Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Tecnologías de la Información

Desarrollo de una aplicación para el seguimiento del estado socioemocional de estudiantes de primaria y secundaria

Alumno:
David Morchón Payo

Tutores:
Diego García Álvarez

A mi familia

Agradecimientos

Quiero dedicar este espacio para expresar mi más profundo agradecimiento a las personas que han sido fundamentales en el desarrollo y culminación de este trabajo de fin de carrera.

En primer lugar, quiero expresar mi más sincero agradecimiento a mi familia. Su amor, apoyo incondicional y constante estímulo han sido el motor que me ha impulsado en cada paso de este camino.

A mis amigos, quienes han estado a mi lado durante toda esta etapa, les agradezco por su compañía, por los momentos de alegría compartidos y por ser un apoyo incondicional en los momentos más difíciles.

A mis compañeros de carrera, con quienes he compartido clases, proyectos y experiencias inolvidables, les agradezco por su colaboración, compañerismo y por ser una fuente constante de inspiración y aprendizaje mutuo.

Por último, pero no menos importante, quiero expresar mi más sincero agradecimiento a mi tutor de este trabajo de fin de carrera, Diego. Su guía experta, dedicación y apoyo continuo han sido fundamentales todos estos meses.

Resumen

El objetivo de este proyecto es desarrollar una aplicación web que permita a los docentes de cualquier institución educativa en Castilla y León poder realizar actividades para sus estudiantes extranjeros, ofreciendo la posibilidad de realizar preguntas sobre las mismas. Además, ofrece la funcionalidad para que los estudiantes puedan registrar sus estados emocionales en cualquier momento. Tendrán acceso a un tablón de estados emocionales desde el que podrán registrar su estado de ánimo.

El proyecto se ha desarrollado como una aplicación web, accesible desde cualquier dispositivo. Se ha empleado Python y el *framework* Flask como herramienta de desarrollo. El marco de trabajo empleado para el desarrollo de la aplicación ha sido Scrum.

Abstract

The objective of this project is to develop a web application that allows teachers of any educational institution in Castilla y León to carry out activities for their foreign students, offering the possibility to ask questions about them. In addition, it offers the functionality for students to record their emotional states at any time. They will have access to an emotional states board from which they will be able to register their moods.

The project has been developed as a web application, accessible from any device. Python and the Flask framework have been used as a development tool. The entire project has been developed using the Scrum agile framework.

Índice general

Lista de Figuras

Lista de Tablas

Lista de Fragmentos de código

Capítulo 1

Introducción

1.1. Introducción

Este trabajo se enmarca dentro del Proyecto de Investigación Educativa, "Diseño y validación de un programa de acogida dirigido al alumnado refugiado procedente de Ucrania escolarizado en la enseñanza básica (EDUCYL2022_01)", financiado por la Consejería de Educación de la Junta de Castilla y León a través de la Dirección General De Innovación y Formación del Profesorado.

1.2. Contexto

En el mundo actual, el uso generalizado de Internet ha transformado radicalmente la forma en que se interactúa y realizamos actividades diarias. La penetración de Internet ha alcanzado cifras sin precedentes, con una gran parte de la población mundial accediendo a la red de manera regular. Según estadísticas recientes, se estima que en torno al 66 % de la población mundial tiene acceso a Internet [26], lo que refleja la creciente importancia de esta herramienta actualmente.

Por otro lado, en el ámbito educativo, la adopción de tecnologías digitales es una tendencia en alza. Las instituciones educativas recurren cada vez más a Internet como una plataforma para expandir sus operaciones, mejorar la eficiencia en sus procesos educativos e intentar conectar con las nuevas generaciones que han nacido en un mundo en el que estas tecnologías son omnipresentes y forman parte integral de su vida cotidiana [39]. A pesar de su popularidad, también existen tendencias en contra de su uso excesivo por sus efectos nocivos en los más jóvenes. [40] [41]

En este contexto, surge la necesidad de desarrollar herramientas digitales que aprovechen al máximo las oportunidades que ofrece Internet. Es en este escenario donde nace AcogeCyL,

un proyecto de la Junta de Castilla y León. Este proyecto quiere ayudarse de una aplicación web diseñada para crear actividades personalizadas y seguir los estados emocionales de niños inmigrantes, refugiados, pertenecientes a minorías o de incorporación tardía. AcogeCyL ofrece soluciones que permiten a los usuarios diseñar actividades adaptadas a las necesidades individuales de cada niño y hacer un seguimiento de su estado socioemocional a lo largo del tiempo. Con funcionalidades que van más allá de las aplicaciones educativas tradicionales, AcogeCyL ofrece ser una herramienta indispensable para promover la integración y el bienestar emocional de los niños extranjeros a través de Internet.

1.3. Motivación

El desarrollo de tecnologías educativas destinadas a niños extranjeros y la monitorización de su estado emocional a través de aplicaciones web representan áreas de investigación y desarrollo de gran relevancia en el ámbito educativo contemporáneo.

En entornos educativos donde los niños tienen diferentes antecedentes culturales y hablan idiomas distintos, las herramientas digitales pueden desempeñar un papel crucial al ofrecer actividades personalizadas y adaptadas a las necesidades individuales de cada niño.

Se quiere desarrollar una aplicación para realizar un sondeo sobre la inclusión y situación socioemocional del alumnado ucraniano, con la finalidad de detectar necesidades y así ajustar los contenidos y estrategias del programa de intervención para la mejora de su inclusión educativa. Se desea conseguir [85]:

1. Conocer el grado de integración del alumno refugiado.
2. Analizar otras variables socioemocionales que pueden determinar su bienestar.
3. Comprobar si hay diferencias entre el alumnado de educación primaria y el de educación secundaria obligatoria.

Para realizar una intervención adecuada, lo primero es caracterizar al alumnado destinatario, para así poder ajustar las estrategias y actividades a sus características Por eso, uno de los objetivos generales de la investigación es comprobar si el alumnado refugiado tiene características diferenciales respecto a otro alumnado que esté en situación de desventaja. Para ello se pretende conocer:

1. El estatus sociométrico y grado de victimización del alumnado refugiado respecto al resto de compañeros y compañeras de la clase
2. Analizar el nivel de autoestima, engagement, gaudibilidad y desarrollo de emociones positivas del alumnado refugiado en relación al resto de sus iguales.
3. Analizar si existen diferencias significativas en función de la etapa educativa y el género.

Este trabajo de fin de grado se propone explorar cómo las herramientas digitales pueden ser aprovechadas para mejorar la experiencia educativa de los niños extranjeros y facilitar su integración en entornos escolares diversos y multiculturales.

1.4. Marco de uso y usuarios objetivo

AcogeCyL es una aplicación web diseñada específicamente para un estudio universitario enfocado en niños extranjeros. Dada su naturaleza de investigación, AcogeCyL no tiene competencia directa en el mercado, ya que su desarrollo se centra en proporcionar una plataforma digital única y adaptada a las necesidades de este grupo demográfico particular.

Los usuarios principales de AcogeCyL son los niños extranjeros y sus familias, así como sus docentes y los investigadores responsables del proyecto. A continuación, se describen en detalle los perfiles de los usuarios objetivos:

- **Estudiantes**: Los niños inmigrantes, refugiados, pertenecientes a minorías o de incorporación tardía son el principal grupo de usuarios de AcogeCyL. Estos estudiantes participan en las actividades diseñadas dentro de la aplicación y utilizan la plataforma para expresar su estado socioemocional a lo largo del estudio. Su interacción con la aplicación es fundamental para recopilar datos relevantes sobre su bienestar emocional y su experiencia en el contexto escolar.
- **Docentes**: Los docentes que trabajan con los niños extranjeros son otro grupo clave de usuarios de AcogeCyL. Estos profesionales utilizan la aplicación para diseñar y asignar actividades específicas a los niños, así como para hacer un seguimiento de su progreso y estado emocional.
- **Investigadores**: Los investigadores responsables del estudio universitario son usuarios esenciales de AcogeCyL. Utilizan la plataforma para recopilar datos, analizar resultados y extraer conclusiones significativas sobre el bienestar emocional y el rendimiento académico de los niños extranjeros.

1.5. Objetivos

1.5.1. Objetivos de desarrollo

El objetivo principal del proyecto es conseguir desarrollar una aplicación web que cumpla los siguientes puntos:

- Desarrollar una aplicación web que satisfaga los requisitos del cliente.
- Desarrollar la documentación siguiendo los principales marcos de trabajo de desarrollo.

- Desarrollar una base de datos capaz de almacenar la información necesaria de la aplicación web.
- Poner la aplicación en producción y realizar pruebas con ella.
- Desarrollar una intefaz fácil de usar
- Cumplir las normas y buenas prácticas de programación.

1.5.2. Objetivos personales

- **Proyecto de Ingeniería del Software**: Poder realizar un proyecto de ingeniería del software desde el principio y aplicar todos los pasos del desarrollo aprendidos a lo largo del grado. Al ser estudiante de INdat y estudiar la rama de computación, querría completar mis conocimientos de Sistemas Web al no poder haber profundizado en ellos anteriormente.
- **Dominar el marco de trabajo *scrum***: Consolidar mi comprensión y aplicación de los principios y prácticas *scrum* que pude aprender en las prácticas de empresa que hice. Pudiendo adaptar y utilizar sus herramientas y técnicas de manera efectiva para gestionar el desarrollo de mi TFG de manera individual.
- **Desarrollar habilidades de gestión de proyectos ágiles**: Relacionado con el punto anterior, mejorar mis habilidades en la gestión de proyectos ágiles, incluyendo la planificación, programación, seguimiento y control de actividades relacionadas con el desarrollo de software.
- **Impulsar la autoorganización y responsabilidad personal**: Fomentar la autoorganización y la responsabilidad personal en la planificación y ejecución del proyecto, estableciendo metas claras y realizables y manteniendo la disciplina y el compromiso para alcanzarlas de manera independiente.
- **Cumplir con los objetivos y plazos establecidos**: Ser capaz de fijar un orden de prioridad en las tareas pendientes y gestionar las fechas límite de cada una. Comprometiéndome a entregar todo en plazo, manteniendo la calidad del software.
- **Adaptación a los cambios**: De acuerdo al marco de trabajo empleado, adaptarse a cambios en los requisitos de manera ágil y efectiva, utilizando los principios de *scrum*.

1.6. Estructura de la memoria

Este documento se estructura de la siguiente forma:

Capítulo 1 Introducción: En este capítulo se describe el contexto y motivación del proyecto y qué objetivos se pretenden cumplir al finalizar el desarrollo del mismo.

Capítulo 2 Requisitos y planificación: En este capítulo se presenta y explica el método de planificación del proyecto, *scrum*, y cómo ha sido empleado para el desarrollo del proyecto. Posteriormente se exponen los requisitos iniciales del desarrollo.

Capítulo 3 Tecnologías utilizadas: En este capítulo, se presentará una descripción detallada de todas las tecnologías y herramientas que fueron utilizadas en el desarrollo del proyecto. Se abordarán aspectos como los lenguajes de programación, los frameworks, las bibliotecas y cualquier otro software relevante que haya sido empleado para la implementación del sistema.

Capítulo 4 Especificación de requisitos: En este capítulo se detallan las funcionalidades, requisitos del sistema y se identifican y detallan los casos de uso obtenidos a partir de las historias de usuario.

Capítulo 5 Análisis: En este capítulo se llevará a cabo un análisis detallado de los requisitos del proyecto identificando las necesidades específicas de los usuarios a partir de los requisitos iniciales.

Capítulo 6 Diseño: En este capítulo se presentan las decisiones de diseño tomadas y sus diagramas correspondientes.

Capítulo 7 Implementación: En este capítulo se detallará el proceso de implementación del sistema desarrollado, y las pruebas realizadas. También se comentarán las principales adversidades encontradas.

Capítulo 8 Pruebas: En este capítulo se detallarán las pruebas realizadas a lo largo del desarollo del proyecto para garantizar la calidad del software implementado.

Capítulo 9 Seguimiento del proyecto: Este capítulo examinará el progreso del proyecto a lo largo del tiempo, destacando los hitos alcanzados, los desafíos enfrentados y las medidas tomadas para abordarlos, con el fin de garantizar la entrega exitosa del producto final.

Capítulo 10 Conclusiones y trabajo futuro: En este capítulo se exponen las conclusiones finales del proyecto y varias propuestas con el fin de mejorar el producto final.

Anexo A Manuales. Se incluye un Manual de Usuario para guiar a los nuevos usuarios por las diferentes funcionalidades de la aplicación.

Anexo B Resumen de enlaces adicionales

Capítulo 2

Planificación

2.1. Introducción

En este capítulo, se explicará en detalle cómo se llevará a cabo la planificación del proyecto de software. Se abordarán las metodologías ágiles, con un enfoque particular en *scrum*, explicando sus principios y beneficios. Además, se describirán los roles específicos dentro del marco de *scrum*, tales como el Product Owner, el *scrum* Master y el equipo de desarrollo. Se presentarán los artefactos esenciales de *scrum*, como el Product Backlog, el *Sprint Backlog* y los Incrementos. Posteriormente, se detalla cómo se aplicará *scrum* en la planificación del proyecto. A continuación, se desarrolla un plan de riesgos, identificando posibles obstáculos y elaborando estrategias para mitigarlos, junto con un análisis de los riesgos identificados. Finalmente, se presenta el *Product Backlog* inicial.

2.2. Marco de trabajo ágil

El marco de trabajo con el que se ha planificado la aplicación es *scrum*, un método ágil. Los métodos ágiles surgieron como respuesta a las limitaciones y rigideces de las metodologías tradicionales de desarrollo de software, las metodologías ágiles se centran en la adaptabilidad, la colaboración y la entrega continua de valor al cliente.

A diferencia de los enfoques tradicionales que valoran la planificación y la documentación exhaustiva desde el inicio, las metodologías ágiles abogan por un enfoque iterativo e incremental, donde los equipos trabajan en ciclos cortos de desarrollo, recibiendo y respondiendo rápidamente a la retroalimentación del cliente y los cambios en los requisitos.

Los métodos ágiles se basan en torno al *Agile Manifesto* o Manifiesto Ágil [44]. Este articula sus cuatro valores y doce principios fundamentales. Sus valores son:

1. ***Las personas y las interacciones*** *sobre procesos y las herramientas.*
2. ***Software funcionando*** *sobre a la documentación exhaustiva.*
3. ***Colaboración con el cliente*** *sobre la negociación de contratos.*
4. ***Responder ante el cambio*** *sobre seguir un plan.*

Esto es, aunque se valoran los elementos de la derecha, se valoran más los elementos de la **izquierda**. Los principios fundamentales del Manifiesto ágil son [45]:

1. *Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.*
2. *Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.*
3. *Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.*
4. *Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.*
5. *Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.*
6. *El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.*
7. *El software funcionando es la medida principal de progreso.*
8. *Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.*
9. *La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.*
10. *La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.*
11. *Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.*
12. *A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.*

La naturaleza iterativa e incremental de las metodologías ágiles me permitió priorizar y desarrollar las características más importantes del producto en iteraciones semanales, lo que me permitió entregar software funcional de manera temprana y continua. Esto facilitó la retroalimentación rápida con mi tutor y nos permitió adaptar el producto de manera ágil a medida que evolucionaban los requisitos y las necesidades.

2.2.1. Scrum

El nombre *Scrum* procede de los *scrums* de rugby y de la imagen de todos empujando juntos por un compromiso común. El proceso se diseñó originalmente para el desarrollo de nuevos productos de software destinados a un mercado competitivo. En este caso, sacar algo al mercado antes que los competidores puede ser más importante que tener una amplia gama de características no esenciales [13].

La característica principal de *scrum* es la rapidez del desarrollo del software funcional, influenciada por la teoría empírica del control de procesos, que postula que el conocimiento proviene de la experiencia y de la toma de decisiones basadas en lo que se conoce. *scrum* emplea un enfoque empírico a través de sus tres pilares [47]:

- Transparencia: Toda la información relevante sobre el trabajo realizado y el trabajo por hacer esté fácilmente disponible y comprensible para todos los miembros del equipo *scrum*, así como para los interesados externos. La transparencia implica que no debe haber secretos ni ocultamientos en relación con el trabajo del equipo.
- Inspección: implica que los artefactos producidos durante el proceso de desarrollo, así como el progreso del equipo hacia la meta del *sprint*, deben ser inspeccionados de manera regular. La inspección ayuda a identificar problemas y desviaciones tempranas, lo que permite corregir estas tempranamente.
- Adaptación: La adaptación implica la capacidad de ajustar el enfoque, el plan y el producto en función de los hallazgos de la inspección. Una vez que se han identificado áreas de mejora durante la inspección, el equipo *scrum* debe ser capaz de adaptarse rápidamente para abordar esos problemas y maximizar el valor entregado.

Estos pilares permiten a los equipos evaluar continuamente su trabajo, identificar áreas de mejora y adaptar su enfoque en consecuencia.

Con todas las características de *scrum* ya conocidas, se puede afirmar que se trata de una metodología de trabajo muy acorde al proyecto que se va a realizar. Se trata de un proyecto relativamente ambiguo en muchos apartados ya que los requisitos funcionales del cliente no son muy estrictos, se valora la rapidez de la entrega de software funcional y hay una estrecha colaboración con el Departamento de Psicología Evolutiva y de la Educación de la Universidad de Valladolid. Estas características seguramente conlleven realizar muchos cambios o modificaciones de forma ágil.

2.2.1.1. Roles

Dado que la esencia de *scrum* es el empirismo, la autoorganización y la mejora continua, los tres roles que existen dentro del *scrum* ofrecen una definición mínima de responsabilidades y rendición de cuentas para que los equipos puedan entregar el trabajo de forma eficaz. Esto permite a los equipos asumir la responsabilidad de cómo se organizan y seguir mejorando. La suma de todos los roles es lo que se llama Equipo *Scrum*. Estos son: [43].

- ***Product Owner***: El *Product Owner* es responsable de maximizar el valor del producto y del trabajo del equipo de desarrollo. Esto implica gestionar el *Backlog* del Producto (explicado en 2.2.1.3). El Product Owner trabaja estrechamente con los *stakeholders* para entender y priorizar los requisitos, así como para garantizar que el equipo de desarrollo esté trabajando en las tareas más valiosas en cada momento. Además, el Product Owner toma decisiones sobre qué funcionalidades se incluyen en cada *Sprint* (explicado en 2.2.1.3) y es responsable de aceptar o rechazar el trabajo realizado por el equipo de desarrollo en los *Sprint Reviews*.
- **Scrum Master**: Es un facilitador y un líder al servicio del equipo *scrum*. Su función principal es ayudar al equipo a entender y adoptar los principios y prácticas de *scrum*, así como a eliminar cualquier impedimento que obstaculice su progreso. El *Scrum Master* también es responsable de proteger al equipo de desarrollo de distracciones externas y de promover un entorno de trabajo colaborativo y eficaz. Además, suele dirigir los eventos (2.2.1.2) y trabaja para mejorar la productividad del equipo de desarrollo.
- **Equipo de desarrollo**: Suele ser un grupo de 3 a 9 personas para optimizar la comunicación y la colaboración. Se trata de profesionales multidisciplinarios que trabajan colaborativamente para realizar los incrementos (explicados en el siguiente punto). Son responsables de planificar, diseñar, desarrollar, probar y entregar el trabajo en cada *Sprint* de manera colaborativa y autoorganizada.

2.2.1.2. Eventos

Con el objetivo de cumplir los principales pilares de *scrum*, se dispone de una serie de eventos basados en el *timeboxing* que se realizan periódicamente. Estos son los siguientes:

- ***Sprint***: Periodo de tiempo fijo y corto en el que se realiza trabajo para entregar un **incremento** de producto potencialmente entregable. Los *Sprints* suceden secuencialmente, es decir, uno empieza cuando termina el anterior. Los requisitos pueden cambiar de un *Sprint* a otro pero nunca pueden cambiar durante un *Sprint*. En *scrum*, los *Sprints* son la base del proceso de desarrollo y generalmente tienen una duración de 1 a 4 semanas, con dos semanas siendo la duración más común..
- ***Sprint Planning***: Es una reunión que marca el inicio de cada *Sprint*. Durante esta reunión, el equipo *scrum* confecciona los requisitos y selecciona los elementos del *Product Backlog* que se trabajarán durante el *Sprint* y crea el plan para completarlos. El artefacto obtenido es el *Product Backlog*.
- ***Sprint Review***: Es una reunión al final de cada *Sprint* en la que el equipo *scrum* y los *stakeholders* se reúnen para inspeccionar el incremento de producto desarrollado durante el *Sprint*. Durante esta reunión, el equipo presenta el trabajo completado y recibe retroalimentación de los *stakeholders*. El *Product Owner* elegirá qué funcionalidades se añadirán al producto (se puede retocar el *Product Backlog*).

- **Sprint Retrospective**: Es una reunión al final de cada *Sprint* en la que el equipo *scrum* reflexiona sobre su desempeño durante el *Sprint* y busca formas de mejorar su proceso de trabajo. Durante esta reunión, el equipo identifica lo que funcionó bien, lo que no funcionó bien y las acciones para mejorar en el próximo *Sprint*. En ocasiones, este evento no se realiza después de todos los *Sprints*, sino cada dos o tres.
- ***Daily Scrum***: Reunión diaria de corta duración en la que cada miembro del equipo expone qué hizo el día anterior, que hará el dia actual y qué obstáculos ha tenido o va a tener.

2.2.1.3. Artefactos

Los artefactos en *scrum* son documentos o elementos tangibles que proporcionan información sobre el trabajo realizado y el trabajo por hacer en el proyecto. Aquí están los tres artefactos principales en *scrum*:

- ***Product Backlog***: El *Product Backlog* es una lista prioritizada de todos los elementos que podrían ser necesarios para el producto. Estos elementos pueden ser características, cambios, mejoras o cualquier otro tipo de trabajo que agregue valor al producto. El *Product Backlog* es propiedad del *Product Owner* y está compuesto por elementos que representan los requisitos del cliente y las necesidades del negocio. Se prioriza en función del valor que aportan al producto y se actualiza regularmente a medida que evolucionan los requisitos y se obtiene nueva información.
- ***Sprint Backlog***: El *Sprint Backlog* es una lista de todas las tareas que el equipo se compromete a completar durante el *Sprint*. Estas tareas se seleccionan del Backlog del Producto durante la Planificación del *Sprint* y representan el trabajo necesario para entregar el incremento de producto deseado al final del *Sprint*. El *Sprint Backlog* es propiedad del equipo de desarrollo y se actualiza durante el *Sprint* según sea necesario para reflejar el progreso y los cambios en el alcance del trabajo.
- **Incremento**: El Incremento es el resultado del trabajo realizado durante un *Sprint* y representa una versión potencialmente entregable del producto. Al finalizar cada *Sprint*, el equipo *scrum* entrega un Incremento que cumple con los criterios de terminación definidos por el *Product Owner*. El Incremento es una versión mejorada y funcional del producto que puede ser inspeccionada y utilizada por los stakeholders para proporcionar retroalimentación y tomar decisiones sobre el siguiente paso del proyecto.

2.2.2. Aplicación de Scrum a la planificación

Dadas las características del proyecto, se decidió inclinarse hacia un marco de trabajo ágil para la planificación del proyecto. Se trata de una aplicación sobre la que el Departamento de Psicología Evolutiva y de la Educación de la Universidad de Valladolid tiene planeadas una serie de funcionalidades y requisitos iniciales muy generales, pero con la posibilidad de que

estos evolucionen a medida que avance el proyecto. Por lo tanto, la elección de una metodología ágil, se presenta como la opción más adecuada para gestionar eficazmente los cambios y adaptarse a las necesidades. El tiempo disponible para desarrollar la aplicación es de aproximadamente cuatro meses, por lo que se necesitará desarrollar el *software* rápidamente. Además, el tutor y yo hemos decidido realizar reuniones semanales en las que revisaremos los avances realizados esa semana y planificaremos el trabajo para la siguiente. Por estas razones se tomó la decisión de planificar con *scrum*. Con *scrum* se espera poder mantener un enfoque flexible y centrado en la entrega continua de valor, mientras se mantiene una colaboración estechamente con mi tutor, que ejerce de *puente* entre el departamento de Psicología Evolutiva y de la Educación y el estudiante. De esta manera se genera una retroalimentación regular para garantizar la adecuación de la aplicación.

2.2.2.1. Selección de roles

En este caso, el estudiante desempeñará dos roles distintos: Grupo de Desarrollo y *Product Owner*, mientras que el tutor de TFG asumirá el papel de *Scrum Master*. Las reuniones se llevarán a cabo presencialmente de manera regular, con una frecuencia semanal.

Actuar como grupo de desarrollo y *Product Owner* permite mantener el control y la responsabilidad sobre el desarrollo del proyecto, al tiempo que se asegura de que los esfuerzos estén alineados con los objetivos y requisitos del proyecto. El tutor, desempeñando el papel de *Scrum Master*, proporcionará orientación, apoyo y facilitación durante todo el proceso, ayudando a sortear obstáculos y a mantener un enfoque claro en la entrega del proyecto.

2.2.2.2. Desarrollo de los eventos

Como se acaba de mencionar en 2.2.2.1 se celebrarán reuniones semanales, la duración de cada *Sprint* será de 1 semana, concretamente, de viernes a viernes. Cada viernes se realizará una reunión de apróximadamente 1 hora y 30 minutos que se dividirá en *Sprint Review*, *Sprint Planning* y *Sprint Retrospective*. Se empleará el tiempo que se crea necesario para cada uno de los tres eventos. Debido a la ocupación del docente y del estudiante, no será posible celebrar *Daily Scrums*, sino *Weekly Scrums*, esto se compensará por la corta duración de los *Sprints* y una comunicación continua por correo electrónico.

2.3. Ciclo de vida del proceso de desarrollo

A pesar de que se use *scrum* para el ciclo de vida del producto, para el ciclo de vida del desarrollo *software* se usa el UP (*Unified Process*). El Proceso Unificado de desarrollo del Software es el proceso de Ingeniería de Software de los autores de UML. En este proyecto se emplea ya que en cada iteración se siguen sus cinco grandes *workflows* [51]:

1. Requisitos. Capturan lo que el sistema debe hacer: Documento de especificación de requisitos.

2. Análisis. Mejora y estructura los requisitos: Documento de análisis.
3. Diseño. Realiza los requisitos en la arquitectura del sistema: Documento de diseño.
4. Implementación. Crea el *software*. Documento de implementación.
5. Pruebas. Verifica que la implementación funciona según se desea: Documento de pruebas.

Puede haber *workflows* extra que no establece UP.

2.4. Stakeholders del proyecto

Según el PMI (*Project Management Institute* [64], una posible definición formal de *stakeholders* o partes interesadas es la siguiente: "personas y organizaciones que participan activamente en el proyecto, o cuyos intereses pueden verse afectados positiva o negativamente como resultado de la ejecución del proyecto o de su finalización con éxito". De acuerdo con esta definición, podemos considerar los siguientes *stakeholders*:

1. **Departamento de Psicología Evolutiva y de la Educación de la Universidad de Valladolid**: Es el principal beneficiario del proyecto. Su interés radica en mejorar los procesos de recopilación, análisis y gestión de datos relacionados con su investigacion.
2. **Profesor Tutor**: Desempeña un papel crucial como guía y asesor del estudiante en el desarrollo del proyecto. Su participación implica proporcionar orientación técnica, validar la viabilidad del enfoque propuesto y evaluar el progreso y la calidad del trabajo realizado. El tutor también puede ofrecer sugerencias y recomendaciones para mejorar el proyecto y asegurar que cumpla con los estándares académicos y profesionales requeridos.
3. **Estudiante**: Es el responsable directo de la ejecución del proyecto. Su objetivo es aplicar sus conocimientos y habilidades técnicas para desarrollar una solución informática que satisfaga las necesidades y expectativas.

2.5. Planificación inicial

El proyecto está enmarcado dentro de un TFG (Trabajo de Fin de Grado) de 12 ECTS, que equivale a 300 horas de trabajo. La presentación del proyecto se realizó el miércoles 14 de febrero de 2024 y la *Kick-off meeting* se celebró esa misma semana el viernes 16 de febrero de 2024. De esta forma, el proyecto se dividirá en 16 *Sprints* de una semana, con posibilidad de ampliar a 17.

La carga del trabajo del estudiante no será uniforme a lo largo de los *Sprints* ya que debe compaginar el desarrollo con el TFG del Grado en Estadística y una asignatura universitaria.

A pesar de esto, el estudiante se compromete a invertir un mínimo de horas semanales. Se estima que la carga de trabajo de cada *Sprint* consistirá de un mínimo de 15 horas y un máximo de 50. Con una inversión de aproximadamente 21 horas semanales, se cumplirían las 300 horas de trabajo esperadas. Sin embargo, el trabajo medio esperado del estudiante será de 30 horas semanales, por lo que pronostica que se invertirán 420 horas totales. En la Tabla 2.1 se muestran los *sprints* y eventos planificados:

2.6. Plan de riesgos

El plan de riesgos es un componente fundamental dentro del proceso de gestión de proyectos, diseñado para identificar, evaluar y gestionar los riesgos potenciales que podrían impactar el éxito del proyecto. En un entorno empresarial dinámico y competitivo, donde la incertidumbre es una constante, el reconocimiento y la gestión proactiva de los riesgos son esenciales para minimizar sus efectos adversos y maximizar las oportunidades de éxito [56].

El concepto de riesgo se refiere a cualquier evento o condición incierta que, de materializarse, pueda tener un impacto positivo o negativo en los objetivos del proyecto [58]. Estos riesgos pueden surgir de diversas fuentes, como cambios en los requisitos del cliente, limitaciones de recursos, fallos tecnológicas, entre otros. Para abordar eficazmente los riesgos, es necesario un enfoque estructurado que comprenda varios elementos clave.

Elementos de análisis de riesgos [48]:

1. **Probabilidad:** Evaluar la probabilidad de que un riesgo específico ocurra durante el curso del proyecto. Esto implica considerar factores como la frecuencia histórica de ocurrencia, la naturaleza del riesgo y la eficacia de las medidas de control existentes.
2. **Impacto:** Determinar el impacto potencial que tendría la materialización de un riesgo en los objetivos del proyecto. Esto puede incluir consecuencias financieras, operativas, de calendario o de calidad, entre otras. Cuanto mayor sea el impacto, mayor será la necesidad de gestionar ese riesgo de manera efectiva.
3. **Plan de Mitigación:** Desarrollar estrategias y acciones preventivas para reducir la probabilidad de ocurrencia o minimizar el impacto de los riesgos identificados. Estas medidas pueden incluir la implementación de controles adicionales, la diversificación de recursos, la mejora de procesos o la adopción de tecnologías de respaldo.
4. **Plan de Contingencia:** Establecer planes de acción alternativos para responder de manera efectiva en caso de que los riesgos se materialicen a pesar de las medidas de mitigación. Los planes de contingencia pueden implicar la asignación de recursos adicionales, la activación de procedimientos de emergencia o la implementación de soluciones alternativas para minimizar las pérdidas y mantener el proyecto en curso.

La formulación de estos 4 parámetros es muy importante para saber cómo de probable es que un riesgo suceda, cuan grave es y qué hacer en ese caso. El Plan de Mitigación [57]

Sprint	Reuniones	Duración	Notas
Sprint 0	*Weekly Scrum, Sprint Review y Sprint Planning*	21/02/24 - 01/03/24	
Sprint 1	*Weekly Scrum, Sprint Review y Sprint Planning*	01/03/24 - 08/03/24	
Sprint 2	*Weekly Scrum, Sprint Review y Sprint Planning*	08/03/24 - 15/03/24	
Sprint 3	*Weekly Scrum, Sprint Review, Sprint Planning y Sprint Retrospective*	15/03/24 - 22/03/24	
Sprint 4	*Weekly Scrum, Sprint Review y Sprint Planning*	22/03/24 - 29/03/24	Parte de Semana Santa
Sprint 5	*Weekly Scrum, Sprint Review y Sprint Planning*	29/03/24 - 05/04/24	Parte de Semana Santa
Sprint 6	*Weekly Scrum, Sprint Review y Sprint Planning y Sprint Retrospective*	05/04/24 - 12/04/24	
Sprint 7	*Weekly Scrum, Sprint Review y Sprint Planning*	12/04/24 - 19/04/24	
Sprint 8	*Weekly Scrum, Sprint Review, Sprint Planning y Sprint Retrospective*	19/04/24 - 26/04/24	
Sprint 9	*Weekly Scrum, Sprint Review y Sprint Planning*	26/04/24 - 02/05/24	
Sprint 10	*Weekly Scrum, Sprint Review y Sprint Planning*	03/05/24 - 10/05/24	Fin de compromisos académicos
Sprint 12	*Weekly Scrum, Sprint Review y Sprint Planning y Sprint Retrospective*	10/05/24 - 17/05/24	
Sprint 13	*Weekly Scrum, Sprint Review y Sprint Planning*	17/05/24 - 24/05/24	
Sprint 14	*Weekly Scrum, Sprint Review y Sprint Planning*	24/05/24 - 31/05/24	
Sprint 15	*Weekly Scrum, Sprint Review y Sprint Planning y Sprint Retrospective*	31/05/24 - 07/06/24	
Sprint 16	*Weekly Scrum, Sprint Review y Sprint Planning*	07/06/24 - 13/06/24	
Sprint 17	*Weekly Scrum, Sprint Review y Sprint Planning*	13/06/24 - 20/06/24	Límite de entrega

Tabla 2.1: Planificación inicial de los *sprints*

dicta unas medidas preventivas para intentar minimizar el impacto y/o la probabilidad del riesgo. Por otra parte, el Plan de Contingencia propone una serie de acciones reactivas que llevar a cabo para minimizar el impacto en caso de que algún riesgo se haya materializado.

Las Tablas 2.2-2.11 detallan cada uno de los riesgos identificados en este proyecto.

Riesgo R01	
Título	Baja o incapacidad del estudiante.
Descripción	Si el estudiante se enferma o queda incapacitado, puede causar retrasos significativos en el proyecto y/o afectar su calidad.
Probabilidad	Alta
Impacto	Medio
Plan de mitigación	▪ Llevar una vida saludable. ▪ No realizar actividades que puedan suponer un riesgo para el físico.
Plan de contingencia	▪ Aumentar la carga de trabajo los días siguientes a la recuperación. ▪ Retrasar la fecha de entrega de ciertas tareas. ▪ En caso de ocurrir al final del proyecto, intentar reducir el alcance de este, intentando que afecte lo mínimo posible en el resultado final.

Tabla 2.2: Riesgo R01. Enfermedad o incapacidad del estudiante.

Riesgo R02	
Título	Baja o enfermedad del tutor
Descripción	El tutor puede no estar disponible debido a una enfermedad, ausencia prolongada u otras circunstancias imprevistas. Esto puede causar retrasos en la orientación, dificultades para obtener un asesoramiento adecuado y afectar al correcto progreso del proyecto.
Probabilidad	Baja
Impacto	Bajo
Plan de mitigación	▪ Llevar una vida saludable. ▪ Establecer una comunicación regular con el tutor.
Plan de contingencia	▪ Buscar y acceder a otros recursos de apoyo académico dentro de la universidad. ▪ Asignación de un tutor suplente. ▪ Ser flexible en cuanto a la programación de reuniones y sesiones de orientación.

Tabla 2.3: Riesgo R02. Baja o enfermedad del tutor.

Riesgo R03	
Título	Cambios en los requisitos del cliente.
Descripción	Los cambios constantes en los requisitos del cliente pueden causar retrasos en el proyecto, aumentar los costos, perjudicar la calidad del producto y afectar la satisfacción del cliente.
Probabilidad	Media
Impacto	Alto
Plan de mitigación	▪ Establecer un proceso claro para la gestión de cambios. ▪ Comunicarse de manera regular con el cliente para comprender y validar los requisitos. ▪ Documentar todos los cambios en los requisitos y evaluar su impacto en el proyecto antes de implementarlos.
Plan de contingencia	▪ Mantener un margen de tiempo y recursos adicionales para abordar posibles cambios en los requisitos. ▪ Priorizar los requisitos críticos y posponer los cambios no esenciales. ▪ Hablar con el cliente para acordar un límite en el número y la frecuencia de cambios en los requisitos.

Tabla 2.4: Riesgo R03. Cambios en los requisitos del cliente.

Riesgo R04	
Título	Problemas de integración de tecnologías
Descripción	Combinar tecnologías o componentes puede resultar en incompatibilidades, fallos de interoperabilidad y retrasos en las entregas.
Probabilidad	Media
Impacto	Bajo
Plan de mitigación	▪ Investigar la compatibilidad de antemano. ▪ Realizar pruebas de integración exhaustivas. ▪ Establecer estándares de comunicación claros.
Plan de contingencia	▪ Identificar una tecnología alternativa integrable.

Tabla 2.5: Riesgo R04. Problemas de integración de tecnologías.

Riesgo R05	
Título	Falta de experiencia en el dominio o tecnologías
Descripción	El estudiante puede no poseer conocimiento o la experiencia necesaria en el dominio o en las tecnologías empleadas en el proyecto. Esto puede dificultar la comprensión de requisitos del cliente, la identificación de soluciones efectivas y la toma de decisiones adecuada.
Probabilidad	Media
Impacto	Medio
Plan de mitigación	▪ Realizar una investigación exhaustiva y un estudio detallado del dominio del proyecto para comprender completamente sus aspectos clave. ▪ Diseñar prototipos y hacer pruebas para acostumbrarse a las nuevas tecnologías lo antes posible.
Plan de contingencia	▪ Identificar una tecnología alternativa ya conocida.

Tabla 2.6: Riesgo R05. Falta de experiencia en el dominio o tecnologías.

Riesgo R06	
Título	Limitación de tiempo debido a otras responsabilidades académicas.
Descripción	El estudiante puede no dedicar suficiente tiempo en ciertos lapsos de tiempo debido a otras responsabilidades académicas. Esto puede provocar retrasos o afectar la calidad del trabajo al aumentar el estrés y la presión sobre el estudiante.
Probabilidad	Alta
Impacto	Medio
Plan de mitigación	▪ Realizar la planificación teniendo en cuenta la carga de trabajo que tendrá el estudiante en cada *sprint*. ▪ Ser flexible en cuanto a la programación de reuniones y plazo de entregas para adaptarse a los compromisos del estudiante.
Plan de contingencia	▪ Reevaluación y ajuste de los plazos de entrega. ▪ Identificar las tareas menos críticas que pueden aplazarse. ▪ Aumentar la carga de trabajo en *sprints* posteriores cuando se tenga más disponibilidad.

Tabla 2.9: Riesgo R06. Limitación de tiempo debido a otras responsabilidades académicas.

Riesgo R07	
Título	Incumplimiento de plazos.
Descripción	El estudiante puede no cumplir con los tiempos establecidos para completar las tareas del proyecto, lo que puede resultar en retrasos, incumplimiento de objetivos y/o pérdida de calidad del producto.
Probabilidad	Alta
Impacto	Medio
Plan de mitigación	▪ Desarrollar una planificación rigurosa del proyecto. ▪ Realizar estimaciones precisas y realistas para cada tarea. ▪ Supervisar regularmente el progreso del proyecto y compararlo con el plan establecido. ▪ Incluir márgenes de tiempo adicionales en el plan de proyecto para imprevistos o tareas que puedan llevar más tiempo del esperado.
Plan de contingencia	▪ Reevaluar los plazos detenidamente e identificar las áreas donde sea posible ajustar las fechas de entrega para recuperar tiempo perdido. ▪ Realizar una replanificación del proyecto con los recursos actuales teniendo en cuenta los retrasos acumulados. ▪ Reducir el alcance del proyecto y enfocarse en las tareas críticas y de alto valor que contribuyan a los objetivos del proyecto.

Tabla 2.7: Riesgo R07. Incumplimiento de plazos.

Riesgo R08	
Título	Mala planificación.
Descripción	La mala planificación puede surgir debido a una deficiente estimación de recursos, tiempo o alcance. Puede causar retrasos en las entregas, sobrecarga de trabajo y agotamiento del estudiante.
Probabilidad	Media
Impacto	Alto
Plan de mitigación	▪ Realizar un análisis detallado de los requisitos del proyecto (objetivos, recursos y restricciones). ▪ Consultar decisiones importantes con el tutor del proyecto. ▪ Realizar estimaciones realistas de tiempo, recursos y esfuerzo para cada actividad.
Plan de contingencia	▪ Evaluar el impacto de la mala planificación y reasignar recursos, si es posible, para abordar áreas críticas afectadas. ▪ Revisar y reajustar el plan de proyecto.

Tabla 2.8: Riesgo R08. Mala planificación.

Riesgo R09	
Título	Pérdida de datos.
Descripción	El estudiante puede sufrir una pérdida, corrupción o eliminación accidental de datos o archivos importantes. Esto puede deberse a fallos informáticos, errores humanos o ataques cibernéticos. La pérdida de datos puede causar retrasos inesperados, pérdida de información crítica y afectar la integridad de los entregables.
Probabilidad	Baja
Impacto	Medio
Plan de mitigación	▪ Utilizar herramientas de *backup* y control de versiones ▪ Utilizar almacenamiento en la nube para almacenar archivos importantes del proyecto.
Plan de contingencia	▪ Identificar la causa de la pérdida de datos, evaluar la importancia de la pérdida y evitar que suceda de nuevo. ▪ Intentar recuperar los datos perdidos.

Tabla 2.10: Riesgo R09. Pérdida de datos.

Riesgo R10	
Título	Fallos o imposibilidad de uso del equipo de trabajo
Descripción	El equipo, herramientas o recursos necesarios para llevar a cabo el proyecto pueden no estar disponibles, ya sea por averías o limitaciones de acceso. Puede desembocar en pérdida de productividad y retrasos.
Probabilidad	Baja
Impacto	Bajo
Plan de mitigación	▪ Realizar un mantenimiento regular y preventivo del equipo de trabajo. ▪ Mantener actualizados tanto el software como el hardware utilizados en el proyecto.
Plan de contingencia	▪ Identificar la causa del fallo lo más rápido posible. ▪ Implementar soluciones temporales en caso de ser posible. ▪ Documentar el incidente y buscar ayuda externa.

Tabla 2.11: Riesgo R10. Fallos o imposibilidad de uso del equipo de trabajo.

Teniendo en cuenta la Tabla 2.12 de riesgos, se puede ordenar los riesgos por su severidad:

		Impacto		
		Bajo	Medio	Alto
Probabilidad	Baja	Riesgo trivial (T)	Riesgo Tolerable (RT)	Riesgo Moderado (RM)
	Media	Riesgo Tolerable (RT)	Riesgo Moderado (RM)	Riesgo Importante (RI)
	Alta	Riesgo Moderado (RM)	Riesgo Importante (RI)	Riesgo Intolerable (IN)

Tabla 2.12: Matriz de riesgos

Entonces, se pueden ordenar los riesgos en función de esta nueva magnitud:

ID	Nombre	Severidad
R02	Baja o enfermedad del tutor.	T
R10	Fallos o imposibilidad de uso del equipo de trabajo.	T
R04	Problemas de integración de tecnologías.	RT
R09	Pérdida de datos.	RT
R05	Falta de experiencia en el dominio o tecnologías.	RM
R01	Baja o incapacidad del estudiante.	RI
R03	Cambios en los requisitos del cliente.	RI
R07	Incumplimiento de plazos.	RI
R08	Mala planificación.	RI
R06	Limitación de tiempo debido a otras responsabilidades académicas.	RI

Tabla 2.13: Tabla de severidad de riesgos.

Cada celda de la Tabla 2.13 contiene una evaluación de la severidad del riesgo [66], que puede ser cualitativa (usando palabras descriptivas como "trivial", "tolerable", "moderado", ïmportante", ïntolerable").

Se trata de una herramienta útil para priorizar los riesgos identificados en el proyecto. Los riesgos que tienen una alta probabilidad de ocurrencia y un alto impacto en el proyecto son considerados de alta severidad y requieren una atención especial en términos de planificación de mitigación y contingencia. Por otro lado, los riesgos que tienen una baja probabilidad de ocurrencia y un impacto bajo pueden no requerir medidas de mitigación inmediatas, pero aún así deben ser considerados para evitar que se conviertan en problemas más adelante en el proyecto.

En resumen, la tabla de severidad de riesgos proporciona una forma estructurada de evaluar y priorizar los riesgos del proyecto, lo que permite al estudiante tomar decisiones efectivamente en caso de que sea necesario.

2.7. Plan de presupuestos

2.7.1. Presupuesto simulado

A pesar de que el trabajo se llevará a cabo por los propios medios del estudiante y sin la intervención de una empresa, se realizará un plan de presupuestos detallado como si se tratara de un proyecto de desarrollo de una. Para ello se efectuará estimación financiera realista. Al elaborar un plan de presupuestos, se considerarán todos los posibles costos asociados, tales

como la oficina, el tiempo invertido, el uso de herramientas y *software* específicos, y cualquier otro gasto relacionado con el desarrollo del proyecto. Este ejercicio no solo proporcionará una visión clara de los recursos necesarios, sino que también facilitará una gestión eficiente del proyecto.

El primer costo a tener en cuenta y, el más importante, es el del personal. Se ha considerado que el rol más indicado para desarrollar una aplicación web es el del *Full-Stack Developer* cuyo salario medio en España según *Glassdoor* [27] es de aproximadamente **2.500€** brutos mensuales sin tener en cuenta su grado de experiencia. Teniendo en cuenta que la duración del desarrollo del proyecto es de 4 meses, el salario total a percibir del desarrollador sería de **10.000€.** Se considera un trabajador de 22 años, como el estudiante, Ingeniero o licenciado, residente en Castilla y León, sin discapacidad y sin ninguna persona a su cargo. Con estos datos, se obtienen los siguientes datos a nivel anual: Se le retenerá el 16,27 % del salario bruto [29] para la cuota del IRPF (Impuesto sobre la Renta de las Personas Físicas), es decir, 4881€. Teniendo en cuenta que el desarrollador es un ingeniero, su base de máxima de cotización es de 4.720,50€ [28], paga un 4,70 % de la base para la Seguridad Social, 221,87€. Por último, se le retenerán 73.17€, un 1.55 % de la misma base de cotización para el desempleo. Teniendo en cuenta que el desarrollador trabajará 4 meses, de los 10.000€ el desarrollador percibirá **7192,84€** netos.

El desarrollador necesita un espacio para trabajar, se ha decidido que por la brevedad de su contratación se abonará mensualmente una tarifa plana mensual de *coworking* en Valladolid para que pueda disfrutar horarios flexibles. Su coste será de 155€ mensuales [30], un total de **620€** para todo el proyecto.

El trabajador necesitará también un equipo que le proporcione la empresa, se ha decidido que la mejor opción es el *Dell XPS 15* [31] con un precio total de 2236€. Teniendo en cuenta que la vida media de los portátiles Dell suele ser de seis años [32], este se amortizará a lo largo de dichos seis años, por lo que su amortización mensual será de 31,06€. Teniendo en cuenta que el desarrollo del proyecto consiste de 4 meses, su coste asciende a **124,22€.**

Por último, aparte del equipo de trabajo, el desarrollador necesita licencias de *software* con los que trabajar. Tanto Python [7] como *Visual Studio Code* [16] y sus paquetes y extensiones son *open source* por lo que su coste es nulo. Sin embargo, se necesitan las licencias de *Visual Paradigm* [3] y *Balsamiq Wireframes* para el análisis, diseño y documentación de la aplicación. La licencia estándar anual de *Visual Paradigm* cuesta 212,28€ [34], de lo que correspondería **70,76€** al periodo de desarrollo. Balsamiq ofrece una licencia de *Balsamiq Cloud* por la que solo es necesario pagar 8,38€ mensuales para poder trabajar con un proyecto mediante internet, lo que sumaría **33,52€** para 4 meses. Por último, la herramienta de comunicación *Microsoft Teams* [20] requiere de una licencia oficial de *Microsoft 365* [35], cuyo precio es de 10€ mensuales, por lo que su coste total sería de **40€**. El resto de tecnologías requeridas son de uso gratuito.

