de la base de datos, desde la que podrán realizar acciones de creación, modificación, eliminación si es necesario.
11. Los administradores serán los encargados de crear las pruebas en la plataforma y asociarlas con la asignatura correspondiente de Moodle a través de un identificador único.
12. Los administradores serán los encargados de la creación de los grupos que participan en el concurso a través de la interfaz de administrador.

Corrección y descarga

13. La herramienta de corrección debe ordenar los problemas por orden temporal en función de cuando han sido entregados en Moodle.
14. La herramienta de corrección debe tener una interfaz que facilite y agilice la corrección, que permita pasar de la descarga de un problema a otro en menos de 7 “clicks”.
15. Debe existir una herramienta que permita filtrar los problemas entregados por número de problema para facilitar la división de la corrección por parte de los profesores.
16. Se debe indicar qué tipo de archivo o archivos ha entregado un grupo pues hay algunos tipos de archivos a los que los profesores no pueden acceder a través de las tablets y solo pueden abrirse desde un ordenador.
17. Debe existir una forma que indique a los profesores qué problemas han sido corregidos, cuáles han sido descargados por otro compañero y se encuentran en corrección y cuáles cuentan con premio de velocidad sin que tengan que ir al ranking para ello.
18. El sistema debe gestionar de forma autónoma los premios de velocidad, con el fin de evitar el fallo humano.

19. Cuando se realice la subida de un problema a Moodle por parte de un grupo, éste debe aparecer en la herramienta de corrección sin necesidad de un refresco manual de la página por parte de los profesores.

Entregas

20. Una entrega debe permitir varios archivos.
21. Una entrega estará asociada a un problema y a un grupo.
22. Una entrega debe admitir reentregas y debe poder ser eliminada a través de Moodle.
23. Cuando se elimine una entrega en Moodle por parte de un grupo, esta debe quedar invalidada en la plataforma, aunque haya sido corregida.
24. Cuando se realice una reentrega del problema X por parte del grupo Y, se debe notificar a los profesores, informando de que se trata de una reentrega y de cuántas entregas anteriores se han hecho de dicho problema. Las entregas anteriores deben quedar invalidadas.

Ranking

25. Las pantallas de visualización en tiempo real deben contar con las tres vistas que se utilizaban en el año 21/22 y que dan la siguiente información.
26. Una tabla con la puntuación general de los grupos, en la que aparezca la puntuación total de los problemas corregidos, la puntuación en revisión de los problemas entregados, pero aún sin corregir, y la suma de ambos, que supone la puntuación total en la prueba.
27. Una tabla en la que aparezca todos los problemas y los grupos y en la que sea sencillo ver qué problemas han sido entregados por cada grupo, cuáles han descargados y se encuentran en corrección, cuáles han sido corregidos y tienen la puntuación máxima, cuáles cuentan con premio de velocidad y cuáles no han sido entregados.
28. Una gráfica en la que se muestre una comparativa de la actuación de los grupos en la prueba, en relación al grupo con mayor puntuación total en cada momento.
29. La información de los rankings debe actualizarse en tiempo real, o en su defecto, cada pocos segundos.

30. Debe existir un ranking final que permita ver la puntuación de los grupos en la prueba (como una "foto finish"). Este debe ser similar y plasmar toda la información que contiene la tabla del R27.
31. Debe existir un ranking global que permita ver la actuación de los grupos en todas las pruebas realizadas hasta la fecha de cada asignatura que muestre la puntuación de cada prueba y la suma de todas ellas.

Exportación de datos

32. Es necesaria una interfaz de exportación de datos que permita la descarga de todos los datos recogidos durante el concurso para el post-procesado del dato.
33. Deben poder descargarse los datos crudos de todas las entregas en formato CSV, para su posterior análisis por los investigadores.
34. Deben poder descargarse los rankings tal y como aparecen en la visualización de las pantallas, tanto los finales de cada prueba como los globales, para la generación de diplomas.
35. Debe poder descargarse un listado CSV todos los grupos que hayan recibido algún premio de velocidad en cualquiera de los problemas de una asignatura.

Moodle

36. Utilización de Moodle como plataforma de entrega de problemas gestión de grupos. Como se ha mencionado anteriormente, la principal limitación de Moodle en el caso particular del MTC es su interfaz de usuario que no facilita una comunicación fluida profesor-profesor. Siendo así, es requisito fundamental que se emplee Moodle como sistema de entrega de los problemas realizados por los alumnos, al tener el concurso carácter oficial y tener que cumplir con lo impuesto por la Universidad. Por otro lado, la gestión de grupos en Moodle es de gran ayuda para la realización del concurso.
37. Gestión del flujo de datos. La gestión de datos debe permitir la sincronización de datos con Moodle, donde se recogen las entregas de los alumnos. Además, existe la necesidad de automatizar y distribuir el flujo de datos, que hasta el momento se realizaba de forma manual.

1.1.1 REQUISITOS NO FUNCIONALES

38. Flexibilidad del dato. Siguiendo el ciclo de vida iterativo de vida del diseño de la solución, se debe permitir la flexibilidad y ampliabilidad de los datos que se recojan del concurso.
39. Visualización. Es necesaria la visualización del progreso del concurso adaptada al espacio físico del Comillas Conecta Lab. Este cuenta con 5 pantallas de diferentes aisladas entre sí y de diferentes tamaños. Los rankings deberán adaptarse a ellos.
40. Integración de los diferentes subsistemas. Es necesaria una arquitectura que interconecte las diferentes infraestructuras preexistentes en un único software de gestión que permita la automatización del dato y de las tareas que realizaba de forma manual la figura del coordinador.
41. Ciberseguridad. Al tratarse de datos de alumnos, y ser por lo tanto datos sensibles, es necesaria una capa de seguridad que limite los accesos a los datos de alumnos y aísle la actividad del MTC del resto de las actividades que se realicen en las asignaturas de Álgebra y Análisis y Cálculo.
42. Sistema distribuido y escalable. Necesidad de un sistema que permita grandes cantidades de datos en periodos cortos de tiempos y la gestión distribuida de los mismos. Debido al creciente número de alumnos participantes en el MTC, que supone un incremento en los docentes necesarios para llevar a cabo el concurso y una gran cantidad de datos que deben ser almacenados y procesados, es necesaria una plataforma escalable y distribuida, que permita la anexión de nuevos grupos y profesores sin alterar el funcionamiento normal del sistema. Se espera que en el futuro haya un número aún mayor de alumnos y además la intención es diseñar una arquitectura que pueda adaptarse a otros tipos de actividades semejantes.
43. Retrocompatibilidad con sistemas preexistentes. Integración con los subsistemas de post-procesado del dato, de tal manera que se mantengan los actuales y se conecten a la interfaz de datos de la plataforma o se integren en la arquitectura del sistema.
44. Deberá cumplir los siguientes requisitos de software/hardware. El software utilizado debe ser compatible con el espacio impuesto para el concurso, el Comillas Conecta Lab, con sus requisitos de red y de hardware (pantallas). Además, debe ser soportado

en dispositivos móviles (tablets) que permitan la movilidad de los profesores alrededor de la sala mientras realizan las actividades de corrección. En los requisitos de red, es necesario particularizar el firewall en el wifi eduroam de la universidad. Además, el software propuesto debe ser compatible con las restricciones del servidor utilizado. Esto es, puertos y protocolos de comunicación soportados por el servidor de la Universidad.

La actividad de los profesores se hará a través de dispositivos móviles (tablets) por lo que es necesario que la plataforma sea compatible con estos.

A partir de este momento en el documento, cada vez que se referencie un requisito aquí establecido se hará de la siguiente forma RFx, para los requisitos funcionales, siendo x el número de requisito, y RNFx, para los no funcionales.

3. PLAN DE TRABAJO

El planteamiento del trabajo estuvo marcado por la realización de cada una de las pruebas que forman parte del MTC en el curso 2022/23.

En total se realizaron ocho pruebas en cuatro fechas distintas, alrededor de las cuales se estructuró el plan de trabajo. Las fechas de realización de las pruebas pueden consultarse en la Tabla 1:

Prueba	Fecha
MTC 1	04/11/2022
MTC 2	02/02/2023
MTC 3	17/03/2023
MTC 4	14/04/2023
Entrega de premios	26/04/2023

Tabla 1: Fechas pruebas MTC

A partir de estas pruebas se estructuró el plan de trabajo en cinco grandes fases, que suponen un ciclo de vida iterativo justificado por la magnitud del proyecto y los requisitos temporales que suponen la realización de cada prueba.

El ciclo de vida del desarrollo de software (CVDS) describe las tareas necesarias para crear una aplicación de software. El proceso de desarrollo pasa por varias etapas a medida que los desde el planteamiento inicial al despliegue final.

Este proyecto se ha llevado a cabo con un ciclo de vida iterativo e incremental, en el que en cada iteración se van liberando prototipos y con cada nueva versión, aumenta la funcionalidad y mejora en calidad respecto a la anterior. [3]

Así, en cada ciclo o iteración, se revisa y mejora el producto. Para este proyecto, hemos hecho coincidir cada iteración con las fechas de las realizaciones de las pruebas del MTC. De esta manera, para cada significativa del concurso, teníamos un prototipo funcional en producción y mejorado respecto al anterior.

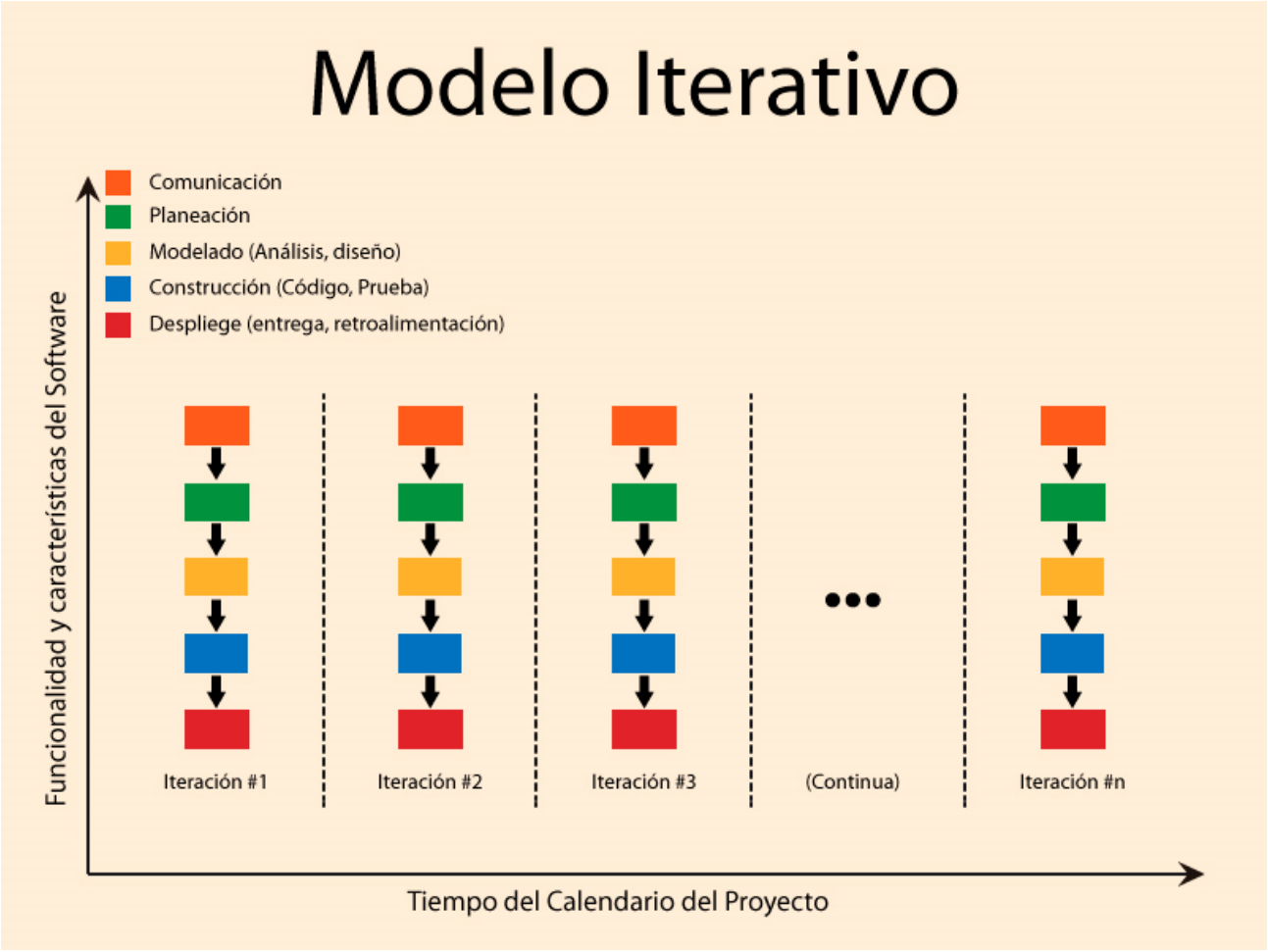


Ilustración 17: Ciclo Iterativo del Desarrollo de Software [5]

Como se puede observar en la Ilustración 17, en cada una de las iteraciones se lleva a cabo una comunicación, que planteamos como reevaluación de requisitos. Después, se lleva a

cabo un análisis de los mismos y un diseño del nuevo software o características adicionales. Tras la construcción del software y las pruebas correspondientes sobre el mismo, se despliega y se evalúa su funcionamiento, en nuestro caso en cada una de las pruebas del concurso.

Así podemos dividir el plan de trabajo del proyecto en 5 iteraciones según el ciclo de vida iterativo del software.

1ª ITERACIÓN. DEFINICIÓN DE REQUISITOS Y MVP

En esta fase se llevó a cabo un estudio del estado de la cuestión y una formalización de los requisitos funcionales del sistema de forma conjunta con el director del proyecto y coordinador del MTC.

En paralelo, estudié diferentes lenguajes de programación y frameworks para utilizar en el proyecto, considerando las ventajas y desventajas de cada uno. Una vez analizada la información, seleccioné Django como framework que mejor se adapta a los requisitos del proyecto.

Para la realización de la primera prueba del MTC era necesario tener una plataforma funcional ya en producción. Es por esto que esta Fase 1 se plantea como una fase inicial del ciclo de vida de la plataforma.

Tras analizar todos los requisitos de la plataforma y debido a las restricciones temporales, se propuso la creación de un MVP que contase con las características básicas y necesarias para dar soporte al concurso.

Un MVP o Producto Mínimo Viable es una versión elemental de un producto que permite validar la idea y aplicabilidad de nuestra aplicación en este caso.

Para la creación del MVP, se llevaron a cabo todas las fases normales de un proyecto de software, aplicados al producto mínimo viable:

1. Evaluación de requisitos mínimos
2. Diseño de la arquitectura
3. Disposición de entorno de trabajo y producción
4. Implementación
5. Pruebas de integración, de validación y de sistema y Puesta en producción

Tras establecer la arquitectura principal, se creó un entorno de trabajo para desarrollo y pruebas basado en una "beta cerrada" y un curso de Moodle de pruebas.

Se creó, en colaboración con los servicios de STIC y los profesores del MTC, un curso de Moodle no asociado a ninguna asignatura real de los profesores, sobre el que se realizaron pruebas de conexión y de validación.

Además, antes de finalizar la 1ª iteración, marcada por la celebración de la primera del concurso, se realizaron pruebas de integración sobre el servidor como una "beta cerrada". Sin alumnos ni profesores, y accediendo al Moodle de pruebas, se realizaron pruebas de integración para asegurar el correcto funcionamiento de la plataforma.

En la evaluación de requisitos del Producto Mínimo Viable tuvimos que extraer las funcionalidades esenciales que debería tener nuestro sistema, que fueron las siguientes.

[1] Conexión con Moodle para la recepción de los problemas entregados por los alumnos.
[2] Interfaz para la descarga y corrección de los problemas por parte de los profesores.
[3] Interfaz de visualización de ranking y notas en tiempo real para ver la evolución de los grupos durante la prueba.
[4] Comunicación síncrona profesor-profesor para evitar problemas en la corrección.

Con estas características básicas cree un MVP que me permitió evaluar la viabilidad de la aplicación y verificar su potencial.

A partir de este producto base, en iteraciones posteriores se fueron introduciendo gradualmente nuevas funcionalidades o características que facilitaban el uso del sistema.

2ª ITERACIÓN. REEVALUACIÓN DE REQUISITOS

Una vez terminada la primera prueba del MTC y teniendo un prototipo mínimo funcional, se inició la 2º iteración evaluando el funcionamiento del MVP.

Se detectaron problemas con el crawler que recogía las entregas de los alumnos de Moodle, lo que se solventó añadiendo un marcapasos que comprobaba el status del crawler cada cierto tiempo. Esto se detallará en profundidad en la sección de diseño de la solución.

En esta fase además se amplía y mejora la automatización del Crawler de Moodle, agregando una fase de inicialización para automatizar el enlace del sistema a cursos y tareas y cargar automáticamente los problemas de una prueba proveniente de Moodle.

En esta iteración, además se pulieron los diferentes niveles de acceso de usuario para garantizar la seguridad de la aplicación y se introdujeron perfiles para todos los docentes participantes en el concurso.

Además, se llevó a cabo una reevaluación de los datos a guardar en la aplicación que pudiesen ser útiles para el análisis de los datos post-prueba.

También se llevaron a cabo pruebas de integración más en profundidad para descartar futuros bugs.

3ª ITERACIÓN. MEJORAS DE ESTABILIDAD Y USABILIDAD

En la 3ª iteración, ya habiendo celebrado dos eventos del concurso, se añadieron las funcionalidades de ranking globales para mostrar la evolución de los grupos durante las dos pruebas combinadas de cada asignatura.

Además, se añadieron mejoras de interfaz que facilitaban el uso de la aplicación por parte de los docentes. y se profundizó en la gestión de excepciones.

Tras la segunda prueba del MTC, se llevó a cabo una mejora de la estabilidad y la usabilidad del sistema.

Me centré en el análisis del tráfico de Moodle que recogía el crawler y en asegurar que el sistema fuese robusto ante la presencia de datos provenientes de Moodle patológicos o comportamientos inesperados por parte de los alumnos o profesores participantes (eliminación/cancelación/resubida/bugs de Moodle).

Se realizó un refinamiento de la gestión de la corrección concurrente y una modularización de todo el acceso a la base de datos, tanto por parte de los profesores como del módulo que conecta Moodle con la plataforma del MTC.

Además, se puso el foco en la automatización de la reasignación de los premios de velocidad.

4ª ITERACIÓN. EXPORTACIÓN DE DATOS

Tras la realización de la tercera prueba del concurso, se volvió a evaluar el funcionamiento de la aplicación con el tutor del proyecto y se añadieron el resto de los sistemas de post-procesado de datos. Para ello, se creó una interfaz de datos retrocompatible que permite acceder a los mismos por parte de los subsistemas preexistentes, que se enumeran a continuación:

1. Asignación de premios finales.
2. Sistemas de generación de diplomas y de presentaciones para las entregas de diplomas,
3. Cálculo de notas y exportación de notas medias finales de alumnos, a cuyos perfiles individuales no tiene acceso la aplicación principal por motivos de seguridad, sistemas de análisis de estadísticas, etc.) y
4. Agregación de nuevos sistemas avanzados de análisis y visualización de datos como un Bar Chart Race de evolución de las puntuaciones del concurso segundo a segundo.

El resto de los sistemas de post-procesado se integraron en la aplicación principal.

5ª ITERACIÓN. FASE FINAL

Tras la realización de la última prueba del MTC, se realizó toda la exportación de los datos recogidos durante el concurso para su análisis.

En esta 5ª iteración, se llevó a cabo una evaluación final de los requisitos de la aplicación para ver si se habían cumplido y se entregó la documentación de la aplicación al director del MTC para el uso de la plataforma en años posteriores. Además, se realizó un estudio de posibles módulos que añadir al sistema para complementar su funcionamiento.

4. ANÁLISIS GENERAL DEL SISTEMA

En este capítulo se realizará un análisis detallado del sistema general y de la estructura general de la arquitectura, que se ha diseñado para cumplir con los requisitos funcionales y no funcionales de alto nivel establecidos en el Análisis y Definición Formal de Requisitos del Planteamiento del problema.

Tras explicar el diseño general de la arquitectura, sobre el que se profundizará en diferentes subsistemas analizando como resolver cada uno de los problemas de la arquitectura, se expondrán las posibles infraestructuras planteadas, y se hará un análisis de cada una, explicando las características que han motivado su rechazo.

DISEÑO GENERAL DE LA ARQUITECTURA

Para el desarrollo de la nueva plataforma de gestión del MTC se ha diseñado e implementado una arquitectura basada en un sistema centralizado de gestión de todo el flujo de datos del concurso en tiempo real.

Se ha utilizado un servidor Django + Apache con una base de datos SQLite3 como sistema central para gestionar la lógica del concurso.

La decisión de Django + SQLite3 se tomó después de realizar una fase exploratoria y analizar diferentes frameworks/arquitecturas y estudiar proyectos con gestión de datos en tiempo real en cada uno. [9][10]

Las principales ventajas de Django incluyen su interfaz de administrador, que facilita la gestión de usuarios y permisos y su escalabilidad, siendo ejemplo de ello Instagram, que se basa en Django y tiene millones de usuarios activos. [9][13]

Además, el framework Django se ha diseñado de tal manera que facilite el desarrollo web, protegiendo automáticamente partes de la arquitectura. Por ejemplo, en el marco Django en

Django proporciona un sistema flexible de almacenamiento de contraseñas y utiliza el algoritmo hash PBKDF2 por defecto.

La aproximación inicial era utilizar el Django crudo. No obstante, los servidores de la Universidad a los que se tenía acceso para desplegar la solución están montados con un Apache para registrar el tráfico de datos y regular los puertos, garantizando la seguridad de la Universidad.

Este sistema centralizado, se conecta a la API de Moodle a través de un bot con credenciales de profesor y permisos limitados que permite al subsistema de crawling interactuar en tiempo real con las asignaturas de Moodle que pertenecen al concurso.

Moodle sigue siendo el programa utilizado por los alumnos como plataforma de trabajo. En las tareas habilitadas en las secciones de la asignatura correspondiente, los alumnos organizados en grupos suben las soluciones a los problemas planteados a Moodle, tal y como se explica en la sección de Moodle en el Planteamiento del problema.

A través del bot que bebe de la API de Moodle, el sistema central de control del Django detecta en tiempo real estas subidas y toma las decisiones necesarias de gestión.

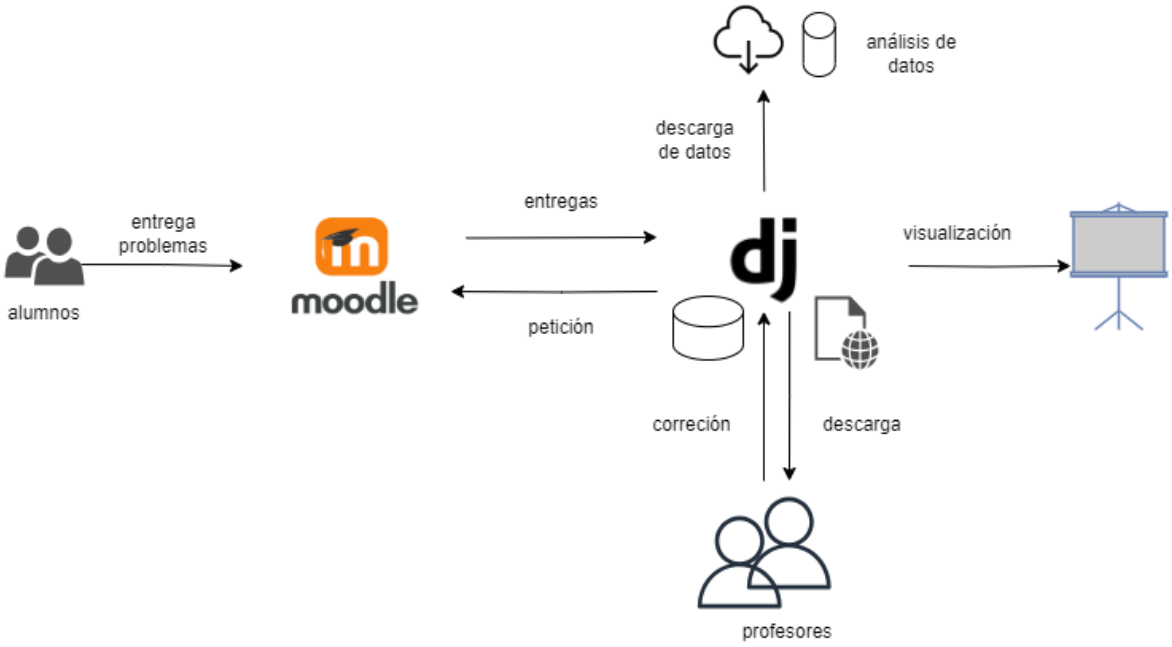


Ilustración 18: Sistema de gestión del MTC propuesto

Estas se detallarán a continuación en el Capítulo 5. e incluyen:

- Determinar si existen problemas nuevos que deben ser corregidos
- Detectar los problemas modificados o eliminados
- Determinar las entregas que deben recibir premios de velocidad
- Determinar las entregas que requieren prioridad de corrección
- Administrar los enlaces de descarga de los ficheros entregados por los alumnos
- Controlar si se ha iniciado la corrección de alguna entrega por parte de un profesor e informar al resto
- Gestionar la corrección y puntuaciones
- Adjudicar puntuaciones temporales
- Actualizar los rankings de acuerdo a la llegada de problemas y a las correcciones de los profesores

El almacenamiento de los archivos asociados a la entrega se delega en Moodle, dejando así la gestión pesada en el backend de Moodle, lo que hace que el sistema sea escalable.

Este submódulo se explicará en profundidad en el Capítulo5. En el momento de la descarga de la entrega por parte del profesor, el Bot accede a la ubicación del archivo en Moodle y, con sus credenciales, lo descarga.

El acceso a la plataforma se hace a través de un cliente web (Mozilla, Chrome…) lo que permite a los profesores acceder a través de dispositivos móviles. Las pantallas inteligentes también se conectan a través del cliente web utilizando credenciales de acceso con permisos limitados a funciones de visualización.

Para el post-procesado de datos también se ha creado una interfaz de datos, a los que los investigadores del Proyecto de Innovación Docente pueden acceder a través del cliente web para descargar todos los datos del concurso.

