ULPGC Universidad de Las Palmas de Gran Canaria | Escuela de Ingeniería Informática

Aplicación web para promover y facilitar herramientas para personas que sufren ansiedad social

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Joaquín Javier Pastore Barrios

TUTORIZADO POR:

Francisco Alexis Quesada Arencibia

Agradecimientos

A mi madre, pareja y amigos, por siempre orientarme y ayudarme a no desistir.

A mi mentor Alexis Quesada Arencibia, por su guía, apoyo, esfuerzo y compromiso en el desarrollo de este proyecto.

Resumen

Este trabajo de fin de carrera presenta una solución innovadora para combatir la ansiedad social, una problemática que ha sido agravada por la pandemia y sus consecuencias. A través de investigaciones exhaustivas, reuniones con profesionales de la salud mental y consultorías con usuarios afectados, se ha logrado entender mejor el problema y desarrollar una herramienta efectiva para ayudar a las personas que sufren de ansiedad social.

La solución consiste en una aplicación web escalable y versátil que ofrece a los usuarios herramientas y recursos para afrontar su trastorno de manera efectiva. La aplicación ha sido diseñada con tecnologías novedosas y una interfaz amigable e intuitiva para asegurar una experiencia de usuario óptima.

En resumen, este trabajo ha culminado en una solución completa y efectiva para combatir la ansiedad social, que puede ser accesible para un gran número de personas. Gracias a una sólida investigación y la colaboración con expertos y usuarios afectados, se ha logrado desarrollar una herramienta útil y de gran impacto en la salud mental de las personas.

Abstract

This final project presents an innovative solution to combat social anxiety, a problem that has been aggravated by the pandemic and its consequences. Through exhaustive research, meetings with mental health professionals, and consultations with affected users, a better understanding of the problem has been achieved and an effective tool has been developed to help people who suffer from social anxiety.

The solution consists of a scalable and versatile web application that offers users tools and resources to effectively cope with their disorder. The application has been designed with state-of-the-art technologies and a friendly and intuitive interface to ensure an optimal user experience.

In summary, this project has culminated in a complete and effective solution to combat social anxiety, which can be accessible to a large number of people. Thanks to solid research and collaboration with experts and affected users, a useful tool with a great impact on people's mental health has been developed.

Índice general

Índice de figuras

Índice de tablas

Anexos

Capítulo 1. Introducción.

1.1 Definición del problema

Los problemas de ansiedad son una realidad que afecta a millones de personas en todo el mundo. Estas reacciones emocionales pueden manifestarse en forma de preocupación, miedo excesivo, estrés, trastornos de ansiedad, depresión, fobias, etc. Estos problemas son particularmente comunes en tiempos de incertidumbre, tales como los que se experimentan en la actualidad. La pandemia de COVID-19 ha provocado cambios significativos en la vida diaria de las personas, incluyendo el aislamiento social, el cierre de negocios y el cambio de trabajo. Estas situaciones pueden aumentar los niveles de ansiedad y la inseguridad, especialmente en aquellas personas que ya luchan contra estos problemas. Es importante que las personas busquen herramientas y recursos para ayudarles a superar estos desafíos, como el apoyo de amigos y familiares, las terapias conductuales y los medicamentos. Entre estos tipos de ansiedades e inseguridades se encuentra la ansiedad social.

"La ansiedad social (también conocida como fobia social) es un trastorno de ansiedad que causa miedo, angustia e inquietud en situaciones sociales. Estas circunstancias pueden incluir reuniones, hablar en público, relaciones cara a cara e incluso realizar preguntas en una clase. Las personas con ansiedad social tienen miedo de ser juzgadas por los demás. Esto puede llevar a comportamientos aislados y evitación de situaciones sociales" (National Institute of Mental Health, 2020)[1]. Según un artículo publicado por *PSIQUION*[1], el aumento de apariciones de fobias como por ejemplo la ansiedad social debido a la pandemia, puede atribuirse principalmente a los cambios en el estilo de vida como los que se comentan anteriormente. "*El aislamiento de la pandemia, la reducción de la interacción social y la incertidumbre generalizada sobre el futuro, han causado una gran preocupación en muchas personas. Esto se ha vuelto todavía más preocupante con el aumento de las redes sociales y el uso de la tecnología para conectarse con los demás*" *(Psiquion, 2021)*[2].

[1]Plataforma web diseñada para ayudar a las personas a mejorar su salud mental y bienestar.

Google Trends es una excelente herramienta para observar cómo las tendencias están cambiando en el tiempo *(Google Trends, 2022)*[3], especialmente en temas relacionados con la salud mental. Para esta situación es ideal, puesto que se puede observar como las búsquedas acerca de la ansiedad y la ansiedad social sufren un incremento notable justo en el comienzo de la pandemia y el confinamiento, tal y como se puede observar en las siguientes **figuras 1.1.1 y 1.1.2.**

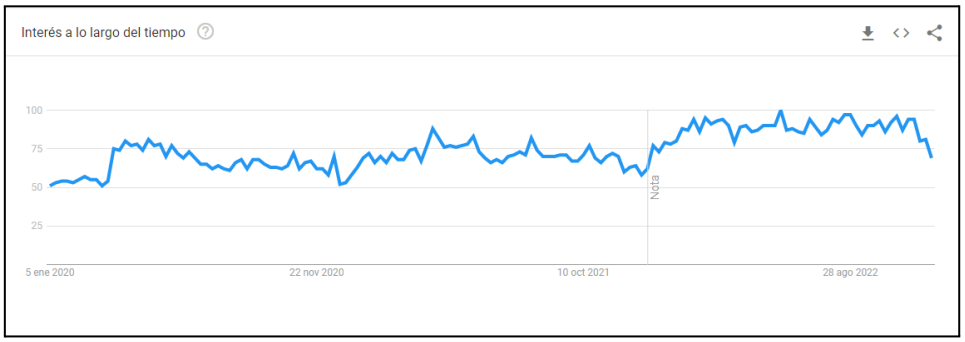


Figura 1.1.1: Interés de la ansiedad desde el año 2020

Fuente: Google Trends

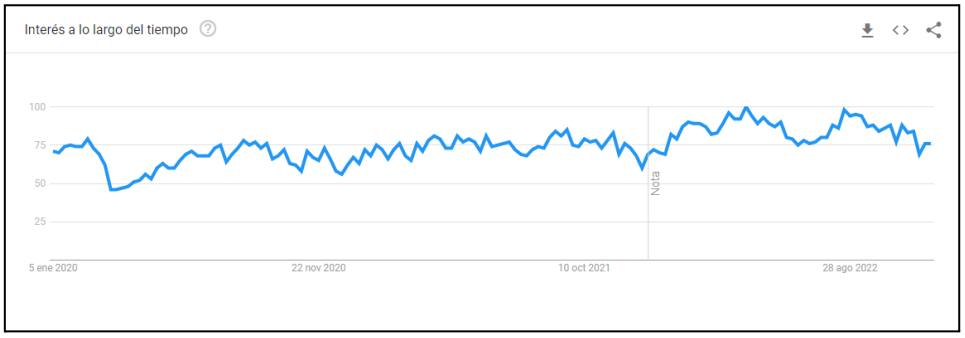


Figura 1.1.2: Interés de la ansiedad social desde el año 2020

Fuente: Google Trends

Durante el año 2021, *Mayo Clinic*[2] [4] realizó un informe en el que se analizaba cómo ciertas situaciones sociales pueden desencadenar sentimientos de ansiedad en personas con trastorno de fobia social. Este fenómeno se observó en encuentros casuales, fiestas o incluso situaciones cotidianas. Situaciones desafiantes que se agravan cuando involucran interactuar con personas con mayor nivel de autoridad, como líderes de negocios, maestros o jefes.

[2]Organización sin fines de lucro, líder mundial en atención médica, investigación y educación.

1.2 Estado del arte

En los últimos años, se han desarrollado diversas herramientas digitales para ayudar a las personas a afrontar los problemas de ansiedad social. Estas herramientas incluyen aplicaciones móviles y sitios web diseñados para proporcionar al usuario información útil acerca de la ansiedad social, así como juegos y otras actividades para ayudar a las personas a desarrollar habilidades para afrontar sus preocupaciones. También pueden proporcionar apoyo a los usuarios a través de foros de discusión, chats grupales y consejeros virtuales. De esta forma, proporcionan recursos útiles para ayudar a las personas a abordar los desafíos de la ansiedad social en un entorno seguro y accesible.

"*Uno de los elementos más angustiantes, y fuente de ansiedad, para las personas con fobia social es la interacción con más de una persona de forma simultánea, así como la comunicación no verbal, cómo sonreír, el sonrojarme, que titubee la voz, sudoración de manos e incluso dentro de la comunicación verbal el poderse quedar bloqueado o decir una tontería" (Blasco R. ,2022)* [5]. Los estudios muestran que el uso de estas herramientas puede ser particularmente útil para aquellas personas con ansiedad social que no se sienten cómodas buscando ayuda profesional. Muchos de estos recursos también ofrecen programas de entrenamiento en habilidades sociales, herramientas para comunicarse con otros usuarios y apoyo emocional.

"La tecnología tiene el potencial de convertirse en una valiosa aliada para cuidar la salud mental. Además de este tipo de apps, existen sistemas de realidad virtual para tratar fobias y otros trastornos y relojes inteligentes que prometen monitorizar el estrés y ayudar a gestionarlo" (Rubio Arroyo I. ,2022)[6]. Algunos sitios web y aplicaciones también están diseñados para ayudar a las personas a encontrar ayuda profesional adecuada. Estos recursos se han demostrado clínicamente eficaces para mejorar los síntomas de la ansiedad social y proporcionar una red de apoyo para aquellos que los necesitan. En la siguiente lista se muestran algunos de los sitios webs con más renombre, además de aplicaciones populares muy bien valoradas por la comunidad *(Spector H. ,2021)*[7].

- **National Institute of Mental Health**[1] ofrece información, recursos y herramientas para ayudar a las personas a entender y comprender la salud mental. El sitio web proporciona varias secciones dedicadas a la ansiedad social, junto con información sobre los signos y síntomas, tratamiento y factores de riesgo. Además, ofrece una sección de preguntas frecuentes sobre la ansiedad social y una lista de recursos adicionales para obtener más información. Asimismo, al ser una plataforma generalizada, suministra una gran cantidad de recursos relacionados con otros tipos de trastornos.

Figura 1.2.1: Visualización del sitio web National Institute of Mental Health

Fuente: National Institute of Mental Health

- **Mayo Clinic**[3] es una organización de atención médica sin fines de lucro. Ofrece un amplio abanico de servicios para la atención de la salud, desde tratamientos clínicos avanzados hasta investigación médica, educación y formación. En su sitio web proporciona información sobre la ansiedad social y otro tipo de trastornos, así como herramientas y recursos para ayudar a las personas a manejar y controlar sus síntomas.

Figura 1.2.2: Visualización del sitio web Anxiety Coach - Mayo Clinic

Fuente: Mayo Clinic

- **Headspace**[8] es una aplicación para IOS diseñada para ayudar a las personas a practicar la meditación y la conciencia. Esta aplicación ofrece herramientas para ayudar a los usuarios a desarrollar una práctica de meditación saludable, a entender mejor los efectos de la meditación y alcanzar un estado mental más equilibrado. También incluye contenido multimedia para ayudar a los usuarios a comprender mejor la práctica de la meditación.

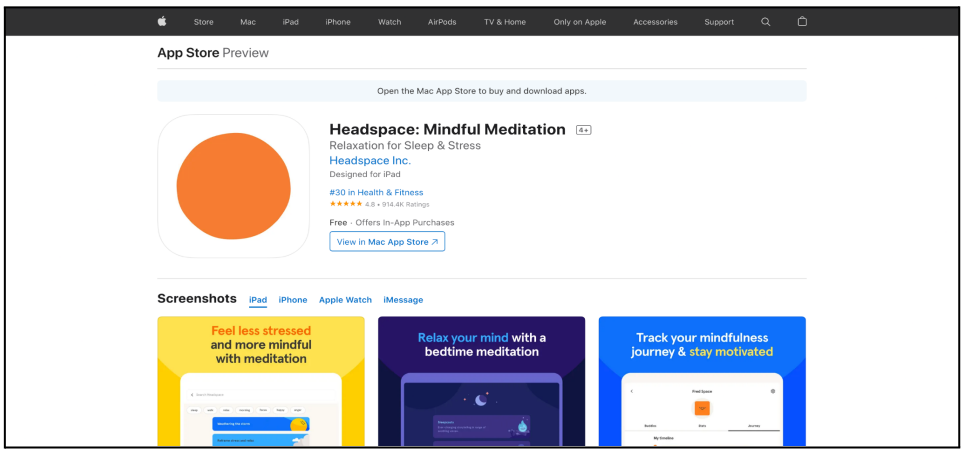


Figura 1.2.3: *App Store Preview* de la aplicación Headspace: Mindful Meditation

Fuente: *App Store*

- **Sanvello: Anxiety & Depression**[9] es una aplicación para IOS diseñada para ayudar a las personas a manejar sus síntomas de ansiedad, estrés y depresión. Esta aplicación ofrece herramientas para ayudar a los usuarios a identificar sus pensamientos y sentimientos, desarrollar habilidades de afrontamiento, establecer límites saludables y conectar con otros para obtener apoyo. Incluye además contenido educativo, herramientas de autoayuda, programas de entrenamiento en habilidades y una herramienta de registro diario para ayudar a los usuarios a comprender mejor sus síntomas.

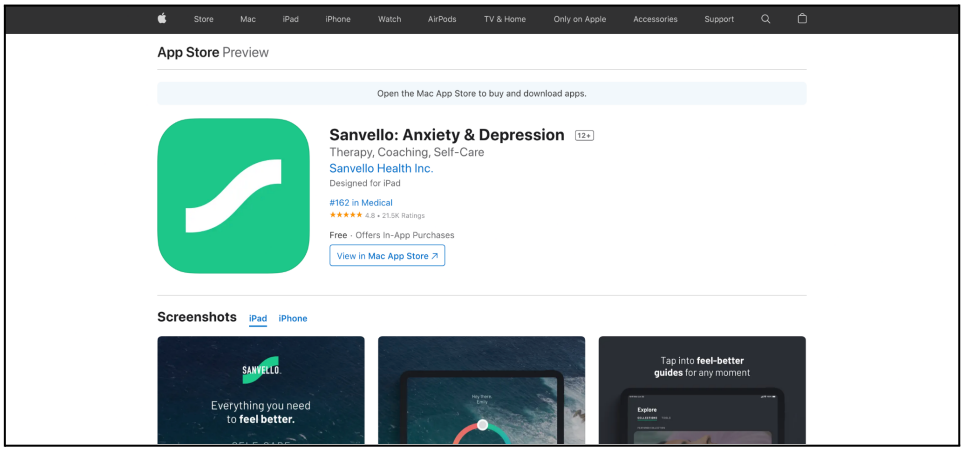


Figura 1.2.4: *App Store Preview* de la aplicación Sanvello: Anxiety & Depression

Fuente: *App Store*

1.3 Solución planteada

Los datos expuestos en el punto *1.1 Definición del problema*, muestran que hay una tendencia creciente en el número de personas con problemas de ansiedad social, lo que significa que hay una gran cantidad de personas que buscan una solución para su problema. Esto refleja la necesidad de una plataforma dedicada a la ansiedad social, que pueda ofrecer recursos para ayudar a aquellas personas a tratar sus problemas de ansiedad social. Por lo tanto, es una excelente oportunidad para aquellos interesados en desarrollar una plataforma especializada en la ansiedad social, que pueda ofrecer servicios de calidad a los usuarios.

Además, la ausencia de una plataforma centrada exclusivamente en la ansiedad social ofrece una posibilidad única para establecer una plataforma con un enfoque

innovador y diferente a la de los demás servicios de asesoramiento mental existentes. Esto permitiría ofrecer una solución novedosa para aquellas personas que sufren dichos problemas, permitiéndoles acceder a recursos que no están disponibles actualmente en otros servicios de asesoramiento mental. Asimismo, al ser una plataforma globalizada, esta podría ofrecer soporte a aquellos usuarios en distintas partes del mundo, aumentando la efectividad de la misma.

De la misma forma que la mayoría de sitios webs y aplicaciones para dispositivos móviles no renuevan su contenido periódicamente, un usuario accede a varios de estos servicios encontrándose constantemente con la misma información de siempre. Siguiendo a la par de esta idea, muchos no disponen de un diseño atractivo que se ajuste a los estándares de calidad de hoy en día, por lo que el atractivo para que el usuario quiera interactuar con la aplicación o el sitio web se pierde. Un diseño innovador y contenido actualizado son elementos cruciales para la eficacia de la aplicación. Según argumenta Valencia en *¿Por qué es importante el diseño web? (Valencia F. ,2021)* [10], "*La gente no confía en los sitios web mal diseñados. Si ven su diseño deficiente o la información parece obsoleta, no confiarán en su sitio. Es posible que vean su sitio como sórdido o turbio porque no tiene un diseño web actualizado*".

Además prosigue con la importancia de la presentación de la web justificando que "*Su sitio web es como un representante de servicio al cliente. Si su sitio web es brillante, moderno y atractivo, su audiencia se sentirá más bienvenida en su página. Dará la impresión de estar abierto y dar la bienvenida a nuevas personas que visiten su sitio web*".

Debido a estos motivos, este proyecto presenta como objetivo precisar y argumentar la necesidad detrás de un sitio web que permita cubrir este problema; que presente todos los medios posibles para que las personas que sufren dicho trastorno tengan un sitio donde acudir con frecuencia para tratar la ansiedad de forma profesional y segura. Esto se pretende lograr a partir de un diseño agradable y minimalista que ofrezca y exhiba contenido actualizado de utilidad y calidad. Dicho contenido debe garantizar la información fidedigna, por lo que es imprescindible que exista una figura experta dentro del sitio web que lo

corrobore y verifique. Con estos requisitos y el objetivo en mente se pretende alcanzar y ayudar a las personas con ansiedad social de forma clara, sencilla y eficaz.

1.4 Asesoramiento experto

Tal y como se argumentaba en el apartado *1.3 Solución planteada*, la ansiedad social se presenta como un asunto delicado que debe tratarse debidamente y de forma profesional. Al tratarse de un aspecto perteneciente a la rama de la psicología, la presencia de una figura experta que permita corroborar el contenido es imprescindible. A día de hoy, la información sobre temas relacionados con la salud mental que se encuentra en línea es inexacta o engañosa. Esto puede conducir a una comprensión errónea de los trastornos mentales, contribuyendo a la desinformación alrededor de ellas y por ende a un efecto negativo en la salud mental de los usuarios.

Según comenta en su informe *(National Library of Medicine, 2021)* [11] , "*al analizar el fenómeno de las fake news en el ámbito de la salud, se pudo observar que el conocimiento infodémico puede causar trastornos psicológicos y provocar pánico, miedo, depresión y fatiga*".

Teniendo en cuenta este estudio, es posible argumentar que la información web debe ser revisada y verificada por un profesional cualificado, como un psicólogo, para garantizar que la información que los usuarios están recibiendo sea precisa y confiable. De este modo, se pueden llevar a cabo reuniones y entrevistas con un profesional para permitir allanar un camino más seguro. Así, los usuarios que visiten la aplicación web podrán saber que toda la información y los recursos disponibles que pueden encontrar dentro del sitio web han sido validados por un capacitado en el ámbito de la psicología y que no se exhiben metodologías y datos de dudosa verificación.

1.5 Peticiones de la comunidad

Con la finalidad de precisar las necesidades de los usuarios que batallan con esta condición día tras día**,** se realizó un previo análisis sobre los posts más comentados de los *subreddits*[3] *r/mentalhealth y r/socialanxiety* en la red social *Reddit*[4]**,** observando los principales temas de los que se discuten, así como las soluciones que aportan los usuarios a los problemas que plantean los demás. Aprovechando la gran interactividad que existe en la red social, sobre todo referentes a estos temas, se procedió a generar un post en ambos *subreddits*, exhibiendo la finalidad de este proyecto y consultando a los usuarios qué tipo de funcionalidad les gustaría que existiera en un servicio web como el que se plantea.

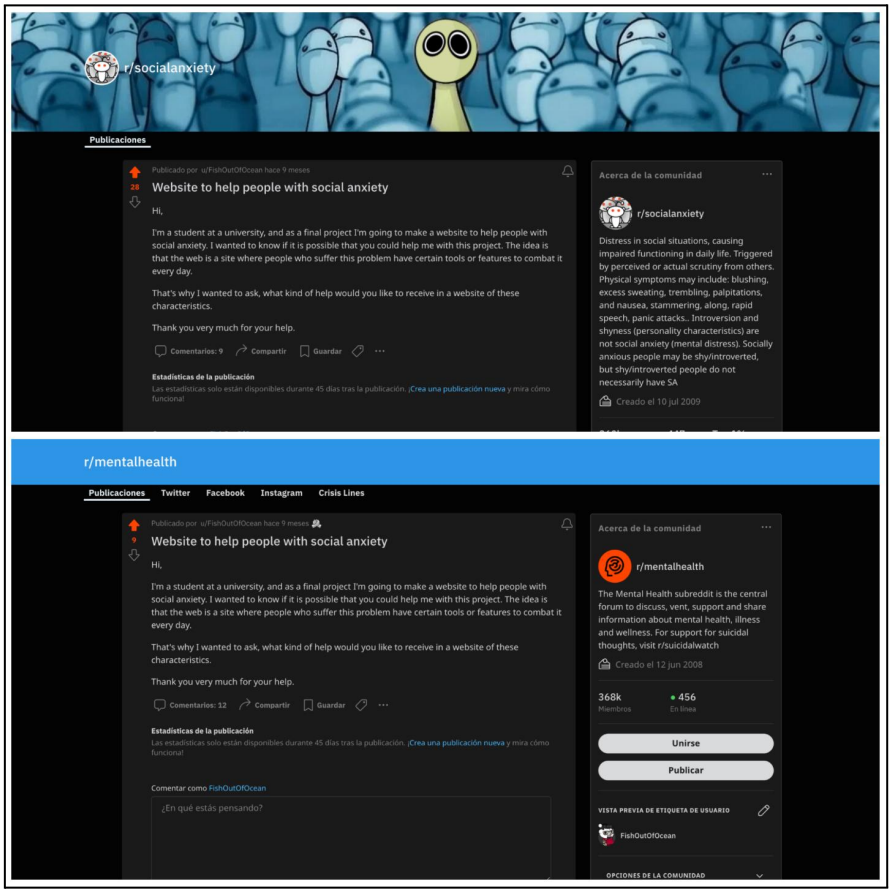


Figura 1.5.1: Publicaciones en los subreddits *r/socialanxiety y r/metalhealth*

Fuente: Reddit

[3]Sección de la página web Reddit, que se dedica a un tema específico.
[4]Plataforma de marcadores sociales y foros de discusión. Está diseñada para permitir que los usuarios compartan contenido y discutan temas entre ellos.

Gracias al apoyo de los usuarios de *Reddit*, se pudo comprender las necesidades que se deben afrontar en el servicio, haciendo especial énfasis en ofrecer ayudar con los ataques de pánico y ansiedad, ejercicios de meditación, técnicas de respiración, un diario personal, recomendaciones de libros, series, películas y videojuegos, así como la posibilidad de que exista un rol de experto que pueda certificar y aprobar las secciones del sitio web.

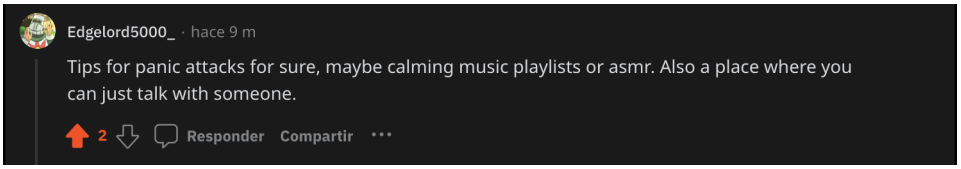


Figura 1.5.2: Comentario de usuario en el subreddit *r/mentalhealth*

Fuente: Reddit

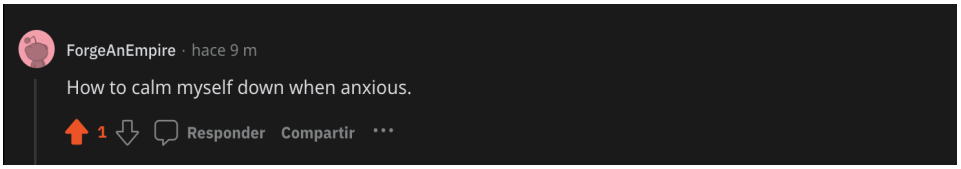


Figura 1.5.3: Comentario de usuario [2] en el subreddit *r/mentalhealth*

Fuente: Reddit

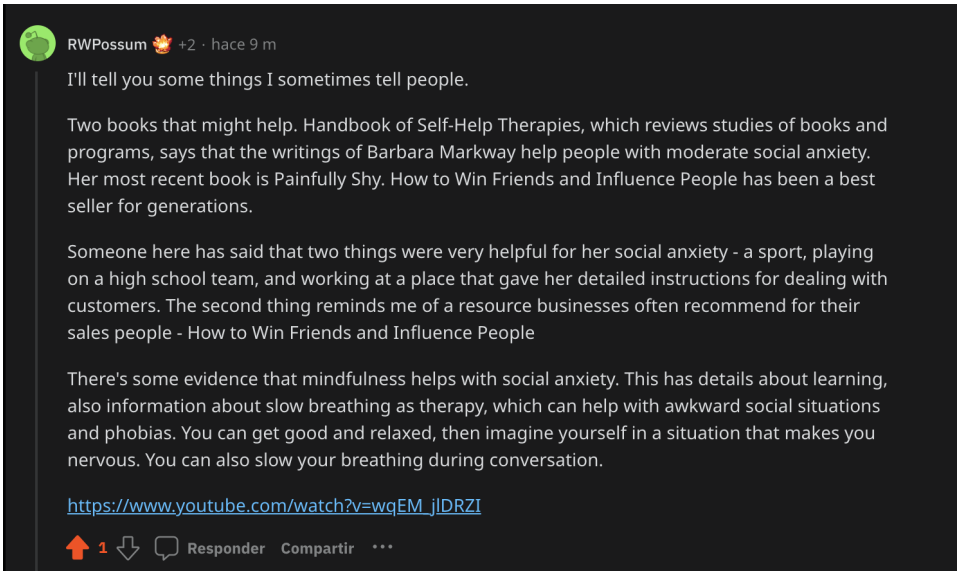


Figura 1.5.4: Comentario de usuario [3] en el subreddit *r/mentalhealth*

Fuente: Reddit

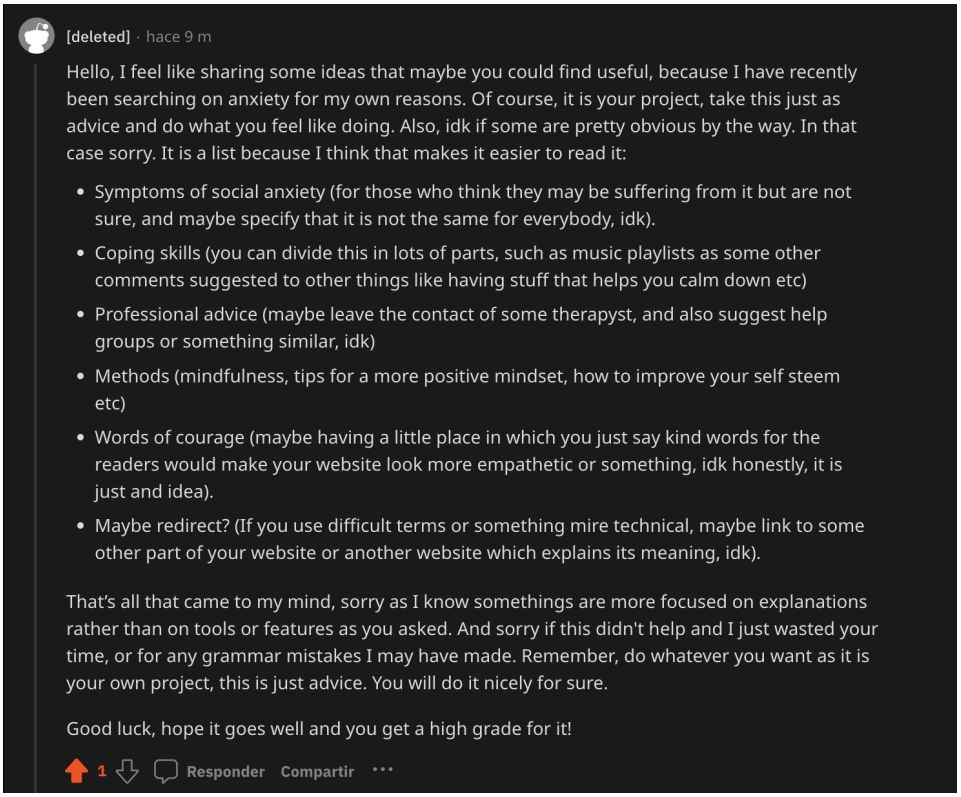


Figura 1.5.5: Comentario de usuario [4] en el subreddit *r/mentalhealth*

Fuente: Reddit

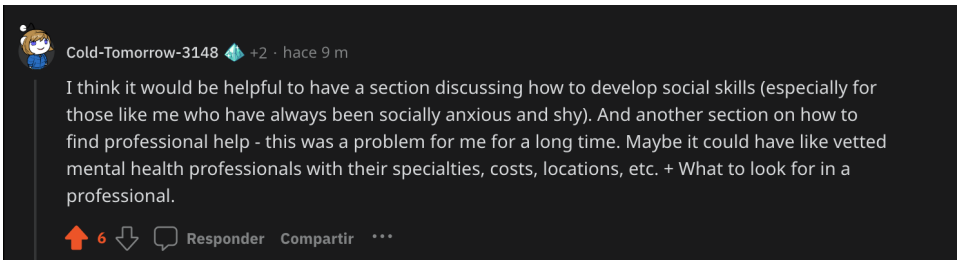


Figura 1.5.6: Comentario de usuario [5] en el subreddit r/*socialanxiety*

Fuente: Reddit

Capítulo 2. Competencias desarrolladas.

