UNIVERSITAT DE BARCELONA

Trabajo de Fin de Grado

GRADO EN INGENIERÍA INFORMÁTICA

Facultad de Matemáticas e Informática
Universidad de Barcelona

MathCampus: una aplicación web de ejercicios matemáticos automatizados para estudiantes universitarios

---

Antoni Tudurí Morente

Director: Nahuel Norberto Statuto
Realizado en: Departamento de Matemáticas y Informática

Barcelona, 10 de junio de 2024

Resumen

Una de las formas en que se puede mejorar el proceso de aprendizaje autónomo es mediante la enseñanza basada en la tecnología; a pesar de ello, muy pocas plataformas educativas permiten a los estudiantes universitarios practicar matemáticas de manera ilimitada y adaptativa.

Es debido a esto que el presente proyecto de fin de grado se centra en la creación de “MathCampus”, una aplicación web diseñada para que los estudiantes universitarios puedan practicar ejercicios matemáticos sin límite de cantidad, totalmente automatizado. Para desarrollar este sistema, se emplean herramientas modernas demandadas en la industria como por ejemplo *Django* para el back-end, *React* para el front-end y *Tailwind* como framework de CSS. Todo esto con el fin de desarrollar una API REST además de un front-end capaz, que no solo cumpla con los estándares de diseño y mejores prácticas, sino que además genere una experiencia de usuario fluida y una arquitectura escalable.

El código fuente de todo el proyecto se encuentra en el siguiente repositorio de GitHub: https://github.com/tuduri11/MathCampus.git

Palabras claves: aprendizaje autónomo, API REST, React, Django, desarrollo web, aplicación web, AWS.

Resum

En el context educatiu actual, la integració de la tecnologia en l'ensenyament és fonamental per millorar l'aprenentatge autònom. Malgrat els avenços tecnològics, no existeixen moltes plataformes educatives que permetin als estudiants universitaris practicar matemàtiques de manera il·limitada i adaptativa.

És per això que aquest projecte de fi de grau es centra en la creació de "MathCampus", una aplicació web dissenyada per a que els estudiants universitaris puguin practicar exercicis matemàtics sense límit de quantitat, totalment automatitzat. Per desenvolupar aquest sistema, s'utilitzen eines modernes demandades a la indústria com ara *Django* en la part de back-end, *React* per la part del front-end i *Tailwind* com a framework de CSS. Tot això per crear una API REST a més d'un front-end intuïtiu i elegant, que no solament compleixi amb els estàndards de disseny i millores pràctiques, sinó que també ofereix una experiència d'usuari fluida i una arquitectura escalable.

Podem trobar el codi del projecte en el següent repositori de GitHub: https://github.com/tuduri11/MathCampus.git

Paraules clau: aprenentatge autònom, API REST, React, Django, desenvolupament web, aplicació web, AWS.

Abstract

In the current educational context, the integration of technology in education is essential to improve autonomous learning. Despite technological advances, there are not many educational platforms that allow university students to practice mathematics in an unlimited and adaptive way.

This project focuses on the development of “MathCampus”, a web application designed to allow university students to practice mathematical exercises in an automated way. Modern technologies with high demand in the market have been used in order to develop it, such as *Django* in the back-end, *React* in front-end and *Tailwind* as a CSS framework. All this to create an intuitive and elegant REST API and front-end, which not only achieves design standards and best practices, but also offers a fluid user experience and a scalable architecture.

We can find the project code in the following GitHub repository: https://github.com/tuduri11/MathCampus.git

Keywords: autonomous learning, REST API, React, Django, web development, web application, AWS.

Índice de contenido

1. Introducción

En la actualidad, la capacidad de aprender y, si se desea, de practicar por su cuenta es muy valiosa con el fin de conseguir el éxito en la formación de uno mismo y en cualquier profesión. Esto tiene importancia fundamental en las matemáticas, ya que los estudiantes se encuentran delante de conceptos que requieren un alto rendimiento y una aplicación precisa.

A conciencia de ello, se ha creado una plataforma web llamada MathCampus, la cual, específicamente, está pensada para los estudiantes universitarios de carreras como matemáticas, ingenierías o similares. El punto de vista de esta tecnología es facilitar la práctica autónoma y aumentar el entendimiento de las matemáticas a partir de ejercicios automatizados. Con ese fin, la herramienta genera problemas matemáticos que los alumnos pueden solucionar, ofreciendo así una asistencia que apoya su formación independiente.

Este proyecto se inició con una investigación de las herramientas didácticas accesibles y de los requerimientos específicos de los estudiantes. Luego, se ha creado la primera versión del tablero Kanban con el fin de administrar de manera eficaz la labor de desarrollo y establecer la importancia respecto a las expectativas. Posteriormente, se ha desarrollado una idea del diseño interno que incluía el diseño del sistema, de la base de datos, de la arquitectura del back-end e incluso un diseño del front-end.

Una vez implementadas estas características, se desplegó la aplicación web en Amazon Web Services (*AWS*).

1.1 Motivación

La creación de la idea de MathCampus se origina por una combinación de motivaciones personales y académicas. En primer lugar, desde el inicio de mi carrera universitaria, he sentido pasión por el desarrollo de software y he tenido un gran deseo por especializarme en este campo, y gracias a ello, he tenido la motivación necesaria para iniciar este gran proyecto que me ha permitido utilizar y profundizar todo lo aprendido.

Aparte de mi propio interés, a partir de la cantidad de años que he estudiado ingeniería, he descubierto que a pesar de la creciente demanda de herramientas para el aprendizaje autónomo, no hay servicios pensados para los estudiantes universitarios que desean ejercitar y perfeccionar sus habilidades en las matemáticas.

La gran mayoría de herramientas se focalizan en niveles educativos más bajos, en particular en el instituto, o no brindan la amplitud necesaria para formar a los estudiantes universitarios.

En conclusión, el desempeño de diseñar un proyecto de software desde cero, desde la definición de los objetivos hasta el despliegue de la aplicación, es algo que me motiva, ya que la ocasión de crear algo nuevo y funcional impulsa mi pasión por el desarrollo de software.

1.2 Contexto

La educación matemática en el nivel universitario enfrenta desafíos únicos, principalmente debido a la naturaleza abstracta de la disciplina. Como se destaca en la literatura, el aprendizaje efectivo de las matemáticas requiere un enfoque continuado y metódico, no siendo suficiente un estudio de última hora, ya que la preparación adecuada en matemáticas debe incluir *mucho ensayo y mucho error*, lo que facilita la retroalimentación continua y permite a los estudiantes analizar y aprender de sus errores [1].

En el contexto actual, en el cual existe una gran dependencia a las tecnologías digitales, se observa una tendencia en auge en el uso de herramientas de aprendizaje online. La necesidad de acceso a recursos educativos de calidad en línea se vuelve aún más importante cuando los estudiantes enfrentan limitaciones para asistir a clases presenciales, ya sea por compromisos laborales, familiares o de cualquier otro tipo, casos que ocurren mucho más en época universitaria.

A partir de aquí, la aplicación va dirigida a los estudiantes universitarios, ofreciendo así una experiencia de práctica más elástica para practicar y mejorar a su propio ritmo y conseguir una réplica instantánea.

1.3 Objetivos

En esta sección explicaremos los objetivos fundamentales y específicos que se han llevado durante la elaboración de este proyecto.

1.3.1 Objetivos principales

El propósito fundamental de este proyecto es desarrollar una aplicación web interactiva pensada para los estudiantes universitarios. Nuestra herramienta web permitirá a los usuarios resolver problemas matemáticos generados automáticamente, facilitando un

aprendizaje autónomo y adaptativo. Además, la aplicación tendrá una apariencia atractiva para el usuario e incluirá herramientas complejas que posibilitan soluciones avanzadas y los pasos necesarios para acceder a ellas, además de un chatbot para esclarecer cualquier duda.

1.3.2 Objetivos específicos

Identificamos los objetivos específicos relacionados con la formación que he recibido a lo largo de mi carrera.

- Crear una interfaz de usuario de la aplicación utilizando un framework de front-end llamado *React*, brindando una experiencia interactiva y ágil con el soporte de *Tailwind* CSS.
- Diseñar e implementar una REST API usando el framework *Django* que permita la comunicación entre los servicios web y otros servicios de terceros:
    - Definición de los endpoints y las acciones necesarias para transferir datos importantes para la elaboración de los ejercicios automatizados.
    - Garantizar conexiones seguras y eficientes entre la aplicación y otros servicios a través de esta REST API.
- Diseñar una arquitectura robusta y escalable e integrar la base de datos PostgreSQL.
- Identificar los requisitos funcionales y no funcionales de los usuarios y construir un tablero Kanban adaptándose a las necesidades de estos.
- Implementar un sistema de autenticación sólido para proteger la seguridad y privacidad de los datos del usuario.
- Asegurar que la ejecución del sitio web sea accesible y funcional para el usuario en diferentes tipos de plataformas y dispositivos.

2. Estado del arte

En esta sección, se investigarán herramientas existentes en el mercado similares a la aplicación desarrollada en este proyecto para encontrar sus debilidades y mejorar la aplicación web a través de sus fallas.

Primero, procedemos a examinar una de las herramientas en línea más conocidas, *Khan Academy*. Es muy popular porque ofrece mucha variedad de cursos y lo más importante para nosotros, muchos recursos en matemáticas. En ellos, existen ejercicios adaptados al nivel de habilidad de los usuarios con los que pueden interactuar, sirviéndose de explicaciones paso a paso y una réplica instantánea de los ejercicios. Aunque *Khan Academy* [20] sea una plataforma muy popular y muy variada, su interfaz de usuario resulta un poco confusa y muy poco elaborada, como podemos observar en la **Figura 1**. La interfaz se basa mayoritariamente en el color blanco y no existe la posibilidad de poner el modo oscuro, algo indispensable en las aplicaciones de hoy en día. Además, la estructuración de la página es muy simple y es difícil moverse entre cursos y ejercicios.

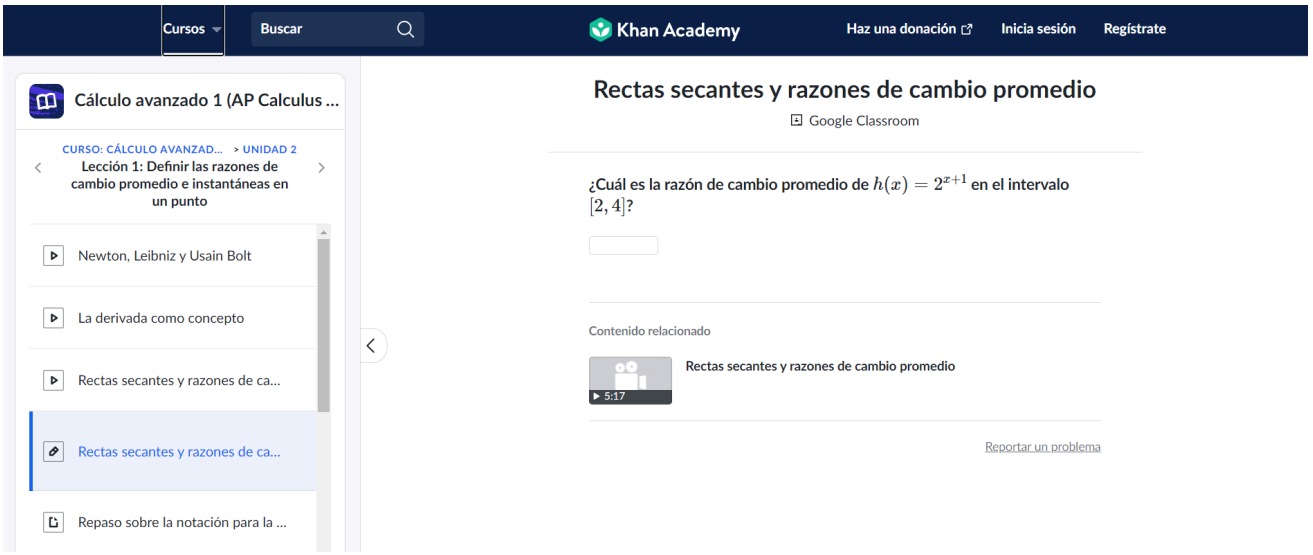

Figura 1. Ejercicio en *Khan Academy*.

Otra aplicación web famosa es *Symbolab*, una plataforma de educación matemática en línea que proporciona herramientas para resolver problemas matemáticos. En él podemos encontrar una gran cantidad de problemas matemáticos de diferente número y apartados

donde los usuarios pueden practicar. Sin embargo, la herramienta tiene algunas limitaciones: en primer lugar, la gran mayoría de las funciones importantes (como las soluciones paso a paso de los ejercicios) requieren una suscripción de pago mensual. Además, el nivel de todos los ejercicios está por debajo del nivel que se requiere para los estudiantes universitarios. Por último, si el usuario desea realizar más de un ejercicio, también se requiere que esté suscrito en la versión premium, tal y como vemos en la **Figura 2**.

Figura 2. Ejercicio en *Symbolab*

A partir de la investigación de estas dos plataformas online, hemos diseñado MathCampus para ofrecer una interfaz de usuario sencilla e intuitiva, permitiendo a los estudiantes centrarse únicamente en la elaboración de los ejercicios deseados. Además, para mejorar la usabilidad, los usuarios tendrán 4 posibles respuestas (si el ejercicio lo requiere), podrán contestarlas y obtener *feedback* instantáneo con la solución correcta, todo esto sin estar suscritos a ninguna suscripción de pago.

3. Análisis

En esta sección se dará un análisis en profundidad del proyecto, examinando diferentes partes importantes para que sea correcta la implementación. Primero, se realizará un estudio de los requerimientos funcionales, es decir, los requerimientos que determinan las tareas y servicios fundamentales que tiene que proveer el sistema a los usuarios. Después, se expondrán los requerimientos no funcionales, que son los que establecen los parámetros de calidad y las limitaciones que el sistema debe poseer. Más tarde, se ejecutará el plan de trabajo (Backlog), que es una secuencia de tareas en orden de importancia y las características que se requieren para llevar a cabo el proyecto. Finalmente, se pondrá en marcha un plan de trabajo Kanban, a fin de facilitar la administración y control de las labores en diversas fases.

Un correcto análisis y diseño de requisitos favorece un entendimiento y definición de las necesidades del usuario y del sistema, con el objetivo de asegurar que el software que se ha desarrollado cumpla con las expectativas esperadas.

3.1 Requisitos funcionales

Los requisitos funcionales describen la funcionalidad y las operaciones que una aplicación debe proporcionar para satisfacer las necesidades y expectativas del usuario. Estos requisitos indican qué puede hacer el sistema y qué interacciones puede realizar el usuario. En nuestro caso, podemos analizar los siguientes requisitos:

- **Registro de usuario:**
    - El sistema debe permitir a los usuarios registrarse proporcionando su nombre, correo electrónico y contraseña válida.
    - El sistema debe validar el formato de correo electrónico proporcionado por el usuario.
    - El sistema debe verificar que el correo electrónico sea único dentro del sistema.
    - El sistema debe verificar que la contraseña proporcionada por el usuario tenga al menos ocho caracteres y contenga al menos un número, una letra minúscula, una mayúscula y un carácter especial.

    - El sistema debe verificar que los nombres y apellidos no contengan números ni caracteres especiales.

- **Autentificación de usuario:**
    - El sistema debe permitir a los usuarios autenticarse con su correo electrónico y contraseña.
    - El sistema debe verificar las credenciales del usuario utilizando una base de datos.
    - El sistema debe proporcionar acceso seguro mediante tokens de acceso y de refresco tras una autenticación exitosa.
    - El sistema debe permitir a los usuarios cerrar sesión de forma segura.
- **Visualización y modificación del perfil del usuario:**
    - El sistema debe permitir a los usuarios ver su perfil y toda la información.
    - El sistema debe permitirle cambiar su contraseña actual por una nueva contraseña.
- **Visualización de universidades, carreras, asignaturas, temas y problemas:**
    - El sistema debe permitir a los usuarios visualizar cualquier universidad y todo su contenido de principio a fin.
- **Chathy:**
    - El sistema debe permitir a los usuarios que puedan conversar con una inteligencia artificial parecida al ChatGPT.
- **Ejercicios:**
    - El sistema debe permitir a los usuarios visualizar y realizar ejercicios.
    - El sistema debe permitir a los usuarios elegir una de las 4 opciones de respuesta (si las hay) y poder ver si han acertado o fallado.
    - El sistema debe permitir a los usuarios ver la solución completa y el paso a paso.

- **Estadísticas:**
    - El sistema debe permitir a los usuarios poder ver sus estadísticas de los ejercicios realizados y ver el ranking mundial de la aplicación junto a su posición.
- **Premium:**
    - El sistema debe permitir a los usuarios convertirse en premium pagando la cuota mensual a través de Stripe para así conseguir *Mathys* (la moneda virtual de la aplicación).
    - El sistema debe permitir a los usuarios desactivar su suscripción premium y poder dejar de pagar la cuota mensual.
- **Favoritos:**
    - El sistema debe permitir a los usuarios agregar cualquier problema matemático a su lista de favoritos.
    - El sistema debe permitir a los usuarios eliminar problemas matemáticos de su lista de favoritos.

3.2 Requisitos no funcionales

Los requisitos no funcionales contemplan características como el rendimiento, la seguridad y la usabilidad. Estos aseguran que el software no se restrinja a realizar su función, sino que además sea de alta calidad. En nuestro caso, analizamos los siguientes requisitos:

- **Usabilidad:**
    - La interfaz de usuario debe ser simple y fácil de navegar para usuarios de diferentes habilidades técnicas y adaptable a diferentes dispositivos, como por ejemplo, ordenadores, tablets, teléfonos móviles…
- **Seguridad:**
    - Los datos se deben encriptar durante la transmisión y el almacenamiento.
    - La plataforma debe dar un alto nivel de protección contra vulnerabilidades de seguridad comunes y acceso no autorizado.

- **Escalabilidad:**
    - La aplicación debe diseñarse para permitir el crecimiento futuro sin comprometer el rendimiento.
- **Rendimiento:**
    - La plataforma debe ser capaz de admitir varios usuarios sin que se note un gran retraso.

3.3 Desarrollo del Backlog

Un Backlog es una lista priorizada de requisitos funcionales y no funcionales necesarios para lograr los objetivos establecidos. Este concepto, adecuado a nuestro proyecto, es muy importante para la planificación y desarrollo de la aplicación web.

El Backlog permite desarrollar una estrategia clara, priorizar tareas y facilitar la colaboración dentro del equipo de desarrollo. Esta sección muestra cómo el uso del Backlog puede conseguir una gestión eficiente del proyecto y garantizar que las necesidades de los usuarios y los objetivos del proyecto se consigan de manera efectiva. En la tabla que se presenta a continuación (**Figura 3**), se puede ver el Backlog completo de todas las funcionalidades de este proyecto.