Esta arquitectura se ha desplegado un servidor de la Universidad, lo que permite a los usuarios conectarse a través de Internet desde cualquier dispositivo. Para el despliegue en el servidor de la universidad se ha utilizado un acceso por VPN, pues el servidor sobre el que se ha desplegado la solución se encuentra en una subred privada cuyos puertos están sellados y que no es accesible desde la red pública. Estos detalles de despliegue se expandirán en el [1]6. Implementación, despliegue y Pruebas desarrolladas

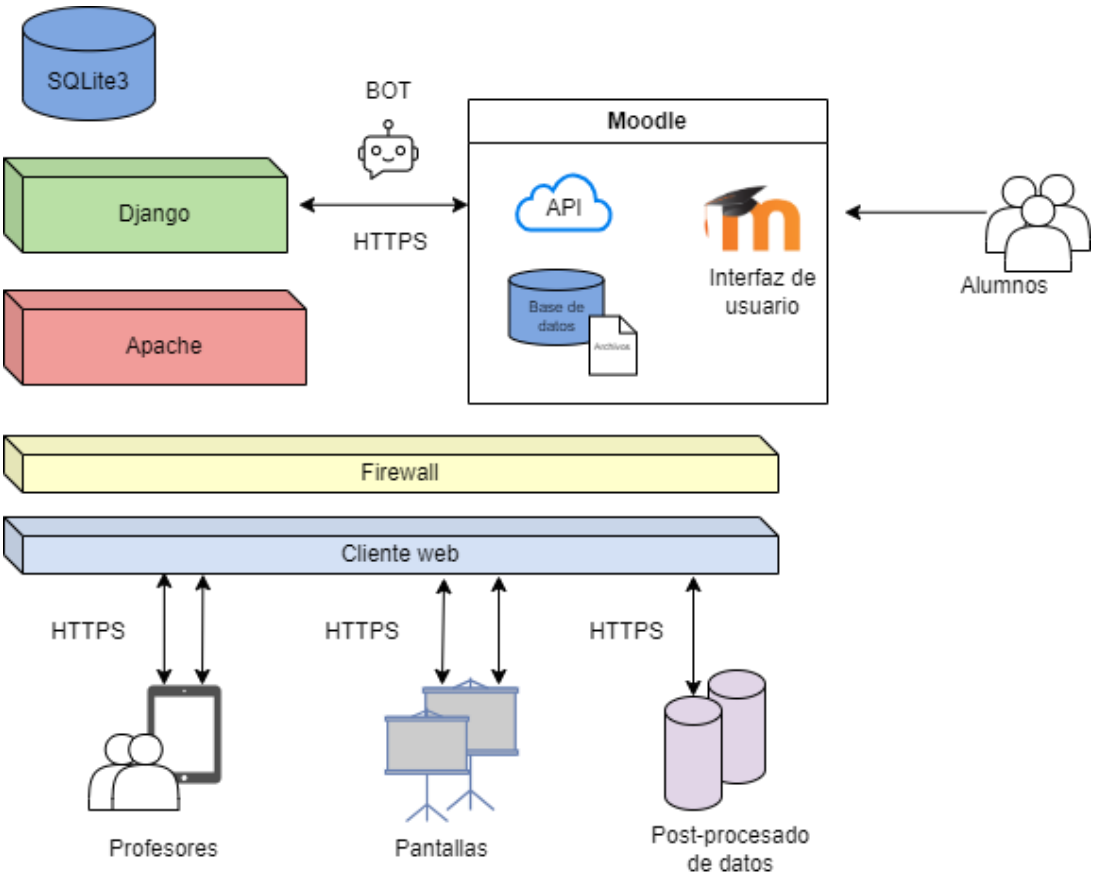


Ilustración 19: Arquitectura de alto nivel

4.1.1 COMUNICACIÓN EN TIEMPO REAL POR WEBSOCKETS

La plataforma se diseñó para la utilización de websockets para la comunicación en tiempo real entre los diferentes clientes de la plataforma.

Los websockets son un protocolo de comunicación que permite la comunicación bidireccional síncrona entre cliente y servidor a través de una conexión persistente. A diferencia del protocolo HTTPS, que es unidireccional, los websockets no requieren que el cliente realice una petición para recibir una actualización del servidor, por lo que se estimó ideal para una plataforma de colaboración en tiempo real como es el MTC.

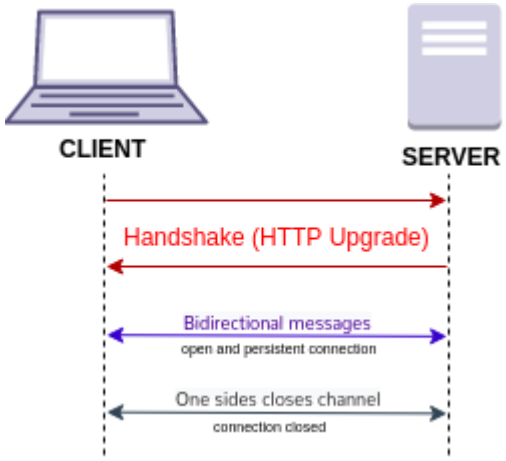


Ilustración 20: Conexión utilizando Websockets [6]

Los websockets, además cuentan con una latencia más baja que el protocolo HTTPS, por lo que las actualizaciones del servidor son más rápidas.

Así, el diseño de la arquitectura inicial contaba con una comunicación híbrida HTTPS + Websockets, lo que permitía una comunicación dinámica al utilizar Websockets para actualizaciones en tiempo real y HTTPS para la comunicación estándar entre cliente y servidor. La utilización de ambos protocolos reducía la necesidad de hacer solicitudes HTTPS frecuentes y recurrentes al servidor, mejorando la eficiencia de la aplicación.

La arquitectura general de alto nivel utilizando esta comunicación híbrida HTTPS + Websocket puede consultarse en la Ilustración 21.

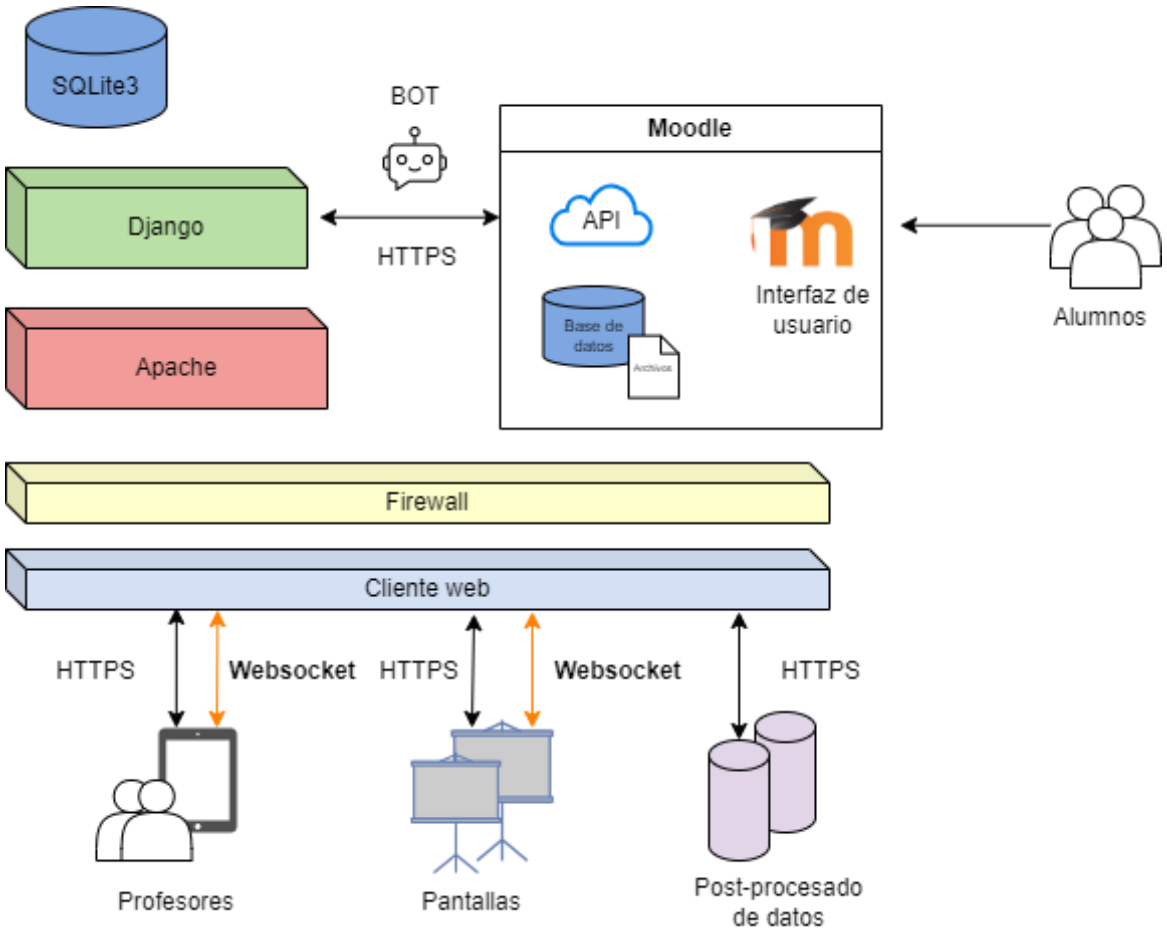


Ilustración 21: Arquitectura de alto nivel con protocolo websocket

La estructura de comunicación híbrida era la siguiente:

- Los clientes (es decir, los diferentes usuarios de la plataforma) se comunicaban inicialmente mediante peticiones HTTPS para acceder a la aplicación y obtener los recursos estáticos de esta.
- Tras cargar los recursos estáticos, el cliente utilizaba HTTPS para obtener una conexión inicial a los Websockets mediante un "handshake" HTTPS.
- El servidor mandaba actualizaciones en tiempo real a los clientes a través de websockets, permitiendo una comunicación dinámica.
- Las diferentes acciones de los profesores (descarga, corrección) se comunicaban por HTTPS al servidor, pues son operaciones de lectura y escritura en la base de datos.

- El servidor procesaba estas acciones y enviaba actualizaciones asíncronas al resto de clientes conectados (resto de profesores y pantallas), lo que garantizaba una actualización inmediata de la información.

En la Ilustración 22 se puede observar cómo se realizaría la comunicación de la corrección de un problema en la comunicación híbrida diseñada. Las flechas negras indican comunicación HTTPS mientras que las naranjas, comunicación por websocket. Se supone que tanto Profesor 2 como pantalla ya han establecido la comunicación inicial y el "handshake" para la comunicación por websockets. El profesor 1 se conecta a la plataforma, realiza una petición HTTPS al servidor, que devuelve todos los recursos estáticos. Tras establecer la comunicación al canal del websocket, realiza una petición de corrección de un problema al servidor, que realiza las acciones lógicas correspondientes (gestión de premios de velocidad…) y comunica a la base de datos la actualización de la nota del problema. Tras recibir la confirmación OK de la base de datos, el servidor lanza un mensaje de broadcast a a todos los clientes conectados a través de websockets que indica que este problema ha sido corregido.

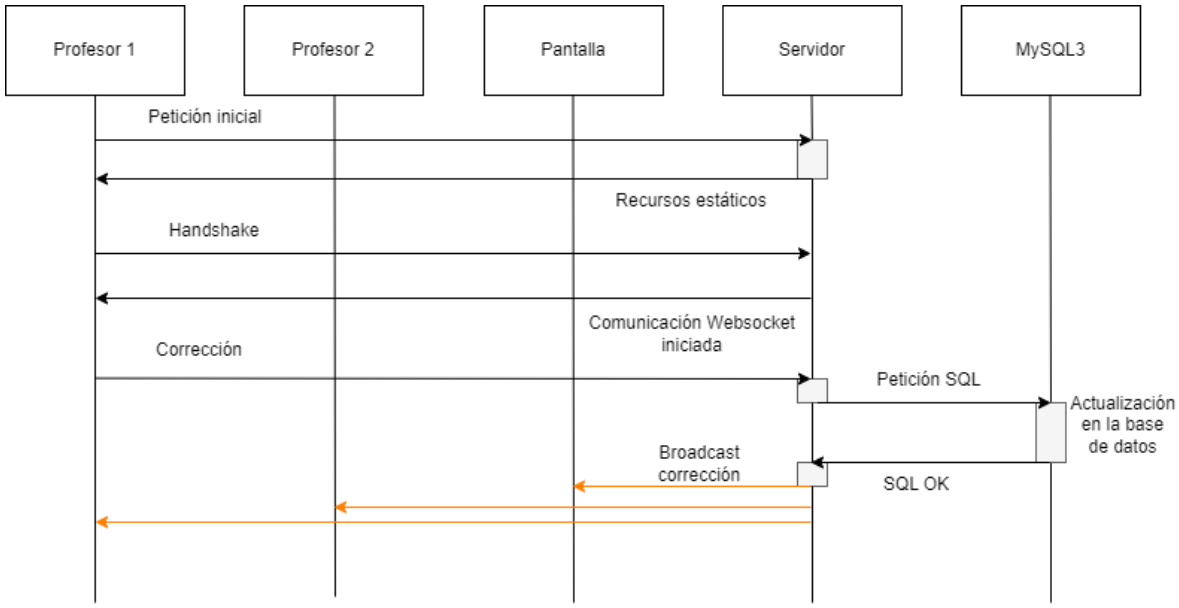


Ilustración 22: diagrama de interacción en comunicación híbrida

En cambio, en la Ilustración 23, se ilustra el caso de comunicación únicamente por HTTPS. En la caja morada se encuentra el flujo de interacciones para la actualización de información. Como se puede observar, el servidor espera a recibir una petición de actualización y cuando la recibe realiza una búsqueda en la base de datos. Tras recibir el mensaje de la base de datos, devuelve la información al profesor 2, en la que se encuentra la corrección del problema por parte del profesor 1.

Esta "caja" de actualización se repetiría para todos los usuarios/clientes de la plataforma que quieran recibir una actualización.

Además de suponer un gran número de peticiones HTTP, este sistema de comunicación también implica un mayor acceso a la base de datos, por lo que la comunicación híbrida es más adecuada para nuestro caso de uso.

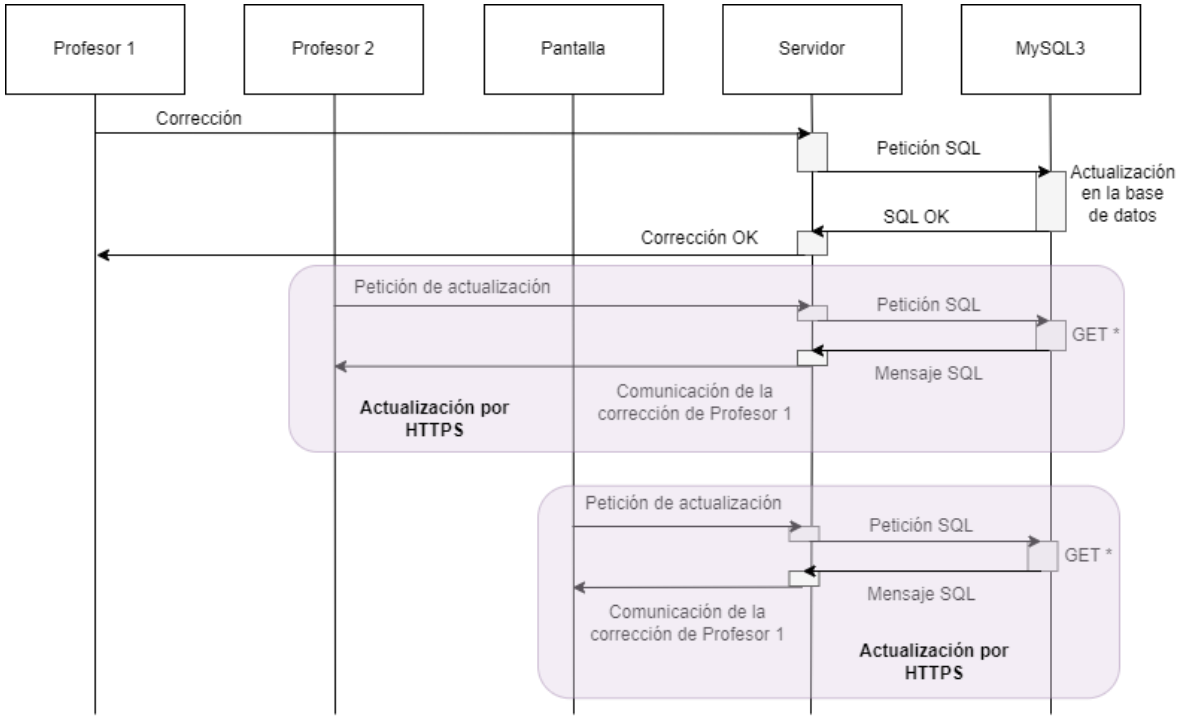


Ilustración 23: diagrama de comunicación con HTTPS

No obstante, el firewall del servidor utilizado no permite la comunicación a través de websockets. El wifi eduroam de la Universidad únicamente permite la comunicación por el puerto 80 y 443, siendo posible realizar un websocket por cualquiera de ellos, utilizando el puerto 80 para conexiones regulares y el 443 para conexiones por Websocket sobre TTL/SSL [9][11]. Sin embargo, estos puertos están filtrados por el Apache que lleva delante el servidor de la Universidad impuesto, imposibilitando el uso de este protocolo de comunicación en producción.

En una fase inicial de análisis se implementó la comunicación con websockets para el MVP. No obstante, en la 2ª iteración de diseño, tras un análisis más detenido de los firewalls del servidor, se descubrió la imposibilidad de usar este protocolo de comunicación. Todas las funcionalidades realizadas en la 1ª iteración mantienen la comunicación por websockets (corrección/descarga/visualización) y en local es posible utilizar este protocolo.

Tras la 2ª iteración, se ha discontinuado la línea de implementación de websockets por las limitaciones de hardware impuestas en el RNF 44.

Para cumplir con las restricciones del servidor en la realización de las pruebas del MTC, se cambió en producción de una comunicación HTTPS + Websocket a una comunicación únicamente por HTTPS, con peticiones de actualización continuas y automáticas por parte de los clientes, para garantizar información actualizada casi en tiempo real.

CONEXIÓN CON MOODLE

Uno de los principales requisitos de la plataforma era el uso de Moodle como plataforma de subida de los problemas (RF 36). Eso conlleva la necesidad de conectar el sistema de gestión de la solución con Moodle.

Moodle sigue siendo la herramienta utilizada por el alumno para la entrega de problemas y además se ha mantenido su uso para la creación de equipos y gestión de los mismos.

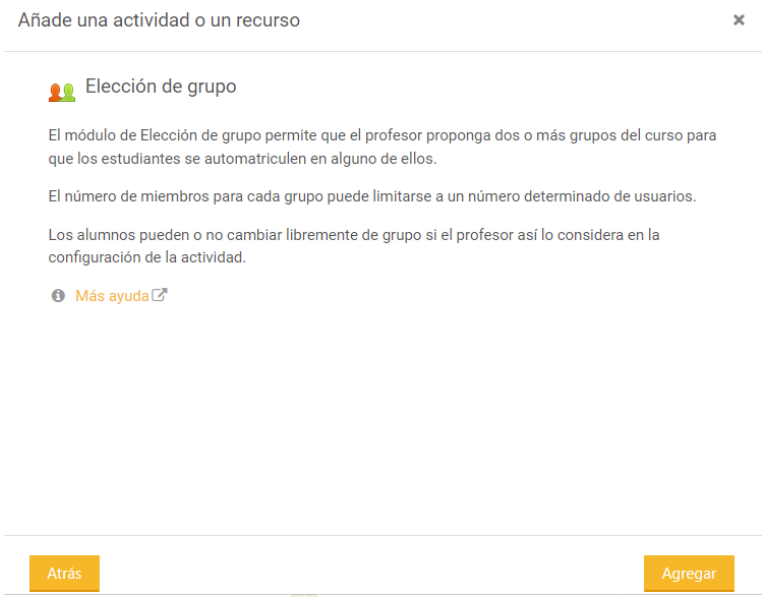


Ilustración 24: gestión de equipos en Moodle

Como se ha explicado en el Estado de la cuestión, una vez divididos los alumnos en equipos, la actividad de un integrante del equipo en Moodle está asociada a la del grupo completo. Un grupo de Moodle y, por tanto, un equipo del concurso funcionará como una única unidad de cara a la corrección de problemas.

Para la nueva gestión de Moodle, se han creado tareas con un formato y nomenclatura específicas, que permiten que el crawler de Moodle detecte estas como parte del concurso. En cada una de las asignaturas del concurso, se ha creado una sección MTC con tareas para cada uno de los problemas de la prueba, siguiendo el siguiente formato:

MTC X: Problema Y, con X haciendo referencia al número de prueba e Y, al problema en particular.

En estas tareas que pueden observarse en la siguiente ilustración, los alumnos suben los diferentes archivos de los problemas que realizaban para su corrección.

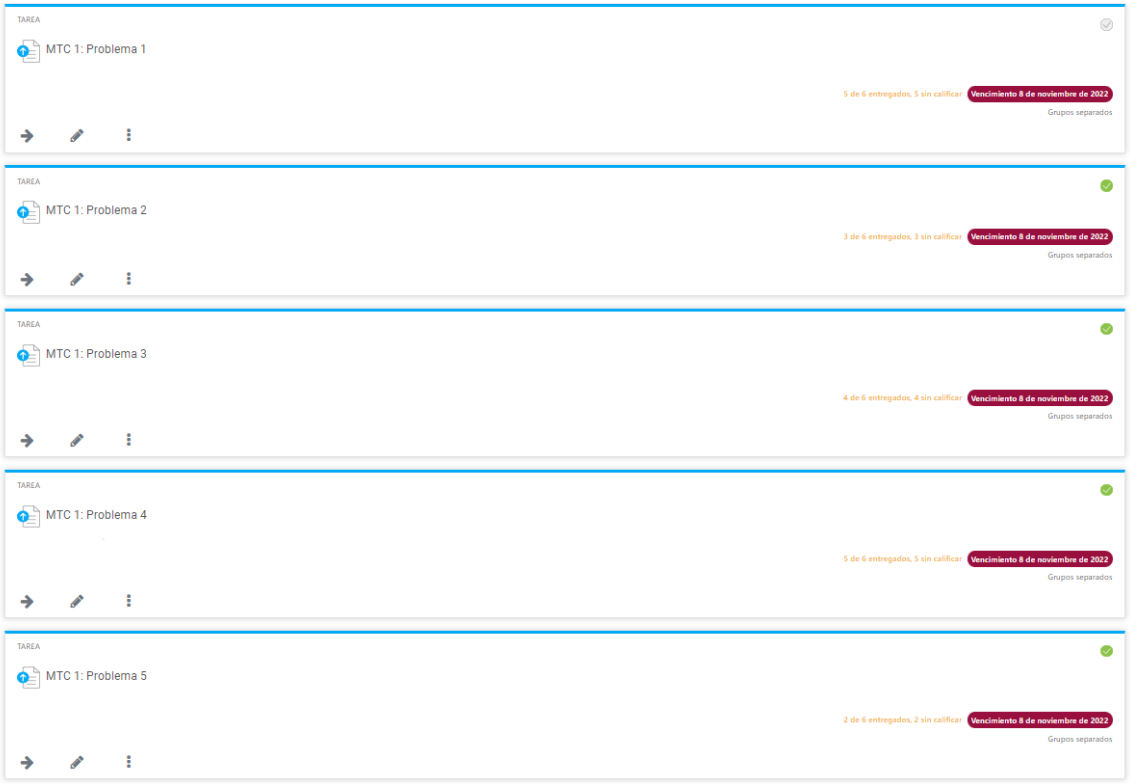


Ilustración 25: Tareas de Moodle para el MTC 1

Otro de los requisitos no funcionales era la ciberseguridad, en todas las capas de la plataforma, incluyendo la comunicación con Moodle. (RF 41).

Teniendo en cuenta estos requisitos, se ha establecido una comunicación con Moodle a través de su API, utilizando un Bot con credenciales de profesor asociado a las asignaturas del concurso.

Para acceder a la API de Moodle se requiere un token de acceso. Así, Moodle restringe sus actividades a usuarios registrados, pero también limita los permisos en función del rango o tipo de usuario. Así, un alumno no podría acceder a tareas de corrección.

En colaboración con los servicios STIC de la Universidad, se creó un usuario "dummy" en la plataforma, asociado únicamente a las asignaturas del concurso. Por lo tanto, con su token

de acceso es imposible acceder al contenido del resto de asignaturas y cursos impartidos en la Universidad.

Este usuario "dummy" o Bot, tiene permisos de profesor limitados, lo que le permite visualizar las entregas de todos los grupos, acceder a ellas, descargar sus archivos asociados y corregirlos en la interfaz de Moodle. Concretamente, el servicio STIC de la universidad ha creado un tipo de perfil con recursos limitados específicos llamado wsservice.

El bot no es un profesor completo, si no que tiene permisos de "non-edting teacher", esto es, puede leer, pero no editar los contenidos de la asignatura, lo que protege al sistema de que un bug borrase actividad de Moodle.

Al tener perfil de usuario, también puede acceder a las funcionalidades de corrección y descarga a través de la Interfaz de Programación de Aplicaciones (API) de Moodle.

Además del rol "non-editing teacher", al bot se le ha asignado un rol adicional creado por el STIC denominado "wsrole_icai", que le permite acceder a las funcionalidades de la API epecífícas necesarias para el proyecto.

La API de Moodle ofrece multitud de funciones que permiten en esencia, realizar las mismas funcionalidades que se pueden hacer a través de su interfaz de usuario. Para garantizar la seguridad, se ha minimizado el uso de estas funciones a las siguientes que son indispensables para el funcionamiento de la plataforma.

Función	Descripción
mod_assign_get_assignments	Devuelve los cursos y tareas del usuario asociado al token
mod_assign_get_submissions	Devuelve las entregas asociadas a la tarea que se pasa como parámetro

Tabla 2: Funciones de la API de Moodle utilizadas

Para garantizar la seguridad de los datos de los alumnos, el Bot creado no accede a la información de la integración de los grupos.

4.1.2 DESCARGA DELEGADA EN MOODLE

Siguiendo con este precepto de seguridad, se ha decidido no almacenar los archivos entregados por los alumnos en el Django utilizado como sistema central. Aunque Django permite el almacenamiento de documentos o ficheros, el hecho de que estos ficheros no se guarden en la plataforma añade una capa de seguridad a la plataforma.

En la base de datos SQLite3 únicamente se almacena la ubicación de estos archivos en Moodle. Para acceder a estos archivos, es necesario un token de acceso. Por ello, cuando un profesor solicita la descarga de un archivo, el sistema de gestión busca en su base de datos la ubicación de este archivo, le añade el token del Bot como parámetro de Autorización y hace una llamada a la API de Moodle para descargar dicho archivo. La lógica de la descarga de problemas se ampliará en el Capítulo 5. Diseño de la Solución Software.

De esta manera, es imposible acceder a los archivos de los alumnos desde la plataforma si no se cuenta con el token de autorización.

Además, esta solución aligera la carga de la plataforma, que no tiene que almacenar grandes ficheros o archivos, únicamente la ubicación de estos. El backend de la plataforma se ha estructurado como un servidor ligero, que se limita a gestionar el flujo de datos. Este servidor ligero no está escalado para carga de datos, para un volumen de carga y descarga de datos grande como sí lo está un backend pesado que maneja datos masivos. La carga de datos en la plataforma del MTC, por lo tanto, supondría un cuello de botella, que se ha solventado con la propia arquitectura de Moodle, que tiene un backend escalado a toda la actividad de la Universidad.

Con la decisión de delegar la descarga en Moodle, no es necesario crear un backend pesado en la aplicación, pues llamamos a Moodle para gestionar esta descarga. Se ha aprovechado la infraestructura de backend que tiene Moodle para escalar la plataforma, a costa de que la interacción con la infraestructura es más compleja.

CIBERSEGURIDAD

Siguiendo con la seguridad de la plataforma, se han implementado las siguientes medidas.

4.1.3 USUARIOS Y AUTENTICACIÓN

En Django, un usuario que representa a una persona o entidad que puede acceder al sistema. Los usuarios se utilizan para autenticar a las entidades que acceden al sistema y determinar los permisos que tienen, es decir, las acciones que pueden realizar.

Django proporciona una implementación predefinida de los usuarios y permisos que se ha utilizado como base para implementar el sistema de autenticación y autorización del MTC. Por lo tanto, la autenticación está centralizada en el sistema de gestión, utilizando las herramientas que proporciona el propio Django para ello.

Siguiendo los requisitos funcionales, se han creado 3 tipos de perfil de usuario, cada uno con unos permisos asociados:

- Profesor
- Pantalla
- Investigador
- Admin

Pantalla

Los usuarios de tipo pantalla únicamente pueden acceder a las ventanas de visualización de la aplicación. Además, no pueden realizar ninguna acción de escritura, modificación o eliminación sobre la base de datos, únicamente de lectura.

El uso del usuario de tipo pantalla está previsto para las pantallas inteligentes del Comillas Conecta Lab, por lo que se limitan sus permisos a visualización, de forma que los alumnos, que tienen acceso a la interfaz táctil de las pantallas del CCL, no puedan acceder y modificar información que no les corresponda.

Profesor

Los usuarios de tipo profesor tienen acceso a la ventana de corrección, así como a las de visualización para poder ver la evolución del concurso desde sus dispositivos móviles. En las funciones de corrección se les da acceso a las funcionalidades visualización y escritura en la base de datos, pudiendo descargar los archivos de las entregas y modificar las notas de las mismas.