Con el desarrollo de este proyecto, se ha abarcado las siguientes competencias relacionadas con la mención de Tecnologías de la Información [14].

Competencia	Definición	Justificación
TI02	*Capacidad para seleccionar, diseñar, desplegar, integrar, evaluar, construir, gestionar, explotar y mantener las tecnologías de hardware, software y redes, dentro de los parámetros de coste y calidad adecuados.*	Durante la realización del proyecto, se ha evaluado y diseñado una aplicación web, siempre optimizando el uso de recursos y costes, sin perjudicar la calidad del software.
TI03	*Capacidad para emplear metodologías centradas en el usuario y la organización para el desarrollo, evaluación y gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, ergonomía y usabilidad de los sistemas.*	Con el fin de garantizar la máxima satisfacción del usuario final, se estudió y seleccionó la metodología ideal para permitir una mejora y constante evolución del proyecto, así como garantizar la realización del mismo.
TI06	*Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil.*	Para certificar el éxito del desarrollo del proyecto, es necesario comprender cómo diseñar e implementar un servicio basado en tecnologías de red como es una aplicación web que se implica en este proyecto.

Tabla 2: Competencias específicas abordadas

Fuente: Propia

Capítulo 3. Metodología y planificación del proyecto.

3.1 Metodología

La metodología seleccionada para la producción de este proyecto ha sido un desarrollo iterativo e incremental, incorporando las buenas prácticas y bases que propone SCRUM. Para entender la importancia de las buenas prácticas de SCRUM, primero es necesario descifrar el significado de un desarrollo iterativo e incremental. Por una parte, un desarrollo iterativo consiste en que, a lo largo de la vida del proyecto, se define una serie de marcos de tiempo, donde se realicen una serie de entrega del mismo, sin la necesidad de que sea una versión final, pero con la idea de poder obtener *feedback*[5] acerca de la entrega, y mejorarla para la siguiente *(Enciende la Luz, 2018)* [15].

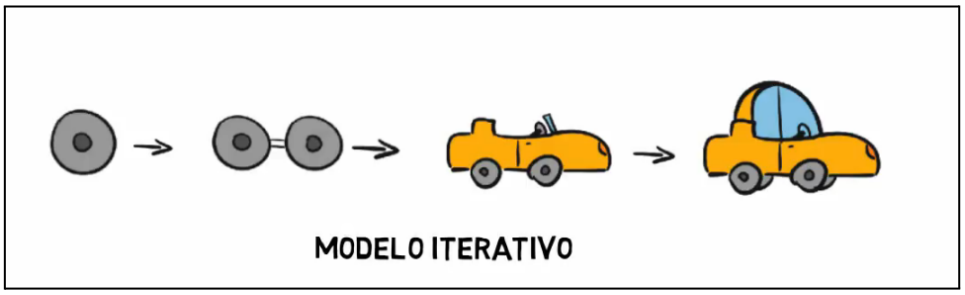


Figura 3.1.1: Esquema modelo de desarrollo iterativo

Fuente: Iterativo e incremental - Enciende la luz

Como se observa en la anterior **figura 3.1.1**, si se pone en contexto la construcción de un coche, se puede visualizar como el vehículo va tomando forma gradualmente con cada iteración, partiendo de una base sencilla como puede ser la creación de una rueda, hasta la elaboración final del automóvil.

Por otra parte, en el desarrollo incremental se analiza un proceso que a cada entrega realizada del proyecto se entregan nuevas funcionalidades por cada nueva iteración

[5]Es la acción de ofrecer información a una persona sobre un resultado. El feedback se da en evaluaciones, consejos o incluso comentarios, y pretende aportar información para futuras mejoras.

(Enciende la Luz, 2018) [15]. En el ejemplo de la siguiente **figura 3.1.2** se observa cómo se unen ambos desarrollos para completar la labor explicada el párrafo anterior, pero en este caso el punto de partida comienza con la creación de un monopatín, pasando por una gran cantidad de vehículos hasta llegar a un coche como entrega final.

Figura 3.1.2: Esquema modelo de desarrollo iterativo e incremental

Fuente: Iterativo e incremental - Enciende la luz

Como principal diferencia de ambos modelos, se puede reflexionar sobre cómo en el desarrollo iterativo e incremental, se ha ido añadiendo nuevas funcionalidades a cada entrega, como son un manillar, un asiento o un motor, mientras que en el desarrollo iterativo solo se ha tenido en cuenta completar el proyecto poco a poco de forma repetida. Otra conclusión que es importante recalcar es la importancia de la definición de los requisitos, ya que es necesario especificar correctamente desde el principio que condiciones tienen que existir para que un producto o servicio se dé por acabado y resulte exitoso *(Ramírez F. ,2022)* [16].

Por lo que se puede concretar que el desarrollo iterativo e incremental es una gran estrategia para desarrollar software de manera más eficiente y efectiva, principalmente permite a los desarrolladores realizar mejoras en la funcionalidad y la calidad del código a través de la retroalimentación continua del usuario. Según el libro "*The Nature of Software Development: Keep It Simple, Make It Valuable, Build It Piece by Piece*" de Ron Jeffries [17], esta estrategia también garantiza que se realicen pruebas regulares y que los cambios se implementen de manera controlada asegurando que el software se mantenga estable y seguro. Además ayuda a mejorar la colaboración entre el equipo de desarrollo, al permitir

que los desarrolladores trabajen juntos para solucionar problemas y mejorar el producto de software.

Una vez definido y entendido el concepto de un desarrollo iterativo e incremental, ahora aparece el papel del SCRUM dentro de este proyecto. SCRUM es un marco de trabajo ágil para el desarrollo de productos que se ha convertido en una de las metodologías ágiles más populares. Está diseñado para ayudar a los equipos a entregar productos de mayor valor en un entorno cambiante y complejo *(Highsmith, 2020)* [18]. Esta metodología se caracteriza por su enfoque en la colaboración, flexibilidad y adaptabilidad, proporcionando un marco que permite a los equipos trabajar juntos para entregar resultados rápidos *(Sutherland, 2020)* [19].

Gracias al método de trabajo que SCRUM propone, Trello [20] surge como una herramienta útil para ayudar a los equipos a desarrollar productos. Esto se debe a que ayuda a los equipos a dividir un proyecto en una serie de tareas y mantener un seguimiento de los avances. De la misma forma permite al equipo organizar el trabajo en marcos de tiempo para realizar una serie de tareas asignadas, también llamados *sprints* y mantener un seguimiento visual de su progreso. Conjuntamente, las tarjetas en el tablero se pueden usar para definir metas y reglas para los *sprints*, así como para asignar tareas y responsabilidades donde se colocan en una lista llamada *Product Backlog*. Trello también admite la colaboración entre equipos, ya que permite a los usuarios agregar comentarios y archivos a las tarjetas, lo que facilita la comunicación y la colaboración entre los miembros del equipo. Para facilitar el desarrollo del proyecto, se ha definido una serie de prioridades para cada tarea, baja, media y alta y se han dividido en tres categorías:

- **Front end.** Categoría que incluye todas las actividades relacionadas con el código, interfaz de usuario y diseño que permite que los visitantes vean e interactúen con el sitio web.
- **Back end.** Categoría que incluye toda la lógica de programación y la base de datos que alimentan al sitio web.
- **Documentación.** Se plantea como una categoría para agrupar todas las tareas relacionadas con la elaboración del informe de este proyecto.

Una vez separada las tareas en categorías y prioridades, se determinan una serie de tareas por *sprints* denominado *Sprint backlog*. Tras la finalización de cada *sprint,* que se ha planteado una duración de tres semanas para cada uno, se realiza una retrospectiva del *sprint*, donde se pone en vista lo que se ha desarrollado, cuantas tareas se han cumplido con lo esperado, con qué calidad se ha completado y sobre todo qué problemas han surgido a lo largo del *sprint*. Además gracias a la elaboración de la gráfica *Burn down*, permite observar si ha existido un problema de sobreestimación o subestimación con la asignación y definición de las tareas.

Como se puede observar en la siguiente **figura 3.1.3**, al comenzar a realizar tareas de investigación, búsqueda de un experto y redacción concreta de los requisitos, existió una ligera sobreestimación de las tareas, debido a que en una primera instancia se reflexionó que dichas actividades iban a acumular muchas más horas de trabajo que las esperadas, ya que era un campo que no se dominaba en su totalidad. En consecuencia se comenzó a examinar y preparar las tareas del siguiente *sprint*.

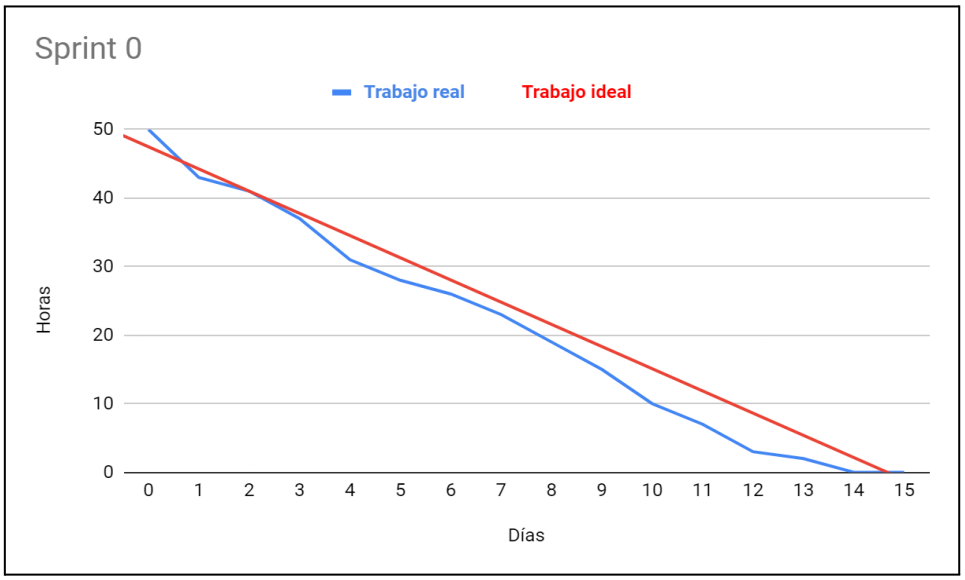


Figura 3.1.3: Gráfica burndown Sprint 0

Fuente: Propia

En el segundo *sprint*, en contraste con el primero, existió una sobreestimación tal y como se puede observar en la **figura 3.1.4**, ya que aunque este *sprint* estaba planteado de

manera que solo fuera el diseño y en el desarrollo e implementación de los *templates*[6] en la parte del *front end,* existieron numerosas complicaciones a la hora de establecer exactamente los diseños realizados. Por consecuencia, el tercer *sprint* sufrió un pequeño retraso al inicio, debido a estos problemas descritos.

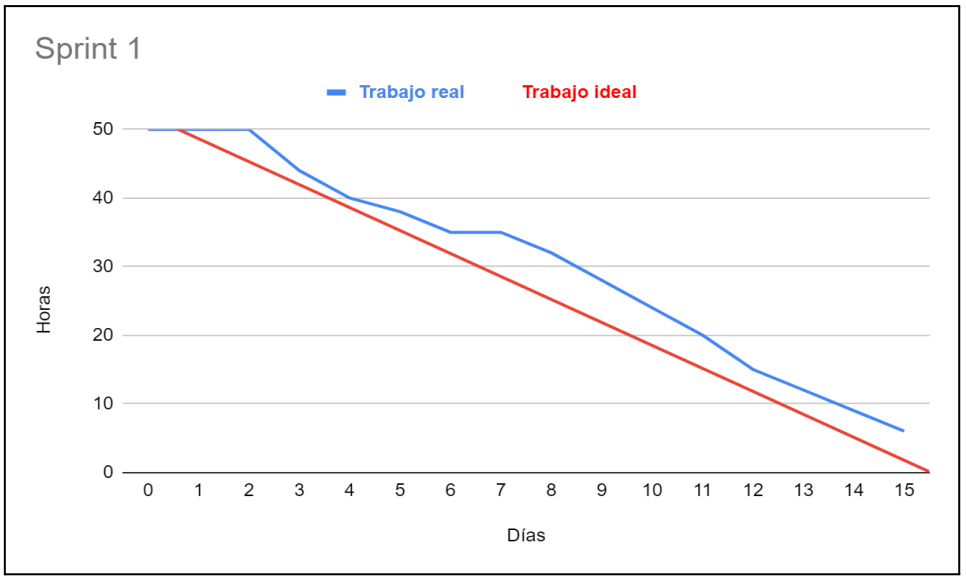


Figura 3.1.4: Gráfica burndown Sprint 1

Fuente: Propia

Tras sufrir el retraso debido al *sprint* anterior, se reflexionó sobre cómo tratar de finalizar esta tarea pendiente, y conseguir acabar las planteadas para este *sprint*. Debido a que ya se contaba con un base contrastada en el desarrollo de *endpoints*[7]*,* se consiguió completar exitosamente las tareas referentes a este asunto, y consecuentemente se alcanzó los objetivos esperados incluyendo la resolución de la tarea atrasada del anterior *sprint* **(Figura 3.1.5)**.

[6]Plantillas que contienen todos los elementos y código necesarios para crear un sitio web, como hojas de estilo, código HTML, JavaScript, imágenes, etc.

[7]Es un punto final en una API web o en una aplicación. Esto se refiere a una dirección URL específica que puede usarse para enviar y recibir información.

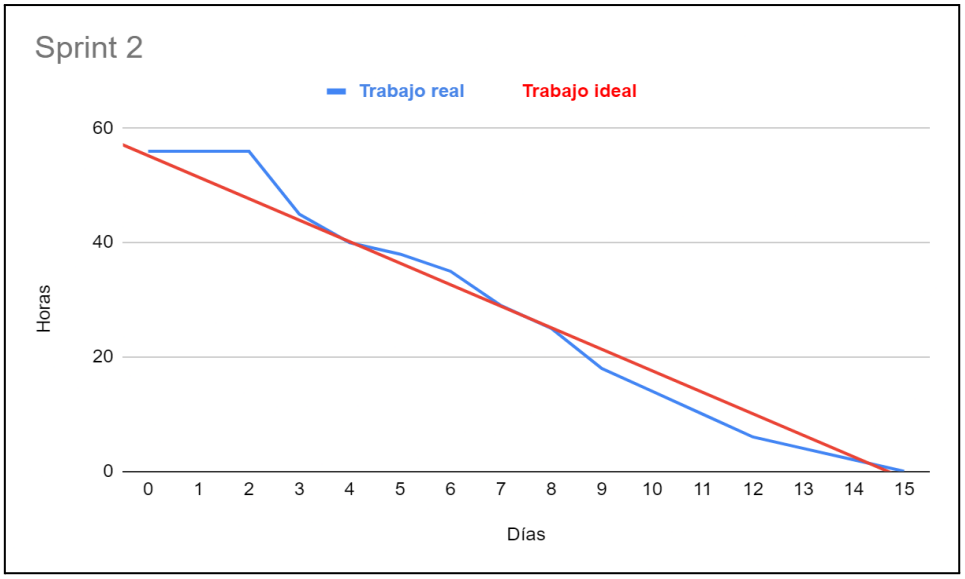


Figura 3.1.5: Gráfica burndown Sprint 2

Fuente: Propia

Por consecuencia de haber adquirido un buen ritmo tras la finalización del *sprint* 2, la resolución del *sprint* 3 fue muy satisfactoria. Se alcanzaron los objetivos de implementación de las llamadas a la API diseñada en el *sprint* 1 así como otras tareas relacionadas en mejorar la experiencia de usuario **(Figura 3.1.6)**.

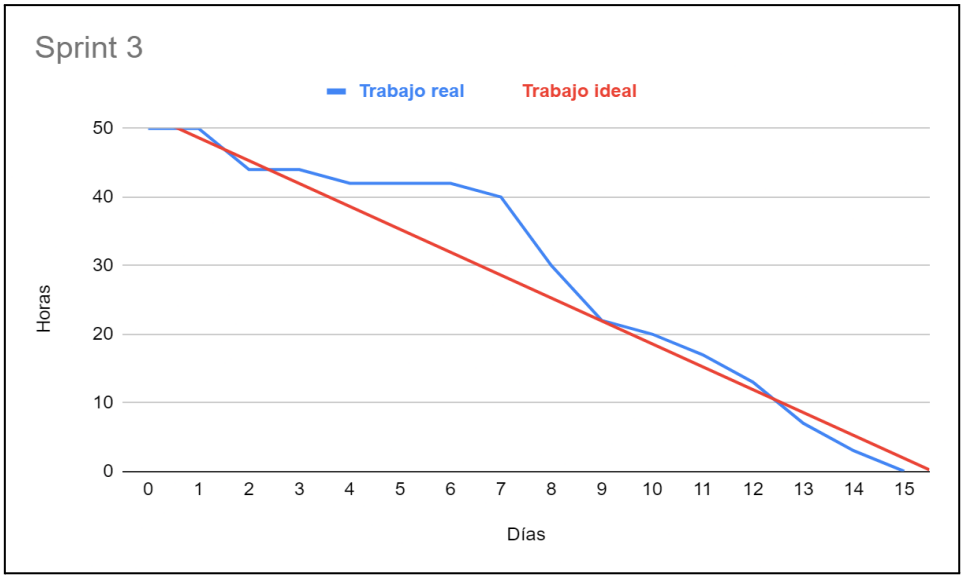


Figura 3.1.6: Gráfica burndown Sprint 3

Fuente: Propia

Tras la conclusión del *sprint* 3, se realizó una reunión con el tutor del proyecto con el fin de transmitir las sensaciones e impresiones, acerca de cómo estaba avanzando el proyecto. En vista de esto, se vio la necesidad de plantear una funcionalidad innovadora, para ofrecer mejores herramientas para los usuarios. De este modo, este *sprint* 4 se planteó exclusivamente en la búsqueda y posterior implementación de dicha nueva funcionalidad **(Figura 3.1.7)**.

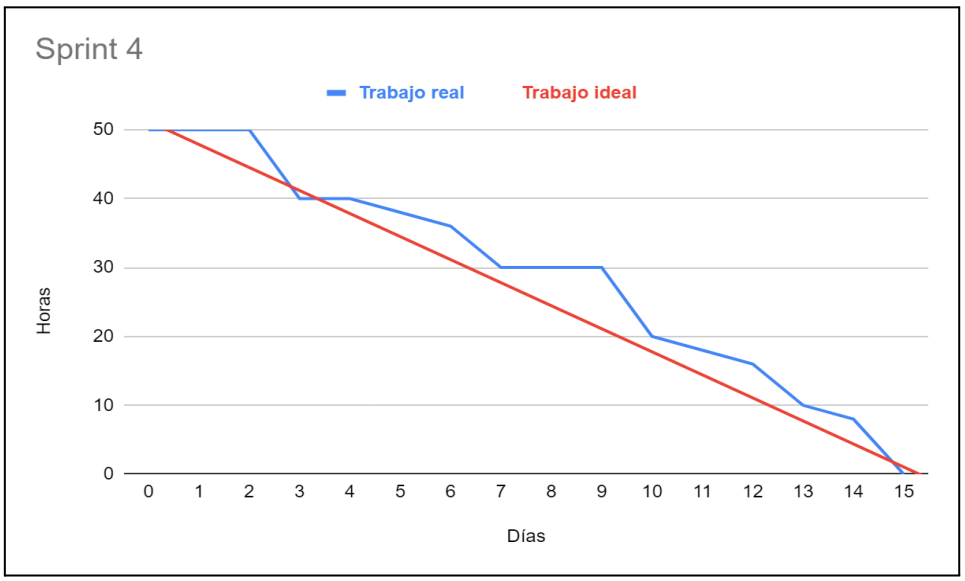


Figura 3.1.7: Gráfica burndown Sprint 4

Fuente: Propia

Por último, el *sprint* final se organizó con el fin de destinar exclusivamente las horas de trabajo a la redacción de la memoria del proyecto, a la preparación de la defensa y al despliegue de la aplicación web **(Figura 3.1.8)**.

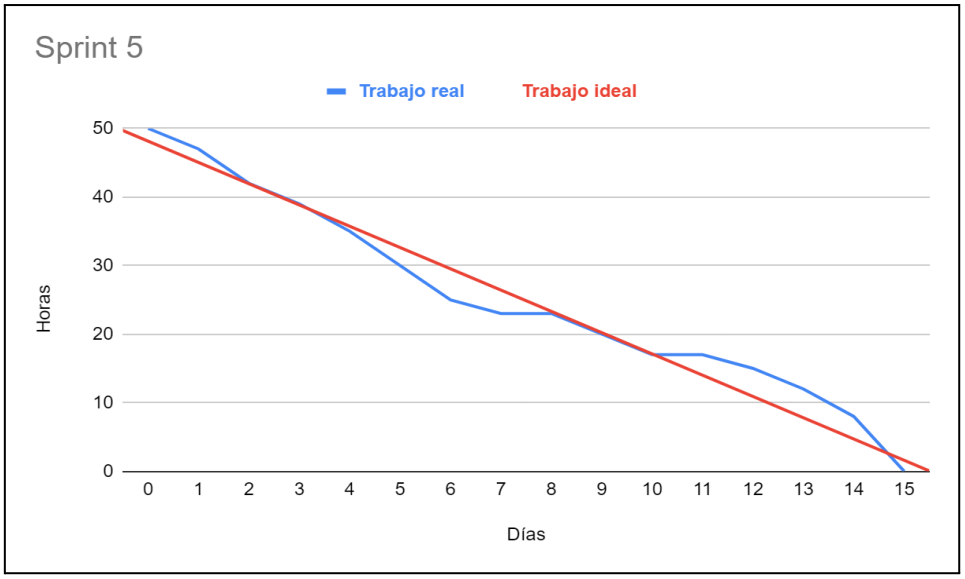


Figura 3.1.8: Gráfica burndown Sprint 5

Fuente: Propia

Siguiendo esta metodología de trabajo, se ha podido ir observando cómo avanzaba el proyecto poco a poco, como funcionalidades planteadas simples en una primera definición han evolucionado y mejorado a unas más complejas, y cómo se adaptan los tiempos de desarrollos a las tareas planteadas. Al realizar el proyecto de manera individual, se ha podido comprobar que al no existir un equipo completo con cada rol establecido, han surgido varios problemas durante la elaboración del mismo. Un ejemplo de esto puede ser en la definición de las tareas y su prioridad, ya que no han sido tan fácil de concretar y definir, además de que al ser un único individuo el desarrollador de las tareas, existía siempre una pequeña sobrestimación con el tiempo asignado.

Una de las claves de esta investigación ha sido comprender mejor los roles dentro de la metodología, por ejemplo la figura del SCRUM Master dentro de un proyecto. El SCRUM Master es un rol clave en la metodología SCRUM, ya que se encarga de guiar al equipo en la implementación de la metodología. Es responsable de ayudar al equipo a entender la estrategia del trabajo, monitorizar el progreso del equipo y ayudar a los miembros del equipo a trabajar de forma colaborativa. También facilita la eliminación de los obstáculos que el equipo enfrenta y ayuda a los miembros del equipo a comprender mejor sus roles y responsabilidades. Además de que es el principal responsable de asegurarse de que el equipo entregue valor con cada *sprint (Highsmith, 2020)* [18].

Esta “ausencia” de este rol ha resaltado levemente en este proyecto, puesto que si la toma de decisiones importantes del proyecto las realiza la misma persona encargada de realizar las tareas, es difícil tener una perspectiva real sobre el ciclo de vida del proyecto. En la siguiente **figura 3.1.9**, se puede observar un pequeño resumen sobre las funciones básicas de este rol.

Figura 3.1.9: Papel y misión del SCRUM MASTER

Fuente: Sortlist - Scrum master: ¿por qué es esencial tener uno en su equipo?

Finalmente, gracias a la elección de una metodología basada en el desarrollo iterativo e incremental, y en el apoyo en las bases que SCRUM describe como buenas prácticas, se ha conseguido ir mejorando el producto poco a poco, agregando más valor a cada funcionalidad desarrollada, y sobre todo comprender y abarcar todas las mejoras que propone este marco de trabajo.

3.2 Planificación

En la definición del proyecto, un papel esencial para el comienzo de su desarrollo, es la sección de la planificación inicial, donde se describe punto por punto, las fases integradas dentro del proyecto, así como las tareas asignadas a cada una de ellas y una estimación en horas.

En la siguiente **tabla 3.2** se puede ver representada la planificación que se realizó en la definición del proyecto, la cual se ve reflejada en el documento de presentación del proyecto TFT01.

Fases	Duración estimada	Tareas
Estudio previo / Análisis	40	**Tarea 1.1:** Entrevistas con diferentes personas que sufren ansiedad social.
		Tarea 1.2: Entrevistas con diferentes psicólogos expertos en la materia.
		Tarea 1.3: Estudio y visualización de las diferentes herramientas que existen en el mercado
		Tarea 1.4: Estudio y análisis de la herramienta de front end, base de datos y back end
Diseño / Desarrollo / Implementación	200	**Tarea 2.1:** Creación de la base de datos
		Tarea 2.2: Comunicación de los datos de la base de datos con la aplicación (Herramienta de back end)
		Tarea 2.3: Desarrollo de los mockups
		Tarea 2.4: Implementación de las plantillas y su funcionalidad (Herramienta de front end)
		Tarea 2.5: Despliegue del servicio
Evaluación / Validación / Prueba	20	**Tarea 3.1:** Comprobación del funcionamiento de la aplicación web
		Tarea 3.2: Validación del sistema para comprobar la calidad del servicio
Documentación / Presentación	40	**Tarea 4.1:** Realización de la documentación del proyecto
		Tarea 4.2: Realización de la presentación del proyecto

Tabla 3.2: Planificación inicial

Fuente: Propia

En el transcurso del desarrollo del proyecto, existieron contratiempos que no se pudieron tener en cuenta al principio de la definición de la planificación. El primero de ellos abarca la parte de psicología de este proyecto, ya que se sobreestimó la facilidad de encontrar testimonios que quisieran hablar abiertamente de su experiencia en relación a este problema, ya que por lo general es un asunto delicado de comentar, el uso de las redes sociales como *Reddit* y una persona cercana que se ofreció voluntaria y que se desconocía que sufría dicho problema, se consiguió agrupar suficiente información para abarcar este problema.

Siguiendo con esta parte, la búsqueda de expertos en la materia también fue una labor costosa, debido a que gran parte de los profesionales contactados, disponían de una agenda muy apretada por lo que era difícil poder concretar un entrevista para estudiar este asunto y comentar las diversas propuestas que habían aportados los distintos usuarios. Por suerte, al cabo de un tiempo, se localizó a dos expertos con los que se pudo concretar una entrevista, ofreciendo ayuda en el proyecto, y aconsejando sobre cómo plantear soluciones a situaciones tan delicadas.

Por consiguiente, luego de haber completado la fase de análisis del problema por parte de la psicología, se comenzó el estudio de las herramientas seleccionadas para el desarrollo de este proyecto. Fue una tarea menos compleja de lo esperado, ya que se disponía de una pequeña base de cada una de ellas, por lo que indagar en sus detalles más profundos y observar las funcionalidades más complejas que ofrecían, no llevó a una dificultad muy elevada.

Consecuentemente, acabó la finalización de estudio, y empezó la fase de desarrollo, donde se creó la base de datos, su comunicación con el *back end* de la aplicación, los *mockups* y su implementación en el *front end* de la aplicación. Seguidamente de la implementación, comenzó la comprobación del funcionamiento de la aplicación web, y validación por parte del experto y por un grupo de personas para observar las expectativas del proyecto contra el resultado final. Finalmente, se dió paso a la elaboración y documentación de la memoria del proyecto, así como el comienzo de la preparación para la defensa de este proyecto.

3.3 Arquitectura del proyecto

Como se puede observar en la siguiente **figura 3.3.1**, la arquitectura del servicio a implementar se puede separar en tres niveles:

- **Data layer**
- **Back end**
- **Front end**

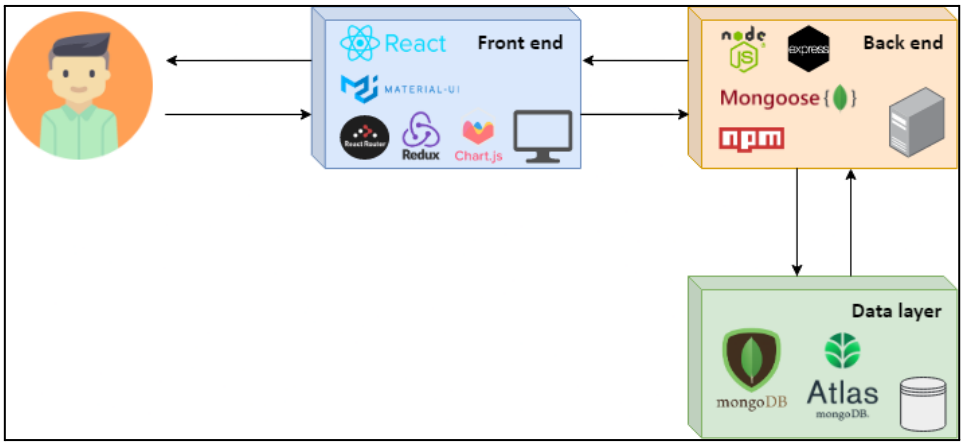


Figura 3.3.1: Arquitectura por niveles del proyecto

Fuente: Propia

En la capa de *data layer* se almacena toda la estructura relacionada con la base de datos y su correspondiente despliegue a la nube. Aquí destaca la plataforma de base de datos NoSQL MongoDB**,** ampliamente utilizada para la creación y manipulación de datos combinando su servicio en la nube MongoDB Atlas, que proporciona una forma fácil y segura de desplegar, administrar y mantener la base de datos MongoDB a escala global. Se utiliza dicho servicio con el fin de ser accesible, flexible y de alta disponibilidad mediante *clusters*[8]*,* con la intención de manipular los datos sin necesidad de contar con la base de datos de manera local *(MongoDB, s.f)*[21].