User Story	Descripción	Criterios de aceptación
U-1	Como usuario no registrado, quiero poder acceder a la plataforma para explorar sus funcionalidades básicas y entender los beneficios de registrarse.	- El usuario puede acceder a la página principal sin haberse registrado. - El usuario puede navegar por las páginas principales sin interrupciones o errores.
U-2	Como usuario no registrado, me gustaría poder registrarme en la plataforma para obtener	- El usuario puede acceder al formulario de registro desde cualquier página de la aplicación web. - Los datos ingresados serán validados para

	acceso completo a todas las funciones y personalizar mi experiencia.	garantizar el cumplimiento de formatos y restricciones. - Si alguna entrada no cumple con la validación requerida, se mostrará un mensaje de error. - Después de un registro exitoso,los usuarios podrán disfrutar de las ventajas de registrarse en la plataforma.
U-3	Como usuario me gustaría poder ver todas las universidades disponibles.	- El usuario debe poder ver todas las universidades disponibles en la aplicación web. - El usuario debe poder hacer clic en cualquier universidad para ver todas sus especialidades.
U-4	Como usuario, quiero poder ver todas las carreras de la universidad seleccionada.	- El usuario debe poder ver todas las carreras de la universidad seleccionada anteriormente. - El usuario debe poder hacer clic en alguna de las carreras para ver todas sus asignaturas.
U-5	Como usuario, quiero poder ver todas las asignaturas de la carrera seleccionada.	- El usuario debe poder ver todas las asignaturas de la carrera seleccionada anteriormente. - El usuario debe poder hacer clic en alguna de las asignaturas para ver todos sus temas.
U-6	Como usuario, quiero poder ver todos los temas de la asignatura seleccionada.	- El usuario debe poder ver todos los temas de la asignatura seleccionada anteriormente. - El usuario debe poder hacer clic en algún tema para ver todos sus ejercicios.

U-7	Como usuario, quiero poder ver todos los ejercicios del tema seleccionado.	- El usuario debe poder ver todos los problemas del tema seleccionado anteriormente. - El usuario debe poder hacer clic en un ejercicio para ver toda su información y poder realizarlo.
UR-1	Como estudiante, quiero poder iniciar sesión en la aplicación para acceder a mis datos personales y funcionalidades exclusivas.	- El estudiante, aun sin iniciar sesión, tiene que poder acceder al formulario de inicio de sesión con campos de correo electrónico y contraseña. - Las credenciales introducidas serán validadas con la base de datos de usuarios registrados. - El usuario debe poder ver un mensaje de error si se ha equivocado. - Al ingresar credenciales válidas, será redirigido y tendrá acceso a todas las funciones que requieren autentificación.
UR-2	Como estudiante, quiero cerrar sesión en la plataforma para asegurar que mi cuenta no sea accesible por otros después de terminar mi sesión.	- El estudiante debe poder acceder al botón de cerrar sesión desde cualquier página de la aplicación web. - Al seleccionar el botón, el usuario será automáticamente desconectado del sistema. - Como confirmación, se redirigirá al formulario de inicio de sesión. - El cierre de sesión eliminará todas las sesiones activas y limpiará los datos temporales para prevenir el acceso no autorizado.
UR-3	Como estudiante, quiero poder ver mi perfil para revisar y	- El estudiante debe poder acceder a su perfil desde cualquier página de la aplicación web.

	gestionar mi información personal y mi progreso académico.	- El perfil tiene que enseñar la información facilitada por el estudiante durante el registro, como el nombre, apellidos, correo electrónico y cualquier otra información importante. - Un usuario no registrado no tendrá la opción de poder acceder a su perfil.
UR -4	Como estudiante, quiero poder realizar un ejercicio para practicar y mejorar mis habilidades en una materia específica.	- El estudiante debe poder acceder a cualquier ejercicio disponible de la aplicación web. - El estudiante debe poder contestar el ejercicio (si tiene el ejercicio es de ese tipo) y ver la solución correcta. - Un usuario no registrado no tendrá la opción de contestar el ejercicio.
UR-5	Como estudiante, quiero poder ver la solución completa para entender mejor un ejercicio concreto.	- El estudiante debe poder ver la solución completa y el paso a paso del ejercicio siempre y cuando tenga un *Mathy* disponible. - Un usuario no registrado no tendrá la opción de ver la solución completa.
UR-6	Como estudiante, quiero ver mis estadísticas para ver mi historial completo de ejercicios.	- El estudiante debe poder acceder a sus estadísticas desde cualquier página de la aplicación web. - La página de las estadísticas tiene que enseñar una gráfica con los ejercicios realizados, los correctos y los incorrectos. - Un usuario no registrado no tendrá la opción de ver sus estadísticas.
UR -7	Como estudiante, quiero ver el ranking general de la	- El estudiante debe poder acceder al ranking desde cualquier página de la aplicación web.

	plataforma para comparar mi rendimiento con el de otros estudiantes.	- El estudiante debe poder ver el ranking de los mejores estudiantes de la web y ver su clasificación. - Un usuario no registrado no tendrá acceso al ranking general.
UR-8	Como estudiante, quiero ser premium para obtener los beneficios que conlleva serlo.	- El estudiante debe poder acceder a la información del premium desde cualquier página de la aplicación web. - El estudiante debe poder acceder a la página de pago de Stripe y así rellenar todos los datos para completar el pago. - Una vez que todo haya ido correctamente, el estudiante recibirá todos los beneficios por ser premium. - Un usuario no registrado tendrá acceso a la información del premium, pero no podrá convertirse.
UR-9	Como estudiante, quiero poder tener acceso a un chat con una inteligencia artificial para obtener respuestas inmediatas a mis preguntas académicas.	- El estudiante debe poder acceder a los problemas donde se muestra un chat con una inteligencia artificial. - El Chat debe poder responder a todas las preguntas de los estudiantes de forma rápida y eficaz.
UR-10	Como estudiante, quiero cancelar mi suscripción premium para detener los pagos mensuales.	- El estudiante debe poder acceder a su perfil desde cualquier página de la aplicación web. - El estudiante puede cancelar su suscripción y suspender los pagos mensuales en cualquier momento.
UR-11	Como estudiante me	- El estudiante debe poder agregar o eliminar

	gustaría agregar o eliminar ejercicios favoritos para poder acceder a ellos fácilmente en el futuro.	ejercicios de la lista de favoritos desde la lista de ejercicios de un tema, desde dentro del propio ejercicio o desde la misma lista de favoritos. - Un usuario no registrado no tendrá acceso a realizar estas operaciones.
UR-12	Como estudiante, quiero ver todos mis ejercicios favoritos para poder acceder a ellos rápidamente.	- El estudiante debe poder acceder a su lista de ejercicios favoritos desde cualquier página de la aplicación web. - Un usuario no registrado no tendrá acceso a su lista de favoritos.
UR-13	Como estudiante, me gustaría cambiar mi contraseña para proteger mi cuenta y mi información personal.	- El estudiante debe poder acceder a su perfil desde cualquier página de la aplicación web. - En el apartado de “cambiar su contraseña”, el estudiante deberá introducir la contraseña actual y después la contraseña nueva. - El estudiante debe crear una contraseña que cumpla una longitud mínima de ocho caracteres y contenga al menos un número, una letra minúscula, una mayúscula y un carácter especial. - Una vez se haya confirmado, la nueva contraseña se habrá guardado en la base de datos y ya se podrá dar uso. - Un usuario no registrado no tendrá acceso a su perfil.
UR-14	Como estudiante, quiero eliminar mi cuenta para dejar de utilizar los servicios y asegurar que toda mi	- El estudiante debe poder acceder a su perfil desde cualquier página de la aplicación web. - El estudiante tiene que poder eliminar su cuenta y que sus datos sean borrados de la

	información personal sea borrada de la plataforma.	base de datos. - Un usuario no registrado no tendrá acceso a su perfil.

Figura 3. Backlog de MathCampus.

3.4 Implementación del Kanban

Una vez acabado el Backlog, creamos un sistema Kanban para gestionar el flujo del trabajo de desarrollo. Un tablero Kanban permite visualizar nuestro trabajo de manera general, limita la cantidad de trabajo y maximiza la eficiencia.

El tablero está dividido en cuatro: “Backlog”, “Doing”, “Testing” y “Done”. Cada user story del Backlog se transformó en una tarjeta Kanban que se colocó en la columna correspondiente según su estado actual. Como podemos observar en la **Figura 4,** se ha usado la aplicación *Trello* para crear el tablero Kanban.

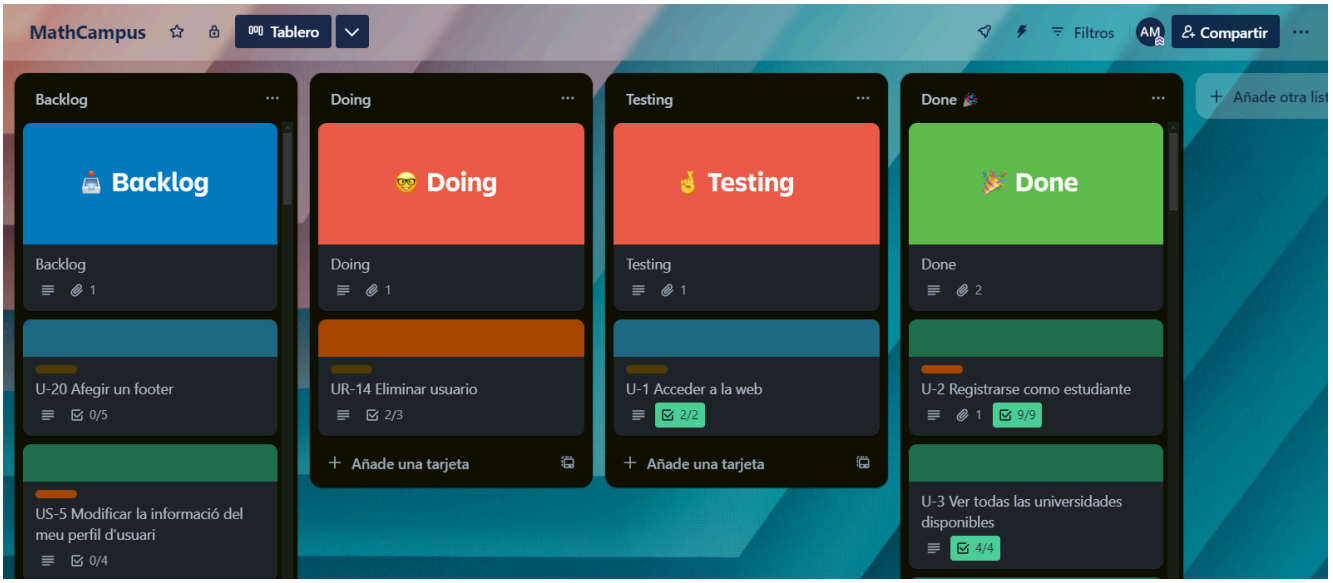

Figura 4. Kanban de MathCampus en *Trello*.

Se han ido celebrando reuniones periódicas vía *Zoom* con el tutor para evaluar el desarrollo y el progreso del trabajo. Estas reuniones han permitido ir reevaluando la priorización de las tareas y también la incorporación de nuevos requerimientos que han ido surgiendo durante el tiempo.

De este modo, nuestro tablero Kanban ha ido experimentando cambios constantemente durante el proceso de desarrollo para adaptarse a las nuevas necesidades que van apareciendo y garantizar que todo siga encaminado hacia los objetivos establecidos al inicio del proyecto.

4. Diseño

En esta sección se analiza el diseño de nuestro proyecto. Primero, se analizará la arquitectura que hemos utilizado en la aplicación acompañada de una breve descripción de sus servicios. A continuación, introduciremos el diseño y la estructura de la base de datos, en la que se incluye la creación del esquema de la base de datos, el desarrollo del diagrama entidad-relación y la definición de las tablas. Finalmente, se describirán las tecnologías utilizadas en este trabajo y la justificación de su uso.

Por lo tanto, una correcta elección de tecnologías a utilizar, sumada a un diseño y estructura de base de datos adecuados, garantizarán una implementación eficaz y robusta del sistema, asegurando que el producto final cumpla con las expectativas esperadas.

4.1 Arquitectura de la aplicación

En la figura que tenemos abajo (**Figura 5)**, se puede observar la arquitectura creada para la aplicación web, así como sus servicios y la base de datos que se ha utilizado.

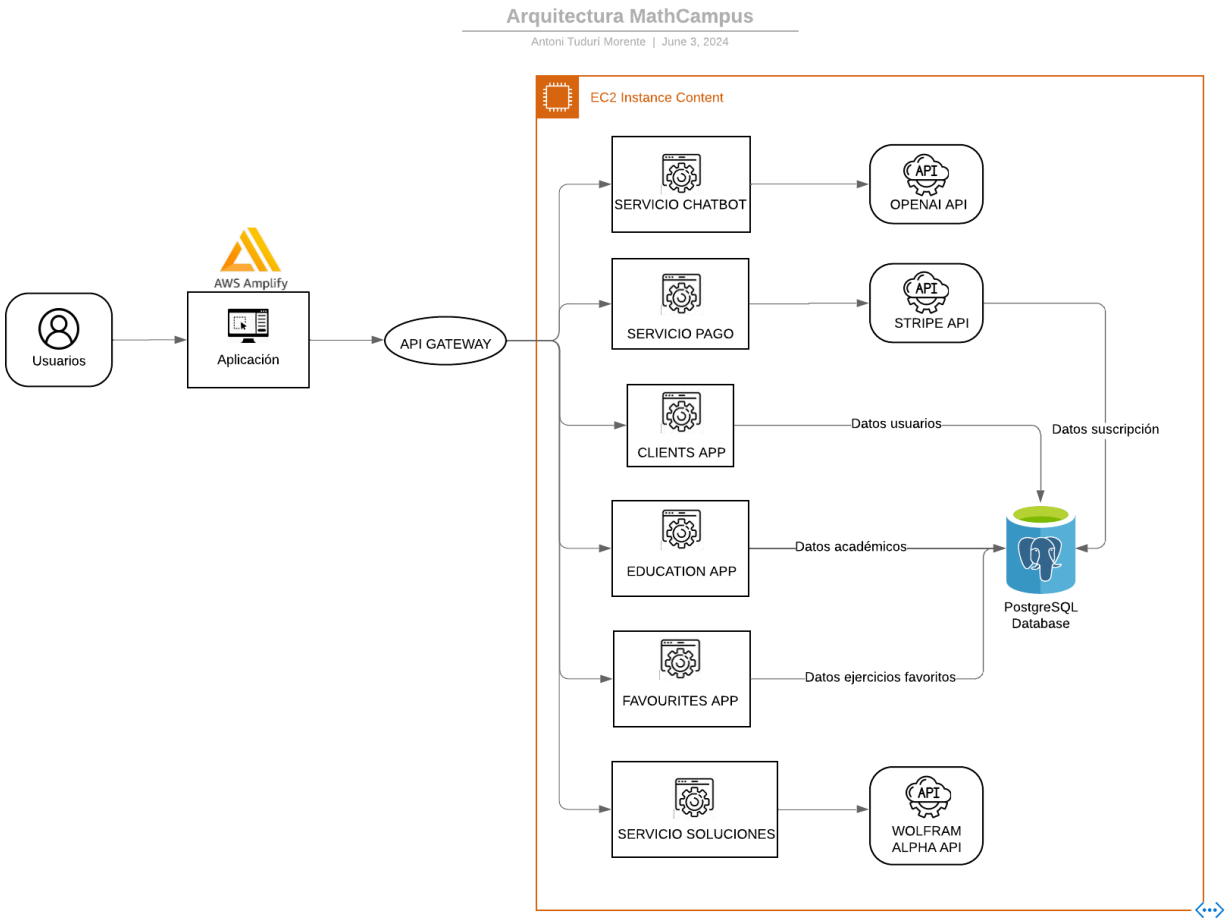

Figura 5. Arquitectura MathCampus.

Como podemos observar, los usuarios interactúan con la página web alojada en *AWS Amplify*, la cual solicita información a través de la API Gateway, la cual se comunica con diferentes servicios.

En el servidor back-end *EC2*, se integran tres servicios que utilizan APIs externas:

- OPENAI API: Esta API nos ayuda a desarrollar y operar el chatbot, que brinda a los usuarios respuestas inteligentes y precisas sobre las consultas de estos.
- STRIPE API: Esta API gestiona el sistema de pagos y suscripciones, facilita transacciones eficientes y seguras y guarda información importante en la base de datos PostgreSQL.

- WOLFRAM ALPHA API: Esta API se utiliza para obtener soluciones de ejercicios paso a paso correctas y detalladas.

Además de estos tres servicios externos mencionados anteriormente, nuestra aplicación también utiliza tres aplicaciones desarrolladas en Django:

- CLIENTS APP: Esta aplicación gestiona toda la información relacionada con los usuarios. Toda esta información es fundamental para brindar a cada usuario una experiencia personalizada y segura.
- EDUCATION APP: Esta aplicación contiene toda la información académica, incluidas las universidades, sus especialidades, asignaturas, temas y ejercicios. Además de esto, se registran los intentos realizados por el usuario en cada ejercicio, facilitando la creación de estadísticas.
- FAVOURITES APP: Esta aplicación permite a los usuarios gestionar sus ejercicios favoritos. Los usuarios pueden guardar ejercicios específicos para acceder rápidamente en un futuro, mejorando así la usabilidad.

Estas tres aplicaciones en Django funcionan de manera integrada con la base de datos PostgreSQL para administrar la información de los usuarios, los recursos educativos y las preferencias personales. Cada aplicación interactúa con diferentes tablas, asegurando que los datos se almacenen de manera organizada y accesible.

La sección **4.2 Diseño y estructuración de la base de datos relacional** describe en detalle la estructura de la base de datos y sus tablas, explicando cómo se han diseñado las relaciones entre tablas, los tipos de datos utilizados, etc.

4.2 Diseño y estructuración de la base de datos relacional

4.2.1 PostgreSQL

PostgreSQL es un potente sistema de gestión de bases de datos relacional ampliamente utilizado y de código abierto. Se destaca por su robustez, escalabilidad y cumplimiento de los estándares SQL, y ofrece muchas características avanzadas:

- Transacciones ACID (Atomicidad, Consistencias, Aislamiento, Durabilidad).
- Subconsultas.

- Consultas complejas.
- Claves foráneas, koins, vistas, triggers y procedimientos almacenados.

4.2.2 Diagrama Entidad-Relación y definición de tablas

Esta sección presenta nuestro diagrama de relaciones entre entidades (**Figura 6**). Esto es clave para visualizar la estructura y las relaciones de la base de datos. Este diagrama detalla las tablas, sus campos y cómo se conectan entre sí, proporcionando una visión clara y organizada de la estructura de datos.