Investigador

El perfil de investigador únicamente tiene acceso a la interfaz de descarga de datos, no pudiendo acceder a la herramienta de corrección ni a las interfaces de visualización en tiempo real.

Admin

El perfil de Administrador tiene acceso total a todas las acciones de la plataforma, permitiendo gestionar usuarios y permisos, además de acceso total a la base de datos.

4.1.4 TOKEN DE ACCESO A MOODLE

Aunque explicado en la subsección de Conexión con Moodle, merece la pena mencionar el Bot generado para gestionar el acceso a Moodle de la plataforma. En lugar de utilizar el token de cada profesor que participa en el concurso, se ha decidido emplear un único token de acceso para un perfil creado única y exclusivamente para el concurso que limita las acciones que se pueden realizar en Moodle con él.

Así, en lugar de tener acceso desde la plataforma a todos los cursos impartidos por cada uno de los profesores que toman parte en el concurso, únicamente se tiene acceso a las asignaturas de Análisis Matemático y Cálculo Vectorial y de Álgebra y Geometría que forman parte del concurso, y a la asignatura de Moodle experimental de pruebas, que se ha creado para las pruebas unitarias y de validación de la plataforma.

La creación de un token único evita tener que almacenar en la información de cada usuario de tipo profesor, su token de acceso a Moodle, previniendo una posible brecha de información en Moodle. Además, como se ha mencionado anteriormente, el bot creado por STIC únicamente tiene acceso limitado a la información del MTC de estas asignaturas.

Además, garantiza que el uso de la plataforma no está limitado a los profesores que imparten las asignaturas de Análisis Matemático y Cálculo Vectorial y Álgebra y Geometría. Si en algún momento puntual se necesita el apoyo de un profesor de la universidad para una prueba del concurso, no es necesario inscribir a este en la asignatura correspondiente de Moodle. Se puede crear un perfil para éste en la plataforma, y con el token del bot, participar de forma normal en la corrección del concurso.

4.1.5 FIREWALL

El propio firewall del servidor de la Universidad donde está desplegada la aplicación del MTC supone una capa de seguridad extra para la plataforma.

Este firewall ha supuesto muchas de las tomas de decisiones de la arquitectura, bloqueando la idea inicial de desplegar una comunicación por websockets debido al bloqueo/filtrado de puertos que realiza el firewall del servidor y descartando algunas de las arquitecturas planteadas por la incapacidad de desplegar aplicaciones P2P sobre la wifi eduroam de Comillas.

Para sobrepasar el firewall del servidor y permitir la comunicación por websockets, se ha planteado la posibilidad de desplegar un Daphne sobre el Apache o incluso de abrir otros puertos, no obstante, estas opciones fueron finalmente descartadas por la imposibilidad de modificar el servidor de la Universidad.

ARQUITECTURAS DESCARTADAS

En la fase inicial de desarrollo se analizaron diferentes arquitecturas para la plataforma del MTC. Tras estudiar los requisitos, se plantearon diferentes posibilidades de arquitectura, y

finalmente, tras analizar los pros y contras de cada una, se optó por la arquitectura explicada en el inicio de este mismo capítulo.

Las arquitecturas que se evaluaron y plantearon como posibles infraestructuras para el MTC son las siguientes:

Arquitectura planteada	Motivos descarte
Datos en nube de Microsoft	Seguridad, eficiencia y acceso a datos
Arquitectura completamente distribuida en local	Escalabilidad, seguridad, acceso a Moodle múltiple
Modelo P2P puro sobre hotspot	Accesibilidad, alcance, espacio
Modelo P2P con servidor ligero	Escalabilidad, alcance, limitación de gestión post-prueba
Plugin Moodle	Seguridad, complejidad técnica, tiempos de respuesta de interfaz de Moodle

Tabla 3: Arquitecturas descartadas

- **Datos en nube de Microsoft**

  Se estudió una arquitectura en nube de Microsoft, servicio que utiliza la Universidad como herramienta de soporte a la actividad docente. La arquitectura planteaba tener como base de datos una serie de ficheros ubicados en la nube de la Universidad en Teams. Sin embargo, la arquitectura no se consideró óptima en términos de eficiencia y acceso a la base de datos basada en ficheros.

  Además, en cuanto a seguridad, esta arquitectura supondría la descarga de ficheros con la información del concurso en los dispositivos en local de los profesores que accedieran a su edición, poniendo en riesgo la seguridad e integridad de los datos del concurso.

- **Modelo P2P local sobre hotspot**

  Originalmente se planteó una arquitectura local peer-to-peer sobre un hotspot.

  El sistema propuesto planteaba una comunicación P2P de clientes pesados sin servidor, que se comunicasen con las visualizaciones por conexiones peer-to-peer a través de un hotspot.

  La red de Comillas planteaba el problema de utilizar una conexión P2P sobre el wifi de la Universidad, por lo que surgió la posibilidad de crear esta arquitectura en local sobre un hotspot creado desde un dispositivo móvil.

  Se realizaron pruebas de distancia de emisión en el Comillas Conecta Lab y de conectividad con el hotspot, descartándose la opción de P2P sobre hotspot por el corto alcance y escalabilidad de la solución propuesta.

  Ante la falta de escalabilidad de la solución, se planteó un modelo P2P distribuido con un servidor ligero.

- **Modelo P2P distribuido con servidor ligero**

  Se planteó una arquitectura totalmente distribuida con clientes pesados y servidor ligero. Cada uno de los profesores y pantallas serían un cliente pesado que realizarían acciones de gestión del concurso y accederían individualmente a la API de Moodle.

  La arquitectura distribuida puede ser una buena opción para proyectos que necesitan alta escalabilidad y disponibilidad de los datos. Sin embargo, es más compleja de implementar, pues implica la gestión de múltiples nodos que se comunican entre sí. Además, para la anexión de un nuevo cliente (profesor o pantalla) a la plataforma es necesario instalar el software de gestión correspondiente en el dispositivo.

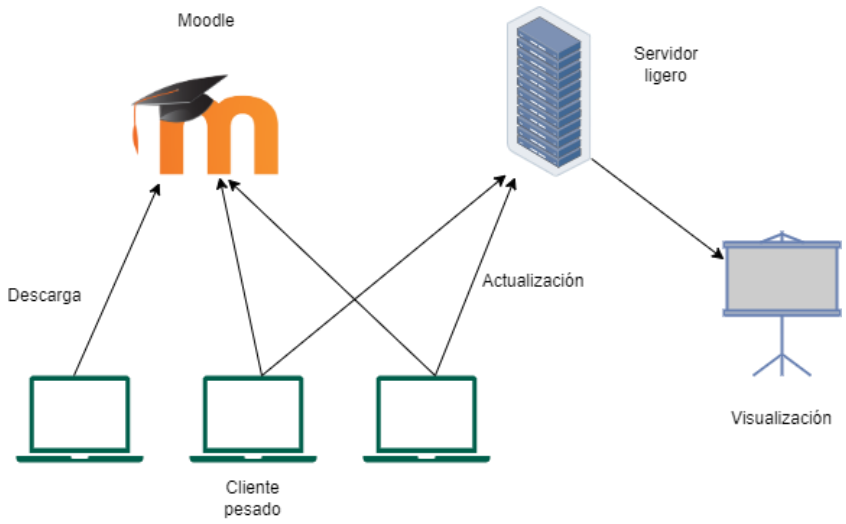


Ilustración 26: Boceto de arquitectura MTC distribuida

Por otra parte, en cuanto a la seguridad, el acceso de cada uno de los clientes pesados a Moodle supone una brecha de seguridad innecesaria. Con la arquitectura Django, es el servidor el único que accede a la API de Moodle con las credenciales del bot, permitiendo así limitar y controlar el acceso a la plataforma de enseñanza.

- **Módulo de control local con arquitectura de comunicación por wifi**

  Antes de saber que teníamos disponible un servidor para desplegar la arquitectura y como alternativa a la solución finalmente desplegada se planteó el uso de un servidor en local. Por incompatibilidades con la red wifi, este servidor local se montaría obre un Hotspot.

  Esta posibilidad planteaba alojar la plataforma actual (Django +SQLite) en un ordenador que funcionase como servidor local y que permitiese conectarse a los dispositivos de su misma red. Esta solución sería capaz de dar soporte al concurso durante la duración de las pruebas, necesitando conectarse todos los profesores y pantallas a la misma wifi para acceder a la plataforma a través de la IP local del ordenador central. No obstante, no sería posible la corrección de los problemas post prueba a distancia, es decir, que cada profesor corrigiese los problemas restantes, que no fuesen corregidos durante la duración de la prueba, desde una localización

diferente. Tras la prueba, la plataforma se cerraría en local y no se podría acceder a ella para corrección, visualización de los rankings, descarga de datos…

Además, sería necesaria una red wifi que tuviese el alcance necesario para abarcar todo el Comilla Conecta Lab.

Como se ha mencionado en la arquitectura P2P+Hotspot, se realizaron pruebas de alcance de la red hotspot en el CCL, evaluando su alcance y escalabilidad.

La plataforma en local limita la escalabilidad y accesibilidad. Además, pueden existir problemas de seguridad en caso de que la comunicación se realice a través de una red no segura.

Por lo tanto, y ante la posibilidad de desplegar la solución sobre el servidor de la Universidad, la opción se descartó por la necesidad de tener un sistema escalable y seguro, que gestione datos en tiempo real, pero también permita la gestión post-prueba.

- **Plugin Moodle**

  Se estudió la posibilidad de crear un plugin personalizado dentro de Moodle de forma que toda la gestión y actividad del concurso se realizase en la plataforma de la Universidad.

  Moodle ya cuenta con perfiles con diferentes niveles de accesos para alumnos, profesores, administradores, con permisos más o menos limitados en función de su rol. Además, el requisito de utilizar Moodle como plataforma para la subida de los problemas por parte de los alumnos, hizo plausible esta arquitectura.

  Sin embargo, esta idea se descartó por varias razones.

  Primero, Moodle es la plataforma utilizada por toda la Universidad para su actividad docente. Teniendo una arquitectura definida, la creación de un plugin podría requerir cambios estructurales en la plataforma, necesitando grandes recursos para poder realizarse. Además, podría suponer un parón de la plataforma si el plugin accediese a servicios indebidos o colapsase el sistema, lo que tendría consecuencias en toda la actividad docente de la Universidad. Más allá de esto, se debería tener acceso a la plataforma para Moodle para la programación del plugin, hecho improbable.

En segundo lugar, uno de los principales problemas de Moodle, que se ha tratado en el Estado de la cuestión es la lentitud de su interfaz. Por muy rápido que sea el backend de Moodle, su interfaz tiene tiempos de respuesta muy lentos. Al plugin personalizado para la gestión del MTC se accedería a través de la interfaz, ralentizado las tareas de corrección y gestión del concurso, uno de los problemas principales a corregir del sistema anterior.

La creación del plugin personalizado se descartó, por tanto, por su complejidad técnica y la necesidad de tener acceso completo a Moodle para su programación. Además, la plataforma no abordaría uno de los requisitos esenciales del sistema, que es la reducción de los tiempos de gestión para obtener un sistema en “tiempo real”.

5. DISEÑO DE LA SOLUCIÓN SOFTWARE

En este capítulo se describirán cada uno de los módulos o subsistemas que componen la solución software del MTC.

DISEÑO DE LA BASE DE DATOS

En esta sección se establece una descripción detallada de las tablas de la base de datos utilizada por la plataforma. El diseño de las tablas de la base de datos viene marcado por los requisitos funcionales de exportación de datos, por el propio funcionamiento de la aplicación y por los datos extraídos de Moodle, de tal manera que se he intentado replicar la estructura de los datos que el crawler recibe de la API de Moodle. Para su diseño se han seguido los RF20, 21, 22, 23 y 24.

Como se ha descrito en la estructura general de la arquitectura, se ha utilizado una base de datos relacional SQLite3.

Los objetos Django que el sistema convierte en tablas de la base de datos SQLite3 son los siguientes:

Asignatura

Información de las asignaturas que componen el concurso (especificar que se trata de una asignatura de Moodle, es decir, asignatura+año)

- MoodleId (Integer). Identificador único de la asignatura que coincide con el id que Moodle asigna a la misma. Clave Primaria.
- Nombre (Char). Nombre de la asignatura
- Año (Integer). Año en el que se imparte la asignatura
- Bonus (Integer). Número de premios de velocidad con los que cuenta la asignatura para cada uno de los problemas de sus pruebas

Prueba

Información de las pruebas realizadas de cada asignatura.

- MoodleId (Integer). Identificador único de la prueba. Clave Primaria.
- Número (Integer). Número de Prueba. No es único. Puede existir una Prueba 1 de Álgebra 2023 y una Prueba 1 de Cálculo 2023
- Fecha (Date). Fecha de realización de la prueba.
- Asignatura (Asignatura). Asignatura a la que pertenece la prueba. Clave Foránea.

Grupo

Información de los grupos participantes en el concurso. No incluye la información de cada uno de los integrantes del grupo pues ésta se extrae de Moodle para el post-procesado del dato y es redundante su inclusión en la base de datos del concurso. Recalcando la ciberseguridad de la plataforma, con esta medida evitamos que el sistema acceda directamente a datos identificables del alumno.

Los grupos no tienen por qué ser los mismos para cada asignatura, pero si deben mantenerse intactos para todas las pruebas de una misma asignatura.

Las mesas del CCL se numeran en función del número de grupo y los alumnos tienen, tanto en las mesas como en las carpetas de material que se les otorgan, su número de grupo, además de su nombre, por lo que se ha añadido el número de grupo en las visualizaciones permitiendo a los profesores identificar a cada grupo rápidamente.

- MoodleId (Integer). Identificador único. Coincide con el identificador de cada grupo en Moodle. Clave Primaria.
- Nombre (Char). Nombre del grupo
- Número (Integer). Número de grupo. Se utiliza para identificar a cada grupo en el Conecta Lab.
- Asignatura (Asignatura). Asignatura a la que pertenece el grupo. Clave Foránea.

Problema

Información de los problemas de cada prueba del concurso. Coincide con la "clase" Tarea de Moodle.

- MoodleId (Integer). Identificador único. Coincide con el Id de Moodle para la tarea donde los grupos entregan sus soluciones. Clave Primaria.
- Número (Integer). Número de problema. No es único. Puede existir un problema 1 de la prueba 1 y un problema 1 de la prueba 2 de la misma asignatura.
- Prueba (Prueba). Prueba a la que pertenece el problema. Clave Foránea.

Entrega

Información sobre las entregas realizadas por los grupos. Coincide con una entrega de Moodle. Una diferencia sustancial con las entregas realizadas en Moodle por los grupos es que, en Moodle cuando se elimina una entrega o se modifica esta, la entrega anterior no se guarda. En la nueva plataforma de gestión, se guardan todas las entregas realizadas por un grupo, estén o no obsoletas, para poder sacar conclusiones en el análisis de datos.

- MoodleId (Integer). Id que asigna Moodle a la entrega. Clave Primaria junto con AttemptNumber
- AttemptNumber (Integer). Número de intento de la entrega. Se inicializa a 0 y se incrementa en 1 por cada reentrega que ocurra. Clave Primaria junto con MoodleId.
- Problema (Problema). Problema al que pertenece la entrega. Clave Foránea.
- Grupo (Grupo). Grupo que ha realizado la entrega. Clave Foránea.
- Tiempo (DateTime). Tiempo en el que se ha realizado la entrega en Moodle.
- Urls (JSON). Diccionario en el que se guardan las URLs de los archivos adjuntos en la entrega. Estas URLs apuntan a Moodle. Como ya se ha comentado, para agilizar la descarga y el almacenamiento de datos, en el sistema de gestión no se guardan los archivos subidos por los alumnos, sino que se almacena la ubicación de ese archivo en Moodle y en el momento que un profesor solicita su descarga, se llama a la API de Moodle para recuperar ese archivo. Este campo se vacía al realizarse una reentrega

porque, como Moodle borra los ficheros anteriores tras una reentrega, la URL quedaría obsoleta.

- Bonus (Boolean). Indica si la entrega tiene premio de velocidad o no. Tiene como valor por defecto False.
- Nota (Integer). Nota que tiene una entrega. Admite el valor NULL si la entrega aún no ha sido corregida.
- Corregido (Boolean). Indica si la entrega ha sido corregida. Por defecto, se inicializa a False.
- Descargado (Boolean). Indica si se han descargado los archivos de la entrega. Por defecto, se inicializa a False.
- Antigua (Boolean). Indica si existe una versión más reciente de esa entrega, es decir, si se ha producido una reentrega por parte del grupo. Si es así, el campo se pondría a True para indicar que la entrega está obsoleta. Por defecto, se inicializa a False.

5.1.1 DIAGRAMA DE TABLAS DE LA BASE DE DATOS

Las relaciones entre las tablas descritas pueden observarse en la Ilustración 27.

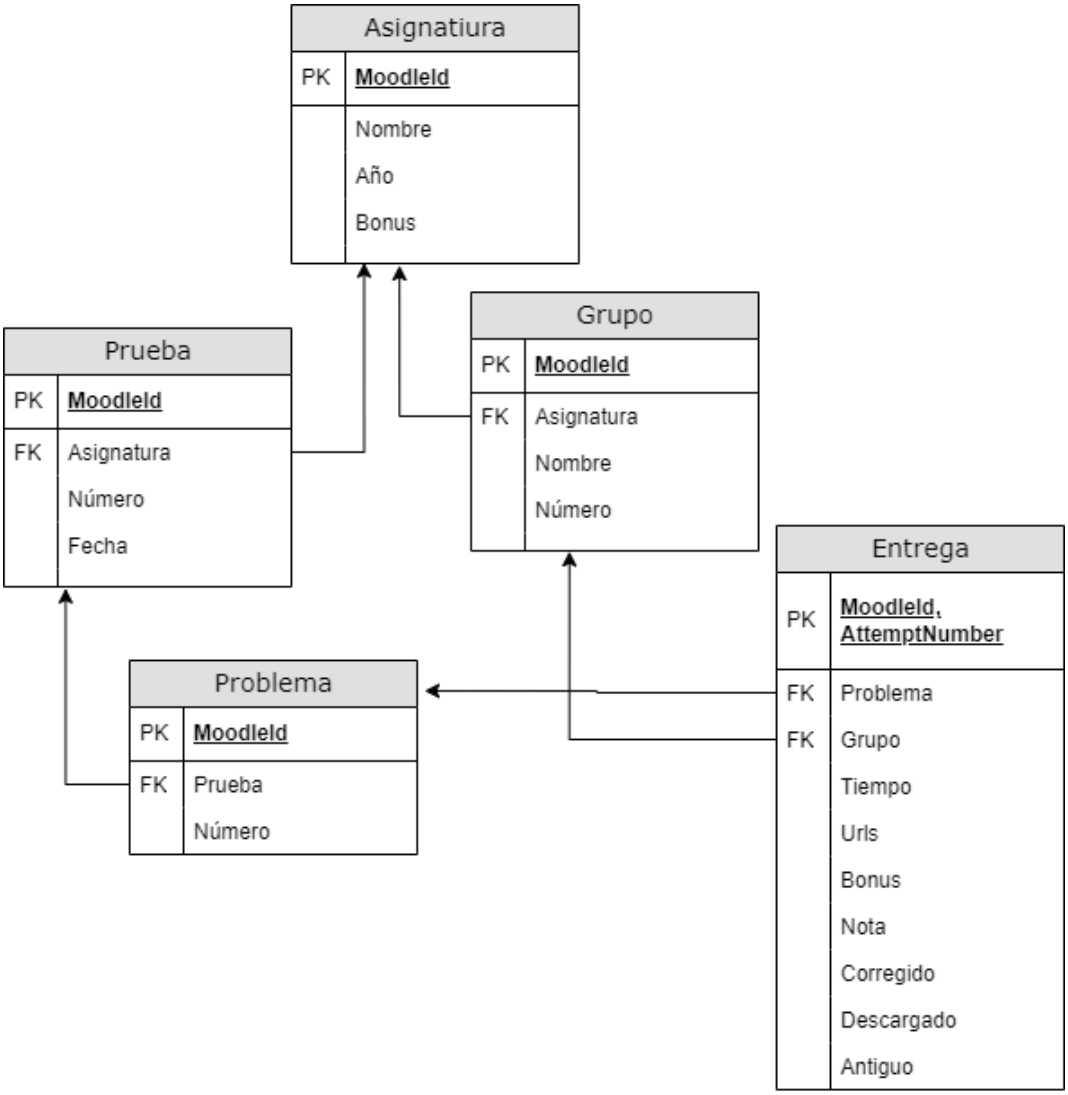


Ilustración 27: Diagrama de Tablas de la Base de Datos

5.1.2 ACCESO A LA BASE DE DATOS

En la segunda iteración de la plataforma se normalizó el acceso a la base de datos, de forma que todos los accesos a la base de datos para lectura, escritura, modificación se realizan a través de las funciones que se encuentran en el módulo utils.py, que puede consultarse en el ANEXO II de este documento.

CONEXIÓN CON MOODLE- CRAWLER

Este subsistema se encarga de gestionar las entregas de los estudiantes y almacenar los datos en la base de datos. Además, se encarga de enviar los datos de las entregas a la herramienta de corrección para su posterior procesamiento. Según el RNF37, *la gestión de datos debe permitir la sincronización de datos con Moodle, donde se recogen las entregas de los alumnos*. Para esta sincronización se ha diseñado el subsistema del Crawler.

El módulo de conexión con Moodle se ha estructurado como un hilo o thread que se ejecuta en paralelo al programa principal y está en escucha continua de las entregas que los alumnos en Moodle con actualizaciones periódicas de 1s. De esta manera no bloquea la ejecución del programa principal.

El crawler se activa únicamente cuando se accede a la prueba por primera vez, para evitar que el acceso de varios usuarios a la prueba lance varios hilos ejecutándose en paralelo y accediendo a la base de datos, creando duplicados y dificultando la gestión.

A través de la API de Moodle, el crawler realiza una petición con la función mod_assign_get_assignments, que devuelve todas las tareas de Moodle a las que el bot tiene acceso. Tras filtrar las tareas correspondientes a la prueba que se está llevando a cabo en ese momento, por cada tarea, que equivale a un problema MTC, se hace una petición a la API de Moodle que devuelve las entregas asociadas a cada tarea. Por cada entrega, el crawler lleva a cabo la siguiente toma de decisiones:

- Determinar si el problema es nuevo y debe ser corregido
- Detectar si el problema ha sido modificado o eliminado
- Asignar premios de velocidad a los problemas entrantes que los merezcan
- Adjudicar puntuaciones temporales

En la Ilustración 28 podemos observar el Diagrama de flujo de los agentes involucrados en la llegada de un problema a la plataforma. El alumno sube un problema a Moodle y Django, a través del Crawler manda continuas peticiones de actualización a la API de Moodle que

devuelve todos los problemas entregados durante el concurso. El crawler realiza tareas de procesado para determinar si los problemas son nuevos o ya se encuentran en la base de datos y detectar si algún problema de los que tiene almacenados ha sido borrado o modificado por un alumno. Estas decisiones de procesado pueden observarse en el Diagrama de Estados de la Ilustración 29. En función de estas decisiones, el crawler realiza tareas de escritura y de actualización en la base de datos.

En el ejemplo a continuación se pueden ver distintas situaciones del crawler en función de la acción llevada a cabo por el alumno en Moodle. También se muestra el caso en el que el crawler realiza una petición de actualización sin que se haya realizado ninguna acción nueva en Moodle desde la última petición. En este caso, el crawler evalúa esta circunstancia y no realiza ninguna modificación sobre la base de datos.

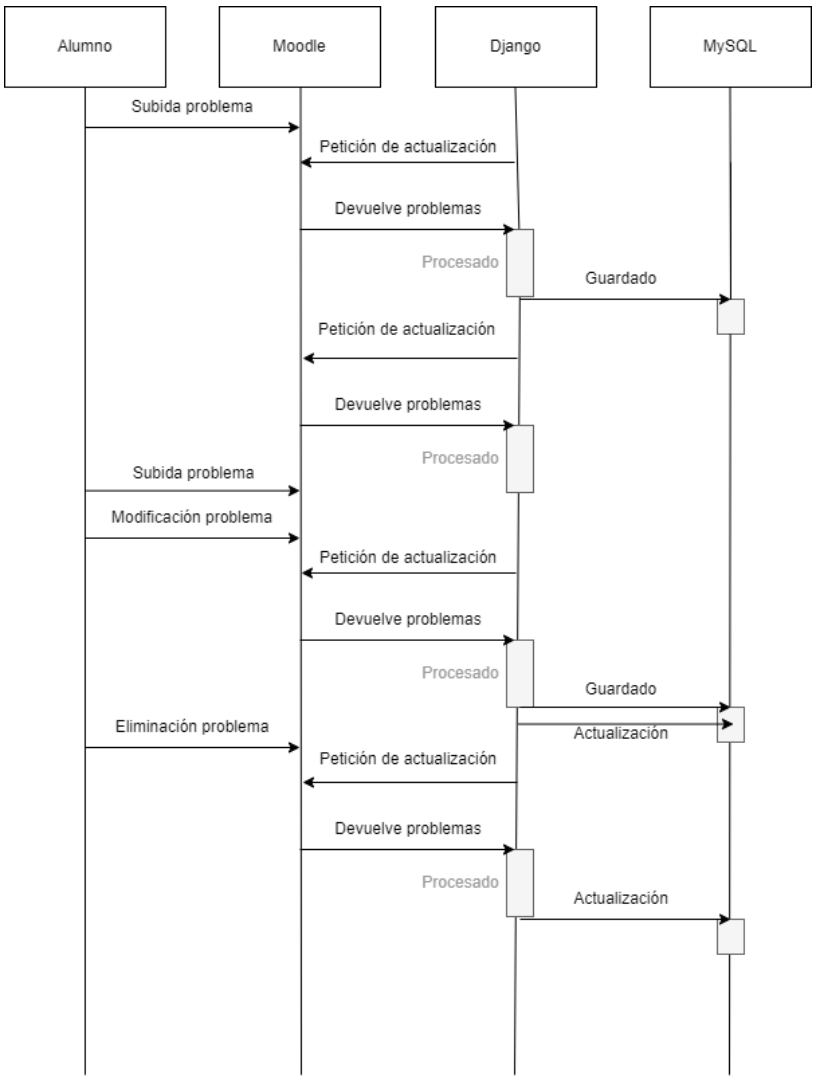


Ilustración 28: Diagrama de flujo de la llegada de un problema

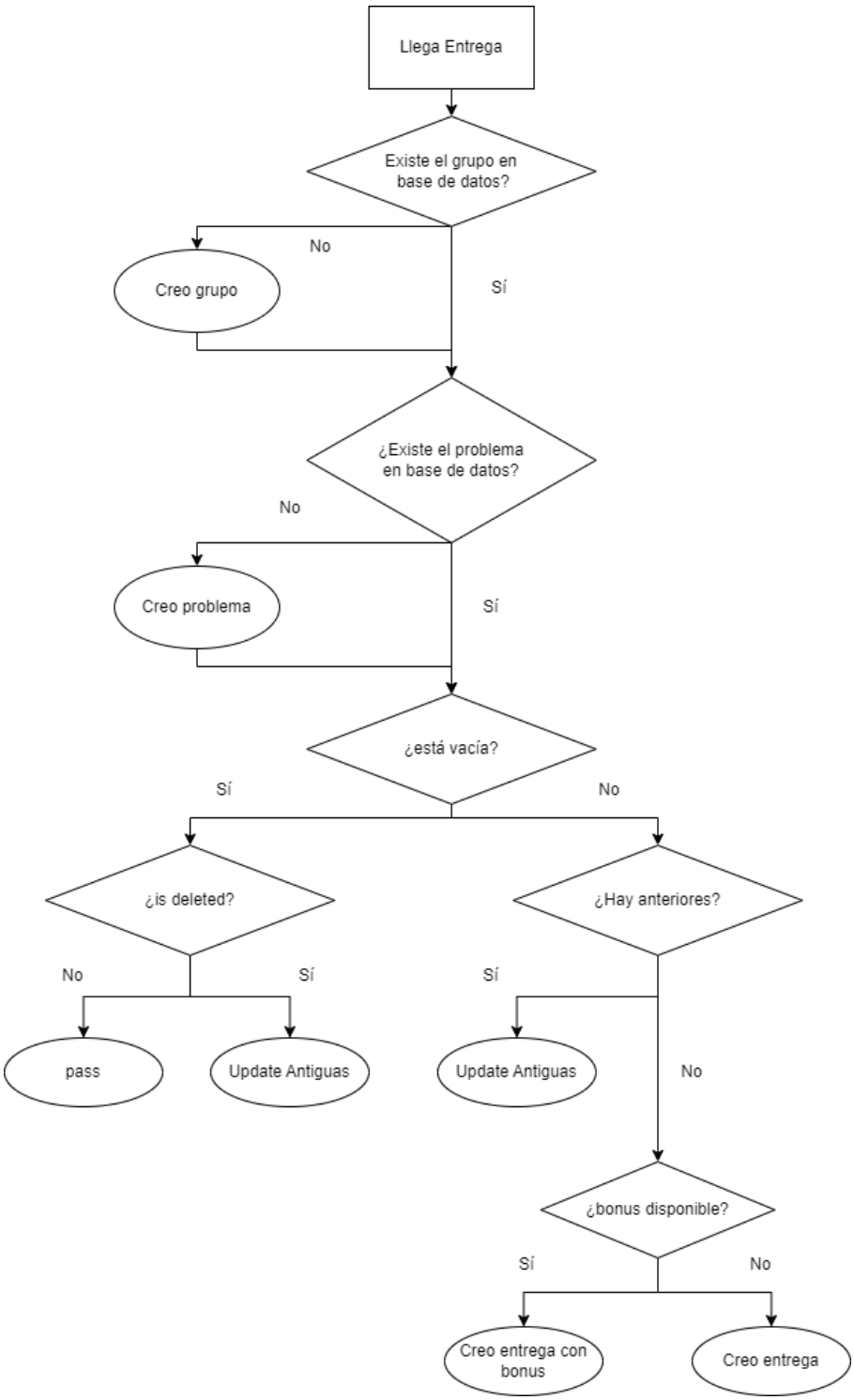


Ilustración 29: Diagrama de toma de decisiones del crawler

En la Ilustración 29 se observa el flujo de decisión que sigue el crawler de Moodle a la hora de considerar una entrega como nueva, modificada o borrada. En primer lugar, comprueba si existe en la base de datos el grupo y el problema a los que poder asociar la entrega. Si no existen, los crea con la información que devuelve la API sobre estos.