En total, el presupuesto estimado asciende a una cifra de **10.888,50€**. En la Tabla 2.14 se muestra un resumen de todos los costes.

Concepto	Precio mensual	Meses	Total
Desarollador *Full-Stack*	2500€	4	10000€
Espacio de *coworking*	155€	4	620€
Equipo de trabajo *Dell*	31,06€	4	124,22€
Licencia *Visual Paradigm*	17,69€	4	70,76€
Licencia *Balsamiq Cloud*	8,38€	4	33,52€
Licencia *Microsoft Teams*	10€	4	40€
COSTE TOTAL			**10.888,50€**

Tabla 2.14: Desglose del presupuesto simulado.

2.7.2. Presupuesto real

En realidad, el desarrollador del proyecto es un estudiante de la Universidad de Valladolid realizando el Trabajo de Fin de Grado, por lo que no tiene coste monetario. En cuanto al espacio de *coworking*, tampoco acarraerá ningún coste pues el estudiante trabajará en todo momento desde su domicilio o desde uno de los numersosos espacios ofrecido por la Universidad de Valladolid.

El estudiante trabaja en un portátil *Huawei Matebook 13* [36] adquirido en 2019 por 1200€. Para tener un punto de vista conservador, se supone que le resta un año de vida al portátil, de esta forma la longitud de su vida útil sería 6 años y su amortización mensual consiste de 16,66€ mensuales. Entonces, el precio del equipo de trabajo a lo largo de los cuatro meses es de **66,66€** en total.

En el presupuesto simulado no se ha tenido en cuenta el consumo de electricidad pues este corre a cuenta del dueño del espacio de *coworking*. En este caso, el estudiante trabajará mayormente desde casa por lo que es necesario presupuestar el uso de luz. Se ha planificado inicialmente en la sección 2.5, que el estudiante empleará 420 horas de su tiempo para el desarrollo del proyecto, lo que supone unas 3 horas y 45 minutos de trabajo diarios de media teniendo en cuenta los 16 *sprints* planificados. El precio medio del kWh se encuentra en 0,1245€/kWh [37]. Suponiendo que todas las horas de trabajo se llevan a cabo en el domicilio del estudiante y que el consumo de enegería del portátil es de 45W, se puede calcular [38] que su coste medio mensual es de 0,68€, por lo que la suma de los cuatro meses de trabajo alcanza los **2,72€**.

Por último, el estudiante no necesitará realizar ninguna inversión en *software* necesario pues la Universidad de valladolid tiene licencia de campus para *Visual Paradigm*, *Balsamiq Wireframes* y *Microsoft 365*. El resto de herramientas son gratis o de código abierto como ya se ha mencionado en el presupuesto simulado.

En la Tabla 2.15 se reúnen todos los costes presupuestados:

Concepto	Precio mensual	Meses	Total
Equipo de trabajo *Huawei*	16,66€	4	66,66€
Electricidad	0,68€	4	2,72€
COSTE TOTAL			**69,38€**

Tabla 2.15: Desglose del presupuesto real.

2.8. *Product Backlog* inicial

En el desarrollo ágil de software, el *Product Backlog* es una herramienta fundamental que sirve como lista de prioridades de todas las tareas y características que se deben implementar en el proyecto.

Una épica es una gran tarea u objetivo que puede desglosarse en tareas más pequeñas y manejables llamadas **historias de usuario**. Un *Product Backlog* en el que se trabaja con *scrum* está constituido por épicas, que pueden dividirse en historias de usuario [69].

Las épicas y las historias de usuario siguen la misma estructura:

- "Como *stakeholder* quiero *funcionalidad* para *objetivo*".

Estas aportan una visión clara de lo que se quiere lograr y permiten una gran flexibilidad en la planificación y ejecución.

El *Product Backlog* es gestionado por el *Product Owner*, en este caso, el estudiante. Este se encarga de priorizar los elementos basándose en el valor que aportan al producto y las necesidades del negocio. Los items del *Backlog*, conocidos como *Product Backlog Item* (PBI), representan las características, requisitos, mejoras, correcciones de errores o cualquier otra tarea que debe realizarse para desarrollar el producto

El Product Backlog que se obtiene inicialmente se muestra en la Tabla 2.16:

ID	Descripción Épica
EP01	Como usuario quiero poder crear una cuenta e iniciar sesión para acceder a mis funcionalidades personalizadas.
EP02	Como docente quiero ser capaz de crear y modificar actividades para poder gestionar y organizar mejor las tareas relacionadas con mi curso.
EP03	Como docente, quiero poder asignar tareas a mis estudiantes y realizar un seguimiento para facilitar su aprendizaje.
EP04	Como docente quiero ser capaz de crear y gestionar estados de ánimo para que los estudiantes puedan reflejar como se sienten.
EP05	Como docente quiero ser capaz de supervisar a todos mis alumnos y sus estados de ánimo actuales y pasados para poder identificar patrones de comportamiento, detectar posibles problemas emocionales y brindar el apoyo necesario para promover un ambiente de aprendizaje positivo y saludable.
EP06	Como estudiante quiero ser capaz de ver todas mis actividades pendientes y sus detalles y poder resolverlas.
EP07	Como estudiante quiero ser capaz de ver todos los estados de ánimo registrados, sus detalles para indicar como me siento.

Tabla 2.16: *Product Backlog* inicial.

2.9. *Product Backlog* final

A lo largo de todos los *sprints* de desarrollo y gracias al contacto con los *stakeholders* del proyecto, se decidió añadir nueva funcionalidad y, en consecuencia, la introducción de nuevos requisitos en forma de épicas al *Product Backlog*. Estas modificaciones al *Product Backlog* se realizaron en las reuniones celebradas semanalmente, es decir, entre un *sprint* y otro.

Estos cambios consistieron en la adición de dos nuevos tipos de usuario a la aplicación y sus respectivas funcionalidades, por lo que se añadieron dos nuevas épicas: **EP08** y **EP09**.

El Product Backlog que se obtiene finalmente se muestra en la Tabla 2.17:

ID	Descripción Épica
EP01	Como usuario quiero poder crear una cuenta e iniciar sesión para acceder a mis funcionalidades personalizadas.
EP02	Como docente quiero ser capaz de crear y modificar actividades para poder gestionar y organizar mejor las tareas relacionadas con mi curso.
EP03	Como docente, quiero poder asignar tareas a mis estudiantes y realizar un seguimiento para facilitar su aprendizaje.
EP04	Como docente quiero ser capaz de crear y gestionar estados de ánimo para que los estudiantes puedan reflejar como se sienten.
EP05	Como docente quiero ser capaz de supervisar a todos mis alumnos y sus estados de ánimo actuales y pasados para poder identificar patrones de comportamiento, detectar posibles problemas emocionales y brindar el apoyo necesario para promover un ambiente de aprendizaje positivo y saludable.
EP06	Como estudiante quiero ser capaz de ver todas mis actividades pendientes y sus detalles y poder resolverlas.
EP07	Como estudiante quiero ser capaz de ver todos los estados de ánimo registrados, sus detalles para indicar como me siento.
EP08	Como familia quiero ser capaz de ver todas las actividades pendientes para familiares, sus detalles para poder resolverlas.
EP09	Como administrador quiero poder ver todas las actividades registradas en el sistema y publicarlas para que todos los estudiantes tengan asignada esa actividad y puedan hacerla.

Tabla 2.17: *Product Backlog* final.

Capítulo 3

Tecnologías utilizadas

3.1. Introducción

En el desarrollo de este Trabajo de Fin de Grado (TFG), se han empleado diversas tecnologías que han facilitado la construcción y gestión del proyecto. La selección de estas herramientas ha sido cuidadosamente realizada con el objetivo de asegurar la eficiencia y la calidad del producto final. Este capítulo está dedicado a describir las tecnologías que se han utilizado, sin entrar en detalles sobre la programación o la base de datos, ya que estos aspectos serán tratados en el capítulo de implementación.

En primer lugar, se presentarán las herramientas de desarrollo utilizadas para la gestión y desarrollo del código. A continuación, se describirán las plataformas que han servido para el análisis, diseño y documentación del desarrollo del proyecto. También se abordarán las herramientas de prueba y validación que han garantizado la calidad del software. Por último se presentará el software empleado para la comunicación.

3.2. Herramientas para el desarrollo y gestión del código

3.2.1. Visual Studio Code

Visual Studio Code [16], comúnmente abreviado como *VS Code*, es un editor de código fuente desarrollado por *Microsoft* [87]. Es una herramienta poderosa y versátil, ampliamente utilizada en la industria del desarrollo de software gracias a sus características avanzadas y su enfoque en la productividad del desarrollador.

VS Code es conocido por ser un editor ligero pero altamente extensible. Ofrece soporte nativo para una amplia gama de lenguajes de programación y tecnologías gracias a su sistema de extensiones, que permite a los desarrolladores personalizar y ampliar la funcionalidad del

editor según sus necesidades específicas. Entre las extensiones que han sido empleadas se pueden destacar las siguientes:

3.2.1.1. Python

La extensión de ***Python*** para Visual Studio Code es una herramienta fundamental para desarrolladores que trabajan con el lenguaje de programación Python. Desarrollada y mantenida por *Microsoft* [87], esta extensión proporciona una amplia gama de características que mejoran significativamente la experiencia de desarrollo en Python. De esta extensión se han empleado varias funcionalidades como: soporte para la depuración, *Linting*, *IntelliSense* y soporte para entornos virtuales.

3.2.1.2. Jinja

La extensión de ***Jinja*** para *Visual Studio Code* es una herramienta esencial para desarrolladores que trabajan con plantillas Jinja en sus proyectos web. Algunas de las funcionalidades clave incluyen:

1. **Sintaxis resaltada:** Facilita la lectura y comprensión del código.
2. **Autocompletado inteligente:** Ofrece sugerencias de autocompletado inteligentes basadas en el contexto, lo que agiliza la escritura del código y reduce los .errores.
3. **Integración con extensiones de Python:** e integra de forma nativa con la extensión de Python para Visual Studio Code, lo que permite una experiencia de desarrollo fluida.
4. **Comprobación de errores en tiempo real:** Permite previsualizar el resultado de las plantillas Jinja directamente en el editor, lo que facilita la depuración y la visualización del contenido generado.

3.2.1.3. Python Enviroment Manager

La extensión ***Python Environment Manager*** [88] para *Visual Studio Code* es una herramienta que facilita la gestión de entornos virtuales de Python dentro del Entorno de Desarrollo Integrado (IDE) de VS Code. Desarrollada por el equipo de *Microsoft* [87], esta extensión ayuda a los desarrolladores a crear, activar y administrar entornos virtuales de Python de manera más eficiente. Se ha desarrollado toda la aplicación en un entorno virtual para gestionar las dependencias y el entorno de desarrollo.

3.2.1.4. Python Debugger

La extensión ***Python Debugger*** [89] para *Visual Studio Code* es una herramienta invaluable para desarrolladores de Python que les permite depurar sus programas de manera

efectiva directamente desde el IDE. Desarrollada por el equipo de Python para *Visual Studio Code*, esta extensión proporciona un conjunto de características avanzadas que simplifican y agilizan el proceso de depuración. La integración del *debugger* se integra perfectamente en el IDE, ha permitido al estudiante pausar la ejecución de su código en puntos específicos, examinar el estado del programa y realizar acciones como inspeccionar variables, evaluar expresiones, inspeccionar la pila de llamadas o avanzar en la ejecución paso a paso.

3.2.1.5. Pylance

La extensión ***Pylance***[90] para *Visual Studio Code* es una poderosa herramienta de análisis estático y autocompletado para Python. Desarrollada por el equipo de *Microsoft* [87], *Pylance* ofrece características avanzadas que mejoran significativamente la experiencia de desarrollo de *Python en VS Code*. Se ha empleado esta extensión para obtener sugerencias inteligentes, detección de errores y de tipos y advertencias en tiempo real mediante se programa.

3.2.1.6. Python Indent

Python Indent[91] ayuda a mantener una estructura de indentación coherente y precisa en archivos de Python. Esta extensión automatiza el proceso de indentación del código Python, lo que facilita la escritura y lectura del código.

3.2.1.7. Prettier

Prettier [93] es una herramienta popular que ayuda a formatear automáticamente el código en una variedad de lenguajes de programación, incluido Python. Desarrollada por el equipo de Prettier, esta extensión ofrece una forma rápida y sencilla de mantener un estilo de codificación consistente y legible en proyectos de software. Se ha empleado también para obtener atajos de teclados que aumenten la eficiencia y comodidad al programar.

3.2.2. GitLab

GitLab [94] es una plataforma de desarrollo de software que proporciona un conjunto completo de herramientas para la gestión del ciclo de vida del desarrollo de aplicaciones. En esencia, *GitLab* se centra en la gestión de repositorios de código, el seguimiento de problemas (issues), la integración continua, la entrega continua (CI/CD), la colaboración entre equipos y mucho más. Es conocido por ser una alternativa de código abierto a GitHub, pero con características adicionales que abarcan más allá de la gestión de repositorios de código. Se ha empleado el GitLab propio de la Universidad de Valladolid [95] para gestionar las versiones con *Git*.

3.2.2.1. Git

Git [96] es un sistema de control de versiones de código abierto diseñado para gestionar el desarrollo de proyectos de software de forma eficiente y colaborativa. Permite a los equipos de desarrollo llevar un registro de los cambios realizados en el código fuente a lo largo del tiempo, lo que facilita la colaboración, la gestión de versiones y la integración de nuevas características en el proyecto.

La estructura de Git se basa en varios elementos:

- **Repositorios:** Un repositorio *Git* es una colección virtual de archivos y carpetas asociados a un proyecto, junto con un historial completo de cambios realizados en esos archivos.
- **Ramificación:** También llamado *branching*, *Git* permite crear ramas independientes del historial principal del proyecto, lo que permite a los desarrolladores trabajar en nuevas características o arreglos de errores sin interferir con el trabajo en la rama principal (generalmente llamada "master.º "main"). Cada rama se crea a través de otra ya existente con el estado en el que se encontrase la rama "madre". Se puede ver un ejemplo de un sistema de ramas en la Figura 3.1.

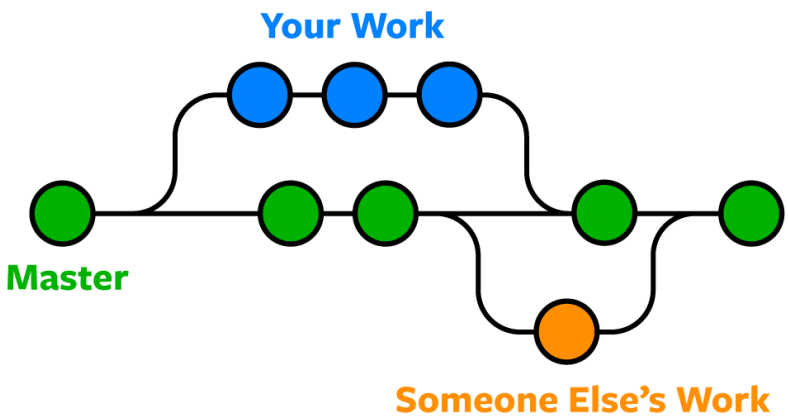

Figura 3.1: Sistema de ramas de *Git*.

Una vez que se completa el trabajo en una rama, los cambios pueden fusionarse de vuelta a la rama principal mediante una operación de fusión, combinando los cambios de una rama en otra. Esta operación se llama *merge*.

Trabajo en local y en remoto

Trabajar en local con *Git* implica realizar cambios en una copia del repositorio almacenada en la propia máquina del desarrollador. Aquí, se pueden crear, modificar y eliminar archivos, así como crear ramas para trabajar en funcionalidades específicas sin afectar al

código principal. Los cambios realizados se registran mediante *commits*, que capturan el estado completo de los archivos en un momento dado. Cada commit tiene un identificador único que permite hacer referencia a él en el historial. Al interactuar con el repositorio remoto, se pueden enviar (*push*) los *commits* realizados en la copia local al repositorio remoto, actualizando así su estado. *Pull* permite traer los *commits* del repositorio remoto a la copia local, fusionándolos automáticamente si es posible. Por último, *fetch* descarga los commits del repositorio remoto al repositorio local, pero no los fusiona automáticamente, permitiendo al desarrollador revisarlos antes de integrarlos en su copia local mediante una fusión manual. Estas operaciones de *push*, *pull*, *fetch* y *commit* son fundamentales para gestionar la sincronización entre los repositorios locales y remotos, facilitando la colaboración entre los miembros del equipo y el seguimiento del progreso del proyecto.

Para cada historia de usuario o grupo de historias de usuario, el estudiante crea una nueva rama en local y no la fusiona de vuelta con la rama *main* hasta que no haya terminado de implementar la historia de uso y probarla.

No se empezó a emplear el repositorio remoto hasta la mitad del proyecto. Como el desarrollo del proyecto se llevó a cabo por un único estudiante, no fue necesario crear ramas en el repositorio remoto.

fork y su uso en el proyecto

Un *fork* de un proyecto en el contexto de *Git* y sistemas de control de versiones en general, se refiere a la acción de copiar un repositorio existente y alojarlo en otro lugar, ya sea en la misma plataforma de alojamiento de repositorios o en una diferente. Este nuevo repositorio es una copia independiente del original y puede ser modificado y gestionado de manera independiente.

Al hacer un fork, se pueden realizar cambios en la copia del proyecto y, si se quiere, proponer esos cambios de vuelta al proyecto original a través de un proceso de solicitud de extracción (*pull request*), comúnmente conocido como *PR*.

Para poner la aplicación en producción, se ha hecho un *fork* del repositorio del proyecto original como parte del proceso de despliegue. Esto permite la implementación de cambios específicos o la necesidad de mantener un control más granular sobre las actualizaciones y parches aplicados a la aplicación.

3.3. Herramientas para análisis, diseño y documentación

3.3.1. Visual Paradigm

Visual Paradigm [3] es una herramienta de modelado y gestión de proyectos que proporciona un entorno integrado para el diseño de software, la creación de diagramas y la gestión de procesos de desarrollo. Es ampliamente utilizada para crear diagramas UML [51] (*Unified Modeling Language*) y otros tipos de diagramas visuales que ayudan a los equipos de desarrollo a planificar, diseñar y comunicar aspectos del software. En este proyecto se

ha utilizado la versión ***Visual Paradigm Standard 17.0*** para la creación de diagramas mostrados en los siguientes capítulos. Se ha podido utilizar esta herramientas gracias a la licencia de campus proporcionada por la Universidad de Valladolid.

3.3.2. Overleaf

Overleaf [4] es una plataforma colaborativa en línea para la escritura y edición de documentos LaTeX. Diseñada para facilitar la creación de documentos científicos, técnicos y académicos. Se ha empleado *Overleaf* porque simplifica el proceso de escritura en LaTeX, haciendo que sea accesible incluso para aquellos con poca experiencia previa y gracias a la posibilidad de invitar del tutor del TFG al proyecto para su revisión.

3.3.3. Balsamiq Wireframes

Balsamiq Wireframes es una herramienta de diseño de interfaces de usuario que permite crear bocetos y prototipos de manera rápida y sencilla. Está diseñada para simular la experiencia de dibujar en una pizarra, proporcionando un conjunto de elementos y controles que facilitan la creación de maquetas visuales de aplicaciones web y móviles. *Balsamiq* es especialmente útil en las primeras etapas del diseño, ya que permite visualizar y probar diferentes ideas de diseño sin necesidad de escribir código. Por ello, se ha empleado para realizar *mockups* de todas las pantallas en la etapa de diseño, antes de implementarlas. Los *mockups* se han realizado con la versión ***Balsamiq Wireframes 4.7.5***.

3.3.4. Boxy SVG

Boxy SVG [6] es una herramienta de edición de gráficos vectoriales escalables (SVG, por sus siglas en inglés) que permite a los usuarios crear, editar y manipular gráficos vectoriales de forma intuitiva y eficiente. Se ha empleado la versión ***Boxy SVG 4.33.0 online*** para realizar retoques a los gráficos SVG obtenidos con *Visual Paradigm*.

3.4. Herramientas de comunicación y organización

3.4.1. Trello

Trello [15] es una herramienta de gestión de proyectos y tareas basada en la web que utiliza un sistema de tableros, listas y tarjetas para organizar y priorizar el trabajo de manera visual y colaborativa. Cada tablero representa un proyecto o área de trabajo, mientras que las listas dentro de los tableros pueden representar diferentes etapas del flujo de trabajo, como "Por hacer", "En progreso" y "Completado". Las tarjetas en las listas son unidades individuales

de trabajo, donde se pueden agregar descripciones, comentarios, fechas límite, etiquetas y archivos adjuntos.

En la Figura 3.2 se muestra el tablero de *Trello* en uno de los *sprints* iniciales del proyecto.

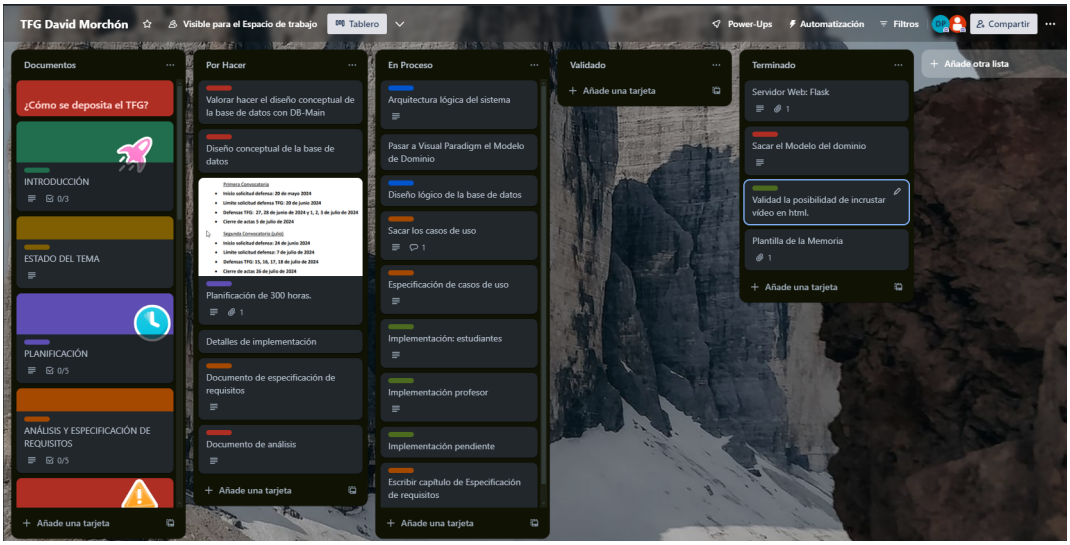

Figura 3.2: Tablero de Trello.

3.4.2. Microsoft Outlook

Microsoft Outlook [19] es un cliente de correo electrónico y gestor de información personal que forma parte del conjunto de aplicaciones de *Microsoft Office*. Outlook ofrece una amplia gama de funciones, incluyendo envío y recepción de correos electrónicos, gestión de calendarios, organización de contactos, seguimiento de tareas y toma de notas. En este proyecto se ha empleado su servicio de correo electrónico como principal medio de comunicación con el tutor cuando era necesario resolver una duda y no se quería esperar hasta la siguiente reunión.

3.4.3. Microsoft Teams

Microsoft Teams [20] es una plataforma de comunicación y colaboración que permite a los equipos trabajar juntos a través de chat, videoconferencias, almacenamiento de archivos y integración con aplicaciones de *Microsoft 365*. Es ampliamente utilizada para facilitar la comunicación en tiempo real y la colaboración eficiente en entornos empresariales y educativos. En este proyecto se ha utilizado como medio para el envío de apuntes, bibliografía o gráficos del tutor al estudiante.

3.5. Herramientas de pruebas

A lo largo del desarrollo del proyecto, en cada iteración se realizaban pruebas del nuevo software desarrollado en dicha iteración y su integración con el software desarrollado hasta el momento. Para ello se utilizaron dos herramientas:

3.5.1. Postman

Postman [21] es una herramienta de desarrollo que facilita la creación, prueba y documentación de APIs. Permite a los desarrolladores enviar solicitudes HTTP, realizar pruebas automáticas y visualizar las respuestas de las APIs de manera intuitiva. Postman es ampliamente utilizado para el desarrollo de servicios web, ya que simplifica la interacción con las APIs, facilita la detección de errores y mejora la colaboración entre los miembros del equipo mediante colecciones compartidas y entornos configurables. En el contexto se las pruebas de la aplicación, se ha empleado en las pruebas unitarias realizadas manualmente para enviar peticiones HTTP ***GET*** y ***POST***.

3.5.2. Flask debugger

El depurador de *Flask* o ***Flask debugger*** [22] es una herramienta integrada en el *framework* de desarrollo web Flask que facilita la identificación y corrección de errores en aplicaciones Flask durante el proceso de desarrollo. Se integra perfectamente con el depurador de Python y permite inspeccionar variables, rastrear la ejecución del código, y ver mensajes de error detallados directamente en el navegador web. Esta funcionalidad ayuda a los desarrolladores a detectar y solucionar problemas de manera eficiente, lo que acelera el proceso de desarrollo y mejora la calidad del software creado con Flask.

Capítulo 4

Especificación de requisitos

En este capítulo se detallan las funcionalidades y requisitos del sistema obtenidos con la participación del cliente, con el objetivo de obtener una guía clara para el desarrollo de la aplicación. En primer lugar, se ofrece una descripción detallada del sistema, donde se destacan sus propósitos y alcances. A continuación, se presenta un glosario de términos clave, esencial para establecer un lenguaje común entre los clientes y los desarrolladores. Seguidamente, se especifican los requisitos funcionales, de información y no funcionales del sistema, detallando las características y comportamientos esperados. Finalmente, se incluye un modelo de casos de uso, que comprende la identificación y descripción de los actores involucrados, así como la especificación detallada de los casos de uso que representan los escenarios de uso de los actores en el sistema. Este conjunto inicial de información proporciona una base sólida para el desarrollo de AcogeCYL. Sin embargo, no debe considerarse como un manual que debe seguirse rigurosamente, pues en *scrum* prevalece la adaptabilidad y colaboración con el cliente, para garantizar su adecuación a las necesidades cambiantes y expectativas de los usuarios.

4.1. Descripción detallada del sistema

Acoge-CYL es un proyecto que se dedica a la ayuda e integración de personas refugiadas y migrantes en Castilla y León, España. La organización proporciona diversos servicios y programas para facilitar la inclusión social y laboral de estas personas en la comunidad. Entre sus actividades, se encuentran el apoyo legal, la formación en idiomas, la orientación laboral, y el acompañamiento en el acceso a servicios básicos como la salud y la educación. Acoge-CYL trabaja en colaboración con entidades públicas y privadas para promover una integración efectiva y sostenible de los refugiados y migrantes en la región [85].

El propósito de AcogeCYL como software educativo es facilitar la creación, gestión y seguimiento de actividades educativas propuestas por docentes para la integración de alumnos refugiados, inmigrantes y de incorporación tardía mediante la provisión de recursos educativos

estructurados, el seguimiento del bienestar emocional y la comunicación eficiente entre todas las partes interesadas. El sistema permitirá la creación y gestión de actividades educativas, la asignación y seguimiento de actividades a los estudiantes. También se quiere permitir la monitorización y gestión de los estados emocionales de los estudiantes para ofrecerles apoyo cuando sea necesario.

Por parte de los estudiantes, la aplicación ofrecerá una interfaz para que completen y sigan sus actividades asignadas y puedan registrar sus emociones en cualquier momento, explicando el porqué. Los familiares tendrán la oportunidad de participar en el aprendizaje del estudiante a través de las actividades familiares.

Se considera muy importante que la herramienta sea multilenguaje para que el idioma no sea una barrera. Sin embargo, gracias a las capacidades de los navegadores modernos, no es estrictamente necesario implementar esta funcionalidad de forma nativa. Los navegadores actuales incorporan traductores automáticos que permiten a los usuarios traducir el contenido a su idioma preferido.

4.2. Glosario de términos

En esta sección, se presenta un compendio de términos clave utilizados en el contexto del proyecto. Este glosario tiene como objetivo establecer un marco de referencia común y facilitar la comprensión de conceptos específicos tanto para el equipo de desarrollo como para los usuarios finales.

- **Actividad**: En el contexto de este proyecto, una actividad se refiere a una tarea, ejercicio o recurso de aprendizaje diseñado por el docente para ser completado por los estudiantes. Las actividades pueden incluir ejercicios prácticos, lecturas, imágenes, vídeo, audios, entre otros, y están destinadas a promover la integración social del estudiante en el proceso educativo. Una actividad puede tener o no preguntas asociadas pero en todo caso el estudiante deberá de indicar que ha terminado una vez lo haya hecho.
- **Estado de Ánimo**: El estado de ánimo representa la disposición emocional o mental predominante de un individuo en un momento dado. En el contexto de este proyecto, los estados de ánimo se utilizan para que los estudiantes expresen sus emociones y sentimientos en relación con su experiencia, ya sea escolar o extraescolar, lo que puede facilitar la comprensión y el apoyo emocional [86].
- **Diario Emocional**: Es una herramienta de seguimiento de los Estados de Ánimo de cada uno de los estudiantes. Se trata de un historial o registro las emociones de cada uno de los estudiantes, incluyendo el instante del registro y la razón aportada por el estudiante. En el contexto educativo, el diario emocional puede utilizarse para identificar patrones emocionales, comprender mejor las reacciones a diferentes situaciones y fomentar el bienestar emocional [86].

- **Asignación**: Una asignación se refiere a la acción de asignar una actividad específica a un estudiante. Una asignación tiene dos estados: pendiente y finalizada. En caso de estar finalizada, se guarda la fecha de finalización para que el docente pueda comprobar el grado de seguimiento del alumno. Una actividad puede ser asignada a un estudiante tantas veces como se quiera, siendo cada vez uan asignación distinta.
- **Recursos Multimedia**: Los recursos educativos son materiales, herramientas o medios utilizados en el proceso de enseñanza y aprendizaje para facilitar la adquisición de conocimientos y/o habilidades por parte de los estudiantes. En el caso de AcogeCYL, estos recursos pueden incluir referencias a libros de texto, artículos, videos, simulaciones, etc. y los propios recursos multimedia que se pueden añadir a una actividad: imágenes, vídeos y/o audios.

4.3. Requisitos

El Glosario de Estándares de la IEEE de Terminología de Ingeniería del Software [70] define un requisito como:

1. *Una condición o capacidad necesitada por un usuario para resolver un problema o conseguir un objetivo.*
2. *Una condición o capacidad que debe tener o cumplir o el sistema o un componente del sistema para debe satisfacer un contrato, estándar, especificación u otro documento impuesto formalmente.*
3. *Una representación documentada de una condición o capacidad como en 1. o 2.*

Como en este proyecto se trabaja con *scrum*, se desglosan las épicas del *Product Backlog* en historias de usuario, más manejables que las épicas y que nos indicarán, a través de las necesidades de los *stakeholders*, los requisitos funcionales de la aplicación.

Para obtener los requisitos del proyecto, se seguirá la guía FURPS+, introducida por Hewlett-Packard (HP) a finales de la década de 1980 [42]. Esta guía clasifica los requisitos en cinco categorías: Funcionalidad, Usabilidad, Confiabilidad (Reliability), Rendimiento (Performance) y Soportabilidad (Supportability), garantizando así una especificación de requisitos completa y de alta calidad. La categoría de Funcionalidad se cubre en el siguiente apartado.

4.3.1. Requisitos funcionales

Se han dividido cada una de las 9 épicas EP01-EP09 especificadas en la Sección 2.9 en las historias de usuario que abarque cada una. Como se ha mencionado en el Capítulo 2, el *Product Backlog* es una lista de prioridades, estas prioridades se representan puntuando

cada una de las historias de usuario con los llamados *story points*, una unidad de medida usada para estimar el esfuerzo requerido para implementar la historia de usuario. Los story points no son una medida fija, sino que dependen de varios factores como la experiencia del desarrollador, el esfuerzo que le supondría, la complejidad, la incertidumbre, el riesgo etc. La escala que se ha aplicado para la medición de cada una de las historias de usuario es la escala de Fibonacci modificada: 0.5, 1, 2, 3, 5, 8, 13, 20, ...

Las Tablas 4.1-4.9 describen cada una de las historias de usuario de las épicas establecidas:

ID	Historia de usuario	Puntos
US01	Como miembro de un centro educativo quiero poder crearme una cuenta para acceder a los recursos y herramientas disponibles en la plataforma.	8
US02	Como miembro de un centro educativo, quiero poder iniciar sesión para acceder de manera segura a los recursos y servicios ofrecidos por la plataforma.	5

Tabla 4.1: Historias de usuario de la épica **EP01**.

ID	Historia de usuario	Puntos
US03	Como docente quiero ser capaz de crear actividades para proporcionar a mis estudiantes recursos para su integración estructurados y adecuados a sus necesidades.	13
US04	Como docente quiero poder ver todas las actividades que he creado para gestionar y organizar mejor mis materiales de enseñanza.	5
US05	Como docente quiero poder ver los detalles de mis actividades creadas para consultar su contenido.	5
US06	Como docente quiero ser capaz de editar las actividades creadas por mí mismo para poder arreglar errores o aportar nueva información o preguntas.	13
US07	Como docente quiero ser capaz de eliminar actividades para mantener el contenido del curso actualizado y relevante para los estudiantes.	2
US08	Como docente quiero ser capaz de añadir preguntas a las actividades que he creado para evaluar a los estudiantes.	5

Tabla 4.2: Historias de usuario de la épica **EP02**.

ID	Historia de usuario	Puntos
US09	Como docente quiero ser capaz de asignar mis actividades a cualquiera de mis alumnos para proporcionarles material de trabajo y personalizar su proceso de integración.	8
US10	Como docente quiero poder ver el estado de todas las asignaciones de una actividad para hacer el seguimiento del trabajo de mis estudiantes.	3
US11	Como docente quiero poder ver las respuestas de mis estudiantes a la asignación de una actividad que hayan acabado para poder aconsejarlos y corregirlos.	5

Tabla 4.3: Historias de usuario de la épica **EP03**.

ID	Historia de usuario	Puntos
US12	Como docente quiero ser capaz de crear estados de ánimo para proporcionar a mis estudiantes emociones con las que se sientan identificados y puedan expresarse.	8
US13	Como docente quiero poder ver todos los estados de ánimo registrados en el sistema para estar al tanto de cuáles hay y saber si es necesario añadir alguno.	3
US14	Como docente quiero poder ver los detalles de cada estado de ánimo registrado en el para saber si la imagen o descripción son correctos.	5
US15	Como docente quiero ser capaz de eliminar estados de ánimo para evitar erratas o estados de ánimo incoherentes o en desuso.	2

Tabla 4.4: Historias de usuario de la épica **EP04**.

ID	Historia de usuario	Puntos
US16	Como docente quiero ser capaz de listar todos los estudiantes a los que imparto clase para tener una visión general de mi grupo de estudiantes.	2
US17	Como docente quiero ser capaz de ver el estado de ánimo actual de cada uno los estudiantes a los que imparto clase para comprender como se siente en un determinado momento.	2
US18	Como docente quiero poder ver el diario emocional de todos los estudiantes a los que imparto clase para poder hacer un seguimiento individualizado de su bienestar emocional, identificar posibles desafíos o preocupaciones, y ofrecer el apoyo necesario	3

Tabla 4.5: Historias de usuario de la épica **EP05**.

ID	Historia de usuario	Puntos
US19	Como estudiante quiero poder ver todas las actividades que me han asignado y que están pendientes para tener mejorar mi proceso de integración.	2
US20	Como estudiante quiero poder ver los detalles de todas las actividades que me han asignado y que están pendientes para comprender claramente el contenido de cada tarea, así como para acceder a la información necesaria para completarlas de manera efectiva.	3
US21	Como estudiante quiero poder resolver las actividades que me han sido asignadas y tengo pendientes para completarlas satisfactoriamente, demostrar mi comprensión del contenido y mostrar al docente mi progreso.	13

Tabla 4.6: Historias de usuario de la épica **EP06**.

ID	Historia de usuario	Puntos
US22	Como estudiante quiero poder ver todos los estados de ánimo registrados en el sistema para encontrar uno con el que me identifique.	5
US23	Como estudiante quiero poder ver los detalles de cada uno de los estados de ánimo registrados en el sistema para poder comprender mejor qué emoción refleja cada uno.	3
US24	Como estudiante quiero poder registrar en cualquier momento con qué estado de ánimo me identifico y la razón para poder expresar mis sentimientos.	3

Tabla 4.7: Historias de usuario de la épica **EP07**.

ID	Historia de usuario	Puntos
US25	Como familia quiero poder ver todas las actividades que me han asignado y que están pendientes para tener una visión clara del trabajo restante por hacer.	2
US26	Como familia quiero poder ver los detalles de todas las actividades que me han asignado y que están pendientes para comprender claramente el contenido de cada tarea, así como para acceder a la información necesaria para completarlas de manera efectiva.	1
US27	Como familia quiero poder resolver las actividades que me han sido asignadas y tengo pendientes para completarlas satisfactoriamente, demostrar mi comprensión del contenido y mostrar al docente mi progreso	1

Tabla 4.8: Historias de usuario de la épica **EP08**.

ID	Historia de usuario	Puntos
US28	Como administrador quiero tener acceso a todas las actividades registradas en el sistema para conocer qué actividades han creado todos los docentes de todos los centros.	1
US29	Como administrador quiero poder publicar cualquier actividad para que esta sea asignada a todos los estudiantes registrados en el sistema y puedan tener acceso a ella.	2

Tabla 4.9: Historias de usuario de la épica **EP09**.

4.3.2. Requisitos no funcionales

En el apartado anterior se han obtenido los requisitos funcionales, los cuales describen las funciones y comportamientos específicos que el sistema debe realizar. Ahora, este apartado de requisitos no funcionales, se enfoca capturar el resto de los elementos del marco FURPS (Funcionalidad, Usabilidad, Confiabilidad, Rendimiento y Soportabilidad), abordando aspectos como la facilidad de uso, la fiabilidad, el rendimiento y la mantenibilidad del sistema.

Al igual que con los requisitos funcionales, se representan en la Tabla 4.10 mediante historias de usuario.

ID	Historia de usuario
NFUS01	Como usuario de la aplicación, quiero que el sistema sea intuitivo, fácil de usar, con un diseño atractivo y que la navegación sea fluida para que pueda realizar mis tareas rápida y satisfactoriamente.
NFUS02	Como usuario de la aplicación, quiero que el sistema esté disponible en todo momento para que pueda acceder a él cuando lo necesite.
NFUS03	Como usuario de la aplicación, quiero que la página de inicio se cargue en menos de 1 segundo para que pueda acceder rápidamente.
NFUS04	Como usuario de la aplicación, quiero que sea compatible con los principales navegadores web para que pueda acceder a ella desde cualquier dispositivo.
NFUS05	Como usuario de la aplicación, quiero que implemente una autenticación segura para proteger mis datos contra accesos no autorizados.
NFUS06	Como usuario de la aplicación, quiero que pueda manejar un aumento del 100 % en el tráfico de usuarios para que pueda seguir usándola sin problemas durante los períodos de alta demanda.
NFUS07	Como desarrollador, quiero que el código fuente siga las mejores prácticas de programación y esté bien documentado para facilitar futuras actualizaciones y mantenimiento del sistema.
NFUS08	Como usuario de la aplicación, quiero que sea compatible con una variedad de plataformas para poder acceder a ella desde cualquier dispositivo que prefiera.
NFUS09	Como usuario de la aplicación, quiero que la aplicación web soporte los traductores integrados en los navegadores para poder traducir el contenido al idioma que prefiera.
NFUS10	Como administrador, quiero tener la capacidad de realizar copias de seguridad automáticas de la base de datos para garantizar la integridad de los datos en caso de fallo del sistema.
NFUS11	Como tutor del Trabajo de Fin de Grado, quiero que la aplicación web se programe utilizando Python y el *framework* Flask para mantener la consistencia y familiaridad con el entorno de desarrollo.
NFUS12	Como desarrollador, quiero que toda la documentación de la aplicación esté disponible en español ya que el uso de la aplicación está limitado a la región de Castilla y León.
NFUS13	Como desarrollador, quiero que la aplicación permita la depuración durante el desarrollo para poder identificar y solucionar errores de manera eficiente.
NFUS14	Como desarrollador, quiero que no haya acoplamiento entre las distintas capas para asegurar la privacidad de la información de los usuarios finales.
NFUS15	Como desarrollador, quiero que la aplicación sea una App Web para poder usar la traducción de página y llegar a alumnos cuya lengua materna no sea el español.

Tabla 4.10: Requisitos no funcionales mediantes historias de usuario.

4.3.3. Requisitos de información

la Tabla 4.11 muestra todos los requisitos de información de la aplicación:

ID	Historia de usuario
INF01	Como equipo de desarrollo quiero guardar los siguientes datos de un **centro**: identificador, nombre, dirección y tipo.
INF02	Como equipo de desarrollo quiero guardar los siguientes datos de un **curso**: identificador, año, letra, etapa, identificador del centro e identificador del profesor responsable de la clase.
INF03	Como equipo de desarrollo quiero guardar los siguientes datos de un **docente**: identificador, clave, nombre, apellidos, email, cuerpo, especialidad, cargo e identificador del centro en el que trabaja.
INF04	Como equipo de desarrollo quiero guardar los siguientes datos de un **estudiante**: identificador, clave, nombre, nacionalidad e identificador del curso en el que estudia.
INF05	Como equipo de desarrollo quiero guardar los siguientes datos de una **actividad**: identificador, nombre, descripción, identificador del docente que la creó, fecha de creación, imagen, vídeo y audio.
INF06	Como equipo de desarrollo quiero guardar los siguientes datos de una **asignación**: identificador, identificador de la actividad asignada, identificador del estudiante al que se le asigna, fecha de asignación, si está finalizada y si la asignación es para la familia del estudiante.
INF07	Como equipo de desarrollo quiero guardar los siguientes datos de una **pregunta**: identificador, texto de la pregunta e identificador de la actividad asociada a la pregunta.
INF08	Como equipo de desarrollo quiero guardar los siguientes datos de una **respuesta**: identificador, texto de la respuesta, identificador de la pregunta a la que responde la respuesta e identificador de la asignación que resuelve.
INF09	Como equipo de desarrollo quiero guardar los siguientes datos de un **estado de ánimo**: identificador, identificador del docente que lo creó, nombre del estado, descripción e imagen.
INF10	Como equipo de desarrollo quiero guardar los siguientes datos de un **estado de ánimo de estudiante**: identificador, identificador del estado de ánimo, identificador del estudiante, texto en el que explica por qué se siente así y la fecha en la que se registra.

Tabla 4.11: Requisitos de información mediante historias de usuario.

4.3.4. Restricciones

A mayores de los requisitos elicitados, pueden existar restricciones. Estas son son limitaciones y condiciones que deben ser cumplidas durante el desarrollo y funcionamiento de la

aplicación.

ID	Historia de usuario
RES01	Como desarrollador, quiero que la aplicación cumpla con el RGPD (Reglamento General de Protección de Datos) para asegurar la confidencialidad de la información de los usuarios.
RES02	Como desarrollador, quiero que la aplicación sea compatible con múltiples plataformas (Windows, MacOS, Linux) para asegurar su accesibilidad.
RES03	Como usuario, quiero que la aplicación sea gratuita.

Tabla 4.12: Restricciones expresadas como historias de usuario.

4.4. Modelo de Casos de Uso

4.4.1. Identificación de los Casos de Uso

Según Gomaa [67], un caso de uso describe una secuencia de interaciones, incluyendo variantes, entre uno o más actores y el sistema. Los casos de uso son útiles para capturar los requisitos funcionales de un sistema y para entender cómo diferentes tipos de usuarios (actores) utilizan el sistema para cumplir con sus tareas. Estos casos ayudan en el diseño y la validación del comportamiento del sistema desde la perspectiva del usuario. Se han definido los siguientes casos de uso que se muestran en la Tabla 4.13:

ID	Nombre	Descripción
UC01	Registro de docente	El sistema deberá comportarse tal como se describe en la especificación UC01. Registro de docente.
UC02	Registro de estudiante	El sistema deberá comportarse tal como se describe en la especificación UC02. Registro de estudiante.
UC03	Iniciar sesión	El sistema deberá comportarse tal como se describe en la especificación UC03. Iniciar sesión.
UC04	Ver lista de actividades	El sistema deberá comportarse tal como se describe en la especificación UC04. Ver lista de actividades.
UC05	Crear una actividad	El sistema deberá comportarse tal como se describe en la especificación UC05. Crear una actividad.

Sigue en la siguiente página

Tabla 4.13 – continuación de la página anterior

ID	Nombre	Descripción
UC06	Ver detalles de una actividad	El sistema deberá comportarse tal como se describe en la especificación UC06. Ver detalles de una actividad.
UC07	Editar una actividad	El sistema deberá comportarse tal como se describe en la especificación UC07. Editar una actividad.
UC08	Asignar una actividad	El sistema deberá comportarse tal como se describe en la especificación UC08. Asignar una actividad.
UC09	Eliminar una actividad	El sistema deberá comportarse tal como se describe en la especificación UC09. Eliminar una actividad.
UC10	Ver lista de asignaciones	El sistema deberá comportarse tal como se describe en la especificación UC10. Ver lista de asignaciones.
UC11	Ver respuesta de una asignación	El sistema deberá comportarse tal como se describe en la especificación UC11. Ver respuesta de una asignación.
UC12	Crear un estado de ánimo	El sistema deberá comportarse tal como se describe en la especificación UC12. Crear un estado de ánimo.
UC13	Ver estados de ánimo	El sistema deberá comportarse tal como se describe en la especificación UC13. Ver estados de ánimo.
UC14	Ver detalles de un estado de ánimo	El sistema deberá comportarse tal como se describe en la especificación UC14. Ver detalles de un estado de ánimo.
UC15	Eliminar un estado de ánimo	El sistema deberá comportarse tal como se describe en la especificación UC15. Eliminar un estado de ánimo.
UC16	Ver estado de ánimo actual de estudiantes	El sistema deberá comportarse tal como se describe en la especificación UC16. Ver estado de ánimo actual de estudiantes.
UC17	Ver diario emocional de un estudiante	El sistema deberá comportarse tal como se describe en la especificación UC17. Ver diario emocional de un estudiante.
UC18	Registrar un estado de ánimo	El sistema deberá comportarse tal como se describe en la especificación UC18. Registrar un estado de ánimo.

Sigue en la siguiente página

Tabla 4.13 – continuación de la página anterior

ID	Nombre	Descripción
UC19	Resolver una actividad	El sistema deberá comportarse tal como se describe en la especificación UC19. Resolver una actividad.
UC20	Publicar una actividad	El sistema deberá comportarse tal como se describe en la especificación UC20. Publicar una actividad.

Tabla 4.14: Casos de uso de la aplicación.

4.4.2. Actores del sistema

En un sistema informático, los actores se refieren a las entidades que interactúan con el sistema de alguna manera. Estos actores pueden ser personas, organizaciones, dispositivos o incluso otros sistemas que tienen un rol específico en la interacción con el sistema informático [51]. Cada actor tiene un conjunto de objetivos y tareas específicas que realizan con el sistema. Entender estos roles ayuda a diseñar un sistema que satisfaga las necesidades de todos los *stakeholders*.