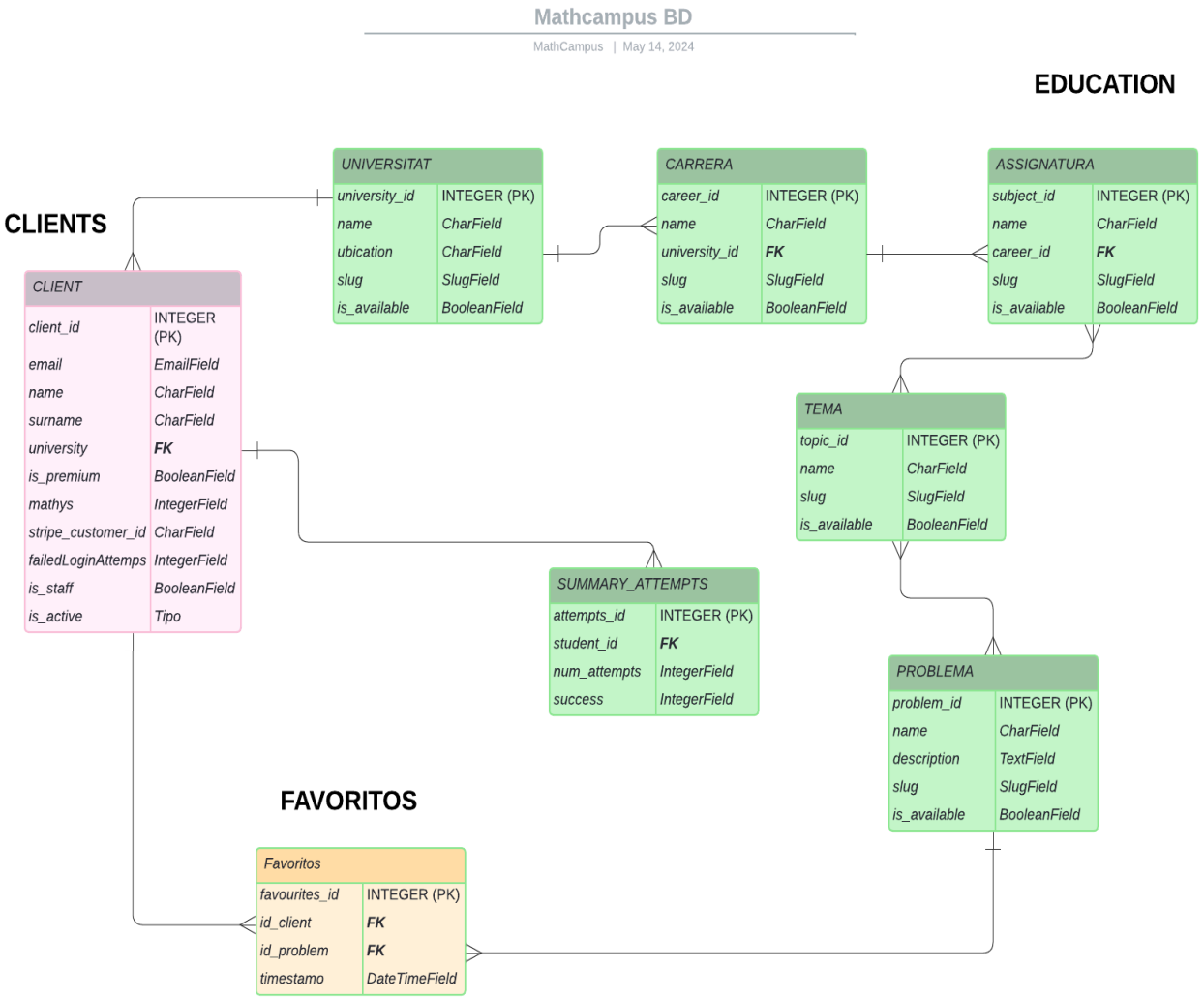

Figura 6. Diagrama Entidad-Relación BD.

Este diseño tiene como objetivo manejar eficientemente y de manera organizada la información relativa a los clientes, las universidades, los grados ofertados y la interacción de los clientes con todo el temario educativo. Para ello, se ha puesto el principal foco en las relaciones entre entidades del módulo *education* con el fin de obtener una mejor optimización y escalabilidad.

La relación entre las entidades *universidad* y *carrera* es de uno a muchos, en la cual cada carrera está vinculada a una única universidad gracias a la llave foránea *university_id*. Esto ocurre también con la relación entre *carrera* y *asignatura*, ya que una carrera puede tener varias asignaturas.

Sin embargo, la relación entre *asignatura* y *tema* es de muchos a muchos, permitiendo que varios temas puedan ser asociados con múltiples asignaturas. Este tipo de relación es ideal dado que el plan docente de una asignatura a menudo puede cubrir varios temas y un tema específico puede ser importante para más de una asignatura. De manera similar, la relación muchos a muchos entre tema y problema permite la asociación de varios problemas con varios temas. Gracias a esta relación, un problema específico puede ser aplicado a varios temas sin la necesidad de duplicarlo, optimizando los recursos y simplificando la administración de contenido.

Gracias a este diseño, se obtienen beneficios muy significativos en términos de escalabilidad y optimización.

Escalabilidad

- Relaciones flexibles. Las relaciones entre *asignatura* y *tema*, y entre *tema* y *problema*, permiten una fácil adaptación
- Modelos de cascada. El uso de las claves foráneas con la opción de "*on_delete=models.CASCADE*" asegura la eliminación automática de registros en tablas padre, eliminando cualquier dato dependiente, facilitando la gestión de datos y manteniendo la integridad de la base de datos a la vez que crece.
- Modularidad de los modelos. Todas las entidades se manejan de manera independiente. Por lo tanto, permite escalar una parte del sistema sin afectar a otras.

Optimización

- Generación automática de *Slugs*. Gracias al uso de *slugs*, la experiencia del usuario es mejorada, ya que se proporcionan URLs legibles. Además, el rendimiento de búsquedas y las consultas a la base de datos son optimizados.
- Automatización en la gestión de datos. En las entidades *universidad*, *carrera*, *asignatura*, *tema* y *problema*, se introduce un método *save* para generar slugs automáticamente, teniendo en cuenta los duplicados. Gracias a ello, se reducen los errores y mejoramos la eficiencia.

- Manejo de la disponibilidad. En cada entidad en el sistema de educación existe la propiedad *is_available*, la cual permite controlar la visibilidad de los elementos en la interfaz del usuario sin tener la necesidad de eliminar registros en la base de datos. Gracias a esta práctica, se optimiza la gestión de los datos y se reduce la carga en las operaciones de mantenimiento.

Entidades

Las entidades de la **Figura 6** han sido diseñadas para asegurar que los datos puedan ser almacenados, recuperados y administrados de manera eficaz.

Tabla *CLIENT*:

- Descripción: es la tabla principal, ya que almacena los usuarios que tienen acceso a la aplicación web. Mantiene registros de los usuarios del sistema, incluyendo sus datos personales y de acceso. Las columnas *is_premium* e *is_staff* nos dirán cuál es el rol del usuario y si tiene permisos.
- Campos clave:
    - *ID*: identificador único para cada usuario.
    - *email, name, username, university*: información personal y de contacto del usuario.
    - *is_premium, stripe_customer_id*: datos que ayudarán para las funcionalidades premium.

Tabla *UNIVERSIDAD*:

- Descripción: tabla que almacena las universidades en el sistema, incluyendo sus datos y disponibilidad. La clave principal es el ID de la universidad, que se incrementa automáticamente y la convierte en un índice, permitiéndonos acceder a la información más rápido.
- Campos clave:
    - *ID*: identificador único para cada universidad.
    - *name, ubication*: información de la universidad.
    - *slug:* permite identificar la universidad y así poder añadirla al enlace.

- *is_available*: indica si la universidad está disponible para los usuarios.

Tabla *CARRERA*:

- Descripción: tabla derivada de universidad que almacena las carreras de una universidad en concreto, incluyendo sus datos y su disponibilidad. Tiene una columna que es *universidad,* que almacena el ID de la universidad a la cual está ligada la carrera.
- Campos clave:
  - *ID*: identificador único para cada carrera.
  - *university_id*: clave foránea que enlaza con la *UNIVERSIDAD*.

Tabla *ASIGNATURA:*

- Descripción: tabla que derriba de carrera y almacena las asignaturas de una carrera en concreto, incluyendo sus datos y su disponibilidad. Tiene una columna que es *carrera,* que almacena el ID de la carrera a la cual está ligada la asignatura.
- Campos clave:
  - *ID*: identificador único para cada asignatura.
  - *career_id:* clave foránea que enlaza con *CARRERA*.

Tabla *TEMA*:

- Descripción: tabla que almacena los temas disponibles en el sistema. Incorpora una relación muchos a muchos con la tabla *ASIGNATURA*, lo que permite asociar el mismo tema con varios diferentes, facilitando una organización flexible de los temas y una gestión eficiente.

Tabla *PROBLEMA:*

- Descripción: tabla que almacena los problemas disponibles en el sistema. Incorpora una relación de muchos a muchos con la tabla *TEMA.* Esto significa que cada problema puede estar relacionado con varios temas diferentes y cada tema puede contener varios problemas. Esto permite organizar los problemas de manera flexible y dinámica según el tema.

Tabla *SUMMARY_ATTEMPTS:*

- Descripción: tabla que almacena un resumen de los intentos del estudiante en varios ejercicios. El propósito es brindar una visión general del rendimiento de los estudiantes a lo largo del tiempo y facilitar el análisis y seguimiento del progreso académico.
- Campos clave:
    - *ID*: identificador único para cada intento.
    - *student_id*: clave foránea que enlaza con el *CLIENT*. identifica al estudiante que realizó los intentos.
    - *num_attempts*: registra el número total de intentos realizados por el estudiante.
    - *success*: cuenta el número de intentos que resultaron en respuestas correctas.

Tabla *FAVORITOS*:

- Descripción: tabla que almacena y gestiona los problemas favoritos seleccionados por los usuarios. Su función principal es facilitar a los usuarios el acceso rápido a problemas específicos que desean visitar o trabajar más adelante.
- Campos clave.
    - *ID*: identificador único para cada lista de favoritos.
    - *id_client*: clave foránea que enlaza con *CLIENT*. Identifica al usuario que ha marcado el problema como favorito.
    - *id_problem*: clave foránea que enlaza con *PROBLEMA*. Identifica el problema específico que ha sido marcado como favorito por el usuario.
    - *timestamp*: campo de fecha que registra cuándo fue marcado el problema como favorito.

4.3 Tecnologías utilizadas

En este apartado, se expondrán las diferentes tecnologías utilizadas durante todo el proyecto, incluidos los diferentes lenguajes de programación, así como los frameworks y finalmente las plataformas que han sido de utilidad de principio a fin de este proyecto.

4.3.1 Lenguajes de programación

Los siguientes lenguajes que se exponen han permitido tener una base sólida para desarrollar aplicaciones web personalizadas y funcionales. Cada lenguaje tiene su propósito específico y hacer una combinación de ellos, permite crear aplicaciones web más enriquecedoras.

HTML

HTML [2] es el lenguaje de marcado estándar utilizado para crear la estructura básica de las páginas web. HTML es utilizado para estructurar todo el contenido de la aplicación, incluyendo textos, enlaces, imágenes y otros elementos multimedia. Resumidamente, es la base para la construcción de páginas web.

CSS

CSS [3] es un lenguaje de hojas de estilo que se utiliza para controlar la presentación, formato y el diseño de documentos HTML. Mediante normas y selecciones, CSS permite aplicar estilos como color, tipos de letra, dimensiones, márgenes y otras propiedades visuales a los elementos HTML. Gracias a ello, se puede crear una interfaz atractiva y profesional, asegurando que la aplicación no solo sea funcional sino también visualmente agradable.

JavaScript

JavaScript [4] es un lenguaje de programación que permite añadir interactividad a las páginas web. Una de las características más destacadas es el manejo de eventos, realizar operaciones en el lado del cliente y comunicar con el servidor sin necesidad de recargar la página, lo cual permite validar datos de entrada, realizar cambios en tiempo real, cargar el contenido de manera asíncrona e interactuar con el usuario en general.

Además, contiene una amplia gama de librerías y frameworks como *React*, que facilitan el desarrollo de aplicaciones web más complejas y potentes, ofreciendo una gran variedad de posibilidades y reduciendo el tiempo que se necesita para programar una página web.

Python

Python [5] es un lenguaje de programación de propósito general, interpretado y de alto nivel. Está diseñado para que el código sea legible y la sintaxis permita a los programadores expresar conceptos en menos líneas de código.

Una de las mayores fortalezas de Python es que proporciona una gran cantidad de librerías y módulos que cubren una amplia gama de áreas, como visualización de datos (*Matplotlib*), aprendizaje automático (*Scikit-learn*), análisis de datos (*NumPy*, *Pandas*) y la web (*Django*).

Las principales razones por las que elegí utilizar este lenguaje fueron su facilidad de uso y su velocidad de escritura de código. Además, es un lenguaje muy conocido y utilizado por grandes empresas, por lo que tiene una gran comunidad y es fácil encontrar tutoriales y ejemplos de código.

4.3.2 Frameworks

Los frameworks son conjuntos de herramientas y librerías que proporcionan una estructura para desarrollar aplicaciones de software. Facilitan el proceso de desarrollo al ofrecer estructuras predefinidas y componentes reutilizables, lo que permite al desarrollador centrarse en la lógica de la aplicación. En esta sección hablaremos sobre aquellos que hemos utilizado en MathCampus.

React

React [6] es una librería de JavaScript de código abierto centrada en la construcción de interfaces de usuario reutilizables. *React* fue desarrollada y es mantenida por Facebook, siendo conocida por su eficiencia y flexibilidad. Por ello, se trata de una de las librerías más populares utilizadas por grandes empresas como Netflix, Twitter, PayPal o Uber.

Una de las características más importantes es la reutilización de componentes. *React* está basado en componentes, lo que significa que las interfaces de usuario se dividen en pequeñas piezas independientes y reutilizables. Cada componente tiene su propia lógica e interfaz de usuario, lo que facilita su mantenimiento a lo largo del proyecto.

React utiliza JSX, una extensión sintáctica que combina JavaScript y HTML, para describir cómo debería verse la interfaz de usuario (componentes). Esto permite a los desarrolladores escribir código parecido a HTML para describir la interfaz de usuario, mientras que al mismo tiempo utilizan JavaScript para gestionar la lógica.

He decidido utilizar el framework de *React* porque lo he utilizado anteriormente durante la carrera, en la realización de proyectos de software. Además, *React* es utilizado en muchas empresas, por lo tanto, existe una gran demanda en el mercado laboral, lo que añade valor a las habilidades perfeccionadas durante este proyecto.

Django

Django [7] es un framework de desarrollo web de alto nivel para Python, diseñado para facilitar el desarrollo de sitios web complejos, conocido por su simplicidad, eficiencia y escalabilidad.

Django proporciona una estructura robusta y completa que facilita la creación de aplicaciones web sofisticadas. Viene con una variedad de funcionalidades incorporadas, como autenticación de usuarios, mapas de URL, plantillas, motor de administración, etc.

Ofrece la posibilidad de definir modelos de datos para representar las entidades de un sistema. En mi caso, me ha permitido definir el modelo de usuarios, universidades, ejercicios, etc. Además, el ORM (Object-Relational Mapping) integrado, permite a los desarrolladores trabajar con bases de datos de una manera intuitiva, utilizando Python para modelar sus bases de datos en lugar de SQL crudo. Esto simplifica el proceso de creación, recuperación, actualización y eliminación de datos (operaciones CRUD).

Django contiene un sistema para administrar automáticamente la interfaz de administración. Con unas pocas líneas de código, se puede obtener una interfaz de administración completa y personalizable para gestionar las entidades del sistema. Esto nos sirve mucho para crear universidades, carreras, etc., sin la necesidad de tocar la base de datos.

He utilizado *Django* porque *Python* es uno de los lenguajes que más domino ya que lo hemos utilizado mucho durante la carrera, a la vez que es uno de los lenguajes más demandados en el mundo laboral. Por ello, mejorar mis habilidades en *Python* durante este proyecto ha tenido mucho valor.

Tailwind CSS

Tailwind CSS [8] es un framework de CSS utilizado para desarrollar interfaces de usuario atractivas de manera simplificada y acelerada.

Tailwind proporciona clases de utilidad atómicas que permiten aplicar estilos específicos a elementos HTML. Esto permite construir interfaces de manera rápida y con gran control sobre el diseño sin tener que escribir CSS personalizado en otro archivo.

De esta manera, este framework ha sido utilizado durante el desarrollo del proyecto porque es uno de los más populares en front-end. En la carrera no lo hemos aprendido y durante estos meses de trabajo he aprendido esta nueva habilidad.

4.3.3 Plataformas

GitHub

GitHub es una plataforma de desarrollo que permite a los desarrolladores colaborar en proyectos utilizando el sistema de control de versiones *Git*. A lo largo del desarrollo de un proyecto, se producen una gran cantidad de modificaciones en el código fuente durante su implementación y despliegue, y el control de versiones *Git* registra cada uno de estos cambios. Cada actualización es registrada mediante un "commit", permitiendo a los desarrolladores la posibilidad de revisar el estado previo del código y, si es necesaria la modificación de este, revertir el estado. Esto es muy útil cuando existe más de un trabajador en el grupo de desarrollo y se modifica código a la vez.

Para realizar un uso correcto de las ramas de *Git* (**Figura 7)**, en el proyecto existe una rama "master" donde solamente se puede hacer "push" al final de una entrega y una rama "testing" que se actualiza constantemente con las nuevas funcionalidades. Ya que he trabajado solo en todo el proyecto, he estado trabajando directamente en la rama "testing" y no he tenido que crear una rama para cada *feature*.

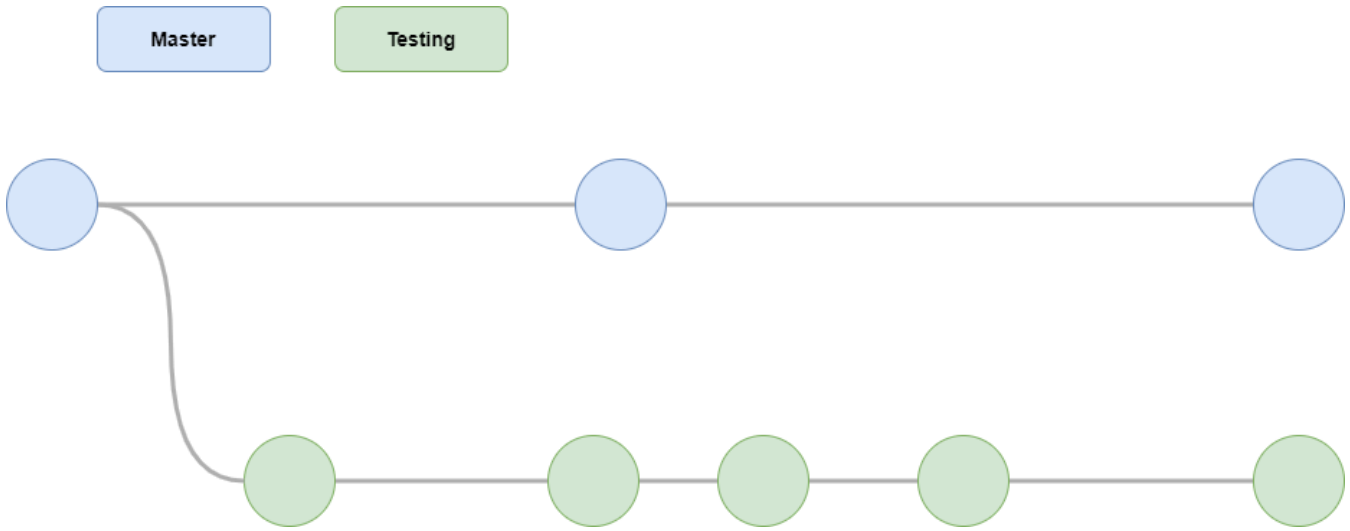

Figura 7. GitFlow utilizando dos ramas.