En segundo lugar, comprueba si la entrega contiene URLs. Las entregas que devuelve la API de Moodle pueden tener dos comportamientos diferentes: con URLs, lo que significa que la entrega es nueva o bien ha sido modificada. En este caso, contiene las URLs de los archivos entregados. Si no contiene URLs, pueden darse otros dos casos: o bien la entrega ha sido eliminada y Moodle mantiene la información de la entrega, pero no almacena los archivos entregados, o se ha creado un borrador para una entrega por parte de un alumno, pero todavía no se ha entregado.

Si seguimos el camino de la izquierda (no hay URLs), en primera instancia comprobamos si la entrega ha sido eliminada. En este caso, actualizamos las entregas anteriores de ese grupo y problema mediante la función update_antiguas(). Esta función lleva toda la lógica de marcar una entrega como antigua, eliminando bonus temporal si es necesario y eliminando las URLs de los archivos de Moodle asociados, pues estos ya no existen en la plataforma de e-learning. Marcar una entrega como antigua indica que ya no forma parte del concurso, y por tanto su calificación no sirve para las puntuaciones finales de los equipos. Sin embargo, se mantiene la información de las entregas antiguas en base de datos para fines estadísticos y analíticos.

Siguiendo la rama de la derecha desde la pregunta "¿está vacía?", comprobamos si existe alguna entrega anterior por ese mismo grupo para ese problema. Si es así, nos indicaría que la entrega ha sido modificada a través de Moodle y se vuelve a realizar la lógica de update_antiguas(). Si no hay entregas anteriores, indica que la entrega es nueva. En ambos casos, se comprueba si la entrega es candidata a bonus de tiempo y si es así, se le asigna. Si no, se crea la entrega con bonus=False.

5.1.3 MARCAPASOS

En la tercera iteración del ciclo iterativo se implementó un marcapasos para el crawler de Moodle. Este módulo se ha asociado a una actualización periódica de interfaz para garantizar un tiempo máximo de caída de 6 segundos. Sólo está activo si algún usuario está conectado al sistema. Si no, deja morir al hilo de forma natural, lo que hace que el bot únicamente esté realizando solicitudes a Moodle mientras un usuario del sistema lo requiera. comprueba que se ha iniciado una instancia del hilo para la prueba actual y que el hilo está vivo y no ha muerto por algún fallo de conectividad con Moodle. Si el crawler no ha sido iniciado, crea una instancia del hilo y la inicia. Si el crawler ha muerto, crea una nueva instancia del mismo y la inicia. Estas acciones se realizan a través de las funciones check_thread_status() y start_moodle() del módulo check.py que puede encontrarse en el código fuente del ANEXO II.

SISTEMA DE GESTIÓN CENTRAL

El sistema de gestión central se encarga de varias tareas entre las que se encuentran:

- Determinar las entregas que requieren prioridad de corrección
- Gestión automática de los premios de velocidad
- Administrar los enlaces de descarga de los ficheros entregados por los alumnos
- Controlar si se ha iniciado la corrección de alguna entrega por parte de un profesor e informar al resto
- Gestionar la corrección y puntuaciones
- Actualizar los rankings de acuerdo a la llegada de problemas y a las correcciones de los profesores

Dentro del sistema central existen los siguientes submódulos que se encargan de realizar las decisiones de gestión mencionadas.

5.1.4 DESCARGA DE FICHEROS

La descarga de los archivos únicamente puede realizarse por un usuario de tipo profesor o administrador, pues forma parte de las tareas de corrección.

Como ya se ha mencionado anteriormente, la descarga de ficheros es delegada. Es decir, en la base de datos de la plataforma, no se almacenan los ficheros y archivos entregados por los alumnos, sino que se almacena la ubicación de estos archivos en Moodle. Cuando un profesor solicita un archivo para su corrección, el sistema central llama a la API de Moodle con el token de Bot y devuelve al profesor el archivo.

En la Ilustración 30, se puede observar el flujo de la descarga de un problema y los agentes implicados. El flujo de actualización al resto de agentes no ha sido incluido pues ya ha sido explicado en diagramas anteriores. Puede consultarse en la *Ilustración 23: diagrama de comunicación con HTTPS*.

El profesor realiza la petición de descarga a la interfaz y ésta la comunica al servidor central. Django recupera la URL del archivo de la base de datos, agrega el token de acceso y le pasa la URL autenticada a la interfaz del profesor. Esta interfaz es la que redirige la solicitud a Moodle y Moodle le devuelve directamente el fichero a la interfaz web, que se lo muestra al profesor.

Así, la comunicación cliente-servidor Django es puramente de control, realmente ligera, mientras que transferencia de datos pesados de las cargas (alumno) y descargas (profesor) de ficheros se realizan únicamente contra el servidor de Moodle, cuyo backend sí que está preparado y escalado para gestionar estos volúmenes en la Universidad.

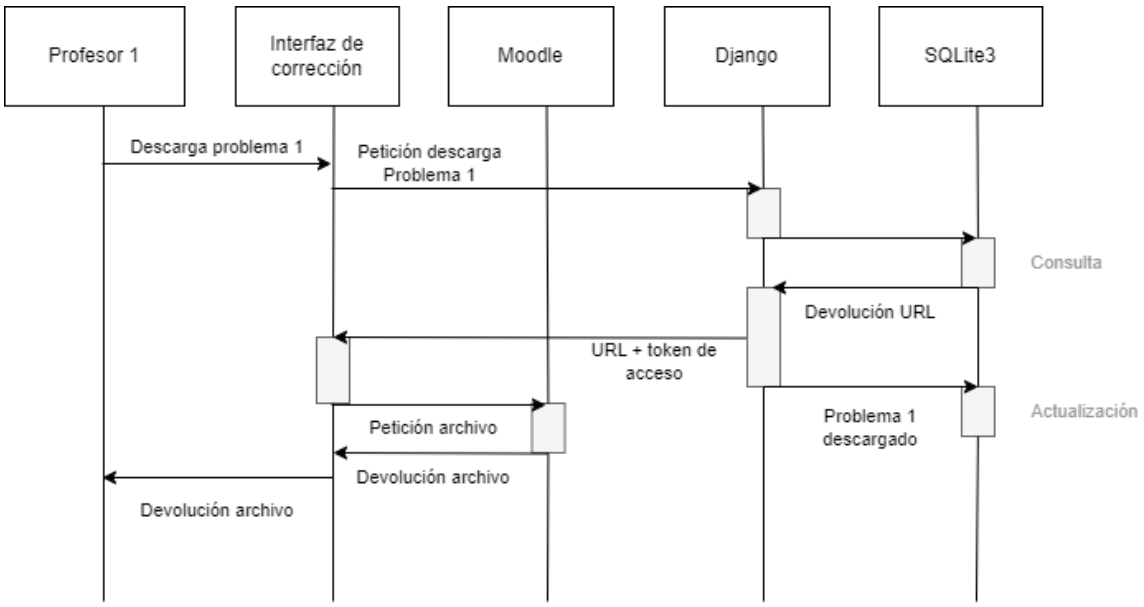


Ilustración 30: Diagrama de flujo de la descarga de una entrega

Las tareas de descarga se realizan desde la interfaz de corrección que se explicará más adelante. A través del botón de Download mostrado en la tarjeta de cada entrega, el profesor puede descargar los archivos correspondientes, como se puede observar en la Ilustración 31.

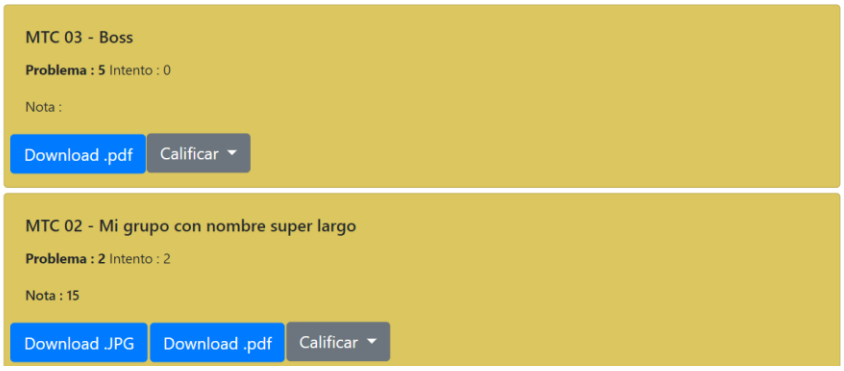


Ilustración 31: Interfaz de descarga

5.1.5 CORRECCIÓN

Las tareas de corrección, al igual que las de descarga están limitadas a los usuarios de tipo profesor o administrador y se realizan desde la interfaz de corrección de la plataforma.

El flujo de datos cuando se realiza una tarea de corrección es el que se puede observar en la Ilustración 32. Se pueden observar dos casos con dos comportamientos diferentes.

En el primero, el profesor pone una nota inferior a 10 y el sistema de control la registra en la base de datos. Podemos observar a continuación la solicitud de actualización por parte de otro profesor y de las pantallas. Esta se ha simplificado por simplicidad, pero el flujo completo de actualización puede observarse detalladamente en la *Ilustración 23: diagrama de comunicación con HTTP*.

En el segundo caso, la entrega está perfecta y el profesor decide calificarla con un 10. En este caso, el sistema de control comprueba si hay algún premio de velocidad disponible para ese problema y actúa sobre esta información, actualizando la nota de la entrega con un 10 o con un 15 si finalmente se le concede el premio.

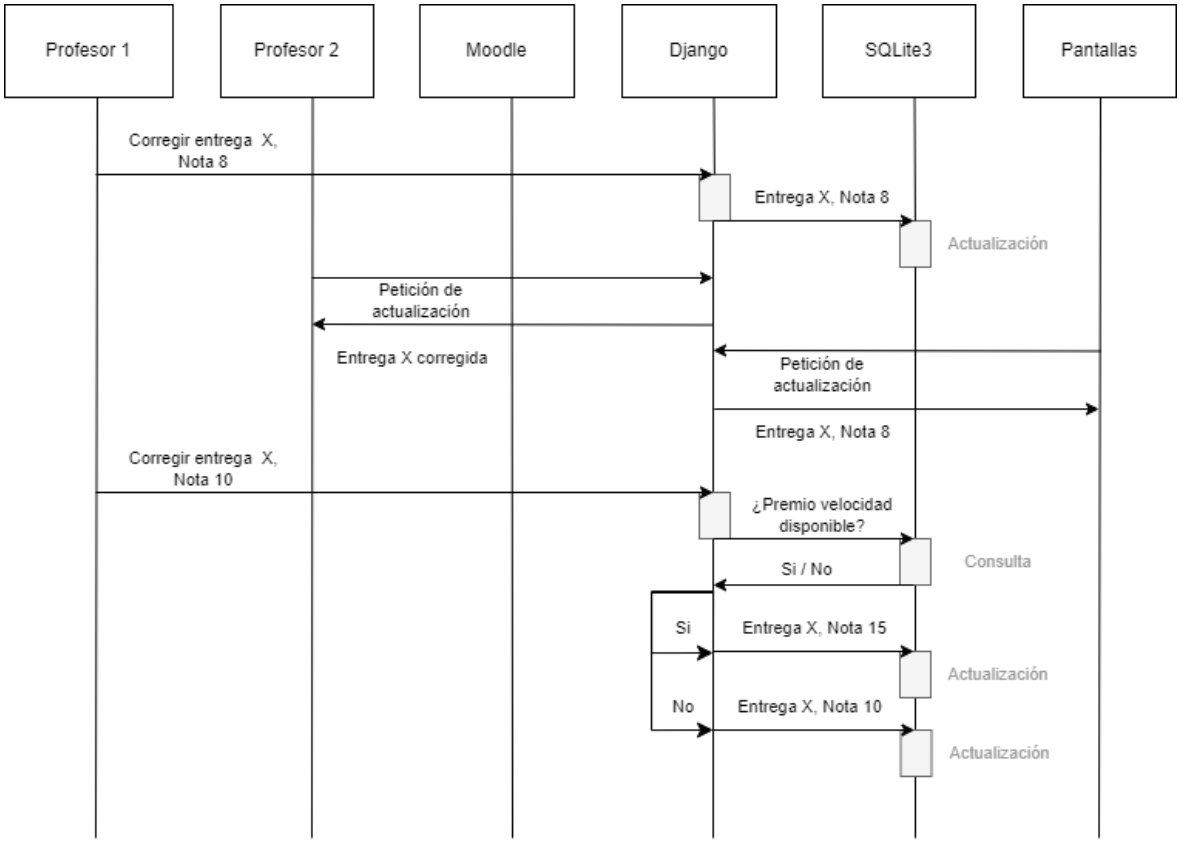


Ilustración 32: Corrección de una entrega

Según el RF18: *Debe existir una herramienta que permita filtrar los problemas entregados por número de problema para facilitar la división de la corrección por parte de los profesores.* Como medida cautelar de prevención, se ha implementado un sistema de comprobación para que, en cada entrega que sea calificada con un 10 se verifique si le corresponde el premio de velocidad.

De esta forma se evita el comportamiento que se observa en la Ilustración 33, donde un profesor corrige la entrega X con un 10, el sistema consulta la base de datos y determina que tiene bonus de tiempo disponible y lo califica con un 15. Posteriormente, el mismo profesor u otro diferente, corrige la entrega Y que ha sido entregada 80 ms antes que la X. El profesor considera que el problema está perfecto y lo corrige con un 10. El sistema considera que no quedan bonus de tiempo disponibles y le deja un 10 de nota. Así, el premio por velocidad que le correspondería a la entrega Y ha sido asignado a la entrega X, simplemente por orden de corrección.

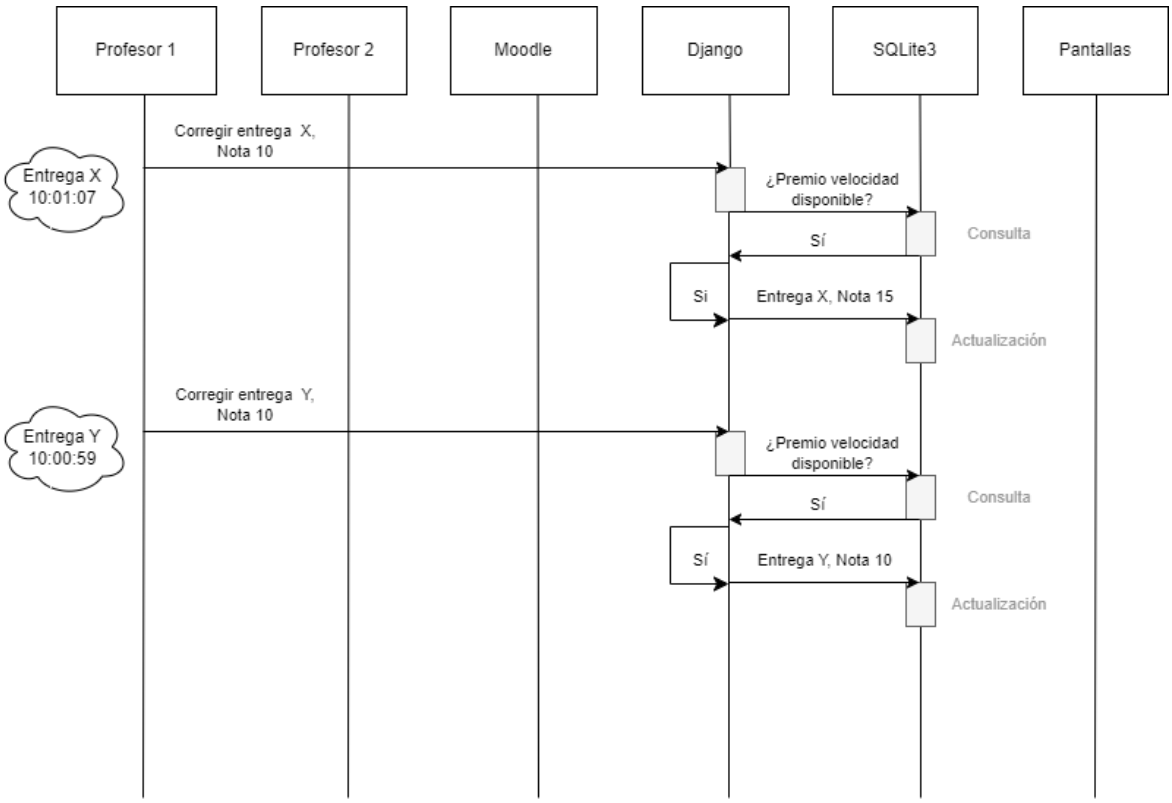


Ilustración 33: Flujo premio velocidad: caso a evitar

Con el sistema de comprobación de bonus de tiempo implementado, ocurriría una situación similar a la que se observa en la Ilustración 34. En este caso, cuando el profesor califica el problema Y con un 10, el sistema detecta que no hay bonus de tiempo disponibles. Sin embargo, compara el tiempo de entrega de Y con las entregas que tienen bonus de tiempo para este problema. Al detectar que X, marcado con bonus, se entregó más tarde, el sistema de comprobación quita el bonus a X para asignarlo a Y.

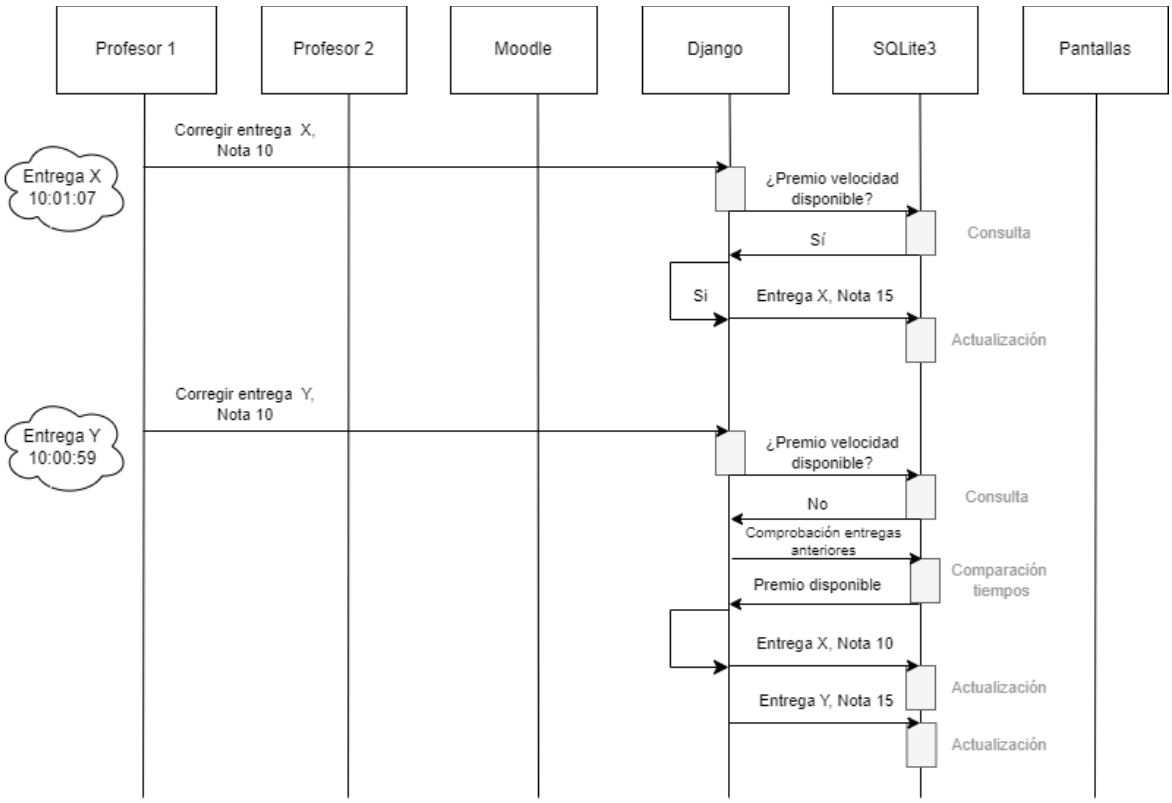


Ilustración 34: Flujo premio velocidad: sistema de comprobación

El flujo de actualización al resto de agentes no ha sido incluido en ambos diagramas pues ya ha sido explicado en diagramas anteriores. Puede consultarse en la *Ilustración 23: diagrama de comunicación con HTTPS.*

VISUALIZACIÓN Y DISEÑO DE LA GUI

Esta subsección se enfoca en uno de los aspectos fundamentales de diseño de software y de esta aplicación en concreto: la visualización.

La usabilidad era uno de los principales requisitos no funcionales: contar con una interfaz de usuario intuitiva y fácil de usar. En este sentido, para el diseño de la GIU, se han tenido en cuenta los diferentes tipos de usuario que forman parte de la plataforma y las funciones que realizan cada uno, buscando que las tareas de corrección y descarga se puedan realizar de manera eficiente y sin confusiones, y que la visualización en las pantallas del Conecta Lab sea limpia y clara.

5.1.6 SCOREBOARD Y RANKINGS

En primer lugar, la visualización de los rankings de la evolución del concurso se ha realizado manteniendo el formato que se utilizaba el curso anterior, con las siguientes visualizaciones, que pueden consultarse en la sección de Análisis y Definición Formal de Requisitos :

1. Las pantallas de visualización en tiempo real deben contar con las tres vistas que se utilizaban en el año 21/22 y que dan la siguiente información.
2. Una tabla con la puntuación general de los grupos, en la que aparezca la puntuación total de los problemas corregidos, la puntuación en revisión de los problemas entregados, pero aún sin corregir, y la suma de ambos, que supone la puntuación total en la prueba.
3. Una tabla en la que aparezca todos los problemas y los grupos y en la que sea sencillo ver qué problemas han sido entregados por cada grupo, cuáles han sido corregidos y tienen la puntuación máxima, cuáles cuentan con premio de velocidad (o son candidatos a él) y cuáles no han sido entregados.
4. Una gráfica en la que se muestre una comparativa de la actuación de los grupos en la prueba, en relación al grupo con mayor puntuación total en cada momento.
5. La información de los rankings debe actualizarse en tiempo real, o en su defecto, cada pocos segundos.

6. Debe existir un ranking final que permita ver la puntuación de los grupos en la prueba (como una "foto finish"). Este debe ser similar y plasmar toda la información que contiene la tabla del R27.
7. Debe existir un ranking global que permita ver la actuación de los grupos en todas las pruebas realizadas hasta la fecha de cada asignatura que muestre la puntuación de cada prueba y la suma de todas ellas.

5.1.6.1 Tiempo Real

Para la visualización en tiempo real, se ha diseñado la siguiente vista, que cuenta con las tablas y gráficas explicadas en los puntos 2,3,4.

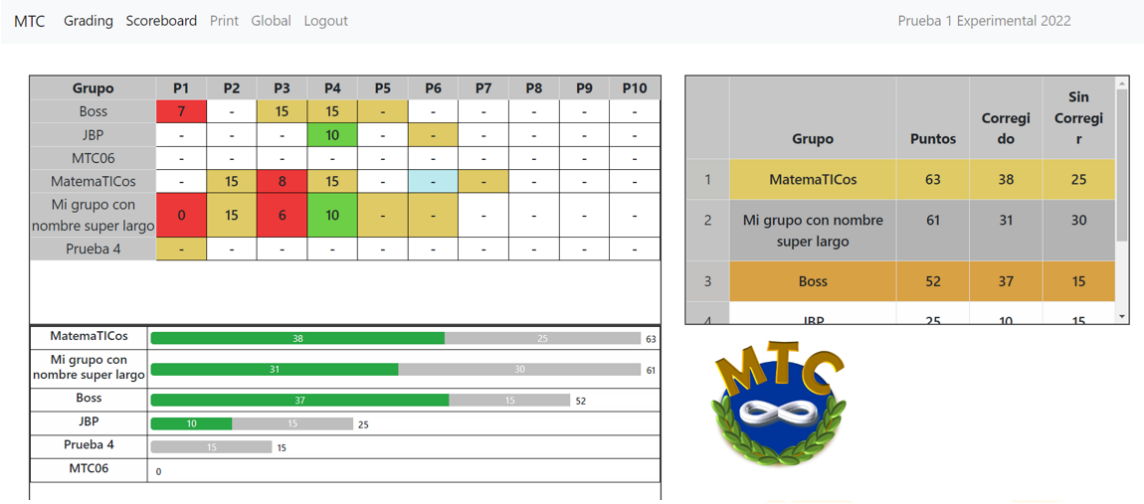


Ilustración 35: Vista de ranking en tiempo real

Como se ha mencionado anteriormente en el documento, para las imágenes de la interfaz de la aplicación se han utilizado datos de prueba y no datos reales de los grupos formados por alumnos.

En la primera tabla, se muestra la información completa por grupo y problema. Para facilitar la visualización, se han establecido los siguientes códigos de colores, en colaboración con los profesores del departamento de Álgebra y Cálculo para garantizar uniformidad y fácil comprensión:

- Dorado: las entregas que cuentan con premio de velocidad. Pueden estar corregidas (15) o sin corregir (-).
- Azul: las entregas que están pendientes de corrección.
- Verde: corregidas con puntuación perfecta, pero sin bonus de tiempo.
- Rojo: corregidas sin puntuación perfecta.
- Blanco: sin entregar

En la gráfica inferior, se muestra la evolución de cada grupo en comparación con el grupo que más puntos lleva conseguidos en la prueba. La parte verde de la barra indica los puntos ya corregidos, mientras que la gris, los que están pendientes de corrección. Como se explica en el Funcionamiento del MTC, a cada entrega que el sistema recoge de Moodle, se le asigna una puntuación temporal de 10 puntos o 15, en caso de contar con bonus de tiempo. Cuando los profesores corrigen, esta nota temporal se sustituye por la definitiva que indica el profesor. El número situado a la derecha de la barra indica el número total de puntos que el equipo podría tener en caso de que los problemas a corregir tuviesen una puntuación perfecta.