En la Tabla 4.15, se muestran cinco actores distintos que interactúan con el sistema. Estos son esenciales para entender y diseñar la funcionalidad del sistema:

ID	Nombre	Descripción
AC01	Usuario	Es el actor genérico del cual se especializan todos los demás actores. Representa a cualquier persona que interactúa con el sistema. Agrupa aquellos casos de uso que todos los usuarios del sistema pueden emplear.
AC02	Docente	Se encarga de crear actividades y estados de ánimo para los estudiantes. Puede revisar las respuestas de los estudiantes a las actividades y sus emociones.
AC03	Estudiante	Accede al sistema para emplear materiales educativos y registrar su estado emocional.
AC04	Familia	Usuario con menos privilegios. Está destinado a realizar actividades familiares que puedan proponer los docentes o el administrador.
AC05	Administrador	Tiene acceso a todas las actividades y puede publicar cualquiera de ellas, es decir, asignarla a todos los estudiantes.

Tabla 4.15: Actores del sistema.

4.4.3. Diagrama de casos de uso

Con el fin de mejorar la visibilidad del diagrama de los casos de uso, se ha optado por separarlo en varios diagramas distintos en función de los actores o la funcionalidad.

En la figura 4.1 se muestran los casos de uso que puede ejecutar cualquier usuario:

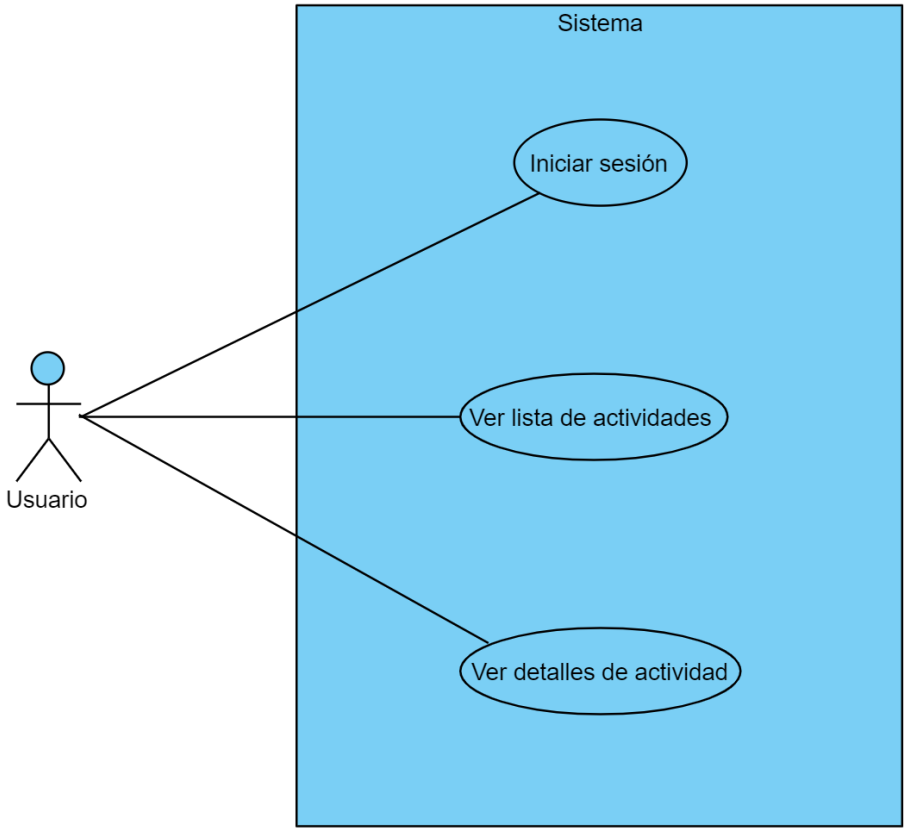

Figura 4.1: Diagrama de Casos de Uso del actor Usuario.

En la Figura 4.2 se muestran los casos de uso del docente relacionados con las actividades mientras que en la Figura 4.3 se muestran los relacionados con los estados de ánimo:

Figura 4.2: Diagrama de Casos de Uso de Actividades del Actor Docente.

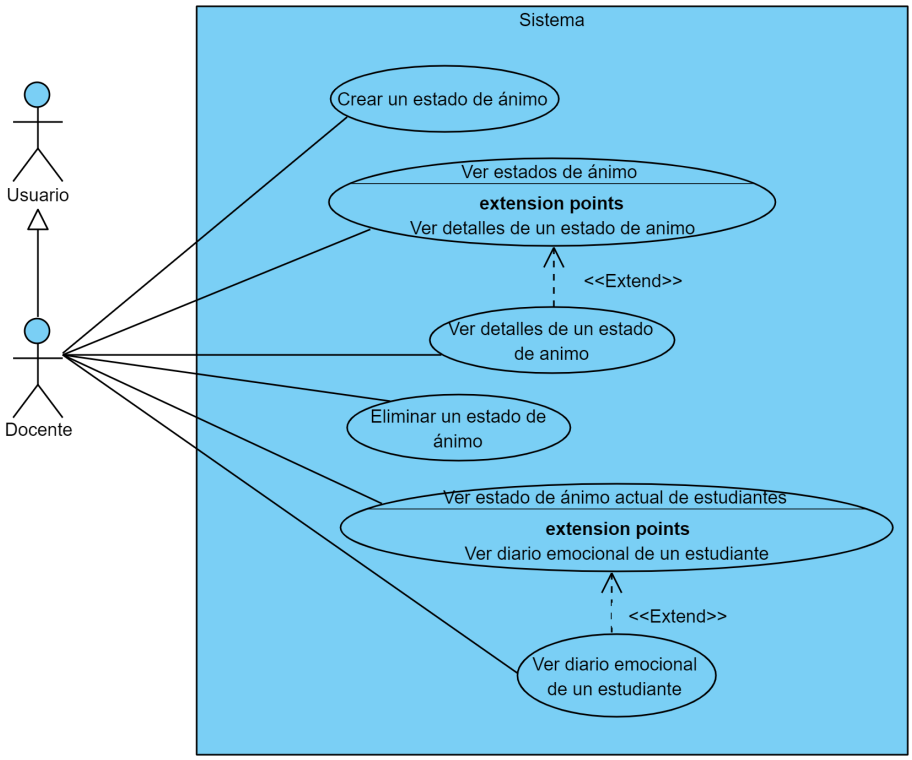

Figura 4.3: Diagrama de Casos de Uso de Estados de Ánimo del Actor Docente.

Los casos de uso del resto de actores están agrupados en el mismo diagrama de casos de uso, se puede ver en la Figura 4.4:

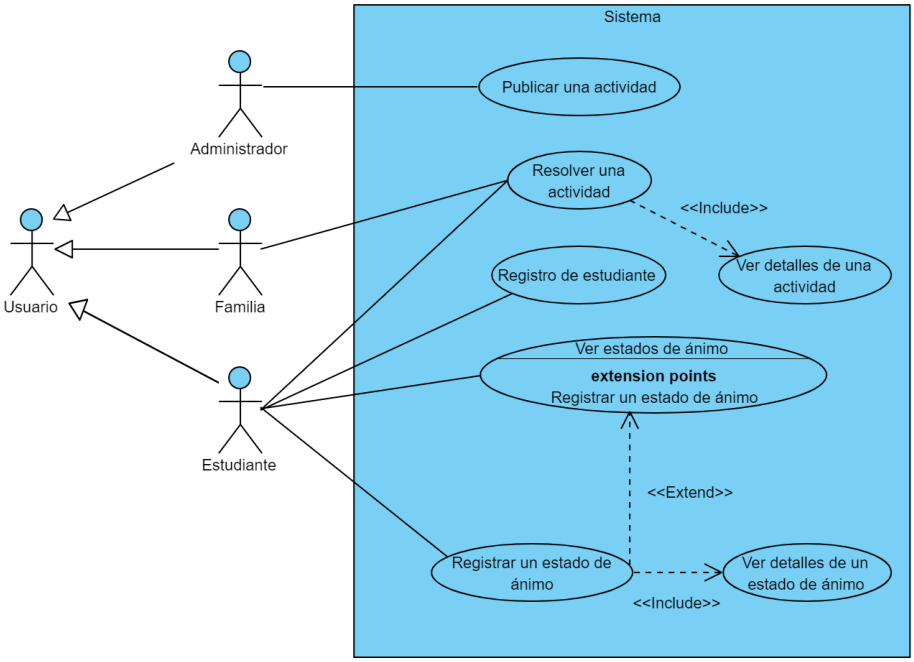

Figura 4.4: Diagrama de Casos de Uso de los actores Estudiante, Familia y Administrador.

4.4.4. Especificación de los Casos de Uso

En esta sección se describirán los casos de uso identificados en la Tabla 4.13.

4.4.4.1. UC01. Registro de docente.

La Tabla 4.16 muestra la especificación del caso de uso UC01.

Caso de uso:	Registro de docente
Descripción:	El caso de uso permitirá el registro de un nuevo docente en el sistema.
Actores:	Docente.
Pre-condiciones:	El docente no debe estar registrado.
Post-condiciones:	El sistema reconoce al nuevo docente permitiéndole realizar los casos de uso que figure como actor.

Secuencia normal

1. El caso de uso comienza cuando el docente indica que desea registrarse en el sistema.

2. El sistema solicita al actor indicar si desea registrarse como estudiante o como docente.

3. El docente selecciona que quiere crearse una cuenta como docente.

4. El sistema muestra y solicita la información necesaria para añadir los datos del nuevo docente.

5. El docente rellena los datos solicitados por el sistema y solicita darse de alta como docente.

6. El sistema comprueba que los datos son válidos.

7. El sistema registra el nuevo docente y el caso de uso finaliza.

Excepciones

3.a), **5.a)** El usuario solicita finalizar el caso de uso, el caso de uso finaliza sin efectos.

6.a) Si los datos no son correctos el sistema muestra un aviso imposibilitando el registro del nuevo docente.

Tabla 4.16: Descripción del caso de uso UC01. Registro de docente.

4.4.4.2. UC02. Registro de estudiante.

La Tabla 4.17 muestra la especificación del caso de uso UC02.

Caso de uso:	Registro de estudiante
Descripción:	El caso de uso permitirá el registro de un nuevo estudiante en el sistema.
Actores:	Estudiante.
Pre-condiciones:	El estudiante no debe estar registrado.
Post-condiciones:	El sistema reconoce al nuevo estudiante permitiéndole realizar los casos de uso que figure como actor.

Secuencia normal

1. El caso de uso comienza cuando el estudiante indica que desea registrarse en el sistema.

2. El sistema solicita al actor indicar si desea registrarse como estudiante o como docente.

3. El estudiante selecciona que quiere crearse una cuenta como estudiante.

4. El sistema muestra y solicita la información necesaria para añadir los datos del nuevo estudiante.

5. El estudiante rellena los datos solicitados por el sistema y solicita darse de alta como estudiante.

6. El sistema comprueba que los datos son válidos.

7. El sistema registra el nuevo estudiante y el caso de uso finaliza.

Excepciones

3.a), **5.a)** El usuario solicita finalizar el caso de uso, el caso de uso finaliza sin efectos.

6.a) Si los datos no son correctos el sistema muestra un aviso imposibilitando el registro del nuevo estudiante y vuelve al paso 4.

Tabla 4.17: Descripción del caso de uso UC02. Registro de estudiante.

4.4.4.3. UC03. Iniciar sesión.

La Tabla 4.18 muestra la especificación del caso de uso UC03.

Caso de uso:	Iniciar sesión
Descripción:	El caso de uso permitirá iniciar sesión al usuario en el sistema.
Actores:	Usuario.
Pre-condiciones:	El usuario no debe tener la sesión iniciada y debe estar registrado en el sistema.
Post-condiciones:	El sistema da acceso a los casos de uso disponibles para el tipo de usuario con sesión iniciada.
Secuencia normal	
1. El caso de uso comienza cuando un actor indica que quiere iniciar sesión en el sistema. **2.** El sistema muestra y solicita la información necesaria para iniciar sesión. **3.** El actor rellena los datos solicitados por el sistema y solicita iniciar sesión. **4.** El sistema comprueba los datos del actor, da acceso al actor al sistema y el caso de uso finaliza.	
Excepciones	
3.a) El usuario solicita finalizar el caso de uso, el caso de uso finaliza sin efectos. **4.a)** Si los datos no son correctos el sistema muestra un aviso al actor imposibilitando el inicio de sesión, el caso de uso vuelve al paso 2.	

Tabla 4.18: Descripción del caso de uso UC03. Iniciar sesión.

4.4.4.4. UC04. Ver lista de actividades.

La Tabla 4.19 muestra la especificación del caso de uso UC04.

Caso de uso:	Ver lista de actividades
Descripción:	El caso de uso permitirá ver la lista de actividades de un usuario registrado en el sistema.
Actores:	Usuario
Pre-condiciones:	El actor debe tener sesión iniciada en el sistema.
Post-condiciones:	El sistema muestra todas las actividades accesibles por el actor.
Secuencia normal	
1. El caso de uso comienza cuando un actor indica que quiere ver su lista de actividades. **2.** Si el actor es un docente, el sistema muestra todas las actividades creadas por el docente.	
Excepciones	
2.a) Si el actor es un estudiante o una familia, el sistema muestra todas las actividades asignadas que el estudiante/familia todavía no ha respondido. **2.b)** Si el actor es un administrador, el sistema muestra todas las actividades registradas en el sistema.	

Tabla 4.19: Descripción del caso de uso UC04. Ver lista de actividades.

4.4.4.5. UC05. Crear una actividad.

La Tabla 4.20 muestra la especificación del caso de uso UC05.

Caso de uso:	Crear una actividad
Descripción:	El caso de uso permitirá al docente crear una nueva actividad y registrarla en el sistema.
Actores:	Docente.
Pre-condiciones:	El docente debe tener sesión iniciada en el sistema.
Post-condiciones:	El sistema registra la nueva actividad y el docente tiene acceso a ella.

Secuencia normal

1. El caso de uso comienza cuando el docente indica que quiere crear una actividad.

2. El sistema muestra y solicita la información necesaria para crear una nueva actividad.

3. El docente rellena los datos solicitados mientras el sistema los valida y confirma.

4. El sistema solicita al docente indicar si desea añadir preguntas a la actividad.

5. El docente indica que quiere añadir preguntas.

6. El sistema solicita que el actor introduzca las preguntas.

7. El docente introduce las preguntas y acepta.

8. El sistema avisa de la acción y solicita confirmación.

9. El docente confirma.

10. El sistema registra la nueva actividad y sus preguntas en el sistema y el caso de uso termina.

Excepciones

3.a), 5a) El docente solicita finalizar el caso de uso, el caso de uso finaliza sin efectos.

5.b) El docente indica que no quiere añadir preguntas. Se registra la nueva actividad en el sistema y el caso de uso termina.

9.a) El docente no confirma, se vuelve al paso 7.

Tabla 4.20: Descripción del caso de uso UC05. Crear una actividad.

4.4.4.6. UC06. Ver detalles de una actividad.

La Tabla 4.21 muestra la especificación del caso de uso UC06.

Caso de uso:	Ver detalles de una actividad
Descripción:	El caso de uso permitirá ver los detalles de una actividad registrada en el sistema.
Actores:	Usuario.
Pre-condiciones:	El actor debe tener sesión iniciada en el sistema y encontrarse en la lista de Actividades.
Post-condiciones:	El sistema da acceso a los detalles de la actividad solicitada por el actor.
Secuencia normal	
1. El caso de uso comienza cuando el actor indica que quiere ver los detalles de la actividad. **2.** Si el actor es docente o administrador, el sistema muestra los detalles de la actividad y las preguntas asociadas y el caso de uso finaliza.	

Tabla 4.21: Descripción del caso de uso UC06. Ver detalles de una actividad.

4.4.4.7. UC07. Editar una actividad.

La Tabla 4.22 muestra la especificación del caso de uso UC07.

Caso de uso:	Editar una actividad
Descripción:	El caso de uso permitirá modificar los datos de una actividad registrada en el sistema.
Actores:	Docente.
Pre-condiciones:	El docente debe tener sesión iniciada en el sistema, tener al menos una actividad registrada en el sistema y encontrarse en la lista de Actividades.
Post-condiciones:	El sistema crea una nueva actividad con los datos modificados de la actividad a editar y el docente tiene acceso a ella.

Secuencia normal

1. El caso de uso comienza cuando el actor solicita modificar los detalles de la actividad.

2. El sistema muestra los detalles de la actividad y permite modificarlos.

3. El docente modifica los detalles deseados mientras el sistema los valida y acepta.

4. El sistema solicita al docente indicar si desea añadir preguntas a la actividad.

5. El docente indica que añadir las preguntas.

6. El sistema muestra las preguntas ya registradas y permite modificarlas o quitarlas y añadir nuevas preguntas.

7. El docente modifica, añade y/o quita preguntas y acepta.

8. El sistema avisa de la acción y solicita confirmación.

9. El docente confirma.

10. El sistema registra la nueva actividad y sus preguntas en el sistema y el caso de uso termina.

Excepciones

3.a) Si los datos no son correctos el sistema muestra un aviso al actor imposibilitando el guardado, el caso de uso sigue en el paso 3.

3.b, 5.a) El docente solicita finalizar el caso de uso, el caso de uso finaliza sin efectos.

5b) Si el docente indica que no quiere añadir preguntas, se registra la nueva actividad editada y el caso de uso termina.

Tabla 4.22: Descripción del caso de uso UC07. Editar una actividad.

4.4.4.8. UC08. Asignar una actividad.

La Tabla 4.23 muestra la especificación del caso de uso UC08.

Caso de uso:	Asignar una actividad
Descripción:	El caso de uso permitirá al docente asignar una actividad a uno o más estudiantes.
Actores:	Docente.
Pre-condiciones:	El actor debe tener sesión iniciada en el sistema, tener una o más actividades registradas, dar clase al menos a un estudiante y encontrarse en la lista de Actividades.
Post-condiciones:	El sistema asigna la tarea a los estudiantes seleccionados y estos podrán acceder a ella.
Secuencia normal	
1. El caso de uso comienza cuando un docente indica que quiere asignar la actividad a estudiantes. **2.** El sistema muestra todos los estudiantes a los que el docente imparte clases y solicita una selección. **3.** El actor selecciona aquellos estudiantes a los que quiera asignar la actividad y acepta. **4.** El sistema avisa de la acción y solicita confirmación. **5.** El docente confirma. **6.** El sistema crea las asignaciones de la actividad a todos los estudiantes seleccionados y las registra. El caso de uso termina.	
Excepciones	
3.a) El usuario solicita finalizar el caso de uso, el caso de uso finaliza sin efectos. **5.a)** El docente no confirma y se vuelve al paso 3.	

Tabla 4.23: Descripción del caso de uso UC08. Asignar una actividad.

4.4.4.9. UC09. Eliminar una actividad.

La Tabla 4.24 muestra la especificación del caso de uso UC09.

Caso de uso:	Eliminar una actividad
Descripción:	El caso de uso permitirá eliminar los datos de una actividad registrada en el sistema.
Actores:	Docente.
Include:	UC06. Ver detalles de una actividad
Pre-condiciones:	El docente debe tener sesión iniciada en el sistema, al menos una actividad registrada y encontrarse en la lista de Actividades.
Post-condiciones:	El sistema da acceso a los datos de la actividad y permite eliminación.
Secuencia normal	
1. El caso de uso comienza cuando el actor solicita borrar los datos de la actividad. **2.** El sistema ejecuta el UC06. Ver detalles de una actividad y muestra los detalles de la actividad y solicita confirmación para eliminar la actividad. **3.** El actor confirma que quiere eliminar los datos de la actividad. **4.** El sistema elimina los datos de la actividad y el caso de uso finaliza.	
Excepciones	
3.a) El docente solicita finalizar el caso de uso, el caso de uso finaliza sin efectos.	

Tabla 4.24: Descripción del caso de uso UC09. Eliminar una actividad.

4.4.4.10. UC10. Ver lista de asignaciones.

La Tabla 4.25 muestra la especificación del caso de uso UC10.

Caso de uso:	Ver lista de asignaciones
Descripción:	El caso de uso permitirá al docente ver las respuestas de todas las asignaciones de actividades respondidas o pendientes de sus estudiantes.
Actores:	Docente.
Pre-condiciones:	El Docente debe tener sesión iniciada en el sistema, impartir clases a un alumno por lo menos y encontrarse en la lista de Actividades.
Post-condiciones:	El sistema da acceso a los datos de todas las asignaciones de los estudiantes a los que imparte clase el docente.
Secuencia normal	
1. El caso de uso comienza cuando el docente indica que quiere ver los datos de todas las asignaciones de la actividad. **2.** El sistema muestra todas las asignaciones de la actividad respondidas o pendientes de los estudiantes del docente. **3.** Si el docente quiere ver la respuesta de una asignación, punto de extensión: UC11. Ver respuesta de una asignación.	

Tabla 4.25: Descripción del caso de uso UC10. Ver lista de asignaciones.

4.4.4.11. UC11. Ver respuesta de una asignación.

La Tabla 4.26 muestra la especificación del caso de uso UC11.

Caso de uso:	Ver respuesta de una asignación
Descripción:	El caso de uso permitirá a un docente ver las respuestas de un estudiante suyo a una asignación de actividad.
Actores:	Docente.
Extends:	UC10. Ver lista de asignaciones
Pre-condiciones:	El docente debe tener sesión iniciada en el sistema, impartir clases a algún estudiante y encontrarse en la lista de respuestas después de haber terminado el caso de uso UC10. Ver lista de asignaciones.
Post-condiciones:	El sistema da acceso a los detalles de la actividad y a las respuestas del alumno a las preguntas asociadas a la actividad asignada.
Secuencia normal	
1. El caso de uso comienza cuando el docente solicita ver la respuesta del estudiante a la actividad.	
2. El sistema muestra los detalles de la actividad, las preguntas asociadas a la actividad, y las respuestas aportadas por el estudiante a las preguntas.	

Tabla 4.26: Descripción del caso de uso UC11. Ver respuesta de una asignación.

4.4.4.12. UC12. Crear un estado de ánimo.

La Tabla 4.27 muestra la especificación del caso de uso UC12.

Caso de uso:	Crear un estado de ánimo
Descripción:	El caso de uso permitirá al docente crear un nuevo estado ánimo y registrarlo en el sistema.
Actores:	Docente.
Pre-condiciones:	El docente debe tener sesión iniciada en el sistema.
Post-condiciones:	El sistema registra el nuevo estado de ánimo y el docente y los estudiantes tienen acceso a ella.
Secuencia normal	
1. El caso de uso comienza cuando el docente indica que quiere crear un estado de ánimo.	
2. El sistema muestra y solicita la información necesaria para crear un nuevo estado de ánimo.	
3. El docente rellena los datos solicitados mientras el sistema los valida y acepta.	
4. El sistema avisa de la acción y solicita confirmación.	
5. El docente confirma.	
6. El sistema registra el nuevo estado de ánimo en el sistema y el caso de uso termina.	
Excepciones	
3.a) El docente solicita finalizar el caso de uso, el caso de uso finaliza sin efectos.	
5.a) Si el docente no confirma, se vuelve al paso 3.	

Tabla 4.27: Descripción del caso de uso UC12. Crear un estado de ánimo.

4.4.4.13. UC13. Ver estados de ánimo.

La Tabla 4.28 muestra la especificación del caso de uso UC13.

Caso de uso:	Ver estados de ánimo
Descripción:	El caso de uso permitirá ver la lista de estados de ánimo registrados en el sistema.
Actores:	Docente y Estudiante.
Pre-condiciones:	El actor debe tener sesión iniciada en el sistema.
Post-condiciones:	El sistema muestra todos los estados de ánimo accesibles por el actor.
Secuencia normal	
1. El caso de uso comienza cuando un actor indica que quiere ver la lista de estados de ánimo.	
2. El sistema muestra todos los estados de ánimo registrados en el sistema.	
3. Si el usuario quiere ver los detalles de un estado de ánimo, , punto de extensión: UC14. Ver detalles de un estado de ánimo.	

Tabla 4.28: Descripción del caso de uso UC13. Ver estados de ánimo.

4.4.4.14. UC14. Ver detalles de un estado de ánimo.

La Tabla 4.29 muestra la especificación del caso de uso UC13.

Caso de uso:	Ver detalles de un estado de ánimo
Descripción:	El caso de uso permitirá ver los detalles de un estado de ánimo registrado en el sistema.
Actores:	Docentes y Estudiantes.
Extends:	UC13. Ver estados de ánimo
Pre-condiciones:	El actor debe tener sesión iniciada en el sistema, debe existir al menos un estado de ánimo y el actor debe encontrarse en la pantalla de los estados de ánimo despues de haber completado el caso de uso UC13. Ver estados de ánimo.
Post-condiciones:	El sistema da acceso a los detalles del estado de ánimo solicitado por el actor.
Secuencia normal	
1. El caso de uso comienza cuando el actor indica que quiere ver los detalles del estado de ánimo.	
2. El sistema muestra los detalles del estado de ánimo y el caso de uso finaliza.	

Tabla 4.29: Descripción del caso de uso UC14. Ver detalles de un estado de ánimo.

4.4.4.15. UC15. Eliminar un estado de ánimo.

La Tabla 4.30 muestra la especificación del caso de uso UC15.

Caso de uso:	Eliminar un estado de ánimo
Descripción:	El caso de uso permitirá eliminar un estado de ánimo registrado en el sistema.
Actores:	Docente.
Pre-condiciones:	El docente debe tener sesión iniciada en el sistema, debe haber al menos un estado de ánimo registrado y el docente debe encontrarse en la pantalla de los estados de ánimo despues de haber completado el caso de uso UC13. Ver estados de ánimo.
Post-condiciones:	El sistema permite la eliminación del estado de ánimo.
Secuencia normal	
1. El caso de uso comienza cuando el actor solicita borrar el estado de ánimo. **2.** El sistema avisa de la acción y solicita confirmación. **3.** El actor confirma que quiere eliminar el estado de ánimo. **4.** El sistema elimina los datos del estado de ánimo y el caso de uso finaliza.	
Excepciones	
3.a) El docente solicita finalizar el caso de uso, el caso de uso finaliza sin efectos.	

Tabla 4.30: Descripción del caso de uso UC15. Eliminar un estado de ánimo.

4.4.4.16. UC16. Ver estado de ánimo actual de estudiantes.

La Tabla 4.31 muestra la especificación del caso de uso UC16.

Caso de uso:	Ver estado de ánimo actual de estudiantes
Descripción:	El caso de uso permitirá al docente ver todos los estudiantes a los que imparte clase y su estado de ánimo actual.
Actores:	Docente.
Pre-condiciones:	El docente debe tener sesión iniciada en el sistema.
Post-condiciones:	El sistema da acceso a los estados de ánimo de todos los estudiantes a los que imparte clase el docente.
Secuencia normal	

1. El caso de uso comienza cuando el docente indica que quiere ver los estados de ánimo actuales de sus estudiantes.

2. El sistema muestra el estado de ánimo de todos de los estudiantes del docente y la última fecha de actualización.

3. Si el docente quiere ver el diario emocional de un estudiante, punto de extensión: UC17. Ver diario emocional de un estudiante.

Tabla 4.31: Descripción del caso de uso UC16. Ver estado de ánimo actual de estudiantes.

4.4.4.17. UC17. Ver diario emocional de un estudiante.

La Tabla 4.32 muestra la especificación del caso de uso UC17.

Caso de uso:	Ver diario emocional de un estudiante
Descripción:	El caso de uso permitirá a un docente ver el diario emocional de un estudiante al que imparte clases.
Actores:	Docente.
Extends:	UC16. Ver estado de ánimo actual de estudiantes
Pre-condiciones:	El docente debe tener sesión iniciada en el sistema, impartir clases a, al menos, un estudiante y debe encontrarse en la pantalla de los estados actuales de sus estudiantes después de haber terminado el caso de uso UC16. Ver estado de ánimo actual de estudiantes.
Post-condiciones:	El sistema da acceso al docente al historial de los estados de ánimo del estudiante.
Secuencia normal	
1. El caso de uso comienza cuando el docente solicita ver el diario emocional del estudiante.	
2. El sistema muestra el historial de todos los estados de ánimo del estudiante incluyendo las fechas de cada uno.	

Tabla 4.32: Descripción del caso de uso UC17. Ver diario emocional de un estudiante.

4.4.4.18. UC18. Registrar un estado de ánimo.

La Tabla 4.33 muestra la especificación del caso de uso UC18.

Caso de uso:	Registrar un estado de ánimo
Descripción:	El caso de uso permitirá al estudiante registrar en su diario emocional el estado ánimo.
Actores:	Estudiante.
Pre-condiciones:	El estudiante debe tener sesión iniciada en el sistema, debe haber al menos un estado de ánimo registrado en el sistema y el estudiante encontrarse en la pantalla de los estados de ánimo después de haber terminado el caso de uso UC13. Ver estados de ánimo.
Post-condiciones:	El sistema registra el nuevo estado de ánimo y el docente y los estudiantes tienen acceso a ella.
Secuencia normal	
1. El caso de uso comienza cuando el estudiante indica que quiere registrar un estado de ánimo.	
2. El sistema muestra y solicita la información necesaria para crear un nuevo estado de ánimo.	
3. El docente rellena los datos solicitados mientras el sistema los valida y acepta.	
4. El sistema avisa de la acción y solicita confirmación.	
5. El actor confirma que quiere registrar el estado.	
6. El sistema registra el nuevo estado del estudiante en el sistema y el caso de uso termina.	
Excepciones	
3.a) El estudiante solicita finalizar el caso de uso, el caso de uso finaliza sin efectos.	
5.a) Si el estudiante no confirma, se vuelve al paso 3.	

Tabla 4.33: Descripción del caso de uso UC18. Registrar un estado de ánimo.

4.4.4.19. UC19. Resolver una actividad.

La Tabla 4.34 muestra la especificación del caso de uso UC19.

Caso de uso:	Resolver una actividad
Descripción:	El caso de uso permitirá al estudiante resolver una actividad que tenga asignada y no haya respondido todavía.
Actores:	Estudiante.
Include:	UC06. Ver detalles de una actividad
Pre-condiciones:	El estudiante debe tener sesión iniciada en el sistema, tener, al menos, una actividad asignada sin resolver y debe encontrarse en la lista de actividades después de haber completado el caso de uso UC04. Ver lista de actividades.
Post-condiciones:	El sistema registra la asignación como finalizada y da acceso al docente a la asignación resuelta.
Secuencia normal	
1. El caso de uso comienza cuando el estudiante indica que quiere resolver la actividad. **2.** El sistema ejecuta el UC06. Ver detalles de una actividad y muestra los detalles de la actividad y solicita la respuesta a las preguntas. **3.** El estudiante responde a las preguntas y acepta. **4.** El sistema solicita confirmación. **5.** El estudiante confirma. **6.** El sistema registra la asignación como resuelta y guarda las respuestas y el caso de uso termina.	
Excepciones	
2.a) Si la actividad no tiene preguntas asociadas el estudiante acepta y salta al paso 4. **3.a)** El estudiante solicita finalizar el caso de uso, el caso de uso finaliza sin efectos. **5.a)** El estudiante no confirma, se vuelve al paso 3.	

Tabla 4.34: Descripción del caso de uso UC19. Resolver una actividad.

4.4.4.20. UC20. Publicar una actividad.

La Tabla 4.35 muestra la especificación del caso de uso UC20.

Caso de uso:	Publicar una actividad.
Descripción:	El caso de uso permitirá publicar actividades existentes en el sistema.
Actores:	Administrador.
Pre-condiciones:	El administrador debe tener sesión iniciada en el sistema y debe haber, al menos, una actividad registrada en el sistema.
Post-condiciones:	El sistema ha asignado la actividad a todos los estudiantes registrados en el sistema.
Secuencia normal	
1. El caso de uso comienza cuando el administrador indica que quiere publicar la actividad. **2.** El sistema avisa de la acción y solicita confirmación. **3.** El administrador confirma la acción de publicación. **4.** El sistema crea una asignación de la actividad a cada uno de los estudiantes registrados en el sistema y el caso de uso finaliza.	
Excepciones	
3.a) El usuario solicita finalizar el caso de uso, el caso de uso finaliza sin efectos.	

Tabla 4.35: Descripción del caso de uso UC20. Publicar una actividad.

Capítulo 5

Análisis

En este capítulo, se presenta el *workflow* de Análisis del sistema a desarrollar, enfocándose en la comprensión profunda del dominio del problema y en cómo el sistema responderá a los requisitos especificados en cada una de las historias de usuario.

Se comienza con el modelo de dominio, que describe los conceptos principales y las relaciones entre ellos, proporcionando una representación clara y estructurada del dominio del problema. A continuación, se incluye el diagrama de clases del dominio, que ofrece una representación gráfica de las clases conceptuales, facilitando la comprensión de la estructura del sistema propuesto.

Posteriormente, se presenta el modelo de análisis, donde se detalla el comportamiento del sistema mediante la asignación de las operaciones a las clases y la realización en análisis de los casos de uso más importantes, donde se explica en detalle el modelado del comportamiento de los estos casos de uso utilizando las clases del modelo de dominio. Esta sección incluye los diagramas de secuencia y explicaciones sobre la participación de cada clase en la realización de las funcionalidades del sistema.

5.1. Modelo de Dominio

Paran desarrollar el modelo de dominio de AcogeCYL, es necesario definir las clases conteptuales clave, sus atributos, las relaciones que hay entre ellas y las reglas de negocio.

5.1.1. Clases conceptuales

- **Centro**: Representa un centro educativo, como una escuela, instituto o colegio. Contiene información como el nombre del centro, la dirección, el número de teléfono de contacto, etc. Los centros educativos pueden tener múltiples cursos y estar asociados con varios docentes y estudiantes.

- **Curso**: Representa un curso dentro de un centro educativo. Puede contener información como el nombre del curso, el nivel educativo al que pertenece, el docente a cargo del curso, etc. Hay un único docente a cargo del curso y cada curso tiene varios estudiantes matriculados.
- **Docente**: Representa a los profesores o maestros que utilizan la plataforma para crear y gestionar actividades educativas. Un docente puede crear actividades, asignarlas a estudiantes, ver el progreso de las asignaciones, gestionar estados de ánimo y realizar otras acciones relacionadas con la enseñanza. Incluye información como nombre, a qué cuerpo de enseñanza pertenece y cuál es su especialidad.
- **Estudiante**: Representa a un estudiante dado de alta en la plataforma. Contiene información como nombre, apellidos, curso, estado de ánimo actual, nacionalidad, etc. Son capaces de responder a actividades y registrar cómo se sienten en su diario emocional.
- **Actividad**: Es una tarea o recurso educativo creado por un docente para sus estudiantes. Puede contener contenido educativo como título, descripción, contenido multimedia, etc.
- **Asignación**: Representa la materialización de la asignación de una actividad específica a un único estudiante por parte del docente. Incluye información como el estado de la asignación y la fecha de finalización.
- **Pregunta**: Esta entidad representa una pregunta dentro de una actividad educativa. Una actividad puede tener tantas preguntas como el docente quiera.
- **Respuesta**: Representa la respuesta proporcionada por un estudiante a una pregunta dentro de una actividad que le ha sido asignada.
- **Estado de Ánimo**: Son los diferentes estados emocionales que pueden ser dados de alta en el sistema por los docentes. Cada estado de ánimo tiene un nombre, descripción y puede tener una imagen. Cualquier estudiante puede registrar que se siente como un estado de ánimo dado en cualquier momento.
- **Estado de Estudiante**: Representa el estado emocional o cómo se siente un estudiante en un momento determinado. Cada estado de estudiante puede estar asociado con un Estudiante específico y un estado de ánimo registrado en el sistema. Los estados de estudiante pueden ser registrados por los propios estudiantes en la plataforma, lo que les permite expresar sus emociones y proporciona a los docentes una visión más completa de su bienestar emocional.

5.2. Diagrama de clases del Modelo de Dominio

Cogiendo como base la descripción detallada de las clases conceptuales y relaciones en la sección anterior, se puede establecer el Diagrama de Clases del Dominio. Este diagrama representa de manera visual las clases conceptuales y cómo están relacionadas entre sí. Es

importante destacar que este diagrama servirá como base para la creación del modelo conceptual de la base de datos, ayudando a definir las tablas y relaciones que serán implementadas en el sistema. Se puede ver el modelo de análisis en la Figura 5.1.

La mayoría de clases del dominio tienen un atributo identificador ya que no existe ningún atributo intrínseco que garantice su unicidad. Puede haber varios Centros con el mismo nombre, Actividades con el mismo nombre, Estados de ánimo con el mismo nombre del estado etc. Esta característica nos interesa ya que el modelo de análisis se empleará como base del modelo conceptual de la base de datos, por ello se añaden los atributos *id*.

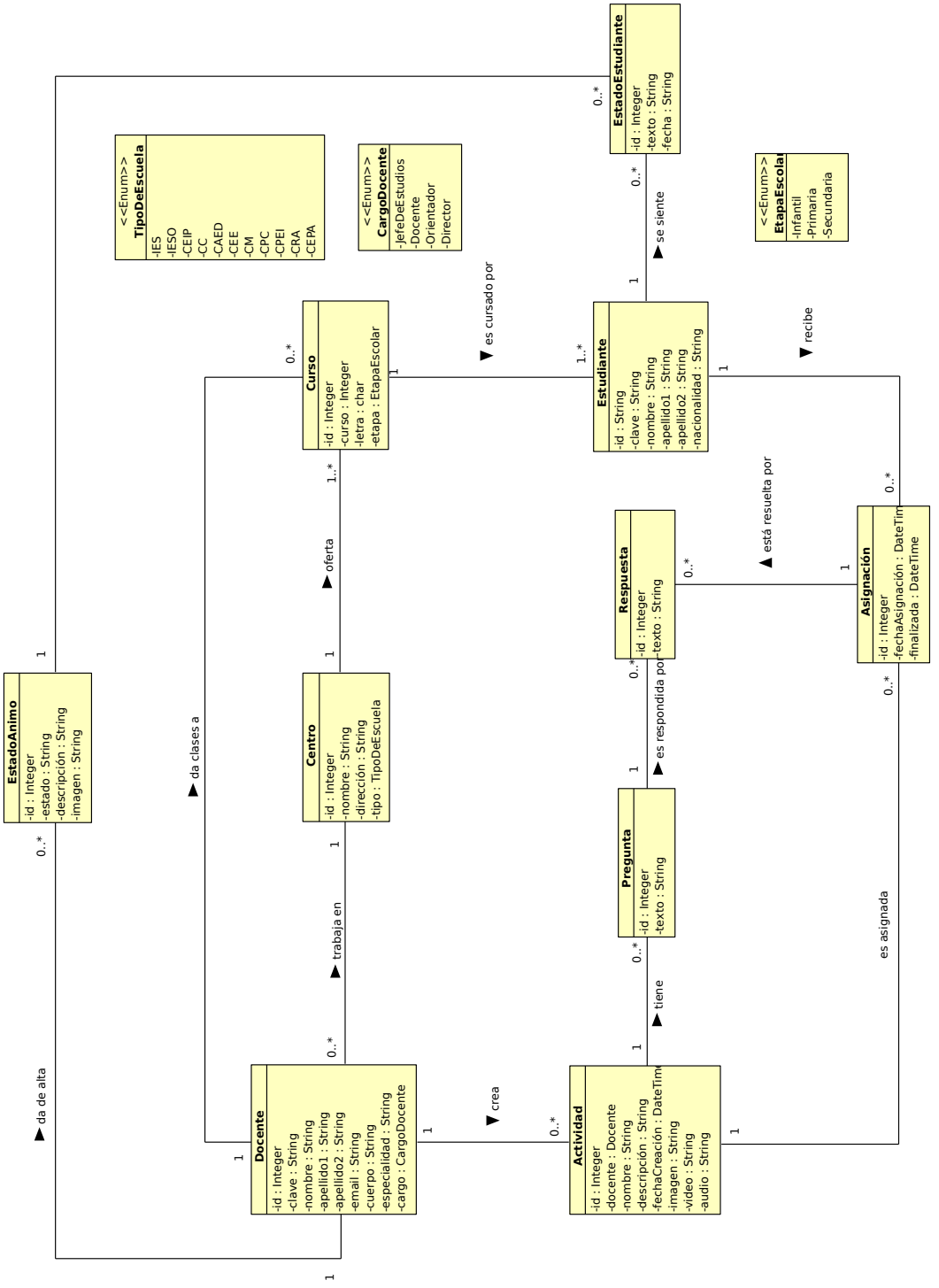

Figura 5.1: Diagrama de clases del dominio de análisis.

5.3. Modelo de Análisis

5.3.1. Clases de Análisis

En esta sección se pretenda proporcionar una visión detallada de las entidades principales, sus atributos, relaciones y comportamiento asociado en el contexto de la aplicación. Este modelo se construye a partir de las historias de usuario y requisitos identificados en el capítulo 4. La principal contribución del Modelo de Análisis respecto al Modelo de Dominio que se acaba de presentar, es que no solo define las clases conceptuales, sino que también modela el comportamiento de cada clase.

Empleamos las entidades identificadas en la sección 5.1 y las definimos de manera más detallada, incluyendo sus atributos, relaciones y funcionalidades asociadas:

- **Centro**: Atributos: Identificador, nombre, dirección y tipo de escuela.

  Responsabilidades: Obtener todos los cursos del centro.

- **Curso**: Atributos: Identificador, curso, letra, etapa, centro y docente.

  Responsabilidades: Obtener estudiantes de un curso, obtener el año del curso, obtener la letra del curso.

- **Docente**: Atributos: Identificador, clave, nombre, apellidos, email, cuerpo, especialidad, cargo y centro, registrar.

  Responsabilidades: Establecer datos, establecer nueva Actividad creada, establecer nuevo Estado de Ánimo creado, obtener identificador, obtener clave, obtener detalles, registrar.

- **Estudiante**: Atributos: Identificador, clave, nombre, apellidos, nacionalidad, curso.

  Responsabilidades: Establecer datos, establecer un nuevo Estado de Estudiante, establecer una nueva Asignación, obtener estudiantes a los que imparte clase un docente dado, obtener identificador, obtener clave, obtener detalles, registrar.

- **Actividad**: Atributos: Identificador, docente, nombre, descripción, fecha de creación, imagen, video, audio.

  Responsabilidades: Establecer datos, modificar datos, añadir Pregunta, añadir una Asignación, obtener todas las Actividades, obtener las Actividades creadas por un Docente dado, obtener las Actividades pendientes dado un Estudiante/Familia, registrar.

- **Asignación**: Atributos: Identificador, actividad, estudiante, fecha de asignación y si está finalizada.

  Responsabilidades: Establecer datos, establecer la Actividad, establecer el Estudiante, establecer actividad como finalizada, comprobar si está finalizada, obtener Asignación dado el identificador.

- **Pregunta**: Atributos: Identificador, texto, actividad.

  Responsabilidades: Establecer datos, establecer la Actividad, obtener las Preguntas dada una Actividad, obtener el texto de la Pregunta.

- **Respuesta**: Atributos: Identificador, texto, pregunta, asignacion.

  Responsabilidades: Establecer datos, establecer la Pregunta, establecer la Asignación, obtener el texto de la Pregunta, obtener las Respuestas dada la Asignación, registrar.

- **Estado de Ánimo**: Atributos: Identificador, docente, estado, descripción, imagen.

  Responsabilidades: Establecer datos, establecer el Docente, obtener todos los Estados de Ánimo creados por un Docente dado, obtener todos los Estados de Ánimo, obtener el Estado de Ánimo dado el nombre, obtener detalles, registrar.

- **Estado de Estudiante**: Atributos: Identificador, estado, estudiante, texto y fecha.

  Responsabilidades: Establecer datos, establecer el Estudiante, establecer el Estado, obtener detalles, obtener todos los Estados de Estudiante de un Estudiante dado, registrar.

5.3.2. Realización de casos de uso de Análisis

En esta sección se presenta la realización de tres de los casos de uso más importantes de la aplicación AcogeCYL. Se ha decidido emplear el Diagrama de Secuencia de UML [51] para modelar la realización de los casos de uso. A través de estos diagramas, podemos comprender mejor cómo se llevan a cabo los procesos clave dentro de la aplicación y asegurarnos de que todos los componentes interactúan de manera coherente y eficiente.

5.3.3. UC05. Crear una actividad

Las clases de análisis involucradas en la realización del caso de uso se muestran en la Figura 5.2 y el diagrama de secuencia en la Figura 5.3.

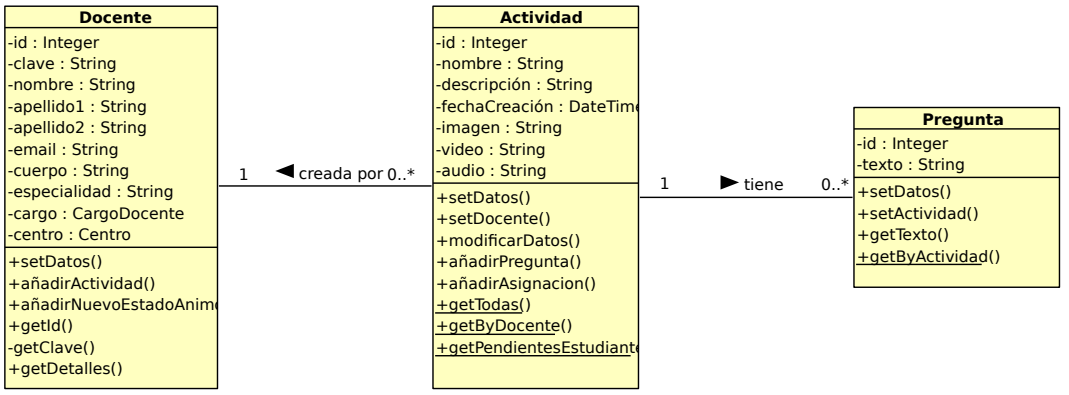

Figura 5.2: Clases que intervienen en el caso de uso UC05. Crear una actividad

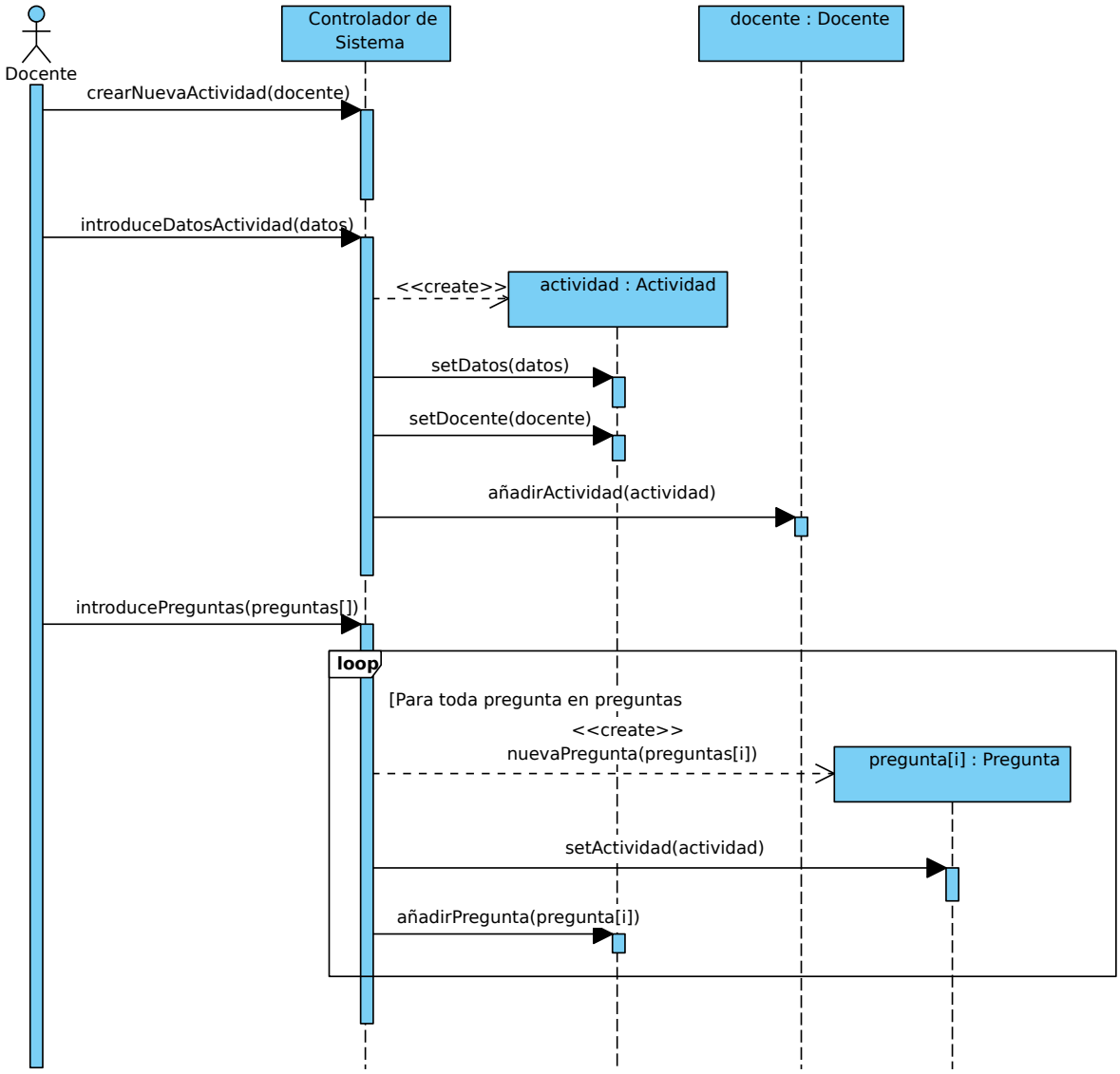

Figura 5.3: Diagrama de secuencia del caso de uso UC05. Crear una actividad

5.3.4. UC16. Ver Estado de Ánimo actual de Estudiantes

Las clases de análisis involucradas en la realización del caso de uso se muestran en la Figura 5.4 y el diagrama de secuencia en la Figura 5.5.