Postman

Postman [9] es una plataforma que ofrece diferentes herramientas que permiten construir, probar, documentar y compartir APIs. Destaca por permitir automatizar pruebas y ofrecer un entorno intuitivo para gestionar solicitudes y respuestas HTTP.

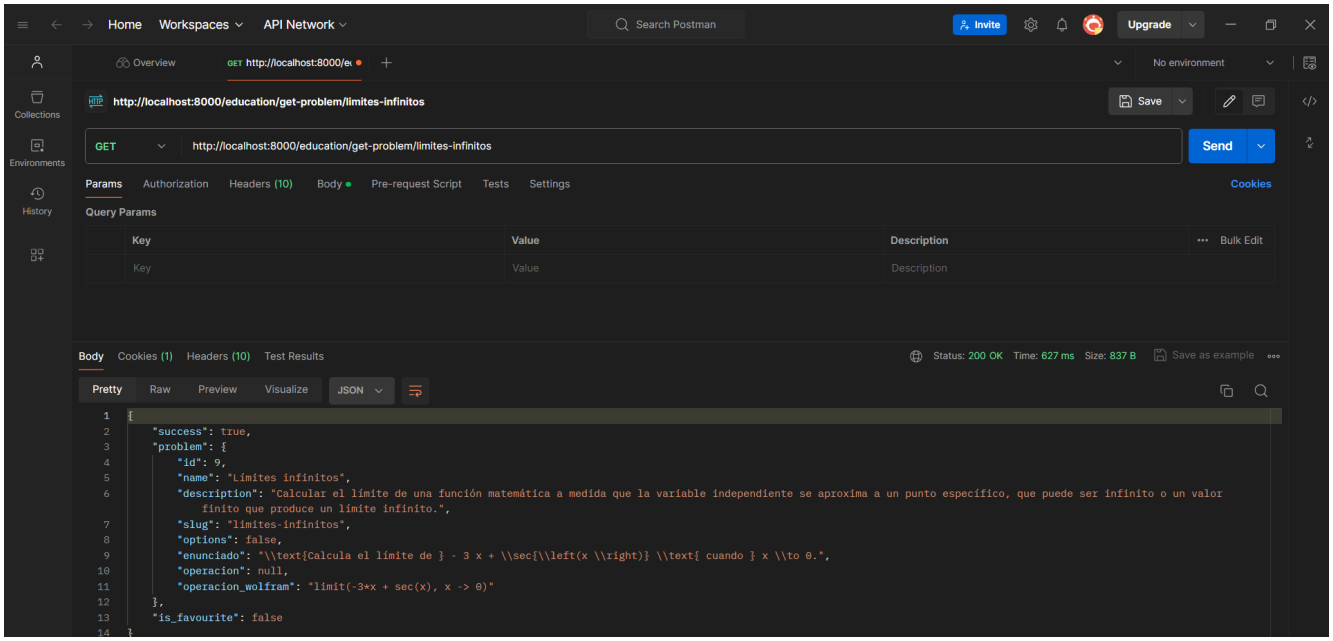

Figura 8. Gestión de solicitud GET en Postman.

pgAdmin 4

pgAdmin 4 [10] es una herramienta de gestión y desarrollo de bases de datos para PostgreSQL, que es uno de los sistemas de gestión de bases de datos relacionales de código abierto más avanzados y populares. Esta es la versión más reciente, ofreciendo una interfaz de usuario web intuitiva y receptiva. Esta aplicación lo que hace es facilitar la administración de las bases de datos PostgreSQL desde cualquier navegador web, eliminando la necesidad de tener descargada una aplicación en el ordenador, pudiendo acceder más fácilmente y gestionar la base de datos de manera más flexible.

Trello

Trello [22] es una herramienta visual que permite a los equipos gestionar cualquier tipo de proyecto y flujo de trabajo, así como supervisar tareas. Es importante para gestionar proyectos que utilizan el método Kanban para ayudar a organizar tareas y proyectos. En este caso, al trabajar solo, lo he utilizado únicamente para organizar las tareas y tener una visión general del proyecto. Anteriormente, se ha explicado cómo hemos hecho nuestro tablero Kanban y en la **Figura 4** se puede observar el tablero desde la aplicación *Trello*.

Stripe

Stripe [11] es una de las plataformas de gestión de pasarelas de pago más populares del mundo. Es una plataforma de pagos en línea que se incorpora en una página web y que permite a las empresas y autónomos gestionar los pagos por productos y servicios mediante su API. La integración de esta API en las aplicaciones web permite a los desarrolladores incorporar fácilmente capacidades de pago en sus aplicaciones y sitios web.

Stripe nos sirve para gestionar con seguridad y eficacia todo el proceso de pagos por Internet, ya que utiliza tecnologías avanzadas de encriptación y cumplimiento con los estándares PCI para proteger los datos de las tarjetas y la información financiera de los usuarios.

Finalmente, *Stripe* proporciona una plataforma con varias funcionalidades importantes y de gran interés:

- Pasarela de pago: integra el sistema de check out y permite a los clientes realizar el pago.
- Generación de facturas.
- Gestión de suscripciones.
- Creación de enlaces de pago.

Amazon Web Services

AWS [19], siglas de *Amazon Web Services*, es la plataforma de servicios en la nube más completa y ampliamente adoptada del mundo que ofrece una amplia gama de servicios y herramientas para computación, almacenamiento, bases de datos, redes, analítica, aprendizaje automático, seguridad, entre otros. Los servicios de *AWS* permiten a las empresas y desarrolladores construir, desplegar y gestionar aplicaciones y servicios a través de una red global de centros de datos gestionados por Amazon.

AWS ofrece varios beneficios [23] para la aplicación web, entre ellos:

- Flexibilidad: con *AWS* se tiene acceso a un entorno virtual que permite cargar el software y los servicios que necesita nuestra aplicación.
- Rentable: se afronta únicamente el costo de la potencia de cómputo, el almacenamiento y demás tipos de recurso.
- Facilidad de uso: diseñado para permitir a cualquier tipo de aplicación un hospedaje de una forma rápida y segura.

- Seguridad: enfoque integral para proteger y reforzar la infraestructura, incluidas medidas físicas, operativas y de software.
- Escalabilidad y alto desempeño: gracias a varias herramientas, la aplicación puede ampliarse o reducirse según la demanda.

5. Implementación y resultados

En el siguiente apartado, se procede a detallar minuciosamente la implementación del proyecto y a mostrar sus resultados finales. Primero, se muestra el diseño de página con todas las páginas de la interfaz de la aplicación web y se examina la estética y la estructura de esta. A continuación, aportamos un enfoque más empresarial, incluyendo el modelo de negocio y cómo se ha monetizado la aplicación web para afrontar los posibles gastos.

Posteriormente, se pondrá el foco en el desarrollo del sistema de autenticación y cómo ha sido implementado *JWT* para garantizar la seguridad y privacidad de los usuarios. Asimismo, revisamos el método de comunicación con los servicios externos, además de explicar cómo funcionan y cómo estos han sido integrados.

Seguidamente, describimos cómo hemos desarrollado la función de los ejercicios automatizados mediante el uso de scripts y las plantillas utilizadas.

Finalmente, se detalla cómo se ha realizado el despliegue del servidor y de la aplicación web en *Amazon Web Services*, incluyendo la configuración y los servicios utilizados.

5.1 Diseño de página

El objetivo principal del diseño de la aplicación es crear una interfaz fácil de usar que permita que los usuarios se adapten a una nueva plataforma. Mi objetivo principal ha sido brindar al usuario una experiencia ágil y fácil de manejar, siguiendo los principios de diseño utilizados en las aplicaciones líderes del mercado.

A continuación, se irá exponiendo de forma detallada cada página o ventana que tiene la aplicación y su trasfondo (**Figura 9**):

Figura 9. Página de inicio.

Al acceder al sitio web, lo primero que se encuentra es la página de bienvenida. Esta página define brevemente lo que es la aplicación web y da opción a registrarse o tener más información de esta. Si estamos en esta página y ya estamos con la sesión iniciada, nos dará la opción de empezar directamente a hacer ejercicios.

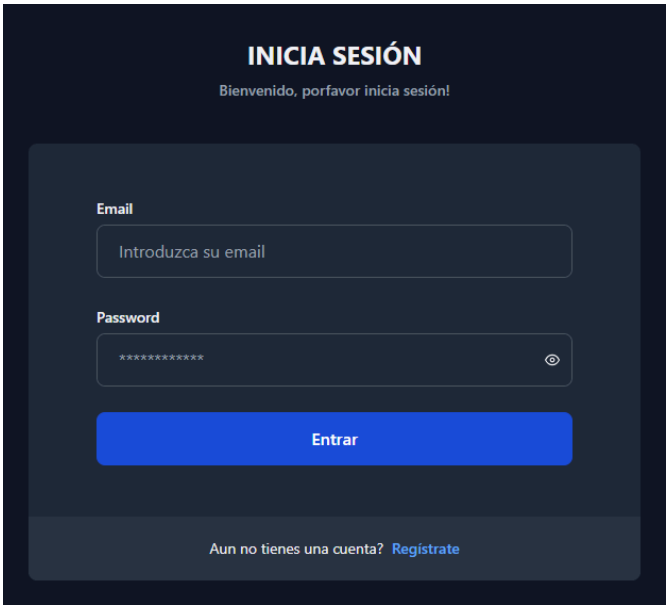

Figura 10. Inicio de sesión.

Cuando ya estamos dentro de la aplicación web, una de las primeras funciones que se encuentran es la página para iniciar sesión (**Figura 10**). Lograr esta interfaz es sencillo, con dos campos de entrada: uno para introducir el email con el que se ha registrado y otro para la contraseña. Además, el usuario dispone de un botón que permite iniciar sesión y justo debajo está la opción de dirigirse a la página de registro en caso de que no se tenga una cuenta de usuario creada.

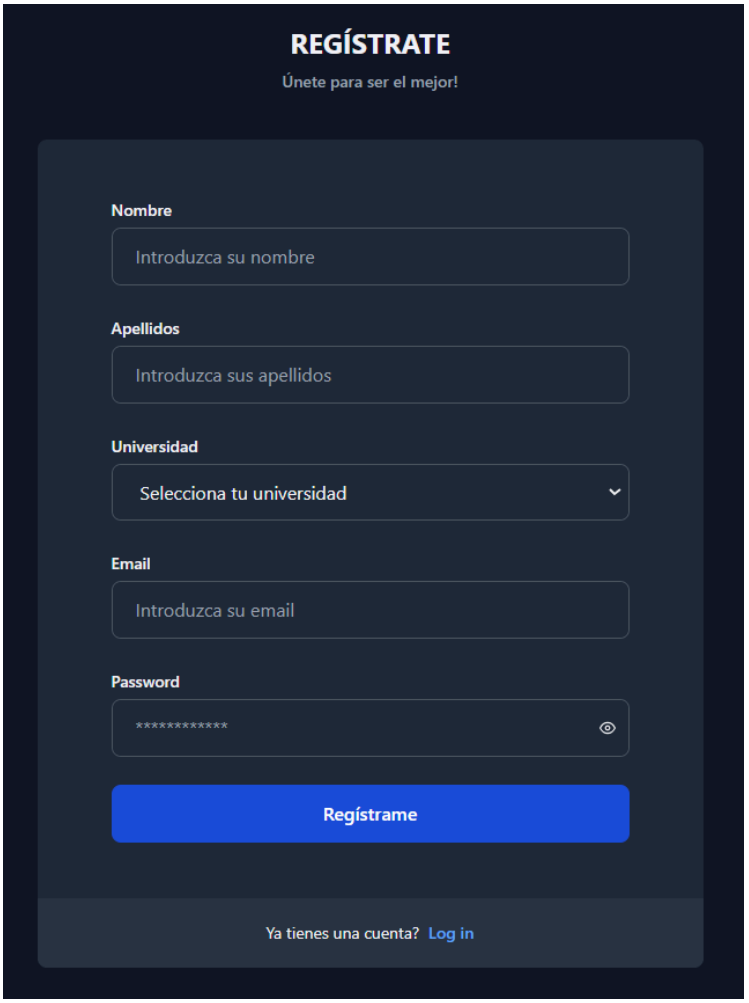

Figura 11. Registro.

La página de registro, **Figura 11**, es una interfaz similar a la de inicio de sesión, pero con más campos de entrada. Todos estos campos son importantes para la creación de un usuario nuevo: nombre, apellidos, universidad, mail y contraseña. Durante el registro, se proporciona una pequeña ayuda para cada campo a través de indicaciones que orientan al usuario sobre cómo completar correctamente el registro. Por lo tanto, esta guía asegura que todos los datos necesarios sean introducidos de manera adecuada y segura **(Figura 12)**.

Además de los campos, aparecen dos botones: el botón “Regístrame”, que llama al back-end para crear un nuevo usuario, y el link “Log in”, que nos redirige a la página de inicio de sesión.

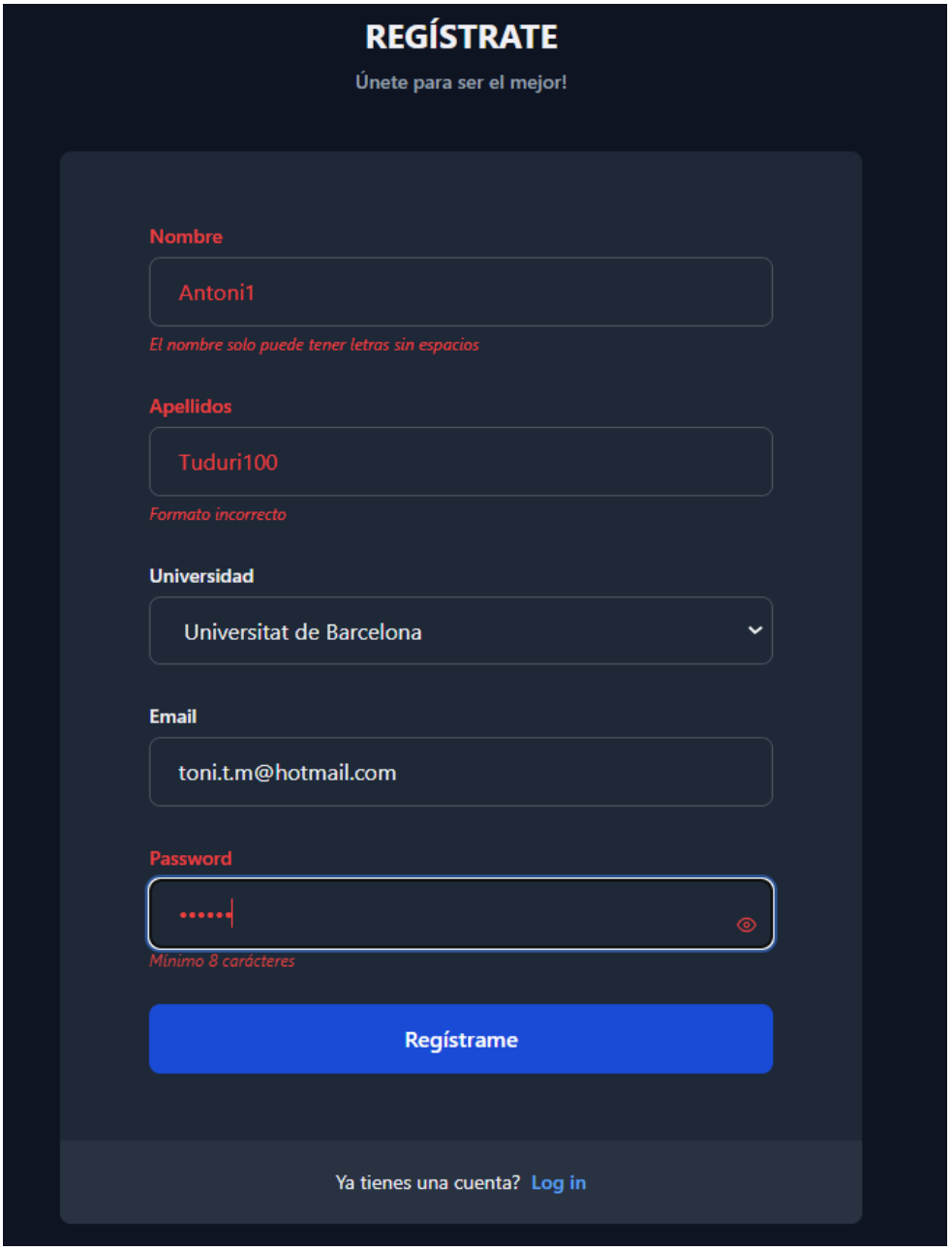

Figura 12. Registro con la guía

Después de iniciar sesión, la aplicación nos redirige automáticamente a la universidad que seleccionamos durante el proceso de registro **(Figura 13)**. Si en el momento del registro indicamos que no pertenecemos a ninguna universidad o que no aparecía en la base de datos, se nos redirigirá a la página principal con todas las universidades disponibles **(Figura 14)**. Cabe destacar que estas páginas permiten la navegación de todos los usuarios, tanto a

registrados como a no registrados. Por lo tanto, la información general de las universidades, sus programas académicos, asignaturas, temas y ejercicios pueden ser vistos por todo tipo de usuario.

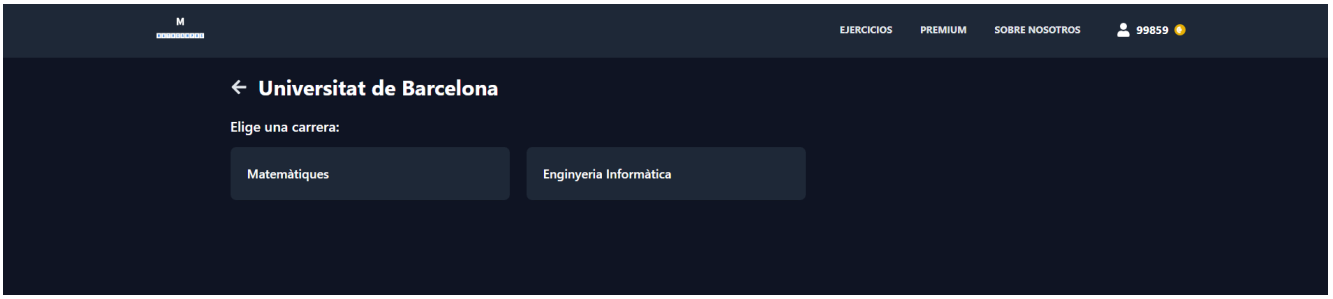

Figura 13. Página de una universidad vista como usuario registrado

Figura 14. Página principal vista como usuario no registrado

En la página principal **(Figura 14)**, podemos observar tres tarjetas (*cards*), todas ellas con el mismo formato: nombre de la universidad y la ubicación justo debajo. Refiriéndonos al diseño, la página principal es caracterizada por la simplicidad y la ausencia de elementos y colores extravagantes. Existe una tarjeta para cada universidad disponible en la base de datos, ofreciendo así una interfaz limpia y organizada que facilita a los usuarios la identificación y selección rápida. Además, estas tarjetas actúan como enlaces directos a páginas donde hay más información específica de las universidades, como los grados universitarios **(Figura 13)**.