En la tabla de la izquierda, se muestra un ranking ordenado de los equipos por puntos, indicando también el total de puntos corregidos y sin corregir.

Esta información se actualiza cada pocos segundos mediante peticiones de actualización por HTTP, como se muestra en la *Ilustración 23: diagrama de comunicación con HTTP*.

La visualización de los rankings está pensada para las pantallas del Comilla Conecta Lab, y su aspecto, tamaño de tablas, letra, colores … se han adaptado para tener una visualización óptima desde estas pantallas.

Para lidiar con el número elevado de grupos, en la visualización, las gráficas oscilan automáticamente si el número de grupos del concurso excede el total de la capacidad de las pantallas.

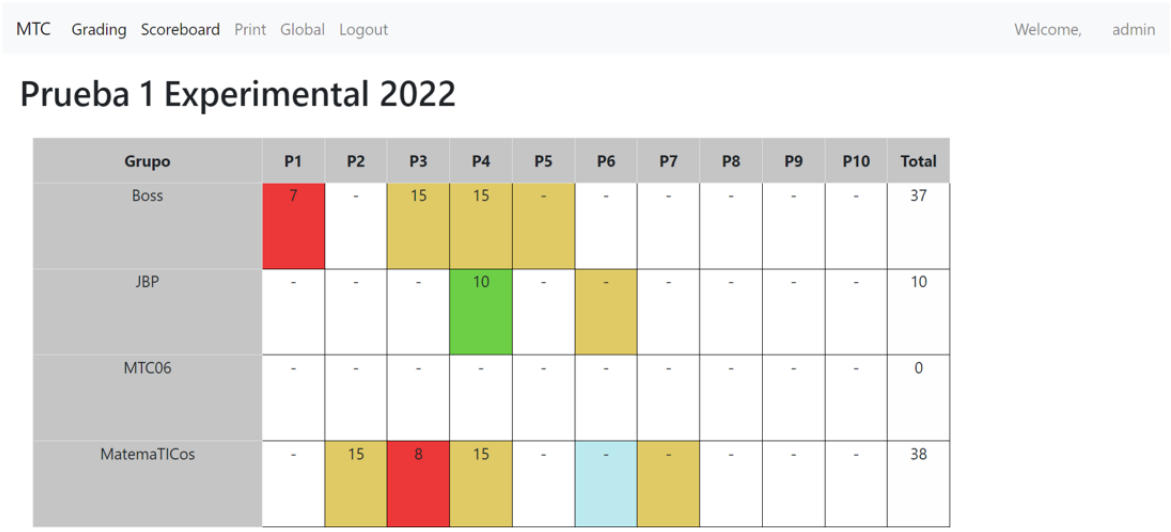
En la parte derecha de la interfaz se ha dejado un espacio para superponer en cada pantalla del CCL un cronómetro. Una de las mejoras planteadas para un futuro del

sistema es la integración de dicho cronómetro y su control centralizado por parte de los profesores en el sistema.

5.1.6.2 Rankings estáticos

En los puntos 6 y 7 mencionados anteriormente se establece la necesidad de contar con un ranking final, que permita ver la puntuación de los grupos en la prueba (como una "foto finish" temporal), y de un ranking global, que permita ver la actuación de los grupos en todas las pruebas realizadas hasta la fecha.

El diseño del primero es muy similar a la tabla de puntuaciones del ranking en tiempo real, contando con el mismo código de colores.



Grupo	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Total
Boss	7	-	15	15	-	-	-	-	-	-	37
JBP	-	-	-	10	-	-	-	-	-	-	10
MTC06	-	-	-	-	-	-	-	-	-	-	0
MatemaTICos	-	15	8	15	-	-	-	-	-	-	38

Ilustración 36: Ranking final MTC

Para el segundo, se mostrarían las puntuaciones de todas las pruebas de la misma asignatura realizadas hasta la fecha. En el caso de la imagen que se muestra a continuación, se trata de la primera prueba de lo que constituye la asignatura "Experimental", en la que se ha trabajado para realizar las diferentes pruebas que se tratan en el 6.

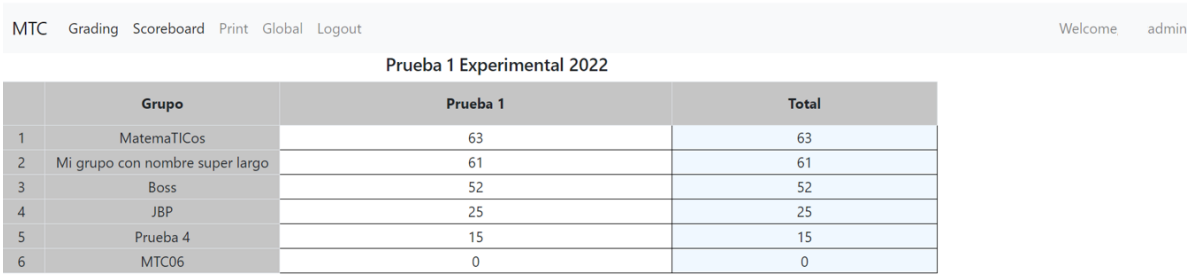


Ilustración 37: Ranking global MTC

5.1.7 VISTAS Y NAVEGACIÓN

Para la navegación entre las diferentes vistas o pantallas de la GUI se ha diseñado un menú de navegación que permite a los usuarios desplazarse por las diferentes vistas.

Este menú es diferente para cada tipo de usuario, permitiendo a los administradores acceder a todas las funcionalidades de la aplicación, pero limitando las vistas a las que tienen acceso los usuarios de tipo profesor y pantalla.

En las siguientes imágenes se pueden apreciar las diferencias entre la barra de navegación para un usuario de tipo admin y un usuario de tipo pantalla. El nombre de usuario y la prueba en la que nos encontramos se encuentran a la derecha de la barra de navegación. Como se puede observar, el usuario de tipo pantalla no puede acceder a la vista de corrección y descarga.

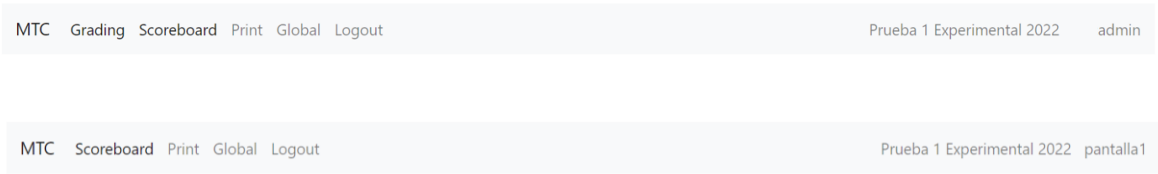


Ilustración 38: Menú de navegación

Los botones de la barra de navegación son los siguientes:

- MTC: lleva a la vista principal, en la que el usuario elige la prueba a la que desea acceder.

- Grading: vista de corrección y descarga.
- Scoreboard: pantalla de ranking en tiempo real.
- Print: ranking final o "foto finish" de la prueba.
- Global: ranking global de la asignatura.
- Logout: cierra sesión y lleva a la página de login.

Además del menú de navegación, desde la vista principal, los usuarios de tipo administrador pueden acceder a la interfaz de descarga de datos, que se explicará a continuación.

La navegación de los usuarios por la aplicación es la siguiente:

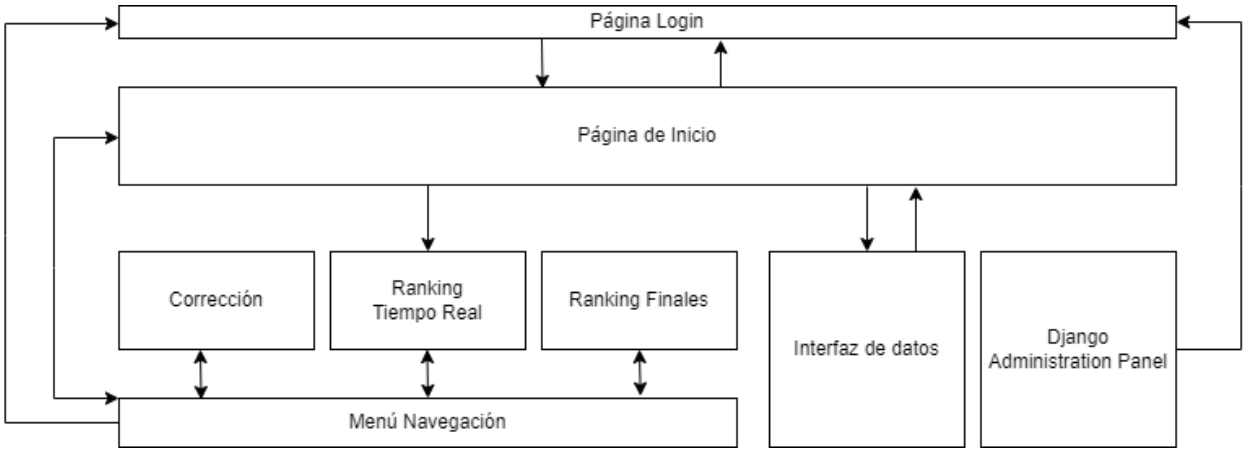


Ilustración 39: Navegación por la aplicación MTC

Desde la página de inicio, los usuarios autenticados acceden a la ventana principal o página de inicio, desde la que seleccionan la prueba a la que quieren acceder.

Una vez seleccionada la prueba, la aplicación redirige al usuario al ranking en tiempo real de la prueba. Desde el menú de navegación [Ilustración 38] situado en la parte superior de la interfaz, el usuario puede acceder al resto de vistas de la aplicación: corrección y ranking finales y globales y a la página de inicio, permitiendo al usuario cambiar de prueba sin necesidad de volver a hacer login. La herramienta de corrección y rankings estáticos también cuentan con menú de navegación, permitiendo al usuario desplazarse entre unas vistas y otras fácilmente.

Desde la página de inicio, los administradores e investigadores también pueden acceder a la interfaz de datos.

Al panel de administrador que implementa Django y que permite la gestión de los usuarios y sus permisos y el acceso a la base de datos de la aplicación se accede directamente a través de su URL y su acceso está limitado a los administradores.

La navegación a los distintos tipos de usuarios está limitada siguiendo los principios de autorización y los requisitos de seguridad explicados en la sección de Análisis y Definición Formal de Requisitos (RF1-RF12).En las siguientes imágenes se pueden observar las páginas y, en consecuencia, las acciones permitidas a cada tipo de usuario (marcadas en verde) y aquellas para las que no tienen autorización.

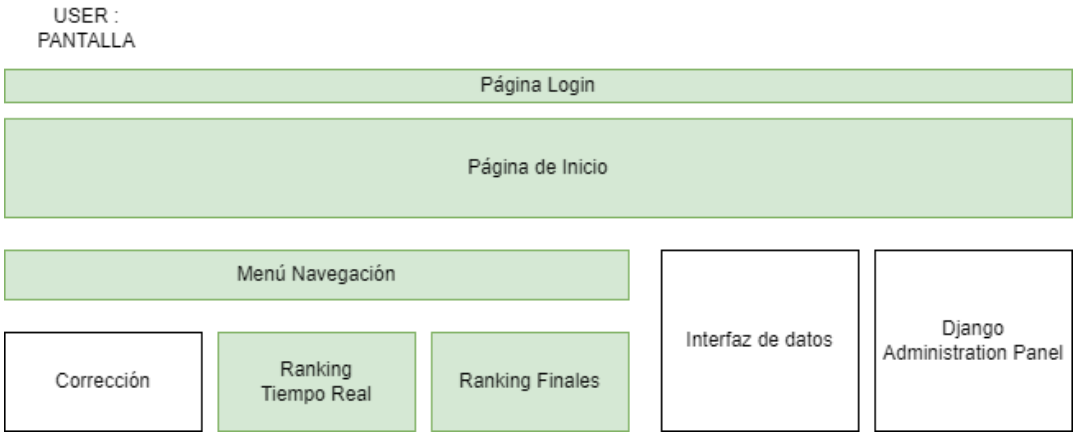


Ilustración 40: Flujo navegación User: Pantalla

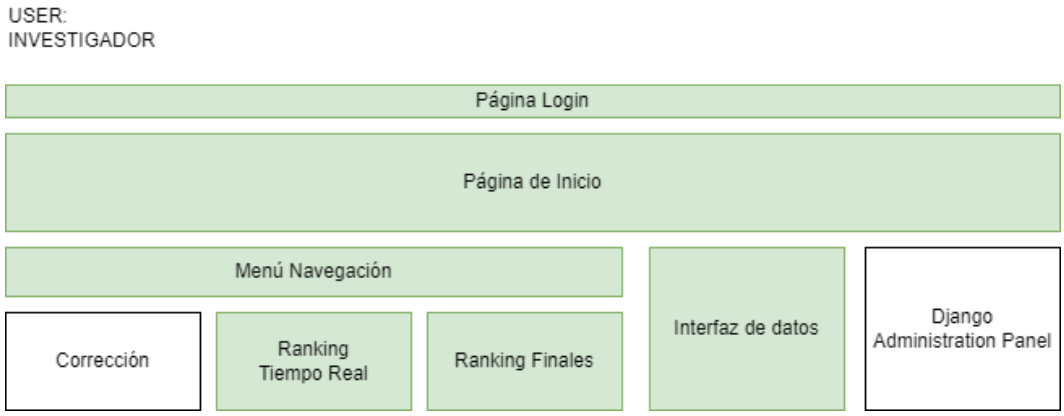


Ilustración 41: Flujo de navegación User: Investigador

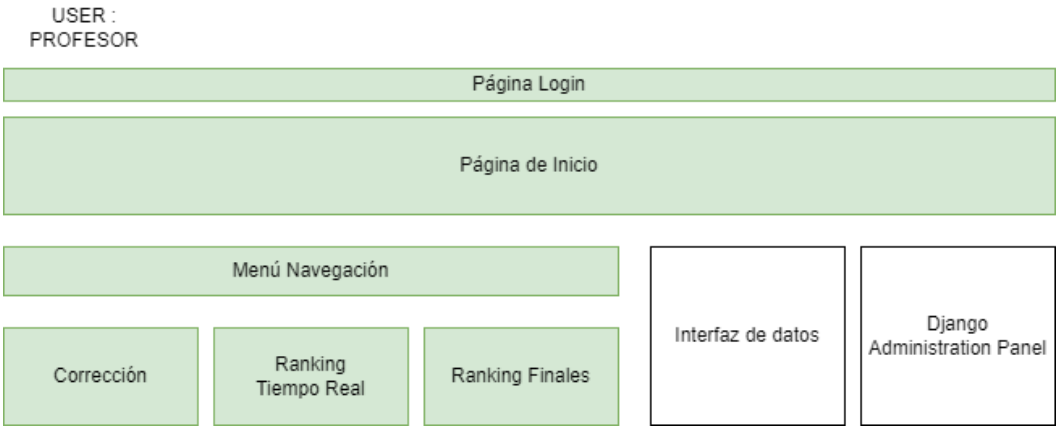


Ilustración 42: Flujo navegación User: Profesor

Ilustración 43: Flujo navegación User: Admin

Las diferentes vistas de la plataforma se muestran a continuación. Las interfaces de ranking y de descarga de datos cuentan con sus propias secciones dentro del documento.

5.1.7.1 Corrección

Desde la interfaz de corrección, los profesores pueden descargar y corregir las entregas entregadas por los alumnos.

Cada entrega está representada por una carta, en la que se muestra la siguiente información y se pueden realizar estas acciones.

- Número de mesa y nombre de equipo
- Número de problema e intento (si es el primer intento, es 0)
- Nota (si ha sido corregido)
- Botón de descarga de archivo. Incluye la extensión del archivo a descargar para que el profesor elija un dispositivo que soporte ese tipo de archivos, siguiendo el RF16. (Hay tantos botones de descarga como archivos entregados por el alumno).
- Botón de calificar: Consiste en un menú desplegable, que permite poner nota del 1 al 10, permitiendo al profesor calificar con un 15 únicamente si el problema es candidato a premio de velocidad.

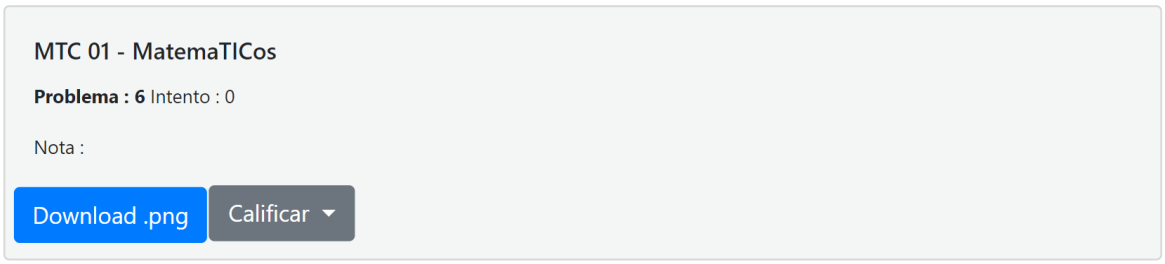


Ilustración 44: Entrega

Según se establece en el RF17: *Debe existir una forma que indique a los profesores que problemas han sido corregidos, cuáles han sido descargados por otro compañero y se encuentran en corrección y cuales cuentan con premio de velocidad sin que tengan que ir al ranking para ello.* Para solventarlo, se ha implementado el siguiente código de colores para la comunicación entre docentes, acordado con los profesores de las dos asignaturas

- Carta dorada. Entrega candidata a premio de velocidad (prioridad de corrección) o corregida y con bonus de velocidad.
- Carta blanca. Entrega sin corregir
- Carta verde. Entrega corregida
- Carta azul. Entrega en corrección por alguno de los profesores. Las entregas se muestran en azul cuando algún profesor descarga un archivo de esta, entendiendo que va a proceder a la corrección del mismo.

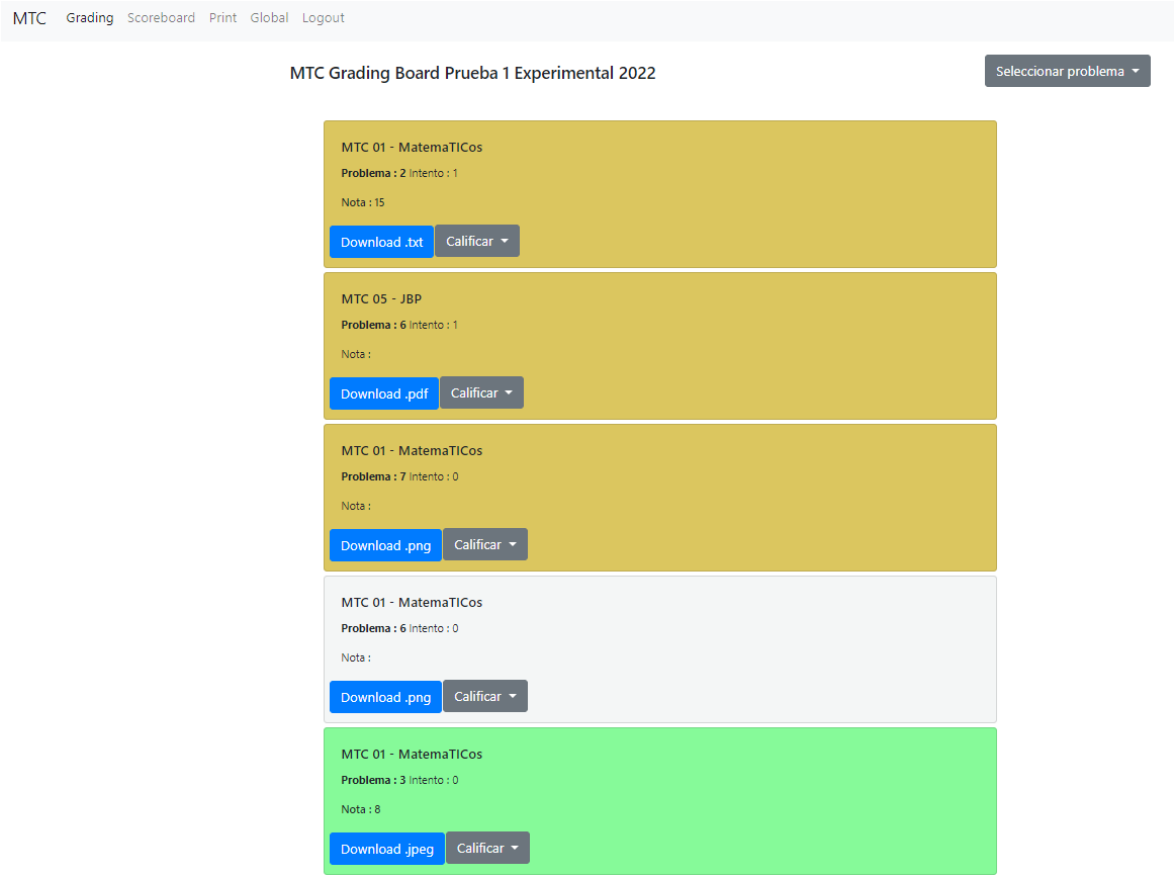


Ilustración 45: Herramienta Corrección

Tras el MTC1, y como parte de la segunda iteración del ciclo incremental e iterativo, tras recibir el feedback de los profesores, se añadió un filtro situado en la parte superior derecha de la ventana y permite visualizar las entregas únicamente del problema seleccionado.

Los profesores comunicaron que normalmente cada uno centra sus esfuerzos en la corrección de un problema determinado durante un intervalo de tiempo de la prueba. Se planteó la opción del filtro para facilitar el uso de la herramienta de corrección, de tal manera que los profesores no tuviesen que bajar por la ventana buscando manualmente las entregas de un problema concreto.

El RF14 planteaba lo siguiente: *La herramienta de corrección debe tener una interfaz que facilite y agilice la corrección, que permita pasar de la descarga de un problema a otro en menos de 7 "clicks".*

Con la nueva herramienta de corrección el clickcount entre la descarga de dos problemas se ha reducido de 7 a 3, obteniendo una rapidez de corrección que supera con creces a la que se tenía en el concurso antes de la implementación de la plataforma MTC.

5.1.7.2 Página Login

La pantalla de login es la siguiente, priorizando un diseño simple y efectivo.

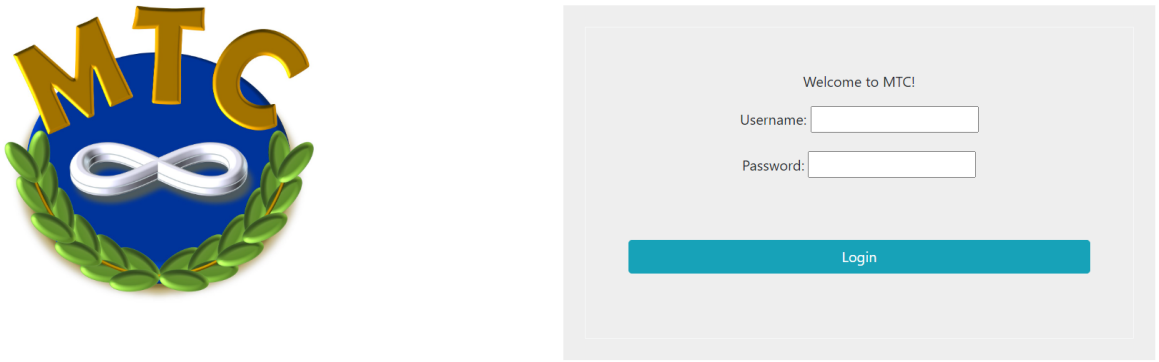


Ilustración 46: Página Login

Una vez introducidos un usuario y contraseña válidos, la aplicación redirecciona al usuario autenticado a la página de inicio, desde la que puede elegir la prueba a la que desea acceder.

5.1.7.3 Página de inicio

Desde la página de inicio, el usuario selecciona en el menú desplegable la prueba a la que quiere acceder.

Desde el botón de "Go to Data Export", los administradores pueden acceder a la interfaz de datos. El resto de los usuarios no ven este botón que da acceso a la descarga de los datos del concurso.

Con el botón de logout, el usuario vuelve a la pantalla de login, permitiéndole cambiar de tipo de usuario o de perfil y desconectando su sesión de la aplicación.

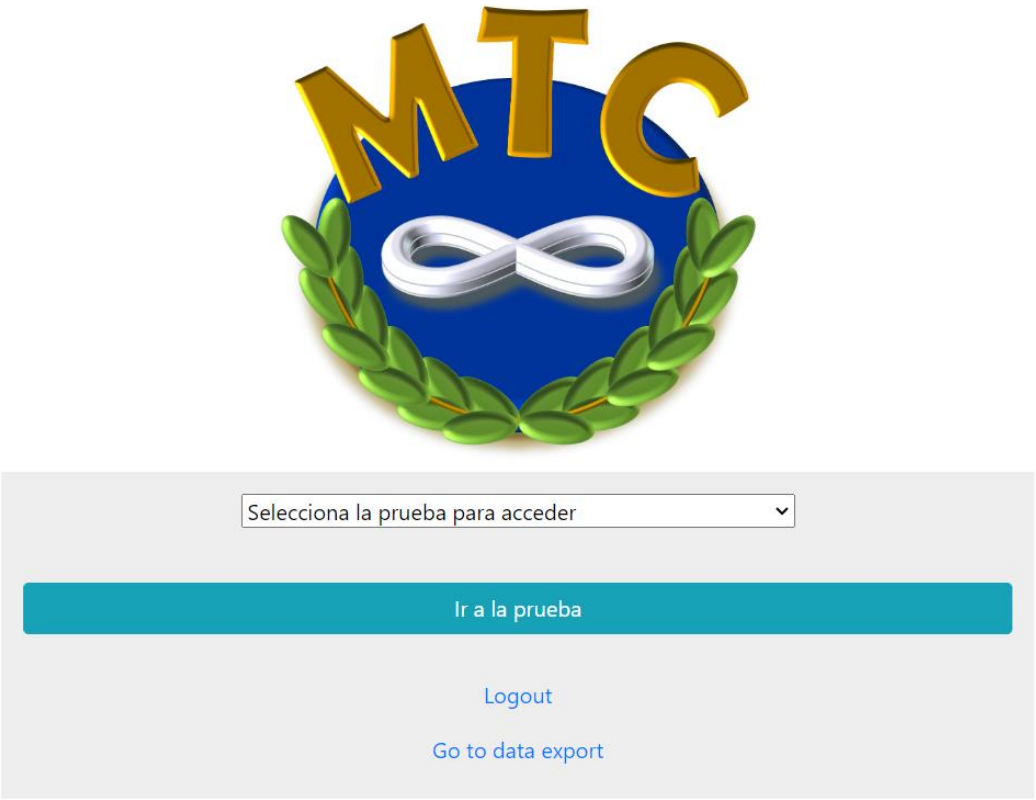


Ilustración 47: Página de inicio

5.1.8 DJANGO ADMIN SITE

El Django Admin Site es una interfaz que se incluye automáticamente al crear una aplicación con el framework Django. Es una de las herramientas más poderosas que ofrece Django [8], proporcionando funciones de gestión de usuarios y grupos, de creación y modificación de

elementos de la base de datos, y gestión de permisos. Siendo la gestión de permisos uno de los principales RF y, por tanto, uno de los motivos de la elección de Django como framework para el desarrollo de la aplicación, el uso del Django Admin Site ha facilitado la gestión de usuarios, grupos y permisos, realizando la mayoría de esta a través del Sitio de Administrador.