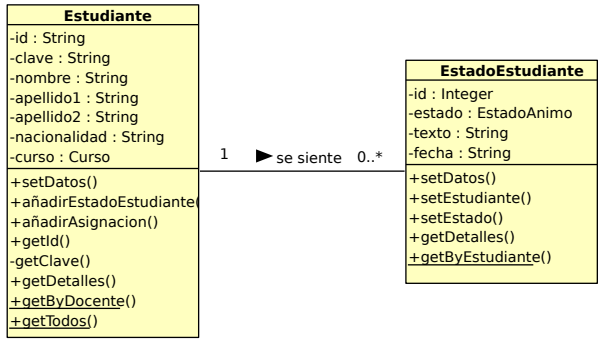

Figura 5.4: Clases que intervienen en el caso de uso UC16. Ver estado de ánimo actual de estudiantes

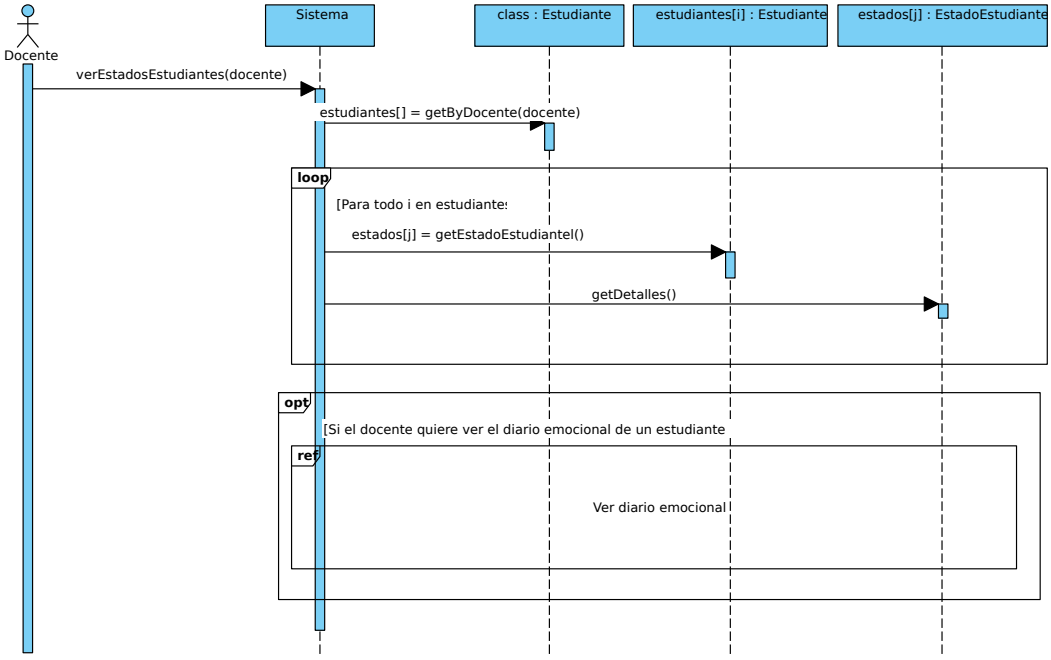

Figura 5.5: Diagrama de secuencia del caso de uso UC16. Ver estado de ánimo actual de estudiantes

5.3.5. UC18. Registrar un estado de ánimo

Las clases de análisis involucradas en la realización del caso de uso se muestran en la Figura 5.6 y el diagrama de secuencia en la Figura 5.7.

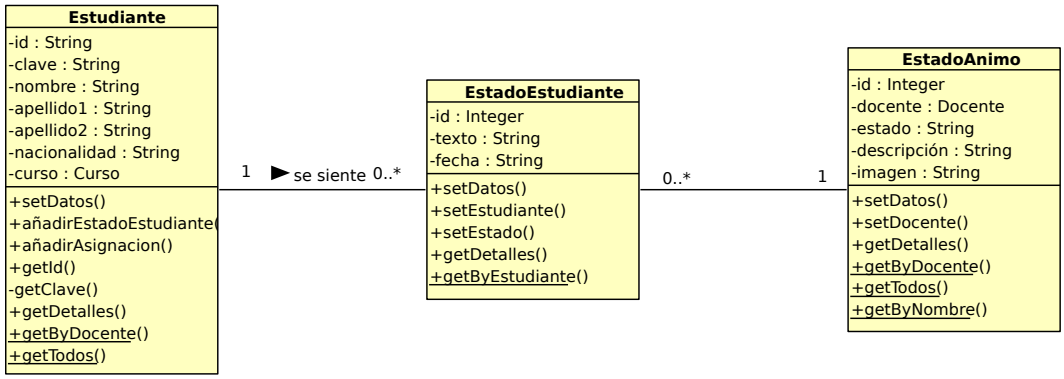

Figura 5.6: Clases que intervienen en el caso de uso UC18. Registrar un estado de ánimo

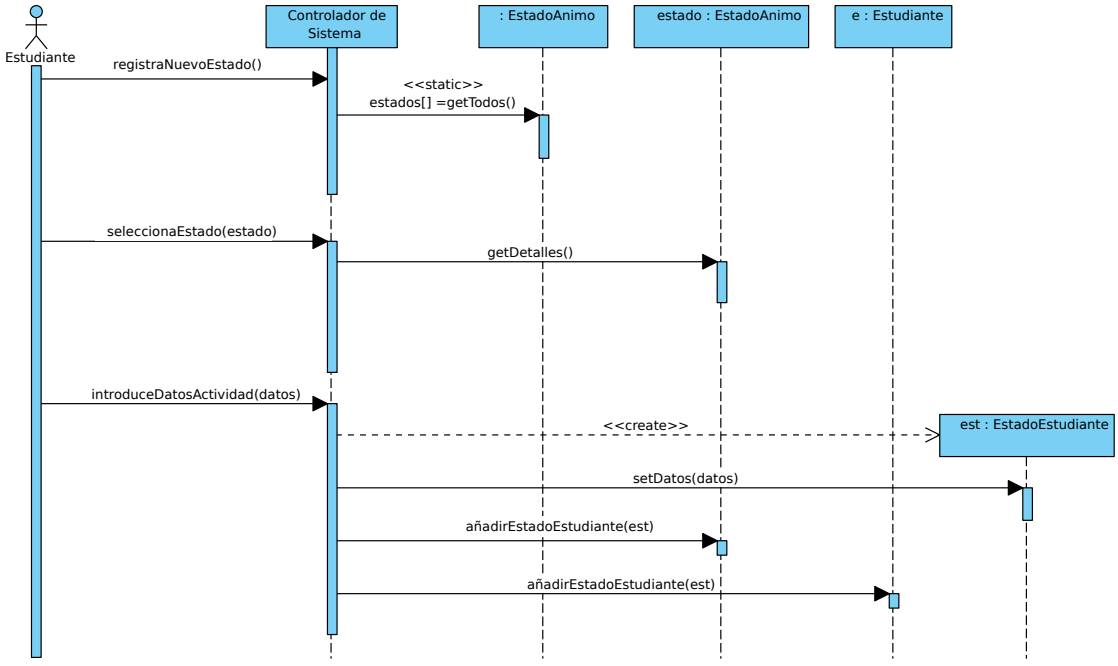

Figura 5.7: Diagrama de secuencia del caso de uso UC18. Registrar un estado de ánimo

Capítulo 6

Diseño

Este capítulo proporcionará una visión detallada de diversos aspectos del *workflow* de Diseño. En primer lugar, se describirá la arquitectura física del sistema y su despliegue. En segundo lugar, se presentará la arquitectura lógica del sistema, describiendo la estructura general y la interacción entre los diferentes componentes.

Dentro de la arquitectura lógica, se discutirán los patrones de diseño implementados, explicando cómo estos patrones contribuyen a la robustez y flexibilidad del sistema, que permiten establecer las clases de diseño, explicando sus atributos, métodos y las relaciones entre ellas.

Asimismo, se incluirá una sección dedicada al diseño de la interfaz gráfica, donde se detallarán los bocetos que modelan la interacción persona computadora.

También se describirá el diseño de la base de datos, incluyendo el modelo de datos y la estructura de las tablas y relaciones para garantizar una gestión eficiente de la información.

Para concluir, se llevará a cabo la realización de un caso de uso de diseño.

6.1. Arquitectura física del sistema

La arquitectura física del sistema describe el hardware disponible para el sistema y cómo los componentes de software se implementan en él. Se realiza un mapeo de los elementos software sobre la arquitectura del hardware físico.

El diagrama estructural que plasma la arquitectura fisica es el diagrama de despliegue 6.1:

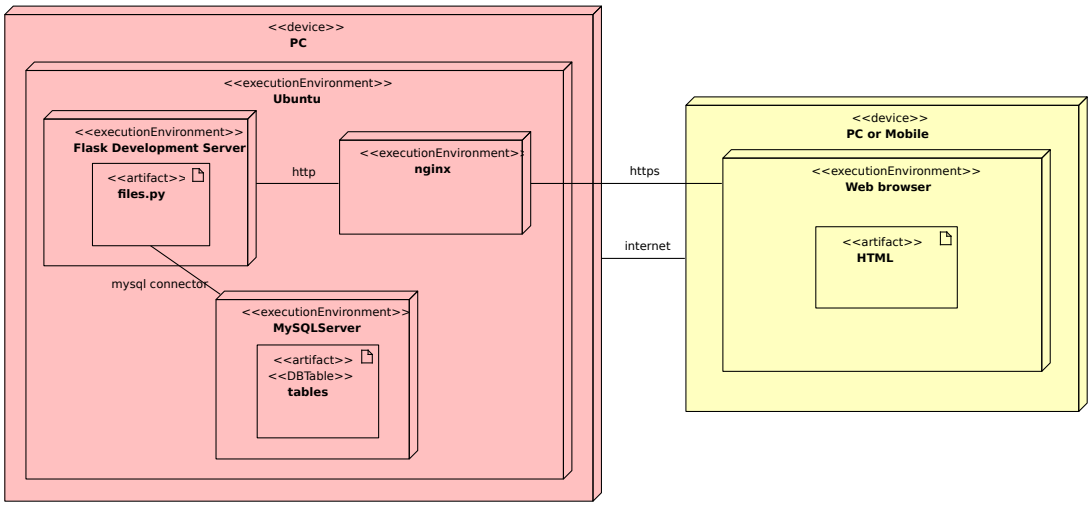

Figura 6.1: Diagrama de despliegue de la arquitectura física.

La aplicación de Flask está alojada en una máquina Ubuntu, en la que también están alojados la base de datos y el servidor web nginx que actúa como *proxy* [50] de la aplicación para que esta sea accesible. La base de datos será accesible para los ficheros de la aplicación gracias al conector *MySQL connector* para Python. Flask y el servidor nginx se comunican mediante el protocolo HTTP.

La máquina Ubuntu actúa como servidor para cualquier cliente que intente acceder por internet. La aplicación generará dinámicamente archivos HTML que serán enviados de manera remota como respuesta a las peticiones de los clientes mediante el protocolo HTTPS.

6.2. Arquitectura lógica del sistema

Como lenguaje de programación se ha empleado Python [7] y Flask [8] como *framework* de desarrollo de páginas web. La arquitectura lógica elegida para la aplicación se basa en el patrón arquitectónico capas. En concreto se han establecido tres capas. Esta arquitectura lógica asegura una organización clara y coherente de los componentes, facilitando la escalabilidad, mantenibilidad y robustez del sistema [49].

El patrón arquitectónico de capas, también conocido como arquitectura de capas, es un enfoque para diseñar sistemas de software que separa la funcionalidad en capas distintas y bien definidas. Cada capa representa un nivel de abstracción, y las capas están organizadas de manera jerárquica, con capas superiores que dependen de las capas inferiores. La arquitectura de capas facilita la separación de responsabilidades y la modularidad del sistema, lo que permite cambios en una capa sin afectar a otras capas. Además, promueve la reutilización de componentes y facilita la escalabilidad al permitir que cada capa se pueda escalar de forma independiente. [49]

La aplicación AcogeCYL se organiza en tres capas principales: la capa de datos, la capa de dominio y la capa de servicios. Cada capa tiene una responsabilidad específica y se comunica con las otras capas mediante interfaces bien definidas.

1. **Capa de Acceso a Datos**: La capa de datos es responsable de la gestión de la persistencia de la información. Aquí se encuentra la bases de datos que permiten el almacenamiento y la recuperación de datos. La interacción con la base de datos está proporcionado mediante los patrones de Diseño DAO-DTO que se verán más adelante [49].
2. **Capa de dominio**: La capa de dominio encapsula la lógica de negocio y las reglas operativas del sistema. Esta capa es crucial para asegurar que las operaciones del sistema sigan las reglas y procesos definidos. Se ha utilizado el patrón *Transaction Script* para estructurar la lógica de negocio [49].
3. **Capa de Servicios**: La capa de servicios es la encargada de proporcionar la funcionalidad del sistema a los usuarios y a otros sistemas externos. Esta capa actúa como intermediario entre la capa de dominio y los clientes (interfaces de usuario o APIs).

A mayores, se ha incluido una Capa de Utilidades. Esta capa incluye funciones que proporcionan funcionalidades comunes y reutilizables en diferentes partes de la aplicación.

Interacción entre capas

La interacción entre las tres capas principales sigue una estructura de capas cerradas, donde cada capa solo puede comunicarse directamente con la capa inmediatamente inferior. La capa de servicios solo se comunica con la capa de dominio y la capa de dominio solo se comunica con la capa de datos. Si tenemos en cuenta la capa de utilidades, todas las capas usan las funcionalidades que ofrece. La comunicación entre las capas se realiza a través de interfaces bien definidas, lo que facilita la separación de responsabilidades y la modularidad del sistema.

6.2.1. Diagrama de paquetes

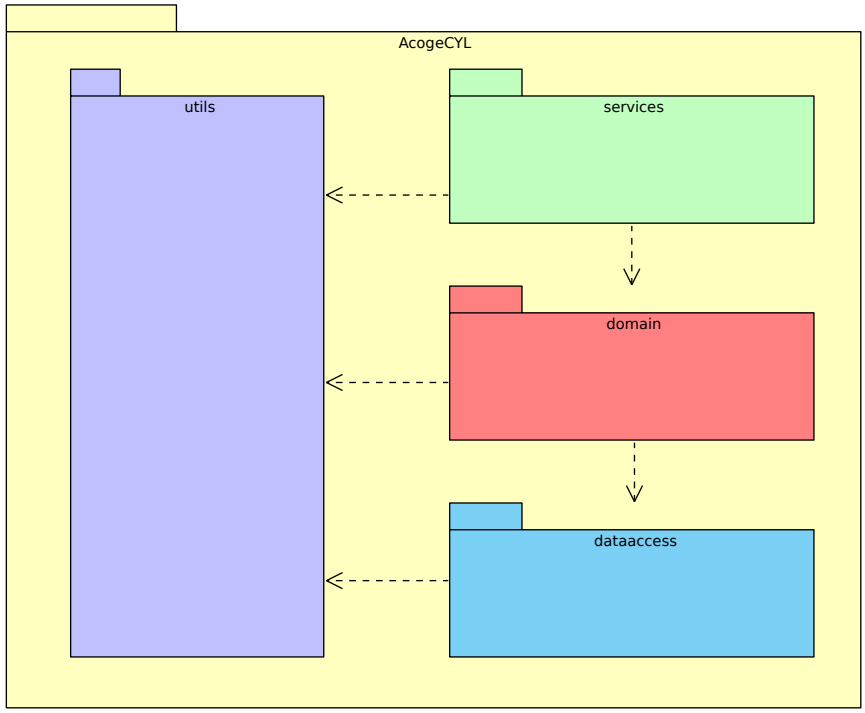

Figura 6.2: Diagrama de paquetes

6.3. Patrones de diseño

6.3.1. Capa de servicios

En la Capa de Servicios de la aplicación, se han aplicado dos patrones en conjunto: *Page Controller* y *Template View* [49] aprovechando las características de Flask.

Patrón *Page Controller*

El patrón *Page Controller* se utiliza para gestionar la lógica de control de cada página individual. Cada página de la aplicación web tiene asignado un método que funciona como una especie de controlador lo que permite una organización clara y manejable del código. El *Page Controller* se encarga de atender a la petición HTTP, actualiza el modelo y decide qué vista mostrar. Por otro lado, el patrón *Template View* se encarga de la representación de las vistas, separando la lógica de presentación del resto del código.

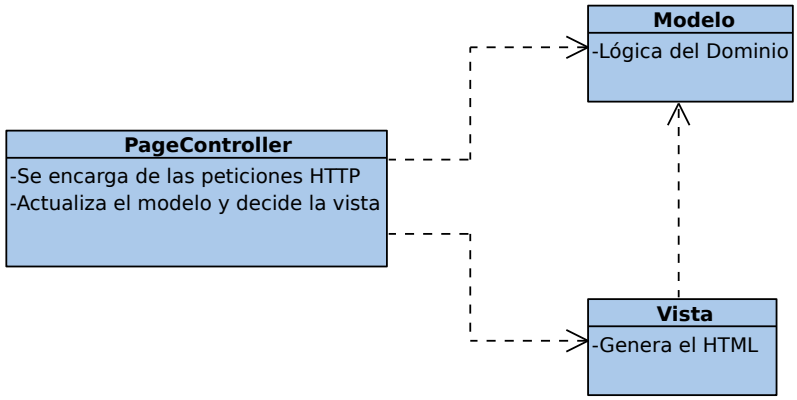

Figura 6.3: Patrón *Page Controller*

La aplicación conjunta de *Page Controller* y *Template View* funciona muy bien para este último patrón, pero puede ser algo incómodo estructurar el módulo para el patrón *Page Controller*. La solución más común de resolver este problema es llamar a un objeto *helper* que se encargue de toda la lógica [49]. En esta aplicación es la capa de dominio la que se encargará de la lógica.

Patrón *Template View*

La idea básica del patrón *Template View* es embeber marcadores en un HTML estático cuando se está construyendo para poder incluir información dinámicamente. Cuando la página se use para atender una solicitud, los marcadores se sustituyen por los resultados de algún cálculo, en el caso de la aplicación mostrará los resultados de consultas a la base de datos. *Flask* funciona con el motor de plantillas *Jinja2* por lo que puede ir un paso más allá y embeber *scriplets* similares a Python en la página para la inclusión de los datos dinámicos mencionados.

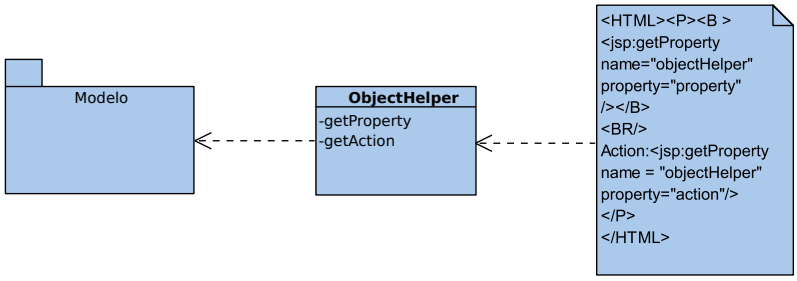

Figura 6.4: Patrón *Template View*

La clave para evitar escribir *scriplets* demasiado grandes y poco manejables es emplear un objeto *helper* que se encargue de toda la lógica de programación para mejorar la modularidad. El objeto *helper* coincide con el *helper* empleado en el patrón *Page Controller*, la capa de

dominio.

6.3.2. Capa de dominio

Transaction Script

En la capa de dominio de la aplicación, se ha adoptado el patrón de diseño *Transaction Script* [49], el cual se caracteriza por representar cada transacción como un *script* independiente que encapsula toda la lógica de negocio relacionada con dicha transacción. Como consecuencia de esta elección, las clases del dominio se centran en representar los datos y no contienen operaciones complejas asociadas. En lugar de eso, la lógica de negocio se implementa en *transaction scripts*, que manipulan los datos a través de estas clases, manteniendo así una estructura clara y directa que facilita la comprensión y el mantenimiento del código. Este enfoque simplificado es especialmente adecuado para aplicaciones con una lógica de negocio relativamente sencilla, como es el caso de este proyecto.

En la arquitectura de la aplicación, cada caso de uso está respaldado por un TS (*Transaction Script*) dedicado que implementa la lógica de negocio asociada a ese caso de uso específico. Cada TS actúa como un controlador de una transacción independiente que gestiona la ejecución de la funcionalidad relacionada con su caso de uso correspondiente. Esta implementación asegura una separación clara de las responsabilidades, donde cada TS se enfoca en una tarea particular y encapsula toda la lógica de negocio necesaria para llevar a cabo esa tarea de manera eficiente y coherente. Además, al seguir el patrón *Transaction Script*, se evita la complejidad innecesaria en las clases del dominio, lo que simplifica el diseño y la mantenibilidad del sistema en su conjunto.

En la Figura 6.5 se pueden ver todas las clases de diseño de la Capa de Dominio.

6.3.2.1. Detalles de una clase de diseño

`ResolverActividadTS`: Script que se encarga de la lógica del caso de uso UC19. Resolver una actividad.

Operaciones:

- `getActividadById()`: Obtiene la Actividad que tenga el identificador especificado.
- `getPreguntasActividad()`: Obtiene todas las Preguntas que tenga la Actividad.
- `guardaRespuestas()`: Guarda las Respuestas de la Asignación especificada.
- `finalizaAsignacion()`: Marca como finalizada la Asignación especificada.
- `getDatosActividad()`: Obtiene los detalles y las Preguntas asociadas a una Actividad.
- `resolverActividad()`: Guarda las Respuestas y finaliza la Asignación.

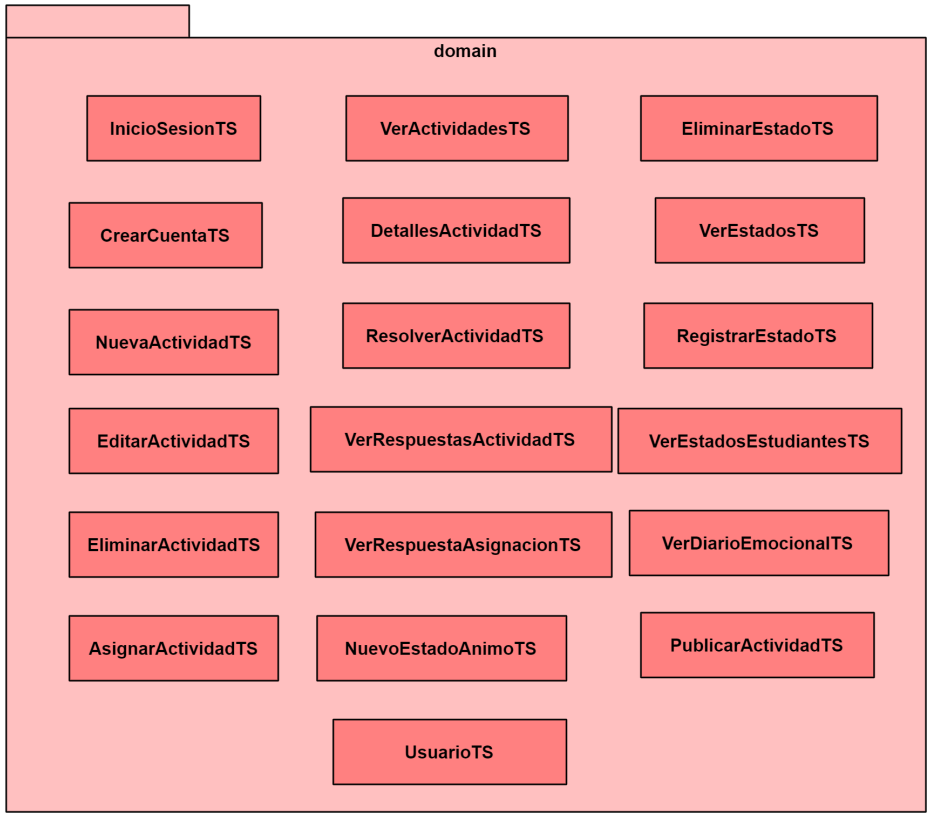

Figura 6.5: Clases de diseño de la Capa de Dominio.

6.3.3. Capa de Acceso a Datos

Patrón Singleton

El patrón Singleton [50] es un patrón de diseño creacional que garantiza que una clase tenga un único objeto y proporciona un punto de acceso global a dicha instancia. Es especialmente útil cuando se necesita un único objeto para coordinar acciones en todo el sistema. Este patrón es particularmente útil en esta aplicación ya que se quiere asegurar que solo haya una conexión a la base de datos, evitando múltiples conexiones innecesarias y manteniendo un control centralizado de este recurso.

Al restringir la creación de instancias a una sola dentro del contexto de la aplicación, el patrón Singleton promueve la eficiencia en el acceso a la base de datos y evita problemas de concurrencia al asegurar que todas las partes del programa accedan a la misma instancia de manera consistente. Esto simplifica la gestión del acceso a la base de datos y facilita la mantenibilidad del código al centralizar la lógica de creación y acceso a la instancia única.

Patrón DAO-DTO

El patrón DAO-DTO en realidad son dos patrones distintos que suelen ser comúnmente utilizados en conjunto, especialmente en los sistemas que siguen el paradigma de arquitectura de tres capas, como en este caso. Ambos patrones sirven para abordar diferentes preocupaciones relacionadas con el acceso a datos y la transferencia de información entre las capas de una aplicación.

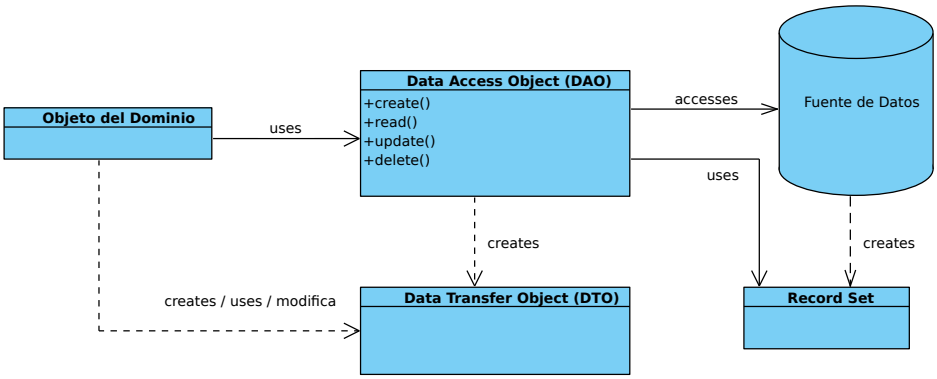

Figura 6.6: Relaciones del patrón DAO-DTO.

El patrón DAO (*Data Access Object*) se utiliza para separar la lógica de acceso a los datos de la lógica de negocio. Los DAOs encapsulan todas las operaciones de acceso a la base de datos mediante operaciones CRUD (*Create, Read, Update, Delete*). Los DAOs proporcionan una API limpia para el resto de la aplicación, en este caso la capa de dominio, ya que es la única que puede interactuar con la capa de datos. Los DAOs emplean DTOs, objetos que se usan para transferir datos entre diferentes capas de la aplicación.

Los DTOs (*Data Transfer Object*) son objetos simples que solo contienen propiedades y no tienen lógica de negocio. Se ha decidido que los DTOs solo transporten JSON, una práctica común en las aplicaciones web con el objetivo de mejorar la consistencia, la simplicidad de la implementación y el desacomplamiento [49].

Empleando los dos patrones juntos, se consigue una interfaz única de acceso para la base de datos que nos garantiza que no haya acoplamiento entre capas. La Figura 6.6 muestra cómo funcionan los DAOs y DTOs conjutamente.

En la Figura 6.7 se pueden ver todas las clases de diseño de la Capa de Acceso a Datos:

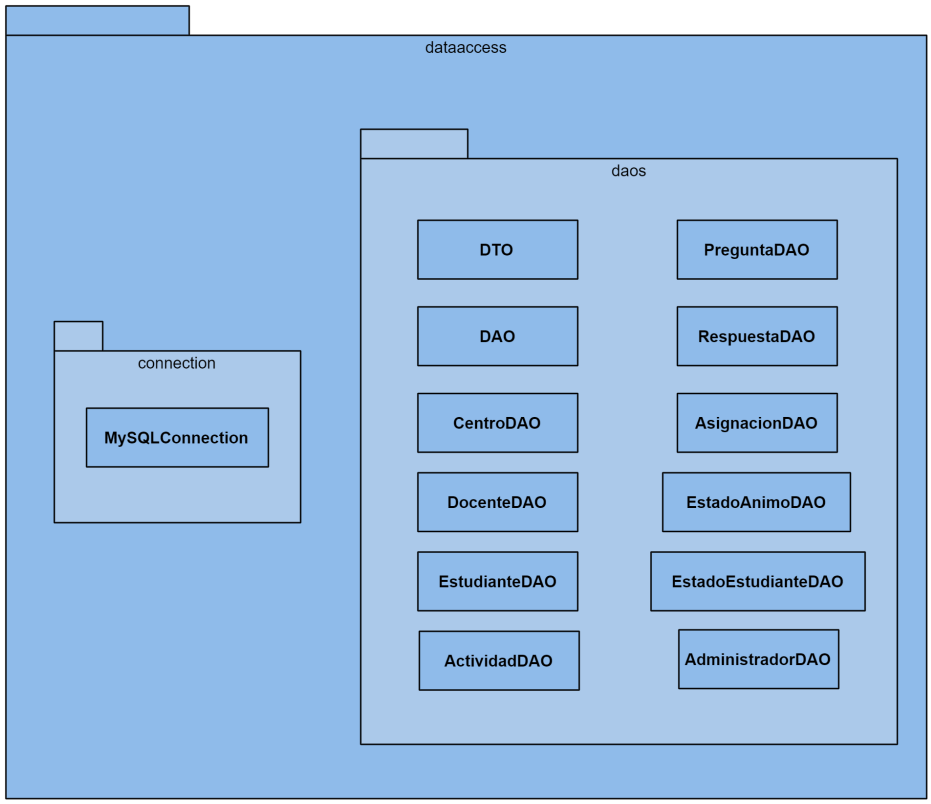

Figura 6.7: Clases de diseño de la Capa de Acceso a Datos.

6.3.3.1. Detalles de una clase de diseño

`ActividadDAO`: Es la clase encargada de realizar las consultas a la base de datos cuando la tabla Actividad está involucrada.

Operaciones:

- `inserta()`: Introduce en la base de datos una Actividad dado.
- `actualiza()`: Actualiza una Actividad dada su identificador.
- `getByDocente()`: Obtiene todas las Actividadas creadas por un Docente.
- `getByDocenteEstudiante()`: Dado un Docente, obtiene todas las actividades a las que ha respondido al menos un Estudiante al que imparte clase.
- `getPendientes(familiar)`: Dado un Estudiante (`familiar == false`) o Familia (`familiar == true`), obtiene todas las Actividades asignadas que aún no ha respondido.
- `getByEstudiante()`: Obtiene todas las Actividades pendientes de un Estudiante.

- `getByFamilia()`: Obtiene todas las Actividades pendientes de una Familia.
- `getById()`: Obtiene una Actividad dado su identificador.
- `getTodos()`: Obtiene todas las Actividades registradas.
- `elimina()`: Elimina una Actividad dado su identificador.
- `publica()`: Asigna una Actividad a todos los Estudiantes registrados.
- `getArchivos()`: Obtiene los nombres de los archivos multimedia de la Actividad.

6.3.4. Capa de Utilidades

La capa de utilidades es una colección de funciones y constantes que proporcionan diversas funcionalidades reutilizables para el resto de capas del sistema. Esta capa incluye herramientas auxiliares, como funciones de tratamiento de archivos o manipulación de fechas. Al centralizar estas operaciones, se promueve la modularidad y reutilización del código, evitando *boilerplate code* y facilitando el mantenimiento y la extensión.

6.4. Diseño de la interfaz gráfica

Antes de proceder con la implementación de cualquier caso de uso, se realiza un *mockup* detallado de las distintas pantallas que conforman la interfaz de usuario. Estas representaciones visuales del diseño de la interfaz permiten planificar y evaluar la disposición y funcionalidad de los elementos gráficos antes de escribir código.

Las figuras 6.8-6.39 muestran los *mockups* que se fueron realizando. Estos bocetos representan las páginas que se generan mediante los controladores del patrón *Page Controller* y las plantillas del patrón *Template View* presentadas en la Sección 6.3.1. En el pie de foto de cada uno de los bocetos se indica el caso de uso al que corresponde.

En las Figuras 6.8, 6.9 y 6.10 se muestran los bocetos de los casos de uso UC01. Registro de docente y UC02. Registro de estudiante. En la Figura 6.8 se solicita la elección entre registrarse como Docente o como Estudiante y en las otras dos se muestran los bocetos de los formularios de registro.

Figura 6.8: Seleccionar rol.

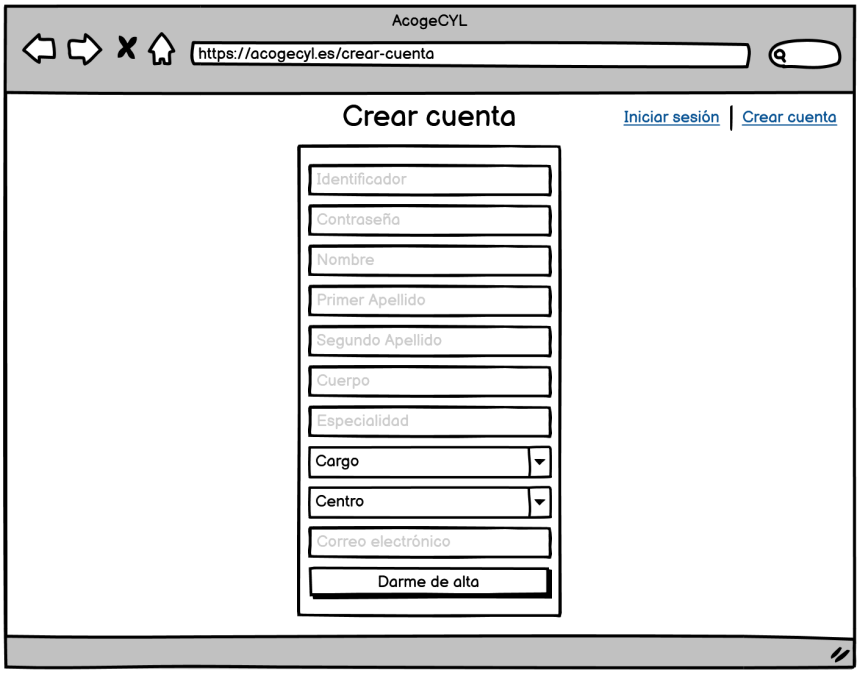

Figura 6.9: Crear cuenta - *Docente*.

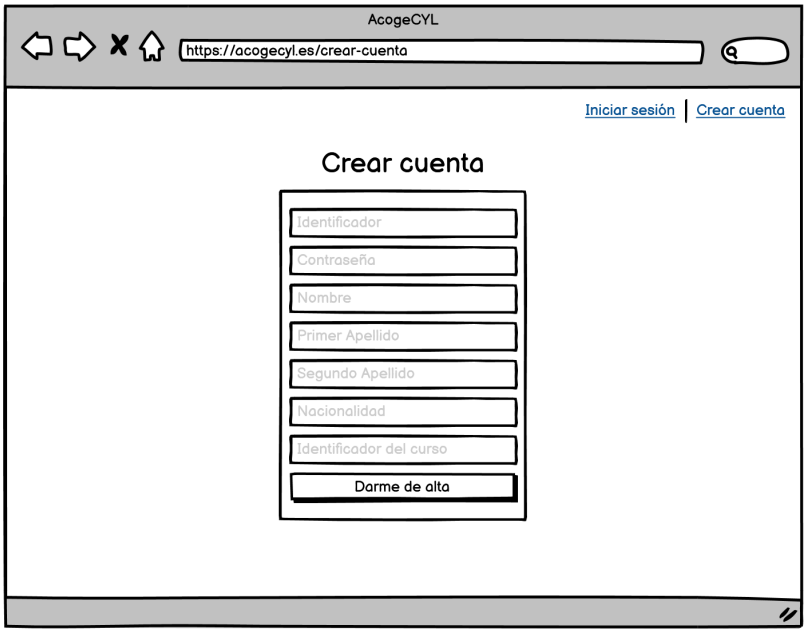

Figura 6.10: Crear cuenta - *Estudiante*.

En la Figura 6.11 se puede ver el *mockup* hecho sobre el UC03. Iniciar sesión.

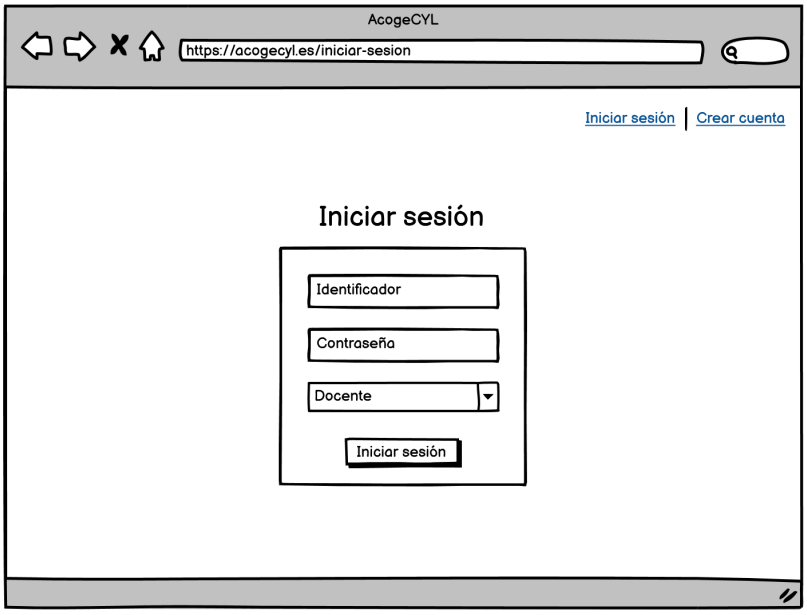

Figura 6.11: Iniciar sesión.

En las Figura 6.12 se muestran el boceto del menú principal del Docente. Si se pulsa en la tarjeta de *Actividades*, se pasa a la pantalla bocetada por la Figura 6.13.

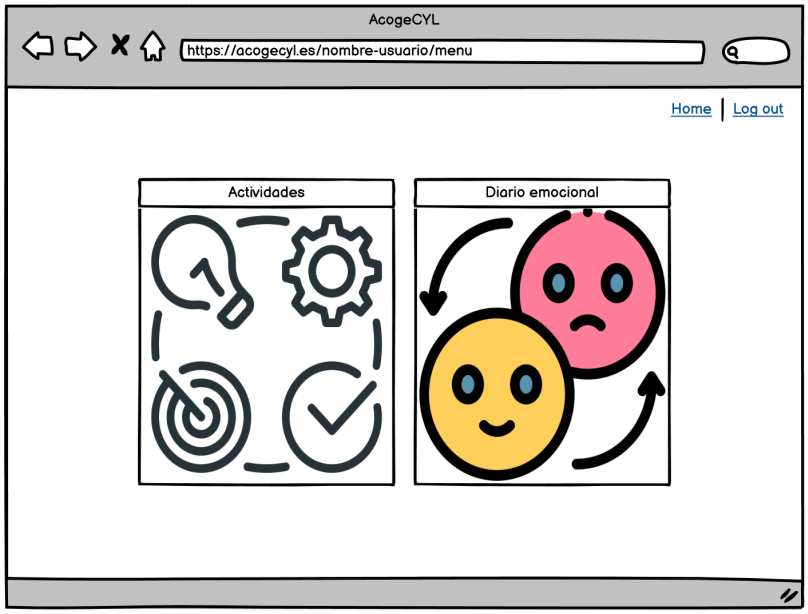

Figura 6.12: Menú principal del Docente.

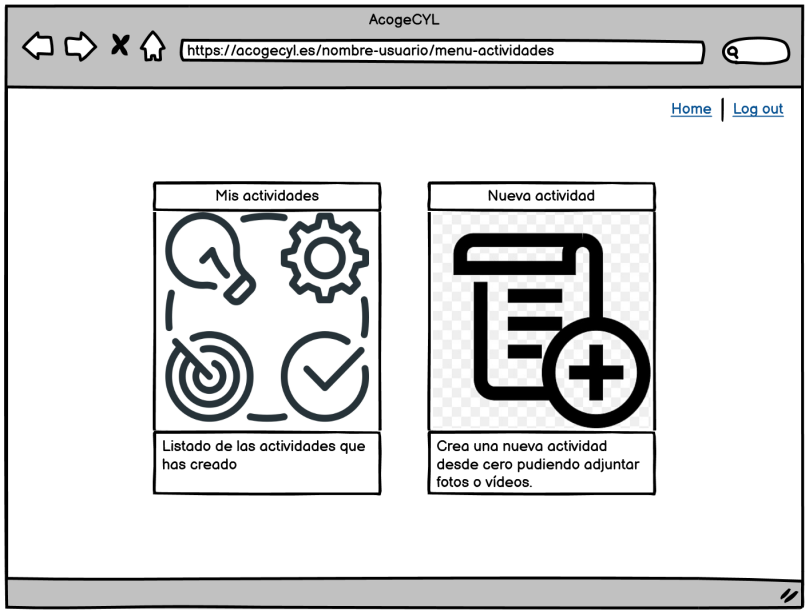

Figura 6.13: Menu Actividades del Docente.

Si en el menú del *mockup* mostrado en la Figura 6.13 se accede a *Nueva actividad*, arranca el UC05. Crear una actividad en la pantalla bocetada en la Figura 6.14 y termina en la Figura 6.18.

Figura 6.14: Crear un Actividad.

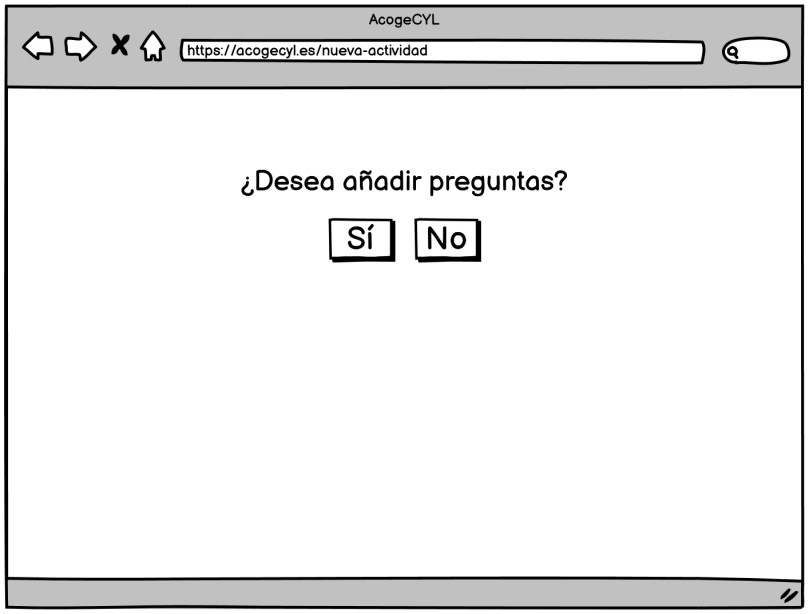

Figura 6.15: Crear una Actividad - Añadir Preguntas.

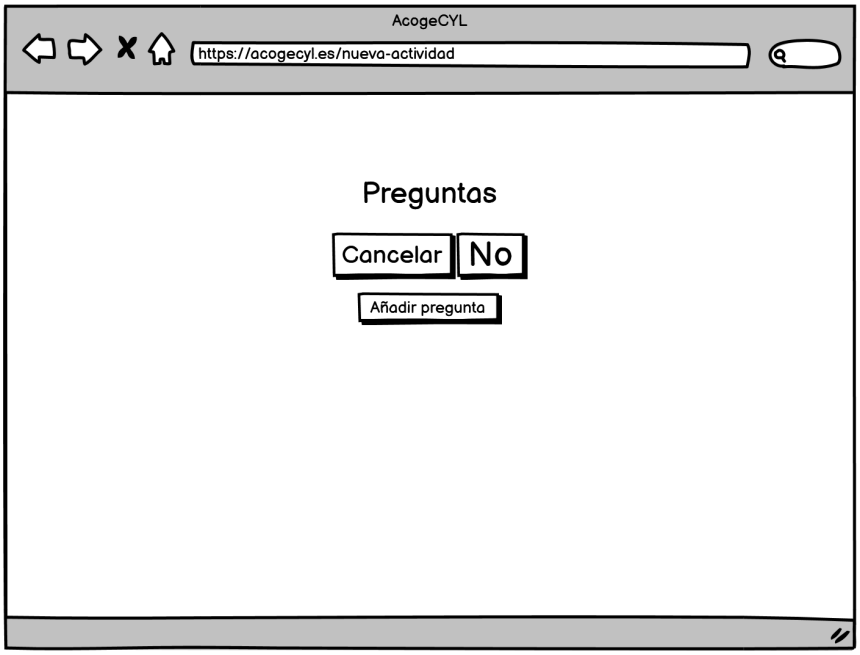

Figura 6.16: Crear una Actividad - Añadir Preguntas vacío.