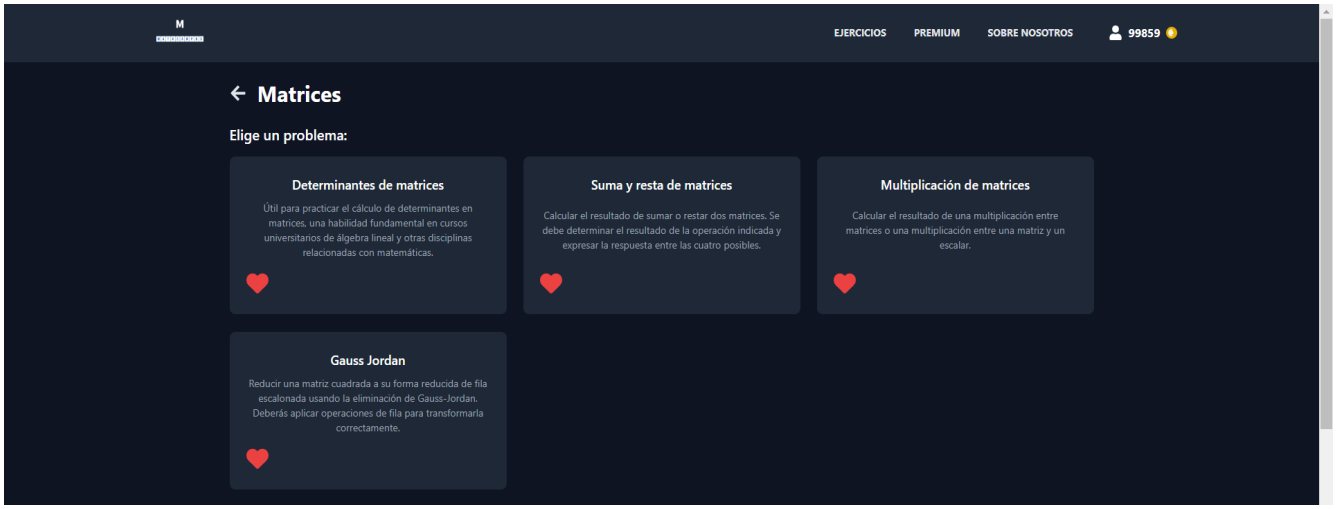

Figura 15. Lista de problemas de un tema

De este modo, el usuario puede navegar entre universidades, carreras, asignaturas, temas, y finalmente llegar a los ejercicios, como se muestra en la **Figura 15**. En esta lista, el diseño es muy parecido al de las tarjetas anteriormente descritas: nombre del problema en letra grande en la parte superior de la tarjeta, seguido por una breve explicación en un tamaño de letra más pequeño. Cabe destacar que en cada tarjeta de problema incluye un icono de corazón, visible solo para usuarios registrados y logueados. Este icono sirve para añadir los ejercicios a la lista de favoritos del usuario. Si el ejercicio ya se encuentra en la lista, saldrá el corazón relleno **(Figura 16)**.

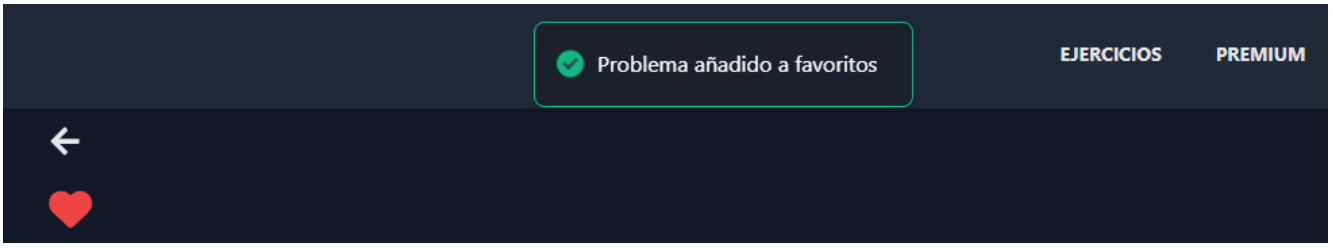

Figura 16. Aviso de ejercicio añadido a favoritos.

Cada tarjeta actúa como enlace directo hacia el problema en cuestión, llevando al usuario a la página específica del problema donde puede profundizar en los detalles o iniciar su resolución **(Figura 17)**.

Figura 17. Vista de un ejercicio con soluciones

Todos los ejercicios tienen una misma plantilla uniforme y se dividen en dos categorías: ejercicios con soluciones posibles y ejercicios sin soluciones definidas.

Ejercicios con soluciones

Cada ejercicio de esta categoría incluye el enunciado, la operación y las soluciones presentadas en Latex, utilizando para ello la librería Katex. Esto permite que el usuario pueda recargar los ejercicios que desee mediante el botón lila situado en la esquina inferior izquierda. El botón azul en la esquina inferior derecha se activa para revelar la solución completa después de que el usuario responda al ejercicio y si además posee más de un token Mathy. Como podemos ver en la **Figura 19**, al obtener la solución completa también obtenemos el paso a paso, información muy importante para que el estudiante comprenda el ejercicio.

Cuando el usuario selecciona una respuesta, hay dos posibles soluciones: que la respuesta sea correcta o incorrecta. En el segundo caso, se le mostrará la solución correcta. Esta interacción se registra en el servidor de manera eficiente para mantener actualizada la tabla de estadísticas del usuario.

Ejercicios sin soluciones

Como se ilustra en la **Figura 18**, los ejercicios sin soluciones funcionan de manera parecida a los ejercicios con soluciones, con la excepción de que, debido a la ausencia de soluciones

predefinidas, el botón de solución completa está siempre disponible y se puede activar siempre que el usuario tenga al menos una moneda virtual *Mathy*.

Hay que destacar que dentro del ejercicio, seguimos teniendo el icono del corazón que nos marca si el ejercicio está en la lista de favoritos o no, y si lo queremos añadir/eliminar de ella.

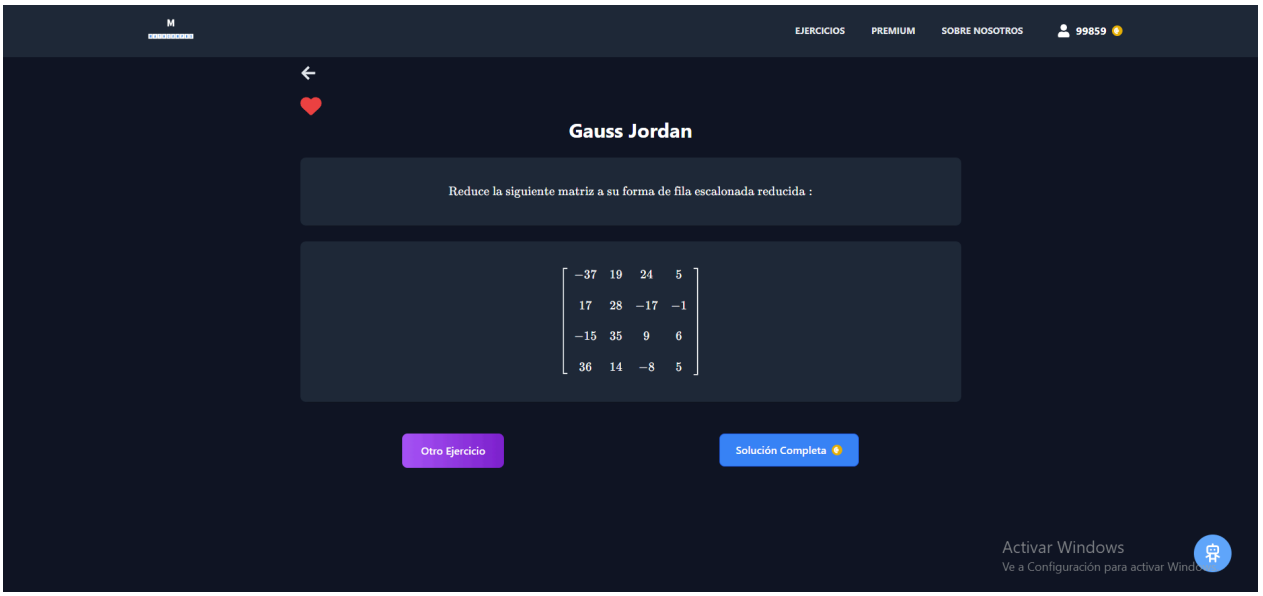

Figura 18. Vista de un ejercicio sin soluciones

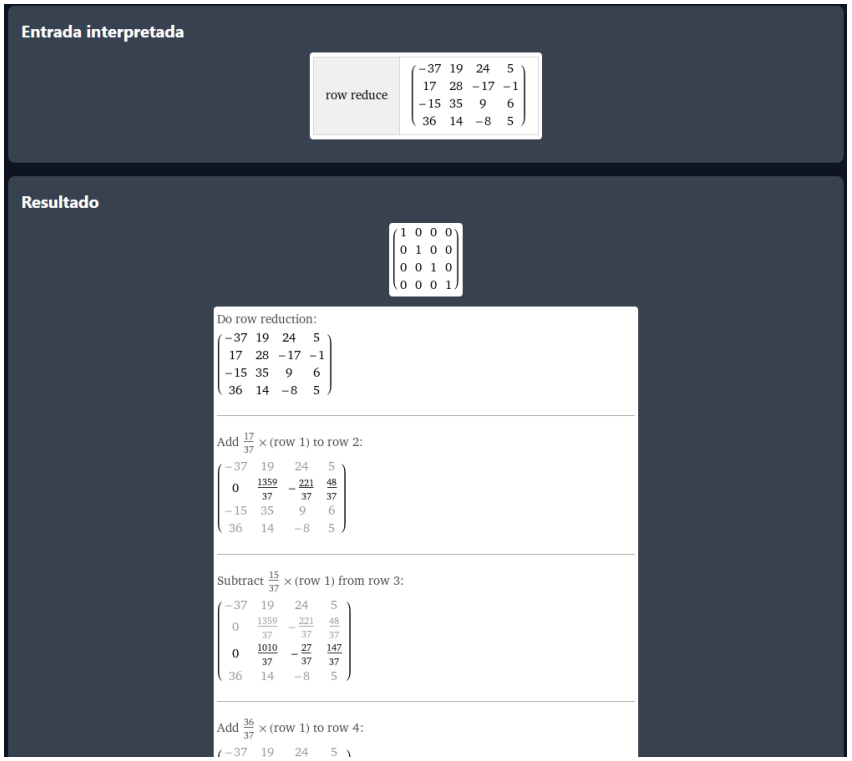

Figura 19. Parte de la solución completa

Finalmente, los usuarios pueden encontrar un botón flotante de color azul con el icono de un robot en la esquina inferior derecha de la página. La **Figura 20** muestra cómo este botón abre una ventana pequeña del chat con inteligencia artificial. Solamente los usuarios registrados pueden plantear preguntas y resolver dudas relacionadas con los ejercicios, sin embargo, además de estar registrado, también debe disponer de al menos una moneda virtual *Mathy*. Este sistema garantiza la disponibilidad y eficiencia del servicio de asistencia por IA para los usuarios que participan y utilizan activamente la plataforma.

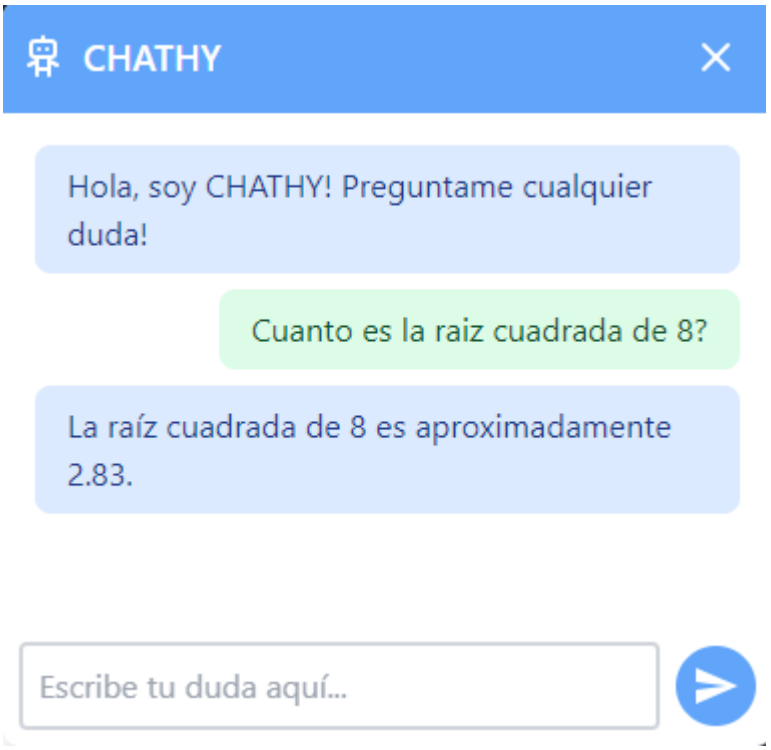

Figura 20. Chat con IA.

La barra de navegación, también conocida como navbar, la podemos encontrar en la parte superior de cualquier página y su contenido varía dependiendo del estado de sesión del usuario.

Si no estamos logueados **(Figura 21)**, tendrá botones para acceder a la información sobre las universidades, detalles sobre los beneficios del premium y sobre la aplicación en sí. Además, se ofrecen botones para iniciar sesión o registrarse, facilitando el acceso a funciones exclusivas para usuarios registrados.

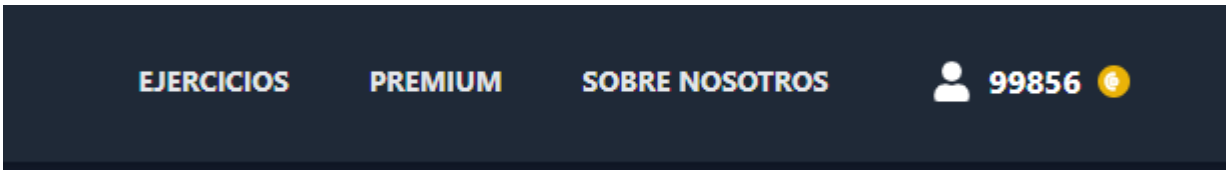

Figura 21. *Navbar* de usuario logueado

Si estamos logueados (**Figura 22)**, a los botones originales se les añade un menú desplegable (*dropdown*) que contiene cuatro opciones **(Figura 23)**:

1. Ver perfil: permite que los usuarios accedan a su perfil personal y vean y modifiquen sus datos.
2. Favoritos: proporciona un enlace directo a la lista de ejercicios que el usuario ha seleccionado como favoritos.
3. Estadísticas: muestra una página con contenido estadístico del usuario.
4. Cerrar sesión: opción segura para salir de la cuenta y volver a la página de inicio de sesión.

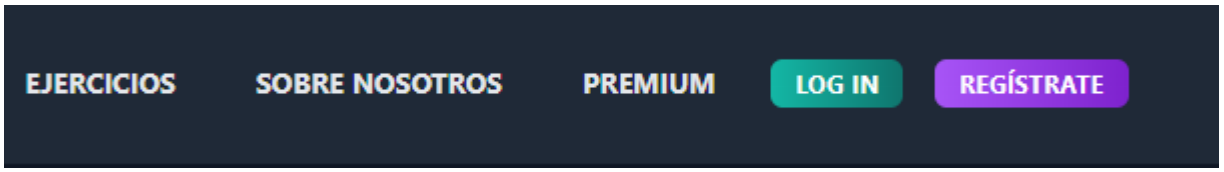

Figura 22. *Navbar* de usuario no logueado

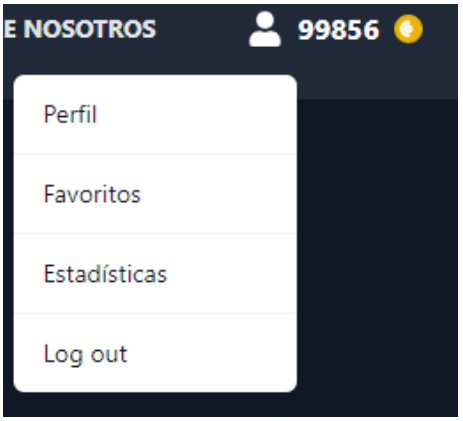

Figura 23. Botón *dropdown*.

Al seleccionar la opción “Perfil” del menú desplegable, se nos redirige a la página de perfil del usuario **(Figura 24)**. En esta página ofrece una vista detallada de toda la información que el usuario ingresó en el registro, además de mostrar la cantidad de tokens *Mathys* disponibles. Aquí, el usuario puede gestionar su información personal y visualizar su estado de suscripción actual.

Para aquellos usuarios que no estén suscritos al servicio premium, la página incluye un botón que les dirige a la página con toda la información del premium. Si el usuario ya es suscriptor premium, se presenta la opción de cancelar la suscripción **(Figura 25)**.