La interfaz de Django Admin Site viene predefinida al crear la aplicación y su menú principal es el siguiente, desde el cual se puede acceder a la gestión de usuarios y grupos y a las diferentes tablas de la base de datos.

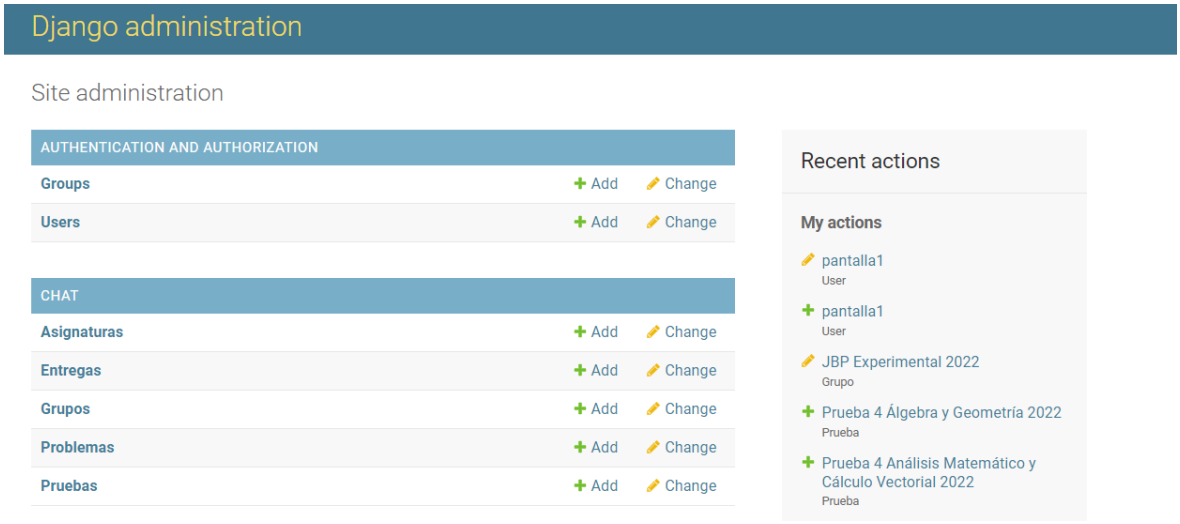


Ilustración 48: Django Administration Site

Entrando en la gestión de usuarios, desde el perfil de usuario, es posible añadirlo a un grupo o grupos en concreto. Estos grupos tienen sus permisos asociados, que se tratan en la parte de Usuarios y autenticación, que hereda el usuario. Además, la consola de administrador permite adjudicar y revocar permisos concretos a un usuario, sobrescribiendo los permisos que hereda de su grupo.

La gestión de usuarios a través del panel de administrador facilita, en un futuro, poder añadir otro tipo de usuarios que requieran permisos diferentes a los de los grupos actuales.

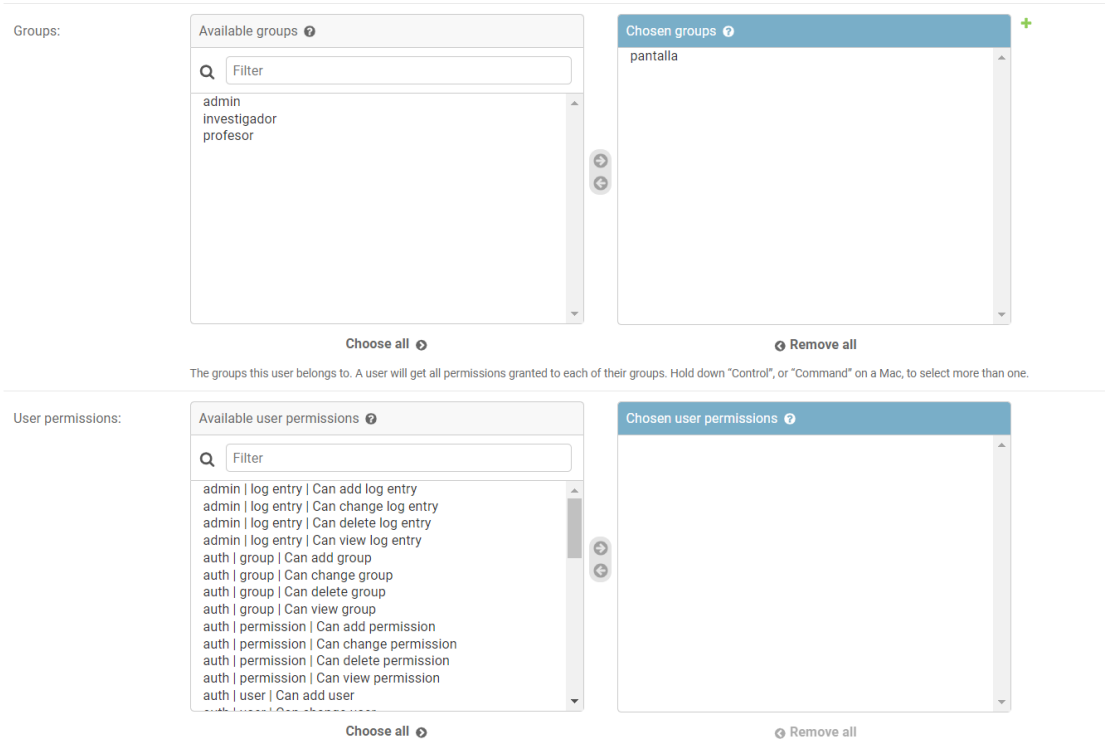


Ilustración 49: Gestión de permisos y grupos en un usuario

INTERFAZ DE DATOS

La Interfaz de datos se ha diseñado acorde con los RF32-35. Se creó en la 4ª Iteración de la aplicación, permitiendo la exportación de los datos en un formato compatible con los sistemas de post-procesado prexistentes.

Para ello se definieron formatos de datos específicos que aportan información sobre diferentes partes del concurso.

La interfaz es accesible únicamente para los administradores, desde la página de inicio. No se ha añadido al menú de navegación, al considerar que no es necesario para el funcionamiento de concurso, si no que complementa su actividad.

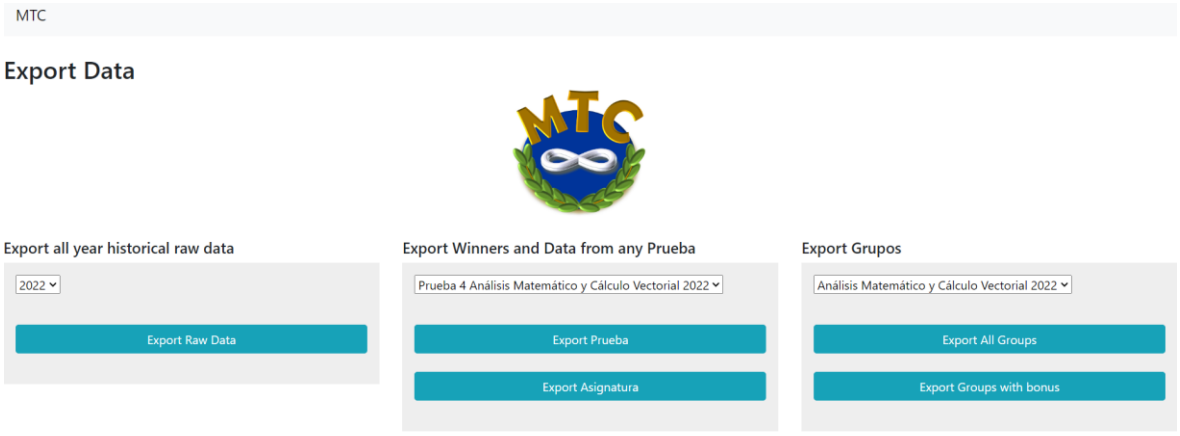


Ilustración 50: Interfaz de datos

La exportación de datos se ha realizado de manera modular, pudiendo descargar los siguientes datos del concurso en formato CSV:

- Los datos en crudo de todas las entregas realizadas, con la siguiente información: [año, asignatura, problema, id, bonus, grupo, número de grupo, nota, tiempo, corregida, descargada, antigua].
- El ranking final de cada prueba
- El ranking global de cada asignatura
- La información de todos los grupos participantes en una asignatura
- Todos los grupos que han recibido algún premio de velocidad en cualquiera de los problemas de una asignatura.

Los datos exportados en CSV se enlazan con los módulos de post-procesado preexistentes, permitiendo la generación automática de diplomas, menciones de honor, presentaciones y otros materiales complementarios a la actividad del concurso.

Las posibilidades que ofrece la salida en crudo de los datos para su análisis son muy amplias. La traza temporal permite hacer un análisis exhaustivo y cuantitativo de lo que ha ocurrido durante el desarrollo de cada prueba, sacando conclusiones importantes en el ámbito docente, pudiendo responder a preguntas cómo:

- ¿En qué franjas de tiempo ha habido más picos de trabajo?
- ¿Existe un efecto llamada en los problemas de tal manera que cuando el primer grupo entrega, el resto pronto sigue sus pasos?
- ¿Cuántos puntos por minuto se ha generado por prueba?

A partir del dato crudo, las posibilidades son infinitas. El hecho de la exportación modular de los datos y la normalización de las tablas de la DB permite añadir en un futuro información que resultar útil para analizar la influencia del concurso en el desarrollo de las capacidades matemáticas del concurso.

6. IMPLEMENTACIÓN, DESPLIEGUE Y PRUEBAS DESARROLLADAS

En este capítulo se detallarán las pruebas de software y hardware realizadas, así como la implementación de la plataforma.

La solución desarrollada no se ha quedado en un diseño teórico o prototipo, sino que se ha llevado a producción desplegándola en un servidor de la Universidad.

Tanto en la fase de diseño como en la de desarrollo, y en la de producción se han ido realizando pruebas para asegurar la calidad del software desarrollado y garantizar que cumple con los requisitos y expectativas del cliente, de acuerdo con el ciclo de vida iterativo implementado como plan de trabajo.

ENTORNO DE TRABAJO

Para el desarrollo del software y las pruebas realizadas, se ha creado un entorno de trabajo que cuenta con un servidor en local y un Moodle de pruebas. Este entorno de trabajo se creó en la primera iteración del ciclo de vida de la plataforma y se ha mantenido durante las iteraciones siguientes.

6.1.1 ENTORNO VIRTUAL Y SERVIDOR LOCAL

En primer lugar, realicé una instalación en local de Django, que crea automáticamente todos los archivos necesarios para el desarrollo de un proyecto. Una vez instalado el framework y creado el proyecto base, creé un entorno virtual de Python con todos los ejecutables necesarios para desplegar el proyecto en un servidor local. El servidor local permite realizar diferentes pruebas de validación o integración.

Para la instalación de Django y sus componentes se ha seguido la guía *Django Introduction and Installation* [9][13]. Para la implementación del entorno virtual se ha utilizado la guía *Python Virtual Environment* [9][12], con el módulo virtualenv.

Una vez creado el entorno virtual, fue posible desplegar el Django crudo + SQLite3 encima.

6.1.2 MOODLE DE PRUEBAS

Como se ha mencionado anteriormente en este documento, para llevar a cabo pruebas de conexión con Moodle en local y en un entorno acotado en producción, se creó una asignatura de Moodle con el Bot como profesor con permisos limitados y que aislaba las pruebas de producción de la actividad de las asignaturas de Álgebra y Geometría y Análisis Matemático y Cálculo Vectorial que forman parte del MTC.

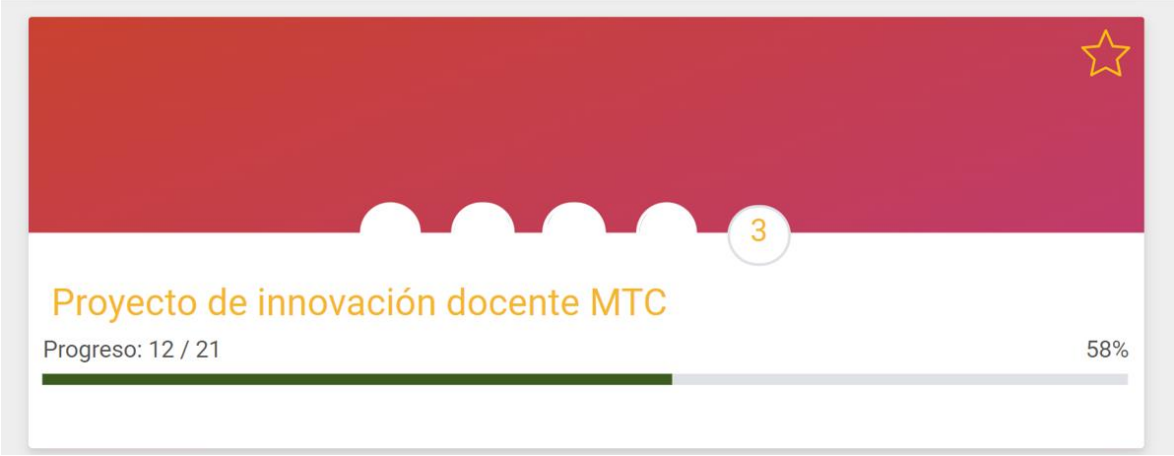


Ilustración 51: Moodle de pruebas

En este Moodle de pruebas se añadieron los profesores participantes en el concurso para poder realizar pruebas de caja negra con ellos. Las imágenes de perfil los profesores en la Ilustración 51: Moodle de pruebas se han censurado por privacidad.

6.1.3 POSTMAN

Se ha utilizado la aplicación de Postman para realizar pruebas sobre la API de Moodle, analizar sus funciones y salidas. La documentación sobre la API de Moodle [1] no es muy

clara, por lo que ha sido necesario analizar las salidas de diferentes funciones con Postman para determinar el comportamiento de Moodle y elegir cómo queríamos traducir/complementar ese comportamiento en la aplicación MTC.

El análisis de las trazas de las llamadas a la API de Moodle ha sido clave a la hora de determinar las funciones de Moodle a las que se le debía dar acceso al bot creado por STIC.

Además, Postman ha sido muy útil a la hora de analizar la información que almacena Moodle de cada entrega realizada y determinar la información que queríamos guardar sobre ellas en la base de datos.

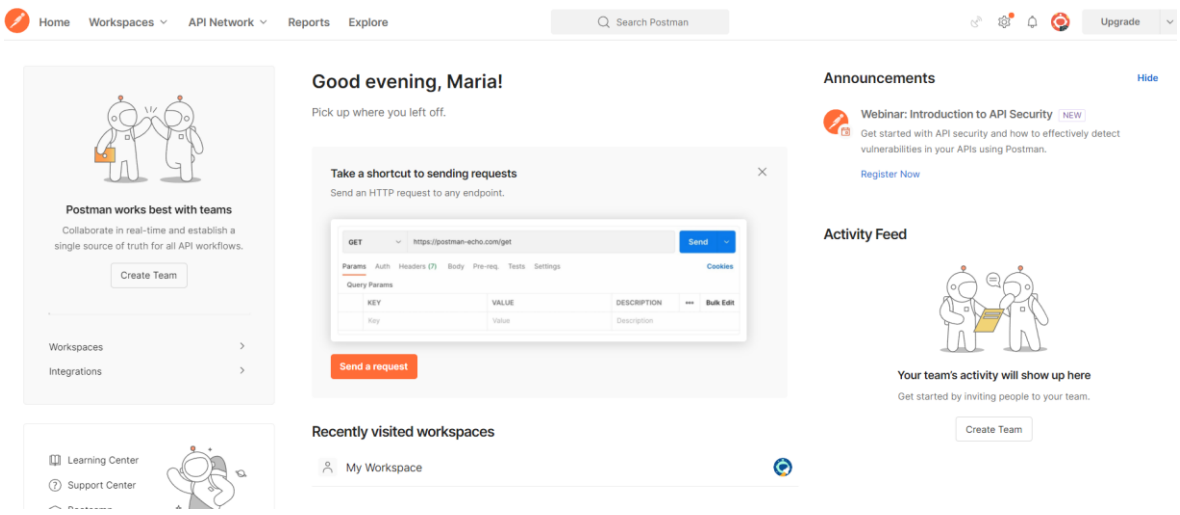


Ilustración 52: Interfaz de Postman

6.1.4 BOT

Por supuesto, la creación del Bot por parte de STIC y la asignación de permisos especiales para poder realizar la conexión con Moodle a través del crawler, ha sido clave para la realización de pruebas en un entorno seguro que no comprometa información de Moodle que no forme parte del MTC.

ENTORNO DE PRODUCCIÓN

El entorno de producción utilizado para el despliegue de la infraestructura y para las diferentes pruebas de validación, de aceptación y de sistema está formado por un servidor de la Universidad y las asignaturas de Moodle de Análisis Matemático y Cálculo Vectorial y Álgebra y Geometría del grado de iMAT.

6.1.5 VPN

El acceso al servidor de la Universidad se ha realizado a través de una VPN. Una Virtual Private Network permite que un dispositivo pueda conectarse a una red corporativa y trabajar en remoto como si se encontrase trabajando en un equipo en la red física.

Para la configuración de la VPN se ha utilizado Global Protect y la guía Acceso VPN creada por el servicio STIC de la Universidad.

Ilustración 53. Global Protect

Una vez protegida la conexión, se ha accedido al servidor con el protocolo SSH con el usuario creado específicamente por el servicio STIC para el despliegue de la solución MTC.

Sobre el servidor Apache de la Universidad se ha instalado el Django + SQLite3 crudo.

En la primera fase, se desplegó la totalidad de la solución, mientras que para el resto de las iteraciones se subieron al servidor de producción los archivos modificados durante la fase de desarrollo de cada una de las iteraciones.

6.1.6 MOODLE

En producción, se utilizaron las asignaturas de Moodle de Análisis Matemático y Cálculo Vectorial y Álgebra y Geometría del grado de iMAT, a las que el Bot está asociado como "non-editing teacher" y "wsrole_icai". Esto es, tiene permisos de lectura y acceso específico a las funciones de la API de Moodle necesarias, pero no puede realizar ninguna modificación sobre el contenido de las asignaturas.

Además de las asignaturas oficiales del concurso, en producción se siguió utilizando el Moodle experimental para las diferentes pruebas beta en entorno acotado de producción.

IMPLEMENTACIÓN

La implementación y despliegue se ha hecho siguiendo el ciclo iterativo del proyecto explicado en el Capítulo 3: Plan de trabajo.

Durante la primera iteración se trabajó en local para el desarrollo del software. Antes de la primera prueba del concurso, se desplegó en el servidor de la Universidad un prototipo funcional (MVP).

Teniendo ya un prototipo en producción, en los ciclos siguientes se siguió trabajando en local, introduciendo mejoras y funcionalidades adicionales, solventando bugs y estabilizando el código.

En cada ciclo, se han realizado diferentes pruebas que se comentarán a continuación y que garantizaban que la plataforma en desarrollo estaba lista para pasar a producción antes de la siguiente prueba.

Una vez realizadas las pruebas correspondientes en local, se liberaba el nuevo prototipo y se realizaban pruebas en un entorno acotado, de tal manera que siempre teníamos un prototipo funcional en producción y mejorado respecto al anterior.

6.1.7 DJANGO Y CÓDIGO DE LA SOLUCIÓN IMPLEMENTADA

El código final de la solución desarrollada se puede consultar en el ANEXO II. Para dar contexto de cada uno de los módulos que forman parte del código desarrollado, en este apartado se incluye una descripción general del funcionamiento de Django como framework de desarrollo.

Django se basa en el patrón de arquitectura MVC (Modelo-Vista-Controlador). Django realiza una separación clara de las capas de datos, presentación y lógica de control. Sus componentes básicos son los siguientes:

- Modelo. El modelo es la capa de datos de Django. En él se ha definido la estructura de la base de datos en clases de Python que luego se traducen automáticamente en tablas de la base de datos SQLite3.
- Vista. Es la capa lógica, que procesa las solicitudes entrantes y devuelve las respuestas correspondientes. Las vistas están definidas como diferentes funciones de Python que procesan la lógica de cada uno de los submódulos explicados en el Capítulo 5.
- Template. Es la capa de presentación de la aplicación. Los diferentes templates representan cada una de las vistas de la aplicación MTC y son archivos HTML con marcadores que permiten incluir datos dinámicos.

En la imagen siguiente se detalla el funcionamiento del modelo MVC por el que se rige Django.

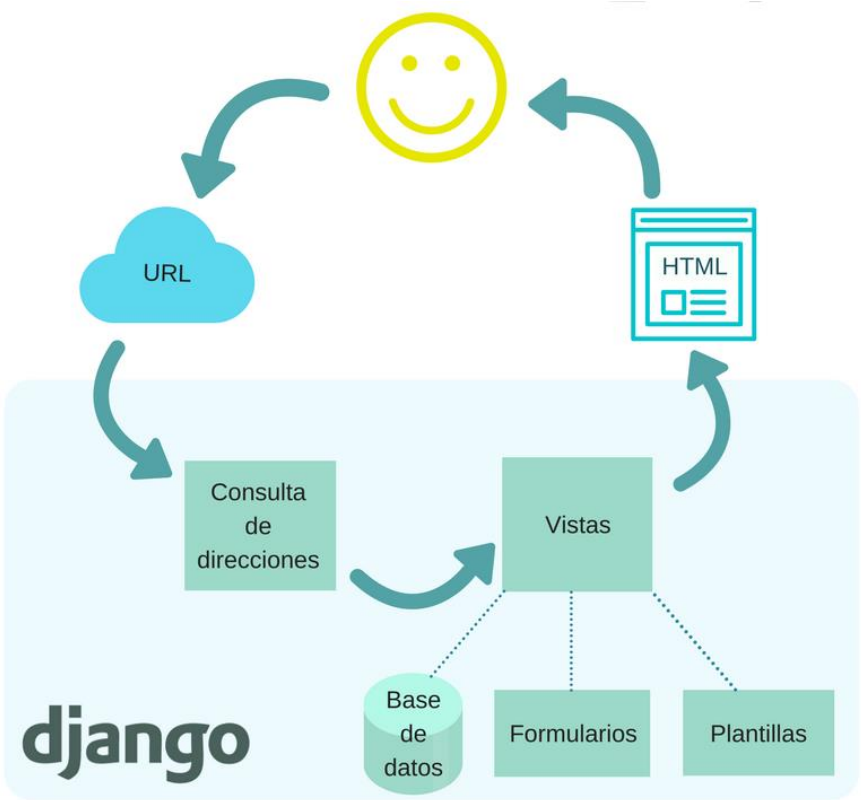


Ilustración 54: Funcionamiento Django [9][15]

Además, al crear un proyecto Django, este incluye una serie de archivos adicionales que proporcionan funcionalidades como enrutamiento URL, autenticación de usuarios o seguridad.

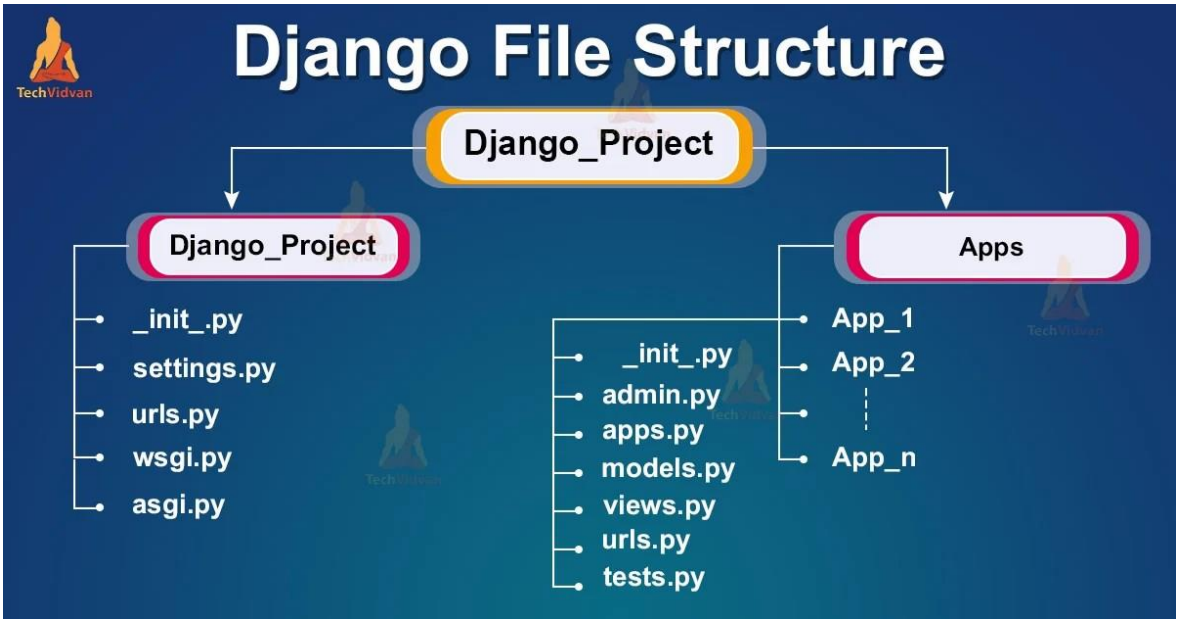


Ilustración 55: Estructura de un proyecto Django [9][14]

PRUEBAS

A lo largo de todas las fases del ciclo de vida incremental e iterativo del proyecto, se han realizado pruebas de diferente índole para identificar y depurar defectos en el software y garantizar el funcionamiento y seguridad de cada uno de los prototipos lanzados.

Se han realizado diferentes tipos de pruebas, cada uno enfocado en un aspecto concreto del software.

6.1.8 PRUEBAS HARDWARE

Como paso previo al desarrollo de la solución, en la fase de análisis de requisitos y de diferentes arquitecturas, se realizaron pruebas de hardware para estudiar la viabilidad de cada una de las infraestructuras planteadas.

Se llevaron a cabo pruebas de distancia de emisión en el Comillas Conecta Lab y de conectividad con el hotspot, descartándose finalmente las opciones que requerían conectividad peer-to-peer por el corto alcance que reflejó el resultado de ambas pruebas.

6.1.9 PRUEBAS UNITARIAS

Se han realizado pruebas unitarias de cada uno de los módulos lógicos expuestos en el Capítulo 5. Diseño de la Solución Software. Estos son:

- Crawler
- Marcapasos
- Base de datos
- Corrección
- Descarga
- Visualización

Para cada uno de ellos se han realizado pruebas de caja blanca, evaluando la estructura interna del software y verificando que el comportamiento del programa se ajusta a lo planteado en el diseño. Se han llevado a cabo caminos de pruebas básicos y pruebas de cobertura de decisión, diseñadas para probar un caso de uso concreto del diagrama de decisión.

Estas pruebas se han realizado en local en cada una de las iteraciones para verificar todo el abanico de casos que pueden darse en corrección, descarga, los distintos módulos de visualización y ejecución del crawler.

Además de estas pruebas unitarias de los módulos propios de la plataforma, se han realizado pruebas sobre la API de Moodle, estudiando su comportamiento a través de las trazas de salida de diferentes peticiones HTTP realizadas a través de Postman.

6.1.10 PRUEBAS DE INTEGRACIÓN

Se han realizado pruebas de integración en local para verificar que los diferentes módulos se integran entre sí funcionando como un todo.

Por ello se han realizado pruebas de caja negra o de sistema cerrado en el servidor local, subiendo tareas a Moodle, descargando y corrigiendo, para comprobar que el flujo de datos era correcto en todos los módulos de la aplicación y que el resultado de cada operación era el esperado en todos los módulos ante el estímulo dado.

Además de pruebas de integración software, se realizaron pruebas de integración hardware, comprobando que los módulos eran compatibles con los requerimientos hardware, esto es, que las pantallas de visualización cumplían los requerimientos de tamaño y formato de las pantallas del Comillas Conecta Lab, y que la herramienta de corrección era cómodamente accesible y funcional desde los dispositivos móviles (tablets) de los profesores.

6.1.11 PRUEBAS DE VALIDACIÓN

Se han llevado a cabo tres tipos de pruebas de validación diferentes:

- Alfa, en el entorno de desarrollo en local.
- Beta, en el entorno de producción, dentro de las cuales se pueden diferenciar dos tipos: beta cerrada, en un entorno acotado utilizando el Moodle de prueba, y beta abierta, en el entorno de producción con los profesores.