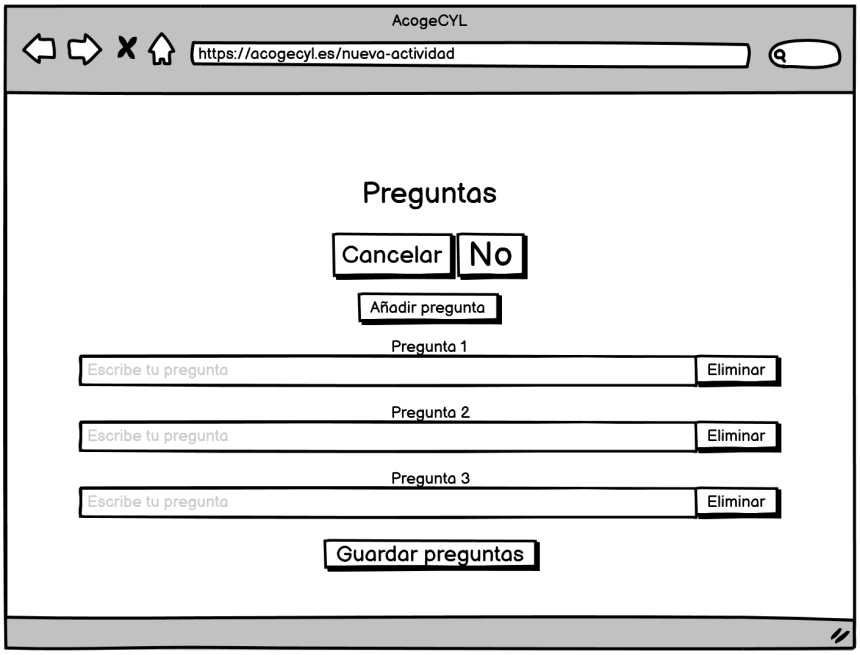

Figura 6.17: Crear una Actividad - Añadir varias Preguntas.

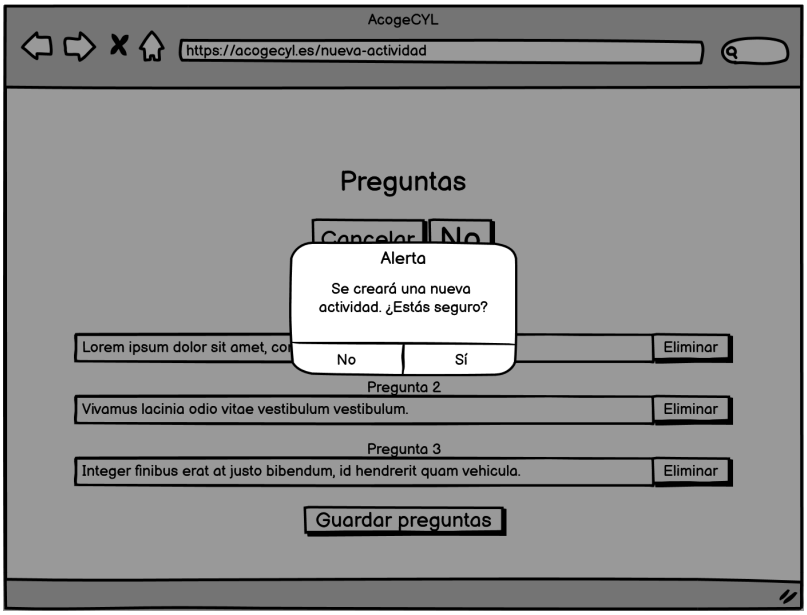

Figura 6.18: Crear una Actividad - Confirmar Preguntas.

Si en el menú bocetado en la Figura 6.13 se accede a *Mis actividades*, se ejecuta el UC04. Ver lista de actividades y se muestran las Actividades como en la Figura 6.19.

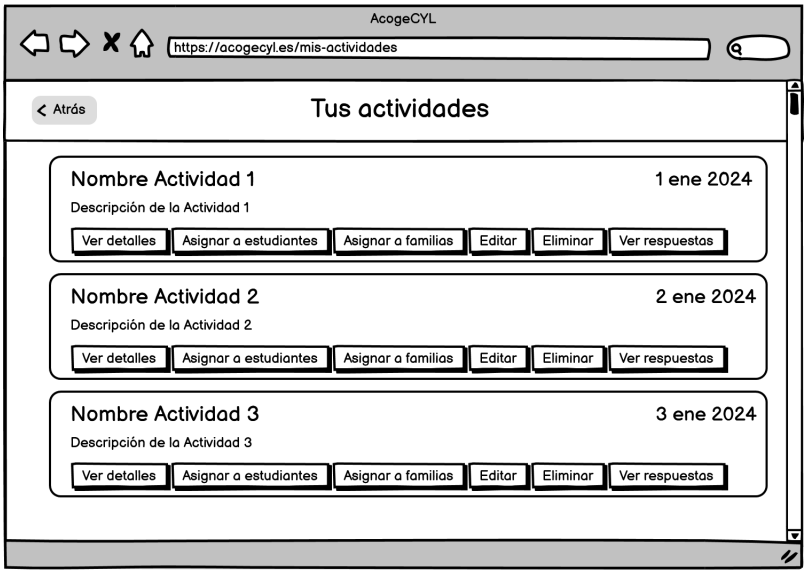

Figura 6.19: Lista de Actividades - *Docente*.

En las Figuras 6.20 y 6.21 se puede ver el boceto de los detalles de una Actividad de UC06. Ver detalles de una actividad.

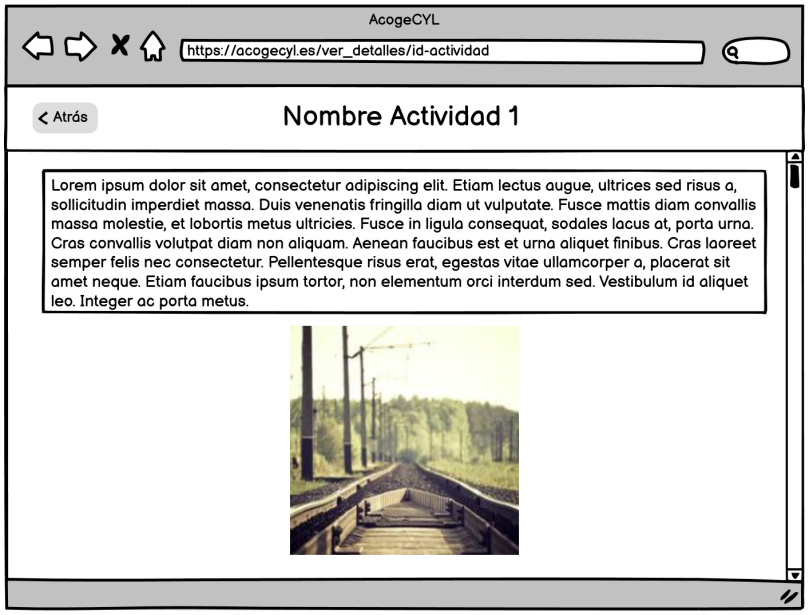

Figura 6.20: Detalles de Actividad parte superior.

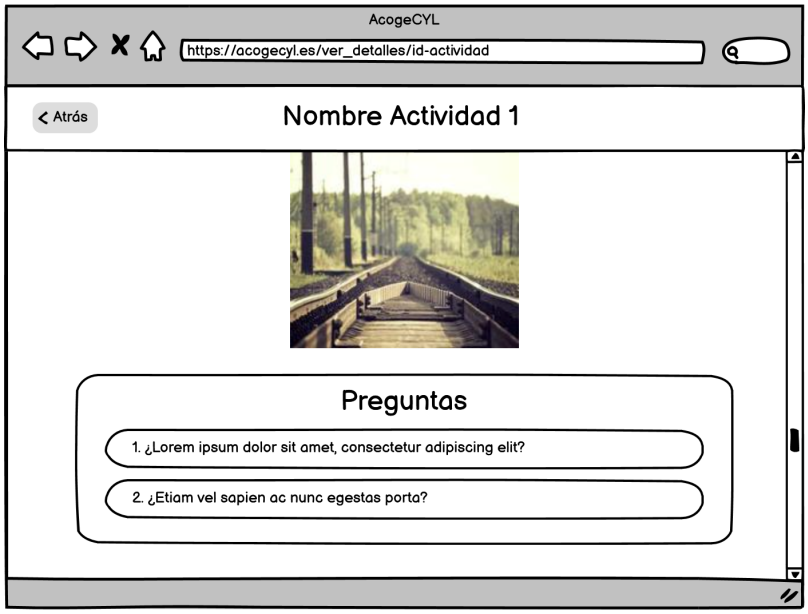

Figura 6.21: Detalles de Actividad parte inferior.

En la Figura 6.22 se muestra el *mockup* de la pantalla de Asignación a Estudiantes para implementar el UC08. Asignar una actividad.

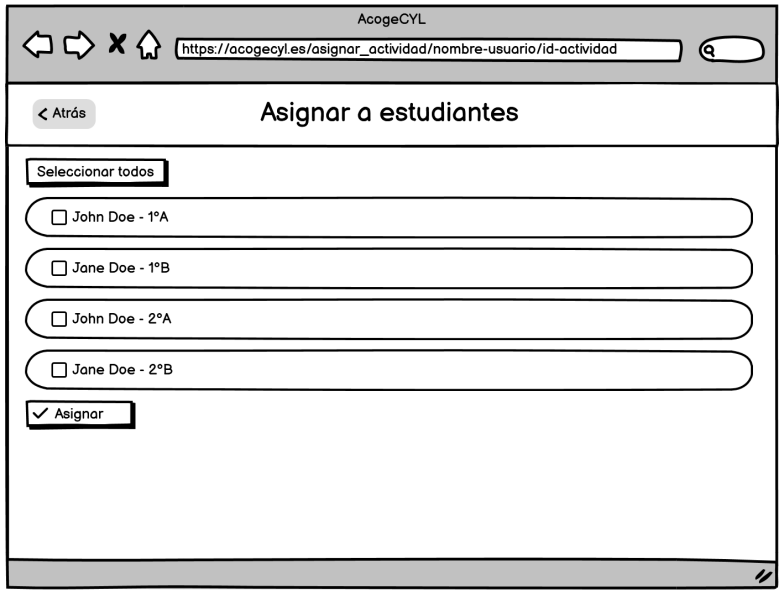

Figura 6.22: Asignar una actividad.

En la Figura 6.23 se muestra el *mockup* del diseño de UC07. Editar una actividad.

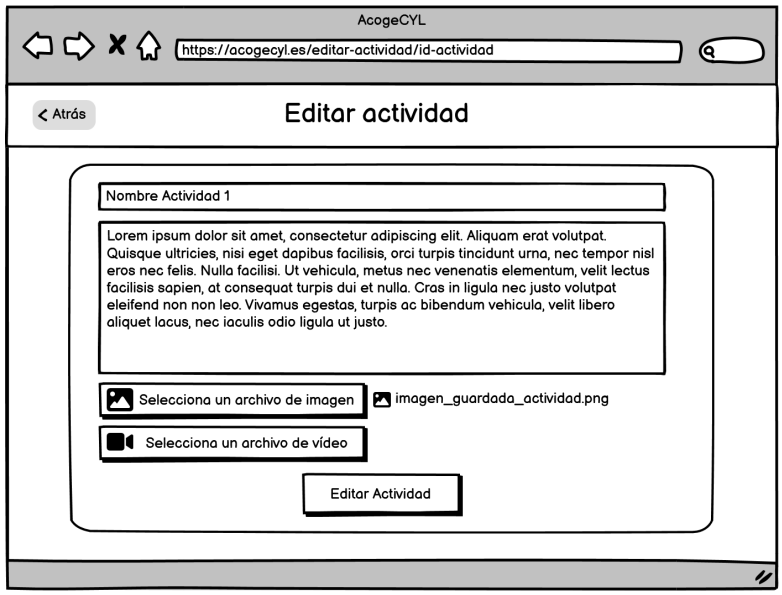

Figura 6.23: Editar una actividad.

Para UC09. Eliminar una actividad, la parte superior de la pantalla es igual que en el *mockup* de los detalles de una Actividad, pero en la parte inferior se añade un botón de *Eliminar actividad*. Se puede ver en la Figura 6.24.

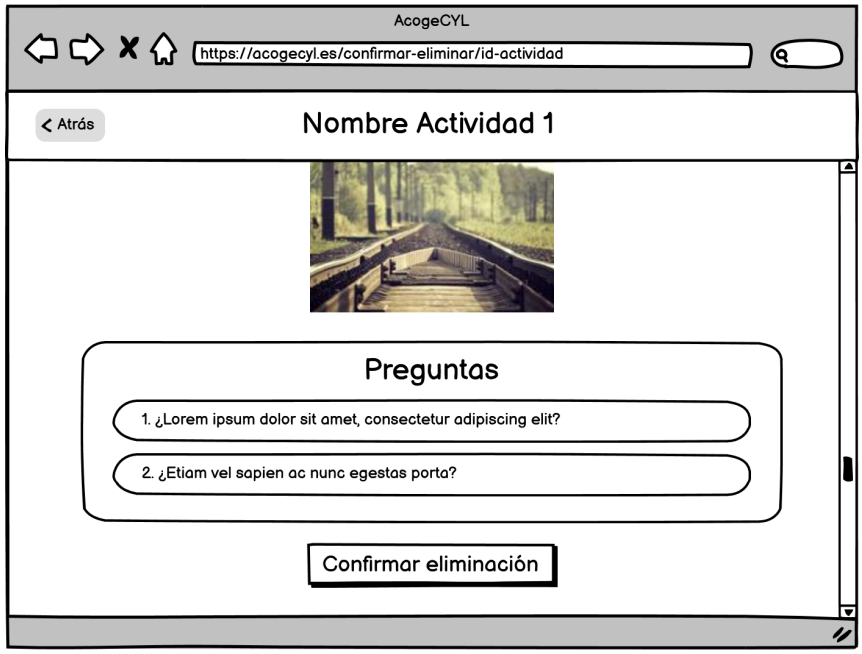

Figura 6.24: Eliminar una Actividad.

El boceto de la pantalla para ver todas las Asignaciones de una Actividad UC10. Ver lista de asignaciones se muestra en la Figura 6.25.

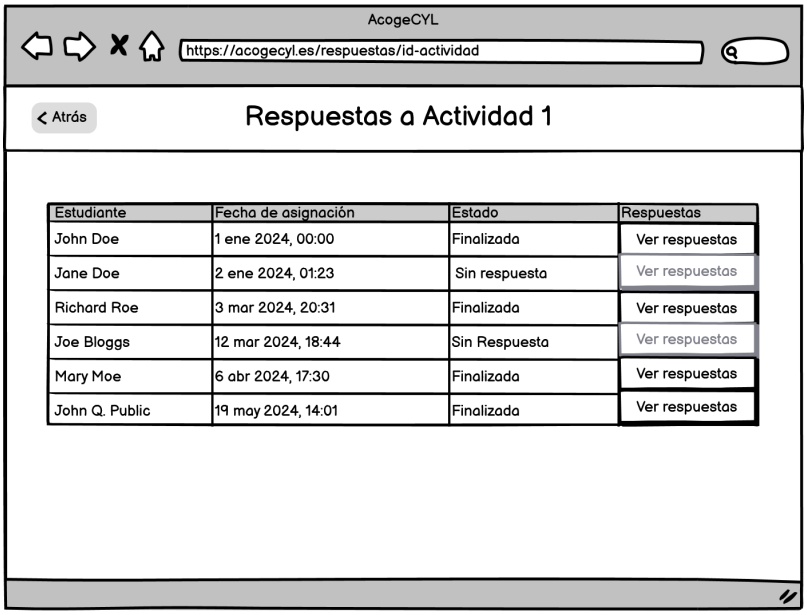

Figura 6.25: Ver lista de Asignaciones de una Actividad.

En caso de que se pulse un botón de *Ver respuestas* de la tabla de la Figura 6.25, se accederá a la pantalla bocetada por la Figura 6.26 del UC11. Ver respuesta de una asignación.

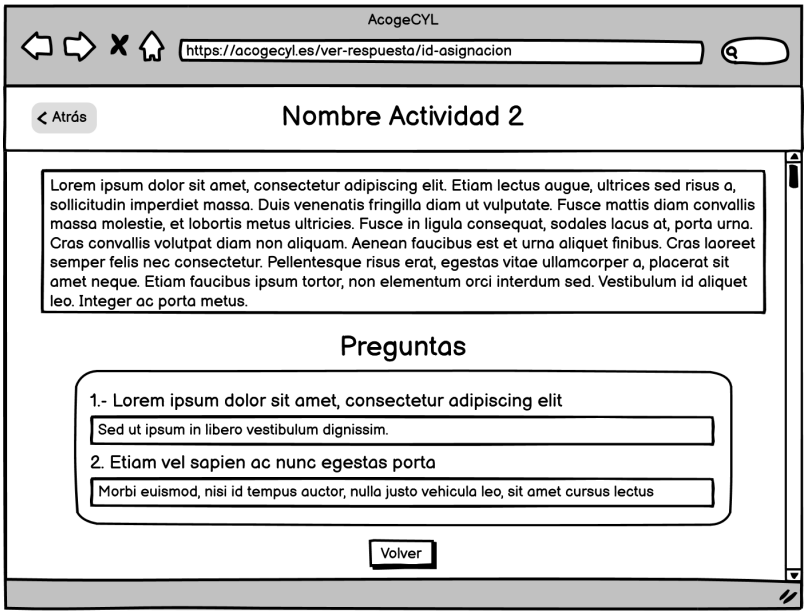

Figura 6.26: Ver la respuesta de un estudiante a una actividad asignada.

Si se inicia sesión como Adminsitrador se mostrará la pantalla bocetada en la Figura 6.27. El Administrador tiene acceso a UC06. Ver detalles de una actividad o UC20. Publicar una actividad.

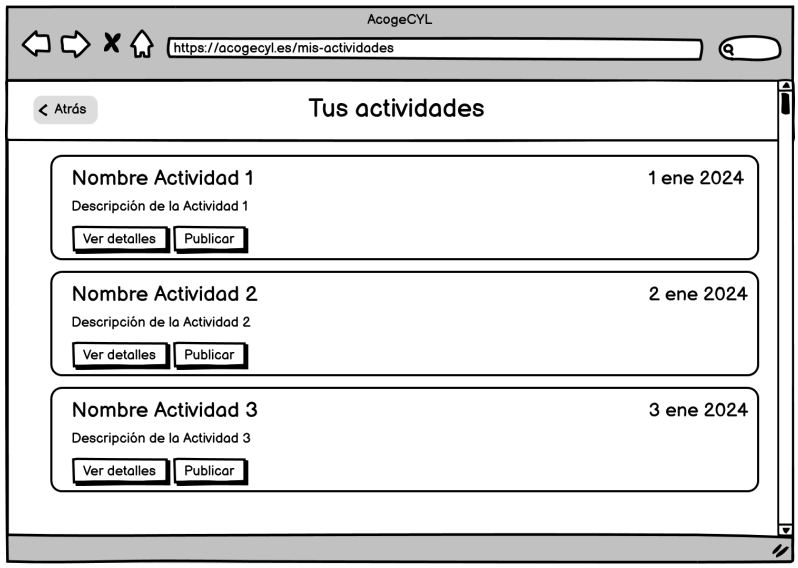

Figura 6.27: Lista de actividades - *Administrador*.

En la Figura 6.28 se presenta el Menú del Diario Emocional del Docente.

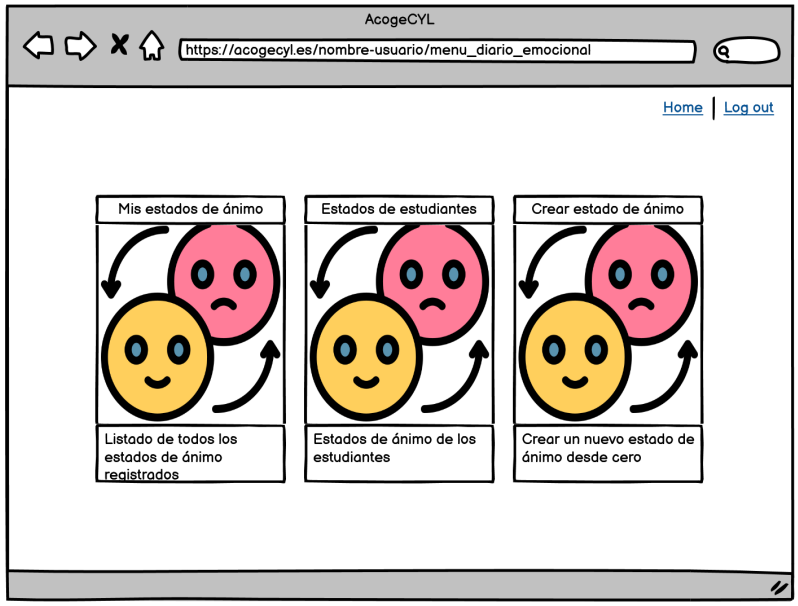

Figura 6.28: Menú Diario Emocional - *Docente*.

En las Figuras 6.29 y las Figuras 6.30 se muestra el *mockup* del listado de Estados de Ánimo del UC13. Ver estados de ánimo del Docente.

Figura 6.29: Mis Estados de Ánimo - *Docente*.

Al posar el cursor del ratón sobre el nombre de la carta de un Estado de Ánimo, cambiará el texto del botón y al pulsarlo se accede a la pantalla bocetada en la Figura 6.31 del UC14. Ver detalles de un estado de ánimo.

Figura 6.30: Mis Estados de Ánimo Hover - *Docente*.

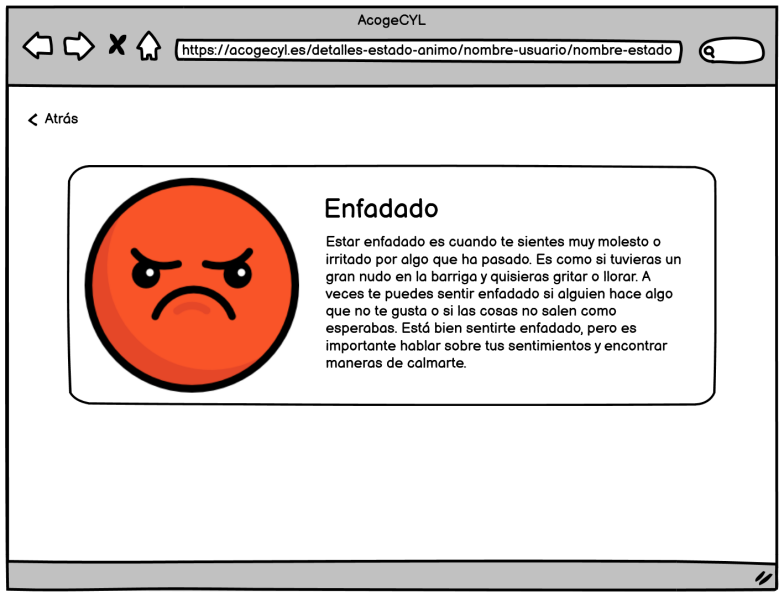

Figura 6.31: Detalles del Estado de Ánimo - *Enfadado*.

Si en el menú bocetado en la Figura 6.28, se accede a *Ver estados estudiantes*, se muestra la pantalla del UC16. Ver estado de ánimo actual de estudiantes cuyo *mockup* se presenta en la Figura 6.32.

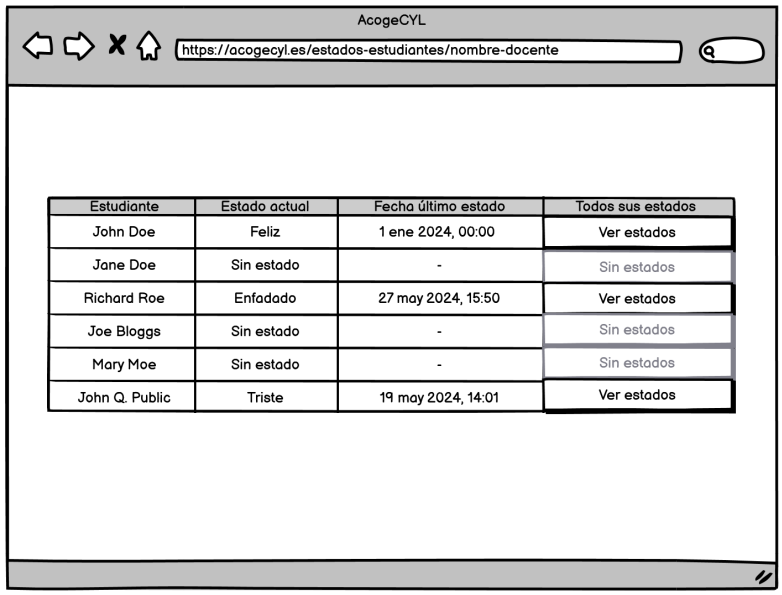

Figura 6.32: Estados de Ánimo actuales de estudiantes.

Si se decide ver el Diario Emocional de un Estudiante (UC17. Ver diario emocional de un estudiante) accediendo a *Ver estados* se mostrará la pantalla del boceto de la Figura 6.33

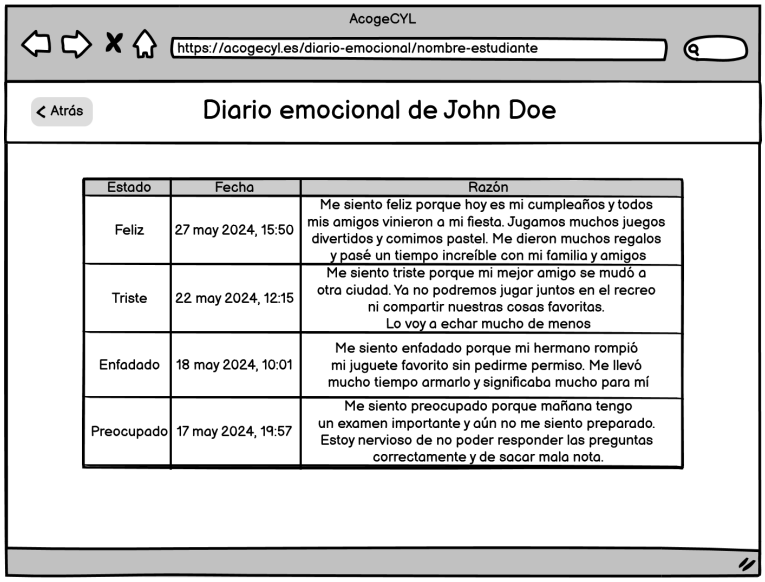

Figura 6.33: Diario emocional de un estudiante.

En la Figura 6.34, se presenta el boceto de la página en la que el Docente puede crear un Estado de Ánimo UC12. Crear un estado de ánimo.

Figura 6.34: Crear un nuevo Estado de Ánimo.

Si se ha iniciado sesión como Estudiante, se mostará el menú bocetado por la Figura 6.35.

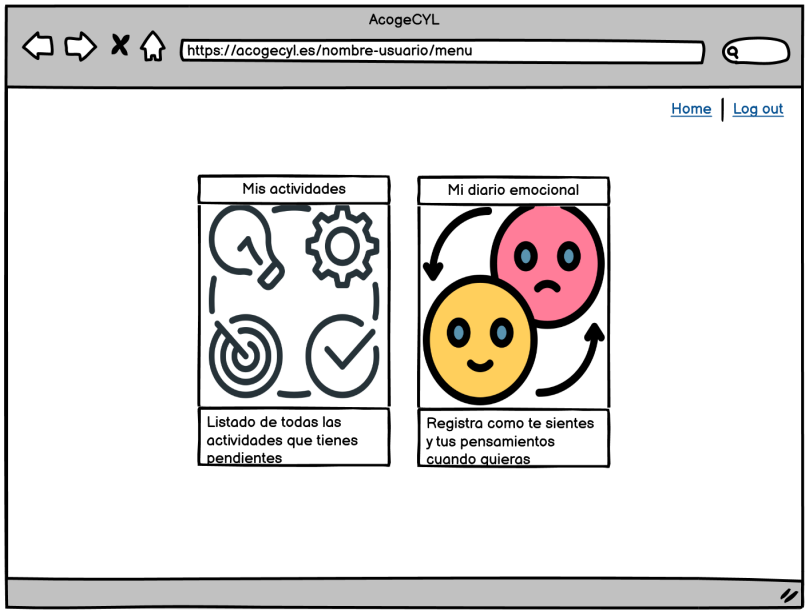

Figura 6.35: Menu del Estudiante.

El boceto de la lista de Actividades del Estudiante se muestra en la Figura 6.36.

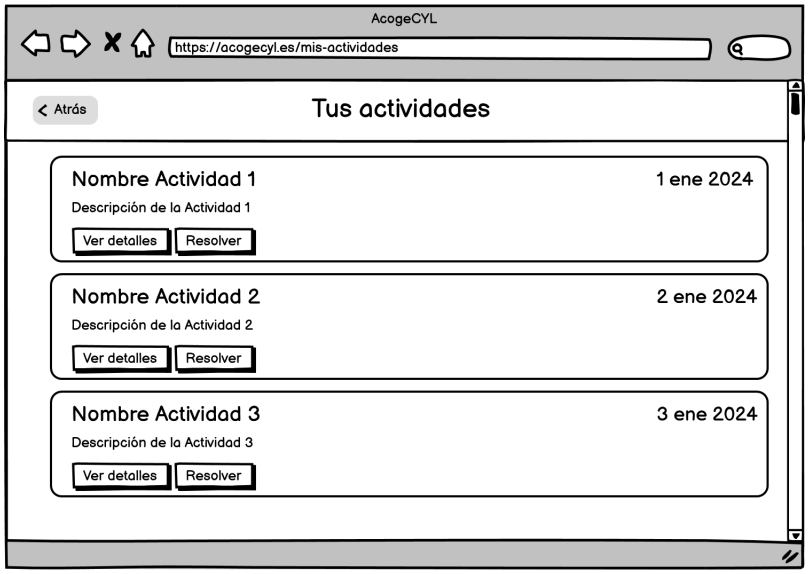

Figura 6.36: Lista de actividades - *Estudiante*.

Si el Estudiante decide iniciar UC19. Resolver una actividad, se mostrará la pantalla bocetada por la Figura 6.37.

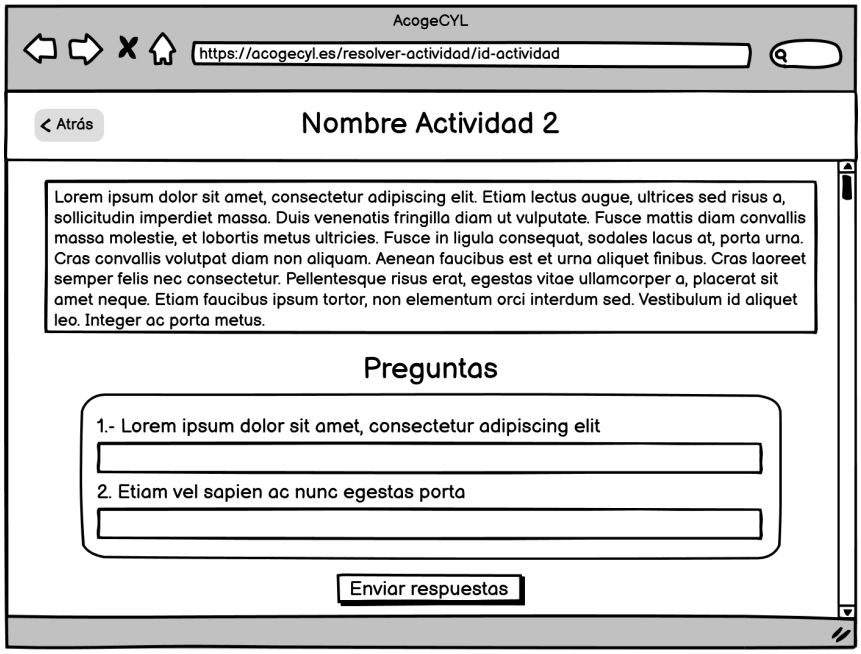

Figura 6.37: Resolver una actividad.

El *mockup* de la Figura 6.38 muestra el listado de Estados de Ánimo. Si el Estudiante decide registrar uno de esos Estados en su Diario Emocional, accederá a la pantalla bocetada por la Figura 6.39.

Figura 6.38: Mis Estados de Ánimo - *Estudiante*.

Figura 6.39: Registrar un Estado de Ánimo - *Estudiante*.

6.5. Diseño relacional de la base de datos

El diseño de las tablas de la base de datos de nuestra aplicación sigue de cerca el modelo de dominio previamente establecido en la Sección 5.2. Cada entidad y algunas asociaciones que han sido identificadas durante la fase de análisis, han sido convertidas en una tabla en la base de datos. Esto asegura que los datos necesarios para las funcionalidades especificadas en las historias de usuario están bien representados y estructurados.

Como se ha mencionado anteriormente, las clases del modelo de dominio forman la base de este diseño de tablas. Cada clase del modelo de análisis corresponde a una tabla de la base de datos.

Además de las tablas derivadas de las clases del modelo de análisis, se ha añadido otra tabla adicional: Administrador. Esta tabla se incluyó debido a cambios en los requisitos del cliente a mitad del desarrollo del proyecto. Esta nueva entidad surgió como resultado de la necesidad de poder publicar actividades para todos los estudiantes registrados en el sistema UC20. Publicar una actividad.

El diseño de las tablas de la base de datos se plasma en la Figura 6.40.

En el diagrama, cada atributo tiene uno o más iconos asociados indicando su tipo. En cuanto a los iconos de la izquierda del nombre: el icono de una llave indica que el atributo es la *primary key* de la tabla, el icono de una tabla con una flecha verde indica que es una *foreign key* y el icono de una tabla indica que es un atributo normal o estándar. En cuanto a los iconos situados a la derecha del nombre del atributo: la *U* roja indica que se trata de un atributo con valores únicos y la *N* blanca indica que puede ser nulo (*nullable*).

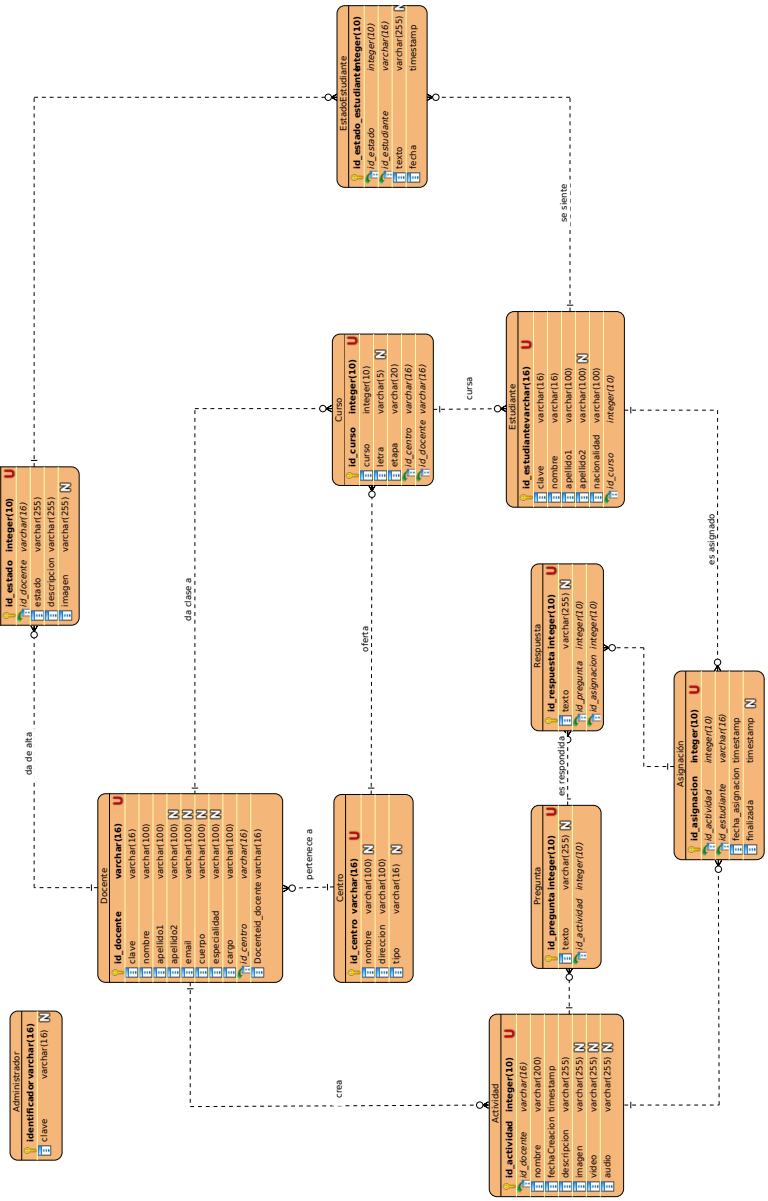

Figura 6.40: Diagrama Entidad Relación

6.6. Realización de caso de uso de diseño.

Para ejemplificar el diseño de comunicación entre las distintas clases de cada una de las capas, se desarrollará el diagrama de secuencia de uno de los casos de uso de los cuales ya se desarrolló en el Documento de Análisis, el caso de uso UC18. Registrar un estado de ánimo. Para no complicar demasiado el diagrama y asegurar su legibilidad, se abreviarán la obtención de varias variables.

Si se desea seguir el flujo de usuario, el Estudiante inicia en la pantalla del *mockup* 6.38. En esta página se muestran todos aquellos Estados de Ánimo registrados en el sistema. El estudiante arranca el caso de uso eligiendo uno de los Estados de Ánimo mostrados. La aplicación entonces, que actúa como *Page Controller*, extrae de la petición HTTP el Estado de Ánimo (*estado*) que el Estudiante quiere registrar y rellena dinámicamente el HTML con sus datos. El Estudiante entonces se sitúa en la página bocetada por el *mockup* 6.39. En esta página el Estudiante indica como se sienta y confirma, enviando su navegador un petición POST. El *Page Controller* recibe la petición, crea el *Transaction Script* y delega la lógica en él. El *Transaction Script* crea el DAO de Estados de Estudiantes, que crea el DTO y la conexión a la base de datos, de la cual obtiene su instancia (al ser este un *Singleton*). A continuación el TS le pasa los datos necesarios en *params* al DTO y le indica al DAO que inserte en la base de datos. El DAO entonces recoge los datos del DTO y ejecuta la inserción. Se cierra la conexión a la base de datos y el caso de uso termina.

6.6.1. UC18. Registrar un Estado de Ánimo

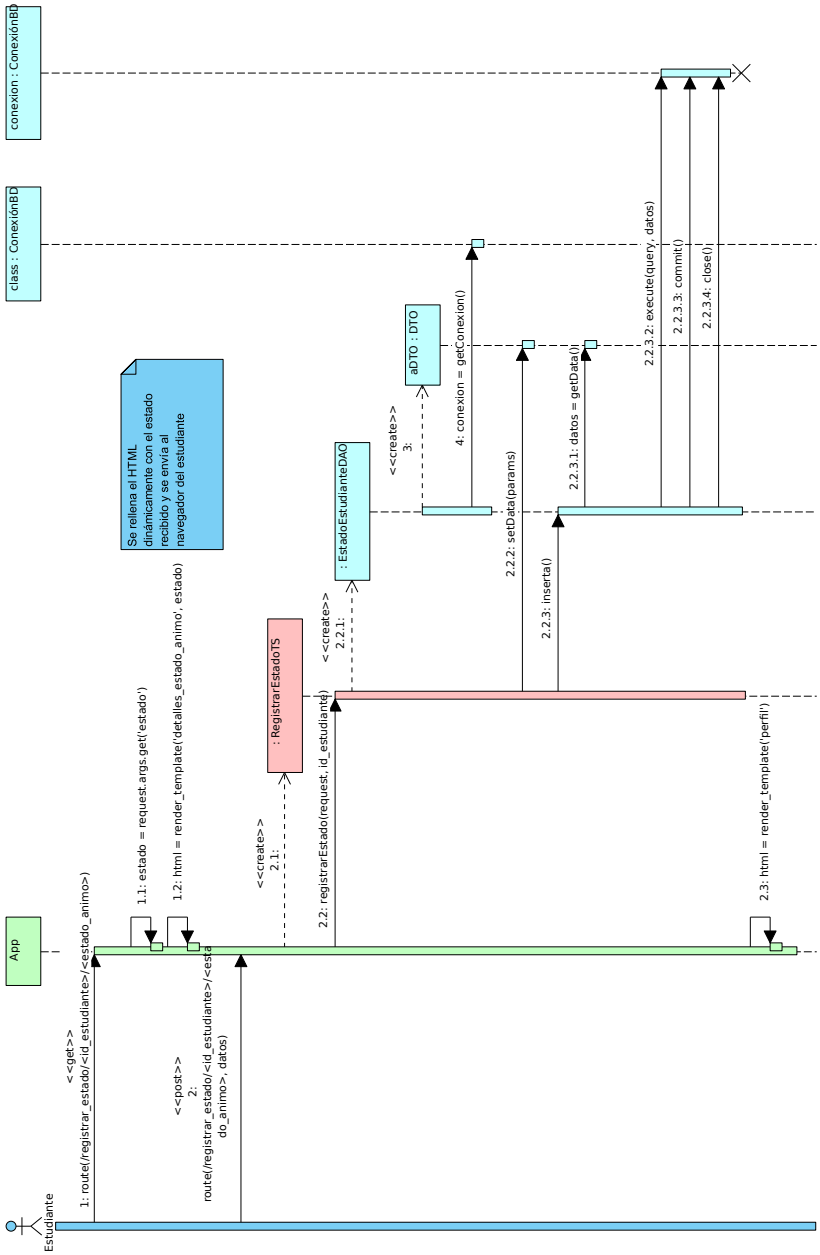

Figura 6.41: Diagrama de secuencia del caso de uso UC18. Registrar un estado de ánimo

Capítulo 7

Implementación

7.1. Introducción

En este capítulo se describen la implementación y las pruebas realizadas a lo largo del proyecto, abarcando el uso de Python y diversos paquetes como Flask. Se detalla la estructura modular del código, la implementación de patrones de diseño como DAO y *Singleton*, y los procedimientos de despliegue del servidor web con Nginx [81] y de la base de datos relacional con MySQL.

7.2. Python

Python [7] es un lenguaje de programación de alto nivel, interpretado y de propósito general, que ha ganado una popularidad significativa debido a su simplicidad, legibilidad y versatilidad. Fue creado por Guido van Rossum y lanzado por primera vez en 1991. Python está diseñado para ser fácil de leer y escribir, con una sintaxis que permite a los desarrolladores expresar conceptos de manera clara y concisa. Este enfoque en la simplicidad y la legibilidad lo convierte en una excelente opción tanto para principiantes como para programadores experimentados. Se ha implementado todo el código Python en la versión 3.10.10 [53].

7.2.1. Paquetes empleados

os

`os` es un módulo de la biblioteca estándar de Python que proporciona una manera de interactuar con el sistema operativo subyacente. Permite a los programas acceder a funcionalidades dependientes del sistema operativo, como la manipulación de archivos y directorios, la gestión de procesos, y la obtención de información del entorno del sistema.

Este paquete se ha usado para poder gestionar el almacenamiento de los archivos multimedia.

Flask

`Flask` es un framework web ligero y flexible para Python que permite a los desarrolladores crear aplicaciones web de manera rápida y sencilla. Desarrollado por Armin Ronacher, Flask sigue el principio de *micro-framework*, lo que significa que proporciona solo las herramientas necesarias para construir una aplicación web básica, dejando la elección y configuración de otras funcionalidades a los desarrolladores según sus necesidades específicas. Esta filosofía hace que Flask sea fácil de aprender y utilizar, especialmente para aquellos que están empezando en el desarrollo web.

La estructura de la capa de servicios viene dictada por el uso de Flask, así como el empleo de los patrones `Template View` y `Page Controller`. Esta influencia se puede observar en el siguiente Fragmento de código:

Fragmento de código 7.1: Función de vista de Crear una Cuenta.

```
@app.route('/crear-cuenta', methods=['GET'])
def signup():
    ts = transaction_scripts.CrearCuentaTS() // Creacion de objeto de dominio
    centros = ts.getCentros() // Recuperacion de datos
    return render_template('signup.html', centros = centros) // Renderizacion de
        pagina
```

En Flask las rutas se gestionan utilizando decoradores en las funciones de vista, en este caso: `@app.route('/crear-cuenta', methods=['GET'])`. Esta función de vista actúa como `Page Controller` cuando se accede a la ruta `/crear-cuenta` mediante una petición `GET`. Se puede observar como delega la lógica en el `Transaction Script` y se encarga de indicar qué vista se va a mostrar. La función `render_template()` sigue el patrón `Template View`, y rellena dinámicamente el HTML '`signup.html`' que funciona como plantilla.

Como se puede observar, `Flask` ofrece una forma sencilla de gestionar las vistas de la aplicación manteniéndolas independientes de las capas inferiores.

babel.dates

El paquete `babel.dates` es una parte del paquete `babel` en Python que proporciona funcionalidades específicas para formatear y manipular fechas y horas de acuerdo con las convenciones locales de diferentes idiomas y regiones. Este módulo se centra en la localización de fechas y horas, permitiendo a los desarrolladores presentar y analizar fechas en el formato preferido por los usuarios finales, así como realizar operaciones comunes con fechas, como cálculos de diferencia de tiempo o extracción de componentes de fecha (día, mes, año, etc.).

El paquete se ha empleado para poder transformar a distintos formatos y mostrar las fechas de creación de Actividades, fecha de finalización de una Asignación etc.

sys

El paquete sys es un módulo de la biblioteca estándar de Python que proporciona acceso a algunas variables y funciones específicas del intérprete de Python y del entorno del sistema. Este módulo es especialmente útil para interactuar con el sistema operativo subyacente, manipular variables de entorno, trabajar con rutas de archivo y acceder a información sobre el intérprete de Python en tiempo de ejecución.

Se emplea en el código junto al paquete `os` para agregar los directorios de las distintas capas al camino de búsqueda de Python en tiempo de ejecución. También se usa para comprobar en qué sistema operativo se está ejecutando la aplicación.

json

El paquete `json` en Python es una parte de la biblioteca estándar que proporciona funciones para trabajar con datos en formato JSON (JavaScript Object Notation). JSON es un formato de intercambio de datos ligero y fácil de leer, ampliamente utilizado en aplicaciones web y servicios web para transmitir datos entre clientes y servidores.

Para minimizar el acoplamiento entre capas y tener flexibilidad y facilidad de deserialización, en la implementación se ha decidido que los DTOs transporten los datos serializados en formato JSON.

bcrypt

El paquete `bcrypt` en Python es una biblioteca que proporciona una forma segura de almacenar contraseñas mediante el algoritmo de hashing bcrypt. Bcrypt es un algoritmo de hashing diseñado específicamente para el *hashing* de contraseñas y está basado en el cifrado *Blowfish*. Fue creado con el objetivo de ser computacionalmente intensivo para desalentar ataques de fuerza bruta.

Fragmento de código 7.2: Ejemplo de uso de `bcrypt`.

```
bcrypt.hashpw(params['password'].encode('utf-8'), bcrypt.gensalt())
bcrypt.checkpw(data['password'].encode('utf-8'),
    dataDocente['clave'].encode('utf-8'))
```

En el Fragmento de código 7.2 se muestran las dos funciones más importantes del paquete: `hashpw()` realiza el *hashing* de la contraseña y `checkpw()` comprueba que la contraseña introducida coincide con la contraseña cifrada con *hash*.

PIL

`PIL` es una biblioteca en Python que proporciona potentes herramientas para la manipulación de imágenes. A pesar de importar `PIL`, realmente la biblioteca que se importa es `Pillow`, una bifurcación de `PIL` que sigue siendo ampliamente utilizada y activamente mantenida.