La capa de *back end* se encuentra en la siguiente parte del esquema y es responsable de todos los procesos necesarios para que exista una comunicación fiable entre la base de datos y el cliente *(Chapaval, M. ,2018)* [22]. Se emplea NodeJS como entorno base para la

[8]Es un conjunto de ordenadores conectados entre sí para formar un sistema de computación altamente escalable y confiable. Se usan para aumentar la disponibilidad y el rendimiento de los servicios informáticos, permitiendo la realización de tareas de computación simultáneas y la distribución de datos.

programación de aplicaciones web. Es un entorno de servidor basado en JavaScript, ideal para el desarrollo de aplicaciones web escalables *(Node.JS ,s.f)* [23]. Acompañando a NodeJS, se emplea el *framework*[9] ExpressJS para agilizar el desarrollo de los servicios web y la librería[10] de JavaScript Mongoose usada para crear un enlace entre la base de datos MongoDB y el marco ExpressJS. Con esto logramos que la parte de *back end* actúe como una API Restful, permitiendo la comunicación entre el servidor y el cliente usando diferentes protocolos *(Chojrin, M. ,2020)* [24].

Por último, se encuentra la capa que interactúa con el usuario cliente, la capa *front end*. Esta dimensión abarca todas las funcionalidades esperadas en un sitio web que permiten dar *feedback* al usuario, como menús desplegables, imágenes, iconos, elementos gráficos y todo lo referente al diseño web *(Chapaval, M. ,2018)* [22]. Para desarrollar dichas funcionalidades se emplea el uso de la librería de JavaScript React, pensada principalmente en el desarrollo de interfaces de usuario y diseñada para ayudar a los desarrolladores a construir aplicaciones web y móviles de forma rápida y sencilla *(React, s.f)* [25]. Además de otras librerías complementarias muy populares como React Router Dom, Material UI, Redux entre otras con el propósito de ofrecer una mejor experiencia de navegación dentro del servicio y un diseño de calidad *(Calatrava, S. G. , 2022)*[26].

[9]Conjunto de herramientas, librerías y patrones de diseño que se utilizan para crear aplicaciones. Estas herramientas proporcionan una estructura básica que los desarrolladores pueden usar para crear aplicaciones más complejas.

[10]Conjunto de código preescrito que se puede reutilizar en varios proyectos de programación. Estas librerías contienen instrucciones que el programador puede usar para realizar tareas comunes sin tener que escribirlas desde cero.

Capítulo 4. Herramientas utilizadas.

4.1 Herramientas de gestión y organización

Para garantizar la eficiencia y productividad dentro de un proyecto, es necesario y vital emplear herramientas de gestión y organización para controlar íntegramente que se va a realizar y cómo. Es realmente importante controlar este tipo de herramientas, ya que normalmente el desarrollo de cualquier producto o servicio, va seguido de un equipo de trabajo con miembros con distintas formas de trabajar.

- **Git**

Git[27] es un herramienta software *open source*[11], diseñada para gestionar proyectos ofreciendo un sistema de control de versiones distribuido. La finalidad de esta herramienta es registrar todas las modificaciones que se elaboran en un repositorio de un proyecto en concreto. Permite trabajar desde local o remoto, usando plataformas pensadas para el almacenamiento de repositorios.

- **Github**

Github[28][29] es un servicio de alojamiento de código y repositorio web para proyectos de software que usa el sistema de control de versiones Git. Está destinado principalmente a desarrolladores y equipos de desarrollo de software para gestionar proyectos compartiendo y colaborando en el código. El servicio fue fundado en 2008 y es una de las plataformas de desarrollo de software más populares del mundo. Además, GitHub ofrece una variedad de características, como herramientas de colaboración, análisis de código, integración con varios servicios, etc.

[11]Programa informático cuyo código fuente está disponible para su uso y modificación gratuitamente.

- **Trello**

Trello[20] es una aplicación basada en el método *Kanban*[12] y sirve para gestionar tareas, permitiendo organizar el trabajo en grupo de forma colaborativa mediante tableros virtuales, compuestos de listas de tareas en formato columnas. En este proyecto, aunque se aborde de forma solitaria, se ha considerado utilizar dicha herramienta con el fin de poder estructurar correctamente las tareas a realizar y poder observar una evolución continua del proyecto, así como tener siempre presente las actividades a implementar.

- **Google Docs**

Google Docs[30] es una herramienta gratuita que permite elaborar documentos, presentaciones u hojas de cálculo y trabajar en ellos desde la nube. Es la herramienta seleccionada para desarrollar esta memoria. debido a su versatilidad para poder acceder desde varios dispositivos y la utilidad que ofrece que esté en la nube con el fin de tener un mejor control sobre la versión de la memoria.

- **MindMeister**

MindMeister[31] es una herramienta de software para la creación de mapas mentales. Permite a los usuarios crear mapas visuales que pueden ser compartidos y organizados en línea. Esto resulta útil para la planificación de proyectos, la toma de decisiones y la organización de ideas. En esta ocasión se ha utilizado para planificar la construcción de la memoria y tener a simple vista todos los conceptos a desarrollar en el documento.

[12]Método para mejorar el flujo de trabajo y reducir los tiempos de entrega. Se basa en la visualización de la información para identificar los cuellos de botella en el proceso de producción y mejorar la eficiencia.

4.2 Herramientas de diseño y planteamiento

En el ámbito de desarrollo de aplicaciones webs, una parte crucial del desarrollo es el planteamiento de la idea, y sobre todo la exposición del concepto a través de los diagramas y *mockups*[13]. Por tal razón, existen numerosos recursos y herramientas que disponen de funcionalidades para poder modelar y planear una ejecución del proyecto a consecuencia. Del mismo modo estas herramientas también permiten conocer en detalle que va a encontrar el usuario cliente cuando visite nuestra aplicación web, que funcionalidades tiene permitidas, cuales son los flujos de las actividades dentro del entorno, etc.

- **Figma**

Figma[32] es una herramienta de creación de prototipos orientada en el diseño de interfaces y experiencias de usuario. Focalizado principalmente en uso mediante la web, también dispone de un *software* de escritorio para dispositivos Windows y macOS. Está enfocado en permitir el trabajo colaborativo y tiene una curva de aprendizaje muy bien desarrollada. Se ha utilizado dicha herramienta para desarrollar los *mockups* de la respectiva aplicación web.

- **Draw.io**

Draw.io[33] es un software distribuido por JGraph Ltd , utilizado para diseñar y generar diagramas de flujo, gráficos y esquemas. Su principal ventaja es que dispone de una versión completamente funcional y gratuita a través del navegador web, y tiene compatibilidad con Google Drive, permitiendo poder guardar archivos en la nube directamente. En la realización de esta memoria se ha utilizado dicha herramienta para explicar la arquitectura del software, la usabilidad del usuario dentro del entorno así como demás esquemas que se encuentran a lo largo del documento.

[13]Maqueta de un producto que se utiliza para demostrar cómo se verá el producto final.

- **Pixel True**

Pixel True[34] es un sitio web desarrollado por Pixel True Studio, principalmente enfocado en establecer una plataforma con recursos artísticos abarcando esencialmente ilustraciones. En una primera instancia, se desarrollaron los *mockups* de la aplicación a través del Figma, usando ilustraciones de dicha página. Sin embargo, tras haber realizado un estudio sobre los colores, tipografía y disposición de elementos, se descartó la idea de utilizar las ilustraciones de este sitio web, debido a que no ofrecía una personalización sobre las ilustraciones.

- **unDraw**

UnDraw[35] es un sitio web creado por Katerina Limpitsouni que dispone y ofrece ilustraciones *open source.* Dentro del apartado de ilustraciones, se incluye una funcionalidad que permite personalizar los colores a través de un selector, dando una gran variedad de personalización, lo que ayuda a mantener la consistencia de colores a través de la aplicación a desarrollar. Todas las ilustraciones que se pueden observar en el producto final han sido utilizadas en este sitio web.

4.3 Herramientas de desarrollo

Para la construcción y desarrollo de una aplicación web, existen numerosas formas y herramientas para abordar dicha aspiración, ya que al ser un mundo tan extenso, cada desarrollo se ajusta mejor dependiendo de la idea general del proyecto. Un ejemplo de esta afirmación se observa al conocer las diferencias entre un desarrollo orientado a una aplicación web estática, una dinámica, una tienda virtual, un portal web o aplicaciones web con gestor de contenidos.

- **Aplicación web estática:** Aplicación web que no contiene ninguna lógica de programación. Estas aplicaciones se componen principalmente de archivos HTML, CSS y JavaScript que se cargan y ejecutan en el navegador del cliente. Estas

aplicaciones no se actualizan con el tiempo y no responden a los cambios de los usuarios *(Maluenda, R. 2022)*[36].

- **Aplicación web dinámica:** Aplicación web que contiene lógica de programación. Estas aplicaciones combinan archivos HTML, CSS y JavaScript con código de programación para crear una aplicación web interactiva. Estas aplicaciones se actualizan con el tiempo y responden a los cambios de los usuarios *(Maluenda, R. 2022)*[36].

- **Tienda virtual o *Ecommerce:*** Aplicación web que permite a los usuarios comprar productos o servicios en línea. Normalmente contiene lógica de programación para gestionar el proceso de compra, el seguimiento de pedidos, el pago y la entrega. Esta aplicación también suele contener una interfaz de usuario intuitiva para que los usuarios puedan fácilmente seleccionar los productos que desean comprar y completar el proceso de compra *(Maluenda, R. 2022)*[36].

- **Portal web:** Aplicación web que contiene una gran cantidad de contenido relacionado con un tema en particular. Estas aplicaciones suelen contener una variedad de contenido, como noticias, videos, juegos, aplicaciones y mucho más. Estas aplicaciones suelen contener lógica de programación para gestionar el contenido y permitir a los usuarios filtrar, buscar y navegar por el contenido *(Maluenda, R. 2022)*[36].

- **Aplicaciones web con gestor de contenido:** Aplicación web que permite a los usuarios crear, editar y gestionar contenido a través del uso de CMS. Esta aplicación suele contener una interfaz de usuario intuitiva para que los usuarios puedan fácilmente agregar, editar y eliminar contenido *(Maluenda, R. 2022)*[36].

Estudiando las diferentes situaciones que se plantean se ha decidido escoger el camino del desarrollo de una web dinámica, con el propósito de poder ofrecer un servicio con contenido personalizado y adaptado a los usuarios, además de presentar actualizaciones constantemente sumando el hecho de que es fácil de mantener y puede ajustarse a los

cambios. Asimismo, el papel del aprendizaje tiene un fuerte valor en la decisión, ya que el desarrollo web dinámico proporciona una visión general de las últimas tendencias en tecnología web, permitiendo estar al tanto de los últimos cambios tecnológicos en la industria.

4.3.1 Lenguajes de marca y programación

- **HTML**

HTML[37] (Lenguaje de Marcas de Hipertexto, del inglés HyperText Markup Language) es el componente más básico de la Web. Define el significado y la estructura del contenido web. Para el apartado del cliente de la aplicación web, HTML es la herramienta indicada para estructurar y exhibir el contenido a los usuarios.

- **CSS**

Las hojas de estilo en cascada CSS[38] son un lenguaje de hojas de estilo que se utiliza para describir la presentación de un documento escrito en HTML o XML (incluidos los dialectos de XML como SVG, MathML o XHTML). CSS describe cómo deben presentarse los elementos en la pantalla, en el papel, en la voz o en otros medios. En este proyecto se ha empleado principalmente para dar forma y estilo a los elementos que se encuentran dentro de la aplicación web en la parte del cliente.

- **JavaScript**

JavaScript[39] es un lenguaje de programación que permite implementar funcionalidades complejas en sitios web. Principalmente se emplea en el lado cliente del sitio web, aunque gracias a *frameworks* de desarrollo y librerías es capaz de emplearse también del lado del servidor. Para esta aplicación web, se ha utilizado tanto en la parte del cliente como del servidor mediante *frameworks* y librerías correspondientes para cada caso.

4.3.2 Frameworks, librerías y paquetes

Los *frameworks*, librerías y paquetes son una parte indispensable de la programación moderna. Estas herramientas proporcionan a los desarrolladores una forma de hacer que sus proyectos sean más eficientes y escalables. Estos recursos ofrecen una gran cantidad de funcionalidades que permiten a los programadores ahorrar tiempo y esfuerzo al momento de crear sus aplicaciones. Además pueden ser utilizados tanto para proyectos pequeños como para grandes sistemas de información

4.3.2.1 Backend

- **Express JS y Node JS**

  NodeJS[40] es un entorno que trabaja en tiempo de ejecución, de código abierto, multiplataforma, que permite a los desarrolladores crear toda clase de herramientas de lado servidor y aplicaciones en JavaScript. Express[41] es un *framework* de aplicaciones web Node.js mínimo y flexible que proporciona un sólido conjunto de características para aplicaciones web y móviles. Ambas herramientas se han empleado para el desarrollo de la parte del servidor de la aplicación web.

- **MongoDB**

  MongoDB[42][43] es una base de datos de documentos que ofrece una gran escalabilidad, flexibilidad, modelo de consultas e indexación avanzado. Es la base de datos empleada para almacenar toda la información relacionada con la aplicación web, como los recursos a exhibir en la aplicación o información de los usuarios.

- **MongoDB Atlas**

  MongoDB Atlas[44] es la base de datos de MongoDB como servicio que permite implementar, utilizar y escalar una base de datos de MongoDB alojada en un

servidor. Gracias a esta funcionalidad que ofrece MongoDB, permite acceder a la información de la BBDD sin la necesidad de alojar la base de datos de manera local.

- **Mongoose**

Mongoose[45] ofrece una solución sencilla, basada en esquemas, para modelar los datos de su aplicación cuando se utiliza un BBDD de MongoDB . Incluye funciones como reparto de tipos, validación, creación de consultas, ganchos de lógica empresarial, etc.

- **Body Parser**

Body Parser[46] es un paquete centrado en analizar el cuerpo de la solicitud de una petición. Su papel en el proyecto es analizar las solicitudes que se realizan a la API Restful, como pueden ser los GET/POST/DELETE/PATCH.

- **Cookie Session**

Cookie Session[47] es un paquete que actúa como middleware para las sesiones de los usuarios en un sitio web. Una sesión de usuario puede ser almacenada de dos maneras principales con las cookies: en el servidor o en el cliente. Este paquete almacena los datos de la sesión en el cliente dentro de una cookie.

- **Cors**

Cors[48] es un paquete de NodeJS para proporcionar un middleware Connect/Express que puede utilizarse para habilitar CORS con varias opciones. Es necesario la utilización de este paquete, ya que las peticiones a la API, se van a

realizar desde el lado cliente de la aplicación, por lo que si no se emplea dicho paquete, se bloquean las peticiones.

- **Dotenv**

Dotenv[49] es un paquete que carga las variables de entorno de un archivo .env en process.env. Se emplea para no introducir credenciales dentro del código de ejecución, como la conexión al servidor de la base de datos.

- **JWT**

Los tokens web JSON (JWT)[50][51] son un medio compacto y seguro para representar reclamaciones que se transfieren entre dos partes. Las reclamaciones de un JWT se codifican como un objeto JSON que se utiliza como carga útil de una estructura JSON Web Signature (JWS) o como texto plano de una estructura JSON Web, JSON Web Encryption (JWE), lo que permite que las reclamaciones estén protegidas en su integridad con un código de autenticación de mensajes (MAC) y/o encriptadas. En este proyecto se utiliza el paquete de JWT de NodeJS, para el envío y obtención de la información de los usuarios y garantizar la integridad de los datos.

- **Bcrypt**

Bcrypt[52] es una librería que ayuda a realizar hash de las contraseñas. Permite garantizar la seguridad e integridad de las contraseñas de los usuarios que se guardan en BBDD.

- **Nodemailer**

Nodemailer[53] es un paquete de Node JS que facilita el envío de correos electrónicos desde aplicaciones hechas con Node JS. Funciona con una variedad de proveedores de servicios de correo electrónico, como Gmail, Outlook, Zoho, Hotmail y muchos más. Está diseñado para ser fácil de usar y permite a los desarrolladores enviar correo electrónico de forma rápida y sencilla. En esta aplicación web se ha

empleado para el envío de correos al servicio de contacto de la web, y para que el usuario pueda recuperar la contraseña a través de su correo electrónico.

4.3.2.2 Front end

- **React**

React[54] es una librería *open source* desarrollada en JavaScript, creada en 2011 por Jordan Walke, un ingeniero de software de Facebook. Se utiliza principalmente para el desarrollo de interfaces de usuario y caracterizado específicamente para desarrollar SPAs (*Single Page Applications*). Uno de sus principales atractivos es que emplea el uso de componentes y permite que sean reutilizables. El objetivo principal de esta librería es ser rápida, escalable y simple.

- **React Router**

React Router[55] es un librería para React que ofrece una colección de componentes para la navegación a través de la aplicación. Con el uso de la librería se obtiene un enrutamiento dinámico, así que tenemos rutas que renderizan un componente en concreto.

- **React Hook Form**

React Hook Form[56] es un librería para React orientada al uso de formularios dentro de la aplicación web. Por medio de esta librería se permite tener un control sobre los datos a introducir dentro de un formulario.

- **React Redux Toolkit**

Redux Toolkit[57] es una librería que simplifica el uso de Redux en una aplicación. Ofrece varias funcionalidades que simplifican la usabilidad que trae su librería superior Redux, como es el uso de la configuración de la *store,* la creación de los *reducers*, lógica de actualización inmutable, etc. Para este proyecto se ha empleado

en el uso de las sesiones de usuarios, así como la conexión la API Restful creada a partir del back end.

- **MUI**

MUI[58] es una librería que ofrece un conjunto muy completo de herramientas de interfaz de usuario. Existe una gran cantidad de componentes reutilizables, además de entregar una gran personalización de los mismos. En el desarrollo del proyecto, existen una gran variedad de componentes que se estructuran por medio de componentes de esta librería.

- **Toastify**

Toastify[59] es un paquete que ofrece alertas personalizables para dar *feedback* al usuario dependiendo del evento que está ocurriendo. En la aplicación web desarrollada, se puede encontrar estas alertas, tanto en las ocasiones de eventos realizados con éxito, como en los que ocurren algún tipo de fallo.

- **React-calendar**

React-calendar[60] es un paquete React que proporciona un calendario interactivo y personalizable para tus aplicaciones web. Los usuarios pueden navegar entre meses, ver eventos programados y agregar eventos personalizados.

- **Chart JS y React Chart JS 2**

Chart.js[61] es un paquete de JavaScript que ayuda a los desarrolladores a crear gráficos en sus aplicaciones web. El paquete proporciona herramientas para crear gráficos de línea, de barras, de pastel, de área y de burbujas. Estos gráficos se pueden personalizar usando varios temas y colores para obtener un aspecto único. Chart.js es una excelente herramienta para las aplicaciones que requieren la visualización de datos. Por otra parte React Chart.js 2[62] es un paquete de React que ayuda a los desarrolladores a agregar gráficos y visualizaciones de datos a sus

aplicaciones web. El paquete proporciona una serie de componentes de React para crear gráficos de línea, de barras, de pastel, de área, de burbujas y mucho más. Estos componentes se pueden personalizar con diferentes opciones. Estas dos librerías se han empleado en la sección del diario personal para representar los datos del usuario a través de dos gráficos.

4.3.3 Otros

- **Visual Studio Code**

Visual Studio Code[63] es un editor de código optimizado que proporciona muchas facilidades para escribir, depurar y probar código. Inicialmente incluye un mínimo de componentes y funciones básicas de un editor con soporte nativo para JavaScript/TypeScript y Node.js, sin embargo, es personalizable con los cientos de plugins o extensiones disponibles para escribir código en diferentes lenguajes. Dispone de una gran variedad de características útiles para agilizar el trabajo, que lo hacen el editor preferido por muchos para trabajar los proyectos.

- **Postman**

Postman[64] es una herramienta que principalmente permite crear peticiones sobre APIs de una forma muy sencilla y poder, de esta manera, probar las APIs. Dispone de una versión web y escritorio para Windows, Mac y Linux. Considerando que en el desarrollo de este proyecto, está planteado la construcción de una API para la obtención y manipulación de los datos, es esencial disponer de una herramienta para probar dichas funcionalidades.

- **Redux DevTools**

Redux DevTools[65] es una extensión para navegadores web que permite depurar los cambios de estado de la aplicación que utiliza Redux. Para este proyecto es una

gran herramienta para controlar los estados de la *store* que se emplea en la aplicación web.

- **Render**

Render[66] es una plataforma de servicios en la nube para la creación y entrega de contenido multimedia. Ofrece a los usuarios una interfaz fácil de usar, herramientas de edición y almacenamiento en la nube, así como una variedad de aplicaciones y servicios para ayudarles a crear y distribuir contenido de manera eficiente. En este proyecto, el despliegue de la *API REST* se ha realizado a través de esta plataforma.

- **Vercel**

Vercel[67] es una plataforma que permite a los desarrolladores desplegar aplicaciones web y sitios web de forma rápida y sencilla. Las aplicaciones desarrolladas con Vercel se ejecutan en la nube, lo que permite a los desarrolladores ahorrar tiempo y recursos al no tener que administrar sus propios servidores. Para este proyecto, se ha empleado para desplegar el *frontend* de la aplicación.

- **Lighthouse**

Lighthouse[68] es una extensión de navegador web desarrollada por Google. Está diseñada para ayudar a los desarrolladores a mejorar el rendimiento, la accesibilidad y el SEO de sus sitios web. Esta herramienta se puede ejecutar directamente desde el navegador o desde la línea de comandos. El informe generado por Lighthouse proporciona una descripción detallada de la calidad de la página web, junto con recomendaciones sobre cómo mejorarla. Es una herramienta de gran utilidad que servirá para evaluar la aplicación web de una forma más técnica.

4.4 Frameworks vs Gestores de contenido

A la hora de comenzar la creación de una aplicación web, puede surgir la pregunta de qué herramientas es mejor utilizar para el desarrollo, si utilizar *frameworks* o gestores de contenido (CMS). Existen numerosas opiniones al respecto, y cada bando tiene su utilidad e inconvenientes. En este proyecto se han seleccionado los *frameworks* como principales herramientas de desarrollo, pero antes de realizar dicha elección, se investigó sobre las ventajas que ofrecía frente a los gestores de contenido (CMS).

Usar *frameworks* para el desarrollo de un sitio web ofrece una serie de ventajas en comparación con los CMS. Esto incluye mayor control del diseño, permitiendo a los desarrolladores personalizar la apariencia y la funcionalidad del sitio web sin estar limitados por los parámetros estándar de los CMS. Esto también se traduce en un mejor rendimiento, ya que los frameworks de desarrollo web generalmente procesan los datos de una manera más eficiente. Además, pueden escalar fácilmente para satisfacer el crecimiento de la demanda y admitir la funcionalidad adicional, mientras que los CMS tienen un conjunto fijo de funciones. Por último, los frameworks de desarrollo web ofrecen mayor seguridad debido a su estructura jerárquica, lo que los hace menos vulnerables a los ataques que los CMS *(Azzouz, M. , 2021)*[69].

En cuanto a diseño, no existe una gran diferencia a simple vista si el diseño de una página web está realizado mediante *frameworks* o un CMS. Sin embargo, algunos *frameworks* ofrecen herramientas y componentes que pueden simplificar el diseño de la página web, permitiendo así que sea mucho más flexible. Por otra parte los CMS emplean el uso de plantillas creadas, que suelen ser muy básicas y pueden limitar la funcionalidad y el diseño de la página web. También es posible que las plantillas no sean compatibles con todas las funciones y características del CMS *(Azzouz, M. , 2021)*[69].

También se debe comentar que a diferencia de los gestores de contenido, los *frameworks* suelen ser más complejos de utilizar que los CMS, lo que puede hacer que el desarrollo de la aplicación web se haga más lento y costoso. Además de que también es necesaria cierta experiencia en el desarrollo de software *(Azzouz, M. ,2021)*[69].

5. Análisis

5.1 Roles de usuario

- **Usuario cliente**

  El principal rol de la aplicación web. Tiene la posibilidad de visualizar todo el contenido del sitio web, así como diversas funciones orientadas a su perfil, como es la gestión del mismo o la creación de un diario personal.

- **Expertos**

  Son los principales exportadores de información dentro del sitio web. Podrán aportar nueva información en las secciones creadas dentro del sitio**.** Serán usuarios básicos pero deberán pasar un proceso para poder convertirse en dicho rol. (Aportar información y acreditación de su respectivo estudio)

- **Administradores**

  Cumplirá la figura estándar de un administrador web clásico. Esto incluye la capacidad de crear, editar y eliminar contenido, así como la gestión de usuarios y la configuración de la página web. Se encargará de que toda la información dentro de la web sea visible, controlará las acciones que pondrán realizar los distintos roles de usuario y estarán al tanto de los posibles errores que ocurran en el sistema.

5.2 Casos de uso

Los casos de uso son útiles porque proporcionan una visión clara de las expectativas y los requisitos del sistema. Esto ayuda a los miembros del equipo a comprender mejor cómo se usará el producto, a identificar los problemas potenciales y a diseñar un producto que satisfaga las necesidades reales del usuario. Los casos de uso también ayudan a los equipos a desarrollar mejores requisitos para el sistema y a mejorar el diseño de la interfaz de usuario *(Docenteunivia, 2014)*[70].