Estas pruebas evalúan el comportamiento completo del sistema en diferentes situaciones y se han llevado a cabo en todas las iteraciones para verificar el funcionamiento correcto del sistema lanzado en cada una.

En la 1ª beta cerrada que se realizó en producción, pero en un entorno acotado, en el que el bot únicamente accedía al Moodle de pruebas, se descubrió la imposibilidad de utilizar websockets sobre el servidor de la Universidad y se planteó la comunicación híbrida que permitía la comunicación por websockets en un entorno local y la actualización periódica por HTTPS en producción, tal como se explica en la sección 4.1.1 Comunicación en tiempo real por websockets de este mismo documento.

Estas betas cerradas se realizaban utilizando el Moodle experimental, donde los profesores del concurso tenían tanto perfil de alumno como de profesor, permitiendo la subida de entregas por parte de los mismos.

6.1.12 PRUEBAS DE USABILIDAD

Se llevaron a cabo pruebas de usabilidad con los profesores del concurso en el entorno de beta cerrada para evaluar la facilidad de uso y la experiencia del usuario. Se realizaron simulaciones completas de subida de entregas, descarga y corrección en el Comillas Conecta Lab.

Además, en el ciclo incremental e iterativo de la plataforma, se utilizaron los diferentes MTC como pruebas de usabilidad, tras las cuales los profesores emitían feedback sobre su experiencia utilizando la aplicación, que repercutía en nuevas mejoras de interfaz en las iteraciones posteriores.

Finalmente, en la última iteración se han recogido las siguientes métricas de usabilidad, basadas en el "click count" de las principales acciones del sistema en comparación con las de Moodle.

Acción	Clickcount Moodle	Clickcount plataforma actual
Descarga archivos	3	1
Cambio de problema y descarga archivo	7	3
Corrección	5	2

Tabla 4: Comparativa Métricas de Usabilidad

6.1.13 PRUEBAS DE SISTEMA

Ante la dificultad de realizar pruebas de volumen, de sobrecarga, rendimiento y fiabilidad en el entorno acotado del Moodle de pruebas, se usó el primer MTC con el prototipo funcional para evaluar estos parámetros.

Las pruebas de volumen y sobrecarga fueron positivas, por lo que se centraron los esfuerzos en mejorar la fiabilidad y estabilidad del sistema.

6.1.14 PRUEBAS DE ACEPTACIÓN

Las pruebas de aceptación verifican que el software cumple con los requisitos planteados por el cliente, en este caso que de soporte a la actividad del Maths Team Contest.

Por lo tanto, cada una de las iteraciones del ciclo ágil iterativo ha supuesto una prueba de aceptación, en el que las pruebas MTC realizadas con alumnos validaban el funcionamiento óptimo y acorde a los requisitos planteados en el inicio de este documento.

El evento que cierra cada una de las iteraciones es una prueba real del MTC, comenzando la iteración siguiente con el análisis de los resultados de las pruebas de aceptación.

7. CONCLUSIONES Y TRABAJO FUTURO

CONCLUSIONES

Del proyecto realizado se pueden extraer las siguientes conclusiones:

- Se ha diseñado una arquitectura que solventa las limitaciones que planteaba el sistema antiguo de gestión del Maths Team Contest.
- Se ha logrado implementar con éxito la plataforma que resuelve el problema planteado.
- Se ha utilizado el sistema para dar soporte a todas las pruebas del MTC en el curso académico 2022-23, evaluando su funcionamiento en un entorno real con alumnos y profesores.
- Se ha podido dar servicio a dos asignaturas de grado (Álgebra y Geometría y Análisis Matemático y Cálculo Vectorial de iMAT) en la evaluación de las capacidades matemáticas de los alumnos.
- La plataforma ha sido empleada por casi 100 alumnos y 5 profesores durante las diferentes pruebas, consiguiendo un volumen de de 1680 problemas entregados.
- Se ha conseguido la sincronización de datos con Moodle, manteniendo la seguridad en ambas plataformas.
- Se ha implementado una arquitectura distribuida y escalable que permite a profesores en remoto participar en la actividad del concurso.
- Los resultados del proyecto se han presentado en la Jornada de Buenas Prácticas en Innovación Docente 2023 de la Universidad Pontificia de Comillas.

TRABAJO FUTURO

En el curso 22/23, la actividad del MTC seguirá en el nuevo proyecto 2223-14, en colaboración con la Unidad de Investigación en Innovación Docente del CIHS. Esto supondrá la incorporación al equipo de dos investigadoras que analizarán el rendimiento del concurso MTC en innovación docente y en la mejora de las competencias matemáticas y de trabajo en grupo de los alumnos.

Para el nuevo proyecto se seguirá utilizando la plataforma desarrollada en este Trabajo de Fin de Grado para toda la gestión del concurso MTC y como centro de recogida de datos para la nueva vertiente del proyecto que pretende investigar los procesos de aprendizaje que se dan en el concurso.

Estos datos se integrarán en plataformas de análisis con otros tipos de muestras que las investigadoras tomarán durante el concurso, para poder realizar un análisis exhaustivo que ponga en valor la influencia del MTC en la mejora de las competencias de los alumnos.

Con vistas a la continuación del proyecto, se planteará la recogida de datos adicionales que puedan ser útiles para sacar percepciones y conclusiones de los mismos. Para ello, se volverá a realizar un análisis de requisitos evaluando los datos disponibles y los que sería relevante añadir, para hacer los ajustes necesarios sobre la base de datos. Como idea futura, se ha planteado la inclusión de los tiempos en los que se realiza la descarga y corrección de las entregas o el usuario que realiza estas actividades.

La incorporación de nuevas personas al equipo supondrá la ampliación de las funcionalidades del usuario de tipo investigador.

La arquitectura se ha dejado modularizada para que sea ampliable en el futuro, pudiendo añadir funcionalidades y avanzar según las necesidades que tenga el concurso.

Una de las posibles mejoras de la plataforma que se plantean es la integración del cronómetro que se utiliza para marcar el tiempo de las pruebas, de tal manera que sea gestionado de forma centralizada por el administrador en la propia interfaz.

Como nota final, esta arquitectura podría servir como base para crear una suite de corrección colaborativa configurable que pudiera utilizarse en otros tipos de proyectos de colaboración docente.

8. BIBLIOGRAFÍA

[1] *Acerca de Moodle - Moodle Docs*. (s.f.). Obtenido de https://docs.moodle.org/all/es/Acerca_de_Moodle

[2] Departamento de Matemática Aplicada – Universidad Pontificia de Comillas (s.f.) *Reglamento del Maths Team Contest*.

[3] Garzas, J. (4 de octubre de 2012). *El ciclo de vida iterativo e incremental y el riesgo de olvidarse del iterativo y quedarse solo con el incremental*. Obtenido de https:/www.javiergarzaas.com/2012/10/iterativo-e-incremental.html

[4] *Oficina de Apoyo a la Innovación Docente – Universidad Pontificia de Comillas.* (s.f.). Obtenido de https://www.comillas.edu/oaid

[5] Roger, S.P. (2005) *Ingeniería del Software: Un enfoque práctico.*

[6] *Wikiwand - WebSocket*. (s. f.). Wikiwand. https://www.wikiwand.com/en/WebSocket

[7] *Comillas - Edición 103*. (s. f.). https://revista.comillas.edu/.edicion.103/#page=12

[8] *Django*. (n.d.). Django Project. https://docs.djangoproject.com/en/4.2/ref/contrib/admin/

[9] Gamez, M. J. (2022, May 24). *Objetivos y metas de desarrollo sostenible - Desarrollo Sostenible*. Desarrollo Sostenible. https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

[10] GeeksforGeeks. (2022). Realtime chat app using Django. *GeeksforGeeks*. https://www.geeksforgeeks.org/realtime-chat-app-using-django/

[11] Websockets. (n.d.). Websockets documentation. https://websockets.readthedocs.io/en/stable/

[12] GeeksforGeeks. (2023). Python Virtual Environment Introduction. GeeksforGeeks. https://www.geeksforgeeks.org/python-virtual-environment/

[13] GeeksforGeeks. (2022b). Django Introduction and Installation. GeeksforGeeks. https://www.geeksforgeeks.org/django-introduction-and-installation/

[14] Team, T. (2021). Django Project Structure and File Structure. TechVidvan. https://techvidvan.com/tutorials/django-project-structure-layout/

[15] García, M. (2017, October 6). Django: ¿por qué usar Django? https://codingornot.com/django-por-que-usar-django

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS

"El 25 de septiembre de 2015, los líderes mundiales adoptaron un conjunto de objetivos globales para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todos como parte de una nueva agenda desarrollo sostenible. Cada objetivo tiene metas específicas que deben alcanzarse en los próximos 15 años." [9]

Los 17 objetivos, que establecen una guía para el desarrollo sostenible a nivel global, son los siguientes.

Ilustración 56: Objetivos de Desarrollo Sostenible [9]

A continuación, se muestran los objetivos con los que se alinea el proyecto.

ODS 4. Educación de calidad

El ODS 4 establece como objetivo "*Garantizar una educación inclusiva, equitativa y de calidad y promover oportunidades de aprendizaje durante toda la vida para todos"[9].*

Por su naturaleza de innovación docente, el MTC tiene como objetivo mejorar la calidad de la educación. Dentro de este, una de las metas es "*De aquí a 2030, aumentar considerablemente el número de jóvenes y adultos que tienen las competencias necesarias, en particular técnicas y profesionales, para acceder al empleo, el trabajo decente y el emprendimiento". [9]* En este contexto, el proyecto desarrollado proporciona al MTC la plataforma y vía necesaria para fomentar la colaboración, organización, comunicación y otras habilidades profesionales en los alumnos participantes.

ODS 9. Industria, Innovación e Infraestructura.

El proyecto también se alinea con el 9º ODS, al desarrollar una infraestructura digital y escalable para la gestión distribuida y análisis de datos de un proyecto de innovación docente. Al diseñar una nueva plataforma, uno de los objetivos del proyecto es mejorar la eficiencia de gestión del MTC y de los datos de los académicos, contribuyendo así al 9º ODS.

Además, al utilizar métodos y tecnologías innovadoras, estableciendo una sincronización de datos con la plataforma de e-learning Moodle, se fomenta la innovación y el desarrollo tecnológico, una de las metas del ODS 9.

ODS 17. Alianza para lograr los objetivos.

La plataforma desarrollada para el ayuda a la colaboración entre docentes para llevar a cabo una actividad que promueve el ODS 4.

En resumen, el proyecto contribuye a los objetivos de desarrollo sostenible al promover una educación de calidad y la colaboración entre diferentes agentes mediante una infraestructura digital escalable que da soporte a la actividad del Maths Team Contest.

ANEXO II: CÓDIGO FUENTE

En esta sección se incluye el código de la solución desarrollada. Se incluyen únicamente los módulos principales con la lógica de control, dejando fuera archivos HTML estáticos, ficheros de configuración generados automáticamente por Django.

Estos módulos son:

- Models.py
- Utils.py
- Moodle.py
- Views.py
- Urls.py
- Routing.py
- Consumers.py
- Admin.py

No se ha incluido el código del frontend pues se prioriza en esta memoria la lógica principal de control y de la gestión del sistema. El resultado de las diferentes interfaces gráficas puede consultarse en las imágenes de la sección de Visualización y Diseño de la GUI y en los submódulos de Interfaz de datos y Corrección.

Los diferentes tokens y urls vulnerables han sido censurados.

MODELS.PY

Este archivo describe los objetos de Python que después se mapean a las tablas de la base de datos SQLite3.

```
from django.db import models

class Asignatura(models.Model):
    nombre = models.CharField(max_length=255)
```

```
    moodleid= models.IntegerField(primary_key=True)    #primarykey
    año=models.IntegerField()
    bonus =models.IntegerField(default=2)

    def __str__(self):
        return ("%s %s " % (str(self.nombre), self.año))

class Prueba(models.Model):
    moodleid=models.IntegerField(primary_key=True)  ##primarykey  no tiene clave
propia de moodle
    numero = models.IntegerField()
    fecha = models.DateField()
    asignatura = models.ForeignKey(Asignatura, on_delete=models.CASCADE)

    def __str__(self):
        return ( "Prueba %s %s" % (str(self.numero), self.asignatura))

class Grupo(models.Model):
    nombre = models.CharField(max_length = 255)
    numero = models.IntegerField(default=0)
    moodleid = models.IntegerField(primary_key=True)
    asignatura = models.ForeignKey(Asignatura, on_delete=models.CASCADE)
    def __str__(self):
        return ("%s %s" % (str(self.nombre), str(self.asignatura)))

class Problema(models.Model):
    moodleid = models.IntegerField(primary_key=True)
    numero = models.IntegerField()
    prueba = models.ForeignKey(Prueba, on_delete=models.CASCADE)

    def __str__(self):
        return "Problema  %s %s" % (str(self.numero), self.prueba)

class Entrega(models.Model):
    moodleid = models.IntegerField()
    problema = models.ForeignKey(Problema, on_delete=models.CASCADE)
    bonus = models.BooleanField()
    grupo = models.ForeignKey(Grupo, on_delete=models.CASCADE)
    nota = models.IntegerField(null=True)
    tiempo = models.DateTimeField()
    urls = models.JSONField(default={})
    attemptnumber = models.IntegerField()
    corregido = models.BooleanField(default=False)
    downloaded = models.BooleanField(default=False)
    antigua= models.BooleanField(default=False)

    #clave primaria doble
    class Meta:
        unique_together = ('moodleid', 'attemptnumber')

    def __str__(self):
        return "%s grupo %s" % (self.problema, self.grupo)
```

```
    def __repr__(self):
        return json.dumps(dict(self), ensure_ascii=False)
```

UTILS.PY

Este archivo recoge las funciones comunes a las diferentes rutinas lógicas. Estas incluyen el acceso a la base de datos y la rutina del marcapasos.

```
from .models import Problema, Entrega, Prueba, Grupo, Asignatura
import json
import requests
from datetime import datetime
import pytz

debug = False


def is_new(date, guardado):
    """
        Comprueba si una entrega ha sido modificada en Moodle.

        Argumentos:
        date      - (datetime) fecha de la entrega
        guardado  - (List<Entrega>) entregas anteriores para ese mismo problema y
grupo
        """
    try:
        attempt = guardado[0].attemptnumber +1
        if guardado[0].tiempo >= datetime.fromtimestamp(date,
tz=pytz.timezone('Europe/Madrid')):
            return False, attempt
        else:
            return True, attempt
    except Exception as e:
        if debug:
            print(e)
        return True, 0

def update_antiguas(guardado, withbonus):
    """
        Lleva toda la lógica de marcar una entrega como antigua, cambiando bonus
si es necesario.

        Argumentos:
        guardado  - (List<Entrega>) entregas anteriores para ese mismo problema y
grupo
```

```
        withbonus - (int) número de entregas que tienen bonus asignado para ese
problema

        Retorno:
        withbonus - (int) número actualizado de entregas que tienen bonus
asignado para ese problema
        """
    try:
        if debug:
            print("check")
        for g in guardado:
            if g.antigua == False:
                g.antigua = True
                g.urls= {}
                g.save()
                if check_bonus(g):
                    quitar_bonus(g)
        if bonus_disponible(g.problema):
            candidatas = buscar_candidatas(g.problema)
            if candidatas:
                asignar_bonus(candidatas[0])
                if debug:
                    print("bonus corrected")
        return withbonus
    except Exception as e:
        if debug:
            print("something went wrong updating antiguas")
            print(e)
        return withbonus

def get_problema(id,numero,prueba):
    """
        Devuelve el problema con el id pedido. Si no existe, lo crea.

        Argumentos:
        id        - (int) moodleid del problema
        numero    - (int) número de problema que corresponde con la tarea de
Moodle
        prueba    - (Prueba) prueba a la que pertenece el problema

        Retorno:
        problema  - (Prolema) problema pedido
        """
    try:
        problema = Problema.objects.get(moodleid=id)
        return problema
    except Problema.DoesNotExist:
        problema = Problema(moodleid = id, numero=numero, prueba=prueba)
        problema.save()
        return problema

def get_grupo(grupoid,asignaturaprueba):
    """
```

```
        Devuelve el grupo con el id pedido. Si no existe, lo crea.

        Argumentos:
        grupoid          - (int) moodleid del grupo
        asignaturaprueba - (Asignatura) asignatura a la que se asocia el grupo

        Retorno:
        grupo            - (Grupo) grupo pedido
        """
    try:
        grupo = Grupo.objects.get(moodleid=grupoid)
        return grupo
    except Grupo.DoesNotExist:
        if debug:
            print("grupo creado")
        try:
            if grupoid!=0:
                grupo = Grupo(moodleid = grupoid, nombre=grupoid,
asignatura=asignaturaprueba)
                grupo.save()
                if debug:
                    print("saved")
                return grupo
        except Exception as e:
            if debug:
                print(e)
            pass


def is_deleted(date, guardado):
        """
        Comprueba si una entrega ha sido borrada por un alumno.

        Argumentos:
        date      - (datetime) fecha de subida de la entrega
        guardado  - (List<Entrega>)entregas anteriores para ese mismo problema y
grupo

        Retorno:
        (Boolean) - True si la entrega ha sido eliminada,  False si no

        """
    try:
        if guardado[0].tiempo > datetime.fromtimestamp(date,
tz=pytz.timezone('Europe/Madrid')):

            return False
        else:
            return True
    except Exception as e:
        if debug:
            print("exception not guardado")
            print(e)
```

```
        return False

def check_bonus(e):
        """
        Comprueba si una entrega tiene premio de velocidad.

        Argumentos:
        e        - (Entrega) entrega

        Retorno:
        (Boolean) - True si la entrega tiene premio,  False si no
        """
    try:
        if e.bonus:
            return True
        else:
            return False
    except:
        return False


def buscar_candidatas(p):
        """
        Busca las entregas candidatas a premio de velocidad, que pueden ser
corregidas con un 10 o sin corregir.

        Argumentos:
        p        - (Problema) problema

        Retorno:
        buscar   - (List<Entrega>) entregas candidatas a premio
        """
    try:
        candidatas1 = Entrega.objects.filter(bonus=False, problema=p,
antigua=False, corregido=False)
        candidatas2 = Entrega.objects.filter(bonus=False, problema=p,
antigua=False, nota=10, corregido=True)
        candidatas= candidatas1 | candidatas2
        buscar = candidatas.order_by("tiempo")
        return buscar
    except:
        return None


## quita el bonus de una entrega que lo tenía
def quitar_bonus(e):
        """
        Quita el premio de velocidad a una entrega que lo tenía.

        Argumentos:
        e        - (Entrega) entrega
        """
    try:
```

```
        e.bonus = False
        if e.corregido and e.nota==15:
            e.nota=10
        e.save()
        return
    except:
        return

def asignar_bonus(e):
        """
        Asigna premio de velocidad a una entrega.

        Argumentos:
        e         - (Entrega) entrega
        """
    try:
        e.bonus = True
        if e.corregido and e.nota==10:
            e.nota = 15
        e.save()
        return
    except:
        return

def count_bonus(p, anterior=None):
        """
        Cuenta los bonus que hay asignados para el problema que se pasa como
parámetro.
        Si se pasa una entrega como segundo argumento, solo cuenta los bonus de
las entregas anteriores a esta.

        Argumentos:
        p          - (Problema) problema
        anterior   - (Entrega) entrega. Por defecto es None

        Retorno:
        withbonus - (int) número de premios de velocidad asignados
        """
    try:
        if anterior is not None:
            withbonus = Entrega.objects.filter(tiempo__lt=anterior.tiempo,
bonus=True, problema=p, antigua=False).count()
        else :
            withbonus = Entrega.objects.filter(bonus=True, problema=p,
antigua=False).count()
        return withbonus
    except:
        return 0


def buscar_withbonus(e):
        """
```

```
        Busca las entregas posteriores a la que se pasa como argumento que tengan
premio de velocidad.

        Argumentos:
        e           - (Entrega) entrega

        Retorno:
        posteriores - (List<Entrega>) entregas posteriores
        """
    try:
        posteriores= Entrega.objects.filter(tiempo__gt=e.tiempo,
problema=e.problema,bonus=True, antigua=False).order_by('-tiempo')
        return posteriores
    except:
        return None

def buscar_guardadas(e):
        """
        Busca las entregas anteriores para el id que se pasa como argumento.

        Argumentos:
        e         - (Entrega) entrega

        Retorno:
        guardado - (list<Entrega>) entregas anteriores
        """
    try:
        guardado = Entrega.objects.filter(moodleid=e["id"]).order_by("-tiempo")
#la mas reciente la primera
        return guardado
    except:
        return None

def bonus_disponible(p):
        """
       Comprueba si hay un bonus disponible para el problema que se pasa como
argumento.

        Argumentos:
        p          - (Problema) problema

        Retorno:
        (Boolean) - True si hay bonus disponible, False si no
        """
    try:
        bonus = p.prueba.asignatura.bonus
        if count_bonus(p) < bonus:
            return True
        else:
            return False
    except:
        if debug:
            print("bonus available error")
```

```
        return False

def start_moodle(prueba):
        """
        Empieza una instancia del crawler para la prueba que se pasa como
argumento si este no está iniciado.

        Argumentos:
        prueba    - (Prueba) prueba
        """
    try:
        global running_threads
        if not check_thread_status(prueba):
            my_thread = threading.Thread(target=Moodle().scrapping,
args=[prueba,])
            my_thread.start()
            running_threads[prueba]=my_thread
            if debug:
                print("hilo empezado")
    except Exception as e:
        if debug:
            print("error hilo")
            print(e)
        pass

def check_thread_status(prueba):
        """
        Verifica si existe el hilo que lee de moodle para una deterinada prueba y
si está activo.

        Argumentos:
        prueba     - (Prueba) prueba

        Retorno:
        (Boolean) - True si el hilo está activo, False si no
        """
    try:
        if debug:
            print("running check")
        global running_threads
        if prueba in running_threads and running_threads[prueba].is_alive():
            if debug:
                print("thread is already running")
            return True  # thread is running
        else:
            return False  # thread is not running
    except Exception as e:
        if debug:
            print(e)
```

MOODLE.PY

El archivo Moodle.py describe el hilo que realiza la rutina del crawler, estableciendo conexión con la API de Moodle.

```
#url de API de Moodle
url="https://***"

import threading
import time
from .models import Problema, Entrega, Prueba, Grupo, Asignatura
import json
import requests
from datetime import datetime
import pytz

from .utils import *

debug = False

class Moodle():

    def __init__(self):
        self.running = True
        pass

    def stop(self):
        self.running = False

    def scrapping(self, prueba):
            """
        Rutina principal que realiza el scrapping de moodle, importando problemas
y entregas.
        Únicamente retorna en caso de excepción

        Argumentos:
        self    - el propio hilo
        prueba  - (int) id de la prueba que se está llevando a cabo

        """

        try:
            while self.running:
                # Cojo la información de la prueba de la prueba y asignatura
asociadas con su moodleid
                prueba = Prueba.objects.get(moodleid=prueba)
                asignaturaprueba = prueba.asignatura
                bonus = prueba.asignatura.bonus
                mtc = "MTC " + str(prueba.numero)
```

```
                #TOKEN DE ACCESO A MOODLE
                token ="************************"
                try:
                    #establezco conexión con la API de moodle
                    #petición mod_assign_get_assignments: devuelve todos los
assignments(tareas) asociados a mi token
                    #esta petición se realiza una única vez cuando empieza el
hilo para recoger los ids de los assignments del MTC
                    query = {'wstoken':token,
                    'moodlewsrestformat':'json',
                    'wsfunction':'mod_assign_get_assignments'
                    }
                    response = requests.get('https://******', params=query)

                    # me quedo con la asignatura del MTC
                    asignatura = [course for course in response.json()["courses"]
if course["id"]==prueba.asignatura.moodleid][0]

                    # me quedo solo con las tareas que pertenezcan a mi prueba
actual del MTC
                    #los assignments del MTC tienen la siguiente nomenclatura:
                    # MTC x : Problema y    (x,y): (numero de prueba, numero de
problema)
                    ids = {}
                    for assignment in asignatura["assignments"]:
                        if mtc in assignment["name"]:
                            try:
                                ids[assignment["id"]]= int(assignment["name"][-
1])

                                if int(assignment["name"][-1])==0:
                                    ids[assignment["id"]]=10
                            except:
                                ids[assignment["id"]]=assignment["id"]
                    if debug:
                        print(ids)

                    while True:
                        #establecemos un tiempo de espera de un segundo entre fin
de procesado de una petición e inicio de la siguiente
                        time.sleep(1)

                        #realizamos una petición a la API de Moodle con la
función mod_assign_get_submissions
                        #se realiza una petición por cada id de tarea que
pertenece a la prueba
                        #esto equivale a una petición por cada Problema de
nuestra base de datos asociado a la prueba actual
                        for id, numero in ids.items():
                            query = {'wstoken':token,
                            'moodlewsrestformat':'json',
                            'wsfunction':'mod_assign_get_submissions',
                            'assignmentids[]':id
                            }
```

```
                            response = requests.get('https://******',
params=query)

                            #filtramos la información devuelta por la API de
moodle y ordenamos las entregas por el tiempo en el que han sido entregadas
                            entregas =
response.json()["assignments"][0]["submissions"]
                            entregas = sorted(entregas, key = lambda x:
x["timemodified"])
                            problema = get_problema(id, numero,prueba)

                            #contamos cuantas entregas actualmente cuentan con
bonus para este problema
                            withbonus =
Entrega.objects.filter(bonus=True,problema=problema).count()

                            #por cada una de las entregas que me devuelve la API
de cada tarea
                            for e in entregas:

                                #busco el grupo en la base de datos, si no existe
lo creo
                                grupo = get_grupo(e["groupid"], asignaturaprueba)
                                if debug:
                                    print(grupo)

                                #busco las entregas que ya existen para este
problema y grupo en la base de datos
                                guardado = buscar_guardadas(e)
                                if debug:
                                    try:
                                        for g in guardado:
                                            print(g)
                                    except:
                                        print("no hay guardados")

                                #¿está vacía?
                                #comprobamos si la información de Moodle contiene
archivos
                                if e["plugins"][0]["fileareas"][0]["files"]: # no
está vacía

                                    #depuramos la información de la ubicación de
los archivos y la almaceno
                                    urls={}
                                    for index in
range(len(e["plugins"][0]["fileareas"][0]["files"])):

urls[index]=e["plugins"][0]["fileareas"][0]["files"][index]["fileurl"]
                                    if debug:
                                        print("not void")

                                    #comprobamos si la entrega es nueva
```

```
                                    #utilizamos la función is_new del módulo
utils.py que devuelve True,0 si es nuevo
                                    new, attemptnumber =
is_new(e["timemodified"], guardado)
                                    if new:
                                        #si es nueva
                                        #si existían entregas anteriores,
actualizo su información con la función update_antiguas de utils.py
                                        #la función devuelve el número
actualizado de entregas que cuentan con bonus tras actualizar la información de
las antiguas
                                        if guardado:
                                            if debug:
                                                print("llega entrega nueva")
                                            withbonus = update_antiguas(guardado,
withbonus)

                                        #creamos una instancia de tipo Entrega
con la información de la API de Moodle
                                        nueva = Entrega(
                                            moodleid = e["id"],
                                            problema = problema,
                                            bonus = False,
                                            grupo = grupo,
                                            tiempo =
datetime.fromtimestamp(e["timemodified"], tz=pytz.timezone('Europe/Madrid')),
                                            urls = urls,
                                            attemptnumber = attemptnumber
                                        )

                                        try:
                                            #comprobamos si hay bonus disponible
para la entrega
                                            #en caso afirmativo, lo asignamos
                                            #guardamos la entega en la base de
datos
                                            if bonus_disponible(problema):
                                                asignar_bonus(nueva)
                                            else:
                                                nueva.save()
                                            if debug:
                                                print("guardado ok")
                                        except Exception as e:
                                            #si ha ocurrido alguna excepción
durante el proceso de gestión de la entrega, el hilo salta a la siguiente entrega
                                            if debug:
                                                print(e)
                                                print("La entrega " +
grupo.nombre + " problema " + str(problema.numero) + " no ha podido guardase")
                                            pass
                                    else:
                                        pass
                                else : # esta vacía
```

```
                                    try:
                                        if debug:
                                            print("void")
                                        # el problema ha sido eliminado
                                        # si existen entregas anteriores se
actualizan con update_antiguas
                                        if guardado:
                                            if is_deleted(e["timemodified"],
guardado):
                                                if debug:
                                                    print("is deleted")
                                                withbonus =
update_antiguas(guardado, withbonus)
                                        else:
                                            if debug:
                                                print("entrega vacia")
                                            pass
                                    except Exception as e:
                                        #si ha ocurrido alguna excepción durante
el proceso de gestión de la entrega, el hilo salta a la siguiente entrega
                                        if debug:
                                            print(e)
                                        pass
                except Exception as e:
                    #si ha ocurrido alguna excepción durante el proceso global,
el hilo termina
                    if debug:
                        print(e)
                        print("thread stopped")
                    self.stop()
        except Exception as e:
            #si ha ocurrido alguna excepción durante el proceso global, el hilo
termina
            if debug:
                print(e)
                print("thread stopped")
            self.stop()
```