Fragmento de código 7.3: Ejemplo de uso de `PIL.Image`.

```
image = Image.open(uploaded_file)
```

```
resized_image = image.resize((310,300))
resized_image.save(os.path.join(UPLOAD_FOLDER_IMAGES, params['id_docente'],
    params['imagen']))
```

Como se puede ver en el Fragmento de código 7.3, se ha empleado el módulo *Image* de `PIL` para realizar la redimensión de las imágenes que suben los docentes al crear estados de ánimo.

Unidecode

La principal funcionalidad de unidecode es la transliteración de caracteres Unicode a caracteres ASCII. Esto significa que convierte caracteres como letras acentuadas y caracteres de otros alfabetos (como el cirílico o el chino) a sus equivalentes más cercanos en el alfabeto latino básico. De esta forma se pueden almacenar datos en su formato original y trabajar con ellos después de solo usar `unidecode`.

Fragmento de código 7.4: Ejemplo de uso de `unidecode`.

```
def getListaBusqueda(self, session, busqueda):
    actividades = self.getListaActividades(session)
    if busqueda != "":
        return [act for act in actividades if unidecode(busqueda.lower()) in
            unidecode(act['nombre'].lower())]
    return actividades
```

En el caso de la aplicación, se emplea unidecode para poder realizar la búsqueda de actividades por nombre sin que haya problemas por las tildes. Se muestra su uso en el Fragmento de código 7.4.

uuid

El paquete `uuid` en Python se utiliza para generar UUIDs (*Universally Unique Identifiers*), que son valores de 128 bits que se usan para identificar información de manera única en sistemas distribuidos. Los UUIDs son útiles porque su probabilidad de colisión es extremadamente baja, lo que los hace ideales para identificar entidades de manera única en bases de datos o sistemas de archivos (como es el caso de AcogeCYL), y más.

Para evitar que haya conflictos por los nombres de los archivos, cuando un docente sube cualquier archivo multimedia, se emplea la función `uuid4()` que genera un UUID completamente aleatorio. En la Capa de Utilidad de la aplicación se generan estos nombres mediante la función que se muestra en el Fragmento de código 7.5:

Fragmento de código 7.5: Ejemplo de uso de *uuid*.

```
def generateUniqueFilename(self, filename):
        unique_id = uuid.uuid4().hex
        _, ext = os.path.splitext(filename)
        return unique_id+ext
```

7.2.2. mysql.connector

Para proporcionar una interfaz de Python para poder trabajar con bases de datos MySQL de una manera eficiente, se instala el conector `MySQL Connector/Python` [54]:

Fragmento de código 7.6: Instalación del conector.

```
$ pip install mysql-connector-python
```

Una vez completada la instalación, se puede verificar que el conector `MySQL Connector/Python` está instalado importándolo en un script o intérprete de Python:

Fragmento de código 7.7: Importación del `MySQL Connector`.

```
import mysql.connector
```

Entonces ya se puede obtener una conexión a la base de datos desde Python:

Fragmento de código 7.8: Declaración de la conexión a la BD.

```
conexion = mysql.connector.connect(
           host="localhost", # Direccion del servidor MySQL
           user="integra",  # Nombre de usuario
           password="contrasena", # Contrasena
           database="ACOGECYL" # Nombre de la base de datos
           )
```

Este paquete nos proporciona una interfaz estándar para ejecutar consultas SQL, obtener resultados y manejar transacciones con bases de datos MySQL.

7.3. HTML

HTML [23], o Lenguaje de Marcado de Hipertexto, es el lenguaje de marcado estándar utilizado para crear y diseñar documentos en la Web. Define la estructura y el diseño de las páginas web mediante un sistema de etiquetas y atributos. Los documentos HTML están compuestos por elementos, cada uno de los cuales cumple un propósito específico y puede contener texto, imágenes, enlaces, formularios y otros contenidos multimedia. Los documentos HTML son archivos de texto que constan de una serie de etiquetas HTML encerradas entre corchetes angulares "<" y ">". Estas etiquetas indican el inicio y el final de los elementos, y también pueden incluir atributos para proporcionar información adicional sobre un elemento.

Como en este proyecto se ha empleado `Flask` en conjunción con el patrón *Template View*, los archivos HTML actúan como plantillas que se llenan dinámicamente con datos proporcionados por las funciones de vista de `Flask`. En el patrón *Template View*, estas plantillas contienen marcadores de posición que se llenan con datos específicos durante el proceso de

renderizado (`Flask` usa la función `render_template()`). Para procesar estas plantillas y rellenar los marcadores de posición con los datos proporcionados por las funciones de vista, `Flask` utiliza un motor de plantillas, en este trabajo se ha usado `Jinja2`. Esto permite generar páginas web personalizadas y dinámicas en respuesta a las solicitudes del cliente.

7.3.1. Jinja2

Jinja2 [9] es un motor de plantillas que permite a los desarrolladores separar la lógica de presentación del código de la aplicación. Es similar a otros motores de plantillas como el de Django [10] y es conocido por su simplicidad, flexibilidad y rendimiento. La sintaxis de Jinja2 es similar a Python, lo que facilita a los desarrolladores de Python aprender y usar Jinja2 rápidamente. Para la interpolación de variables se utilizan los dobles corchetes: {{ }} y {% %} se usa para las declaraciones de control de flujo como bucles o condicionales.

En el Fragmento de código 7.9 se muestra un ejemplo de una llamada a la función `render_template()`. Se quiere renderizar la plantilla `historial_estados.html` con los datos proporcionados: `nombre` y `estados`.

Fragmento de código 7.9: Renderización de una plantilla.

```
return render_template('historial_estados.html', nombre = nombre, estados = estados)
```

Se puede ver como `render_template()` proporciona una API pasando datos desde la lógica de la aplicación a las vistas de manera eficiente y sin necesidad de preocuparse por los detalles de la manipulación del HTML o la interacción con el sistema de plantillas.

Fragmento de código 7.10: Ejemplo de uso de Jinja2

```
<div class="column is-8 is-offset-2 has-text-centered">
    <table>
        <thead>
        <tr>
            <th>Estado</th>
            <th>Fecha</th>
            <th>Razon</th>
        </tr>
        </thead>
        <tbody>
        {% for estado in estados %}
            <tr class = "table-text">
                <td>{{ estado['estado'] }}</td>
                <td>{{ estado['fecha'] | spanish_datetime('full') }}</td>
                <td>{{ estado['texto'] }}</td>
            </tr>
        {% endfor %}
        </tbody>
    </table>
</div>
```

En el Fragmento de código 7.10 se muestra como se crea una tabla dinámicamente con tantas filas como elementos tenga la lista `estados[]`. Para ello se itera sobre la lista empleando un bucle {`% for%`} y se rellena cada celda de cada fila con su respectivo atributo. En este caso, se conocen los tres atributos de cada estado de ánimo que se quieren mostrar; sin embargo, en caso de que no se conocieran, se podría construir el número de columnas dinámicamente también.

7.3.2. CSS

CSS (*Cascading Style Sheets*) [11], es un lenguaje de estilo utilizado para definir la presentación y el aspecto visual de documentos HTML y XML. Con CSS, se puede controlar el diseño, la tipografía, los colores, los márgenes, el espaciado y otros aspectos de la apariencia de una página Web. El objetivo principal de CSS es separar el contenido de la presentación, lo que permite una mayor flexibilidad y mantenibilidad del código.

Se ha hecho un uso limitado de CSS por dos razones. Se ha seguido una línea minimalista en el diseño de las páginas de la aplicación para que el diseño sea intuitivo y sencillo para los estudiantes y por el uso de *Bulma CSS*.

7.3.2.1. Bulma CSS

Bulma CSS [17] es un *framework* de diseño de código abierto basado en CSS que facilita la creación de interfaces de usuario (UI) limpias, modernas e interactivas. Utiliza un sistema de columnas flexible basado en *flexbox* para facilitar el diseño y la organización de los elementos en la página. *Bulma* proporciona una variedad de componentes preestilizados, como botones, formularios, tarjetas y barras de navegación, que se pueden utilizar fácilmente para construir interfaces atractivas y funcionales.

Este *framework* toma su nombre del personaje **Bulma** en la serie manga y anime japonés Dragon Ball, cuyo color de pelo coincide con el color del tema original de *Bulma CSS*. El creador de *Bulma CSS*, Jeremy Thomas, eligió este nombre como referencia al personaje, reflejando su simplicidad, flexibilidad y fortaleza.

Fragmento de código 7.11: Ejemplo de empleo de *Bulma CSS*.

```
<button class="button is-block is-info is-large is-fullwidth">Login</button>
```

En el Fragmento de código 7.11 se puede observar cómo personalizar un botón de HTML sin necesidad de recurrir a CSS clásico. Hay una serie de comandos sencillos predefinidos que se pueden introducir dentro del atributo *class*. Se trata de una especie de API que simplifica mucho la personalización de los documentos HTML. En la Figura 7.1 se puede observar el diseño del botón `Login` obtenido gracias a *Bulma CSS*, al igual que el resto de elementos que se pueden observar. El color de fondo verdoso que se puede apreciar es el color de pelo de **Bulma**.

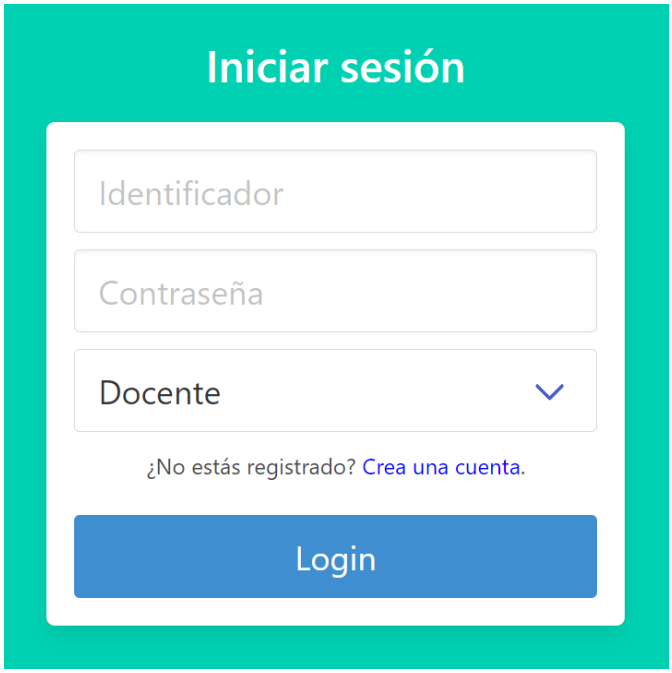

Figura 7.1: Inicio de sesión de la aplicación.

7.3.3. JavaScript

JavaScript [12] es un lenguaje de programación de alto nivel, dinámico e interpretado, que se utiliza principalmente en el desarrollo web. Es conocido por su capacidad de manipular el Document Object Model (DOM) de las páginas web, lo que permite crear experiencias de usuario interactivas y dinámicas. Se ha empleado para poder incorporar características dinámicas a las plantillas HTML como añadir tantas preguntas como se quiera y para controlar eventos e incluir interactividad. Al ser funciones muy sencillas, se han incorporado dentro de los propios ficheros HTML mediante la marca <script>.

Fragmento de código 7.12: Ejemplo de empleo de JavaScript.

```
<script>
    function autoresize(textarea) {
        textarea.style.height = 'auto';
        textarea.style.height = textarea.scrollHeight + 'px';
    }

    window.addEventListener('DOMContentLoaded', (event) => {
        var textareas = document.querySelectorAll('textarea');
        textareas.forEach(autoresize);
    });
</script>
```

En el Fragmento de código 7.12 se muestran dos funciones distintas implementadas en la plantilla `detalles_actividad.html`. La función `autoresize(textarea)` sirve para modificar el tamaño del `textarea` en función del tamaño del texto que contiene. Quién se encarga de llamar a esta función es el controlador de eventos que figura debajo. `DOMContentLoaded` es el tipo de evento que el controlador está escuchando. Este evento se dispara cuando el documento HTML inicial ha sido completamente cargado y analizado, sin esperar a que las hojas de estilo, imágenes y subtramas terminen de cargarse.

7.4. Estructura del código

7.4.1. Capa de Servicios

En la Capa de Servicios de la aplicación, se ha adoptado una estructura modular que se centra en un único archivo Python. Este archivo sirve un `Page Controller` donde se gestionan todas las rutas y funciones de vista de la misma. Estas funciones de vista, que son controladores en el contexto de Flask, procesan las solicitudes del cliente y devuelven respuestas HTML generadas dinámicamente.

Dentro de este archivo, debido a la estructura inherente de Flask, se han organizado las rutas utilizando un enfoque basado en funciones, donde cada ruta está asociada a una función de vista específica. Esto proporciona una estructura clara y concisa para gestionar las solicitudes de los clientes y manejar la lógica de negocio correspondiente.

La decisión de mantener todas las rutas y funciones de vista en un solo archivo, se ajusta a las convenciones de Flask, al igual que su nombre: `routes.py`. Esta estructura facilita el mantenimiento y la extensibilidad de la aplicación. Además, permite tener una visión completa y centrada en un solo lugar de todas las funcionalidades disponibles en la aplicación, lo que facilita la colaboración y la comprensión del código. Previamente en el Fragmento de código 7.1 se muestra un ejemplo de función de vista.

7.4.2. Capa de Dominio

Dentro de la capa de dominio, se ha adoptado una estructura que se centra en un único archivo Python denominado `transaction_scripts.py`. Este archivo actúa como un repositorio centralizado que alberga todos los `Transaction Scripts` necesarios para la gestión de las operaciones del negocio.

Cada `Transaction Script` en este archivo encapsula varias operaciones específicas del dominio, implementando la lógica necesaria para llevar a cabo esa operación de manera coherente. Cada *Script* corresponde a un caso de uso y abarca las tareas necesarias para llevarla a cabo. En el Fragmento de código 7.13 se puede ver la implementación completa del `Transaction Script` del caso de uso UC04. Ver lista de actividades.

Fragmento de código 7.13: Implementacion de un *Transaction Script*

```
class DetallesActividadTS:

    def getActividadById(self, activity_id):
        uTS = UsuarioTS()
        return uTS.getActividadById(activity_id)

    def getPreguntas(self, activity_id):
        uTS = UsuarioTS()
        return uTS.getPreguntasActividad(activity_id)

    def getDatosActividad(self, activity_id):
        return self.getActividadById(activity_id), self.getPreguntas(activity_id)
```

`UsuarioTS` encapsula toda la funcionalidad compartida por varios `Transaction Scripts` distintos con el fin de evitar código repetido.

7.4.3. Capa de Acceso a Datos

La capa de acceso a datos es responsable de interactuar con la base de datos y gestionar la persistencia de los datos de la aplicación. Se ha estructurado esta capa en dos paquetes principales: `connection` y `daos`.

En el paquete connection, se encuentra la lógica relacionada con la gestión de la conexión al servidor *MySQL*. Este paquete contiene un único archivo Python llamado *connection.py*, en el cual se define una clase `singleton` que se encarga de establecer y mantener la conexión con el servidor *MySQL*.

En el paquete daos, se encuentran todos los Data Access Objects (DAOs) de la aplicación. Estos objetos son responsables de encapsular la lógica necesaria para interactuar con la base de datos y realizar operaciones de lectura y escritura sobre los datos.

En este paquete, se ha optado por organizar todos los DAOs en un único archivo Python llamado `daos.py`. En este archivo, cada DAO está definido como una clase separada, con métodos específicos para realizar consultas y actualizaciones en la base de datos relacionadas con una tabla específica. Todos los DAOs heredan de un DAO genérico.

En el Fragmento de código 7.14 se puede observar un fragmento de la implementación de `PreguntaDAO`, el `DAO` encargado de la tabla Pregunta:

Fragmento de código 7.14: Fragmento de la implementación de un DAO

```
class PreguntaDAO(DAO):
    def __init__(self):
        super().__init__()

    def inserta(self):
        if(self.currentDto != None):
            data = self.currentDto.getData() // Se extraen los datos del DTO
            cursor = self.conexionInstancia.obtener_cursor() // Se obtiene la
                instancia de la conexion a la BD
            consulta = """
            INSERT INTO pregunta (texto, id_actividad)
            VALUES (%s, %s)
            """
            datos = (data['pregunta'], data['id_actividad'])
            cursor.execute(consulta, datos) // Se ejecuta la consulta
            self.conexion.commit()
            cursor.close()

    def eliminaByActividad(self):
        if(self.currentDto != None):
            id_actividad = self.currentDto.getData() // Se extrae el id de la
                actividad del DTO
            cursor = self.conexionInstancia.obtener_cursor()
            cursor.execute(f"DELETE FROM pregunta WHERE id_actividad =
                '{id_actividad}'") // Se ejecuta la consulta indicada
            self.conexion.commit()
            cursor.close()
```

Se han incluido dos funciones distintas del `DAO`:

1. `inserta()`: Inserta una nueva Pregunta en la base de datos dados su texto y el identificador de la Actividad a la que pertenece.
2. `eliminaByActividad()`: Elimina todas aquellas instancias de la tabla Pregunta cuyo identificador de Actividad coincida con el recibido.

Al igual que los *Transaction Scripts* les pasan los datos necesarios para la consulta a un DTO, de donde los DAOs extraen dichos datos. En caso de que la consulta devuelva resultados, estos se serializan en formato JSON y los DAOs se los pasan al DTO, de donde los *Transaction Scripts* podrán extraer los resultados en dicho formato.

7.4.4. Capa de Utilidad

La Capa de Utilidad de nuestra aplicación está diseñada para proporcionar funciones y herramientas auxiliares que son utilizadas por diversas partes del sistema. Esta capa contiene

un único archivo Python denominado utils.py, en el cual se encapsulan varias funciones que son empleadas por el resto de las capas de la aplicación. Estas funciones están diseñadas para realizar tareas comunes y repetitivas, lo que facilita la reutilización del código y mejora la mantenibilidad del sistema.

7.5. Despliegue del servidor web

El despliegue de la aplicación web se realiza utilizando Nginx [81], un servidor web de alto rendimiento que también puede funcionar como un proxy inverso, balanceador de carga, y caché HTTP. Nginx es conocido por su alta eficiencia, capacidad de manejar múltiples conexiones simultáneas y su baja huella de memoria, lo que lo hace ideal para servir aplicaciones web modernas y cumplir con los requisitos no funcionales que se quieren satisfacer.

La aplicación web AcogeCYL está alojada en un servidor FDS (*Flask Development Server*) que se encuentra en una máquina Ubuntu [24]. Como no es recomendable que un FDS atienda a peticiones directamente, se ha instalado un servidor Nginx que actúa como *proxy*. La configuración de Nginx para nuestra aplicación incluye:

1. Se instala Nginx:

```
$ sudo apt install nginx
```

2. Se accede al directorio `/etc/nginx/site-available` y se crea el archivo `application.conf`.

```
$ cd /etc/nginx/site-available
$ vi application.conf
```

3. Dentro de `application.conf` se define un bloque server que especifica la dirección IP y el puerto en el que Nginx escucha las solicitudes.

```
server {
    server_name valderaduey.infor.uva.es;
    client_max_body_size 20M; # Establece el l mite de
       ↪ tama o de solicitud a 20 megabytes
    location / {
        proxy_pass http://127.0.0.1:5000;  # Direccion y el
           ↪  puerto de Flask
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For
           ↪ $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    listen 443 ssl;
```

```
    ssl_certificate /etc/letsencrypt/live/valderaduey.infor
        ↪ .uva.es/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/valderaduey.
        ↪ infor.uva.es/privkey.pem
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}

server {
    if ($host = valderaduey.infor.uva.es) {
        return 301 https://$host$request_uri;
    }
    listen 80;
    server_name valderaduey.infor.uva.es;
    return 404;
}
```

Se puede ver que la máquina escucha peticiones HTTPS a través del puerto 443, estas peticiones se redirigen al servidor de `Flask` en local, que está escuchando en el puerto 5000. En caso de que la petición llegue al puerto 80, que es es HTTP, se convierte en HTTPS y se devuelve el código de error 404 a la petición original. El tamaño máximo establecido de una petición es de 20 MB (*megabytes*), aunque se puede aumentar en caso de que se quiera admitir la subida de archivos pesados, como en el caso de un docente adjuntando un vídeo a una de sus actividades.

4. Se crea un enlace dinámico para simplificar la gestión y el despliegue de la configuración del servidor web. En el contexto de Nginx, el enlace dinámico se refiere específicamente a la creación de un enlace simbólico desde el directorio de configuración activa (sites-enabled) hasta el directorio donde se almacenan las configuraciones disponibles (sites-available). Esto permite que la configuración del servidor web se actualice automáticamente cuando se agregan, eliminan o modifican archivos de configuración en el directorio sites-available.

```
    $ sudo ln -s /etc/nginx/sites-available/application.
        ↪ conf /etc/nginx/sites-enabled
```

5. Se comprueba que todo está correcto y se reinicia el servicio.

```
    $ sudo nginx -t
    $ sudo systemctl restart nginx
```

6. Se actualiza la lista de paquetes disponibles en los repositorios configurados en el sistema y se instala *certbot* con la extensión de Nginx para dar servicio HTTPS.

```
    $ sudo apt update
    $ sudo apt-get install python3-certbot-nginx
```

7. Obtenemos el certificado para proporcionar comunicaciones seguras (SSL/TLS) entre los clientes y el servidor Nginx:

```
$ sudo certbot --nginx -d valderaduey.infor.uva.es
```

8. Entonces ya se tiene el servidor desplegado con los detalles que se incluyeron en `etc/nginx/sites-enabled/application.conf`

9. Para renovar automáticamente el certificado, se puede utilizar el comando:

```
$ sudo certbot renew --dry-run
```

   Es importante ejecutar regularmente certbot renew para garantizar que tus certificados SSL/TLS se mantengan actualizados y no expiren accidentalmente. El uso de `--dry-run` permite probar este proceso de renovación de manera segura antes de realizarlo en producción.

7.6. Despliegue de la base de datos

En esta sección, se aborda el proceso de despliegue de una base de datos MySQL, junto con el conector MySQL para Python, en un entorno de desarrollo o producción. El despliegue efectivo de la base de datos es crucial para garantizar un funcionamiento eficiente de las aplicaciones que dependen de ella.

Todos los datos de la aplicación son almacenados en una base de datos alojada en una máquina Ubuntu. Se procedió con el despliegue de la base de datos mediante los siguientes pasos:

1. Se actualizan los paquetes y se instala MySQL:

```
$ sudo apt update
$ sudo apt install mysql-server
```

2. Se verifica el estado del servicio MySQL para asegurar que esté en funcionamiento correctamente:

```
$ sudo systemctl status mysql
```

3. Se ejecuta el *script* `mysql_secure_installation` que guía a través de una serie de configuraciones de seguridad iniciales como la configuración de la contraseña de root, la eliminación de cuentas de usuario anónimas, la desactivación del inicio de sesión remoto para el usuario root y la eliminación de la base de datos de pruebas, entre otras cosas.

```
$ sudo mysql_secure_installation
```

4. A continuación se accede al *shell* de MySQL:

```
$ sudo mysql -u root -p
```

   Donde se tendrá que ingresar la contraseña de *root* que se configuró durante la instalación.

5. Se crea la base de datos y el usuario que se usará a partir de ahora. Se le dan todos los privilegios al usuario:

```
mysql> CREATE DATABASE ACOGECYL;
mysql> CREATE USER 'integra'@'localhost' IDENTIFIED BY
   ↪ 'contrasena';
mysql> GRANT ALL PRIVILEGES ON ACOGECYL.* TO 'integra'@
   ↪ 'localhost';
mysql> FLUSH PRIVILEGES;
```

6. Se permite al firewall UFW (*Uncomplicated Firewall*) de Linux permitir el tráfico entrante para el servicio de MySQL.

```
$ sudo ufw allow mysql
```

7. Una vez se tiene la base de datos desplegada, solo queda acceder al usuario creado y ejecutar los archivos ***sql*** nominados según la convención de MySQL [55].

```
$ mysql -u integra -p contrasena
    mysql> source crear_tablas.sql
    mysql> source insertar_datos.sql
```

El archivo `crear.sql` contiene todas las sentencias necesarias para crear las tablas de la base de datos siguiendo las directrices de nomenclatura de MySQL [55]. En caso de que las tablas ya existan, su ejecución no tiene efecto. El archivo `insertar_datos.sql` inserta en las tablas de la BD las instancias que se han empleado para realizar pruebas a lo largo del desarrollo del proyecto.

Capítulo 8

Pruebas

8.1. Introducción

Una prueba de software es un proceso sistemático y controlado que tiene como objetivo evaluar la calidad y el funcionamiento de un software. Consiste en ejecutar el software bajo condiciones específicas para identificar errores, fallos, defectos o comportamientos inesperados, con el fin de corregirlos y mejorar la calidad del producto final [82].

Existen distintos niveles de pruebas con distintos objetivos [83]:

- **Pruebas unitarias:** Nivel más bajo de pruebas y se centra en probar componentes individuales o unidades de código de forma aislada. Las pruebas unitarias verifican que cada unidad del software funcione según lo esperado de acuerdo con su diseño.
- **Pruebas de integración:** Implican probar la interacción entre diferentes unidades o módulos del software para garantizar que funcionen juntos según lo previsto. Las pruebas de integración verifican que los componentes integrados se comuniquen correctamente y produzcan los resultados esperados cuando se combinan.
- **Pruebas de sistema:** Verifican que todo el sistema de software se comporte según lo esperado cuando todos los componentes están integrados e interactúan entre sí. Se prueban el sistema en su conjunto contra sus requisitos y especificaciones.
- **Pruebas de aceptación:** Se realizan con los usuarios con el objetivo de validar de que el *software* responde realmente a sus necesidades.

En este Capítulo solo se ha documentado un caso de prueba de sistema para cada caso de uso. Estos casos de pruebas se realizaron una vez finalizado el desarrollo de la aplicación. Sin embargo, según avanzaba el proyecto se realizaron continuas pruebas unitarias y pruebas de sistema después de la implementación de cada caso de uso.

En ingeniería de software, un caso de prueba es una especificación de las entradas, las condiciones de ejecución, el procedimiento de prueba y los resultados esperados que definen una prueba única que debe ejecutarse para lograr un objetivo de prueba de *software* concreto, como seguir una ruta de programa determinada o verificar el cumplimiento de un requisito específico [84]. En la Sección 8.3 el objetivo de cada caso de prueba es seguir el escenario principal de cada caso de uso y verificar su correcto funcionamiento.

8.2. Pruebas unitarias

A lo largo del desarrollo del proyecto, después de la implementación de cada función, fuera una función de vista, de un DAO o de un *Transaction Script*, el estudiante realizaba pruebas unitarias probando todas las entradas posibles y comparando las salidas obtenidas con las esperadas. Estas pruebas no han sido inlcuidas en el documento por la gran cantidad de funciones implementadas y el alto número de pruebas realizadas.

8.3. Pruebas de sistema

Se ha realizado una prueba de sistema para cada caso de uso con el fin de validar el funcionamiento del *software* en su conjunto, garantizando que cumple con los requisitos y especificaciones definidos en cada caso de uso. Estas pruebas también ayudarán a identificar posibles problemas de integración o errores de diseño.

8.3.1. Caso de prueba TC01

Identificador	TC01
Precondiciones	El usuario ha elegido que quiere registrarse como Docente y no tiene una cuenta creada con el identificador que ha elegido.
Datos de Entrada	Nombre, apellidos, un identificador, contraseña, cuerpo del profesorado, especialidad, cargo, centro en el que imparte clases y su *email*.
Resultado Esperado	Se ha registrado un nuevo docente.
Escenario	Se presiona el botón ***Crear cuenta*** en la barra de navegación. Se escoge la tarjeta ***Docente*** para registrar un Docente. Se introducen los datos solicitados y se hace *click* en ***Darme de alta***.
Trazabilidad	Prueba de sistema del caso de uso UC01. Registro de docente
Ejecución	2024-06-07
Defectos encontrados	Ninguno

Tabla 8.1: Tabla del caso de prueba TC01.

8.3.2. Caso de prueba TC02

Identificador	TC02
Precondiciones	El usuario ha elegido que quiere registrarse como Estudiante y no tiene una cuenta creada con el identificador que ha elegido.
Datos de Entrada	Identificador, contraseña, apodo, nacionalidad e identificador del curso.
Resultado Esperado	Se ha registrado un nuevo estudiante.
Escenario	Se presiona el botón ***Crear cuenta*** en la barra de navegación. Se escoge la tarjeta ***Estudiante*** para registrar un Estudiante. Se introducen los datos solicitados y se hace *click* en ***Darme de alta***.
Trazabilidad	Prueba de sistema del caso de uso UC02. Registro de estudiante
Ejecución	2024-06-07
Defectos encontrados	Ninguno

Tabla 8.2: Tabla del caso de prueba TC02.

8.3.3. Caso de prueba TC03

Identificador	TC03
Precondiciones	El usuario debe encontrarse en la página de iniciar sesión y debe tener una cuenta creada.
Datos de Entrada	Identificador de usuario y contraseña válidos.
Resultado Esperado	Se ha iniciado sesión exitosamente.
Escenario	Se introducen el usuario y contraseña válidos. Se hace *click* en ***Login***.
Trazabilidad	Prueba de sistema del caso de uso UC03. Iniciar sesión
Ejecución	2024-06-07
Defectos encontrados	Ninguno

Tabla 8.3: Tabla del caso de prueba TC03.

8.3.4. Caso de prueba TC04

Identificador	TC04
Precondiciones	El usuario debe haber iniciado sesión.
Datos de Entrada	Ninguno.
Resultado Esperado	Se muestra la lista de actividades.
Escenario	Se selecciona la tarjeta ***Mis actividades***.
Trazabilidad	Prueba de sistema del caso de uso UC04. Ver lista de actividades
Ejecución	2024-06-07
Defectos encontrados	Ninguno.

Tabla 8.4: Tabla del caso de prueba TC04.

8.3.5. Caso de prueba TC05

Identificador	TC05
Precondiciones	El Docente debe haber iniciado sesión y debe encontrarse en su menú de Actividades.
Datos de Entrada	Nombre de la Actividad, descripción de la Actividad, recursos multimedia y Preguntas.
Resultado Esperado	Se ha creado una nueva Actividad.
Escenario	El Docente selecciona la tarjeta Nueva actividad. Se introduce el nombre y se introduce la descripción en sus áreas de texto correspondientes. Se adjuntan los recursos multimedia y se presiona en ***Aceptar***. Se selecciona que el botón ***Sí*** para añadir Preguntas. Se selecciona el botón ***Añadir pregunta*** y se introduce una Pregunta. Se hace *click* en ***Guardar preguntas***.
Trazabilidad	Prueba de sistema del caso de uso UC05. Crear una actividad
Ejecución	2024-06-07
Defectos encontrados	El contador de las preguntas añadidas no contabiliza correctamente.

Tabla 8.5: Tabla del caso de prueba TC05.

8.3.6. Caso de prueba TC06

Identificador	TC06
Precondiciones	El usuario debe de haber iniciado sesión y situarse en la lista de Actividades. Debe haber al menos una Actividad registrada en el Sistema.
Datos de Entrada	Ninguno.
Resultado Esperado	Se muestran los detalles de la Actividad.
Escenario	El usuario selecciona ***Detalles*** en la tarjeta de la Actividad cuyos detalles se quieren ver.
Trazabilidad	Prueba de sistema del caso de uso UC06. Ver detalles de una actividad
Ejecución	2024-06-07
Defectos encontrados	Ninguno.

Tabla 8.6: Tabla del caso de prueba TC06.

8.3.7. Caso de prueba TC07

Identificador	TC07
Precondiciones	El Docente debe haber iniciado sesión,
Datos de Entrada	Descripción de la Actividad, recursos multimedia y Preguntas.
Resultado Esperado	Se ha creado una nueva actividad editando la actividad seleccionada.
Escenario	Se selecciona el botón ***Editar*** de la tarjeta de la Actividad a editar. Se modifica la descripción, se adjuntan los recursos multimedia y se presiona en *Aceptar*. Se selecciona el botón ***Sí*** para añadir preguntas. Se selecciona el botón ***Añadir pregunta*** y se introduce una nueva pregunta a mayores de la que ya había. Se hace *click* en ***Guardar preguntas***.
Trazabilidad	Prueba de sistema del caso de uso UC07. Editar una actividad
Ejecución	2024-06-07
Defectos encontrados	La interfaz no muestra los archivos que ya están adjuntos.

Tabla 8.7: Tabla del caso de prueba TC07.

8.3.8. Caso de prueba TC08

Identificador	TC08
Precondiciones	El Docente debe haber iniciado sesión, debe situarse en la lista de actividades, debe haber creado al menos una Actividad y debe impartir clase a al menos un Estudiante.
Datos de Entrada	Ninguno.
Resultado Esperado	Se ha asignado la actividad a los estudiantes seleccionados.
Escenario	Se presiona el botón ***Asignar a estudiantes*** de la tarjeta de la Actividad a asignar. Se selecciona el cuadro de selección de aquellos Estudiantes a los que se les quiera asignar la Actividad. Se hace *click* en ***Asignar*** y se confirma que se quiere realizar la acción.
Trazabilidad	Prueba de sistema del caso de uso UC08. Asignar una actividad
Ejecución	2024-06-07
Defectos encontrados	Ninguno.

Tabla 8.8: Tabla del caso de prueba TC08.

8.3.9. Caso de prueba TC09

Identificador	TC09
Precondiciones	El Docente debe haber iniciado sesión, debe situarse en la lista de actividades y debe haber creado al menos una Actividad.
Datos de Entrada	Ninguno.
Resultado Esperado	Se ha eliminado la Actividad.
Escenario	Se presiona el botón ***Eliminar*** de la tarjeta de la Actividad a eliminar. Se presiona el botón ***Confirmar eliminación*** y se confirma que se quiere realizar la acción.
Trazabilidad	Prueba de sistema del caso de uso UC09. Eliminar una actividad
Ejecución	2024-06-07
Defectos encontrados	Ninguno.

Tabla 8.9: Tabla del caso de prueba TC09.

8.3.10. Caso de prueba TC10

Identificador	TC10
Precondiciones	El Docente debe haber iniciado sesión, debe situarse en la lista de actividades y debe haber creado al menos una Actividad.
Datos de Entrada	Ninguno.
Resultado Esperado	Se muestra la lista de respuestas a la actividad.
Escenario	Se presiona el botón ***Ver respuestas*** de la tarjeta de la Actividad cuyas respuestas se quieren ver.
Trazabilidad	Prueba de sistema del caso de uso UC10. Ver lista de asignaciones
Ejecución	2024-06-07
Defectos encontrados	Ninguno.

Tabla 8.10: Tabla del caso de prueba TC10.

8.3.11. Caso de prueba TC11

Identificador	TC11
Precondiciones	El Docente debe haber iniciado sesión, debe situarse en la lista de respuestas de la Actividad, debe haber creado al menos una Actividad, debe haber asignado la Actividad a un Estudiante al que imparta clase y el Estudiante debe de haber finalizado la Asignación.
Datos de Entrada	Ninguno.
Resultado Esperado	Se muestra la Respuesta de la Asignación.
Escenario	Se presiona el botón ***Ver respuestas*** de la fila de la Asignación cuya Respuesta se quieren ver.
Trazabilidad	Prueba de sistema del caso de uso UC11. Ver respuesta de una asignación
Ejecución	2024-06-07
Defectos encontrados	Ninguno.

Tabla 8.11: Tabla del caso de prueba TC11.

8.3.12. Caso de prueba TC12

Identificador	TC12
Precondiciones	El docente debe haber iniciado sesión y debe encontrarse en su menú de estados de ánimo.
Datos de Entrada	Nombre, descripción e imagen.
Resultado Esperado	Se ha creado un nuevo Estado de Ánimo.
Escenario	El usuario selecciona la tarjeta Crear un Estado de Ánimo. Se introduce el nombre, se introduce la descripción, se adjuntan la imagen y se presiona en ***Aceptar***. Se confirma que se quiere crear el Estado de Ánimo.
Trazabilidad	Prueba de sistema del caso de uso UC12. Crear un estado de ánimo
Ejecución	2024-06-07
Defectos encontrados	Ninguno.

Tabla 8.12: Tabla del caso de prueba TC12.

8.3.13. Caso de prueba TC13

Identificador	TC13
Precondiciones	El Usuario debe haber iniciado sesión.
Datos de Entrada	Ninguno.
Resultado Esperado	Se muestran todos los Estados de Ánimo registrados.
Escenario	Se selecciona la tarjeta ***Mis stados de ánimo***.
Trazabilidad	Prueba de sistema del caso de uso UC13. Ver estados de ánimo
Ejecución	2024-06-07
Defectos encontrados	Ninguno.

Tabla 8.13: Tabla del caso de prueba TC13.

8.3.14. Caso de prueba TC14

Identificador	TC14
Precondiciones	El Docente debe haber iniciado sesión, debe encontrarse en el listado de Estados de Ánimo y debe haber al menos un Estado de Ánimo registrado en el Sistema.
Datos de Entrada	Ninguno.
Resultado Esperado	Se muestran los detalles del Estado de Ánimo.
Escenario	Se presiona el botón con el nombre del Estado de Ánimo de la tarjeta del Estado de Ánimo.
Trazabilidad	Prueba de sistema del caso de uso UC14. Ver detalles de un estado de ánimo
Ejecución	2024-06-07
Defectos encontrados	Ninguno.

Tabla 8.14: Tabla del caso de prueba TC14.

8.3.15. Caso de prueba TC15

Identificador	TC15
Precondiciones	El Docente debe haber iniciado sesión, debe encontrarse en el listado de Estados de Ánimo y debe haber al menos un Estado de Ánimo registrado en el Sistema.
Datos de Entrada	Ninguno.
Resultado Esperado	Se ha eliminado el Estado de Ánimo.
Escenario	Se presiona el botón ***Eliminar*** de la tarjeta del Estado de Ánimo. Se confirma la acción.
Trazabilidad	Prueba de sistema del caso de uso UC15. Eliminar un estado de ánimo
Ejecución	2024-06-07
Defectos encontrados	Ninguno

Tabla 8.15: Tabla del caso de prueba TC15.

8.3.16. Caso de prueba TC16

Identificador	TC16
Precondiciones	El Docente debe haber iniciado sesión y debe encontrarse en su menú de Estados de Ánimo.
Datos de Entrada	Ninguno.
Resultado Esperado	Se muestra el Estado de Ánimo actual de los Estudiantes a los que imparte clase el Docente.
Escenario	Se selecciona la tarjeta ***Estados de Estudiantes***.
Trazabilidad	Prueba de sistema del caso de uso UC16. Ver estado de ánimo actual de estudiantes
Ejecución	2024-06-07
Defectos encontrados	Ninguno.

Tabla 8.16: Tabla del caso de prueba TC16.

8.3.17. Caso de prueba TC17

Identificador	TC17
Precondiciones	El Docente debe haber iniciado sesión, debe encontrarse en el listado de Estados de Estudiantes, debe impartir clase a al menos un Estudiante, debe haber al menos un Estado de Ánimo registrado en el Sistema y el Estudiante debe de haber registrado al menos un Estado de Estudiante.
Datos de Entrada	Ninguno.
Resultado Esperado	Se muestra el diario emocional del estudiante.
Escenario	Se presiona el botón ***Ver estados*** de la fila de la Asignación cuya Respuesta se quieren ver.
Trazabilidad	Prueba de sistema del caso de uso UC17. Ver diario emocional de un estudiante
Ejecución	2024-06-07
Defectos encontrados	Ninguno.

Tabla 8.17: Tabla del caso de prueba TC17.

8.3.18. Caso de prueba TC18

Identificador	TC18
Precondiciones	El Estudiante debe haber iniciado sesión y debe haber al menos un Estado de Ánimo registrado en el Sistema.
Datos de Entrada	Razón por la que se registra el Estado de Estudiante.
Resultado Esperado	Se registra un nuevo Estado de Estudiante asociado al Estado de Ánimo y el Estudiante.
Escenario	Se selecciona la tarjeta ***Mi diario emocional***. Se escribe la razón por la que se quiere registrar el Estado, se acepta, y se confirma la acción.
Trazabilidad	Prueba de sistema del caso de uso UC18. Registrar un estado de ánimo
Ejecución	2024-06-07
Defectos encontrados	Ninguno.

Tabla 8.18: Tabla del caso de prueba TC18.

8.3.19. Caso de prueba TC19

Identificador	TC19
Precondiciones	El Estudiante debe haber iniciado sesión, debe encontrarse en la lista de actividades y debe tener al menos una Asignación de una Actividad.
Datos de Entrada	Respuestas a las Preguntas de la Actividad.
Resultado Esperado	La Asignación de la Actividad queda finalizada.
Escenario	El Estudiante presiona el botón ***Resolver*** de la tarjeta de la Actividad. En caso de que la Actividad tenga preguntas, el Estudiante introduce las Respuestas a las Preguntas. Se presiona el botón ***Resolver actividad*** y se confirma que quiere realizar la acción.
Trazabilidad	Prueba de sistema del caso de uso UC19. Resolver una actividad
Ejecución	2024-06-07
Defectos encontrados	Ninguno.

Tabla 8.19: Tabla del caso de prueba TC19.

8.3.20. Caso de prueba TC20

Identificador	TC20
Precondiciones	El Administrador debe haber iniciado sesión y debe haber al menos una Actividad registrada en el Sistema.
Datos de Entrada	Ninguno.
Resultado Esperado	La Actividad queda asignada a todos los Estudiantes.
Escenario	El Administrador presiona el botón ***Publicar*** de la tarjeta de la Actividad a publicar y confirma la acción.
Trazabilidad	Prueba de sistema del caso de uso UC20. Publicar una actividad
Ejecución	2024-06-07
Defectos encontrados	No se pide confirmación antes de publicar la actividad.

Tabla 8.20: Tabla del caso de prueba TC20.

8.4. Pruebas de aceptación

Las pruebas de aceptación conforman una fase crucial en el ciclo de vida del desarrollo de software, ya que valida que el sistema cumple con los requisitos establecidos y está listo para ser desplegado en un entorno de producción. Esta prueba asegura que todas las funcionalidades implementadas funcionan correctamente y que el software es apto para su uso por parte del cliente o usuario final [48].

En el contexto de este proyecto, se llevó a cabo una prueba de aceptación el día 10 de abril de 2024. En esta fecha, el tutor del Trabajo de Fin de Grado y el estudiante se reunieron con una profesora del departamento de Psicología Evolutiva y de la Educación para realizar la prueba, una de las *stakeholders* del proyecto.

Detalles de la prueba

1. **Presentación del *Software:*** El estudiante presentó una descripción general del sistema, destacando las principales funcionalidades del software.
2. **Demostración en vivo:** Se realizó una demostración en vivo del software, mostrando cómo se utilizan las distintas características y cómo estas se alinean con los requisitos iniciales del proyecto.
3. **Ejercicios prácticos:** La profesora participó en una serie de ejercicios prácticos diseñados para probar diferentes aspectos del software. Estos ejercicios se enfocaron en simular escenarios de uso real para evaluar la respuesta del sistema bajo condiciones similares a las del entorno operativo.
4. **Evaluación y *Feedback:*** Tras la demostración y los ejercicios prácticos, la profesora proporcionó *feedback* detallado sobre la funcionalidad, usabilidad y rendimiento del software. Se discutieron posibles mejoras y ajustes necesarios para optimizar el sistema.

Resultados de la prueba

La prueba de aceptación concluyó de manera exitosa. La profesora destacó varios puntos fuertes del software, incluyendo:

1. **Facilidad de uso y atractivo visual:** El software fue considerado intuitivo, fácil de usar y visualmente atractivo, lo que es crucial para su adopción en entornos educativos.
2. **Funcionalidad completa**: Todas las funcionalidades principales fueron probadas y funcionaron correctamente, cumpliendo con los requisitos establecidos.
3. **Rendimiento:** El sistema mostró un buen rendimiento durante la prueba, respondiendo de manera ágil a las interacciones del usuario.

Además, se identificaron algunas mejoras que fueron documentadas y se planificaron para su implementación en futuras iteraciones del desarrollo como:

- Incluir un usuario de tipo administrador que sea capaz de asignar una actividad a todos los estudiantes asignados.
- Los docentes deben de ser capaz de ver las respuestas de los estudiantes a todas las asignaciones respondidas, incluso a aquellas asignaciones cuyas actividades no han sido creadas por ellos.
- Incluir *feedback* gráficamente cuando se ha realizado una acción.
- Los estados de ánimo deben ser globales, es decir, un estudiante o docente debe ser capaz de ver cualquier estado de ánimo, independientemente de quién lo haya creado.

Capítulo 9

Seguimiento del proyecto

9.1. Introducción

Este capítulo ofrece una visión detallada del progreso del proyecto a lo largo de su desarrollo, dividido en *sprints*. En este análisis exhaustivo, se documentan las tareas completadas en cada *sprint*, proporcionando una perspectiva cronológica y detallada de la evolución del proyecto. Cada sprint representa una etapa significativa en el ciclo de desarrollo, donde se abordan diversas tareas y se alcanzan hitos clave. Este seguimiento permite comprender la dinámica y organización del trabajo, los desafíos enfrentados y los logros obtenidos en cada fase del proyecto.

En este contexto, cada tarea abordada en el seguimiento del proyecto puede clasificarse de dos formas. Por un lado, una tarea puede estar directamente relacionada con una historia de usuario, lo que implica que contribuye directamente a la implementación de una funcionalidad específica solicitada por el usuario final. Estas tareas puede corresponder a una fase del proceso de desarrollo de software asociada a una historia de usuario, como la planificación, el diseño, la implementación o las pruebas. Estas tareas son fundamentales para avanzar en el ciclo de vida de desarrollo de la funcionalidad en cuestión. Las tareas también se pueden clasificar como aquellas que no están directamente ligadas a historias de usuario, pero que son igualmente importantes para el progreso general del proyecto, como la configuración del entorno de desarrollo, la corrección de errores o la investigación sobre una tecnología.

Para cada tarea abordada en el seguimiento del proyecto, se llevará a cabo una documentación que incluirá varios aspectos clave:

1. Nombre de la tarea.
2. Tiempo estimado para completarla.
3. Tiempo empleado para completarla.

4. Estado de la tarea.

Esta información proporcionará una visión clara del progreso real de la tarea en comparación con el planificado. Además, se mantendrá actualizado el estado de la tarea, indicando si está en progreso, completada o si ha surgido algún problema que requiera atención adicional.

En el caso de que varias tareas estén relacionadas bajo una misma historia de usuario, se optará por documentarlas de manera conjunta. Esto permitirá tener una visión integrada de los esfuerzos dedicados a implementar esa funcionalidad específica, facilitando así la comprensión del progreso global de la historia de usuario en cuestión.

9.2. Seguimiento de los *Sprints*

9.2.1. Sprint 0 (13/02/24 - 21/02/24)

El martes 12 de febrero de 2024, el estudiante se reunió con el tutor para obtener contexto e información sobre la idea del trabajo a realizar. En esta reunión se dió a conocer al estudiante la idea del proyecto, su contexto y su motivación. También se le presentaron las tecnologías que se habían planeado emplear.

Después de que el estudiante se informara y adquiriera conocimientos básicos sobre la materia, el viernes 16 de febrero de 2024 se realizó la *kickoff meeting* o reunión de arranque para dar inicio al Sprint 0 del desarrollo de la aplicación. Durante esta reunión de arranque, se establecieron los objetivos principales, los requisitos del Departamento de Psicología Evolutiva y de la Educación, se definieron roles y se delinearon los próximos pasos. Esta reunión marca el punto de partida del proyecto.