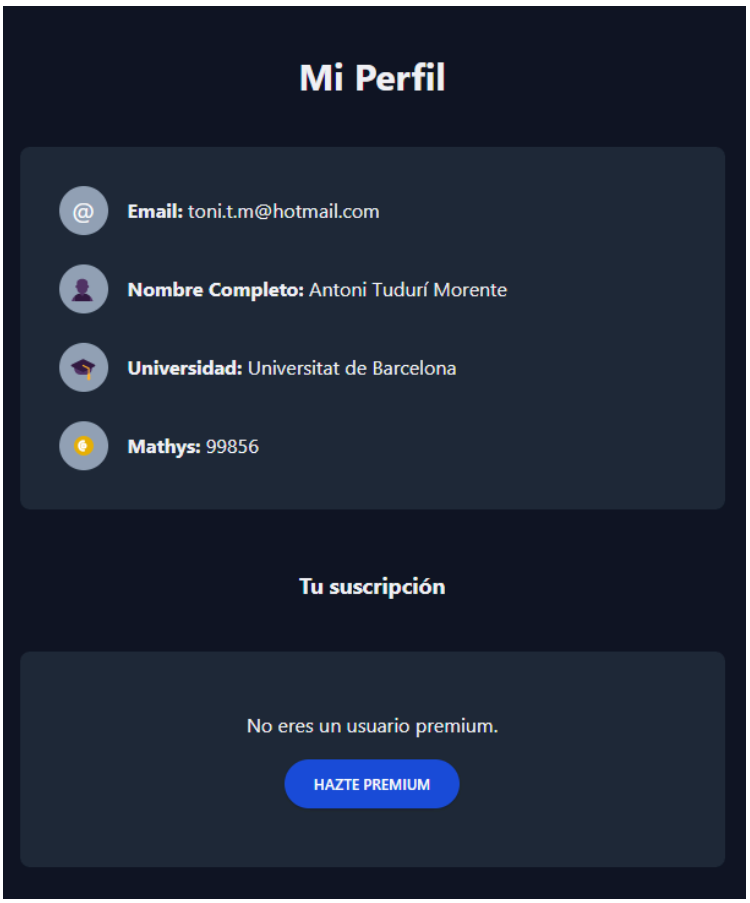

Figura 24. “Mi perfil” del usuario (1ª parte)

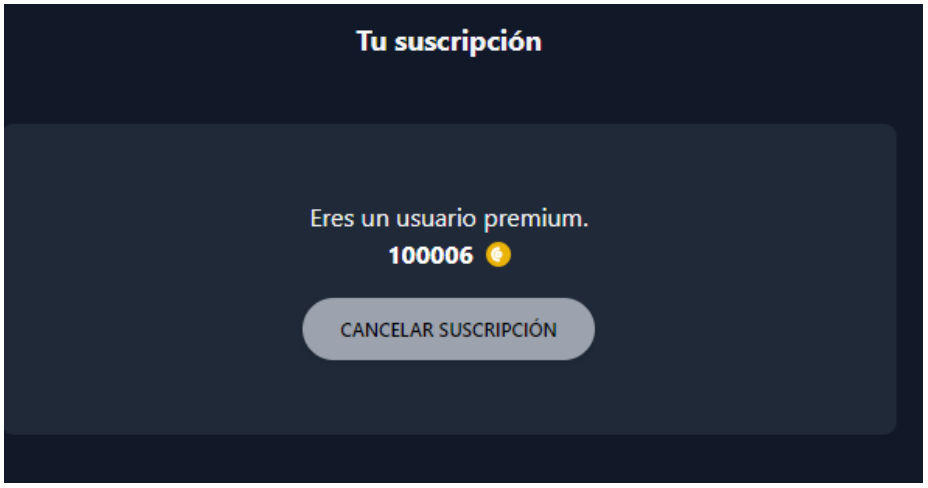

Figura 25. Suscripción premium

En esta misma página del perfil, el usuario tiene la opción de cambiar su contraseña actual **(Figura 26)**. Esta interfaz sigue un diseño similar al inicio de sesión explicado anteriormente. El usuario debe ingresar la contraseña actual e insertar una nueva contraseña. La nueva contraseña deberá cumplir con los mismos requisitos establecidos durante el proceso de registro para asegurar su fortaleza y seguridad.

Además, en la parte inferior de la página, habrá un botón grande y rojo que permitirá eliminar la cuenta. Una vez pulsado, aparecerá un panel de confirmación para asegurarse de que el usuario realmente desea proceder. Si se confirma, todos los datos asociados con el usuario serán borrados de la base de datos de manera permanente, eliminando así cualquier información personal y de actividad que haya sido registrada en la plataforma.

Figura 26. "Mi perfil" del usuario (2ª parte)

Al seleccionar la opción “Favoritos” del menú desplegable, el usuario accede a una página que muestra la lista de los ejercicios que ha marcado como favoritos **(Figura 27)**. Esta página se organiza igual que la lista de ejercicios de un tema específico, utilizando el mismo diseño de tarjetas, donde se muestra el título, la descripción y el icono de corazón, que indica si el ejercicio está actualmente en la lista de favoritos.

Desde esta página, los usuarios pueden gestionar fácilmente su lista de favoritos, añadiendo o eliminando ejercicios según sea necesario.

Figura 27. Lista de favoritos del usuario

Al seleccionar la opción “Estadísticas”, el usuario es dirigido a una página que presenta sus estadísticas principales junto con un ranking general, todo ello recuperado desde el back-end **(Figura 28)**. En la primera sección, el usuario puede visualizar su rendimiento. Esto incluye el número total de intentos realizados, el número de éxitos y su tasa de éxito

expresada en porcentaje. Para facilitar la interpretación de los datos, se proporciona un gráfico.

La segunda sección presenta un ranking mundial, enseñando a los 10 mejores estudiantes de la aplicación. Esta lista destaca a los usuarios con los mejores desempeños, incentivando una competencia saludable y motivación entre los participantes. Para aquellos usuarios que no están entre los diez primeros, se muestra su posición actual en el ranking general, permitiendo a cada uno conocer su standing en comparación con el conjunto más amplio de usuarios.

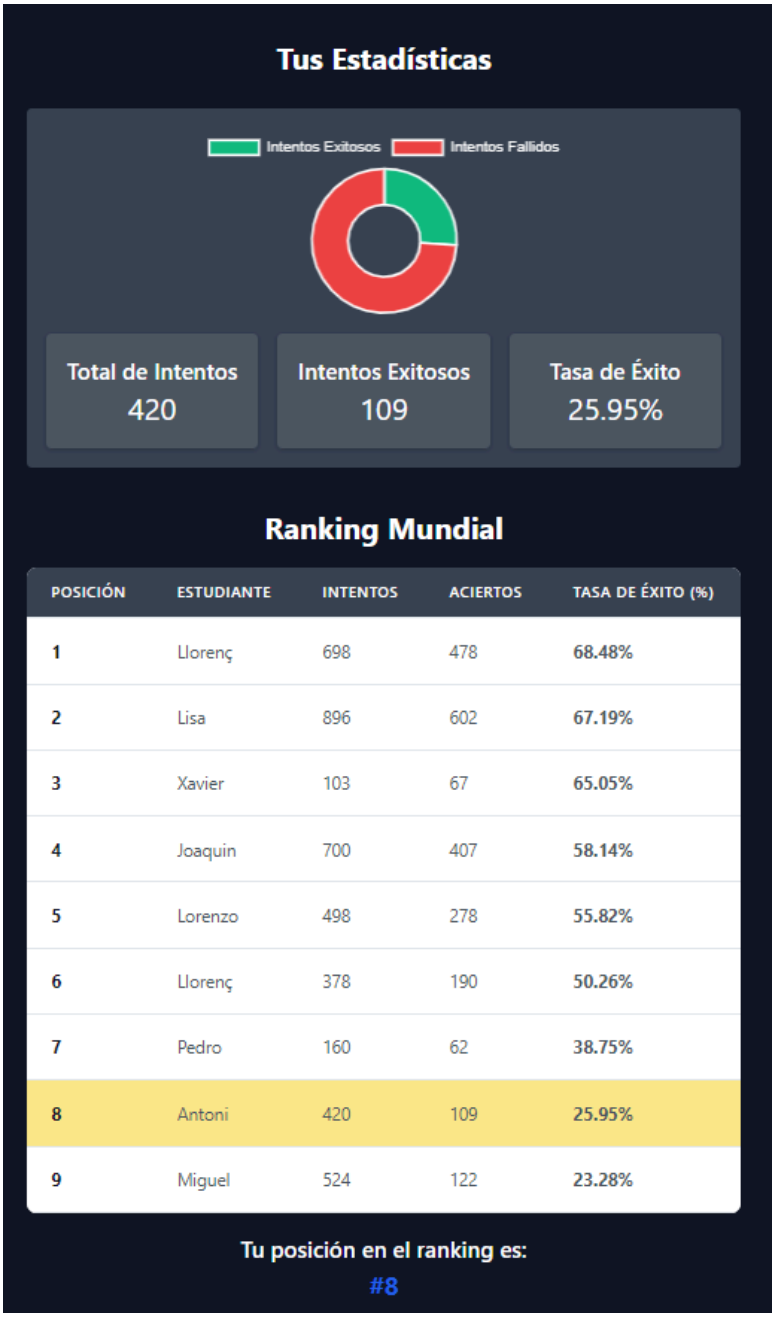

Figura 28. Estadísticas del usuario

En la sección premium, accesible desde la barra de navegación o desde varias páginas, los usuarios pueden encontrar información detallada sobre las características y beneficios del servicio premium, contrastando con las funcionalidades del acceso básico **(Figura 29)**. Además, se incluye un botón de color lila en la parte derecha de la página que sirve como enlace directo a la plataforma de pago *Stripe*. Este botón permite a los usuarios proceder a añadir sus datos de tarjeta de crédito y completar el proceso de pago.

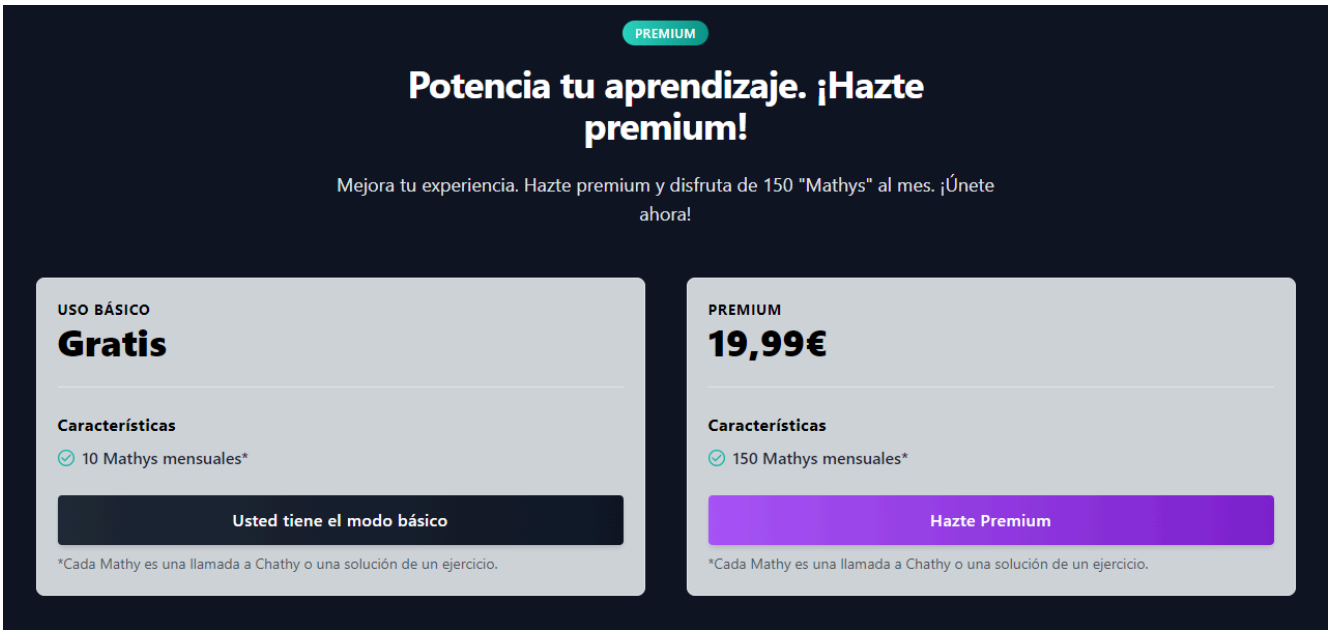

Figura 29. Página premium

Colores

Cuando estamos navegando por la interfaz de una página web, los colores pueden parecer pequeños detalles sin importancia, pero en el fondo, juegan un papel vital en la experiencia y percepción de los usuarios. Unos tonos de color correctos pueden ser diferenciales entre una interfaz buena y una mala, ya que pueden afectar a la usabilidad y accesibilidad.

Gracias al uso de *Tailwind* CSS, ha sido mucho más sencillo la elección de colores, ya que este framework proporciona una amplia gama de colores predefinidos con diversas tonalidades de cada uno. A continuación, vamos a detallar los colores principales utilizados en la interfaz de usuario:

 **Bg-gray-900** (*#1A202C*). Es el color principal de la aplicación. El tono oscuro aporta una sensación de elegancia y es agradable a la vista. Además, la gran mayoría usa el "modo oscuro" para buscar ese tipo de colores para descansar la vista, por lo tanto, directamente ayudamos a minimizar la fatiga visual, permitiendo que los usuarios puedan interactuar con la aplicación web más tiempo de manera cómoda. Finalmente, este color hace destacar a los botones y lugares de interés, creando una jerarquía visual clara.

 **Bg-blue-700** (*#2B6CBO*). El tono de este azul distintivo lo usamos en los botones, así son destacados en la interfaz y logran captar la atención del usuario. Este contraste facilita la navegación del usuario, haciendo más fácil encontrar los controles importantes de manera más rápida.

 **Purple-700** (*#6B46C1*). Este tono de color lila vivo también lo usamos para los botones y tiene la misma función que el color explicado anteriormente. De este modo, también destaca en la interfaz y logra captar la atención de los usuarios.

 Gray-200 (*#E2E8F0*). Los textos secundarios y que no requieren de mucha atención, son escritos en este tono grisáceo. De esta manera, como son secundarios, se requiere de menos contraste en comparación con los textos más importantes que directamente son del color blanco básico.

5.2 Fundamentos del Sistema de Suscripción Premium y los *Mathys*

Esta sección la tomamos desde un punto de vista empresarial, dejando de lado la parte técnica de nuestro proyecto. La idea es crear una estrategia para monetizar la aplicación y a la vez, mejorar la experiencia del usuario. Para ello, creamos el sistema de suscripción mensual y la moneda virtual de la aplicación, los *Mathys*.

Esta estrategia nos ayudará a generar ingresos para poder mantener las APIs externas y poder seguir con el crecimiento de la aplicación. Para conseguirlo, se ha pensado en ofrecer

las funcionalidades más importantes del sistema, como las soluciones completas paso a paso y el uso del *chatbot*, a cambio de una moneda virtual *Mathy*. Por lo tanto, la moneda virtual juega un papel muy importante en nuestra aplicación, ya que fomenta la interacción continua y activa en la plataforma. Como es requerido en las funcionalidades más importantes, motivamos a los usuarios a permanecer activos en la suscripción, aumentando así la monetización.

Sin esta estrategia, las APIs de terceros serían llamadas sin control ninguno y el coste de estas podría llegar a ser muy elevado, además de haber retrasos en las conexiones usuario-servidor y obtener así una experiencia del usuario muy mala.

Para concluir, gracias a este enfoque empresarial, garantizamos una fuente de ingresos para contar con más recursos, mejorar la aplicación y así tener un objetivo más allá del desarrollo de un proyecto de software.

5.3 Sistema de autenticación

Existen varios sistemas para autentificar a los usuarios en aplicaciones web, pero nosotros nos hemos decantado por la autenticación basada en JSON Web Tokens, más conocida por las siglas JWT. Este método ha sido utilizado durante el grado en varios proyectos de software porque es una manera muy simple de manejar de forma segura y eficiente las sesiones de usuarios en las aplicaciones web más modernas.

Cuando un usuario ya registrado inicia sesión con éxito, se generan dos tipos de tokens: un *access token* y un *refresh token*. El primero expira rápidamente y se utiliza en solicitudes en las cuales se necesita autorización para establecer conexión con los *endpoints* de la API. El *refresh token*, el cual expira más tarde, se usa para generar nuevos *access tokens* cuando este ya ha expirado.

Estos dos tipos de tokens son manejados en el *front-end* mediante las conocidas *cookies*. La *cookie* del *refresh token* es almacenada en el navegador del usuario y, cuando es necesario, es enviada al servidor *back-end* para solicitar un nuevo *access token*. Por ejemplo, si un usuario (ya logueado) quiere ver su información en el perfil, el servidor deberá saber si es un usuario con permisos y cuando lo confirme, le enviará la información requerida.

Cuando se requiere una autorización en una solicitud al back-end, el *access token* tiene que ser incluido en el encabezado de autorización como un *Bearer Token*, tal cual se ve en la

Figura 30. Por lo tanto, el back-end deberá verificar la validez del token y comprobar si ese usuario tiene la autorización para la solicitud.

Los *endpoints* del back-end que requieren autorización utilizan la información contenida en el JWT antes de permitir el acceso a los recursos. Cualquier usuario sin permisos que intente acceder a un recurso en el cual los permisos son requeridos, será denegado por el back-end y este le devolverá un código de estado HTTP 401 (Unauthorized) o 403 (Forbidden).

```
let token = await getAccessToken();
let response = await fetch(`${SERVER_DNS}/accounts/delete-account`, {
    method: 'DELETE',
    headers: {
        'Authorization': `Bearer ${token}`,
        'Content-Type': 'application/json',
    },
});
```

Figura 30. Solicitud al back-end que requiere autorización.

5.4 Llamadas a la API

La conexión entre el servidor back-end y el front-end del usuario es una pieza clave en el desarrollo de la aplicación. Para ello, debemos tener en cuenta las peticiones desde el front-end a la vez que todos los *endpoints* del servidor.

Por un lado, las peticiones del front-end las podemos definir como las solicitudes que el cliente realiza al servidor para así recibir o enviar datos mediante diferentes métodos HTTP como GET, POST, PUT o DELETE.

Por otro lado, los endpoints son definidos como URL específicas proporcionadas por el servidor para manejar las solicitudes anteriores. Cada endpoint está conectado a una funcionalidad de la aplicación, por ejemplo, un endpoint *nuestra-api/data* que devuelva cualquier tipo de datos requeridos desde el front-end del usuario.

Llamadas a API desde React

Una de las funcionalidades más importantes en *React* son los *hooks*. Como ya hemos explicado anteriormente, *React* funciona con componentes y los *hooks* nos facilitan la integración de la lógica del estado y efectos secundarios de estos. Existen varios *hooks*, pero nos hemos centrado en dos de los más importantes:

useState [13]

Sirve para manejar el estado dentro de los componentes de *React*. Cuando se realiza una petición de datos desde el front-end, es necesario tener un estado para guardar los datos que nos llegan desde el servidor. Este estado se gestiona localmente dentro del componente mediante el hook *useState*.

useEffect [12]

useEffect es un hook de *React* que nos permite ejecutar una función de código cuando se monta el componente o cuando cambia alguna de las dependencias. Cuando no hay dependencias, este *hook* se invoca automáticamente después de que el componente se renderice y cada vez que es actualizado.

A continuación, veremos un ejemplo de pseudocódigo para llamar al endpoint de una API utilizando los *hooks* explicados anteriormente:

```
import React, { useState, useEffect } from 'react';

function AppExemple() {
  const [data, setData] = useState([]);

  useEffect(() => {
      const fetchData = async () => {
          try {
              const response = await fetch('https://api.example.com/data');
              const json = await response.json();
              setData(json);
          } catch (error) {
              console.error('Error fetching data:', error);
          }
      };

      fetchData();
  }, []);
}
```

Figura 31. Pseudocódigo para consumir una API en React.

En la **figura 31**, podemos observar un pequeño ejemplo de cómo hemos llamado a la API al momento de montarse un componente de *React*. Dentro del hook *useEffect*, definimos la función *fetchData* para ejecutar la solicitud GET mediante la función *fetch()* hacia la URL de la API deseada. Al obtener la respuesta de la API, convertimos los datos recién llegados a formato JSON mediante la función *json()* para después guardarlos en el estado local *data*, el cual ha sido inicializado localmente con el hook *useState.*

5.5 Servicios externos

Una API [21] es un conjunto de reglas y protocolos que permiten a distintas apps comunicarse e interactuar. Las APIs son intermediarias, ya que te permiten acceder, interactuar y utilizar datos específicos de servicios externos. En este apartado, comentaremos cuáles de estas herramientas han sido utilizadas y cómo se han implementado.

5.5.1 OpenAI API

Para la implementación de *Chathy*, el chatbot, hemos utilizado la API que nos proporciona OpenAI, la empresa detrás de ChatGPT.