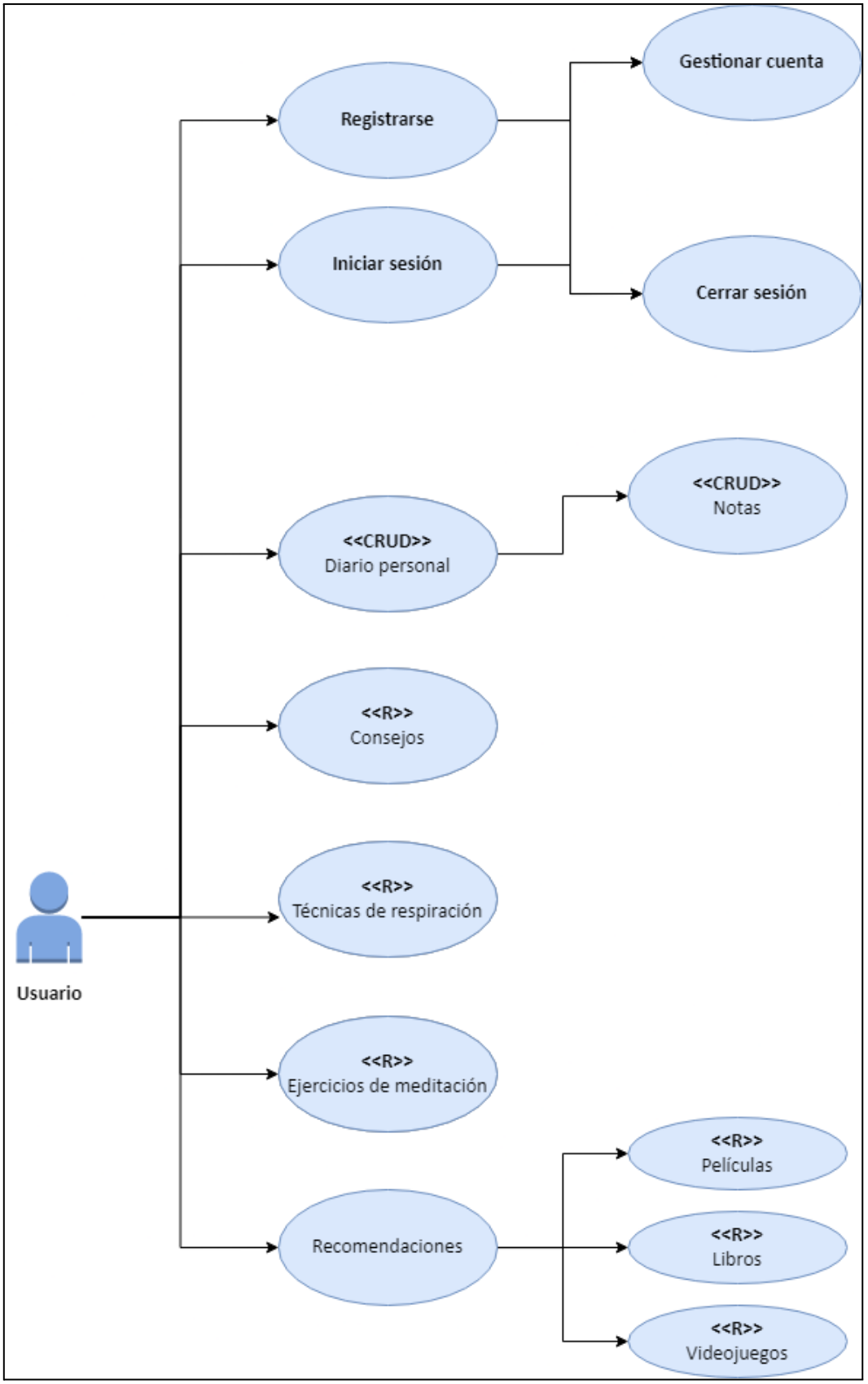


Figura 5.2.1: Diagrama de casos de uso de un usuario cliente

Fuente: Propia

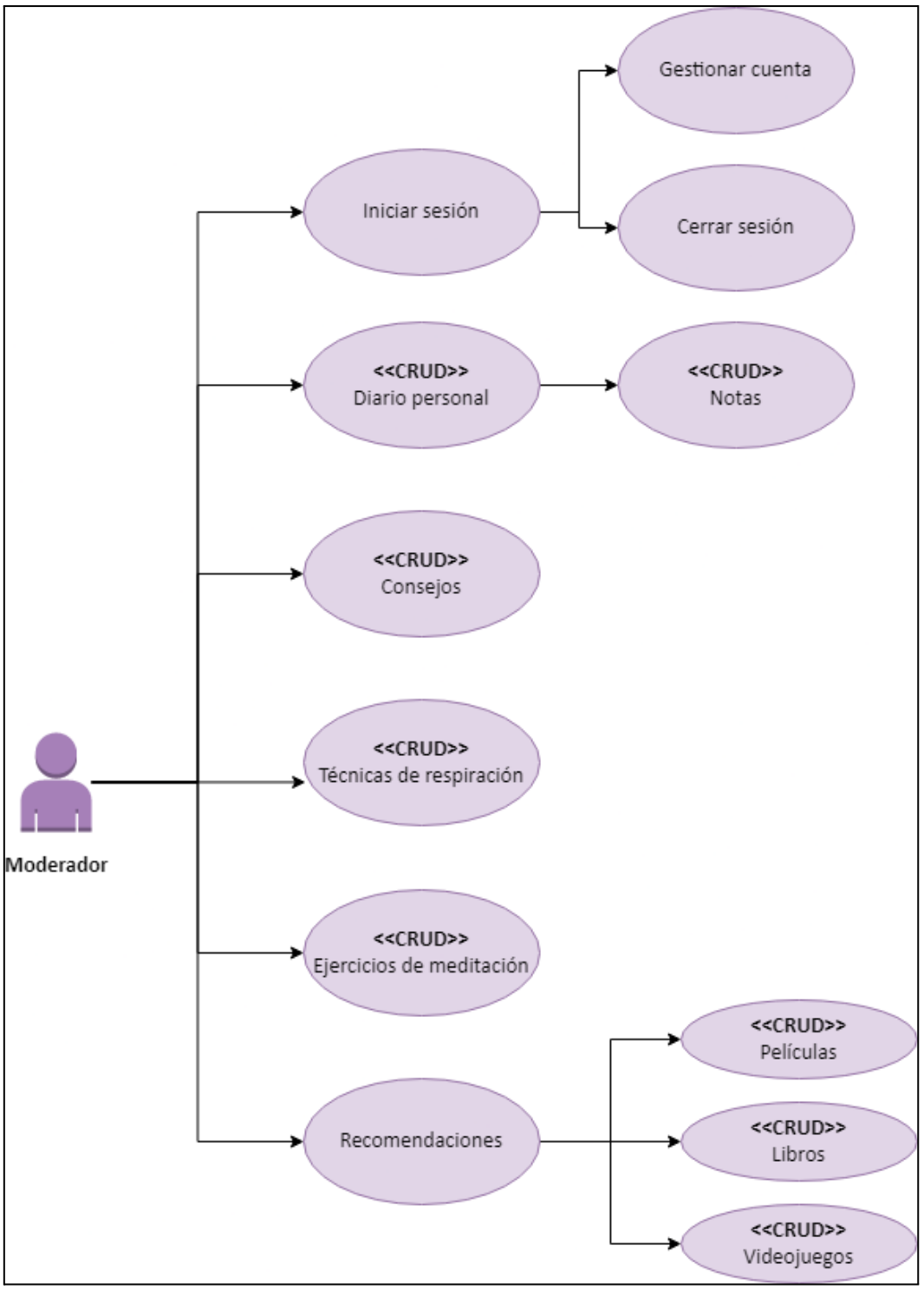


Figura 5.2.2: Diagrama de casos de uso de un moderador

Fuente: Propia

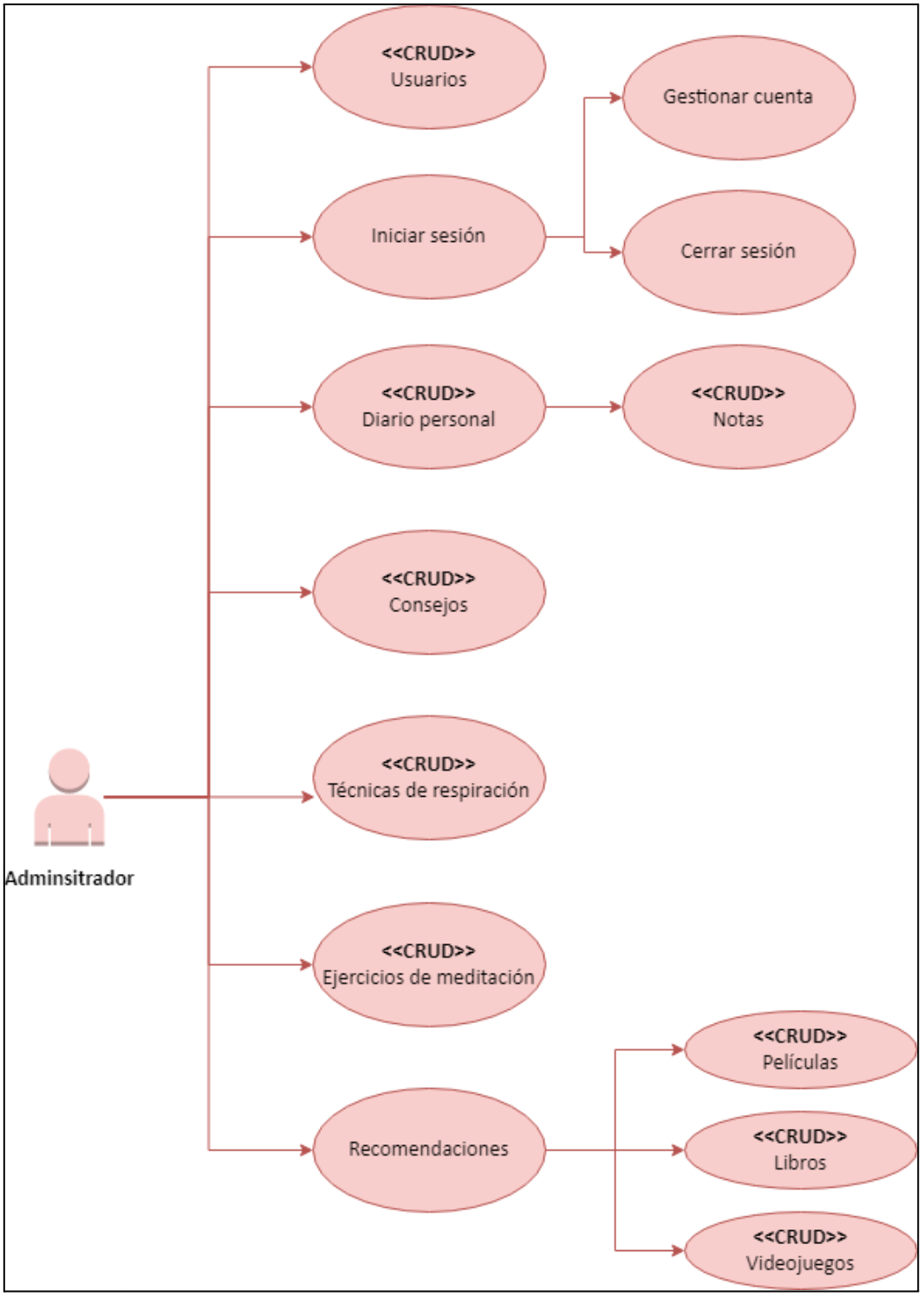


Figura 5.2.3: Diagrama de casos de uso de un administrador

Fuente: Propia

En las siguiente **tabla 5.2** se proporciona una breve descripción acerca de los diagramas representados anteriormente:

Id	Caso de uso	Rol	Descripción
1	Registrarse	Usuario	El usuario puede crearse una cuenta en el sitio web
2	Gestionar cuenta	• Usuario • Moderador • Administrador	Si tiene una cuenta, puede gestionarla
3	Iniciar sesión	• Usuario • Moderador • Administrador	Puede iniciar sesión con su cuenta en el sitio web
4	Cerrar sesión	• Usuario • Moderador • Administrador	Puede cerrar sesión con su cuenta en el sitio web
5	Diario personal	• Usuario • Moderador • Administrador	Puede ver, crear, editar y eliminar sus diarios personales
6	Notas	• Usuario • Moderador • Administrador	Puede ver, crear, editar y eliminar sus notas pertenecientes a un diario personal
8	Consejos	Usuario	El usuario puede ver los consejos exhibidos en el sitio web
		• Moderador • Administrador	Puede ver, crear, editar y eliminar los consejos exhibidos en el sitio web
9	Técnicas de respiración	Usuario	El usuario puede ver las técnicas de respiración exhibidas en el sitio web
		• Moderador • Administrador	Puede ver, crear, editar y eliminar las técnicas de respiración exhibidas en el sitio web
10	Ejercicios de meditación	Usuario	El usuario puede ver los ejercicios de meditación exhibidos en el sitio web
		• Moderador • Administrador	Puede ver, crear, editar y eliminar los ejercicios de meditación exhibidos en el sitio web

11	Películas	Usuario	El usuario puede ver las películas exhibidas en el sitio web
		● Moderador ● Administrador	Puede ver, crear, editar y eliminar las películas exhibidas en el sitio web
12	Libros	Usuario	El usuario puede ver los libros exhibidos en el sitio web
		● Moderador ● Administrador	Puede ver, crear, editar y eliminar los libros exhibidos en el sitio web
13	Videojuegos	Usuario	El usuario puede ver los videojuegos exhibidos en el sitio web
		● Moderador ● Administrador	Puede ver, crear, editar y eliminar los videojuegos exhibidos en el sitio web
14	Usuarios	Administrador	Puede ver, crear, editar y eliminar los usuarios dentro del sistema

Tabla 5.2: Síntesis de los casos de uso

Fuente: Propia

5.3 Requisitos

Los requisitos son los requerimientos o necesidades que un sistema o producto debería cumplir para satisfacer a un usuario, en otras palabras, los requisitos definen qué debe hacer el software, cómo debe verse y las condiciones que deben cumplirse para que se considere exitoso. Estos requisitos deben ser identificados, especificados y verificados antes de que un sistema sea desarrollado, normalmente llamada la fase de recopilación de requisitos. En el ámbito de los requisitos se clasifican en dos tipos, requisitos funcionales y requisitos de no funcionales *(Visure Solutions, 2023)*[71].

5.3.1 Requisitos funcionales

Son los requisitos que el usuario final exige específicamente como facilidades básicas que debe ofrecer el sistema. Todas estas funcionalidades deben incorporarse necesariamente al sistema como parte del contrato. Se representan o expresan en forma de entrada que debe darse al sistema, la operación realizada y la salida esperada. Básicamente, son los requisitos declarados por el usuario que se pueden ver directamente en el producto final, a diferencia de los requisitos no funcionales *(GeeksforGeeks, 2022)*[72].

- El sistema permitirá a los usuario cliente crear una cuenta nueva en el sitio web.
- El sistema permitirá a los usuarios iniciar sesión en el sitio web.
- El sistema permitirá a los usuarios cerrar sesión en el sitio web.
- El sistema permitirá a los usuarios gestionar su cuenta en el sitio web.
- El sistema permitirá a los usuarios gestionar sus diarios personales.
- El sistema permitirá a los usuarios gestionar sus notas personales en los diarios.
- El sistema permitirá a los usuarios visualizar el contenido de consejos en el sitio web.
- El sistema permitirá a los usuarios visualizar el contenido de técnicas de respiración en el sitio web.
- El sistema permitirá a los usuarios visualizar el contenido de ejercicios de meditación en el sitio web.
- El sistema permitirá a los usuarios visualizar el contenido de recomendaciones de películas en el sitio web.
- El sistema permitirá a los usuarios visualizar el contenido de recomendaciones de libros en el sitio web.
- El sistema permitirá a los usuarios visualizar el contenido de recomendaciones de videojuegos en el sitio web.
- El sistema permitirá a los administradores y moderadores gestionar el contenido de consejos en el sitio web.
- El sistema permitirá a los administradores y moderadores gestionar el contenido de técnicas de respiración en el sitio web.

- El sistema permitirá a los administradores y moderadores gestionar el contenido de ejercicios de meditación en el sitio web.
- El sistema permitirá a los moderadores gestionar el contenido de recomendaciones de películas en el sitio web.
- El sistema permitirá a los moderadores gestionar el contenido de recomendaciones de libros en el sitio web.
- El sistema permitirá a los moderadores gestionar el contenido de recomendaciones de videojuegos en el sitio web.
- El sistema permitirá a los administradores gestionar a los usuarios en el sitio web.

5.3.2 Requisitos no funcionales

Los requisitos no funcionales son las restricciones o requisitos impuestos al sistema. Especifican los atributos de calidad del software y abordan cuestiones como la escalabilidad, mantenibilidad, rendimiento, portabilidad, seguridad, fiabilidad y entre otras. Los requisitos no funcionales plantean cuestiones vitales para la calidad de los sistemas de software. Si los requisitos no funcionales no se abordan correctamente, los resultados pueden ser usuarios clientes y desarrolladores insatisfechos, *software* incoherente y tiempo/costes excesivos para reparar el *software (GeeksforGeeks, 2022)*[Z3].

- El sistema debe poseer interfaces de usuarios usables, intuitivas y con una experiencia fluida.
- El sistema controlará los permisos otorgados a cada rol de usuario.
- El sistema proporcionará un diseño *responsive*[14] para que su contenido sea visible en la mayor cantidad de dispositivos posibles.
- El sistema proporcionará mensajes de información, de éxito y de error que sean informativos.
- El único rol de usuario con total control del sistema, así como su acceso a la base de datos es el rol de administrador.

[14]Tipo de diseño web que se adapta a cualquier dispositivo, pantalla y resolución. Esto significa que un sitio web puede ser visto de manera clara y fácilmente legible en una computadora de escritorio, una tableta o un teléfono móvil.

- El sistema debe permitir que su información sea de fácil modificación con el propósito de ofrecer el contenido más actualizado posible.
- El sistema contará con un manual de usuario detallado explicando las funcionalidades existentes.

Capítulo 6. Diseño.

6.1 Objetivos del diseño

El diseño web es una parte esencial de la creación de un sitio web exitoso. Esto se debe a que el diseño de un sitio web afecta la usabilidad, la facilidad para encontrar información y el enfoque general de la marca. Un buen diseño asegura que los visitantes encuentren la información relevante de la forma más eficiente posible y que los sitios se vean visualmente atractivos. También los sitios bien diseñados se posicionan mejor en los motores de búsqueda y conducen a aumentar el tráfico diferenciándose de la competencia. Por lo que, un diseño web efectivo es esencial para el éxito de cualquier sitio web *(Coppola, M. ,2023)*[74].

Por otra parte es indispensable entender cual es tu público objetivo, ya que de esta manera sabrás cómo orientar tu sitio web y qué elementos son los más relevantes. En este proyecto el público objetivo son personas que sufren ansiedad social y los elementos relevantes será información y recursos alrededor de este tema central. Por consiguiente, es fundamental comprender que expectativas tienen estos usuarios acerca de tu sitio web, e intentar acercarse a esta visión lo máximo posible.

6.2 Importancia del diseño en la experiencia del usuario

La experiencia de usuario se refiere a la forma en que los usuarios interactúan con un sitio web y su facilidad de uso. Un sitio web con una buena experiencia de usuario debe ser intuitivo, fácil de navegar y debe proporcionar la información que los usuarios necesitan de manera clara y concisa. Una mala experiencia de usuario, por otro lado, puede ser confusa, difícil de navegar y puede no proporcionar la información no deseada. *(Vilardi, R. ,2022)*[75]. Por esta razón, es importante asegurarse de que un sitio web tenga una buena experiencia de usuario. Principalmente apoyándose en una interfaz intuitiva, con un diseño atractivo que sea compatible con diferentes tamaños de pantalla, con un contenido de calidad y fácil entendimiento para los usuarios, además de ofrecer soporte técnico y realizar pruebas de usabilidad para comprobar que la aplicación se comporta como se espera.

6.3 Diseño de los mockups

Aprovechando los recursos que se comentan en el *capítulo 4. Herramientas utilizadas*, se utilizó las imágenes proporcionadas por *Pixel True* para desarrollar los mockups correspondientes de la aplicación. Se decidió orientar el diseño de la web, de forma llamativa para atraer a los usuarios y tener como una gran baza un diseño innovador. Por lo que se empleó la paleta de colores correspondientes del paquete de imágenes de *Pixel True* para dar homogeneidad a la aplicación web.

El uso de un patrón de colores es un elemento clave para cualquier diseño de página web exitoso. Se debe a que los colores pueden tener un gran impacto en la percepción del usuario sobre la calidad de la página web. El uso de un patrón de colores consistente y coherente ayuda a transmitir un mensaje unificado y puede ser una herramienta poderosa para reforzar la marca y la identidad de la aplicación.

Figura 6.3.1: Paleta de colores

Fuente: Propia

Se incluyen en el documento *Anexo I: Diseño web,* los *mockups* de la primera versión diseñada para la aplicación web.

6.4 Cambio del diseño

Al finalizar el desarrollo de los *mockups*, se reflexionó acerca del diseño elegido, y se concluyó que no se ajustaba al propósito que se estaba esperando. Con el objetivo de comprender aún más la importancia del diseño web, y sobre todo orientado a la elección de colores y tipografía, se tomó la decisión hacer un cambio significativo en el diseño de la aplicación. Después de haber leído y comprendido, un informe de psicología del color por *Nick Kolenda*[76] *y* una sección acerca de la tipografía en el sitio web de N*ational Alliance of Mental Illness*[77]*,* se optó por usar colores más relajantes para transmitir tranquilidad. Además, se adaptó la tipografía para que sea clara y fácil de leer para que los usuarios se sientan cómodos al interactuar con la aplicación. Estos cambios ayudarán a mejorar la usabilidad y la experiencia del usuario, creando un ambiente seguro y relajado para aquellas personas que sufren ansiedad social.

El diseño era preciso de perfeccionar debido a que su versión anterior no lograba obtener la repercusión óptima sobre la facilidad de uso y la vivencia del usuario, ya que el tono, la tipografía y la imagen elegidos no se adaptaban a las ideas esenciales que se esperaban sobre la aplicación web. Por lo que tras el estudio, se decidió realizar una evolución sobre los elementos a representar teniendo en cuenta lo aprendido anteriormente.

Primero de todo, se optó por una tipografía clara, de fácil lectura y comprensión como recomienda N*ational Alliance of Mental Illness,* dando como una buena elección la tipografía *Open Sans.* En segundo lugar, la tonalidad del sitio web, tal y como habla en su informe *Nick Kolenda* los colores tienen una gran importancia a la hora de representar emociones y sentimientos, por lo que su decisión iba a estar orientada a sentimientos como la tranquilidad y el afecto, muy vinculado a colores como el púrpura y el morado, desembocando en la paleta de colores de la **figura 6.13**. Por último, se exploraron otras opciones para las imágenes que representarían diferentes recursos dentro del sitio web, descubriendo los recursos de *UnDraw* de *Katerina Limpitsouni,* siendo estos *open-source*[15] y

[15] Código fuente de un programa informático que está disponible de forma gratuita para cualquier usuario, con el fin de permitir que los usuarios desarrollen el software de acuerdo con sus propias necesidades y de distribuirlo como deseen.

fácilmente adaptables a la paleta de colores que se había seleccionado tras la alteración del diseño.

Tras todo este ajustes de diseño y cambio en la representación de los elementos, también se distribuyó los componentes de diferente forma para afinar la imagen de un sitio web moderno y minimalista. En consecuencia, se volvió a generar una serie de *mockups* para observar cómo había mejorado el diseño comparado con la antigua versión y para obtener una visión general de cómo iba a verse la aplicación con esta nueva interfaz planteada. Este cambio de diseño también está recogido en el *Anexo I: Diseño web.*

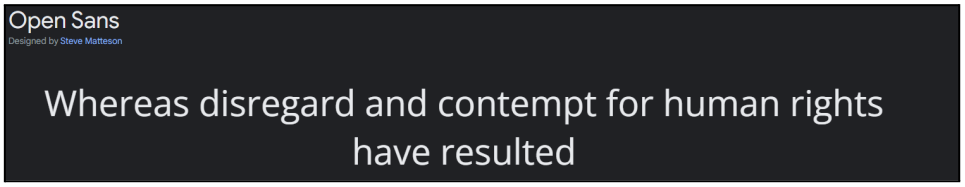


Figura 6.4.1: Tipografía de la aplicación web

Fuente: Propia

Figura 6.4.2: Nueva paleta de colores

Fuente: Propia

Capítulo 7. Desarrollo.

Una vez terminada la fase de análisis y planificación del proyecto, comenzó la fase de desarrollo de la aplicación web. Tal y como se comenta en el *punto 1.5 Peticiones de la comunidad,* se estructuró el proyecto en base a las demandas que se publicaron en los *subreddits,* además de las aportaciones que propuso el experto con el que se debatió acerca del propósito del proyecto. A modo de resumen, se encuentra en la siguiente **figura 7** un esquema general del proyecto.

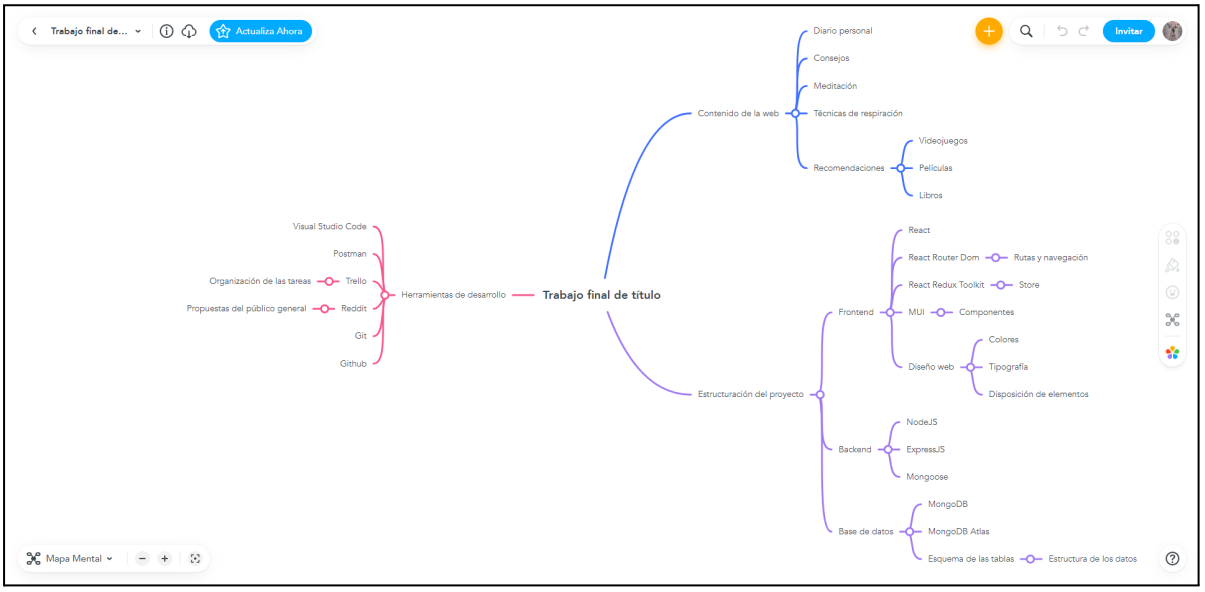


Figura 7: Esquema general del proyecto

Fuente: Propia

A lo largo del proyecto se garantizó detenidamente que cada una de las partes se desarrollará de forma adecuada. Para la primera parte, se usó *React* para construir la interfaz de usuario, para la segunda parte se empleó *Node.js* para crear una *API REST* para el *backend,* para la tercera parte se estableció una combinación de protocolos de comunicación para conectar el cliente y el servidor, para la cuarta se terminaron de afinar algunas funcionalidades, y por último quedó el despliegue de la aplicación. Durante todo el desarrollo del proyecto se utilizó GIT como herramienta de gestión y Github como plataforma de alojamiento de ambos proyectos por separado.

7.1 Integración de los templates

Tras completar los *mockups* mostrados en el punto *6.4 Cambio de diseño*, se empezó a integrarlo a través de *React*. Se decidió que era más ágil empezar por este punto, ya que se contaba con una buena experiencia en trasladar ideas planteadas en *Figma* a una proyecto de *React,* además de que es una práctica muy común realizar primero el diseño y posteriormente agregar toda la funcionalidad.

Primero de todo, se buscó plasmar el diseño sin incluir todavía la funcionalidad al 100% de cada componente, ya que es más sencillo trabajar con datos de prueba como puede ser un archivo local *json* y acomodar los elementos, así como adaptar la distribución a todas las resoluciones posibles con el fin de lograr un diseño *responsive*. Para ello se empleó el uso de la librería *MUI* que proporciona componentes ya creados y dispone de una configuración base para determinadas resoluciones. Asimismo, ya que *React* permite que sus componentes se puedan compartir y reutilizar en otras secciones, se trabajó dicho proyecto alrededor de esta idea intentando optimizar el uso de los componentes.

Un ejemplo de esto se puede encontrar en los componentes de consejos, técnicas de respiración y ejercicios de meditación. Como se observa en la siguientes **tres figuras 7.1.1, 7.1.2 y 7.1.3**, los componentes tienen como componente padre a <*SectionLayot*> cambiando simplemente el *props* qué le pasa en cada caso. Se puede ver cómo se envía el título, una imagen y los elementos relacionados con ese componente. Los *props* se usan principalmente para pasar valores, funciones y otros datos desde un componente padre a uno hijo. Esto ayuda a construir aplicaciones con componentes reutilizables y con una estructura lógica y sencilla. Esta reutilización de componentes también se observa en otras secciones de la aplicación web como son las de libros, películas y videojuegos de la sección de recomendaciones, donde tienen como componente padre a <*ItemsGrid*>.

```jsx
import imgAdviceBackground from "../images/undraw_positive_attitude_re_wu7d.svg"
import SectionLayout from "../components/SectionLayout"
import { useGetAdviceQuery } from "../libraries/api/apiSlice"
import { CheckRequest } from "../components/CheckRequest"

const Advice = () => {
  const { data: advice, isLoading, isError, refetch } = useGetAdviceQuery()
  return (
    <CheckRequest isLoading={isLoading} isError={isError} refetch={refetch}>
      <SectionLayout
        title={"Consejos"}
        img={imgAdviceBackground}
        exercises={advice}
      />
    </CheckRequest>
  )
}

export default Advice
```

Figura 7.1.1: Componente *Advice*

Fuente: Propia

```jsx
import imgBreathingBackground from "../images/undraw_yoga_re_i5ld.svg"
import SectionLayout from "../components/SectionLayout"
import { useGetBreathsQuery } from "../libraries/api/apiSlice"
import { CheckRequest } from "../components/CheckRequest"

const Breathing = () => {
  const { data: breaths, isLoading, isError, refetch } = useGetBreathsQuery()
  return (
    <CheckRequest isLoading={isLoading} isError={isError} refetch={refetch}>
      <SectionLayout
        title={"Respiración"}
        img={imgBreathingBackground}
        exercises={breaths}
      />
    </CheckRequest>
  )
}

export default Breathing
```

Figura 7.1.2: Componente *Breathing*

Fuente: Propia

```jsx
import SectionLayout from "../components/SectionLayout"

import imgMeditationBackground from "../images/undraw_mindfulness_re_mmjó.svg"
import { useGetMeditationsQuery } from "../libraries/api/apiSlice"
import { CheckRequest } from "../components/CheckRequest"

const Meditation = () => {
  const {
    data: meditations, isLoading, isError, refetch
  } = useGetMeditationsQuery()
  return (
    <CheckRequest isLoading={isLoading} isError={isError} refetch={refetch}>
      <SectionLayout
        title={"Meditación"}
        img={imgMeditationBackground}
        exercises={meditations}
      />
    </CheckRequest>
  )
}

export default Meditation
```

Figura 7.1.3: Componente *Meditation*

Fuente: Propia

Volviendo a la utilización de la librería *MUI*, a lo largo del proyecto se encuentran componentes ya creados como *<Box>, <Grid>, <Typography>, <TextField>, <Button>, <Modal>* entre otros. Esto permite agilizar enormemente la producción y creación de los componentes además de que proporciona seguir un patrón en la estructuración de los elementos dentro de la aplicación.

<Grid> es uno de los componentes más empleados en el proyecto, debido a que permite organizar fácilmente los elementos ya que plantea una estructura de 12 columnas, posibilitando la distribución de recursos según sea conveniente y ajustando según la resolución. Dentro de las dimensiones existen *xs, sm, md, lg, xl* donde cada propiedad tiene un ancho máximo para controlar su tamaño en cada situación. Una condición que se da en muchos componentes es que en la dimensión *xs* se coloque un 12 para que ocupe el total de la pantalla en resoluciones pequeñas. Esta configuración está orientada sobre todo a dispositivos móviles, puesto que si existen varios componentes dentro de un componente padre, se requiere que se ajuste a una visualización sencilla para estos dispositivos, como podría ser en forma de columna, uno debajo del otro y ocupando el total de la pantalla. Por otra parte al colocar *md* a 6, si tenemos dos componentes en una resolución media-grande, cada componente ocupa 6 columnas de las 12 en total, es decir cada uno ocupa la mitad en

el componente del *<Grid>*. Estas dimensiones y sus valores permiten controlar sencillamente la manera de distribuir los elementos, así como la capacidad de adaptarse a la mayor cantidad de resoluciones sin tener que poseer una hoja de estilos (CSS) por componente y tener que fijar unas reglas para cada situación. La siguiente **figura 7.1.4** representa cómo se adapta el diseño a distintos dispositivos y resoluciones.