Dada la falta de experiencia del estudiante en el ámbito de las aplicaciones web, se priorizó la dedicación de la mayor parte del tiempo disponible a la lectura de documentación especializada y al análisis de ejemplos prácticos de implementaciones previas encontradas en Internet. Aún así, se pudieron llevar a cabo varias tareas:

- **Búsqueda de información inicial:** Se dedicó gran parte del *Sprint* a la búsqueda de información y al aprendizaje de uso de tecnologías como *Flask* o *Bulma CSS*.
- ***Product Backlog* inicial.** Se identificaron las necesidades básicas requeridas por el departamento de Psicología Evolutiva y de la Educación, de las cuales se extrajeron los requisitos generales que se formularon como épicas en el *Product Backlog*.
- **Obtención de las historias de usuario:** Una vez obtenidas las épicas, estas se desgranaron para obtener las historias de usuario y así obtener también los requisitos funcionales.
- ***Scrum*:** Se decidió que el proyecto se planificaría mediante el marco de trabajo *Scrum* debido al corto tiempo de desarrollo y la necesidad de implementar software rápidamente.

- **Modelo de dominio inicial:** El estudiante realizó una primera aproximación al modelo de dominio.
- **Arquitectura lógica:** Se decidió conjuntamente emplear una arquitectura lógica de tres capas con presentación, dominio y datos.
- **Tablero de Trello:** Se creó el tablero de *Trello* y se empezaron a planificar tareas a realizar.

En resumen, el Sprint 0 fue el primer paso importante hacia el desarrollo del proyecto del estudiante. Estableció las bases necesarias para avanzar en el proyecto. En total se emplearon 18 horas.

9.2.2. Sprint 1 (21/02/24 - 01/03/24)

En este primer *Sprint* se empezó a desarrollar el software. Durante este y el siguiente *sprint*, la cantidad de horas empleadas (14 horas) fue inferior a la planificada (en torno a 20 horas) pues el TFG del grado en Estadística requirió de numerosas horas en estas semanas.

Los tareas que se realizaron en el *sprint* 1 están detalladas en la Tabla 9.1.

Historia de usuario	Tareas	Tiempo estimado	Tiempo empleado	Estado
US01	Lectura de documentación y ejemplos.	3 horas	2 horas	Finalizada
US02	Lectura de documentación y ejemplos.	2 horas	2 horas	Finalizada
-	Búsqueda de información y puesta a punto de software.	8 horas	10 horas	Finalizada

Tabla 9.1: Desglose de tareas del *sprint* 1.

Mientras se recogía información y el estudiante adquiría conocimiento sobre el uso de las tecnologías, se fueron realizando varias demos para conseguir experiencia con Flask. Fue necesario consultar el funcionamiento de las rutas de Flask [75] y del archivo `__init__.py` [76] de cada aplicación. Se visualizó un vídeo de YouTube [102] para aprender a manejar los conceptos más básicos de la construcción de aplicaciones web con esta herramienta.

A pesar de que se realizó una demo de creación de cuenta, que correspondería con la historia de usuario **US01**, no se realizó ningún tipo de análisis o diseño sobre esta por lo que no se considera como un avance en el desarrollo de dicha historia de usuario. Se querría haber empezado el desarrollo de las historias de usuario US01 y US02, pero no fue posible por falta de tiempo.

El estudiante necesitó investigar sobre LaTeX debido a su poca experiencia con esta herramienta de preparación de documentos.La falta de familiaridad con el entorno y la sintaxis de LaTeX lo llevó a consultar la documentación oficial en el sitio web del Proyecto LaTeX [5]. La búsqueda se centró sobre todo en la creación de Tablas [60] [61] [65], Tablas largas [68], Listings [74] y gráficos SVG [73].

9.2.3. Sprint 2 (01/03/24 - 08/03/24)

Como se ha mencionado en el *sprint* anterior, en este *sprint* tampoco se pudo invertir las 20 horas programadas por causas acedémicas.

En el *sprint* anterior ya se sentaron las bases de trabajar con todas las tecnologías y se obtuvo una idea clara de como empezar las iteraciones de desarrollo. En la Tabla 9.2 se muestran los avances que se consiguieron este *sprint*.

Historia de usuario	Tareas	Tiempo estimado	Tiempo empleado	Estado
-	Creación de una demo de la base de datos	4 horas	2 horas 30 minutos	Finalizada
US01	▪ Obtención de los Casos de Uso equivalentes ▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	3 horas	2 horas 30 minutos	Finalizada
US02	▪ Obtención del Caso de Uso equivalente ▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	2 horas 30 minutos	2 horas	Finalizada
US03	▪ Obtención del Caso de Uso equivalente ▪ Análisis	4 horas	2 horas	En progreso

Tabla 9.2: Desglose de tareas del *sprint* 2.

Por la gran similitud de las historias de usuario en las que se trata el inicio de sesión y la creación de una cuenta, se hizo el análisis y el diseño de estas prácticamente en paralelo. El

tutor proporcionó un punto de partida con una *applet* que tenía incorporada el registro de una cuenta por lo que el tiempo empleado en estas tareas fue menor del esperado.

Se encontraron un par de inconvenientes con el *hashing* de las contraseñas pero se encontraron soluciones en *Stack Overflow* [97] [98] [100] [99].

También se necesitó revisar el funcionamiento del tipo de datos enumerados `ENUM` en SQL pues el estudiante no estaba familiarizado con su uso [106].

En la etapa de diseño de **US01** y **US02** surgió una duda respecto a la redimensión dinámica de los cuadros de texto en HTML, también se pudo encontrar una solución en *Stack Overflow* [101]. También se tuvo que consultar la documentación de *Balsamiq Wireframes* para entender cómo emplear los *data grids* [72] al diseñar los *mockups*.

Por último, el último día del *Sprint* se pudo realizar el análisis del caso de uso **US03**, consultando la documentación de tratamiento de fechas de *Python* [107] y de subida de archivos de *Flask* [108].

9.2.4. Sprint 3 (08/03/24 - 15/03/24)

En este *sprint* se pudieron invertir las horas requeridas según la planificación, pero no se pudo invertir más horas para compensar la falta de horas invertidas hasta el momento.

Historia de usuario	Tareas	Tiempo estimado	Tiempo empleado	Estado
US03	▪ Diseño ▪ Implementación ▪ Pruebas	6 horas	6 horas	En progreso
US04 y US19	▪ Obtención del Caso de Uso equivalente ▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	6 horas	4 horas 30 minutos	Finalizada
US05 y US20	▪ Análisis ▪ Diseño ▪ Implementación	6 horas	10 horas	En progreso

Tabla 9.3: Desglose de tareas del *sprint* 3.

En total se han invertido 20 horas y 30 minutos. El desglose de las tareas realizadas en el *sprint* 3 se puede ver en la Tabla 9.3.

En la etapa de diseño de la historia de usuario **US05** surgieron varios problemas, aunque uno de ellos fue el que acaparó la mayor parte del tiempo. Este problema se trataba de poder descargar las imágenes o vídeos que el docente subía al sistema de archivos de la aplicación. Se consultó la documentación de *Flask* sobre los ficheros estáticos y dos dudas parecidas que ya habían sido respondidas en *DevPress* [110] y en *Stack Overflow* [111]. Sin embargo, este problema no se pudo resolver hasta el siguiente *sprint*. La funcionalidad requerida para **US05** y **US20** es la misma por lo que se desarrollan conjuntamente. Se aplica exactamente lo mismo a **US04** y **US19**.

A pesar de que el contenido multimedia no se mostrara, se pudo completar una iteración completa sobre la historia de usuario **US04**. Solo faltaba que las imágenes y vídeos se mostraran para los detalles de las actividades se mostraran correctamente.

9.2.5. Sprint 4 (15/03/24 - 22/03/24)

En este *sprint* se aumentó la carga de trabajo para recuperar gran parte del tiempo perdido hasta el momento. En la Tabla 9.4 se reflejan las tareas realizadas este *sprint*.

El primer día del *sprint* se pudo resolver el problema de gestión de los archivos multimedia del caso de uso **US04** [77]. Se tuvo que realizar una reestimación del tiempo invertido en esta tarea pues ya había sobrepasado el tiempo estimado con creces.

Prácticamente todo el trabajo de este *sprint* giró en torno a la edición de las actividades y al diseño de un menú con navegación intuitiva. Se añadió la fecha de creación de las actividades en el menú, fue necesario consultar [112].

Por otra parte, se añadió la posibilidad de incluir un audio en las Actividades. Se estimó que se necesitaría de solo una hora para realizarlo pero al final se necesitó de 3 horas pues *Bulma CSS* [17] tiene incompatibilidad con los audios por culpa de su uso por defecto de *Flexbox*, se pudo encontrar una solución a este problema en *Stack Overflow* [113].

Historia de usuario	Tareas	Tiempo estimado	Tiempo empleado	Estado
-	Adición de audios a las Actividades	1 hora	3 horas	Finalizada
US05 y US20	▪ Implementación ▪ Pruebas	2 horas	2 horas	Finalizada
US06	▪ Obtención del Caso de Uso equivalente ▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	10 horas	12 horas	Finalizada
US07	▪ Obtención del Caso de Uso equivalente ▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	2 horas	2 horas	Finalizada
-	Análisis, diseño e implementación del menú de Estudiantes y Docentes y su navegación	8 horas	7 horas	Finalizada

Tabla 9.4: Desglose de tareas del *sprint* 4.

9.2.6. Sprint 5 (22/03/24 - 29/03/24)

En este *sprint* también se superó el número de horas invertidas en el proyecto, aunque fuera muy ligeramente. De las 20 horas previstas, se invirtieron 21 horas. En el *sprint* anterior no quedó ninguna tarea pendiente, por lo que se empezará el desarrollo de nuevas historias de usuario. A partir de ahora puede que se realicen pequeñas iteraciones sobre ciertas historias de usuario en caso de que se quiera modificar algo, ya sea por requisitos cambiantes o con tal de mejorar lo ya implementado.

En principio no se había prevista emplear tiempo de este *sprint* para la mejora del menú, pero ya se había empezado a hablar con el Departamento de Psicología Evolutiva y de la Educación para hacer una reunión y realizar una prueba de aceptación, por lo que se creyó necesario mejorar la interfaz. Para ello se realizaron varios retoques, se reorganizó el menú [120] [119] y se añadieron iconos de *FontAwesome* [18]. Hubo un pequeño contratiempo por un conflicto de versiones de *FontAwesome* pero se pudo resolver gracias a una duda resuelta en *Stack Overflow* [118].

En la tabla 9.5 se muestra el trabajo realizado en el *sprint* 5.

Historia de usuario	Tareas	Tiempo estimado	Tiempo empleado	Estado
-	Mejora de la interfaz del menú	-	3 horas	Finalizada
US09	▪ Obtención del Caso de Uso equivalente ▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	6 horas	8 horas	Finalizada
US10	▪ Obtención del Caso de Uso equivalente ▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	4 horas	3 horas	Finalizada
US12	▪ Obtención del Caso de Uso equivalente ▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	2 horas	2 horas	Finalizada
US13 y US22	▪ Obtención del Caso de Uso equivalente ▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	3 horas	3 horas	Finalizada
US14	▪ Obtención del Caso de Uso equivalente ▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	2 horas	2 horas	Finalizada

Tabla 9.5: Desglose de tareas del *sprint* 5.

Para la creación de la página de asignaciones de actividad a estudiante, la historia de usuario **US09**, fue necesario recurrir a *checkboxes*, por lo que se buscó ejemplos de implementaciones [114]. Al principio, la recogida de información de las *checkboxes* no funcionaba correctamente, problema que se pudo resolver gracias una duda también resuelta en *Stack Overflow* [115].

Por último, para la creación de estados de ánimo, fue necesario añadir una forma de cambiar de tamaño todas las imágenes que el docente adjuntaba para mantener la estética y uniformidad del listado de estados de ánimo. Para ello se consultó la documentación de *PIL* [124] y como hacer *resize* en *Flask* [121] [122]. Finalmente se decidió emplear *PIL*, aunque hubo un problema inesperado ya que no se detectaba su instalación en el *Virtual Enviroment* a pesar de que sí lo estaba, se pudo encontrar una solución en *Stack Overflow* [123].

9.2.7. Sprint 6 (29/03/24 - 05/04/24)

Durante el desarrollo del *sprint* actual, a pesar de que coincidía con un período de vacaciones, se tomó la decisión de dedicar tiempo adicional al proyecto para compensar las horas que no se habían invertido plenamente en los sprints iniciales. Esta elección estratégica se hizo con tal de mantener el impulso y cumplir con los objetivos establecidos. Teniendo en mente que en este mes el estudiante tendría que seguir avanzando el TFG del Grado en Estadística y realizar varias entregas de las prácticas de una asignatura. En total, el estudiante trabajó 26 horas y 30 minutos en el proyecto este *sprint*, se pueden ver las tareas realizadas en la tabla 9.6.

En este *sprint* se empezó a implementar la serialización JSON como objeto que los DTOs transportarán, esta decisión se hizo en el *sprint* 3 pero se decidió no implementar hasta más adelante para no complicar la lógica por el momento. Su implementación tomó más tiempo del esperado por incompatibilidades del formato JSON con los instantes de tiempo *datetime* y los *boolean* de Python. Se pudo resolver el problema de las fechas modificándolas [126] y de los *boolean* gracias a una solución encontrada en Internet [125].

Posteriormente, cuando se estaba implementando la eliminación de estados de ánimo, funcionalidad requerida por la historia de usuario **US15**, se produjo un *overflow* de las letras de los botones [127] [128], aún así no se invirtió más tiempo del esperado.

Para evitar la colisión de imágenes con el mismo nombre en el sistema de archivos, se añadió una lógica de renombrado de archivos empleando el paquete `uuid` [92] después de haber consultado soluciones distintas en internet usando fechas [117] o la estructura de archivos [116].

Historia de usuario	Tareas	Tiempo estimado	Tiempo empleado	Estado
-	Introducción del uso de JSON con los DTOs	2 horas	5 horas	Finalizada
-	Reestructuración del menú	-	2 hora 30 minutos	Finalizada
-	Esquema y búsqueda de información para el documento final del TFG	-	1 hora	Finalizada
US15	▪ Obtención del Caso de Uso equivalente ▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	2 horas	2 horas	Finalizada
US16 y US17	▪ Obtención del Caso de Uso equivalente ▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	4 horas	4 horas	Finalizada
US23 y US24	▪ Obtención del Caso de Uso equivalente ▪ Análisis ▪ Diseño ▪ Implementación	4 horas	5 horas	Finalizada
-	Modificación de la lógica y nomenclatura para imágenes	-	3 horas	Finalizada
-	Pruebas de sistema y arreglos de toda la funcionalidad implementada hasta el momento	1 hora	4 horas	Finalizada

Tabla 9.6: Desglose de tareas del *sprint* 6.

En la etapa de análisis de las historias de usuario **US16** y **US17**, se decidió que las dos funcionalidades requeridas: ver la lista de alumnos y ver sus estados de ánimo, se implementarían juntas. Es decir, se crearía un listado de todos los estudiantes del docente donde también se incluiría su estado de ánimo actual. En la etapa de diseño, se decidió emplear tablas para el listado de alumnos, por lo que se necesitó documentación sobre ellas [72] [130].

En cuanto a las historias de usuario **US23** y **US24**, se decidió implementarlas conjuntamente, es decir, el estudiante es capaz de ver los detalles del estado de ánimo sobre el que quiere registrar un estado. No se tuvo tiempo para realizar pruebas sobre el software implementado.

9.2.8. Sprint 7 (05/04/24 - 12/04/24)

Durante este *sprint* se concretó la fecha de la reunión con el Departamento de Psicología Evolutiva y de la Educación para presentar una versión avanzada de la aplicación y realizar una prueba de aceptación. A pesar de ello, este *sprint* fue en el que menos horas se inviertieron. La necesidad de completar un trabajo de una asignatura del Grado en Estadística y la continuación del TFG del mismo grado no dejó tiempo libre para el desarrollo del proyecto. Se invirtieron solo 16 horas y 30 minutos durante el *sprint* 7. Se puede ver cómo se emplearon esas horas en la tabla 9.7.

Historia de usuario	Tareas	Tiempo estimado	Tiempo empleado	Estado
US18	▪ Obtención del Caso de Uso equivalente ▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	5 horas	3 horas	Finalizada
US23 y US24	▪ Pruebas	1 hora	1 hora 30 minutos	Finalizada
US08	▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	10 horas	8 horas	Finalizada
US04 y US19	▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	4 horas	4 horas	Finalizada

Tabla 9.7: Desglose de tareas del *sprint* 7.

A pesar de no disponer de una gran cantidad de horas en este *sprint*, se realizaron avances importantes. En el Sprint 6 (29/03/24 - 05/04/24) no se tuvo tiempo suficiente para terminar

todas las historias de usuario de la épica EP04, aunque solo faltaba **US18**. Su desarrollo fue más rápido de lo previsto, pues su diseño e implementación fueron muy similares a los de la historia de usuario **US17**.

Se le permitió al docente añadir tantas preguntas a una actividad como desee, como indica el caso de uso **US08**, para ello fue necesario consultar como añadir dinámicamente tantas preguntas como se quiera [129]. Derivado de esta nueva funcionalidad, cuando se acceda a los detalles de una actividad, se deberán mostrar las preguntas asociadas a esa actividad, en caso de que haya. Para cumplir estos requisitos, se realizó una nueva iteración sobre **US04** y **US19**.

9.2.9. Sprint 8 (12/04/24 - 19/04/24)

En la reunión celebrada el viernes 12 de abril, se obtuvo un *feedback* positivo pero también se recibieron varias recomendaciones de cara a mejorar la aplicación y facilitar su manejo para los docentes. Entre los cambios mencionados, los más importantes son:

- Se debe añadir un usuario de tipo administrador que pueda ver todas las actividades registradas y se las pueda asignar a todos los estudiantes registrados en el sistema.
- Se debe crear un usuario de tipo familia que pueda resolver actividades familiares.
- Los estados de ánimo deben de ser globales, da igual qué docente los haya creado.
- Se deben quitar los datos personales de los estudiantes.

Como no hay ninguna tarea pendiente del *sprint* anterior, se decidió comenzar con la implementación del nuevo tipo de usuario familia y administrador. Tras una etapa de análisis, se decidió que el usuario familia fuera una extensión de cada estudiante, ya que su funcionalidad está muy limitada y no puede existir una familia sin que un miembro de dicha familia sea el estudiante. Se definieron las historias de usuario **US25**, **US26** y **US27** y se pasó a su desarrollo. También se añadieron las historias de usuario derivadas del usuario administrador. Se pueden ver detalladamente en la Tabla 9.8.

Historia de usuario	Tareas	Tiempo estimado	Tiempo empleado	Estado
-	Actualización de documentación, base de datos y código	1 hora 30 minutos	2 horas	Finalizada
US25	▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	2 horas	2 horas	Finalizada
US26	▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	1 hora	1 hora	Finalizada
US27	▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	1 hora	1 hora	Finalizada
US28	▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	1 hora	1 hora 30 minutos	Finalizada
US29	▪ Obtención del Caso de Uso equivalente ▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	2 hora	2 hora	Finalizada
-	Inicio de escritura de la memoria del TFG	4 horas	4 horas	En progreso

Tabla 9.8: Desglose de tareas del *sprint* 8.

9.2.10. Sprint 9 (19/04/24 - 26/04/24)

Durante el *sprint* 9, se tenía la intención de finalizar el desarrollo de la aplicación con el objetivo de poder desplegarla en producción y comenzar las pruebas en entornos reales. Este

paso era crucial para evaluar el desempeño y la funcionalidad del software en condiciones auténticas de uso, permitiendo identificar y corregir posibles errores o mejoras necesarias. La meta era asegurar que la aplicación estuviera completamente operativa y lista para su uso por parte de los usuarios finales. Con tal de conseguir este objetivo, se trabajaron 29 horas en este *sprint*. Al final del *sprint* se consiguió la implementación de las funcionalidades especificadas por las dos últimas historias de usuario que todavía no se habían tratado: **US11** y **US21**. En la Tabla 9.9 se puede ver en detalle qué tareas se llevaron a cabo:

Historia de usuario	Tareas	Tiempo estimado	Tiempo empleado	Estado
-	Actualización de documentación, base de datos y código	2 horas	2 horas	Finalizada
US11	▪ Obtención del Caso de Uso equivalente ▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	12 horas	10 horas	Finalizada
US21	▪ Obtención del Caso de Uso equivalente ▪ Análisis ▪ Diseño ▪ Implementación ▪ Pruebas	12 horas	15 horas	Finalizada
-	Continuación de la documentación de la memoria del TFG	3 horas	2 horas	En progreso

Tabla 9.9: Desglose de tareas del *sprint* 9.

Desde el inicio del desarrollo del proyecto, **US11** y **US21** se consideraban las historias de usuario que requerían una funcionalidad más compleja, por eso se estimó una inversión de horas tan alta en implementarlas.

Dentro de la actualización de código de código, se realizó un pequeño ajuste para centrar correctamente las imágenes en las cartas de los estados de ánimo [131].

En cuanto a las historias de usuario, se necesitó hacer uso de preguntas resueltas en *Stack Overflow*, pues se estaba teniendo un problema en la estructura de herencia de plantillas de Jinja2 [134] [135]. Para la implementación de **US21**, se decidió especializar el la plantilla de HTML de detalles de una actividad y añadir el cuestionario abajo. Sin embargo, a mayores del problema ya mencionado, surgieron otros dos problemas. El primer problema se trataba del desconocimiento de la palabra clave en SQL para introducir el instante de finalización de

una asignación, se pudo resolver fácilmente consultando la documentación en internet [132]. El segundo problema consistía en un comportamiento inadecuado del botón para resolver la actividad, se pudo resolver gracias a que otro usuario de *Stack Overflow* tuvo el mismo problema [133].

9.2.11. Sprint 10 (26/04/24 - 03/05/24)

Durante este *sprint*, no fue posible invertir tiempo en el desarrollo del proyecto debido a una confluencia de responsabilidades académicas. La carga de trabajo incluía la preparación y defensa de dos trabajos importantes, así como la realización de un examen final de la asignatura del Grado en Estadística. Estas obligaciones demandaron una atención considerable, lo que resultó en la imposibilidad de avanzar en las tareas planificadas para el sprint 9. La prioridad asignada a estos compromisos fue esencial para asegurar un desempeño académico óptimo y que no hubiera riesgo de acudir a una convocatoria extraordinaria, lo que habría lastrado aún mas el desarrollo del proyecto.

9.2.12. Sprint 11 (03/05/24 - 10/05/24)

Al igual que en el Sprint 10 (26/04/24 - 03/05/24), las responsabilidades académicas más urgentes no permitieron invertir tiempo en el desarrollo del proyecto.

9.2.13. Sprint 12 (10/05/24 - 17/05/24)

A partir de este *sprint*, solo se realizarán retoques o ligeras modificaciones al proyecto software pues ya se considera que está implementado, el mes restante hasta la fecha de entrega se dedicó a la redacción de la memoria del TFG y a realizar las correcciones recomendadas por el tutor. También, el inicio este *sprint* marca el instante en el que la dedicación del estudiante se centra únicamente en el desarrollo de este proyecto, pues ha finalizado el resto de responsabilidades académicas. Esta disponibilidad se refleja en el tiempo dedicado en cada *sprint* a partir de este, en el que se han dedicado 54 horas y 30 minutos. En la Tabla 9.10 se puede ver los avances realizados este *sprint*:

Historia de usuario	Tareas	Tiempo estimado	Tiempo empleado	Estado
-	Redacción del Capítulo 2	-	22 horas 30 minutos	Pendiente de corrección
-	Redacción del Capítulo 4	-	32 horas	Pendiente de corrección

Tabla 9.10: Desglose de tareas del *sprint* 12

9.2.14. Sprint 13 (17/05/24 - 24/05/24)

Al igual que en el *sprint* anterior Sprint 12 (10/05/24 - 17/05/24), se ha dedicado la mayor parte del tiempo a la redacción de la memoria final del TFG. Aún así, se ha empleado una menor cantidad de horas en la mejora del producto final resolviendo *bugs* y mejorando la experiencia del usuario. Se puede ver en detalles el uso de las 52 horas y 45 minutos invertidos en la Tabla 9.11.

Historia de usuario	Tareas	Tiempo estimado	Tiempo empleado	Estado
-	Corrección y mejora del Capítulo 2	-	8 horas	Finalizada
-	Corrección y mejora del Capítulo 4	-	10 horas	Finalizada
-	Redacción y documentación del Capítulo 5	-	20 horas 15 minutos	Pendiente de corrección
-	Redacción y documentación del Capítulo 6	-	10 horas	En progreso
-	Mejora del *feedback* de la interfaz	-	2 horas 30 minutos	Finalizada
-	Solución de *bugs*	-	2 horas	Finalizada

Tabla 9.11: Desglose de tareas del *sprint* 13.

En este *sprint* se llevaron a cabo tres tipos de tareas paralelamente: La redacción/corrección de los documentos que compondrán la memoria final del TFG, la solución de *bugs* y la mejora de la interfaz.

Se consiguieron resolver *bugs* de bajo impacto pero notables que pueden hacer el uso de la aplicación algo confuso. En concreto, se corrigió el contador del número de preguntas de la plantilla `add_preguntas`, que no funcionaba correctamente. También se corrigió un bug de la plantilla `asignar_actividad.html` donde al hacer click sobre el nombre de cualquier estudiante, solo se seleccionaba el *checkbox* del estudiante que figuraba primero en la lista.

En cuanto a la mejora de la interfaz, se añadió una petición de confirmación antes de realizar una acción [136] [137] y se incluyó una serie de mensajes *flash* [138] propios de *Flask* para dar *feedback* sobre la acción realizada.

9.2.15. Sprint 14 (24/05/24 - 31/05/24)

Este *sprint* se dedicó únicamente a la redacción del documento final del TFG. En total se invirtieron 65 horas, cuyo desglose se puede ver en la Tabla 9.12.

Historia de usuario	Tareas	Tiempo estimado	Tiempo empleado	Estado
-	Corrección y mejora del Capítulo 5	-	20 horas	Finalizada
-	Redacción y documentación del Capítulo 6	-	20 horas	Pendiente de corrección
-	Redacción y documentación del Capítulo 9	-	25 horas	En progreso

Tabla 9.12: Desglose de tareas del *sprint* 14.

En las recomendaciones del tutor para mejorar el Documento de Análisis, hubo varia información y secciones que añadir y gráficos que corregir por lo que tomó más tiempo de lo normal.

9.2.16. Sprint 15 (31/05/24 - 07/06/24)

Casi todo el *sprint* 15 se ha dedicado a avanzar el la redacción del documento final del TFG. A mayores se dedicaron 5 horas a lo largo de dos días distintos (5 y 6 de junio) para arreglar problemas que hacían que la aplicación se cayera. Se ha mantenido al igual que en los *sprints* anteriores un alto número de horas de trabajo. En total se invirtieron 65 horas y 30 minutos, en la Tabla 9.13 se puede ver cómo se repartieron.

Historia de usuario	Tareas	Tiempo estimado	Tiempo empleado	Estado
-	Corrección y mejora del Capítulo 5	-	20 horas 30 minutos	Finalizada
-	Corrección y mejora del Capítulo 6	-	14 horas 15 minutos	Finalizada
-	Redacción y documentación del Capítulo 9	-	25 horas 45 minutos	En progreso
-	Pruebas del sistema y correcciones	-	5 horas	Finalizada

Tabla 9.13: Desglose de tareas del *sprint* 15.

A lo largo del *sprint* se consiguieron arreglar dos problemas recurrentes de la aplicación. El primer *bug* sucedía cuando un Estudiante registraba un estado nuevo cuando solo había un Estado de Ánimo guardado en la base de datos. El *Transaction Script* encargado de mostrar los Estados de Ánimo registrados en el sistema esperaba que se le pasara una lista de Estados de Ánimo (aunque fuera de longitud 1), sin embargo este recibía un Estado de Ánimo. El segundo *bug* sucedía cuando se intentaba crear un nuevo Estudiante. Si se

introducía un identificador de curso no registrado, el programa se colgaba. Los dos errores pudieron corregirse en apenas 3 horas y se pudo invertir el resto del tiempo en pruebas o errores menores.

9.2.17. Sprint 16 (07/06/24 - 14/06/24)

El *sprint* 16 se dedicó a documentar toda la implementación llevada a cabo a lo largo del proyecto y documentar una serie de casos de prueba para verificar el correcto funcionamiento de la aplicación. También se revisó toda la documentación redactada hasta el momento, corrigiendo pequeños errores y la bibliografía. Previamente a la serie de casos de pruebas del sistema se terminó de revisar la interfaz del usuario y, posteriormente a ellas se arreglaron aquellos errores que se encontraron. En total se hizo uso de 55 horas 30 minutos a lo largo del *sprint*. Se puede ver el tiempo empleado en detalle en la Tabla 9.14.

Historia de usuario	Tareas	Tiempo estimado	Tiempo empleado	Estado
-	Retoques y cambios en los Capítulos anteriores	-	5 horas	Finalizada
-	Redacción, documentación y corrección del Capítulo 3	-	7 horas	Finalizada
-	Redacción y documentación del Capítulo 7	-	18 horas	Pendiente de corrección
-	Redacción y documentación del Capítulo 8	-	16 horas	Pendiente de corrección
-	Redacción y documentación del Capítulo 9	-	4 horas 30 minutos	Pendiente de corrección
-	Solución de pequeños errores	-	2 horas	Finalizada
-	Mejoras de la IU	-	3 horas	Finalizada

Tabla 9.14: Desglose de tareas del *sprint* 16.

9.2.18. Sprint 17 (14/06/24 - 20/06/24)

Al tratarse del último *sprint* del proyecto, el *sprint* se dedicó enteramente a rematar detalles del documento final y hacer pequeños retoques en el código. Se escribió el Capítulo 9 donde se reúnen las conclusiones finales del proyecto, se revisaron y corrigieron Capítulos anteriores y se añadió el manual de usuario para hacer de guía a nuevos usuarios. Se puede ver las tareas el tiempo empleado en cada tarea en la Tabla 9.15.

Historia de usuario	Tareas	Tiempo estimado	Tiempo empleado	Estado
-	Corrección y mejora del Capítulo 7	-	1 hora	Finalizada
-	Corrección y mejora del Capítulo 8	-	2 horas	Finalizada
-	Corrección y mejora del Capítulo 9	-	18 horas	Finalizada
-	Redacción, corrección y mejora del Capítulo 10	-	3 horas	Finalizada
-	Redacción, corrección y mejora del Manual de Usuario	-	12 horas	Finalizada
-	Retoques del código	-	1 hora	Finalizada

Tabla 9.15: Desglose de tareas del *sprint* 17.

En total se dedicaron 37 horas hasta el día 18 de junio de 2024. No se trabajó en los días 19 y 20 de junio al ya haber finalizado el proyecto.

9.3. Conclusiones del seguimiento

9.3.1. Tiempo total del proyecto

En la sección 2.5 se estimó que, a pesar de que el TFG consista de 300 horas de trabajo, el estudiante emplearía en torno a unas 120 horas extra, es decir, 420 horas en total. Finalmente, sumando todo el tiempo invertido a lo largo de todos los *sprints*, se ha alcanzado un tiempo empleado de 524 horas y 15 minutos, un 24,82 % por encima del tiempo estimado en la fase de planificación. Este aumento se debe en gran parte a que no se tenía previsto dedicar tantas horas a la confección del documento final del TFG.

Todas las reuniones se celebraron los días previstos (cada viernes) o en jueves en caso de que fuera más conveniente, como en el *sprint* 14. A mayores, ha habido dos semanas en las que no ha sido posible reunirse, una por las vacaciones de semana santa y otra por los compromisos académicos del estudiante. Estas *Scrum Weekly* que estaban planificadas para los días 29/03/2024 y 03/05/2024 respectivamente no se llegaron a celebrar a pesar de estar planificadas. Sin embargo, se mantuvo una continua comunicación por correo electrónico durante dichos *sprints* para comunicar cualquier duda o pedir sugerencias sobre cómo avanzar.

Sobre la distribución de la carga de trabajo, es muy rápido darse cuenta que ha sido muy desigual mayormente a causa del resto de responsabilidades académicas del estudiante. La media de horas empleadas en el proyecto previamente a la finalización del resto de compromisos universitarios, 16 horas y 15 minutos, es claramente inferior a la media de horas empleadas posteriormente, 55 horas. En la planificación de riesgos de la Sección 2.6 ya se tuvo en cuenta que esto pudiera suceder en el riesgo R06, por lo que el estudiante decidió

aumentar drásticamente el tiempo dedicado al proyecto pues, no solo no se habían conseguido invertir las 20 horas estimadas semanales, sino que también hubo dos *sprints* en los que el estudiante no pudo trabajar en el proyecto. Estos contratiempos también causaron cambios en los plazos de entrega y una reevaluación de la planificación.

Finalmente, el proyecto estuvo en desarrollo desde el 14 de febrero hasta el 18 de junio, aproximadamente 4 meses de trabajo.

9.3.2. Evaluación de los costes finales

9.3.2.1. Coste simulado final

En la Sección 2.7 se estimó que el coste total simulado alcanzaría la suma de 10.888,50€. Todos los costes se calcularon mensualmente y se multiplicaron por los 4 meses que se pronosticó que duraría el desarrollo del proyecto. Como la duración total del proyecto fue correctamente estimada, los costes simulados finales coinciden con los presupuestados. En la Tabla 9.16 se puede ver su desglose.

Concepto	Precio mensual	Meses	Total
Desarollador *Full-Stack*	2500€	4	10000€
Espacio de *coworking*	155€	4	620€
Equipo de trabajo *Dell*	31,06€	4	124,22€
Licencia *Visual Paradigm*	17,69€	4	70,76€
Licencia *Balsamiq Cloud*	8,38€	4	33,52€
Licencia *Microsoft Teams*	10€	4	40€
COSTE TOTAL			**10.888,50€**

Tabla 9.16: Desglose de los costes totales simulados.

9.3.2.2. Coste real final

Al igual que se acaba de comentar en los costes simulados finales, la duración del proyecto coincide con la duración pronosticada, no ha sido necesario adquirir un nuevo equipo de trabajo ni una herramienta software inesperadamente. Los costes reales finales coinciden con los planificados en la sección 2.7. En la Tabla 9.17 se puede ver la distribución de los costes.

Concepto	Precio mensual	Meses	Total
Equipo de trabajo *Huawei*	16,66€	4	66,66€
Electricidad	0,68€	4	2,72€
COSTE TOTAL			**69,38€**

Tabla 9.17: Desglose de los costes totales reales.

Capítulo 10

Conclusiones y trabajo futuro

10.1. Introducción

La culminación de este trabajo de desarrollo se refleja en este capítulo de Conclusiones y Trabajo Futuro. Desde el inicio de este proyecto, han surgido numerosos desafíos y diferentes obstáculos. Se ha dedicado numerosas horas de trabajo para alcanzar los objetivos establecidos. Sin embargo, cada paso dado, cada desafío superado, ha servido para afianzar y aprender nuevos conocimientos relacionados con el desarrollo de sistemas web y la Ingeniería del Software, conceptos que no se estudian con tanta profundidad en la mención de computación.

10.2. Conclusiones

En esta sección, se presentan las conclusiones finales derivadas del proyecto llevado a cabo. A lo largo de los cuatro meses empleados, se ha realizado todo el trabajo con el fin de conseguir desarrollar una aplicación que satisficiera tanto los objetivos del Departamento de Psicología Evolutiva y de la Educación como los personales. En cuanto a los objetivos de desarrollo planteados en la Sección 1.5.1:

1. Se ha conseguido desarrollar una aplicación web que satisfaga los requisitos del cliente. El alcance de la aplicación cubre todos los requisitos funcionales y se han respetado todos los requisitos no funcionales.
2. Se ha conseguido desarrollar la documentación siguiendo las principales metodologías y prácticas de *scrum*.
3. Se ha conseguido desarrollar una base de datos capaz de almacenar la información necesaria de la aplicación web. Se ha desplegado una base de datos relacional en un servidor MySQL que cubre la necesidad de la persistencia de datos de la aplicación.

4. Se ha conseguido poner la aplicación en producción y hacer pruebas con ella en centros educativos.

5. Se ha conseguido desarrollar una interfaz fácil de usar. A lo largo del desarrollo de la aplicación, se centraron gran parte de los esfuerzos en conseguir una interfaz de usuario intuitiva y sencilla para los estudiantes.

6. Se ha conseguido cumplir las normas y buenas prácticas de programación. Se han respetado las convenciones de nomenclatura de Python y SQL, se han respetado las directrices de los patrones empleados y para el desarrollo de cada historia de usuario, se han seguido los *workflows* del Proceso Unificado.

En cuanto a los objetivos personales presentados en la sección 1.5.2:

1. Se ha conseguido realizar un proyecto de Ingeniería del Software desde el principio y se han adquirido conocimientos de desarrollo de software y sistemas web.

2. Se ha conseguido aplicar el marco de trabajo *scrum* satisfactoriamente.

3. Se ha conseguido desarrollar habilidades de gestión de proyectos ágiles y trabajar bajo presión con una fecha límite.

4. Se ha impulsado la autoorganización y responsabilidad personal al haber conseguido trabajar de manera eficaz en paralelo dos TFGs y una asignatura del Grado en Estadística.

5. Se han cumplido con los objetivos y plazos establecidos en la planificación del proyecto.

6. Se ha adaptado la aplicación a los cambios de requisitos que han ido surgiendo a lo largo del proyecto.

10.3. Trabajo Futuro

A lo largo del desarrollo de este proyecto, se han identificado diversas oportunidades y áreas de mejora que podrían ampliar y mejorar la implementación actual. Sin embargo, debido a las restricciones de tiempo y recursos, no ha sido posible abordar todas estas ideas por lo que se han quedado fuera del alcance de la aplicación. En esta sección, se exploran algunas de estas líneas futuras potenciales, destacando las áreas específicas donde podrían realizarse mejoras significativas en el futuro.

- **Nuevas funcionalidades del Administrador:** El usuario Administrador tiene una gama de funcionalidades disponibles muy limitada pues se añadió a raíz de un nuevo requisito añadido el séptimo *Sprint*. Como funcionalidades adicionales, el Administrador debe poder supervisar y gestionar todos los centros y cursos registrados en el sistema.

- **Roles de Docentes:** Cuando un docente se registra, se guarda información sobre su rol en el centro: Profesor, Orientador, Jefe de estudios o Director. Se puede mejorar la especialización de este tipo de usuarios si se añadiera un menú con acciones exclusivas de cada rol. El profesor mantendría sus acciones implementadas en este proyecto, el orientador podría revisar los diarios emocionales de todos los estudiantes de su centro y entablar conversaciones o citar reuniones con ellos, el director se encargaría de supervisar a los docentes de su centro, etc.

- **Apartado *perfil*:** Una posible vía de desarrollo es la adición del apartado del *perfil de usuario* añadiendo opciones para personalizar la información personal, cambiar la foto de perfil y mostrar el estado de ánimo en caso de que sea estudiante. También sería útil incluir una sección donde los usuarios puedan ver su historial de actividad en la plataforma.

- **Incluir de más de un recurso multimedia de cada tipo:** En las actividades solo se permite adjuntar un recurso multimedia de cada tipo como mucho: una imagen, un vídeo y un audio. Una mejora evidente sería permitir un mayor número de archivos de cada tipo, limitado a un tamaño conjunto fijo.

- **Mejora de la intefaz de las Actividades:** A pesar de que la interfaz de usuario de la aplicación recibió el visto bueno por parte del Departamente de Psicología Evolutiva y de la Educación de la Universidad de Valladolid, a lo largo del desarrollo y al poner la aplicación en producción, surgieron varias posibles mejoras para la interfaz:

  - Añadir una barra de navegación global para todos los usuarios.
  - Alternar la exposición de la imagen, vídeo y audio de las actividades mediante un botón para cada uno de ellos.
  - Añadir un *pop up* propio de la aplicación para pedir confirmación en vez de usar el propio incorporado del navegador web.

- **Mejora en la asignación de actividades:** Incluir un sistema de filtrado de estudiantes a los que se le va a asignar una actividad. Se puede añadir tanto un método que haga que se muestren solo los alumnos que pertenecen a una clase elegida como una barra de búsqueda.

Apéndice A

Manuales

En esta sección, se presentan los manuales necesarios para comprender y utilizar la aplicación web. Dado que se trata de una aplicación web, no es necesario un manual de instalación, ya que no requiere ningún proceso de instalación en los dispositivos de los usuarios. El Manual de Usuario proporciona imágenes de la aplicación e instrucciones claras para ayudar a los usuarios a navegar y aprovechar al máximo todas las funcionalidades de la aplicación.

A.1. Manual de usuario

Al tratarse de una aplicación web se necesita de un navegador web. Cada vez que se accede a la aplicación, se hace como un usuario no identificado, la pantalla inicial al acceder a la aplicación se puede ver en la Figura A.1.

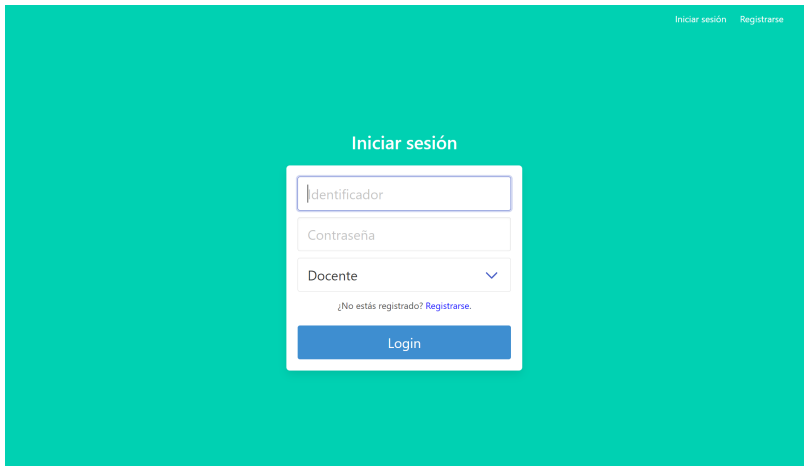

Figura A.1: Pantalla de inicio de sesión.

Es rápido ver que las únicas dos funcionalidades disponibles son **Iniciar sesión** y **Crear una cuenta**, al acceder a la aplicación el usuario se sitúa en la pantalla de inicio de sesión y hay dos formas de acceder a registrarse: mediante la barra de navegación o presionando en *Registrarse* debajo de la lista desplegable donde se elige el rol con el que se quiere iniciar sesión.

Para registrarse solo hace falta rellenar los campos que son solicitados. Se puede ver la pantalla de registro para estudiantes en la Figura A.2.

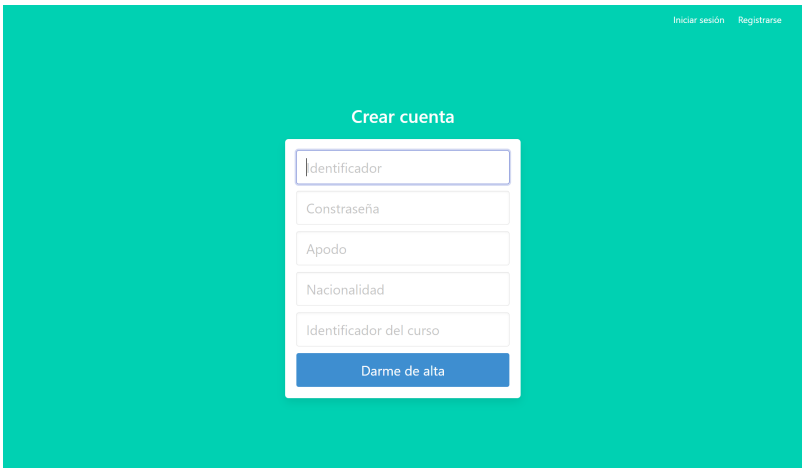

Figura A.2: Pantalla de registro de estudiante

En la aplicación existen cuatro roles diferenciados, sin embargo, los dos roles que aglutinan la mayor parte de la funcionalidad de la aplicación son **docente** y **estudiante**, por lo tanto se profundizará en ellos. A niveles prácticos el rol de familia solo es una extensión de un estudiante con menos privilegios (por lo que no tiene funcionalidades distintas) y el administrador tampoco tiene ninguna pantalla propia de su rol.

A.1.1. Docente

Cuando un docente inicia sesión, accede a un menú formado por dos tarjetas mostrado en la Figura A.3. En este menú puede elegir si acceder a su menú de actividades o a su menú de diario emocional.

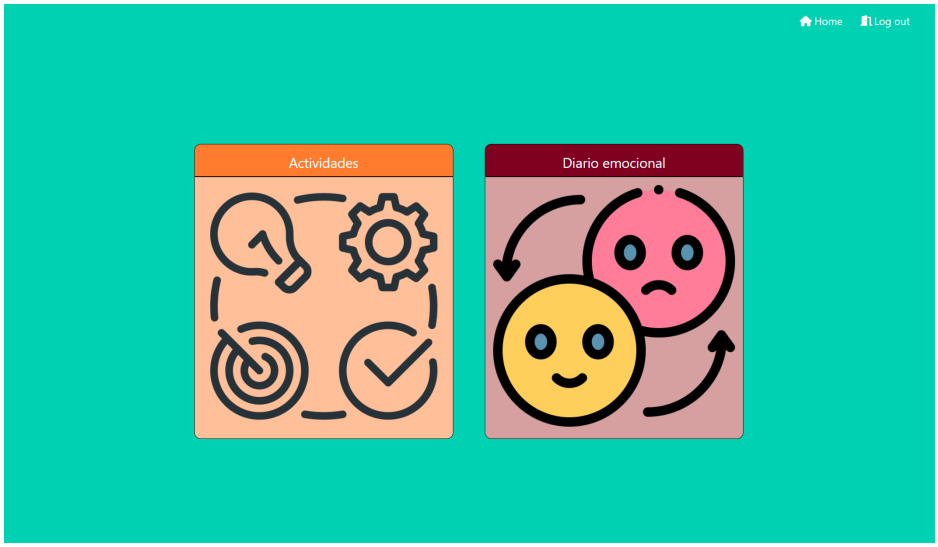

Figura A.3: Menú principal del docente

Si accede al menú de actividades presionando su tarjeta, se mostrarán dos tarjetas como se puede ver en la Figura A.4. A partir de todas las pantallas habrá un botón *Atrás* en la esquina superior izquierda para hacer más fluida la navegación.

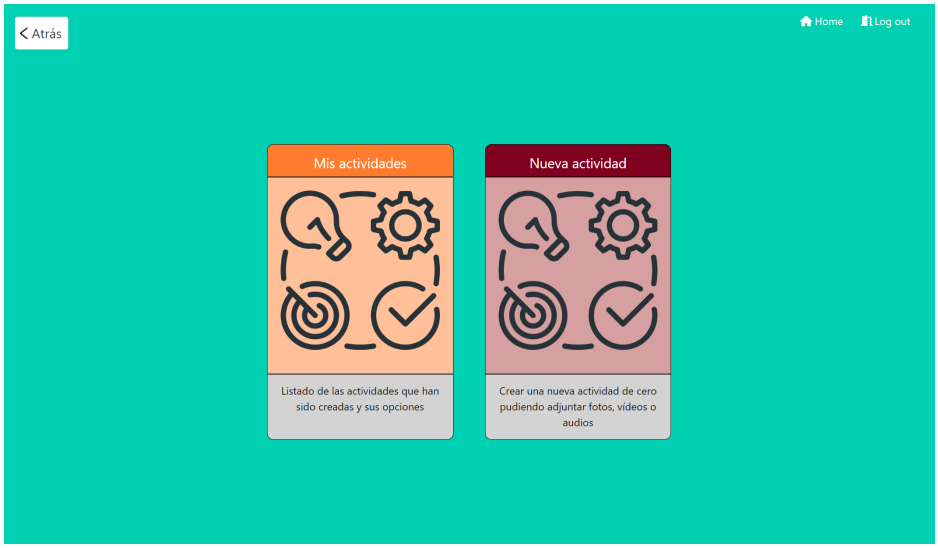

Figura A.4: Menú de Actividades

En el menú de actividades de la Figura A.4, si el docente elige la targeta de *Nueva actividad*, podrá crear una nueva actividad desde cero. Inicialmente se le mostrará la pantalla

de la Figura A.5 donde se introducirá el nombre y descripción de la actividad y, en caso de que el docente quiera, un archivo de cada tipo de recurso multimedia, que puede ser una imagen, un vídeo o un audio.