OpenAI ofrece diferentes modelos de chatGPT, pero nosotros utilizaremos el modelo GPT 3.5 Turbo. Ninguno de estos modelos es gratis, pero su coste es muy bajo, casi nulo. El coste de usar la API GPT-3.5 Turbo es de 0,0015$ por cada 1.000 tokens de entrada y 0,0020$ por cada 1.000 tokens de salida para la variante de 4.000 tokens [14], permitiendo así un uso eficiente y rentable de la API.

La implementación [15] consta de realizar una llamada a la API de ChatGPT para obtener una respuesta generada por el modelo deseado (gpt-3.5-turbo). Como se puede ver en la **Figura 32**, se envía un array de mensajes al modelo, donde el primer mensaje tiene el rol de *system* y proporciona una instrucción específica para guiar la respuesta del modelo. El segundo tiene el rol de *user* y contiene el input del usuario, que sería la pregunta o comentario que ha enviado el usuario por el chat.

```
response = client.chat.completions.create(
    model="gpt-3.5-turbo",
    messages=[{"role": "system", "content": "Explica todo como si fueras un profesor de matemáticas de una universidad
            {"role": "user", "content": f"{user_input}"}]
)
```

Figura 32. Llamada a la API de OpenAI

La solicitud de la **Figura 32** nos devuelve los datos en un formato JSON con mucha información variada, de la cual solamente nos interesa la sección del contenido del mensaje, (la respuesta al input del usuario). Esta respuesta es la parte importante que necesitamos extraer del JSON y enviar al front-end para poder mostrarla al usuario a través del chatbot **(Figura 33)**.


```
[
    "message",
    [
        [
            "content",
            "La raíz cuadrada de 8 es igual a aproximadamente 2.83."
        ],
        [
            "role",
            "assistant"
        ],
        [
            "function_call",
            null
        ],
        [
            "tool_calls",
            null
        ]
    ]
]
```


Figura 33. Parte de la respuesta dada por la API de OpenAI

5.5.2 Wolfram Alpha API

La funcionalidad más importante de la aplicación es la posibilidad de ver la solución completa paso a paso de los ejercicios. Para conseguirlo, se ha integrado la API de Wolfram Alpha para que los usuarios puedan obtener un feedback al momento y así comprender y resolver cualquier tipo de duda sobre el ejercicio deseado.

La API de Wolfram proporciona varios modelos adaptados para cualquier necesidad. El modelo que nos interesa es el “Result + Step by Step”, el cual es gratuito para un número limitado de usos. Sin embargo, cuando se excede un cierto número de llamadas, deja de ser gratuito.

Para integrar la API en el servidor [16], hay que configurar los parámetros necesarios en la llamada. Como podemos ver en la **Figura 34**, los parámetros son: la consulta del usuario, la clave API, el formato de salida deseado (en este caso JSON) y el modelo que queremos utilizar. Una vez tenemos todo preparado, se envía una solicitud GET a la URL http://api.wolframalpha.com/v2/query con los parámetros anteriores. Una vez recibidos los datos, estos son procesados para obtener y utilizar la información deseada.

```
try:
    WOLFRAM_API_KEY = os.getenv("WOLFRAM_API_KEY")
    query = request.data.get('query','')
    url= "http://api.wolframalpha.com/v2/query"
    params= {
        "input": query,
        "appid": WOLFRAM_API_KEY,
        "output": "JSON",
        "podstate": "Result__Step-by-step+solution"
    }
    response = requests.get(url,params=params)
    data= response.json()
```

Figura 34. Pseudocódigo para obtener una respuesta de la API de Wolfram Alpha

La solicitud de la API devuelve un JSON con mucha información variada. El front-end seleccionará la información relevante según el tipo de ejercicio. Antes de enviar los datos a front-end, se buscará si existe una solución paso a paso (**Figura 35**) para el ejercicio. En el caso de que exista, se realiza otra solicitud GET para obtenerla. Si no existe, se envía la solución completa sin detalles paso a paso. En ambos casos, se reduce la cantidad de *Mathys* del usuario.


```
"title": "Possible intermediate steps",
"img": {
    "src": "https://www6b3.wolframalpha.com/Calculate/MSP/MSP52831hc16e55a7g10f6600005gg2e986h30gfai4?MSPStoreType=image/gif&s=9",
    "alt": "Do the following division:\n8 ÷ 2\nDraw a collection of 8 blocks:\n\nMake 2 groups by moving 2 blocks from the pile, one for each group. Now
       there are 6 blocks left:\n \nMove one more block into each of the 2 groups. There are 4 blocks left:\n \nMove one more block into each of the 2
       groups. There are 2 blocks left:\n \nMove one more block into each of the 2 groups. There are 0 blocks left:\n\nThere are no blocks in the pile.
       Count the number of blocks in each group. This is the quotient:\n4\nThe remainder is 0:\n0\nExpress the answer using words:\nAnswer: | \n | 4 with
       remainder 0",
    "title": "Do the following division:\n8 ÷ 2\nDraw a collection of 8 blocks:\n\nMake 2 groups by moving 2 blocks from the pile, one for each group. Now
       there are 6 blocks left:\n \nMove one more block into each of the 2 groups. There are 4 blocks left:\n \nMove one more block into each of the 2
       groups. There are 2 blocks left:\n \nMove one more block into each of the 2 groups. There are 0 blocks left:\n\nThere are no blocks in the pile.
       Count the number of blocks in each group. This is the quotient:\n4\nThe remainder is 0:\n0\nExpress the answer using words:\nAnswer: | \n | 4 with
       remainder 0",
    "width": 530,
    "height": 1136,
    "type": "2DMathPlot_1",
    "themes": "1,2,3,4,5,6,7,8,9,10,11,12",
    "colorinvertable": true,
    "contenttype": "image/gif"
},
"plaintext": "Do the following division:\n8 ÷ 2\nDraw a collection of 8 blocks:\n\nMake 2 groups by moving 2 blocks from the pile, one for each group. Now
    there are 6 blocks left:\n \nMove one more block into each of the 2 groups. There are 4 blocks left:\n \nMove one more block into each of the 2
    groups. There are 2 blocks left:\n \nMove one more block into each of the 2 groups. There are 0 blocks left:\n\nThere are no blocks in the pile. Count
    the number of blocks in each group. This is the quotient:\n4\nThe remainder is 0:\n0\nExpress the answer using words:\nAnswer: | \n | 4 with remainder
```


Figura 35. Paso a paso en una respuesta de la API de Wolfram Alpha

5.5.3 Stripe API

En este apartado, vamos a llevar a la práctica la estrategia de monetización a partir de la gestión de la suscripción premium mensual y el pago de esta. Para ello, hemos integrado la API de *Stripe*. Todo este proceso se divide en varios pasos con el fin de manejar las transacciones de manera segura y eficiente:

1. **Configuración de la API.** En primer lugar, registramos la empresa en la plataforma Stripe y obtenemos las llaves personales (la pública y la privada). La llave privada tiene que ser ocultada en el servidor para conseguir la comunicación segura con la API, mientras que la clave pública se utiliza desde el front-end. Es importante destacar que durante todo el proceso de desarrollo se ha utilizado el modo de prueba y que cualquier tipo de dato es ficticio.

2. **Catálogo de productos.** Cuando ya tenemos la empresa registrada, tenemos que crear un producto para la venta. En nuestro caso, el producto que vamos a vender es la suscripción mensual.

3. **Enlace de pago.** Para manejar las suscripciones premium [17], entre varias opciones, se ha optado por utilizar la interfaz de Stripe para facilitar los pagos **(Figura 36)**. Una vez que los usuarios seleccionan la opción de obtener el servicio premium, son redirigidos directamente a la página de cobros de *Stripe*. Esto no solo simplifica la implementación, sino que también maximiza la seguridad, ya que toda la información de pago se procesa directamente en los sistemas seguros de *Stripe*. Una vez finalizado el pago, se vuelve a redirigir al usuario a la aplicación web donde le saldrá un mensaje de confirmación o de error.

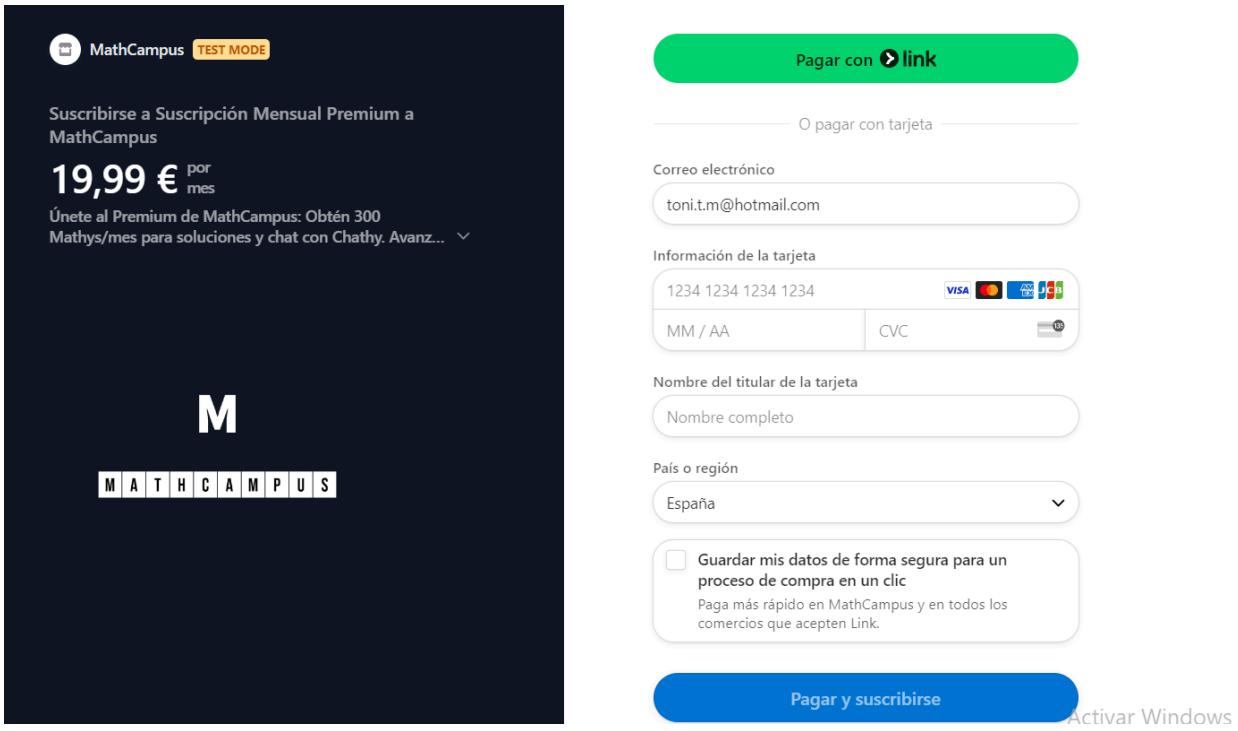

Figura 36. Interfaz de pago de *Stripe*

4. **Dashboard de *Stripe*.** El panel de Stripe proporciona una vista integral de las suscripciones. Como vemos en la **Figura 37**, desde el panel, podemos visualizar la información relacionada con las suscripciones mensuales: la información de los usuarios, el estado de los pagos, las fechas de facturación, etc. También nos permite gestionar las suscripciones, actualizarlas, cancelarlas o renovarlas.

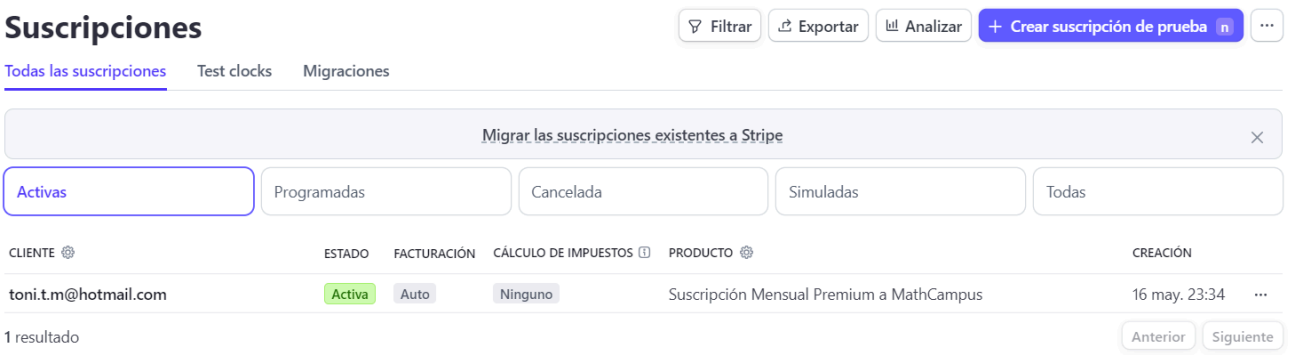

Figura 37. Dashboard *Stripe*

5. **Manejo de respuestas y webhooks.** Los Webhooks [18] son una herramienta que sirve para escuchar eventos en la cuenta de Stripe en el punto de conexión para que en la integración se puedan activar reacciones de manera automática. Por ejemplo, cuando ocurre un evento, como por ejemplo una transacción exitosa, *Stripe* envía una notificación en forma de mensaje *HTTP POST* al endpoint configurado. Esto permite que el servidor de la aplicación web reciba actualizaciones en tiempo real sobre los eventos, lo que facilita la sincronización automática de estados de cuenta y la gestión de clientes sin intervención manual.

En el endpoint que hemos creado para el funcionamiento de los Webhooks, primero intentamos construir un objeto de evento *Stripe* a partir del payload JSON y la firma digital, utilizando la clave API de Stripe **(Figura 38)**.

```
payload = request.body
sig_header = request.META['HTTP_STRIPE_SIGNATURE']
event = None
try:
    event = stripe.Event.construct_from(
        json.loads(payload),sig_header, stripe.api_key
)
```

Figura 38. Pseudocódigo de cómo construir un evento *Stripe*

Después de validar y construir el objeto evento, tenemos que verificar el tipo de evento y ejecutar diferentes acciones basadas en ese tipo **(Figura 39)**:

- *customer.created*: si se crea un nuevo cliente, añadimos el "customer_id" proporcionado por *Stripe* en la base de datos del usuario.
- *Invoice.payment_succeeded*: si un pago de factura se completa con éxito, se verifica si el usuario ya está marcado como premium. Si no lo estaba, cambia su estado a premium y se le añaden los *Mathys* correspondientes. Si el usuario ya era premium, simplemente añade los *Mathys* correspondientes sin cambiar el estado.
- *customer.subscription.deleted*: si una suscripción se cancela, se cambia el estado del usuario a no premium.
- *invoice.payment_failed*: si un pago de factura falla, ocurre lo mismo que en el anterior evento, se cambia el estado del usuario a no premium.

```
# Handle the event
  if event['type'] == 'customer.created':
      customer_data  = event['data']['object']
      handle_customer_created(customer_data )
  elif event.type == 'invoice.payment_succeeded':
      handle_payment_succeeded(event)
  elif event.type == 'customer.subscription.deleted':
      handle_subscription_deletion(event)
  elif event.type == 'invoice.payment_failed':
      handle_payment_failed(event)
  else:
      print('Unhandled event type {}'.format(event.type))
  return HttpResponse(status=200)
```

Figura 39. Pseudocódigo del manejo de un evento *Stripe*

Finalmente, para obtener una cuenta premium y pagar la suscripción mensual, se puede utilizar la tarjeta con el número 4242 4242 4242 4242, con fecha de vencimiento 30/28 y código CVV 123.

5.6 Scripts de los ejercicios

En esta sección se explicará cómo se generan los ejercicios automatizados y cómo enviamos toda la información necesaria a la plantilla del front-end.

Primero de todo, se explicará cómo obtenemos el ejercicio para poder verlo desde el front-end. Los ejercicios se obtienen a través de su *slug*, que es un identificador único para cada ejercicio. Este es enviado al endpoint de *GetProblemView* para obtener el ejercicio a través del slug. En la app *education*, existe una carpeta llamada *generators* **(Figura 40)**.

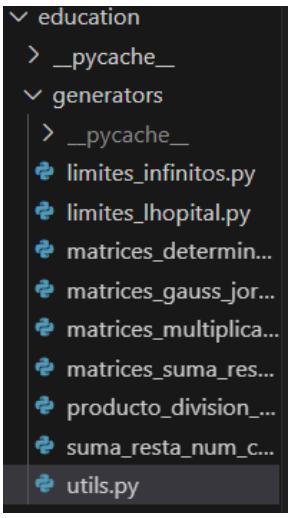

Figura 40. Carpeta generators con los scripts

En esta carpeta están todos los scripts de cada ejercicio disponible. De este modo, si el problema está disponible, se intenta generar dinámicamente usando el script de la carpeta *generators*, que corresponde al *slug* del problema. Si el generador está disponible y el problema se genera con éxito, los datos se combinan y se envían al front-end.

Hay que destacar que los scripts de los generadores tienen que ser nombrados igual que los *slugs* únicos de los ejercicios, así facilitamos su importación y ejecución dinámica del código. Con ello, automatizamos el proceso de carga (**Figura 41**) y ejecución de la función específica dentro de cada script. De esta manera, los ejercicios se generan dinámicamente en función de los *slugs* proporcionados por los usuarios, optimizando el flujo del trabajo.

```python
#Función para obtener el problema a través del slug
def get_problem_by_slug(slug):
    try:
        # Intenta importar dinámicamente el módulo basado en el slug
        problem_module = import_module(f"education.generators.{slug.replace('-', '_')}")
    except ModuleNotFoundError:
        return None

    # Asume que cada módulo de generador tiene una función llamada `generate_problem`
    problem_data = problem_module.generate_problem()
    return problem_data
```

Figura 41. Pseudocódigo de cómo obtener el ejercicio automatizado desde los scripts

Hablando más en detalle de los scripts, nombrar uniformemente a la función principal *generate_problem* de dentro de cada script tiene una importancia crucial para la eficiencia del sistema. Al hacerlo, se permite que el código de la aplicación invoque dinámicamente la función correcta de generación de problemas sin errores de mapeo, independientemente del ejercicio específico que se solicite, facilitando así la expansión y el mantenimiento del código, ya que cualquier generador nuevo sólo necesita implementar esta función para integrarse sin problemas.

El mantener un formato de retorno (**Figura 42**) para todos los scripts es muy importante para asegurar la compatibilidad con la plantilla del front-end, que espera una estructura específica de datos. En este formato incluimos:

- *options:* indica si el ejercicio tiene opciones de respuesta.
- *enunciado*: en Látex.
- *operacion* (si hay): en Látex.
- *operación_wolfram*: versión compatible con Wolfram Alpha.

En el caso de que sí que haya opciones, se añadirá *respuesta_correcta*, con la solución correcta, y *respuestas_erroneas* con las tres respuestas incorrectas.