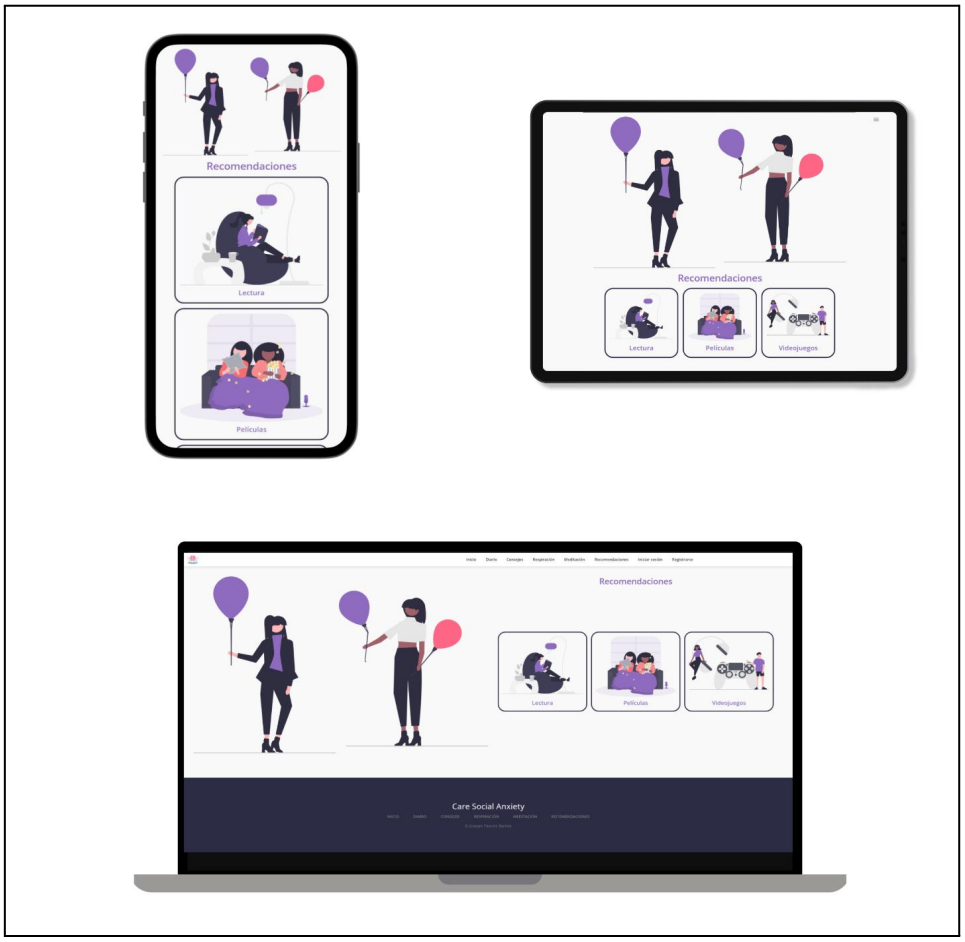


Figura 7.1.4: Visualización de la aplicación web en diferentes dispositivos

Fuente: Propia

Por otra parte, como se comentó en el punto *6.2 Importancia del diseño en la experiencia del usuario,* garantizar una buena experiencia de usuario es crucial para el éxito de la aplicación web. Por eso a través de la librería de *React Router,* se ha implementado un sistema de navegación dinámico, permitiendo al usuario poder cambiar de una sección a otra rápidamente y optimizando los recursos de cada ruta visitada. Además de que ofrece un mayor flexibilidad para el desarrollo como acceso a los parámetros de la URL,

optimización del código y una mayor escalabilidad para implementar nuevas rutas en la aplicación. A través del siguiente esquema de la **figura 7.1.5** se puede ver representado el funcionamiento de la barra de navegación.

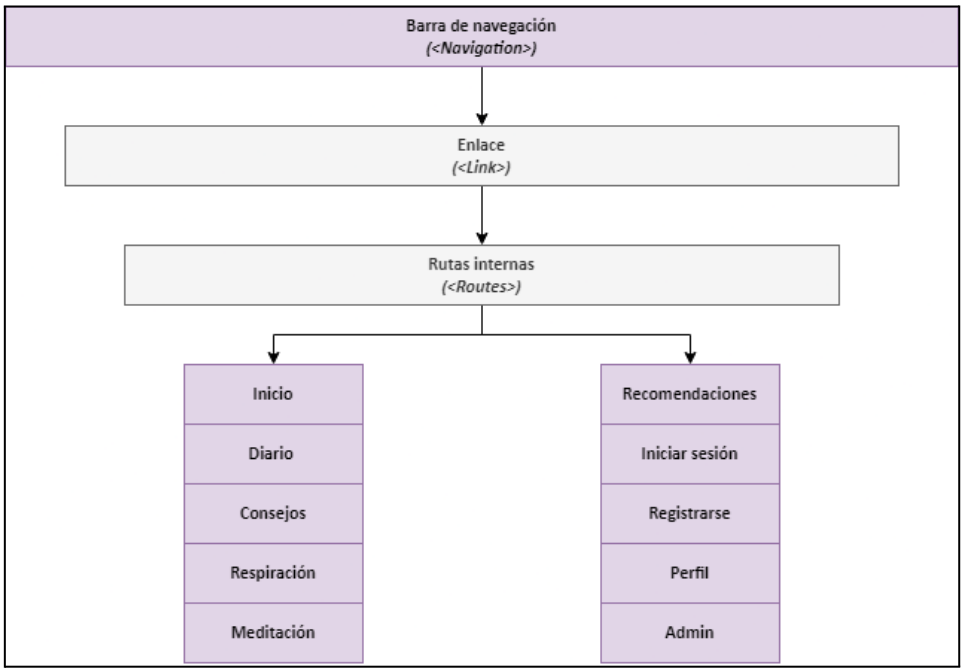


Figura 7.1.5: Comportamiento de la barra de navegación

Fuente: Propia

Estas herramientas claves de *React Router como <Routes>, <Route> y <Link>* se encuentran principalmente en el componente de *<Navigation>, en el <Landing Page> y* en otros elementos diversos que requieren de una redirección a otra sección o recurso del sitio web. Para definir las rutas de la aplicación se emplea el *<Route>* como componente padre y *<Routes>* como componente hijo por cada una de las rutas dinámicas a establecer de la aplicación. También se dispone del componente *<Link>,* muy útil ya que permite navegar a otra página haciendo clic o pulsando sobre él, normalmente colocado en botones y en los elementos de las barras de navegación. Básicamente, *un <Link>* representa un *<a>* accesible con un *href* real que apunta al recurso al que enlaza. Cabe resaltar que toda esta funcionalidad es permitida gracias al englobar todo el sitio web en el componente *<BrowserRouter>,* ya que inyecta propiedades a los componentes para poder acceder al historial de navegación, realizar redirecciones, etc. En las siguientes tres **figuras 7.1.6, 7.1.7, 7.1.8** se puede contemplar las implementaciones en *React*.

```jsx
import React from "react"
import ReactDOM from "react-dom"
import { BrowserRouter } from "react-router-dom"
import { Provider } from "react-redux"

import "./index.css"
import App from "./App"
import { store } from "./libraries/store/store"

ReactDOM.render(
  <React.StrictMode>
    <Provider store={store}>
      <BrowserRouter>
        <App />
      </BrowserRouter>
    </Provider>
  </React.StrictMode>,
  document.getElementById("root")
)
```

Figura 7.1.6: Componente <*BrowserRouter*> englobando a todo el proyecto como componente hijo

Fuente: Propia

```jsx
const LandingPage = () => {
  const userAuth = useSelector((state) => state.user)
  const userSession = sessionStorage.getItem("token")
  const dispatch = useDispatch()

  useEffect(() => {
    if (userAuth.token === undefined && userSession !== null) {
      dispatch(setToken(userSession))
    }
  }, [userAuth, userSession, dispatch])

  return (
    <Routes>
      <Route path="/login" element={<Login />} />
      <Route path="/signup" element={<Signup />} />
      <Route path="/password_reset" element={<ResetPassword />} />
      <Route path="/new_password/:token" element={<NewPassword />} />
      <Route path="/" element={<Navigation />}>
        <Route index element={<Home />} />
        <Route path="journal" element={<Journal />} />
        <Route path="journal/:journalId" element={<JournalDetails />} />
        <Route path="advice" element={<Advice />} />
        <Route path="breathing" element={<Breathing />} />
        <Route path="meditation" element={<Meditation />} />
        <Route path="recommendations" element={<Recommendations />} />
        <Route path="recommendations/movies" element={<Movies />} />
        <Route path="recommendations/books" element={<Books />} />
        <Route path="recommendations/videogames" element={<Videogames />} />
        <Route path="user_id" element={<Profile />} />
        <Route path="admin" element={<AdminDashboard />} />
        <Route path="moderator" element={<ModeratorDashboard />} />
        <Route path="*" element={<Navigate to="/" />} />
      </Route>
    </Routes>
  )
}
```

Figura 7.1.7: Definición de las rutas dinámicas de la aplicación en el componente <*Landing Page*>

Fuente: Propia

```
<div className={styles.navItems}>
          <ul className={styles.list}>
            <Link to="/" className={styles.items}>
              Inicio
            </Link>
            <Link to="/journal" className={styles.items}>
              Diario
            </Link>
            <Link to="/advice" className={styles.items}>
              Consejos
            </Link>
            <Link to="/breathing" className={styles.items}>
              Respiración
            </Link>
            <Link to="/meditation" className={styles.items}>
              Meditación
            </Link>
            <Link to="/recommendations" className={styles.items}>
              Recomendaciones
            </Link>
            {user.token !== undefined ? (
              <>
                <Link to="/user_id" className={styles.items}>
                  Mi Perfil
                </Link>
                <Button
                  onClick={logout}
                  type="submit"
                  variant="text"
                  sx={{ color: "#f94144", fontWeight: "bold" }}
                >
                  Cerrar sesión
                </Button>
              </>
            ) : (
              <>
                <Link to="/login" className={styles.items}>
                  Iniciar sesión
                </Link>
                <Link to="/signup" className={styles.items}>
                  Registrarse
                </Link>
              </>
            )}
            {user.token !== undefined && user.data.roles.find(element => element.name === "admin") ? (
              <>
                <Link to="/admin" className={styles.items}>
                  Admin
                </Link>
              </>
            ) : (
              <></>
            )}
            {user.token !== undefined && user.data.roles.find(element => element.name === "moderator") ? (
              <>
                <Link to="/moderator" className={styles.items}>
                  Moderador
                </Link>
              </>
            ) : (
              <></>
            )}
          </ul>
        </div>
```

Figura 7.1.8: Barra de navegación del componente *Navigation*

Fuente: Propia

7.2 Desarrollo de la API REST

Una vez terminada la implementación de los *templates* en el lado del cliente, se empezó con la creación de la base de datos y los *endpoints* del lado del servidor. Observando las distintas secciones exhibidas en la aplicación web, era necesario analizar y esquematizar qué tipo de recursos iba a mostrar cada sección y sobre todo que estructura iba a tener cada tipo de datos a mostrar **(Figura 7.2.1).**

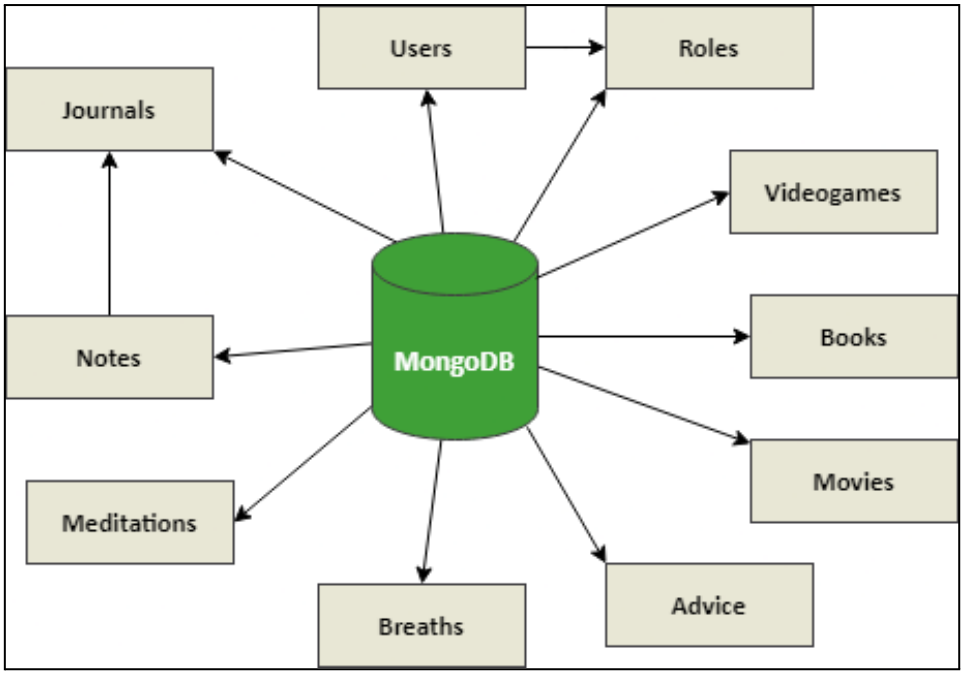


Figura 7.2.1: Esquema de base de datos

Fuente: Propia

Primeramente, se estableció el esquema de la base de datos y las propiedades de cada colección. Al emplear una base de datos no relacional como es *MongoDB* había que declarar correctamente si algunas colecciones eran independientes o tenían dependencia con otras. Un ejemplo de esto, es que las colecciones de *Notes*, tienen una dependencia clara con las colecciones de *Journals*, ya que no puede existir notas de un diario que no existe, o la colección *Users* que tiene una clara dependencia con la colección de *Roles*, puesto que no puede existir un usuario sin un rol asignado.

Todas las creaciones de las colecciones y sus documentos no se ejecutaron de manera manual a través de un gestor de base de datos, como puede ser a través de la aplicación de *MongoDBCompass*, sino que se haría mediante *ExpressJS* y la librería de *mongoose*. Para conseguir dicho cometido se dividió el proyecto del *backend* en varias carpetas separando las funcionalidades de cada archivo. Actualmente se divide en 4 grupos, *controllers*, *middleware*, *models* y *routes* **(Figura 7.2.2)**.

En la carpeta de *controllers*, se agrupan todos los archivos que tienen como misión principal controlar todas las acciones que se realizan en los documentos. En este proyecto,

estas funcionalidades incluyen todo lo relacionado con la obtención, creación, actualización y eliminación de los documentos, llamado normalmente a esta agrupación de funcionalidades como CRUD. De esta manera, existirá un controlador por cada colección de la base de datos.

En la carpeta de *middleware*, se encuentran los archivos que tienen como misión principal proporcionar una capa de abstracción entre los servicios y la lógica de la aplicación. Para esta aplicación web se ha visto necesario tener recursos asignados a la verificación de cuando un usuario quiere iniciar sesión o crearse una cuenta además de verificar que rol tiene asignado. Esta parte ayuda a controlar que no existan usuarios repetidos controlando el nombre de usuario y el correo electrónico, y aún más importante examinando el rol para comprobar los permisos de cada uno y que por ejemplo un usuario de tipo cliente no pueda acceder al contenido del usuario administrador asegurando que no existan problemas graves de seguridad.

En la carpeta de *models*, se localizan los archivos que tienen como misión principal representar el modelo de los documentos en cada colección. Se utilizan para definir una estructura de datos para una colección de documentos en una base de datos. Esta estructura se usa para asegurar que los documentos sean consistentes y contengan los datos correctos. Al igual que los controladores, existirá un modelo por cada colección en la base de datos.

Por último, en la carpeta de *routes* se ubican los archivos que tienen como misión principal proporcionar una manera estructurada de solicitar y recibir recursos desde el servidor. Son un conjunto de URL que permiten a los clientes solicitar y obtener información de la API. Estas rutas se usan para definir cómo exponer los recursos de la API para que los clientes puedan acceder a ellos de forma segura y sencilla. Dado que para una colección existen varias formas de trabajar con sus recursos, cada colección de la base de datos tendrá su propio archivo de configuración para establecer cada ruta y su operación equivalente.

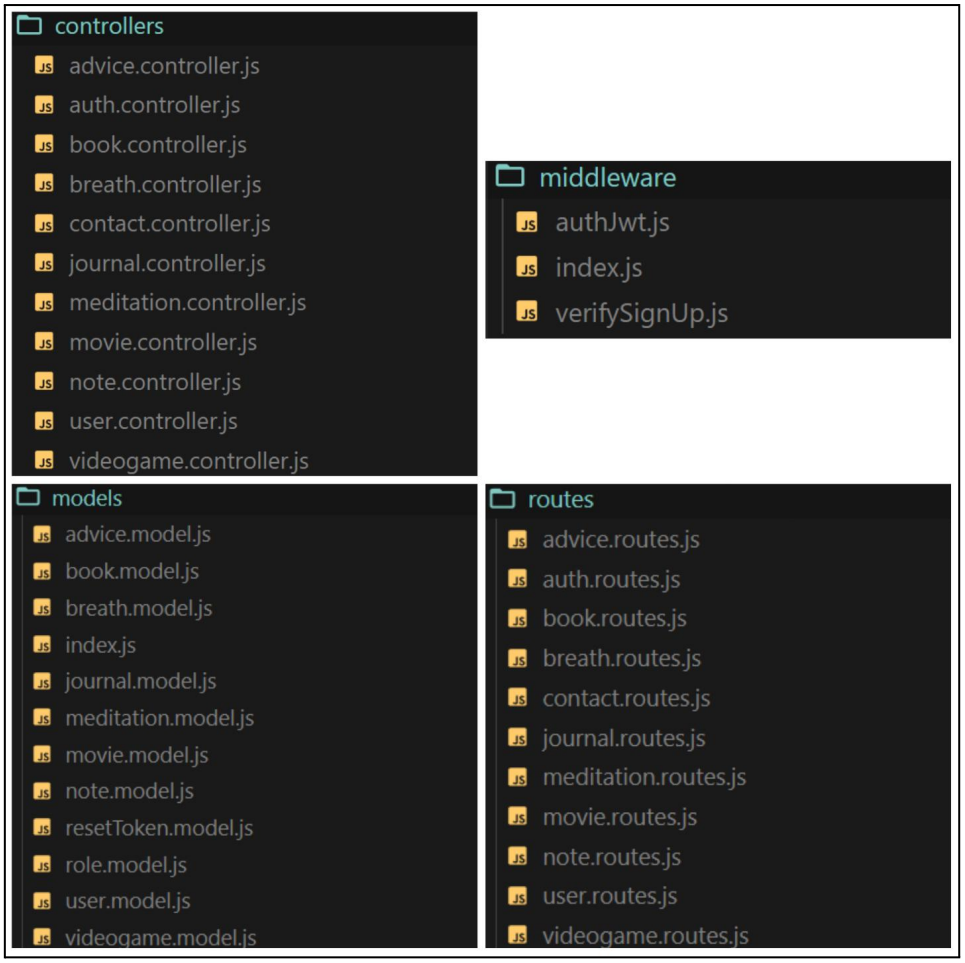


Figura 7.2.2: Esquema del proyecto - *backend*

Fuente: Propia

Tras modularizar correctamente las funciones y objetivos de cada parte, se continuó con la creación de la *API REST.* Este desarrollo es clave para establecer la comunicación entre la parte del cliente y del servidor, permitiendo el intercambio de información de forma fácil y segura.

Una *API REST* funciona de la siguiente manera: un cliente, como una aplicación web, envía una solicitud a un servidor a través de una URL específica y un método *HTTP*[16] (por ejemplo, *GET, POST, PUT, DELETE*). El servidor recibe la solicitud, realiza una acción específica dependiendo del método *HTTP* (por ejemplo, obtener datos, agregar datos, actualizar datos, eliminar datos). Dicha solicitud devuelve una respuesta al cliente en formato JSON o XML,

[16]Protocolo de comunicación para transferencia de hipertexto que se utiliza para establecer conexiones entre servidores web y navegadores web.

que incluye un código de estado *HTTP* (por ejemplo, 200 para éxito, 404 para no encontrado, 500 para error del servidor). El cliente recibe la respuesta y, si el código de estado es 200, realiza la acción solicitada; de lo contrario, recibe una respuesta de error. En la siguiente **figura 7.2.3** se puede ver esquematizado el funcionamiento de una *API REST*.

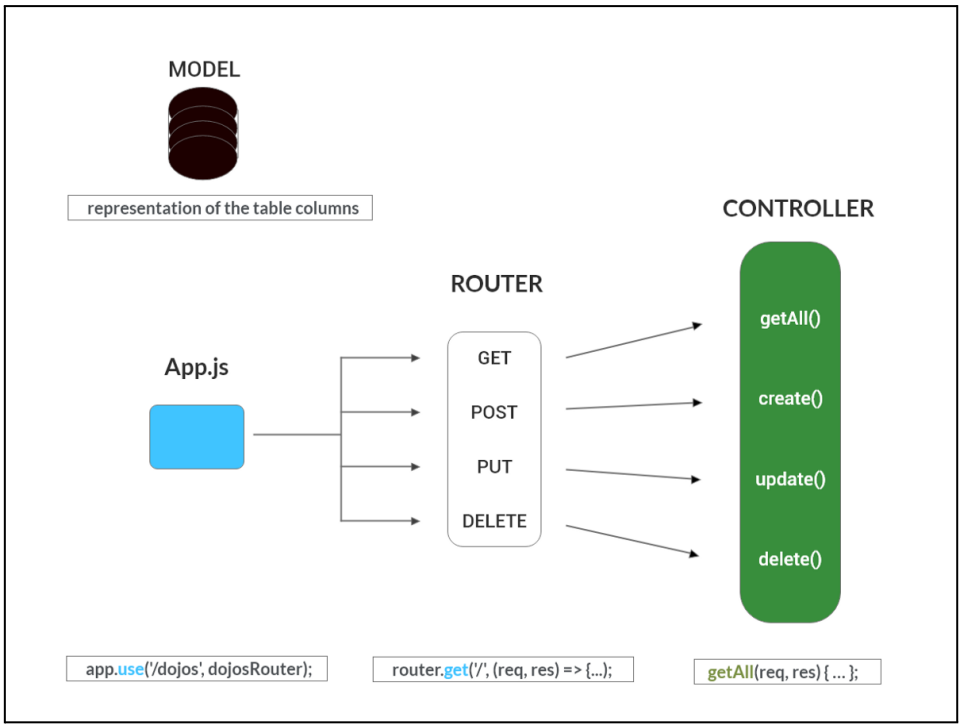


Figura 7.2.3: Esquema del funcionamiento de una *API REST*

Fuente: NodeJS API-Part 5 / Model/Router/Controller structure

Por otra parte, aunque la *API REST* cuente con el funcionamiento básico de un *CRUD* para cada una de las colecciones de la base de datos. Existen ciertas funcionalidades claves orientadas al envío de correos electrónicos, recuperación de la contraseña y a la gestión de permisos de los usuarios.

Para la gestión de correos tenemos dos archivos, el controlador *contact.controller.js* y la ruta *contact.routes.js* para realizar la petición. Ya que la funcionalidad solo está pensada para que los usuarios clientes puedan mandar correos, solo se necesita una petición *POST* para enviar dicha información, tal y como se puede observar en la **figura 7.2.4.**

```
const controller = require("../controllers/contact.controller")
module.exports = function (app) {
  app.use(function (req, res, next) {
    res.header("Access-Control-Allow-Headers", "Origin, Content-Type, Accept")
    next()
  })
  app.post("/api/contact", controller.sendEmail)
}
```

Figura 7.2.4: Estructura de la ruta para el envío de correos electrónicos

Fuente: Propia

En cuanto al controlador, el contenido del correo electrónico se obtiene de los datos enviados en el cuerpo de la solicitud, por lo que haciendo uso del módulo/librería *nodemailer* se envía el correo electrónico devolviendo una respuesta de estado de acuerdo con el resultado. El usuario receptor de dicho correo es uno personalizado que se emplea para la recolección de los *emails* enviados por los usuarios en el lado del cliente. Esta funcionalidad de enviar correos desde el cliente está planteada para que los usuarios puedan aportar ideas y consultar dudas que tengan con la aplicación, además de reportar fallos y errores que encuentren en su funcionamiento. En la siguiente **figura 7.2.5** se observa la estructura del controlador del envío de correos electrónicos.

```
const nodemailer = require("nodemailer")

require("dotenv").config({ path: "../.env" })

exports.sendEmail = async (req, res) => {
  let transporter = nodemailer.createTransport({
    host: "smtp.gmail.com",
    port: 465,
    secure: true,
    auth: {
      user: process.env.USER_EMAIL,
      pass: process.env.USER_PASSWORD,
    },
  })
  let mailOptions = {
    to: process.env.USER_EMAIL,
    from: req.body.email,
    subject: "Contacto - "+req.body.name,
    text:
      `Nombre: ${req.body.name}`+ "\n\n" +
      `Email: ${req.body.email}`+ "\n\n" +
      `Teléfono: ${req.body.number}`
      + "\n\n" +
      req.body.message
  }
  transporter.sendMail(mailOptions, (err, info) => {
    if (!info) {
      return res.status(409).json({ message: err })
    }
    return res.status(200).json({ message: info })
  })
}
```

Figura 7.2.5: Estructura del controlador para el envío de correos electrónicos

Fuente: Propia

Con relación a la recuperación de contraseñas, se encuentra una estructura similar de definición de controlador, modelo y ruta, pero para este caso se ha agrupado dicha funcionalidad del controlador y gestión de las rutas en el apartado de autenticación (*auth.controller.js y auth.routes.js*), debido a que dicha funcionalidad está muy relacionada con la gestión de usuarios y sus correspondientes datos. De todas maneras, sí cuenta con su propio modelo para la estructura del reinicio de contraseña (*resetToken.model.js*).

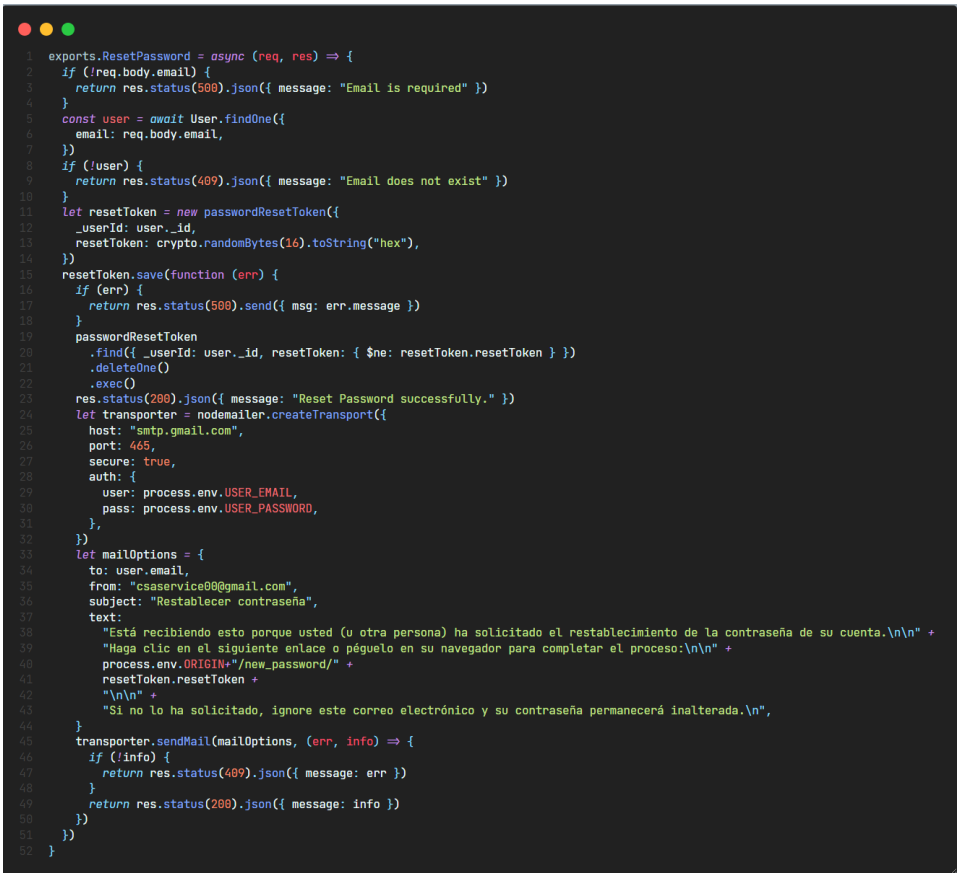
La creación del modelo se ha realizado con el propósito de tener un control sobre las peticiones y de almacenar el *token* de acceso para la renovación de la contraseña. Además, posee una estructura que permite identificar que usuario realizó dicha solicitud y en qué momento la realizó. Su modelo es exhibido en la siguiente **figura 7.2.6.**

```
const mongoose = require('mongoose')
const Schema = mongoose.Schema

const resettokenSchema = new Schema({
  _userId: {
    type: mongoose.Schema.Types.ObjectId,
    required: true,
    ref: "User",
  },
  resetToken: { type: String, required: true },
  createdAt: { type: Date, required: true, default: Date.now, expires: 43200 },
})

let RESET_TOKEN = mongoose.model('passwordResetToken', resettokenSchema)
module.exports = RESET_TOKEN
```

Figura 7.2.6: Modelo resetToken

Fuente: Propia

Para realizar todo el proceso de restablecimiento de contraseña, se dispone de tres principales funciones. ResetPassword es la encargada de enviar la solicitud por correo al destinatario y de asignarle un token para que pueda realizar dicha petición de cambio de contraseña. Además, almacena los datos correspondientes usando el modelo comentado anteriormente, en base de datos. ValidPasswordToken tiene como principal misión comprobar que la URL que está accediendo el usuario es correcta, y que el token que tiene asignado es correcto. Por último, NewPassword certifica que el token sigue siendo el correcto, realiza un cifrado de la contraseña especificada y asigna la nueva contraseña al usuario indicado. Las siguientes tres **figuras 7.2.7, 7.2.8, 7.2.9** manifiestan la implementación de las funcionalidades descritas anteriormente.



```
exports.ResetPassword = async (req, res) => {
  if (!req.body.email) {
    return res.status(500).json({ message: "Email is required" })
  }
  const user = await User.findOne({
    email: req.body.email,
  })
  if (!user) {
    return res.status(409).json({ message: "Email does not exist" })
  }
  let resetToken = new passwordResetToken({
    _userId: user._id,
    resetToken: crypto.randomBytes(16).toString("hex"),
  })
  resetToken.save(function (err) {
    if (err) {
      return res.status(500).send({ msg: err.message })
    }
    passwordResetToken
      .find({ _userId: user._id, resetToken: { $ne: resetToken.resetToken } })
      .deleteOne()
      .exec()
    res.status(200).json({ message: "Reset Password successfully." })
    let transporter = nodemailer.createTransport({
      host: "smtp.gmail.com",
      port: 465,
      secure: true,
      auth: {
        user: process.env.USER_EMAIL,
        pass: process.env.USER_PASSWORD,
      },
    })
    let mailOptions = {
      to: user.email,
      from: "csaservice00@gmail.com",
      subject: "Restablecer contraseña",
      text:
        "Está recibiendo esto porque usted (u otra persona) ha solicitado el restablecimiento de la contraseña de su cuenta.\n\n" +
        "Haga clic en el siguiente enlace o péguelo en su navegador para completar el proceso:\n\n" +
        process.env.ORIGIN+"/new_password/" +
        resetToken.resetToken +
        "\n\n" +
        "Si no lo ha solicitado, ignore este correo electrónico y su contraseña permanecerá inalterada.\n",
    }
    transporter.sendMail(mailOptions, (err, info) => {
      if (!info) {
        return res.status(409).json({ message: err })
      }
      return res.status(200).json({ message: info })
    })
  })
}
```