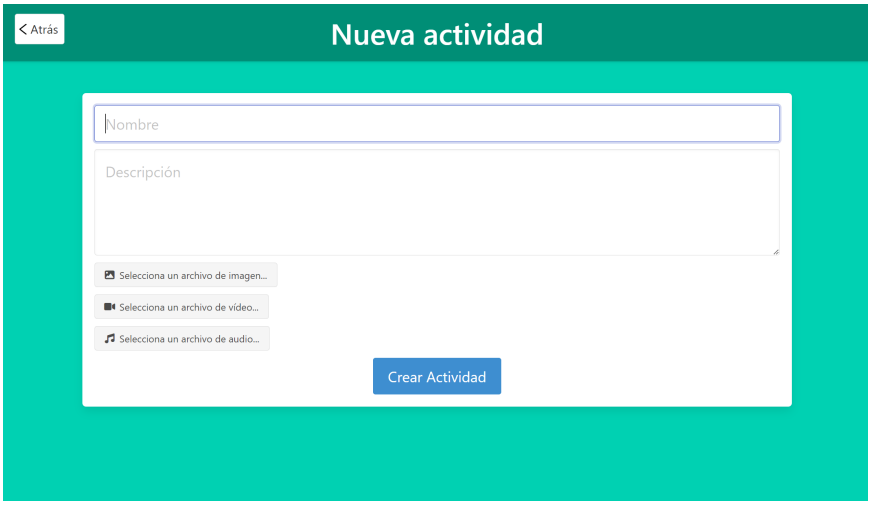

Figura A.5: Crear una Actividad

Una vez que el docente haya introducido todos los datos o adjuntado los ficheros que desee, puede pulsar el botón *Crear Actividad* y se le preguntará si quiera añadir preguntas. En caso de que no quiera, la actividad se crea y se redirige al docente a su lista de actividades creadas. En caso de querer añadir preguntas, es redirigido a la pantalla mostrada en la Figura A.6.

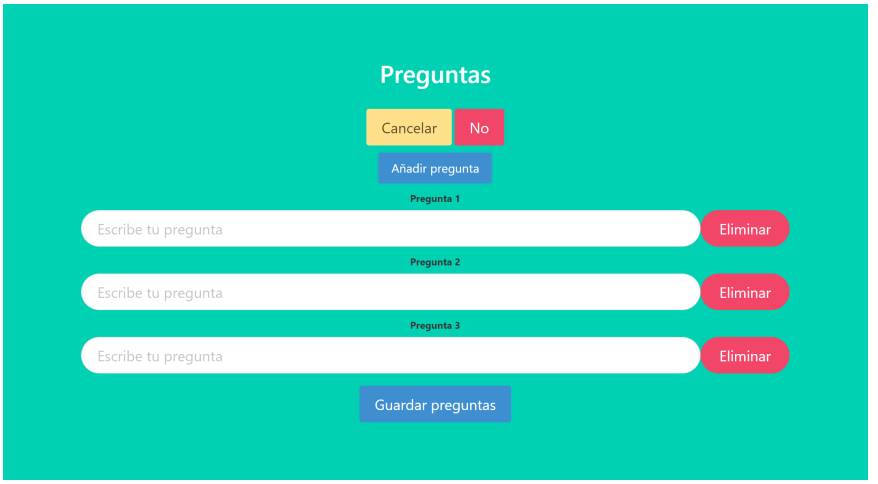

Figura A.6: Añadir Preguntas a una Actividad

En la Figura A.6 se muestran ya tres cajas de texto para introducir preguntas pero inicialmente no se tiene ninguna, se pueden añadir una a una presionando el botón *Añadir pregunta*. El docente en todo momento puede indicar que *No* quiere añadir preguntas y se creará la actividad o puede pulsar el botón *Cancelar* y se eliminarán todas las preguntas a la vez. En caso de que haya terminado de introducir las preguntas que desee, presiona *Guardar preguntas* y estás se guardarán asociadas a la actividad creada. El docente entonces será redirigido a su lista de actividades.

Volviendo al menú de actividades, si el docente hace *click* en la tarjeta *Mis actividades*, podrá ver un listado de todas sus actividades creadas y las opciones asociadas a cada una como se muestra en la Figura A.7.

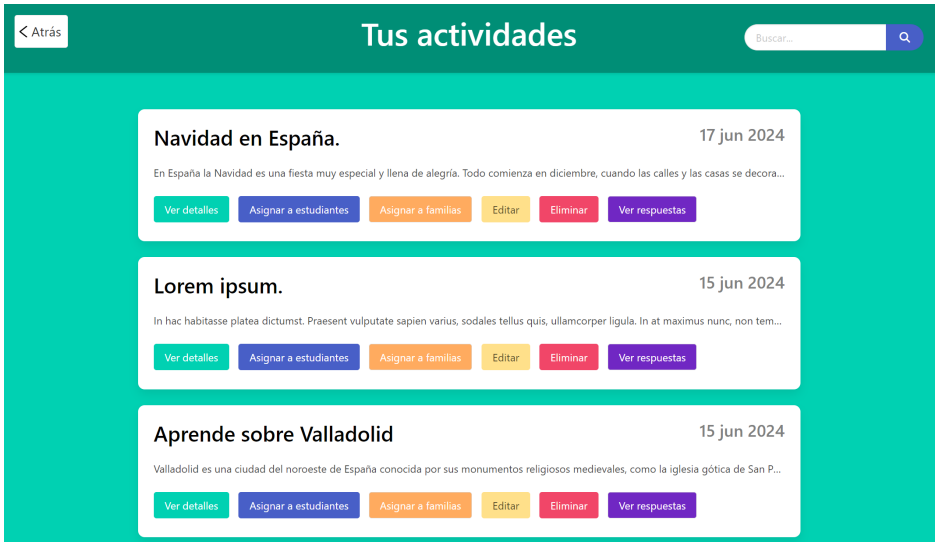

Figura A.7: Lista de Actividades del Docente

En cuanto a las opciones que tiene el docente, si se pulsa en cualquiera de los botones de la tarjeta de una actividad, se iniciará la funcionalidad descrita por el nombre del botón. Si el docente pulsa el botón detalles de una actividad, se mostrarán sus detalles y sus preguntas asociadas en caso de que tenga. Se puede ver la pantalla de detalles de una actividad en la Figura A.8.

Figura A.8: Detalles de una Actividad

Si presiona el botón *Asignar a estudiantes* o *Asignar a familias*, se mostrará un listado de los estudiantes a los que el docente imparte clase, se puede ver en la Figura A.9.

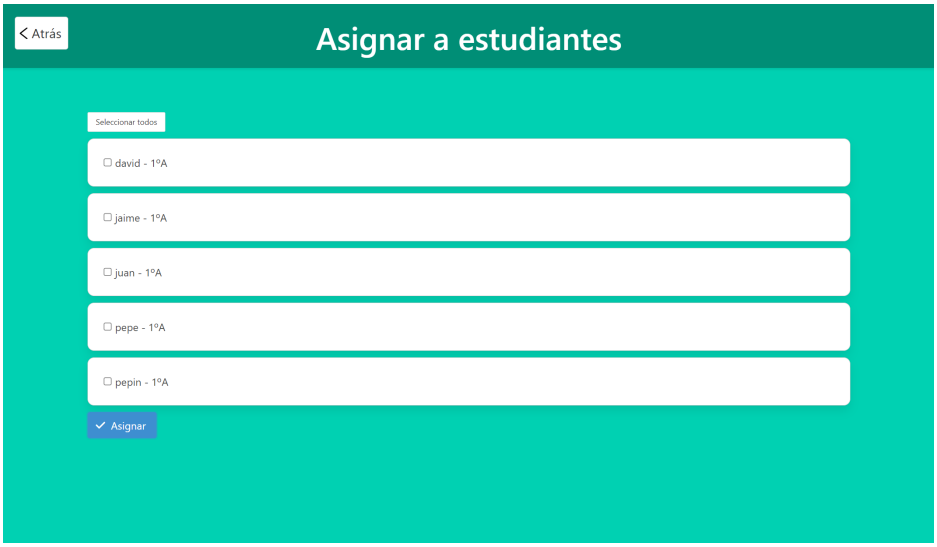

Figura A.9: Asignar una Actividad

El docente podrá seleccionar a los estudiantes o familias a los que les quiere asignar la actividad presionando en el cuadro de selección de cada uno o en su nombre directamente. También puede seleccionar todos los estudiantes a la vez usando el botón *Seleccionar todos*. Se registrarán todas las asignaciones de la actividad a los estudiantes seleccionados cuando se presione el botón *Asignar* y se confirme.

Siguiendo con el orden de los botones de la tarjeta de cada actividad, el docente también tiene permitido editar cualquier actividad creada por él. Si presiona el botón *Editar* se seguirá el mismo flujo de usuario que al crear una actividad, solo que los campos solicitados estarán ya rellenados con los datos de la actividad a editar, como se puede ver en la Figura A.10.

Figura A.10: Asignar una Actividad

Se puede ver en la Figura A.10 como los campos Nombre y Descripción tienen los datos ya introducidos y también se muestra la imagen adjuntada. Si se pulsa el botón Editar Actividad, se sigue el mismo flujo que al crear una actividad.

En la Figura A.7, en cada tarjeta aparece el botón de color rojo *Eliminar*. Este botón sirve, como indica su nombre, para eliminar una actividad que haya creado el docente. Si se presiona, el docente es redirigido a la pantalla mostrada en la Figura A.11. Esta pantalla muestra los detalles de la actividad y muestra un botón solicitando la confirmación de la eliminación. Si el docente presiona el botón *Confirmar eliminación*, la actividad será eliminada del sistema y el docente será redirigido de vuelta a su lista de actividades A.7.

Figura A.11: Eliminar una Actividad

Por último, un docente tiene la posibilidad de ver las respuestas de todas las asignaciones de una actividad a un estudiante. Si presiona el botón *Ver respuestas*, se muestra la pantalla de la Figura A.12 donde se pueden ver todas las asignaciones. Se muestra el estudiante al que se le asignó la actividad, la fecha en la que se asignó y, en caso de que el estudiante haya finalizado la asignación, la fecha de finalización.

Figura A.12: Respuestas a una Actividad

En caso de que el estudiante haya finalizado la asignación, también se habilita el botón *Ver respuesta*. Si se pulsa este botón, se puede ver los detalles de la actividad y las respuestas del estudiante a sus preguntas como se muestra en la Figura A.13. En caso de que no haya

preguntas, solo se verán los detalles de la actividad.

Figura A.13: Actividad resuelta

Si desde el menú inicial del docente de la Figura A.3 se hace *click* en la tarjeta de *Diario emocional*, se accede el menú de diario emocional del docente mostrado en la Figura A.14.

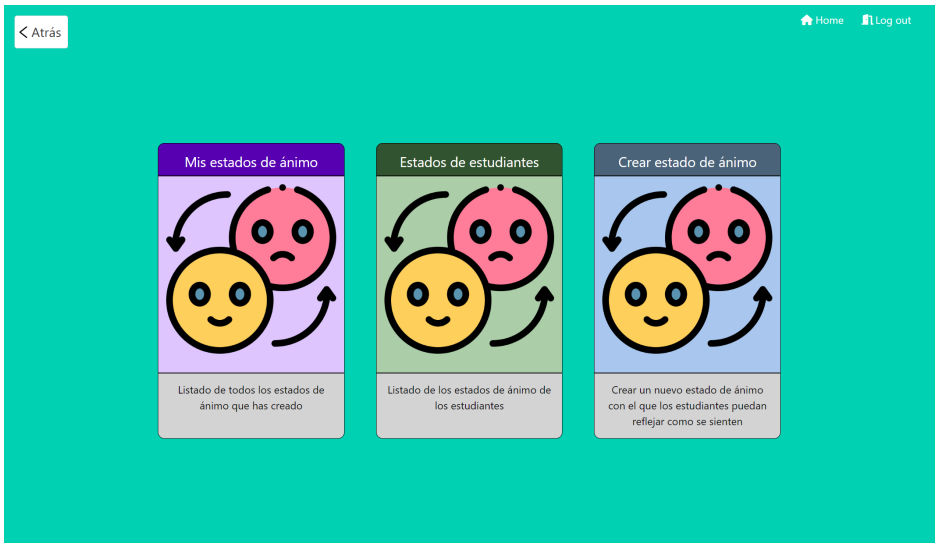

Figura A.14: Menu del Diario Emocional

En este menú se muestran tres tarjetas: *Mis estados de ánimo*, *Estados de estudiantes* y *Crear estado de ánimo*. Si pulsa la tarjeta *Crear estado de ánimo*, el docente tiene acceso a

una página similar a la de crear una actividad como se puede ver en la Figura A.15.

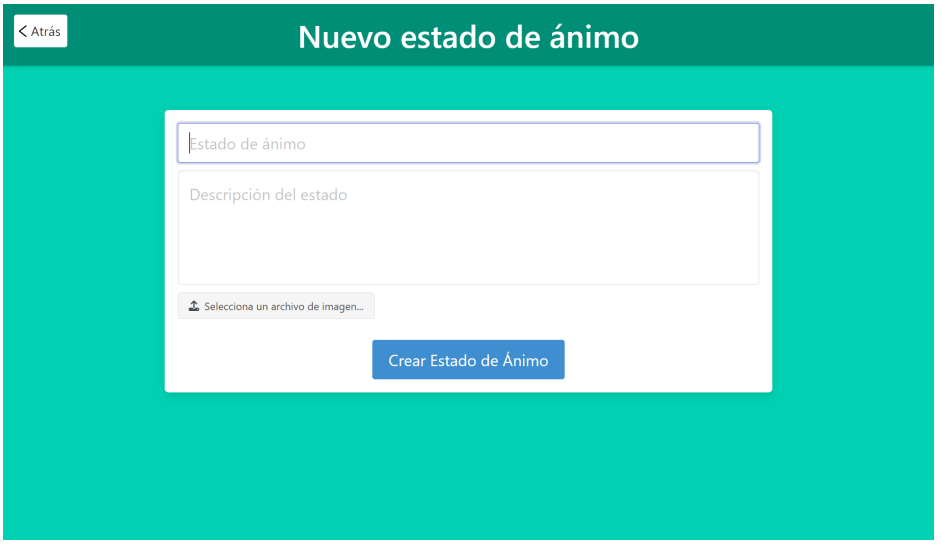

Figura A.15: Crear estado de ánimo

En esta página el docente puede introducir el nombre y descripción de un estado de ánimo que quiera registrar en el sistema. También podrá adjuntar una imagen descriptiva sobre el estado de ánimo. Si el docente pulsa en *Crear Estado de Ánimo*, el estado se creará y se registrará en el sistema a su nombre. El docente entonces es redirigido a su menú de estados de ánimo.

Volviendo al menú de estados de ánimo de la figura A.14, si el docente hace *click* sobre la tarjeta de *Mis estados de ánimo* se mostrarán todos los estados de ánimo registrados en el sistema creados por él. Se puede ver esta pantalla en la Figura A.16.

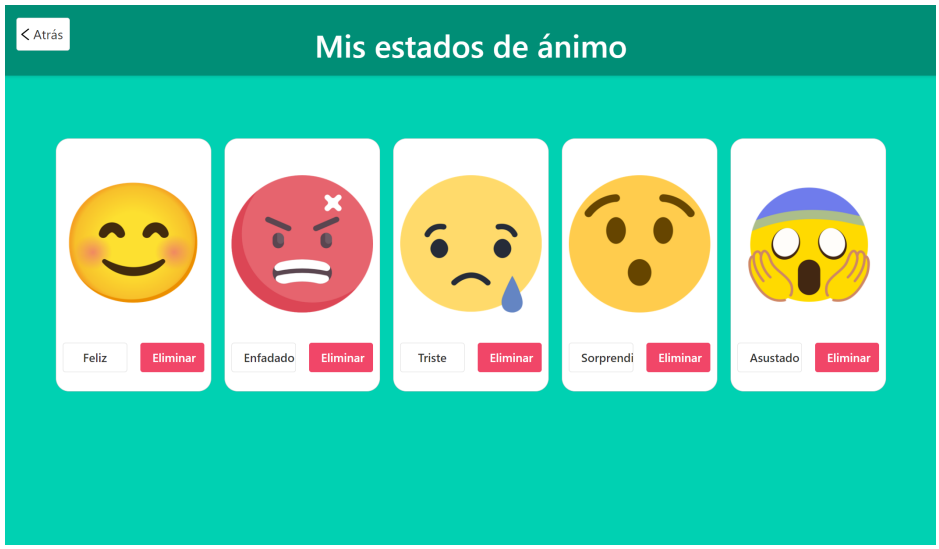

Figura A.16: Lista de los estados de ánimo.

Cada estado de ánimo está representado en una tarjeta donde se ve muestra su imagen y su nombre. Si el docente pulsa el botón donde figura el nombre del estado de ánimo, accederá a la página donde se muestran los detalles del estado de ánimo. Los detalles del estado *Feliz* se pueden ver en la Figura A.17.

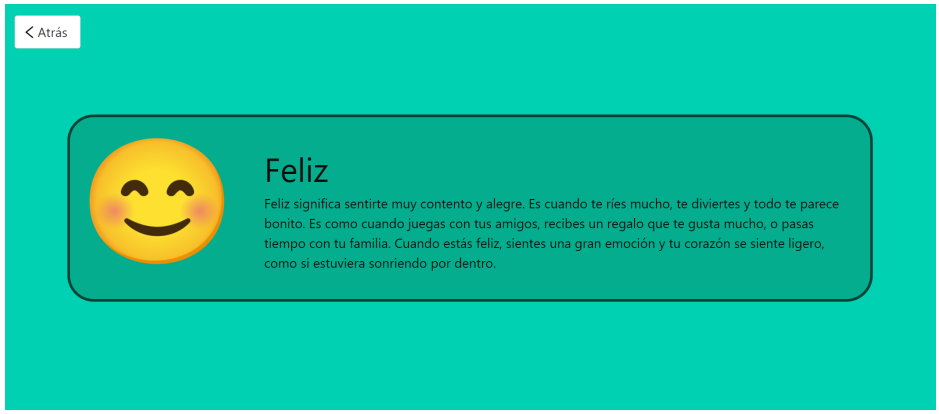

Figura A.17: Detalles de un estado de ánimo.

En la lista de estados de ánimo de la Figura A.16 también se puede ver que hay un botón *Eliminar* en cada uno de los estados de ánimo. Si el docente pulsa este botón, y confirma la acción, se eliminará el estado de ánimo correspondiente.

Volviendo al menú de la Figura A.14, si el docente presiona la tarjeta *Estados de estudiantes*, se mostrará el estado de ánimo actual de todos los estudiantes a los que el docente

imparte clase, también se mostrará el momento en el que se registraron como dicho estado de ánimo. Esto se puede ver en la Figura A.18.

Figura A.18: Estados actuales de los estudiantes.

Al final de la fila de cada estudiante, el docente tiene acceso al botón *Ver estados* que está habilitado solo si estudiante ha registrado como se siente en algún momento. Si el docente presiona este botón, se accede al diario emocional del estudiante, como se puede ver en la Figura A.19.

Figura A.19: Diario emocional de un estudiante.

En el diario emocional de cada estudiante se muestra su historial de estados de ánimo

ordenado cronológicamente. De cada estado que haya registrado el estudiante, se muestra el nombre, el instante de tiempo en el que se registró y la razón por la cual se siente así.

A.1.2. Estudiante

Después de iniciar sesión como estudiante, se muestra un menú o pantalla principal como se puede ver en la Figura A.20. En el menú se tienen dos tarjetas distintas: *Mis actividades* y *Mi diario emocional*.

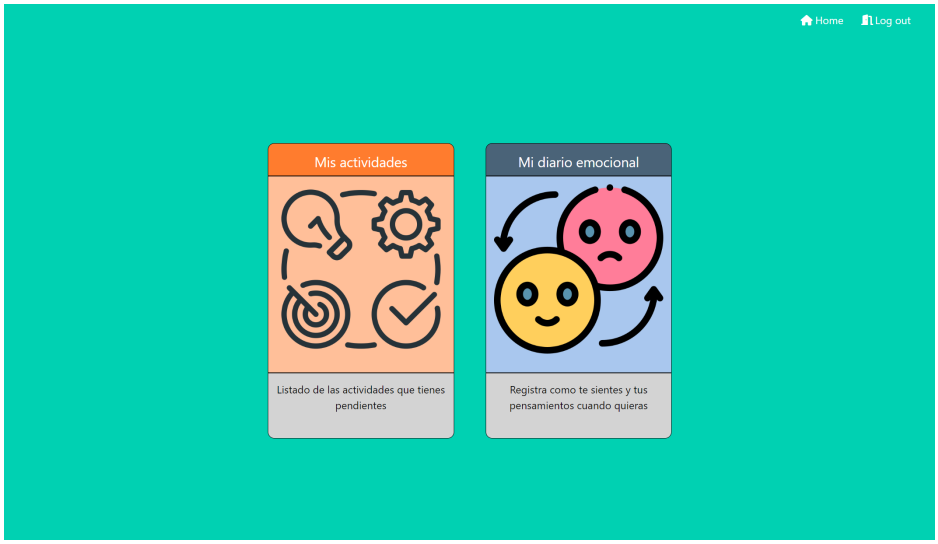

Figura A.20: Menú del estudiante.

Si el estudiante pulsa en la tarjeta *Mis actividades*, se le mostrará la lista de actividades que se le han sido asignadas y aún no ha respondido o finalizado. Al igual que los docentes, tienen la posibilidad de realizar una búsqueda por nombre de las actividades mediante la barra de búsqueda de la esquina superior derecha. La lista de actividades que se muestra en la Figura A.21 es la misma que la mostrada a los docentes en la Figura A.7, solo que el estudiante solo tiene dos permisos distintos sobre cada actividad.

Si el estudiante presiona el botón *Ver detalles*, se mostrarán en la pantalla (Figura A.22) los detalles de la actividad igual que se le mostrarían al docente. Además, se muestra el botón *Resolver actividad* que si es pulsado por el estudiante, este será redirigido a la pantalla que permite resolver esa actividad como se muestra en la Figura A.23.

Figura A.21: Lista de Actividades del Estudiante.

Figura A.22: Detalles de una actividad.

Al pasar a resolver una actividad, se le muestran los detalles de la actividad y las preguntas asociadas debajo de estos, en caso de que haya. El estudiante entonces, en caso de que haya preguntas, las responde y presiona el botón *Enviar respuestas* para finalizar la asignación.

Figura A.23: Resolver una actividad.

En caso de que el estudiante no quiera ver los detalles de la actividad para pasar a resolver la asignación, puede pulsar el botón *Resolver* en la tarjeta de la actividad asignada que quiera resolver.

Si en el menú principal el estudiante presiona la tarjeta *Mi diario emocional*, es redirigido a la pantalla mostrada en la Figura A.24.

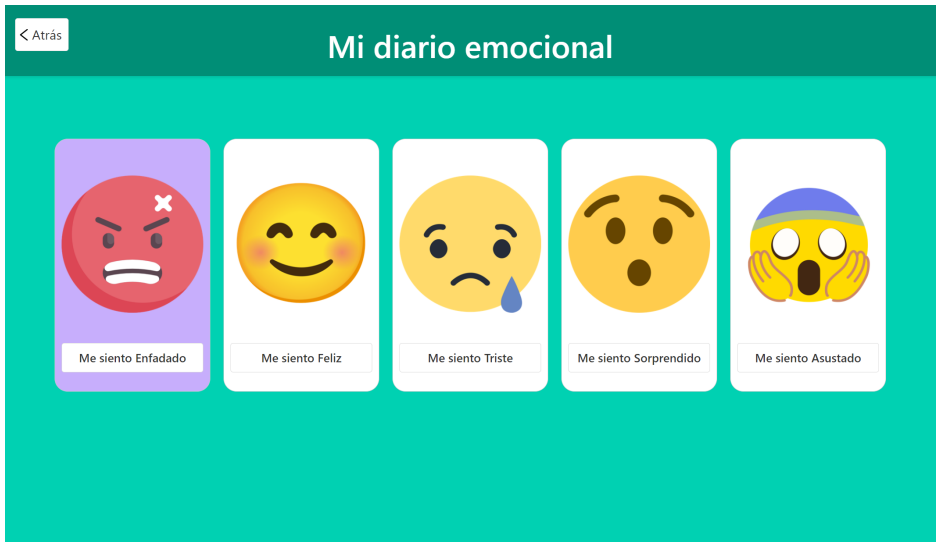

Figura A.24: Lista de Estados de Ánimo.

En esta página se muestran todos los estados de ánimo registrados en el sistema. El estado actual del estudiante siempre figurará el primero de la lista y estará resaltado en un color violeta como se puede ver en la Figura A.24. El estudiante puede presionar el botón *Me siento <estado>* para acceder a la pantalla mostrada en la Figura A.25.

Figura A.25: Registrar un estado de estudiante.

Finalmente, en esta pantalla se le muestran los detalles del estado de ánimo al estudiante y se solicita indicar por qué se siente así. Una vez el estudiante introduce la razón y presiona el botón *Registrar mi estado de ánimo*, se registra el estado del estudiante en el sistema y se redirige al estudiante a la lista de estados de ánimo.

Apéndice B

Resumen de enlaces adicionales

Los enlaces útiles de interés en este Trabajo Fin de Grado son:

- Repositorio del código: https://gitlab.inf.uva.es/davmorc/tfg_informatica.

Bibliografía

[1] Universidad de Valladolid. Proyecto docente del trabajo de fin de grado 2023-2024 (Mención Computación). https://apps.stic.uva.es/guias_docentes/uploads/2023/545/46978/1/Documento.pdf. Accedido: 28/02/2024.

[2] TT Inkscape. Inkscape. https://inkscape.org/es/. Accedido: 28/02/2024.

[3] Visual Paradigm International. Visual Paradigm. https://www.visual-paradigm.com/. Accedido: 28/02/2024.

[4] Overleaf. Overleaf, editor de latex online. https://es.overleaf.com/. Accedido: 15/02/2024.

[5] The project website. https://www.latex-project.org/. Accedido: 15/02/2024.

[6] Boxy SVG. Boxy svg - un potente editor de svg. https://boxy-svg.com/. Accedido: 10/04/2024.

[7] Python Software Foundation. Python. https://www.python.org/. Accedido: 15/02/2024.

[8] Flask. Flask, framework para desarrollo de aplicaciones web. https://flask.palletsprojects.com/. Accedido: 15/02/2024.

[9] Armin Ronacher. Jinja2 documentation. https://jinja.palletsprojects.com/en/3.1.x/, 2023. Accedido: 15/02/2024.

[10] Django Software Foundation. The django template language. https://docs.djangoproject.com/en/stable/ref/templates/language/, 2023. Accedido: 11/05/2024.

[11] World Wide Web Consortium (W3C). Cascading style sheets (css). https://www.w3.org/Style/CSS/. Accedido: 15/02/2024.

[12] Douglas Crockford. *JavaScript: The Good Parts*. O'Reilly Media, 2008. Consultado: 26/05/2024.

[13] Scrum.org. What is scrum? https://www.scrum.org/resources/what-scrum-module. Accedido: 20/02/2024.

[14] DeepL GmbH. Deepl translator. https://www.deepl.com/es/translator. Accedido: 07/04/2024.

[15] Atlassian. Trello. https://trello.com/. Accedido: 14/02/2024.

[16] Microsoft. Visual studio code. https://code.visualstudio.com/. Accedido: 14/06/2024.

[17] Bulma. Un potente framework css para un diseño web moderno. https://bulma.io/. Accedido: 15/02/2024.

[18] Font Awesome. Un conjunto robusto de íconos vectoriales para tus proyectos. https://fontawesome.com/. Accedido: 20/02/2024.

[19] Microsoft Corporation. Microsoft outlook. https://outlook.office.com/. Accedido: 14/02/2024.

[20] Microsoft Corporation. Microsoft teams. https://teams.microsoft.com/. Accedido: 18/02/2024.

[21] Postman. Una herramienta versátil para probar y desarrollar apis. https://www.postman.com/. Accedido: 26/02/2024.

[22] Pallets Projects. Flask debugger. https://flask.palletsprojects.com/en/2.1.x/debugging/. Accedido: 14/02/2024.

[23] World Wide Web Consortium. *HTML Living Standard*. Accedido: 06/05/2024.

[24] Ubuntu: The leading operating system for pcs, iot devices, servers and the cloud. https://ubuntu.com/, 2024. Canonical Ltd.

[25] Visual Paradigm. What is package diagram? https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-package-diagram/. Accedido: 28/04/2024.

[26] Statista. Global digital population as of april 2024. https://www.statista.com/statistics/617136/digital-population-worldwide/, 2024. Accedido: 28/03/2024.

[27] Glassdoor. Salario de un desarrollador full stack en españa. https://www.glassdoor.com/Salaries/spain-full-stack-developer-salary-SRCH_IL.0,5_IN219_KO6,26.htm. Accedido: 01/03/2024.

[28] Seguridad Social. Cotización y recaudación trabajadores. https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537#36538. Accedido: 01/03/2024.

[29] Banco Santander. Calculadora de irpf. https://www.bancosantander.es/particulares/cuentas-tarjetas/cuentas-corrientes/calculadora-irpf. Accedido: 01/03/2024.

[30] Leburo Cowork. Precio coworking valladolid. https://leburocowork.es/precio-coworking-valladolid. Accedido: 01/03/2024.

[31] Dell. Xps 15 laptop. https://www.dell.com/es-es/shop/porttiles-de-dell/porttil-xps-15/spd/xps-15-9530-laptop. Accedido: 01/03/2024.

[32] GreenCodeTech810. Average laptop lifespan by brand. https://medium.com/@greencodetech810/average-laptop-lifespan-by-brand-128c11394dd3. Accedido: 01/03/2024.

[33] Balsamiq. Licencia de balsamiq. https://balsamiq.com/buy/. Accedido: 01/03/2024.

[34] Visual Paradigm. Licencia anual de visual paradigm. https://www.visual-paradigm.com/shop/vp.jsp. Accedido: 01/03/2024.

[35] Microsoft. Microsoft office. https://www.office.com/. Accedido: 01/03/2024.

[36] Huawei. Especificaciones de huawei matebook 13. https://consumer.huawei.com/sg/laptops/matebook-13/specs/. Accedido: 02/03/2024.

[37] Tarifa Luz Hora. Precio kwh. https://tarifaluzhora.es/info/precio-kwh. Accedido: 02/03/2024.

[38] Energy Use Calculator. Uso de electricidad de un portátil. https://energyusecalculator.com/electricity_laptop.htm, 2024. Accedido: 02/03/2024.

[39] Pew Research Center. U.s. generations technology use. https://www.pewresearch.org/short-reads/2019/09/09/us-generations-technology-use/. Accedido: 29/03/2024.

[40] Kawsar Salam, Dhayendre Moonasar, Patricia R Hunter, and Babatunde Olowokure. Mitigating risk factors for foodborne illnesses: A comprehensive review of proactive measures and emerging concerns. *International Journal of Environmental Research and Public Health*, 20(11):4330, 2023. Consultado: 29/03/2024.

[41] NewYork-Presbyterian. What does too much screen time do to children's brains? https://healthmatters.nyp.org/what-does-too-much-screen-time-do-to-childrens-brains/, 2023. Accedido: 29/03/2024.

[42] Mike Watson. *Managing Smaller Projects: A Practical Approach*. Management Concepts, 2006. Consultado: 29/03/2024.

[43] Atlassian. Scrum. https://www.atlassian.com/agile/scrum. Accedido: 31/03/2024.

[44] Agile Alliance. The agile manifesto. https://www.agilealliance.org/agile101/the-agile-manifesto/. Accedido: 01/03/2024.

[45] Principles behind the agile manifesto. https://agilemanifesto.org/principles.html. Accedido: 01/03/2024.

[46] ScrumVersity. Scrum characteristics. https://www.scrumversity.org/scrum-characteristics. Accedido: 04/03/2024.

[47] Ken Schwaber and Jeff Sutherland. The scrum guide. Accedido: 03/03/2024.

[48] Ian Sommerville. *Software Engineering*. Addison-Wesley, 7th edition, 2005. Consultado: 17/04/2024.

[49] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, Boston, MA, 2002. Consultado: 12/05/2024.

[50] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, 2002. Consultado: 12/05/2024.

[51] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison Wesley, 2005. Consultado: 05/04/2024.

[52] Guido van Rossum, Barry Warsaw, and Nick Coghlan. Pep 8 – style guide for python code. Technical report, Python Software Foundation. Consultado: 15/03/2024.

[53] Python Software Foundation. Python 3.10.10. https://www.python.org/downloads/release/python-31010/. Accedido: 18/05/2024.

[54] Oracle Corporation. *MySQL Connector/Python Developer Guide*. Oracle Corporation, 2024.

[55] MySQL. Mysql server naming organization guidelines. https://dev.mysql.com/doc/dev/mysql-server/latest/PAGE_NAMING_ORGANIZATION_GUIDELINES.html. Accedido: 20/05/2024.

[56] R.P. Higuera and Y.Y. Haimes. *Software Risk Management*. Technical report (Carnegie Mellon University. Software Engineering Institute). Carnegie Mellon University, Software Engineering Institute. Consultado: 10/03/2024.

[57] ProjectManager.com. How to mitigate risk in a project: Tips and strategies. https://www.projectmanager.com/blog/mitigate-risk-project. Accedido: 10/03/2024.

[58] Open Textbook Library. Risk management planning.

[59] Community. How to fix table position? https://tex.stackexchange.com/questions/9485/how-to-fix-table-position.

[60] Overleaf. Tables.

[61] David Carlisle. The tabularx package documentation. https://texdoc.org/serve/tabularx/0.

[62] User. Background color for rows and columns of a table, 2017.

[63] Project Management Institute. Project management institute. Accedido: 28/02/2024.

[64] PMI Standards Committee and Project Management Institute. *A Guide to the Project Management Body of Knowledge*. PMBOK guide. Project Management Institute, 1996. Consultado: 01/03/2024.

[65] Community of TeX Stack Exchange. How to use multirow? https://tex.stackexchange.com/questions/308667/how-to-use-multirow.

[66] LaTeX.org. Table keeps appearing bottom of page. https://latex.org/forum/viewtopic.php?t=1212.

[67] Hassan Gomaa. *Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures*. Cambridge University Press, 2011. Consultado: 24/04/2024.

[68] Overleaf. A longtable example. https://es.overleaf.com/latex/examples/a-longtable-example/xxwzfxkxxjmc. Consultado: 23/03/2024.

[69] Innolution. Essential scrum - chapter 5: Requirements and user stories. https://innolution.com/essential-scrum/table-of-contents/chapter-5-requirements-and-user-stories#:~:text=In%20place%20of%20detailed%20upfront,item%20represents%20desirable%20business%20value. Consultado: 20/03/2024.

[70] IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, pages 1–84, 1990. Consultado: 22/03/2024.

[71] LogRocket Blog. The fibonacci sequence and story points in agile: A cautionary tale. https://blog.logrocket.com/product-management/fibonacci-sequence-story-points-agile/#:~:text=Story%2020%2040%20100, 2023. Accedido: 03/03/2024.

[72] Tess Gadd. Table and data grid guidelines. https://balsamiq.com/learn/ui-control-guidelines/tables-data-grids/, 2024. Accedido: 23/02/2024.

[73] Stack Exchange Community. How to include svg diagrams in latex? https://tex.stackexchange.com/questions/2099/how-to-include-svg-diagrams-in-latex. Accedido: 25/04/2024.

[74] TeX Stack Exchange Community. How to present a Python code snippet efficiently in LaTeX. https://tex.stackexchange.com/questions/475826/how-to-present-a-python-code-snippet-efficiently-in-latex, 2019. Accedido: 2024-06-07.

[75] Stack Overflow. Specify routes in flask. https://stackoverflow.com/questions/29182592/specify-routes-in-flask. Accedido: 20/02/2024.

[76] GitHub. Intro to flask. https://github.com/lalithpolepeddi/intro-to-flask/tree/master/intro_to_flask. Accedido: 21/02/2024.

[77] Stack Overflow. Flask download file not working. https://stackoverflow.com/questions/44482913/flask-download-file-not-working, 2017. Fecha de acceso: 14/03/2024.

[78] Tex Exchange Community. Displaying linux commands in . https://tex.stackexchange.com/questions/84185/displaying-linux-commands-in-latex, 2012. Accedido: 07/06/2024.

[79] EFF CERTBOT. Using certbot with nginx. https://eff-certbot.readthedocs.io/en/stable/using.html#nginx. Accedido: 06/06/2024.

[80] Stack Overflow. E: Package 'python-certbot-nginx' has no installation candidate. https://stackoverflow.com/questions/64571233/e-package-python-certbot-nginx-has-no-installation-candidate. Accedido: 06/06/2024.

[81] Nginx. https://nginx.org/en/, 2024. Accedido: 28/04/2024.

[82] Ibm software testing. https://www.ibm.com/topics/software-testing. Accedido: 01/06/2024.

[83] Javatpoint. Levels of testing. https://www.javatpoint.com/levels-of-testing. Accedido: 01/06/2024.

[84] ISO/IEC/IEEE. *Systems and software engineering – Vocabulary*. ISO/IEC/IEEE, 2010(e) edition, 12 2010.

[85] Departamento de Psicología de Valladolid. Diseño y validación de un programa de acogida dirigido al alumnado refugiado procedente de ucrania escolarizado en la enseñanza básica. Memoria de proyecto de Acogecyl, 2022. Consultado: 08/03/2024.

[86] Elina E. Ketonen, Julia Dietrich, Julia Moeller, Katariina Salmela-Aro, and Kirsti Lonka. The role of daily autonomous and controlled educational goals in students' academic emotion states: An experience sampling method approach. *Learning and Instruction*, 53:10–20, 2018.

[87] Microsoft. Microsoft corporation. https://www.microsoft.com, 2024. Accedido: 01/06/2024.

[88] Microsoft. Python environment manager. https://marketplace.visualstudio.com/items?itemName=ms-python.venv. Accedido: 01/06/2024.

[89] Microsoft. Python debugger. https://marketplace.visualstudio.com/items?itemName=ms-python.python, 2024. Accedido: 01/06/2024.

[90] Microsoft. Pylance. https://marketplace.visualstudio.com/items?itemName=ms-python.vscode-pylance, 2024. Accedido: 01/06/2024.

[91] Kevin Rose. Python indent. https://marketplace.visualstudio.com/items?itemName=kevinrose.vsc-python-indent, 2024. Accedido: 01/06/2024.

[92] Python Software Foundation. uuid — uuid objects according to rfc 4122. https://docs.python.org/3/library/uuid.html, 2024. Accedido: 05/06/2024.

[93] Prettier. Prettier. https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode, 2024. Accedido: 01/06/2024.

[94] GitLab. Gitlab. https://about.gitlab.com/, 2024. Accedido: 01/06/2024.

[95] Universidad de Valladolid. Gitlab, 2024. Accedido: 01/06/2024.

[96] Linus Torvalds. Git. https://git-scm.com/, 2005. Accedido: 01/06/2024.

[97] Stack Overflow. Flask Bcrypt ValueError: Invalid salt. https://stackoverflow.com/questions/34548846/flask-bcrypt-valueerror-invalid-salt. Accedido: 03/06/2024.

[98] Stack Overflow. ValueError: Invalid salt in Flask. https://stackoverflow.com/questions/68037750/valueerror-invalid-salt-in-flask. Accedido: 03/06/2024.

[99] Spring Framework Documentation. Spring Security BCrypt Documentation. https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/crypto/bcrypt/BCrypt.html. Accedido: 03/06/2024.

[100] Stack Overflow. bcrypt.checkpw() returns TypeError: Unicode objects must be encoded before checking. https://stackoverflow.com/questions/40577867/bcrypt-checkpw-returns-typeerror-unicode-objects-must-be-encoded-before-checkin. Accedido: 03/06/2024.

[101] Stack Overflow. Dynamically increase input type text textbox width according to the characters entered. https://stackoverflow.com/questions/20727692/dynamically-increase-input-type-text-textbox-width-according-to-the-characters-e. Accedido: 03/06/2024.

[102] 4Geeks Academy Español. Crea tu propia app con Python y Flask. https://www.youtube.com/watch?v=ZWdagmCMTuo. Accedido: 24/02/2024.

[103] Stack Overflow. Storing exam questions in a database. https://stackoverflow.com/questions/23677003/storing-exam-questions-in-a-database, 2014. Accedido: 05/03/2024.

[104] Stack Overflow. Current date (CURDATE) not working as default date value. https://stackoverflow.com/questions/20461030/current-date-curdate-not-working-as-default-date-value, 2013. Accedido: 05/03/2024.

[105] Stack Overflow. Remove scrollbars from textarea. https://stackoverflow.com/questions/19424887/remove-scrollbars-from-textarea, 2013. Accedido: 06/03/2024.

[106] Stack Overflow. How do I add more members to my enum type column in MySQL? https://stackoverflow.com/questions/1501958/how-do-i-add-more-members-to-my-enum-type-column-in-mysql, 2009. Accedido: 06/03/2024.

[107] Python Documentation. datetime — Basic date and time types. https://docs.python.org/3/library/datetime.html, current. Accedido: 07/03/2024.

[108] Flask Documentation. File Uploads in Flask. https://flask.palletsprojects.com/en/2.3.x/patterns/fileuploads/, current. Accedido: 07/03/2024.

[109] Flask Documentation. Static Files in Flask. https://flask.palletsprojects.com/en/2.3.x/tutorial/static/, current. Accedido: 06/03/2024.

[110] DevPress. Unable to Retrieve Files from Send from Directory in Flask. https://devpress.csdn.net/python/6304cbc3c67703293080e041.html, current. Accedido: 13/03/2024.

[111] Stack Overflow. Unable to Retrieve Files from send_from_directory in Flask. https://stackoverflow.com/questions/17681762/unable-to-retrieve-files-from-send-from-directory-in-flask, 2013. Accedido: 13/03/2024.

[112] Babel Documentation. Babel Dates and Times. https://babel.pocoo.org/en/latest/api/dates.html, current. Accedido: 17/03/2024.

[113] Stack Overflow. audio tag GUI not visible. https://stackoverflow.com/questions/11968309/audio-tag-gui-not-visible. Accedido: 17/03/2024.

[114] CSS Scan. CSS Checkbox Examples. https://getcssscan.com/css-checkboxes-examples, current. Accedido: 24/03/2024.

[115] Stack Overflow. Uncaught ReferenceError: checkbox is not defined. https://stackoverflow.com/questions/75568580/uncaught-referenceerror-checkbox-is-not-defined. Accedido: 23/03/2024.

[116] Django Forum. Rename uploaded image based on the field that it's uploaded from. https://forum.djangoproject.com/t/rename-uploaded-image-based-on-the-field-that-its-uploaded-from/20844. Accedido: 24/03/2024.

[117] Stack Overflow. How to rename upload image while uploading (Flask, Python). https://stackoverflow.com/questions/49900621/how-to-rename-upload-image-while-uploading-flask-python. Accedido: 24/03/2024.

[118] Stack Overflow. Some of Font Awesome's icons are not working and some appear as an empty square. https://stackoverflow.com/questions/59515847/some-of-font-awesomes-icons-are-not-working-and-some-appears-as-an-empty-square. Accedido: 23/03/2024.

[119] Stack Overflow. Stick a div to the right border of a div defined later. https://stackoverflow.com/questions/11565090/stick-a-div-to-the-right-border-of-a-div-defined-later. Accedido: 24/03/2024.

[120] Stack Overflow. How to position an element to the right side. https://stackoverflow.com/questions/19263524/how-to-position-an-element-to-the-right-side. Accedido: 24/03/2024.

[121] Flask-Resize Documentation. Flask-Resize Documentation. https://flask-resize.readthedocs.io/. Accedido: 26/03/2024.

[122] Stack Overflow. Resize an uploaded/requested image by Flask. https://stackoverflow.com/questions/70394135/resize-an-uploaded-requested-image-by-flask. Accedido: 26/03/2024.

[123] Stack Overflow. No module named PIL (Visual Studio Code error). https://stackoverflow.com/questions/70795319/no-module-named-pil-visual-studio-code-error. Accedido: 26/03/2024.

[124] Fredrik Lundh. Python Imaging Library (PIL). https://python-pillow.org/. Accedido: 26/03/2024.

[125] Stack Overflow. Converting 'True' JSON to Python equivalent (True). https://stackoverflow.com/questions/33722662/converting-true-json-to-python-equivalent-true. Accedido: 02/04/2024.

[126] W3Schools. Python String replace(). https://www.w3schools.com/python/ref_string_replace.asp. Accedido: 02/04/2024.

[127] Stack Overflow. Overflow to left instead of right. https://stackoverflow.com/questions/218065/overflow-to-left-instead-of-right. Accedido: 01/04/2024.

[128] Stack Overflow. How can I prevent overflowing button? https://stackoverflow.com/questions/70435905/how-can-i-prevent-overflowing-button. Accedido: 01/04/2024.

[129] Stack Overflow. How do I add textboxes dynamically in JavaScript? https://stackoverflow.com/questions/1390319/how-do-i-add-textboxes-dynamically-in-javascript. Accedido: 10/04/2024.

[130] W3Schools. HTML Tables. https://www.w3schools.com/html/html_tables.asp, current. Accedido: 14/04/2024.

[131] SitePoint Community. Centering Image Inside Flexbox. https://www.sitepoint.com/community/t/centering-image-inside-flexbox/287444, 2020. Accedido: 21/04/2024.

[132] MySQL Documentation. MySQL 8.4 Reference Manual: Date and Time Functions. https://dev.mysql.com/doc/refman/8.4/en/date-and-time-functions.html. Accedido: 20/04/2024.

[133] Stack Overflow. How to prevent buttons from submitting forms? https://stackoverflow.com/questions/932653/how-to-prevent-buttons-from-submitting-forms. Accedido: 21/04/2024.

[134] Stack Overflow. Can we append to a block rather than overwrite? https://stackoverflow.com/questions/1724466/can-we-append-to-a-block-rather-than-overwrite. Accedido: 22/04/2024.

[135] Stack Overflow. Extend block content / endblock in an html page does not work. https://stackoverflow.com/questions/71678310/extend-block-content-endblock-in-an-html-page-does-not-work. Accedido: 22/04/2024.

[136] Stack Overflow. Flask confirm action. https://stackoverflow.com/questions/32911578/flask-confirm-action. Accedido: 18/05/2024.

[137] Stack Overflow. How to add confirmation dialog for post request with flask. https://stackoverflow.com/questions/52752734/how-to-add-confirmation-dialog-for-post-request-with-flask. Accedido: 18/05/2024.

[138] Flask Documentation. Flashing. https://flask.palletsprojects.com/es/main/patterns/flashing/. Accedido: 20/05/2024.