Por último, se han utilizado librerías populares como *Sympy* y *Numpy,* las cuales son esenciales para facilitar las operaciones matemáticas dentro de los scripts. *Sympy* es útil para manipular expresiones matemáticas, lo que es ideal para generar enunciados y operaciones, mientras que *Numpy* es excelente para manejar operaciones numéricas y cálculos que involucran grandes volúmenes de datos o matrices.

```
return {
    "options": True,
    "enunciado": enunciado_latex,
    "operacion": operacion_latex,
    "operacion_wolfram": operacion_wolfram,
    "respuesta_correcta": result_matrix_latex,
    "respuestas_erroneas": incorrect_answers
}
```

Figura 42. Pseudocódigo del retorno de un script con opciones

5.7 Despliegue en AWS

5.7.1 Despliegue del back-end en AWS EC2

La búsqueda de un rendimiento óptimo para nuestro proyecto es una de las causas por las que utilizamos *EC2* de *AWS* para desplegar nuestro servidor. Este contenedor nos permite una gestión escalable y segura de los recursos del servidor, manejando el tráfico y necesidades de almacenamiento. Gracias a este servicio, es posible lanzar y gestionar máquinas virtuales en la nube. En este apartado, se explicará paso a paso cómo se ha realizado todo el proceso.

Primero, dentro del gran catálogo de posibilidades ofrecidas por *AWS*, lanzamos una instancia en Ubuntu y configuramos las reglas de seguridad necesarias, permitiendo así el tráfico en los puertos requeridos (SSH, HTTP, HTTPS). Una vez lanzada, podemos observar nuestra instancia creada y en ejecución desde el panel de instancias proporcionado por *AWS* (**Figura 43**). En este, podemos ver toda la información de la instancia, como todos sus detalles, estado y alarmas, monitoreo, seguridad, almacenamiento, etc.

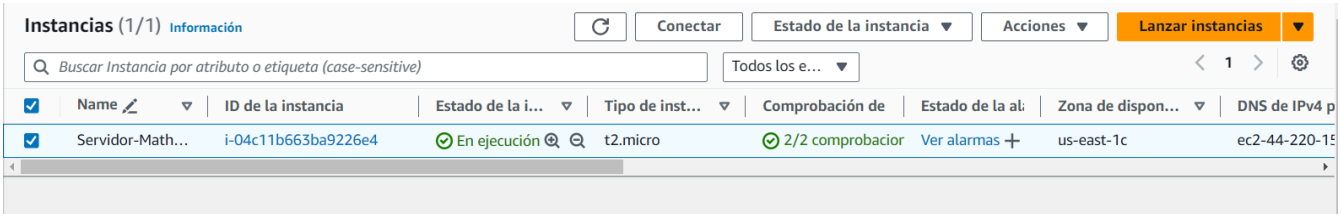

Figura 43. Panel de instancias en AWS EC2.

Cuando ya estamos dentro de la instancia del contenedor, procedemos a instalar todas las dependencias necesarias: *Python*, *pip*, *Nginx* y *Git*, además del entorno virtual de Python. Una vez comprobado que se han instalado correctamente, clonamos el repositorio GitHub, aplicamos las migraciones de la base de datos y configuramos las variables del entorno del servidor (claves privadas, etc.).

Configuramos *Nginx* para manejar directamente las solicitudes HTTP y HTTPS. Este actuará como servidor web para la aplicación, gestionando las solicitudes y el servicio de archivos estáticos, asegurando un buen rendimiento y una buena gestión de las conexiones.

Uno de los temas más importantes es la seguridad en las comunicaciones del servidor. Por ello, obtuvimos un dominio a través de GoDaddy para implementar HTTPS para proteger la transferencia de datos. Para conseguirlo, utilizamos herramientas sofisticadas como *Certbot* y *Let's Encrypt,* las cuales nos facilitaron la obtención y la configuración de certificados SSL.

Finalmente, es buena práctica mantener el servidor en ejecución constante a través de *tmux*, un multiplexor de terminal que permite crear sesiones persistentes.

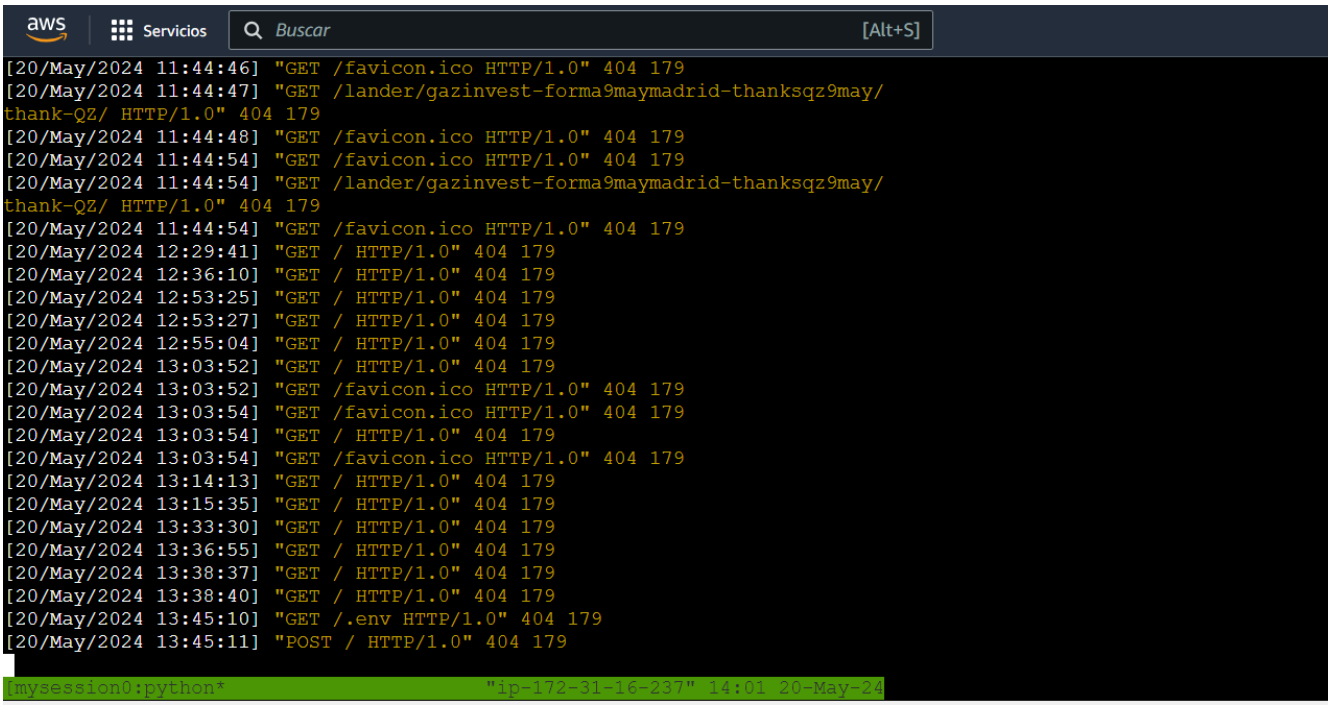

Figura 44. Sesión *tmux* con el servidor en ejecución

5.7.2 Despliegue del front-end en AWS Amplify

Una vez el servidor está en funcionamiento, toca configurar el despliegue del front-end mediante *AWS Amplify*, una plataforma diseñada específicamente para el despliegue de aplicaciones web, la cual facilita la integración continua y el despliegue automático.

Primero conectamos el repositorio Github donde se encuentra el front-end y se configuran los parámetros de build y deploy. *Amplify* tiene la capacidad de detectar automáticamente que estamos usando *React* y ajusta la configuración adecuada mediante el archivo *amplify.yml* (**Figura 45**). Este archivo define las fases de *prebuild*, *build* y *postbuild*, asegurando la ejecución correcta de todos los pasos.

Cuando ya tenemos todo configurado, al hacer *push* en la rama principal del repositorio GitHub, *AWS Amplify* lo reconoce y activa automáticamente el proceso de despliegue, asegurando la actualización del front-end con la última versión del código.

amplify.yml

```yaml
version: 1
frontend:
  phases:
    preBuild:
      commands:
        - npm ci --cache .npm --prefer-offline
    build:
      commands:
        - npm run build
  artifacts:
    baseDirectory: build
    files:
      - '**/*'
  cache:
    paths:
      - .npm/**/*
```

Figura 45. Archivo *amplify.yml*

Una vez se ha desplegado la aplicación, se configura un dominio personalizado adquirido en GoDaddy. Gracias a *Amplify*, los registros DNS necesarios son proporcionados para ser configurados en el panel de administración DNS de GoDaddy. La configuración de SSL se gestiona automáticamente, asegurando que todas las conexiones sean seguras.

Por último, el panel de administración de aplicaciones de *AWS Amplify* (**Figura 46**) nos permite ver y editar toda la información importante de la aplicación. Este panel facilita la gestión de configuraciones, monitoreo de despliegues, ajustes de parámetros de build, entre otras, proporcionando un control completo sobre el ciclo de vida de la aplicación.

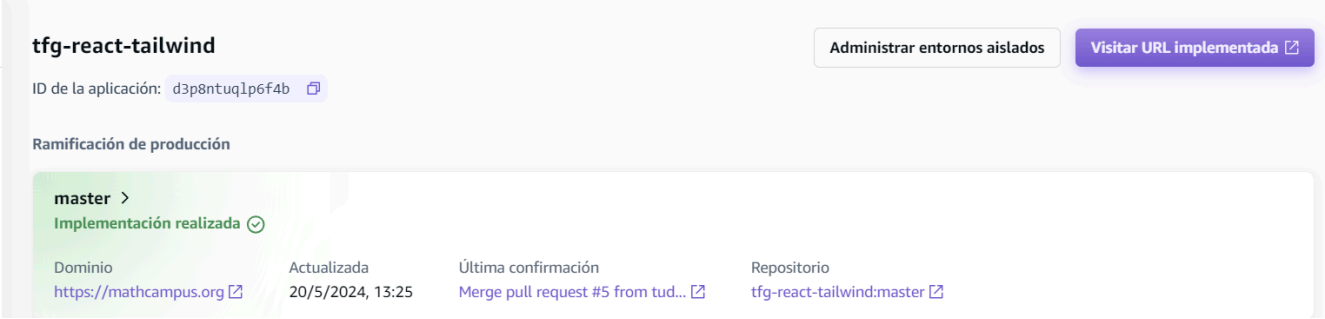

Figura 46. Panel de aplicaciones *AWS Amplify*.

Finalmente, los usuarios pueden acceder a la aplicación y disfrutar de todas sus funcionalidades visitando https://mathcampus.org.

5.8 Resultados

Para finalizar con la implementación, se procede a describir la prueba final para comprobar el correcto desempeño de las funcionalidades disponibles más importantes:

1. Entrar en https://mathcampus.org.
2. Registrarse en la aplicación web.
3. Ver el perfil.
4. Hacerse premium.
5. Dirigirse al apartado de universidades y moverse entre ellas para llegar a un ejercicio disponible.
6. Responder un ejercicio.
7. Solicitar la solución completa paso a paso.
8. Cerrar sesión.

Para un uso ilimitado de las funcionalidades en las que se requieren *Mathys*, se dispone de una cuenta pública para que se puedan realizar cualquier tipo de pruebas y experimentar al 100% con la aplicación. El correo es: “tfg@gmail.com” y la contraseña: “Tfg2024.”.

El código se encuentra en el repositorio GitHub https://github.com/tuduri11/MathCampus.git. Aunque existen muchas otras funcionalidades, estas son las más destacables, ya que han sido el objetivo principal, desde el inicio hasta el final.

6. Conclusiones y proyección

El objetivo principal de este proyecto ha sido diseñar una aplicación web desde cero, desde el análisis y diseño hasta el despliegue. Por lo tanto, hemos conseguido todo lo propuesto con el tutor al principio del trabajo

Por lo que respecta al front-end, hemos diseñado una interfaz simple e intuitiva con la que los estudiantes puedan interactuar fácilmente, lo que facilita su inmersión y concentración en los ejercicios. Las interfaces de usuario siempre pueden mejorar y, en un posible futuro, uno de los principales objetivos sería mejorarla, pero siempre teniendo en cuenta que lo más importante es la comodidad de los estudiantes a la hora de realizar ejercicios.

Por la parte del back-end, considero que he desarrollado una REST API escalable, consistente y segura, aplicando todas las técnicas que he aprendido en varias asignaturas como *Diseño de Software*, entre otras. Sobre todo, hemos conseguido que sea escalable y fácil de mantener para tener una posible proyección de futuro.

Por otra parte, no se han cumplido todos los requisitos del Backlog y, de cara a un futuro trabajo, se deberían desarrollar:

- Front-end y back-end para editar toda la información personal del usuario.
- Paso a paso de los ejercicios en el idioma del usuario.
- Mejoras de seguridad para el acceso de endpoints.
- Front-end para el pago del Premium.
- Facilidad para añadir los scripts de los ejercicios automatizados.
- Mejoras en el diseño del front-end.
- Despliegue continuo en el servidor *EC2*.
- El uso de *Amazon RDS* para manejar PostgreSQL en *AWS*.

Para concluir este trabajo, ha sido muy interesante aprender tanto en esta área del desarrollo web. Como ya expliqué en otro apartado, siempre he tenido interés en este ámbito y con este proyecto he aprendido mucho y creo que estoy preparado para empezar mi carrera como desarrollador de software.

Me he enfrentado a muchos desafíos durante el desarrollo de este proyecto, pero gracias a ser un estudiante muy autodidacta, los he conseguido superar y llevar a cabo todos los objetivos. Uno de los retos más importantes ha sido trabajar con tecnologías y conceptos totalmente nuevos. Nunca había trabajado con *Tailwind CSS* y trabajar desde cero me ha requerido mucha dedicación y paciencia, y gracias a ello, salgo con una nueva habilidad aprendida.

Una de las dificultades más grandes ha sido el despliegue de la aplicación en *Amazon Web Services*. En esta parte presenté muchos obstáculos debido a la complejidad y a las muchas posibilidades de configuración y gestión en la nube. Para lograr todos mis objetivos en este aspecto, recurrí a documentación oficial de *AWS* en su página web y a varios recursos de aprendizaje como videotutoriales en la web. A partir de aquí, conseguí desplegar la aplicación con éxito y con ello, la finalización de este gran proyecto, adquiriendo valiosas habilidades en la administración de sistemas en la nube, que me serán útiles para futuros proyectos y en mi carrera laboral.

7. Bibliografía

[1] Carpio, J. A. (2013). El primer problema con las matemáticas: No se entiende lo que se lee. RTVE: https://www.rtve.es/noticias/20131209/primer-problema-con-matematicas-no-se-entiende-se-lee/812561.shtml

[2] MozDevNet. “HTML: HyperText Markup Language”: https://developer.mozilla.org/en-US/docs/Web/HTML

[3] MozDevNet. “CSS: Cascading Style Sheets”: https://developer.mozilla.org/en-US/docs/Web/CSS

[4] MozDevNet “JavaScript”: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[5] Documentación Oficial de Python: https://docs.python.org/3/tutorial/index.html

[6] Documentación Oficial de React: https://react.dev/

[7] Documentación Oficial de Django: https://docs.djangoproject.com/en/4.2/

[8] Documentación de Tailwind CSS: https://v2.tailwindcss.com/docs

[9] Información y API de Postman: https://www.postman.com/

[10] Documentación oficial de pgAdmin: https://www.pgadmin.org/docs/pgadmin4/8.5/index.html

[11] ¿Qué es y cómo funciona Stripe? https://getquipu.com/blog/que-es-y-como-funciona-stripe/#que-es

[12] React Hook useEffect: https://legacy.reactjs.org/docs/hooks-effect.html

[13] React Hook useState: https://react.dev/reference/react/useState

[14] API GPT-3.5 Turbo. https://openai.com/index/gpt-3-5-turbo-fine-tuning-and-api-updates/

[15] Implementación API GPT-3.5 Turbo https://platform.openai.com/docs/guides/text-generation

[16] API Wolfram Alpha: https://products.wolframalpha.com/show-steps-api/documentation

[17] Documentación Suscripciones Stripe: https://docs.stripe.com/billing/quickstart

[18] Webhooks Stripe: https://docs.stripe.com/webhooks

[19] Documentación oficial de AWS: https://docs.aws.amazon.com/

[20] Información Khan Academy https://learnopoly.com/104-khan-academy-statistics/

[21] Qué son las APIs de terceros: https://kinsta.com/es/blog/apis-de-terceros/#:~:text=utilizando%20Kinsta%20APM-,Qu%C3%A9%20son%20las%20APIs%20de%20terceros,datos%20espec%C3%ADficos%20de%20servicios%20externos.

[22] Web oficial de Trello: https://trello.com/es/tour

[23] Visión general de los beneficios - AWS: https://aws.amazon.com/es/application-hosting/benefits/#:~:text=AWS%20le%20permite%20seleccionar%20el,servicios%20que%20necesita%20su%20aplicaci%C3%B3n.

8. Anexos

8.1 Endpoints Cliente

Endpoint	Descripción	Parámetros	Método
api/accounts/register	Crear un nuevo usuario.	{'email','name','surname','password','university'}	POST
api/accounts/login	Comprobación de los datos para el inicio de sesión.	{ "email", "password" }	POST
api/accounts/logout	Invalida el token de autentificación actual.	-	POST
api/accounts/get-token	Obtiene un token JWT para autenticar el usuario.	-	POST
api/accounts/refresh-token	Actualiza el token JWT para extender su tiempo de validez.	-	POST
api/accounts/change-password	Cambia la contraseña del usuario.	{“current_password”, “new_password”}	POST
api/accounts/get-profile	Obtiene la información del usuario.	{ "email"}	POST
api/accounts/get-mathys	Obtiene los mathys de un usuario.	-	POST
api/accounts/update-mathys	Actualiza los mathys de un usuario	{“mathys”}	POST
api/accounts/cancel-subscription	Cancela la suscripción de un usuario.	-	POST
api/accounts/delete-account	Elimina un usuario.	-	DELETE
api/accounts/get-ispremium	Comprueba si un usuario es premium	-	POST
api/accounts/chatgpt	Conexión con la API de OPENAI	{“message”	POST

8.2 Endpoints Education

Endpoint	Descripción	Parámetros	Método
api/education/get-universities	Obtiene todas las universidades disponibles.	-	GET
api/education/get-problem/<slug:slug>	Obtiene el ejercicio a resolver.	{“slug”}	GET
api/education/<slug:slug>/careers	Obtiene las carreras de una universidad en particular.	{“slug”}	GET
api/education/<slug:slug>/subjects	Obtiene las asignaturas de una carrera en particular.	{“slug”}	GET
api/education/<slug:slug>/topics	Obtiene los temas de una asignatura en particular.	{“slug”}	GET
api/education/<slug:slug>/problems	Obtiene los ejercicios de un tema en particular.	{“slug”}	GET
api/education/get-wolfram	Conexión con la API de Wolfram Alpha.	{“query”}	POST
api/education/exerciseDone	Guardar los resultados de las resoluciones de ejercicios.	{“results”}	POST
api/education/ranking	Obtiene el ranking general de los usuarios.	-	GET
api/education/userStats	Obtiene las estadísticas de un usuario en concreto.	-	GET

8.3 Endpoints Favourites

Endpoint	Descripción	Parámetros	Método
api/favourites/addFavourites/<slug:slug>	Añade a la lista de favoritos un ejercicio en particular.	{“slug”}	POST
api/favourites/get-favourites	Obtiene la lista de ejercicios favoritos de un usuario.	-	GET