Figura 7.2.7: Función *ResetPassword*

Fuente: Propia

```
exports.ValidPasswordToken = async (req, res) => {
  if (!req.body.resetToken) {
    return res.status(500).json({ message: "Token is required" })
  }
  const user = await passwordResetToken.findOne({
    resetToken: req.body.resetToken,
  })
  if (!user) {
    return res.status(409).json({ message: "Invalid URL" })
  }
  User.findById({ _id: user._userId })
    .then(() => {
      res.status(200).json({ message: "Token verified successfully." })
    })
    .catch((err) => {
      return res.status(500).send({ msg: err.message })
    })
}
```

Figura 7.2.8: Función *ValidPassword*

Fuente: Propia

```
exports.NewPassword = async (req, res) => {
  passwordResetToken.findOne(
    { resetToken: req.body.resetToken },
    function (err, userToken, next) {
      if (!userToken) {
        return res.status(409).json({ message: "Token has expired" })
      }

      User.findOne(
        {
          _id: userToken._userId,
        },
        function (err, userEmail, next) {
          if (!userEmail) {
            return res.status(409).json({ message: "User does not exist" })
          }
          return bcrypt.hash(req.body.newPassword, 10, (err, hash) => {
            if (err) {
              return res.status(400).json({ message: "Error hashing password" })
            }
            userEmail.password = hash
            userEmail.save(function (err) {
              if (err) {
                return res
                  .status(400)
                  .json({ message: "Password can not reset." })
              } else {
                userToken.remove()
                return res
                  .status(201)
                  .json({ message: "New password set successfully" })
              }
            })
          })
        }
      )
    }
  )
}
```

Figura 7.2.9: Función *newPassword*

Fuente: Propia

Finalmente, dentro de los middlewares que se comentaron anteriormente, se encuentra el *authJWT.js.* Este código se encarga de verificar el *token* y los permisos de los usuarios de una aplicación. Es importante examinar dicho token para determinar qué rol tiene el usuario y qué permisos debe tener. Para verificar dicha información se cuenta esencialmente con 3 funciones. La función *verifyToken* examina el token que realiza la solicitud y determina si esa petición está autorizada o no. Las función de *isAdmin* comprueba si esa petición está realizada por un administrador, permitiendo o no la ejecución de la misma. Por último, *havePermissions* comprueba si el rol es administrador o moderador para que pueda realizar dicha petición, ya que en algunas partes del sitio web la misma acción puede ser realizada por ambos roles, como los *CRUD* de los recursos. Funciones e implementación en las rutas, exhibidas en las siguientes **figuras 7.2.10, 7.2.11, 7.2.12 y 7.2.13.**

```
verifyToken = (req, res, next) => {
  const bearerHeader = req.headers['authorization']
  if (!bearerHeader) {
    return res.status(403).send({ message: "No token provided!" })
  }
  req.token = bearerHeader.split(" ")[1]
  jwt.verify(req.token, config.secret, (err, decoded) => {
    if (err) {
      return res.status(401).send({ message: "Unauthorized!" })
    }
    req.userId = decoded.id
    next()
  })
}
```

Figura 7.2.10: Función *verifyToken*

Fuente: Propia

```
isAdmin = (req, res, next) => {
  User.findById(req.userId).exec((err, user) => {
    if (err) {
      res.status(500).send({ message: err })
      return
    }
    Role.find(
      {
        _id: { $in: user.roles },
      },
      (err, roles) => {
        if (err) {
          res.status(500).send({ message: err })
          return
        }
        for (let i = 0; i < roles.length; i++) {
          if (roles[i].name === "admin") {
            next()
            return
          }
        }
        res.status(403).send({ message: "Require Admin Role!" })
        return
      }
    )
  })
}
```

Figura 7.2.11: Función *isAdmin*

Fuente: Propia

```
havePermissions = (req, res, next) => {
  User.findById(req.userId).exec((err, user) => {
    if (err) {
      res.status(500).send({ message: err })
      return
    }
    Role.find(
      {
        _id: { $in: user.roles },
      },
      (err, roles) => {
        if (err) {
          res.status(500).send({ message: err })
          return
        }
        for (let i = 0; i < roles.length; i++) {
          if (roles[i].name === "moderator" || roles[i].name === "admin" ) {
            next()
            return
          }
        }
        res.status(403).send({ message: "Require Admin/Moderator Role!" })
        return
      }
    )
  })
}
```

Figura 7.2.12: Función *havePermissions*

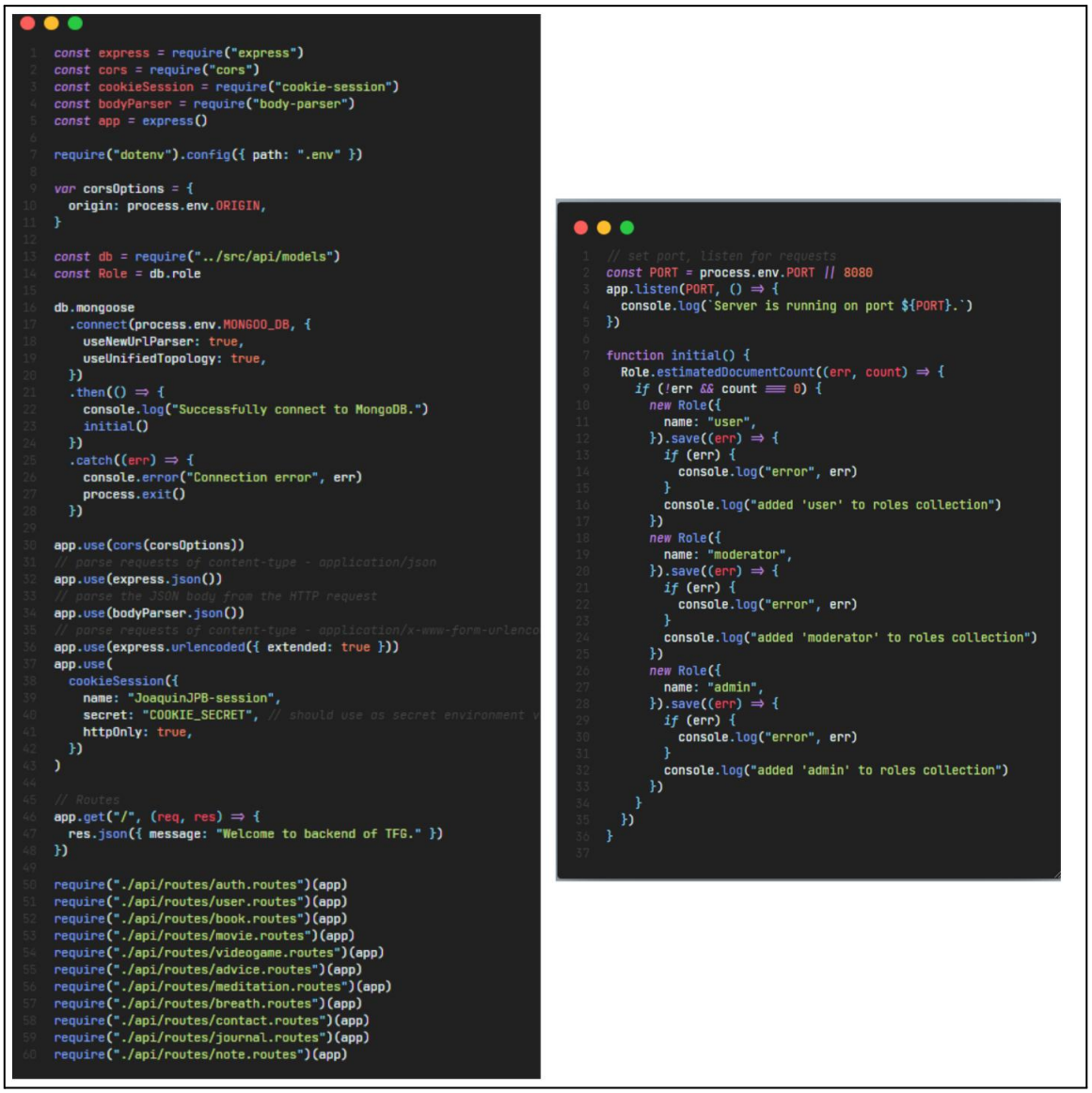
Fuente: Propia

```
const { authJwt } = require("../middleware")
const controller = require("../controllers/user.controller")
module.exports = function (app) {
  app.use(function (req, res, next) {
    res.header("Access-Control-Allow-Headers", "Origin, Content-Type, Accept")
    next()
  })
  app.get(
    "/api/admin/users",
    [authJwt.verifyToken, authJwt.isAdmin],
    controller.getAllUsers
  )
  app.get(
    "/api/admin/users/:nickname",
    [authJwt.verifyToken, authJwt.isAdmin],
    controller.getUserByNickname
  )
  app.post(
    "/api/admin/users",
    [authJwt.verifyToken, authJwt.isAdmin],
    controller.createUser
  )
  app.delete(
    "/api/admin/users/:id",
    [authJwt.verifyToken, authJwt.isAdmin],
    controller.deleteUser
  )
  app.patch(
    "/api/admin/users/:id",
    [authJwt.verifyToken, authJwt.isAdmin],
    controller.updateUser
  )
}
```

Figura 7.2.13: Definición de las rutas para la gestión de usuarios utilizando *middlewares*

Fuente: Propia

Finalmente como archivo general del proyecto, y el encargado de desplegar toda la configuración del servidor y *q*ue administra las peticiones y respuestas entre un cliente y un servidor se encuentra el *server.js.* Utiliza cors para administrar la seguridad y el envío de cookies a través de la sesión. También se conecta a una base de datos MongoDB para almacenar los datos recibidos. Además, contiene las rutas para administrar las peticiones relacionadas con los recursos establecidos. Finalmente, también inicializa roles predeterminados para los usuarios que se conectan al servidor por si no todavía no están creados. En la siguiente **figura 7.2.14** se ve dividida en dos partes la configuración del *server.js* descrita anteriormente.



```
const express = require("express")
const cors = require("cors")
const cookieSession = require("cookie-session")
const bodyParser = require("body-parser")
const app = express()

require("dotenv").config({ path: ".env" })

var corsOptions = {
  origin: process.env.ORIGIN,
}

const db = require("../src/api/models")
const Role = db.role

db.mongoose
  .connect(process.env.MONGOO_DB, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
  })
  .then(() => {
    console.log("Successfully connect to MongoDB.")
    initial()
  })
  .catch((err) => {
    console.error("Connection error", err)
    process.exit()
  })

app.use(cors(corsOptions))
// parse requests of content-type - application/json
app.use(express.json())
// parse the JSON body from the HTTP request
app.use(bodyParser.json())
// parse requests of content-type - application/x-www-form-urlenco
app.use(express.urlencoded({ extended: true }))
app.use(
  cookieSession({
    name: "JoaquinJPB-session",
    secret: "COOKIE_SECRET", // should use as secret environment v
    httpOnly: true,
  })
)

// Routes
app.get("/", (req, res) => {
  res.json({ message: "Welcome to backend of TFG." })
})

require("./api/routes/auth.routes")(app)
require("./api/routes/user.routes")(app)
require("./api/routes/book.routes")(app)
require("./api/routes/movie.routes")(app)
require("./api/routes/videogame.routes")(app)
require("./api/routes/advice.routes")(app)
require("./api/routes/meditation.routes")(app)
require("./api/routes/breath.routes")(app)
require("./api/routes/contact.routes")(app)
require("./api/routes/journal.routes")(app)
require("./api/routes/note.routes")(app)
```

```
// set port, listen for requests
const PORT = process.env.PORT || 8080
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}.`)
})

function initial() {
  Role.estimatedDocumentCount((err, count) => {
    if (!err && count === 0) {
      new Role({
        name: "user",
      }).save((err) => {
        if (err) {
          console.log("error", err)
        }
        console.log("added 'user' to roles collection")
      })
      new Role({
        name: "moderator",
      }).save((err) => {
        if (err) {
          console.log("error", err)
        }
        console.log("added 'moderator' to roles collection")
      })
      new Role({
        name: "admin",
      }).save((err) => {
        if (err) {
          console.log("error", err)
        }
        console.log("added 'admin' to roles collection")
      })
    }
  })
}
```

Figura 7.2.14: Contenido del archivo server.js

Fuente: Propia

7.3 Comunicación entre cliente y servidor

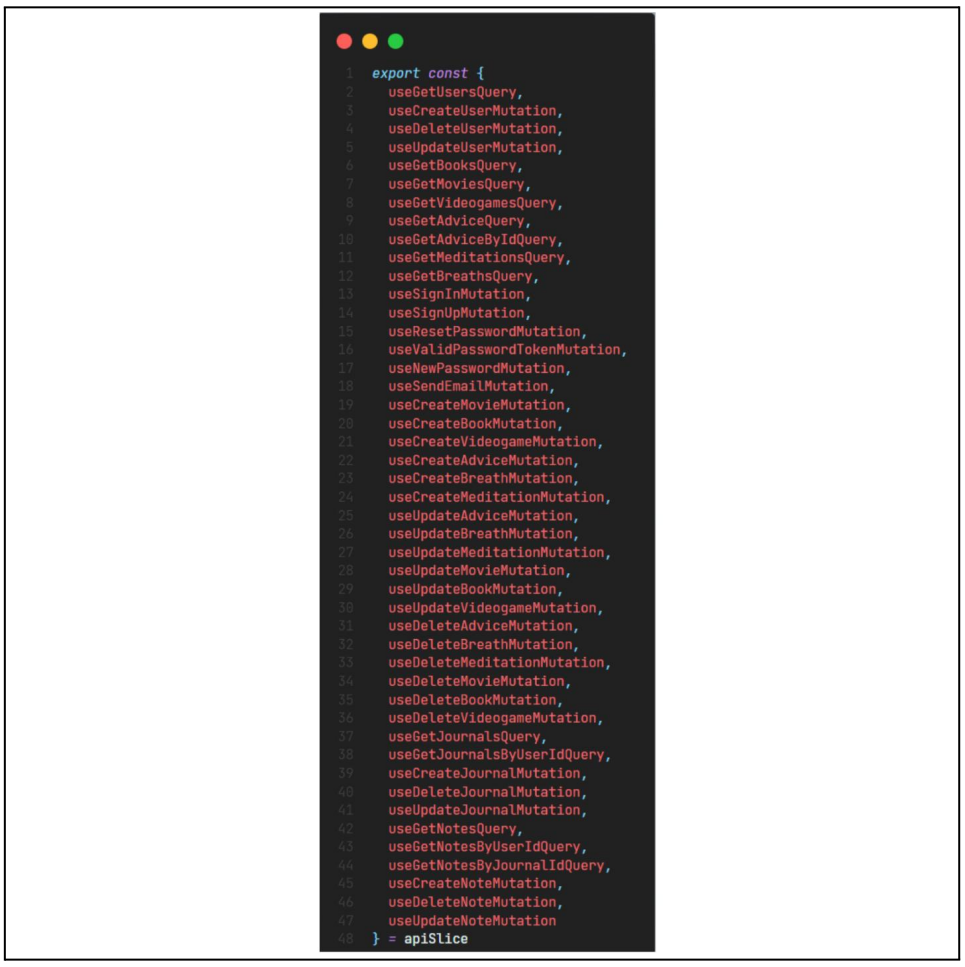
Tras la finalización de la división del lado del servidor y la integración de los *templates* del lado del cliente, quedaba establecer la comunicación de ambas partes y añadir toda la funcionalidad al *frontend*. Para implementar dicha conexión de servidor y cliente, se empleó la librería *React Redux Toolkit* en el lado cliente, debido a que dispone de una funcionalidad llamada *createAPI, la cual* ofrece un sistema de definición de conjuntos de *endpoints* que describen cómo se recuperan datos desde una API permitiendo funciones asíncronas y configuración para la obtención y transformación de los datos. En consecuencia, se generó un archivo *apiSlice.jsx* teniendo como objetivo principal albergar toda la estructura del conjunto de *endpoints* definidos en el *backend*. En la siguiente **figura 7.3.1** se puede observar un pequeño fragmento de este archivo.

```jsx
import { createApi, fetchBaseQuery } from "@reduxjs/toolkit/query/react"

export const apiSlice = createApi({
  reducerPath: "api",
  baseQuery: fetchBaseQuery({
    baseUrl: process.env.REACT_APP_API_URL,
    prepareHeaders: (headers) => {
      const token = sessionStorage.getItem("token")
      if (token) {
        headers.set("authorization", `Bearer ${token}`)
      }
      return headers
    },
  }),
  endpoints: (builder) => ({
    getUsers: builder.query({
      query: () => ({
        url: "admin/users",
      }),
    }),
    createUser: builder.mutation({
      query: (payload) => ({
        url: "admin/users",
        method: "POST",
        body: payload,
      }),
    }),
    deleteUser: builder.mutation({
      query: (payload) => ({
        url: `admin/users/${payload}`,
        method: "DELETE",
        body: payload,
      }),
    }),
    updateUser: builder.mutation({
      query: (payload) => ({
        url: `admin/users/${payload._id}`,
        method: "PATCH",
        body: payload,
      }),
    }),
```

Figura 7.3.1: Fragmento del archivo *apiSlice.jsx*

Fuente: Propia

Por tanto en cada componente donde sea necesaria la interacción con la API existirá un hook de *React* encargado de la operación correspondiente. Además será personalizado por cada colección y por cada acción a realizar, utilizando para la obtención de datos un *GET*, para la creación un *POST*, para la actualización un *PATCH* y un *DELETE* para la eliminación. En la siguiente **figura 7.3.2** se puede contemplar los *hooks* necesarios para realizar todas las operaciones esenciales con la API REST.



```
export const {
  useGetUsersQuery,
  useCreateUserMutation,
  useDeleteUserMutation,
  useUpdateUserMutation,
  useGetBooksQuery,
  useGetMoviesQuery,
  useGetVideogamesQuery,
  useGetAdviceQuery,
  useGetAdviceByIdQuery,
  useGetMeditationsQuery,
  useGetBreathsQuery,
  useSignInMutation,
  useSignUpMutation,
  useResetPasswordMutation,
  useValidPasswordTokenMutation,
  useNewPasswordMutation,
  useSendEmailMutation,
  useCreateMovieMutation,
  useCreateBookMutation,
  useCreateVideogameMutation,
  useCreateAdviceMutation,
  useCreateBreathMutation,
  useCreateMeditationMutation,
  useUpdateAdviceMutation,
  useUpdateBreathMutation,
  useUpdateMeditationMutation,
  useUpdateMovieMutation,
  useUpdateBookMutation,
  useUpdateVideogameMutation,
  useDeleteAdviceMutation,
  useDeleteBreathMutation,
  useDeleteMeditationMutation,
  useDeleteMovieMutation,
  useDeleteBookMutation,
  useDeleteVideogameMutation,
  useGetJournalsQuery,
  useGetJournalsByUserIdQuery,
  useCreateJournalMutation,
  useDeleteJournalMutation,
  useUpdateJournalMutation,
  useGetNotesQuery,
  useGetNotesByUserIdQuery,
  useGetNotesByJournalIdQuery,
  useCreateNoteMutation,
  useDeleteNoteMutation,
  useUpdateNoteMutation
} = apiSlice
```

Figura 7.3.2: *Hooks definidos en apiSlice.jsx*

Fuente: Propia

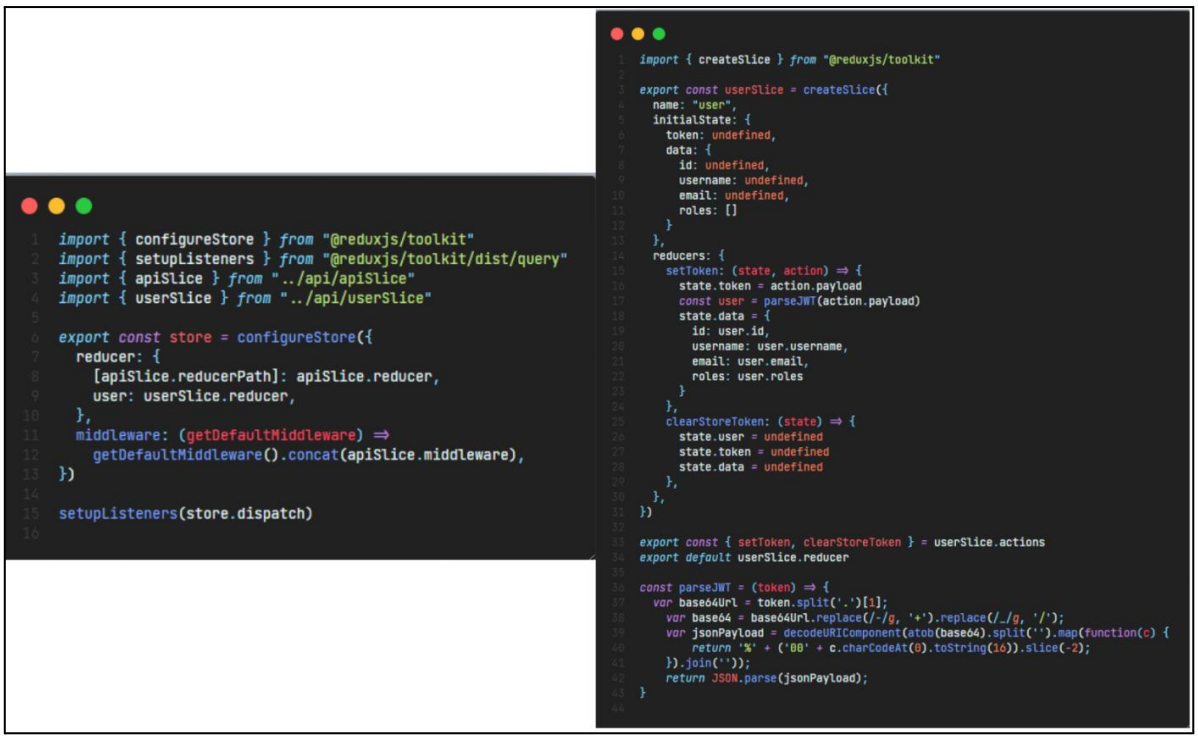
Un ejemplo representativo de esto, se puede observar en las secciones de *Consejos*, *Respiración* y *Meditación* donde a través de estos hooks se realiza la operación de obtención de datos mediante un *GET*, tal y como se puede observar en la siguiente **figura 7.3.3.**

Figura 7.3.3: Obtención de los recursos mediante *Hook*

Fuente: Propia

7.4 Implementación de funcionalidades finales

En el proyecto, se ha implementado un sistema de gestión de inicio de sesión de usuarios a través de *React, Redux Toolkit* usando la *store* y un *slice* personalizado orientado al usuario. La *store* de *Redux Toolkit* es una herramienta de almacenamiento de estado global para aplicaciones de React. Esta herramienta se utiliza para almacenar todos los datos de la aplicación, como los estados de las entidades, los datos del usuario, etc. Además, se pueden crear "slices" para cada entidad relacionada con la aplicación, como usuarios, productos, etc. Esto permite una mejor organización de los datos de la aplicación y un mejor control sobre la forma en que se almacenan y se accede a ellos **(Figura 7.4.1).**



```
import { configureStore } from "@reduxjs/toolkit"
import { setupListeners } from "@reduxjs/toolkit/dist/query"
import { apiSlice } from "../api/apiSlice"
import { userSlice } from "../api/userSlice"

export const store = configureStore({
  reducer: {
    [apiSlice.reducerPath]: apiSlice.reducer,
    user: userSlice.reducer,
  },
  middleware: (getDefaultMiddleware) =>
    getDefaultMiddleware().concat(apiSlice.middleware),
})

setupListeners(store.dispatch)
```

```
import { createSlice } from "@reduxjs/toolkit"

export const userSlice = createSlice({
  name: "user",
  initialState: {
    token: undefined,
    data: {
      id: undefined,
      username: undefined,
      email: undefined,
      roles: []
    }
  },
  reducers: {
    setToken: (state, action) => {
      state.token = action.payload
      const user = parseJWT(action.payload)
      state.data = {
        id: user.id,
        username: user.username,
        email: user.email,
        roles: user.roles
      }
    },
    clearStoreToken: (state) => {
      state.user = undefined
      state.token = undefined
      state.data = undefined
    },
  },
})

export const { setToken, clearStoreToken } = userSlice.actions
export default userSlice.reducer

const parseJWT = (token) => {
  var base64Url = token.split('.')[1];
    var base64 = base64Url.replace(/-/g, '+').replace(/_/g, '/');
    var jsonPayload = decodeURIComponent(atob(base64).split('').map(function(c) {
        return '%' + ('00' + c.charCodeAt(0).toString(16)).slice(-2);
    }).join(''));
    return JSON.parse(jsonPayload);
}
```

Figura 7.4.1: Estructura de la store y de useSlice

Fuente: Propia

Gracias a esto, mediante el uso de la *apiSlice* que se comentó anteriormente para realizar la petición al servidor y validar las credenciales del usuario, se guarda el *token* de autenticación en la *store* y se redirecciona al usuario a la página principal una vez que se ha validado el inicio de sesión. Después, por medio del *hook useSelector* se accede al estado del usuario guardado en la *store*. Esta herramienta permite recuperar los datos del usuario guardados en la *store*, como el ID, el nombre de usuario, el correo electrónico, etc., lo cual es esencial para llevar a cabo cualquier acción relacionada con el usuario.

Por ejemplo, este *hook* es utilizado en el componente *<Navigation>* para mostrar la opción de navegación a la sección de *Mi Perfil* ocultando el inicio de sesión y registrarse, además de ofrecer la posibilidad de cerrar sesión. También se usa en el componente *<Journal>,* puesto que la funcionalidad de los diarios personales solo está pensado para los usuarios registrados. Además, permite saber el rol del usuario y habilitar el panel de administración para los moderadores y administradores de la aplicación web. En las siguientes dos **figuras 7.4.2 y 7.4.3** se puede examinar las diferencias de algunos componentes cuando un usuario ha iniciado sesión o no.

Figura 7.4.2: Cambio de barra de navegación si el usuario está registrado o no

Fuente: Propia

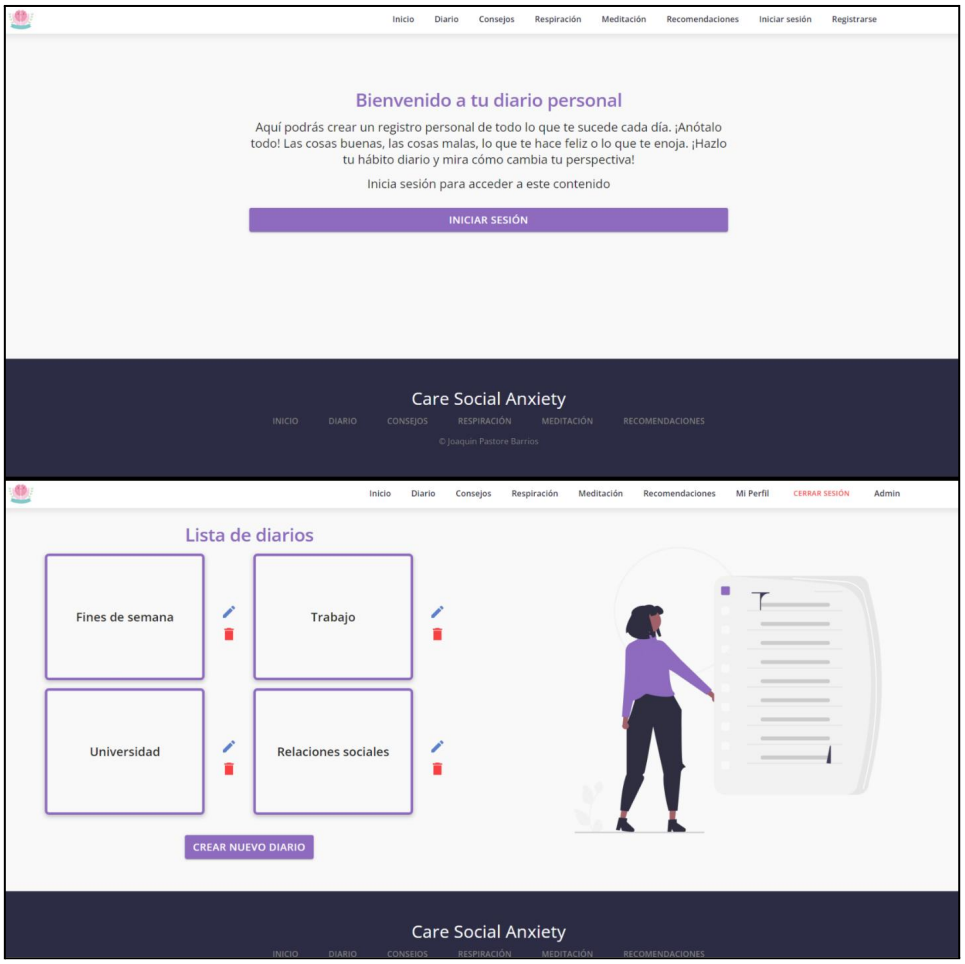


Figura 7.4.3: Componente <Journal> si el usuario está registrado o no

Fuente: Propia

Para el control de los formularios dentro de la aplicación web se ha empleado *useForm*, un *Hook* personalizado que permite gestionar fácilmente toda la funcionalidad requerida en los formularios. Este *useForm* está presente en todos los formularios donde es necesaria una validez de los datos. En el inicio de sesión y registro, comprueba los campos de correo electrónico, nombre de usuario y contraseña, con el objetivo de que cumplan una reglamento dado garantizando que como mínimo exista una base de seguridad para los

datos. Por ejemplo, en este caso, no permite que los campos estén vacíos, deben tener una longitud mínima y deben cumplir con una expresión regular dada para el caso en concreto.