VIEWS.PY

Procesa las solicitudes HTTP realizadas por los clientes definiendo las funciones de vista.

```
from django.shortcuts import render, redirect
from django.contrib.auth.decorators import permission_required, user_passes_test
from .models import Entrega, Problema, Prueba, Grupo, Asignatura
from channels.layers import get_channel_layer
channel_layer = get_channel_layer()
from asgiref.sync import async_to_sync
from .moodle import Moodle
```

```
from .utils import *
from .check import *

from threading import Thread
from datetime import datetime, timedelta

import csv
from collections import OrderedDict
import json
import datetime
import time
from django.http import StreamingHttpResponse
from django.http import HttpResponse, HttpResponseServerError,
HttpResponseForbidden


debug = False


@permission_required("chat.change_entrega")
def chatPage(request, prueba, *args, **kwargs):
    """
        Define la vista para la herramienta de corrección y descarga de una
prueba
        Se utiliza el decorador @permission_required("chat.change_entrega")
        para asegurarse de que el usuario tenga los permisos necesarios para
acceder a la vista.

        Argumentos:
        request - solicitud
        prueba  - (int) id de la prueba que se solicita
        """
    #Si el usuario no está autenticado, redirigir a la página de inicio de sesión
    if not request.user.is_authenticated:
        return redirect("login-user")

    #Filtra las entregas asociadas a la prueba que se pasa como argumanto
    prueba = Prueba.objects.get(pk=prueba)
    all_objects= Entrega.objects.filter(problema__prueba = prueba,
antigua=False).order_by("downloaded","tiempo")
    problemas = Problema.objects.filter(prueba=prueba).order_by("numero")
    notas =[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

    #Crea un contexto con las entregas, las notas disponibles, la prueba actual y
los problemas que pertenecen a la prueba
    context= {'all_objects': all_objects, 'notas': notas, 'prueba':
prueba,'problemas':problemas}

    #Devuelve la plantilla de la herramienta de corrección con el contexto creado
    return render(request, "chat/chatPage.html", context)

@permission_required("chat.change_entrega")
def searchPage(request, prueba, problema, *args, **kwargs):
```

```
    """
        Define la vista para el filtro de la herramienta de corrección y descarga
de una prueba
        Se utiliza el decorador @permission_required("chat.change_entrega")
        para asegurarse de que el usuario tenga los permisos necesarios para
acceder a la vista.

        Argumentos:
        request - solicitud
        prueba  - (int) id de la prueba que se solicita
        problema - (int) id del problema que se quiere filtrar
        """
    #Si el usuario no está autenticado, redirigir a la página de inicio de sesión
    if not request.user.is_authenticated:
        return redirect("login-user")

    prueba = Prueba.objects.get(pk=prueba)
    problemas = Problema.objects.filter(prueba=prueba).order_by("numero")

    #Filtra las entregas asociadas a la prueba y problema seleccionado
    all_objects= Entrega.objects.filter(problema__moodleid = problema,
antigua=False).order_by("downloaded","tiempo")
    notas =[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    actual = Problema.objects.get(pk=problema)

    #Crea un contexto con las entregas, las notas disponibles, la prueba actual,
sus problemas y el problema
    #sobre el que se han filtrado las entregas
    context= {'all_objects': all_objects, 'notas': notas, 'prueba': prueba,
'problemas':problemas, 'actual':actual.numero}

    #Devuelve la plantilla de la herramienta de corrección con el contexto creado
    return render(request, "chat/chatPage.html", context)

def startPage(request, *args, **kwargs):
    """
        Define la vista para la página de inicio
        Todos los usuarios autenticados tienen acceso a ésta.
        Argumentos:
        request - solicitud
        """
    #Si el usuario no está autenticado, redirigir a la página de inicio de sesión
    if not request.user.is_authenticated:
        return redirect("login-user")
    try:
        pruebas = Prueba.objects.all().order_by("-fecha")
    except:
        asignatura = Asignatura(año=2022, nombre="Algebra", moodleid=40424)
        asignatura.save()
        pruebas= Prueba(asignatura=asignatura, numero=1, moodleid=1, fecha="2022-
11-03")
        pruebas.save()
```

```
    #Crea un contexto con las todas las pruebas disponibles
    context= {"pruebas":pruebas}

    #Devuelve la vista de la página de inicio con las pruebas disponibles como
contexto.
    return render(request, "chat/index.html", context)



def startHilo(request, value, *args, **kwargs):
    """
        Comprueba si el crawler que recoge las entregas de Moodle para la prueba
determinada está activo y si no, lo inicia
        Argumentos:
        request - solicitud
        value   - (int) id de la prueba para la cual se quiere activar el crawler
        """
    #Si el usuario no está autenticado, redirigir a la página de inicio de sesión
    if not request.user.is_authenticated:
        return redirect("login-user")
    #Función de utils.py que comprueba el estado del crawler y lo inicia si no
está activo
    start_moodle(value)
    #Redirige la petición del usuario a la vista de ranking
    return redirect("score", prueba=value)


@permission_required("chat.change_entrega")
def download(request, entrega,attemptnumber,url):
        """
        Define la rutina de descarga de una entrega.
        Se utiliza el decorador @permission_required("chat.change_entrega")
        para asegurarse de que el usuario tenga los permisos necesarios para
acceder a la vista.

        Argumentos:
        request         - solicitud
        entrega         - (int) id de la entrega seleccionada
        attemptnumber   - (int) número de veces que ha sido reentregada
        url             - (int) id del archivo a descargar (una entrega puede
contar con varios archivos)
        """
    #Si el usuario no está autenticado, redirigir a la página de inicio de sesión
    if not request.user.is_authenticated:
        return redirect("login-user")

    try:
        #Busca la URL del archivo en la base de datos
        pedido = Entrega.objects.get(moodleid=entrega,
attemptnumber=attemptnumber)

        #Comprueba que el crawler sigue activo
        start_moodle(pedido.problema.prueba.moodleid)
```

```
        #Indica que la entrega ha sido descargada
        pedido.downloaded=True
        pedido.save()

        #llama a la API de Moodle con la dirección del documento y el token de
autenticación del bot
        url = pedido.urls[str(url)] + "?token=*******"
        #Devuelve al usuario el archivo pedido
        return redirect(url)

    except:
        pass

@permission_required("chat.change_entrega")
def calificar(request, entrega, attemptnumber, nota):
            """
        Define la rutina de corrección de una entrega.
        Se utiliza el decorador @permission_required("chat.change_entrega")
        para asegurarse de que el usuario tenga los permisos necesarios para
acceder a la vista.

        Argumentos:
        request         - solicitud
        entrega         - (int) id de la entrega seleccionada
        attemptnumber   - (int) número de veces que ha sido reentregada
        nota            - (int) nota  (una entrega puede contar con varios
archivos)
        """
    #Si el usuario no está autenticado, redirigir a la página de inicio de sesión
    if not request.user.is_authenticated:
        return redirect("login-user")

    try:
        #Busca la entrega, le asigna la nota y la marca como corregida
        entrega = Entrega.objects.get(moodleid=entrega,
attemptnumber=attemptnumber)
        nbonus= entrega.problema.prueba.asignatura.bonus
        entrega.nota=nota
        entrega.corregido=True
        entrega.save()

        #Rutina de comprobación del premio de velocidad

        if check_bonus(entrega):
            #Si la nota está marcada con premio de velocidad, pero la nota es
<10, deja de ser candidata
            if nota<10:
                quitar_bonus(entrega)
                #Al quitar el premio a la entrega, se buscan las siguientes
entregas candidatas a
                #premio de velocidad en función del tiempo de entrega
                #Si hay alguna candidata, se le asigna el premio de
                if bonus_disponible(entrega.problema):
```

```
                    candidatas = buscar_candidatas(entrega.problema)
                    if candidatas:
                        if debug:
                            print("bonus corrected")
                        asignar_bonus(candidatas[0])
        #Si la nota es un 10, comprobamos si la entrega tiene premio de velocidad
marcado
        #Si no, comprobamos si es candidata. Si resulta candidata, la marcamos
con premio de velocidad
        #Y le ponemos un 15. Quitamos el premio a las entrega posterior que
estiviese marcada con bonus sin merecerlo
        if nota == 10:
            if not check_bonus(entrega): #bonus == False
                if count_bonus(entrega.problema, entrega) < nbonus:
                    asignar_bonus(entrega)
                    if count_bonus(entrega.problema)> nbonus:
                        posteriores= buscar_withbonus(entrega)
                        if posteriores:
                            quitar_bonus(posteriores[0])
            else:
                #Si está marcada con premio de velocidad, le ponemos un 15.
                asignar_bonus(entrega)
    except Exception as e:
        if debug:
            print(e)
        pass
    finally:
        #Redirige  al usuario a la herramienta de corrección filtrando por el
problema de la entrega corregida
        return redirect("buscador",prueba=entrega.problema.prueba.moodleid,
problema=entrega.problema.moodleid)

def scorePage(request, prueba,*args, **kwargs):
        """
        Define la vista del ranking en tiempo real.

        Argumentos:
        request         - solicitud
        prueba          - (int) id de la prueba actual
        """
    #Si el usuario no está autenticado, redirigir a la página de inicio de sesión
    if not request.user.is_authenticated:
        return redirect("login-user")

    prueba =Prueba.objects.get(pk=prueba)
    asignatura=prueba.asignatura
    entregas_prueba=Entrega.objects.filter(problema__prueba = prueba)
    grupos = Grupo.objects.filter(asignatura=asignatura).order_by('nombre')
    problemas = Problema.objects.filter(prueba=prueba).order_by("numero")
    notas={}
    totales={}
    #Calcula las notas totales y provisionales de cada grupo en la prueba para
las diferentes gráficas
```

```
    for grupo in grupos:
        totales[grupo.nombre]={}
        totales[grupo.nombre]["total"]=0
        totales[grupo.nombre]["provisional"]=0
        totales[grupo.nombre]["puntos"]=0
        notas[grupo.nombre]={}
        for problema in problemas:
            notas[grupo.nombre][problema.moodleid]={}
            entregas = entregas_prueba.filter(grupo=grupo, problema= problema,
antigua=False).order_by('-tiempo')
            if not entregas:
                notas[grupo.nombre][problema.moodleid]["final"]="-"
                notas[grupo.nombre][problema.moodleid]["provisional"]=0
            else:
                if entregas[0].corregido == True:

notas[grupo.nombre][problema.moodleid]["final"]=entregas[0].nota
                    totales[grupo.nombre]["total"]+= entregas[0].nota
                    totales[grupo.nombre]["puntos"] += entregas[0].nota
                    notas[grupo.nombre][problema.moodleid]["provisional"]=0
                else:
                    notas[grupo.nombre][problema.moodleid]["final"]="-"
                    if entregas[0].bonus== False:
                        totales[grupo.nombre]["provisional"]+=10
                        totales[grupo.nombre]["puntos"] += 10
                        notas[grupo.nombre][problema.moodleid]["provisional"]=10
                    else:
                        totales[grupo.nombre]["puntos"] += 15
                        totales[grupo.nombre]["provisional"]+=15
                        notas[grupo.nombre][problema.moodleid]["provisional"]=15

    #Genera un ranking organizando un los grupos por nota total
    ranking = sorted(totales.items(), key = lambda x: x[1]["puntos"],
reverse=True)

    #Crea un contexto con los datos por grupo
    context = {"notas":notas,"problemas":problemas, "totales":totales,
"ranking":ranking, "prueba":prueba}

    #Redirige al usuario a la página de ranking pasando como argumento el
contexto creado
    return render(request, "chat/scoreboard.html", context)


def scoreFinalPage(request, prueba, export,*args, **kwargs):
    """
        Define los datos del ranking final de cada prueba, tanto para la vista
como para la exportación.

        Argumentos:
        request         - solicitud
        prueba          - (int) id de la prueba actual
        export          - (int) si es 1, exporta los datos
```

```
                                si es 0, redirige al usuario a la página de
ranking final
    """

    #Si el usuario no está autenticado, redirigir a la página de inicio de sesión
    if not request.user.is_authenticated:
        return redirect("login-user")

    #Si el usuario solicita la exportación y no tiene permiso para ello, genera
un error
    if not request.user.is_staff and export != 0:
        return HttpResponseForbidden("You don't have permission to access this
page.")
    try:
        #Calcula los datos del ranking final
        prueba =Prueba.objects.get(pk=prueba)
        asignatura=prueba.asignatura
        entregas_prueba=Entrega.objects.filter(problema__prueba = prueba)
        grupos = Grupo.objects.filter(asignatura=asignatura).order_by('nombre')
        problemas = Problema.objects.filter(prueba=prueba).order_by("numero")
        notas={}
        for grupo in grupos:
            notas[grupo.nombre]={}
            notas[grupo.nombre]["total"]=0
            for problema in problemas:
                notas[grupo.nombre][problema.moodleid]={}
                entregas = entregas_prueba.filter(grupo=grupo, problema=
problema, antigua=False).order_by('-tiempo')
                if not entregas:
                    notas[grupo.nombre][problema.moodleid]["final"]="-"
                    notas[grupo.nombre][problema.moodleid]["provisional"]=0
                else:
                    if entregas[0].corregido == True:

notas[grupo.nombre][problema.moodleid]["final"]=entregas[0].nota
                        notas[grupo.nombre][problema.moodleid]["provisional"]=0
                        notas[grupo.nombre]["total"]+=entregas[0].nota
                    else:
                        notas[grupo.nombre][problema.moodleid]["final"]="-"
                        if entregas[0].bonus== False:

notas[grupo.nombre][problema.moodleid]["provisional"]=10
                        else:

notas[grupo.nombre][problema.moodleid]["provisional"]=15

        #Si se solicita la descarga, se genera un CSV con los datos a exportar
        if export==1:
            try:
                response = HttpResponse(content_type='text/csv')
                response['Content-Disposition'] = 'attachment;
filename="ranking.csv"'
                response.write(u'\ufeff'.encode('utf8'))
```

```
                writer = csv.writer(response, delimiter=';',
quoting=csv.QUOTE_ALL, quotechar='"')
                headers = ["grupo"]
                keys = set().union(*notas.values())
                columns = list(keys)

                writer.writerow(["grupo", "total"] + [p.numero for p in
problemas])

                for grupo, values in notas.items():
                    row = [grupo]
                    for key, val in values.items():
                        try:
                            row.append(val["final"])
                        except:
                            row.append(val)
                    writer.writerow(row)
                #Se devuelve al usuario el archivo CSV generado
                return response

            #Si ocurre algún error en la creación del archivo CSV, se notifica
del error al usuario
            except Exception as e:
                error_message = "Error exporting data: {}".format(str(e))
                return HttpResponseServerError(error_message)
        #Si no se ha solicitado la descarga, se redirige al usuario a la página
del ranking final
        context = {"notas":notas,"problemas":problemas, "prueba":prueba}
        return render(request, "chat/scoreboardFinal.html", context)

    #Gestiona las excepciones
    except Exception as e:
        if debug:
            print(e)
        pass

def globalRanking(request, prueba, export, *args, **kwargs):
    """
        Define los datos del ranking global de la asignatura hasta la prueba
actual, tanto para la vista como para la exportación.

        Argumentos:
        request         - solicitud
        prueba          - (int) id de la prueba actual
        export          - (int) si es 1, exporta los datos
                                si es 0, redirige al usuario a la página de
ranking global
    """
    #Si el usuario no está autenticado, redirigir a la página de inicio de sesión
    if not request.user.is_authenticated:
        return redirect("login-user")
    #Si el usuario solicita la exportación y no tiene permiso para ello, genera
un error
```

```
    if not request.user.is_staff and export != 0:
        return HttpResponseForbidden("You don't have permission to access this
page.")
    #Genera los datos
    try:
        actual =Prueba.objects.get(pk=prueba)
        asignatura=actual.asignatura
        grupos = Grupo.objects.filter(asignatura=asignatura).order_by('nombre')
        pruebas = Prueba.objects.filter(asignatura=asignatura,
fecha__lte=actual.fecha).order_by("fecha")
        notas={}
        for grupo in grupos:
            notas[grupo.nombre]={}
            notas[grupo.nombre]["total"]=0
            for p in pruebas:
                entregas_prueba=Entrega.objects.filter(problema__prueba = p)
                problemas = Problema.objects.filter(prueba=p).order_by("numero")
                notas[grupo.nombre][p.numero]=0

                for problema in problemas:
                    entregas = entregas_prueba.filter(grupo=grupo, problema=
problema, antigua=False).order_by('-tiempo')
                    if not entregas:
                        pass
                    else:
                        if entregas[0].corregido == True:
                            notas[grupo.nombre][p.numero] += entregas[0].nota
                            notas[grupo.nombre]["total"] += entregas[0].nota
                        else:
                            if entregas[0].bonus== False:
                                notas[grupo.nombre][p.numero] += 10
                                notas[grupo.nombre]["total"] += 10
                            else:
                                notas[grupo.nombre][p.numero] += 15
                                notas[grupo.nombre]["total"] += 15

        ranking = dict(sorted(notas.items(), key = lambda x: x[1]["total"],
reverse=True))
        #Si se solicita la descarga, se genera un CSV con los datos a exportar
        if  export==1:
            try:
                response = HttpResponse(content_type='text/csv')
                response['Content-Disposition'] = 'attachment;
filename="global_ranking.csv"'
                response.write(u'\ufeff'.encode('utf8'))
                writer = csv.writer(response, delimiter=';',
quoting=csv.QUOTE_ALL, quotechar='"')
                headers = ["grupo"]
                keys = set().union(*ranking.values())
                columns = list(keys)

                writer.writerow(["grupo", "total"] + [p.numero for p in pruebas])
```

```
                for grupo, values in ranking.items():
                    row = [grupo]
                    for key, val in values.items():
                        row.append(val)
                    writer.writerow(row)
                #Se devuelve al usuario el archivo CSV generado
                return response
            #Si ocurre algún error en la creación del archivo CSV, se notifica
del error al usuario
            except Exception as e:
                error_message = "Error exporting data: {}".format(str(e))
                return HttpResponseServerError(error_message)

        #Si no se ha solicitado la descarga, se redirige al usuario a la página
del ranking global
        context = {"notas":ranking,"pruebas":pruebas, "prueba":actual,
"asignatura": asignatura}
        return render(request, "chat/global.html", context)

    except Exception as e:
        if debug:
            print(e)
        pass

@user_passes_test(lambda user: user.is_staff, login_url='/auth/login/')
def exportRaw(request, año):
        """
        Define la rutina de exportación del dato crudo.
        Se utiliza el decorador @user_passes_test()
        para asegurarse de que el usuario tenga los permisos necesarios para
acceder a la vista.
        Si no los tiene, se le redirige a la página de login.

        Argumentos:
        request         - solicitud
        año             - (int) año de los datos a exportar
        """
    try:
        #Se crea un CSV con todos los datos de las entregas y se devuelve al
usuario
        response = HttpResponse(content_type='text/csv')
        response['Content-Disposition'] = 'attachment; filename="raw_data.csv"'
        response.write(u'\ufeff'.encode('utf8'))
        writer= csv.writer(response, delimiter=';', quoting=csv.QUOTE_ALL,
quotechar='"')

writer.writerow(["año","asignatura","prueba","problema","id","bonus","grupo","num
ero_grupo","nota","tiempo","corregido","downloaded","antigua"])

        entregas = Entrega.objects.filter(problema__prueba__asignatura__año=año)
        for e in entregas:

writer.writerow([año,e.problema.prueba.asignatura.nombre,e.problema.prueba.numero
```

```
,e.problema.numero,e.moodleid,e.bonus,e.grupo.nombre,e.grupo.numero,e.nota if
e.nota else None,e.tiempo,e.corregido,e.downloaded, e.antigua])
        return response
    #Si ocurre algún error en la creación del archivo CSV, se notifica del error
al usuario
    except Exception as e:
        error_message = "Error exporting data: {}".format(str(e))
        return HttpResponseServerError(error_message)


@user_passes_test(lambda user: user.is_staff, login_url='/auth/login/')
def export(request):
    """
        Define la vista de la interfaz de datos.
        Se utiliza el decorador @user_passes_test()
        para asegurarse de que el usuario tenga los permisos necesarios para
acceder a la vista.
        Si no los tiene, se le redirige a la página de login.

        Argumentos:
        request         - solicitud
        """
    try:
    pruebas = Prueba.objects.order_by("-fecha")
    asignaturas = Asignatura.objects.order_by("-año")
    años = Asignatura.objects.values_list('año', flat=True).distinct()
    context= { 'pruebas': pruebas, 'asignaturas': asignaturas, 'años':años}
    #Redirige al usuario a la interfaz de datos con el contexto creado
    return render(request, "chat/export.html", context)

@user_passes_test(lambda user: user.is_staff, login_url='/auth/login/')
def export_groups_with_bonus(request, asignatura, withbonus):
    """
        Define la exportación de los grupos que han recibido algún premio de
velocidad.
        Se utiliza el decorador @user_passes_test() para asegurarse de que el
usuario
        tenga los permisos necesarios para acceder a la vista.
        Si no los tiene, se le redirige a la página de login.

        Argumentos:
        request         - solicitud
        asignatura      - (id) asignatura para la que se quieren exportar los
grupos con premio
        """
    try:
        if withbonus==1:
            grupos = Grupo.objects.filter(entrega__bonus=True,
entrega__antigua=False,
entrega__problema__prueba__asignatura__moodleid=asignatura).distinct()

        else:
```

```
            grupos =
Grupo.objects.filter(asignatura__moodleid=asignatura).distinct()

        asignatura = Asignatura.objects.get(pk=asignatura)
        response = HttpResponse(content_type='text/csv')
        response['Content-Disposition'] = 'attachment; filename="with_bonus.csv"'
        response.write(u'\ufeff'.encode('utf8'))

        writer = csv.writer(response, delimiter=';', quoting=csv.QUOTE_ALL,
quotechar='"')

        writer.writerow([ asignatura.nombre + " " + str(asignatura.año)])

        if withbonus ==1:
            writer.writerow(["premios de velocidad " ])
        for grupo in grupos:
            writer.writerow([grupo.nombre, grupo.numero])
        #Se devuelve al usuario un archivo CSV con los grupos que han tenido
algún premio de velocidad
        #en alguna de las pruebas realizadas de la asignatura
        return response

    except Exception as e:
        error_message = "Error exporting data: {}".format(str(e))
        return HttpResponseServerError(error_message)
```

URLS.PY

Define los patrones de URLs que admite la plataforma y determina que función de views.py debe gestionar la petición HTTP asociada a cada URL.

```
from django.urls import path, include
from chat import views as chat_views
from django.contrib.auth.views import LoginView, LogoutView


urlpatterns = [
    path("", chat_views.startPage, name="start-page"),
    path("chat/<int:prueba>", chat_views.chatPage, name="chat-page"),
    path("chat/<int:prueba>/<int:problema>", chat_views.searchPage,
name="buscador"),
    path("download/<int:entrega>/<int:attemptnumber>/<url>",
chat_views.download),
    path("calificar/<int:entrega>/<int:attemptnumber>/<int:nota>",
chat_views.calificar),
    path("scoreboard/<int:prueba>", chat_views.scorePage, name="score"),
```

```
    path("scoreboard/static/<int:prueba>/<int:export>",
chat_views.scoreFinalPage, name="static"),
    path("startHilo/<int:value>", chat_views.startHilo),
    path("delete",chat_views.deleteDuplicates),
    path("global/<int:prueba>/<int:export>", chat_views.globalRanking),
    path("exportraw/<int:año>", chat_views.exportRaw),
    path("export", chat_views.export, name="export"),
    path("export/grupos/<int:asignatura>/<int:withbonus>",
chat_views.export_groups_with_bonus),
    # login-section
    path("auth/login/", LoginView.as_view
         (template_name="chat/LoginPage.html"), name="login-user"),
    path("auth/logout/", LogoutView.as_view(), name="logout-user"),
]
```

ROUTING.PY

Define los patrones URL para el protocolo websocket.

```
## Define los patrones URL para el protocolo websocket
from django.urls import path , include
from chat.consumers import ChatConsumer, MyConsumer

websocket_urlpatterns = [
    path("" , ChatConsumer.as_asgi()) ,
    path("chat/", MyConsumer.as_asgi()),
]
```

CONSUMERS.PY

Define la lógica de los clientes websocket.

```
import json
from channels.generic.websocket import AsyncWebsocketConsumer
from asgiref.sync import async_to_sync
from channels.generic.websocket import WebsocketConsumer


class ChatConsumer(AsyncWebsocketConsumer):
    async def connect(self):
        self.roomGroupName = "chat"
        await self.channel_layer.group_add(
```

```
            self.roomGroupName ,
            self.channel_name
        )
        await self.accept()
    async def disconnect(self , close_code):
        await self.channel_layer.group_discard(
            self.roomGroupName ,
            self.channel_layer
        )
    async def receive(self, text_data):
        print("MESSAGE RECEIVED")
        text_data_json = json.loads(text_data)
        message = text_data_json["message"]
        username = text_data_json["username"]
        await self.channel_layer.group_send(
            self.roomGroupName,{
                "type" : "chat.message" ,
                "message" : message ,
                "username" : username ,
            })

    async def chat_message(self , event) :
        print("message sent")
        message = event["message"]
        username = event["username"]
        await self.send(text_data = json.dumps({"message":message
,"username":username}))

    async def print(self, event, type = "print"):
        print('we are in print consumer')
        print(message)



class MyConsumer(WebsocketConsumer):
    def connect(self):
        self.roomGroupName = "chat"
        async_to_sync(self.channel_layer.group_add)(self.roomGroupName,
self.channel_name)
        self.accept()

    def receive(self, text_data):
        # Called with either text_data or bytes_data for each frame
        text_data_json = json.loads(text_data)
        message = text_data_json["message"]
        username = text_data_json["username"]
        # You can call:

        async_to_sync(self.channel_layer.group_send)(
            "chat",
            {
                "type": "chat.message",
                "username": username,
```

```
                "message": message,
            }

)
    def chat_message(self , event) :
        print("message sent")
        message = event["message"]
        username = event["username"]
        self.send(text_data = json.dumps({"message":message
,"username":username}))

    def disconnect(self, close_code):
        async_to_sync(self.channel_layer.group_discard)("chat",
self.channel_layer)
```

ADMIN.PY

Define los modelos sobre los que los administradores pueden trabajar en la interfaz de administrador.

```
from django.contrib import admin

# Define los modelos sobre los que el administrador puede trabajar en la interfaz
de administrador

from .models import *
admin.site.register(Asignatura)
admin.site.register(Prueba)
admin.site.register(Grupo)
admin.site.register(Problema)
admin.site.register(Entrega)
```