En relación al correo electrónico por ejemplo su expresión regular se centra en que contenga un @ en el campo rellenado. Para la contraseña se busca que los usuarios se impliquen más en formatos seguros, obligándolos a que exista como mínimo una mayúscula, una minúscula, un numéro, un carácter especial y una longitud mínima de 8 caracteres. Para el nombre de usuario el formato es un poco más flexible y simplemente se busca que tenga una longitud mínima. Además, desde el *backend* se controla que no exista nombres de usuarios y correos electrónicos repetidos.

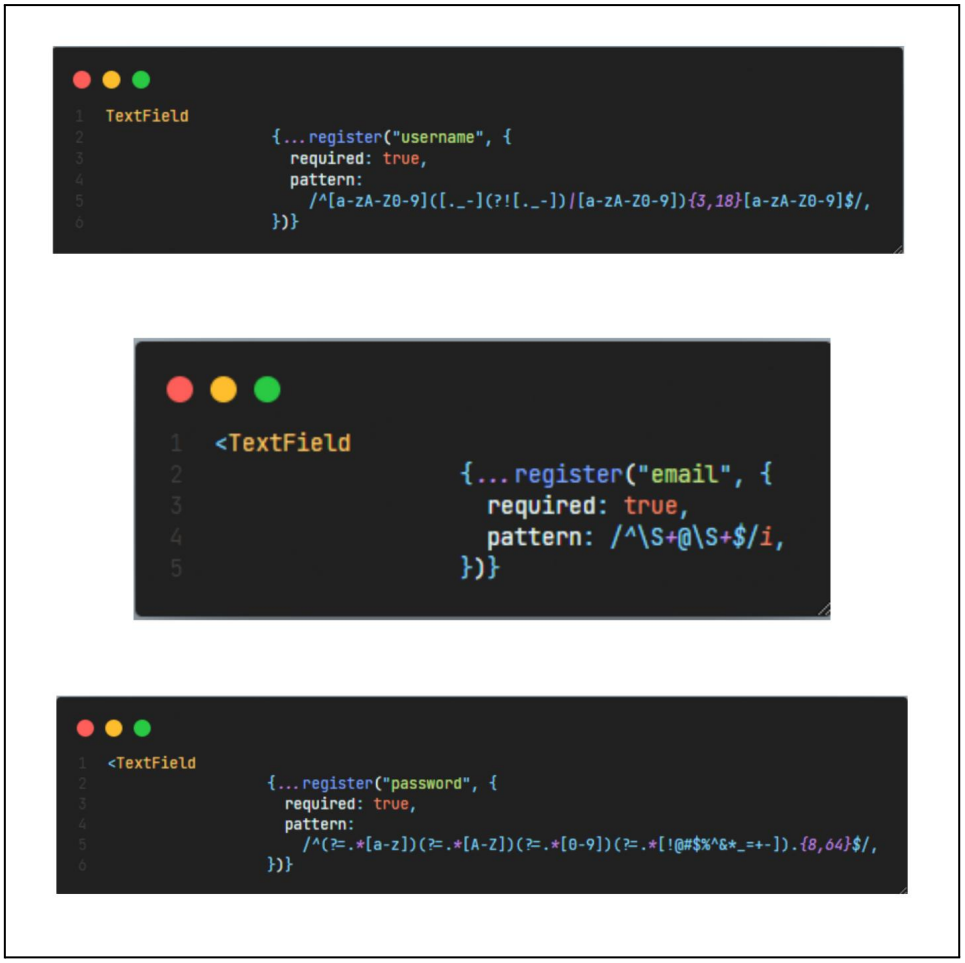


Figura 7.4.4: Campo de los formularios con su expresión regular

Fuente: Propia

En consideración de la experiencia de usuario, se ha utilizado la librería de *React-Toastify* para personalizar los mensajes de éxito y de error. Por tanto, por cada acción dentro de la aplicación web donde el usuario deba esperar una respuesta, aparecerá un mensaje en la esquina superior derecha indicando el estado de su respuesta. En la siguiente **figura 7.4.3** se muestra un ejemplo de esta implementación.

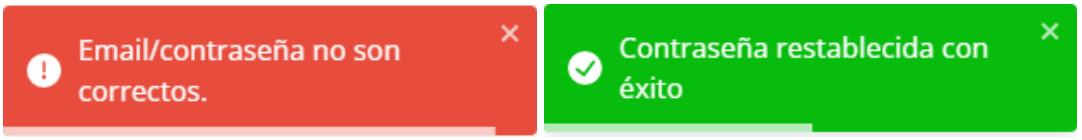


Figura 7.4.5: Mensajes de éxito y error

Fuente: Propia

En lo que respecta a la sección del diario personal, se dispone principalmente de dos librerías externas. Por una parte, es esencial mostrar un calendario con el que interactuar para apuntar tu día a día. Para esta funcionalidad se emplea React-calendar, puesto que ofrece un componente ya creado que permite seleccionar el día, mes y año, además de su fácil configuración e implementación.

Figura 7.4.6: Calendario de la sección *Diario*

Fuente: Propia

Por otra parte, para reproducir gráficamente las notas que vayan realizando los usuarios y con vista de dar una mejor representación del trabajo diario, se ha empleado la

librería de *React-chartjs-2*. Dado que la librería ya ofrece componentes ya creados y con su propia implementación, se ha seleccionado el componente de *<Bar>* para la representación de barras verticales y *<Doughnut>* para la representación en formato donut. En las barras verticales se verá el total de días según el estado de ánimo fijado por cada nota, y en el donut además del total de días, representará un porcentaje del total.

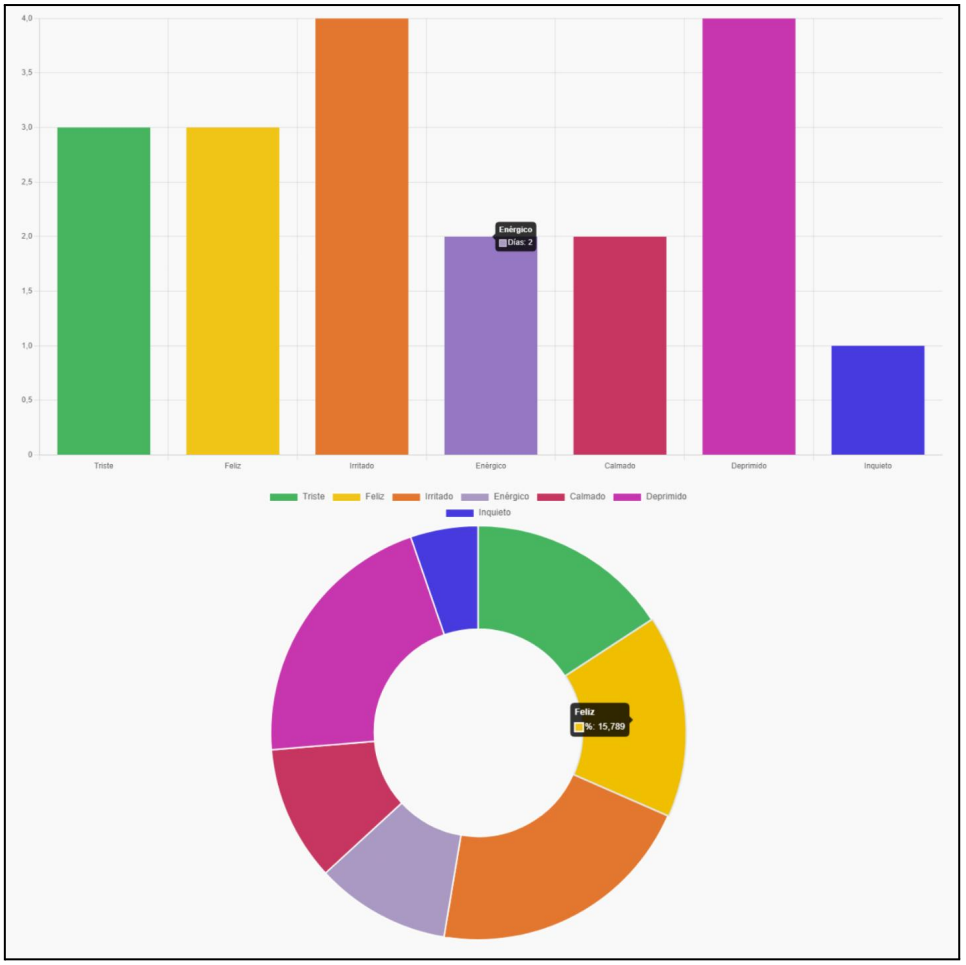


Figura 7.4.7: Diagramas de barras y tipo donut

Fuente: Propia

Para explicar detalladamente toda la funcionalidad restante de la aplicación web y sus características, acceder al *Anexo II: Manual de usuario.*

7.5 Despliegue de la aplicación

Aprovechando de que la plataforma de *MongoDB* cuenta con una sección llamada *MongoDB Atlas*, donde permite subir tu base de datos a la nube, se empezó a realizar el despliegue en dicho recurso. Para ello, primero se creó una cuenta en la plataforma y se estableció un nuevo clúster. Después, se seleccionó la opción de "Importar una base de datos existente" y se seleccionó el archivo JSON o CSV con la base de datos.

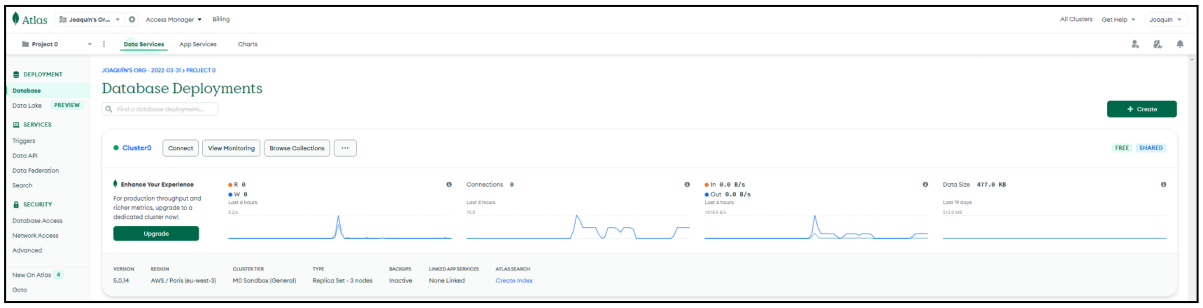


Figura 7.5.1: Cluster creado en la plataforma MongoDB Atlas

Fuente: Propia

Una vez creado el clúster con su correspondiente importación de los datos, desde el lado del servidor se tuvo que cambiar la configuración con la que se accedía a la base de datos. Actualmente se accede al servidor de *MongoDB Atlas*, mediante el uso de variables de entorno donde se encuentran las credenciales para acceder a dicho clúster y a la base de datos en particular. Cabe destacar que en esta configuración de variables de entorno se encuentra, a parte del acceso a la base de datos, las credenciales para la cuenta de correo donde se reciben y envían los mensajes de la aplicación, el puerto y la URL permitida mediante el *CORS* para que se pueda hacer uso de esta *API REST.*

Figura 7.5.2: Variables de entorno *API REST*

Fuente: Propia

Para el despliegue de la *API REST*, se siguió el tutorial de José Antonio Muñoz Jiménez (@jamj2000 en Github)[78] que tiene alojado en la plataforma de Github, donde explica cómo desplegar un BACKEND con Node, Express y MongoDB en distintas plataformas. Para esta ocasión se ha seguido la guía para su subida en *Render* ya que se tenía previa experiencia con la plataforma y su configuración es sencilla. Para ello, se necesitaban las credenciales de Github, para acceder a la plataforma. Una vez que se concedieron los permisos necesarios, se buscó el repositorio donde se alojaba el proyecto. Después, se añadieron las variables de entorno necesarias. Finalmente, se realizó la configuración final y la subida de la API al servidor.

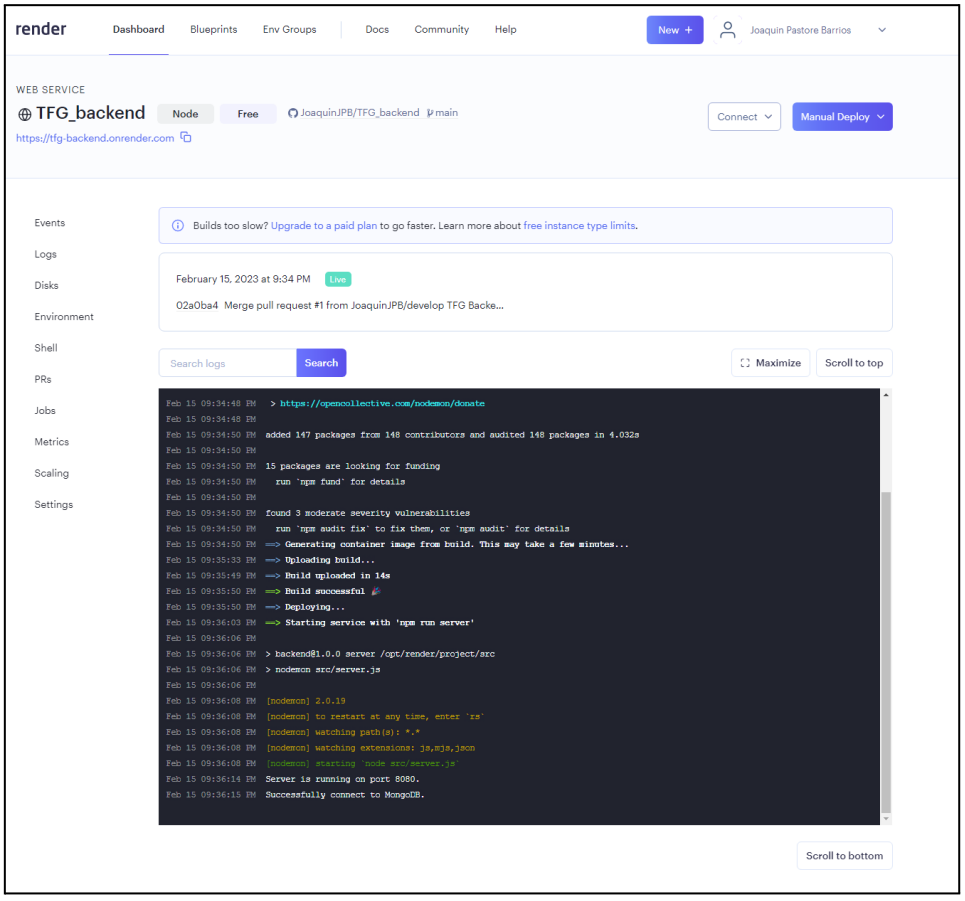


Figura 7.5.3: Despliegue de proyecto en *Render*

Fuente: Propia

Para el lado del cliente, se utilizó la plataforma de *Vercel* ya que dispone de una mejor integración para proyectos orientados con frameworks de frontend. Además de que, durante el desarrollo Render proporcionó algunos problemas con este proyecto en concreto.

Siguiendo la dinámica anterior con la subida de la *API REST,* en la plataforma de Vercel se continuaría de la misma forma. Para resumir sería, conectar tu cuenta con Github, darle los permisos requeridos, añadir las variables de entorno y desplegar el proyecto.

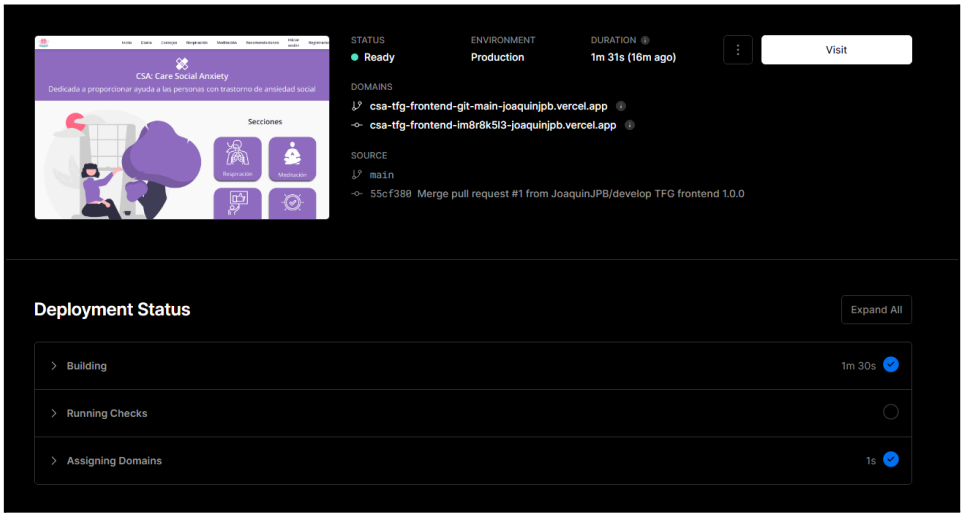


Figura 7.5.3: Despliegue de proyecto en *Vercel*

Fuente: Propia

Para finalizar se empleó la extensión de Google Chrome, *Lighthouse* debido a que permite evaluar de manera objetiva y detallada el rendimiento, accesibilidad, buenas prácticas y optimización para motores de búsqueda de una aplicación web. Al utilizar herramientas como *Lighthouse*, se pueden optimizar la velocidad de carga de la página, la experiencia del usuario y la accesibilidad para personas con discapacidades. También se pueden mejorar las prácticas de SEO y aumentar la visibilidad y el tráfico de la aplicación web en los motores de búsqueda.

Observando la siguiente **figura 7.5.4,** se saca varias conclusiones. Respecto a buenas prácticas, la aplicación se encuentra en el máximo de puntuación, garantizando que se cumple con los estándares de calidad y de seguridad esperados y que a lo largo del desarrollo se ha optimizado el código para cumplir con dicha función.

En cuanto al SEO, tiene también la misma puntuación, esto es debido principalmente a que todos los elementos de la web están adaptados correctamente para los motores de búsquedas.

En relación a la accesibilidad, se puede observar como la extensión da una serie de consejos que se pueden aplicar para mejorar dicho apartado. Por ejemplo, exhibe que no existe una nomenclatura clara para los botones, debido al empleo de los botones de *MUI* y que los elementos y el fondo no cumple con un estándar de relación de contraste suficiente.

Por último, el apartado de rendimiento comenta que existen ciertos problemas de cargas con las imágenes, y que en sitios donde la conexión no es muy estable, puede llevar a que la carga del sitio web se ralentice considerablemente. Además, suma la carga de librerías de React, que muchas de sus partes no son realmente necesarias para la aplicación web.

Por tanto gracias a *Lighthouse*, se puede conocer al detalle las próximas mejoras que se deberían considerar para aumentar el rendimiento y la accesibilidad además de las futuras funcionalidades que se planean para el futuro.

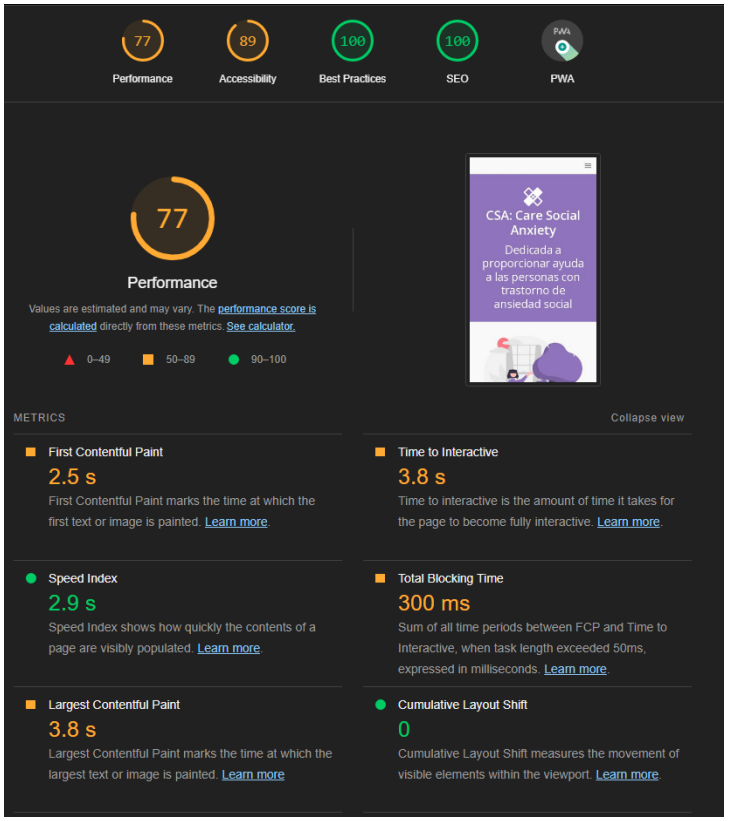


Figura 7.5.4: Análisis realizado por *Lighthouse*

Fuente: Propia

Capítulo 8. Resultados, conclusiones y futuras mejoras.

8.1 Resultados

Las conclusiones de la aplicación web se segmentan en tres categorías: usabilidad, contenido y resultados finales. Además, se sugieren varias mejoras que se pueden implementar para mejorar el servicio y aumentar la satisfacción de los usuarios.

En cuanto a la usabilidad, la aplicación fue diseñada con una interfaz intuitiva que facilita el acceso a los contenidos y herramientas de la misma. También se hizo especial hincapié en la seguridad y privacidad de los usuarios, garantizando que los datos de los usuarios no se compartieran con terceros. Además se atendió a la demanda de los usuarios y se orientó la aplicación en torno a esta ideas y sus correspondientes funcionalidades.

Con respecto al contenido, el sitio web ofrece una amplia variedad de herramientas y materiales de apoyo para ayudar a personas que sufren del trastorno en cuestión. Estos recursos incluyen información detallada sobre los síntomas, causas y tratamientos del trastorno, así como estrategias prácticas para gestionar los desafíos asociados con él. Además, el sitio se compromete a seguir mejorando y actualizando su contenido en el futuro, con el objetivo de ofrecer siempre la información más actualizada y útil posible para su audiencia.

Por último, los resultados finales fueron satisfactorios. Se consiguió construir una herramienta sostenible, versátil y escalable, usando las últimas tecnologías y ofreciendo una solución clara y sencilla al problema planteado. Además, para asegurar que la aplicación cumplía con los estándares de usabilidad y satisfacía las necesidades de los usuarios, se llevó a cabo una prueba de usuario con un grupo seleccionado de usuarios. Los resultados de esta prueba fueron muy positivos, ya que los usuarios encontraron la aplicación fácil de usar y

cumplió con sus expectativas. Los comentarios y sugerencias de los usuarios también fueron muy valiosos para mejorar aún más la aplicación.

8.2 Futuras mejoras

Aunque la aplicación ha obtenido resultados satisfactorios, se pueden implementar mejoras para optimizar su servicio. En primer lugar, se podría agregar una sección de foros para que los usuarios puedan compartir sus experiencias y brindarse apoyo mutuo. Esto fomentaría la creación de una comunidad dentro de la aplicación y propiciaría la solidaridad entre los usuarios.

Asimismo, se podría mejorar la usabilidad de la aplicación, haciendo que la interfaz sea más intuitiva y agregando filtros que faciliten la búsqueda de contenido relevante. Además, se podrían incorporar nuevas herramientas, como tests de autoevaluación y contenido educativo sobre la ansiedad social, como cursos y certificados.

Por último, sería útil incorporar tecnologías avanzadas como la inteligencia artificial para proporcionar asesoramiento personalizado a los usuarios. Esto permitiría a los usuarios obtener consejos y sugerencias especializadas y adaptadas a sus necesidades individuales, lo que sería especialmente útil cuando no hay profesionales disponibles en ese momento.

8.3 Conclusiones finales

En conclusión, una aplicación web para personas que sufren de ansiedad social puede ser una herramienta valiosa para brindar apoyo y ayuda a aquellos que luchan con esta afección. La ansiedad social es una condición muy común y debilitante que puede limitar la capacidad de una persona para interactuar socialmente, lo que puede tener un impacto negativo en su bienestar emocional y su calidad de vida en general.

Una aplicación web diseñada específicamente para abordar la ansiedad social puede proporcionar información útil, recursos y técnicas de afrontamiento para ayudar a las

personas a manejar sus síntomas y mejorar su capacidad para interactuar con los demás de manera efectiva y saludable.

Además, una aplicación web puede ser accesible en cualquier momento y lugar, lo que significa que las personas pueden recibir apoyo y recursos cuando lo necesiten, incluso fuera de los horarios de atención de los profesionales de la salud mental.

En resumen, una aplicación web para la ansiedad social puede ser una herramienta útil y efectiva para apoyar a aquellos que luchan con esta afección y mejorar su bienestar emocional y calidad de vida. Sin embargo, es importante tener en cuenta que una aplicación web no puede reemplazar la atención médica y el apoyo profesional de los psicólogos o terapeutas, y que siempre es recomendable buscar ayuda si se siente abrumado por la ansiedad social.

Capítulo 9. Bibliografía.

[1] National Institute of Mental Health (s. f.). Trastorno de ansiedad social: Más allá de la simple timidez. https://www.nimh.nih.gov/health/publications/espanol/trastorno-de-ansiedad-social-mas-alla-de-la-simple-timidez

[2] Online, P. (2021, 10 mayo). La influencia del COVID-19 en la aparición de fobias. Psiquion - Plataforma de psicología online. https://www.psiquion.com/blog/covid-fobias

[3] Google. (2022). Descubre qué está buscando el mundo. https://trends.google.com/trends/?geo=ES

[4] Mayo Clinic. (2021, 20 agosto). Trastorno de ansiedad social (fobia social) - Síntomas y causas. https://www.mayoclinic.org/es-es/diseases-conditions/social-anxiety-disorder/symptoms-causes/syc-20353561

[5] Blasco, R. (2022, 26 octubre). Nuevas tecnologías para el tratamiento de la fobia social | Psicólogo especialista Barcelona. Psicólogo especialista Barcelona https://psicologo-especialista-barcelona.com/blog/nuevas-tecnologias-tratamiento-fobia-social/

[6] Rubio Arroyo, I. (2022, 11 febrero). Tecnología para tratar fobias y prevenir suicidios. Sacyr Blog. https://www.sacyr.com/-/tecnologia-para-tratar-fobias-y-prevenir-suicidios

[7] Spector, H. (2021, 25 octubre). Best Apps to Help With Social Anxiety (Mindfulness & Meditation). SimplePractice. https://www.simplepractice.com/blog/social-anxiety-apps/

[8] App Store. (2012, 2 febrero). Headspace: Mindful Meditation. https://apps.apple.com/us/app/headspace-com-meditation-mindfulness/id493145008

[9] App Store. (2015, 28 enero). Sanvello: Anxiety & Depression. https://apps.apple.com/us/app/pacifica-anxiety-stress-depression/id922968861

[10] What's Up? - Mental Health App - Apps on Google Play. (s. f.). https://play.google.com/store/apps/details?id=com.jacksontempra.apps.whatsup

[11] Valencia, F. (2021, 28 junio). ¿Por qué es importante el diseño web? Escape Digital. https://miescapedigital.com/por-que-es-importante-el-diseno-web/

[12] Plan International. (2021, 5 octubre). El 46% de las niñas sienten depresión, tristeza, estrés, preocupación o ansiedad a causa de . . . PLAN INTERNATIONAL. https://plan-international.es/noticias/depresion-trsiteza-informacion-falsa-internet

[13] Omedes, E. (2021, 9 octubre). Ansiedad, depresión o estrés: los efectos que sufren casi la mitad de las niñas y adolescentes a causa de las 'fake news'. www.20minutos.es - Últimas Noticias. https://www.20minutos.es/noticia/4845337/0/efectos-ninas-adolescentes-desinformacion-fake-news-informe/

[14] Universidad de Las Palmas de Gran Canaria. (s. f.). Objetivos y Competencias del GII. https://www2.ulpgc.es/archivos/plan_estudios/4008_40/ObjetivosyCompetenciasdelGII.pdf

[15] Enciende la Luz. (2018, 19 enero). Iterativo e incremental [Vídeo]. YouTube. https://www.youtube.com/watch?v=_qUlL01th2s

[16] Ramírez, F. (2022, 5 diciembre). La gran importancia de la toma de requisitos en el desarrollo de software - ITSoftware. ITSoftware | Apps | Software | Data Analytics. https://itsoftware.com.co/content/toma-de-requisitos-desarrollo-software/

[17] Jeffries, R. (2015). The Nature of Software Development: Keep It Simple, Make It Valuable, Build It Piece by Piece. Pragmatic Bookshelf.

[18] Highsmith, J. (2020). Agile Project Management: Creating Innovative Products (2nd ed.). Addison-Wesley.

[19] Sutherland, J. (2020). Scrum: The art of doing twice the work in half the time. Random House.

[20] Gestiona los proyectos de tu equipo desde cualquier lugar | Trello. (s. f.). https://trello.com/es

[21] MongoDB. (s. f.). MongoDB: The Developer Data Platform. https://www.mongodb.com/

[22] Chapaval, M. (2018, 21 febrero). Qué es Frontend y Backend: diferencias y características. Platzi. https://platzi.com/blog/que-es-frontend-y-backend/

[23] Node.Js. (s. f.). Node.js. https://nodejs.org/en/about/

[24] Chojrin, M. (2022). Qué es una API REST (API RESTful) https://platzi.com/clases/1638-api-rest/21611-que-significa-rest-y-que-es-una-api-restful/

[25] React – Una biblioteca de JavaScript para construir interfaces de usuario. (s. f.). React. https://es.reactjs.org/

[26] Calatrava, S. G. (2022, 12 septiembre). Las mejores librerías de React en 2022. Profile Software Services. https://profile.es/blog/librerias-react/

[27] Git - Getting Started - What is Git?. (s. f.). https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F

[28] GitHub: Let's build from here. (s. f.). GitHub. https://github.com/

[29] Brown, K. (2019, 13 noviembre). What Is GitHub, and What Is It Used For? How-To Geek. https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/

[30] Documentos de Google: editor de documentos online | Google Workspace. (s. f.). https://www.google.com/docs/about/?hl=es

[31] MeisterLabs. (s. f.). MindMeister: Mapas Mentales y Lluvia de Ideas en línea. MindMeister. https://www.mindmeister.com/es

[32] Figma: the collaborative interface design tool. (s. f.). Figma. https://www.figma.com/

[33] Draw.io | diagrams.net - free flowchart maker and diagrams online. (s. f.). https://app.diagrams.net/

[34] Pixel True | 2000+ High Quality Illustrations & Animations. (s. f.). https://www.pixeltrue.com/packs

[35] unDraw - Open source illustrations for any idea. (s. f.). https://undraw.co/

[36] Maluenda, R. (2022, 5 mayo). Tipos de desarrollo de aplicaciones web: ejemplos y características. Profile Software Services. https://profile.es/blog/desarrollo-aplicaciones-web/

[37] HTML: Lenguaje de etiquetas de hipertexto | MDN. (2022, 30 noviembre). https://developer.mozilla.org/es/docs/Web/HTML

[38] CSS: Cascading Style Sheets | MDN. (2022, 25 septiembre). https://developer.mozilla.org/en-US/docs/Web/CSS

[39] JavaScript - Aprende sobre desarrollo web | MDN. (2022, 30 noviembre). https://developer.mozilla.org/es/docs/Learn/JavaScript

[40] Introducción a Express/Node - Aprende sobre desarrollo web | MDN. (2022, 30 noviembre). https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction

[41] Express - Node.js web application framework. (s. f.). https://expressjs.com/

[42] MongoDB. (s. f.). MongoDB: La Plataforma De Datos Para Aplicaciones. https://www.mongodb.com/es

[43] MongoDB. (s. f.-b). ¿Qué es MongoDB? https://www.mongodb.com/es/what-is-mongodb

[44] MongoDB. (s. f.-a). MongoDB Atlas: La mejor forma de utilizar MongoDB en la nube 1. https://www.mongodb.com/presentations/mongodb-atlas-la-mejor-forma-de-utilizar-mongodb-en-la-nube-1

[45] Mongoose ODM v6.8.4. (s. f.). https://mongoosejs.com/

[46] body-parser analiza el cuerpo de la solicitud POST - programador clic. (s. f.). https://programmerclick.com/article/2883931076/

[47] npm: cookie-session. (2021, 16 diciembre). npm. https://www.npmjs.com/package/cookie-session

[48] npm: cors. (2018, 4 noviembre). npm. https://www.npmjs.com/package/cors

[49] npm: dotenv. (2022, 29 septiembre). npm. https://www.npmjs.com/package/dotenv

[50] npm: jsonwebtoken. (2022, 21 diciembre). npm. https://www.npmjs.com/package/jsonwebtoken

[51] Jones, M. (2023, 18 mayo). RFC 7519: JSON Web Token (JWT). https://www.rfc-editor.org/rfc/rfc7519

[52] npm: bcrypt. (2022, 6 octubre). npm. https://www.npmjs.com/package/bcrypt

[53] Reinman, A. (s. f.). Nodemailer :: Nodemailer. https://nodemailer.com/about/

[54] React – A JavaScript library for building user interfaces. (s. f.). React. https://reactjs.org/

[55] React Router - Docs Home v6.3.0. (s. f.). React Router. https://reactrouter.com/en/v6.3.0

[56] React Hook Form - Home. (s. f.). React Hook Form - Simple React forms validation. https://react-hook-form.com/

[57] Redux Toolkit | Redux Toolkit. (s. f.). https://redux-toolkit.js.org/

[58] MUI. (s. f.). MUI: The React component library you always wanted. https://mui.com/

[59] React-toastify | React-Toastify. (s. f.). https://fkhadra.github.io/react-toastify/introduction/

[60] npm: react-calendar. (2022, 8 noviembre). npm. https://www.npmjs.com/package/react-calendar

[61] Chart.js. (s. f.). Open source HTML5 Charts for your website. https://www.chartjs.org/

[62] React-chartjs-2 | React-chartjs-2. (s. f.). https://react-chartjs-2.js.org/

[63] Visual Studio Code - Code Editing. Redefined. (2021, 3 noviembre). https://code.visualstudio.com/

[64] Postman. (s. f.). https://www.postman.com/

[65] Redux DevTools. (s. f.). Chrome Web Store. https://chrome.google.com/webstore/detail/redux-devtools/lmhkpmbekcpmknklioeibfkpmmfibljd?hl=en

[66] Render. (s.f). https://render.com/

[67] Vercel. (s.f). https://vercel.com/

[68] Lighthouse. (s. f.). Chrome Web Store. https://chrome.google.com/webstore/detail/lighthouse/blipmdconlkpinefehnmjammfjpmpbjk?hl=es

[69] Azzouz, M. (2021, 23 de agosto). Diseño de sitios web : ¿CMS o Framework? | Izeelogo. Izeelogo, el generador de logos online. https://www.izeelogo.com/es/blog/crear-un-sitio-web-cms-o-framework

[70] Docenteunivia(2014, 27 agosto). LOS CASOS DE USO SUS VENTAJAS Y DESVENTAJAS. ADMINISTRACIÓN DE REQUERIMIENTOS. https://administracionderequerimientos.wordpress.com/2014/08/27/los-casos-de-uso-sus-ventajas-y-desventajas/

[71] Visure Solutions. (2023, 1 enero). Definición de requisitos: ¿Qué es y cómo aplicarlo? | Guía completa. https://visuresolutions.com/es/blog/requirements-definition/

[72] GeeksforGeeks. (2022, 2 diciembre). Functional vs Non Functional Requirements. https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/

[73] GeeksforGeeks. (2022, diciembre 2). Non-functional Requirements in Software Engineering. https://www.geeksforgeeks.org/non-functional-requirements-in-software-engineering/

[74] Coppola, M. (2023, 20 enero). ¿Qué es el diseño web? Definición, características e importancia. https://blog.hubspot.es/website/diseno-web

[75] Vilardi, R. (2022, 5 julio). La importancia de las experiencias de usuario: UX Basics y el universo de la usabilidad. https://www.wearemarketing.com/es/blog/importancia-de-la-experiencia-de-usuario-ux.html

[76] Color Psychology in Marketing: A Full Guide. (2022, 15 junio). Nick Kolenda. https://www.nickkolenda.com/color-psychology/

[77] Fonts and Typefaces (s. f.). NAMI, the National Alliance on Mental Illness, https://www.nami.org/NAMInet/NAMI-Marketing/NAMI-Identity-Guide/Fonts-and-Typefaces

[78] BACKEND (con Node, Express y MongoDB) README (2023, 3 febrero). https://github.com/jamj2000/tiendabackend

ANEXO I. Diseño web.

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Joaquín Javier Pastore Barrios

TUTORIZADO POR:

Francisco Alexis Quesada Arencibia

Índice general

I.Primer diseño

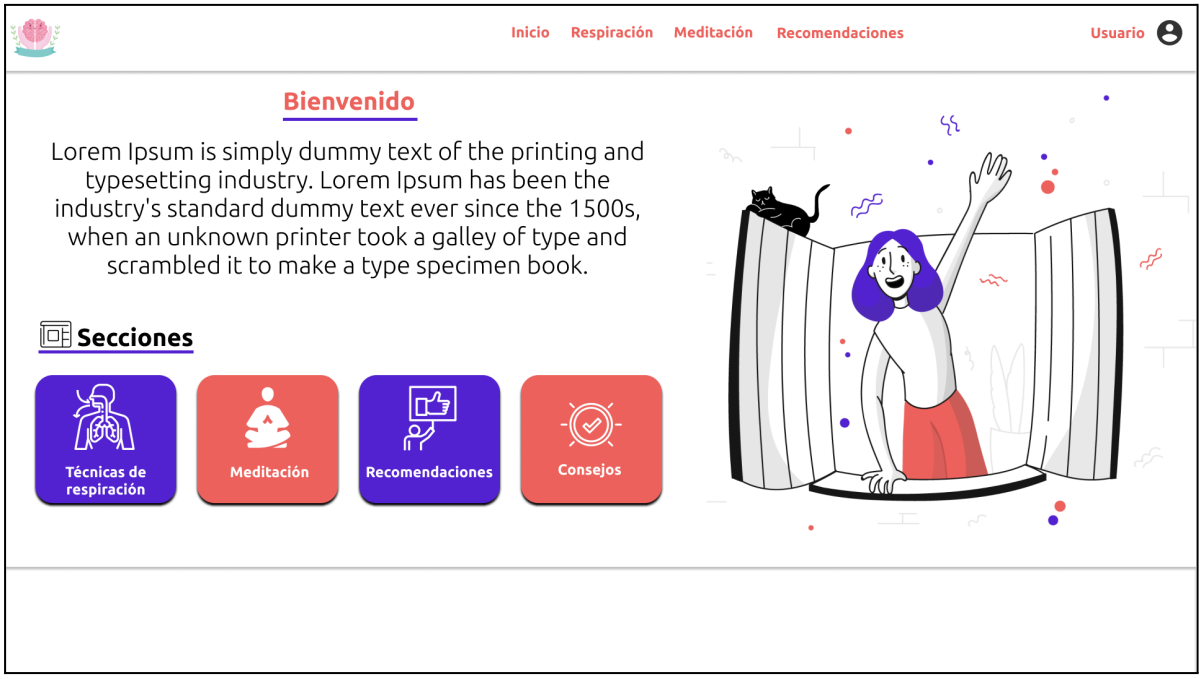


Ilustración 1 : *Mockup* de la sección *Home*

Fuente: Propia

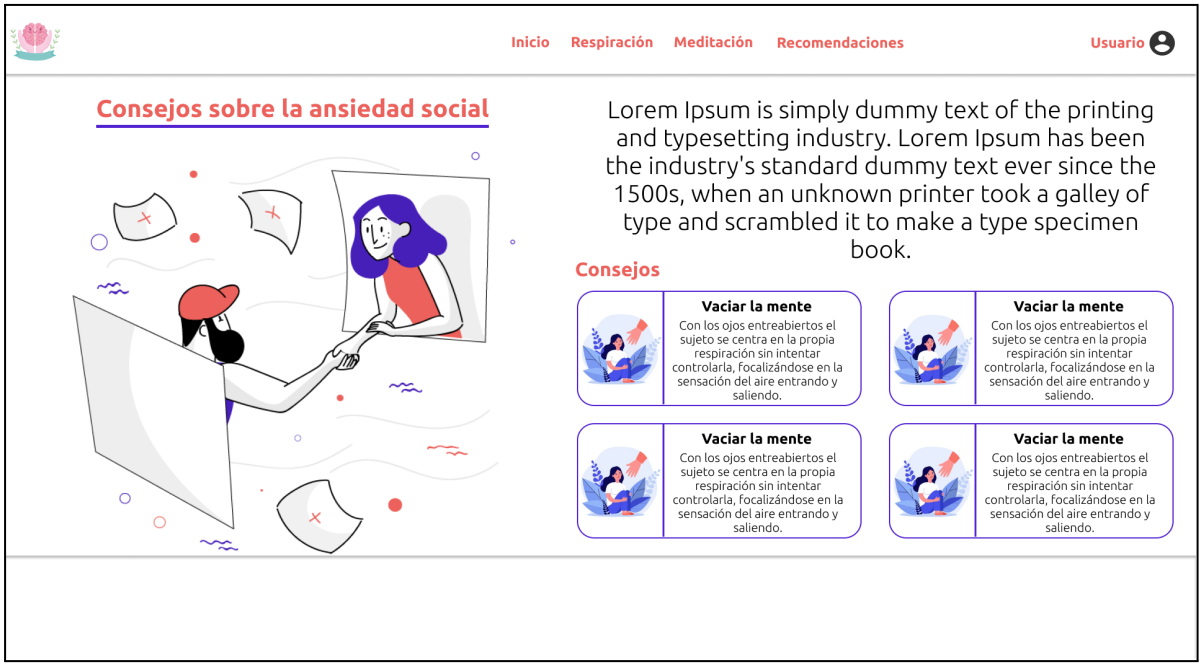


Ilustración 2: *Mockup* de la sección *Consejos*

Fuente: Propia

Ilustración 3: *Mockup* de la sección *Técnicas de respiración*

Fuente: Propia

Ilustración 4: *Mockup* de la sección *Meditación*

Fuente: Propia

Ilustración 5: *Mockup* de la sección *Mi perfil*

Fuente: Propia

Ilustración 6: *Mockup* de la sección *Recomendación de lectura*

Fuente: Propia

Ilustración 7: *Mockup* de la sección *Recomendación de películas*

Fuente: Propia

Ilustración 8: *Mockup* de la sección *Recomendación de videojuegos*

Fuente: Propia

Ilustración 9: *Mockup* de la sección *Recomendaciones*

Fuente: Propia

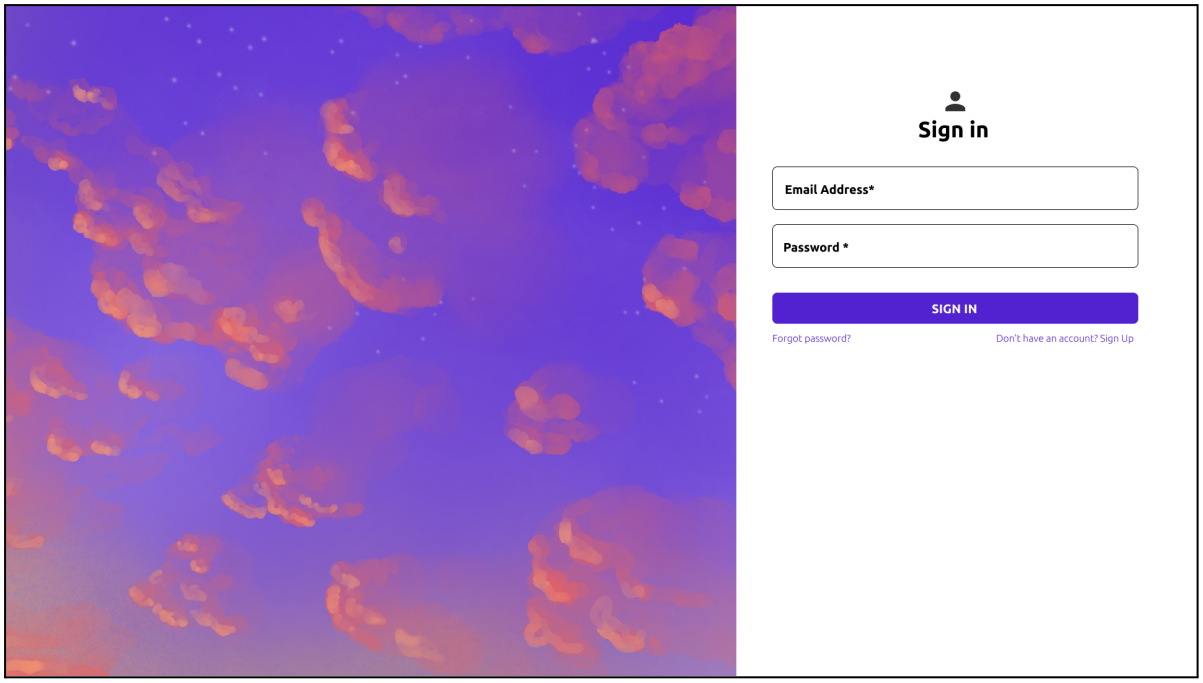


Ilustración 10: *Mockup* de la sección *iniciar sesión*

Fuente: Propia

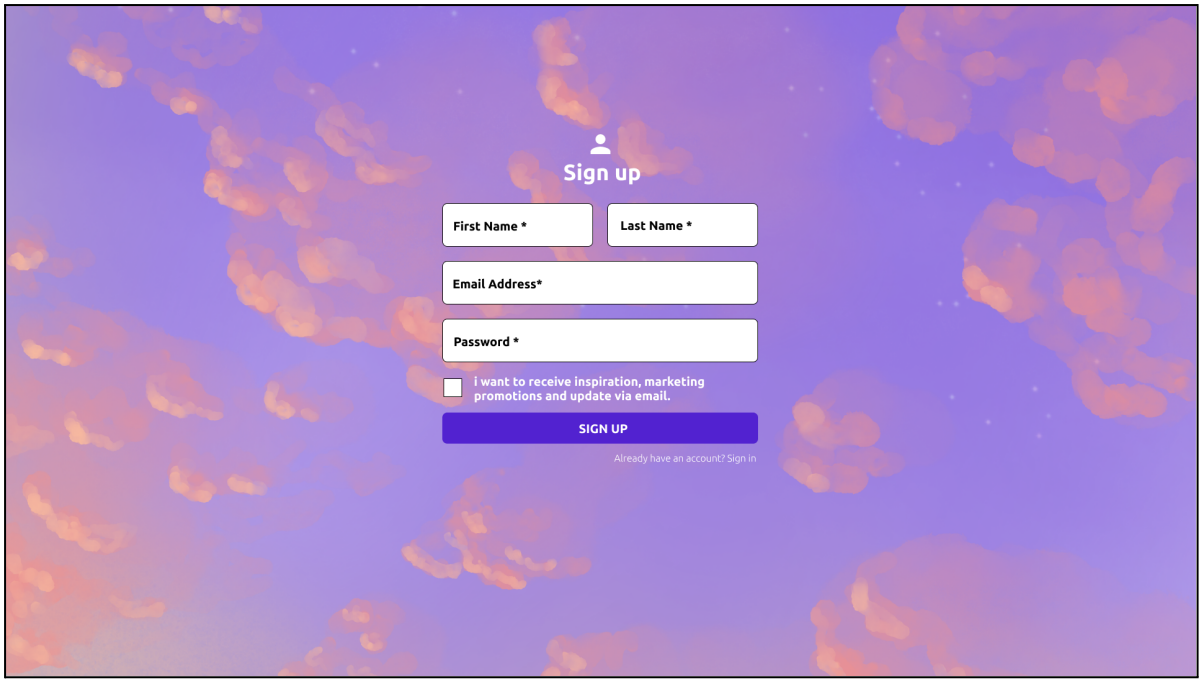


Ilustración 12: *Mockup* de la sección *registrarse*

Fuente: Propia

II. Nuevo diseño

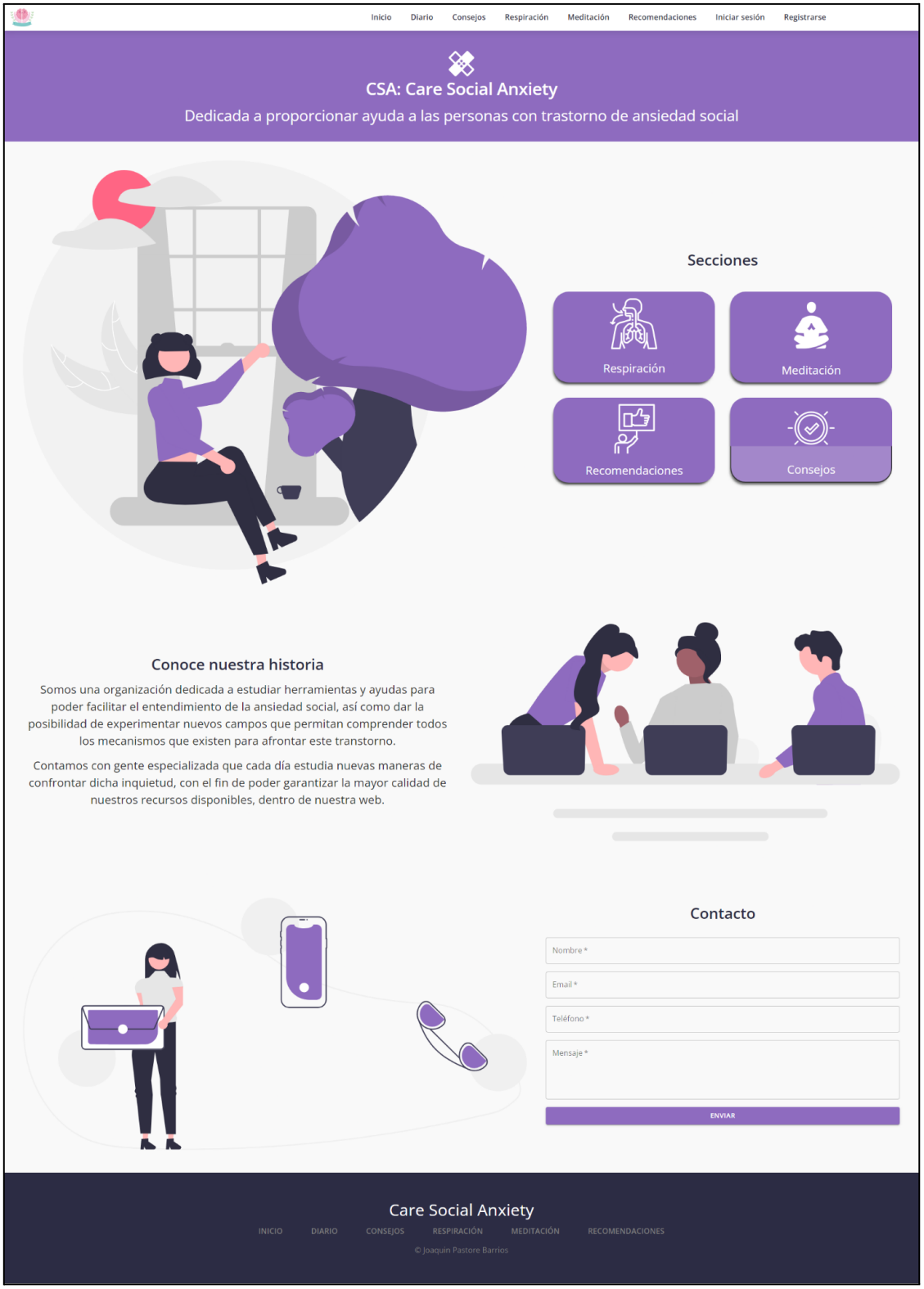


Ilustración 13: *Mockup* de la sección *home* tras el cambio de diseño

Fuente: Propia

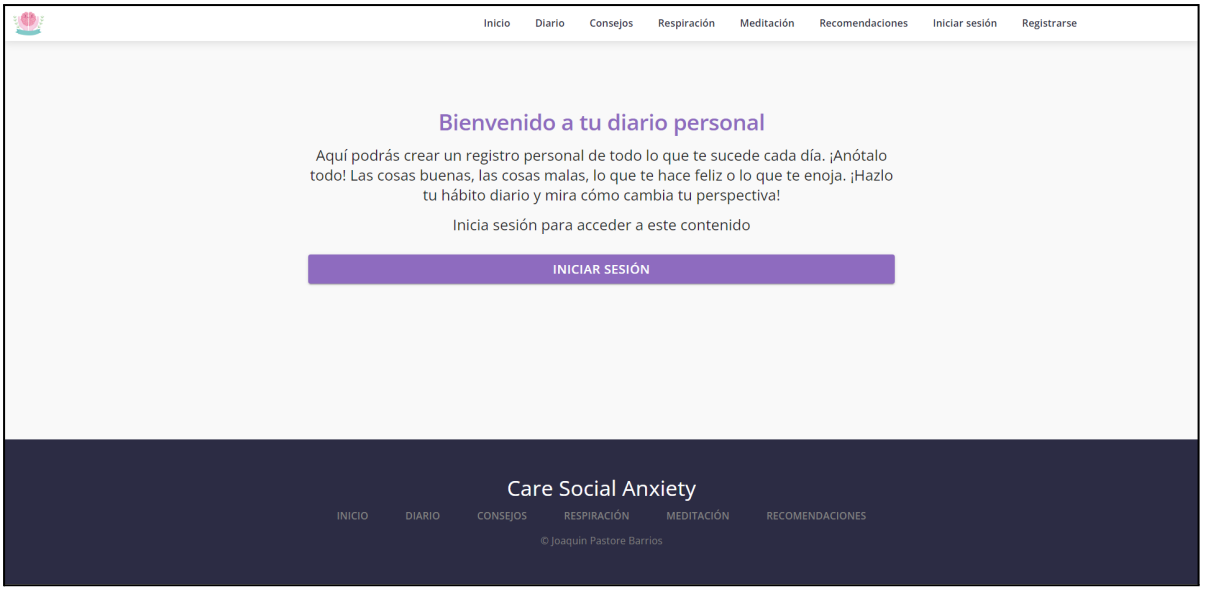


Ilustración 14: *Mockup* de la sección diario personal con un usuario con sesión no iniciada

Fuente: Propia

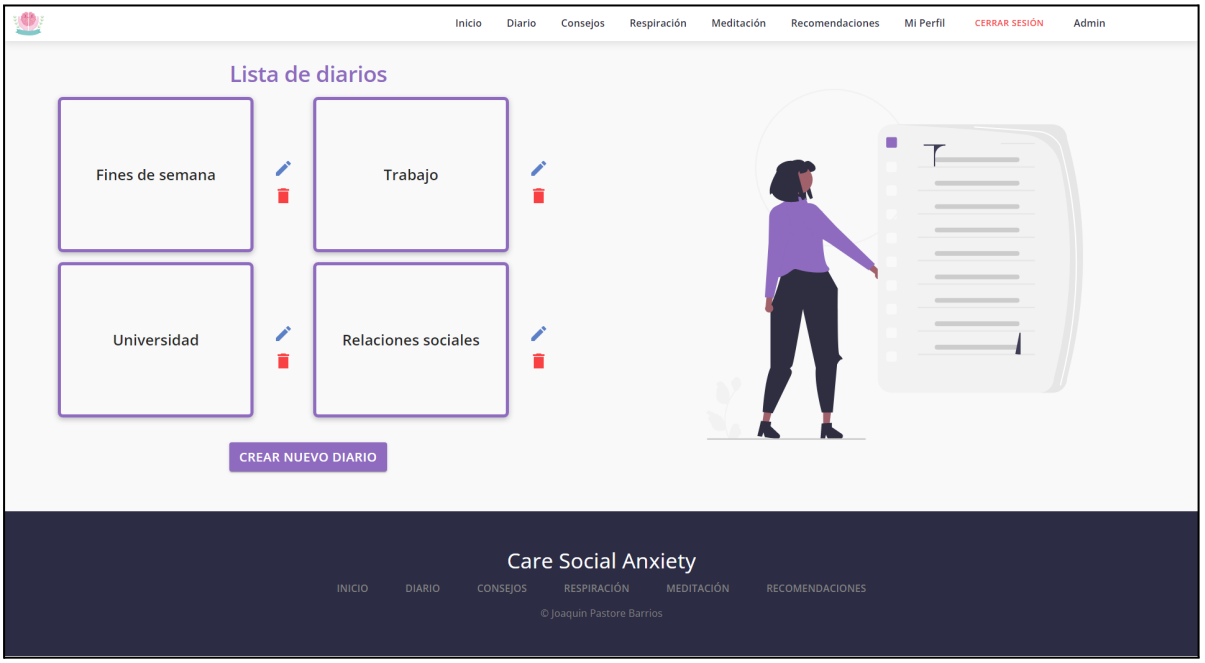


Ilustración 15: *Mockup* de la sección diario personal con un usuario con sesión iniciada

Fuente: Propia

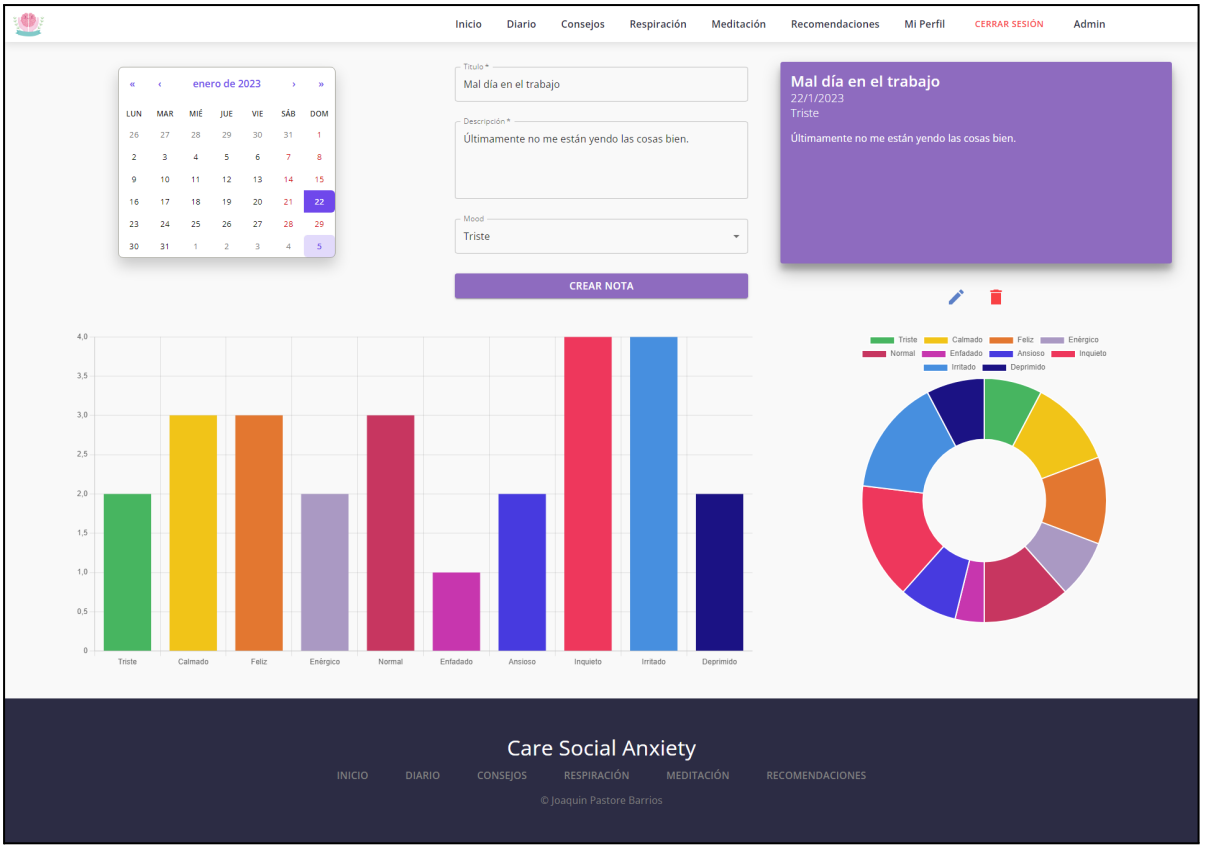


Ilustración 16: *Mockup* de la sección diario personal con un usuario con sesión iniciada

Fuente: Propia

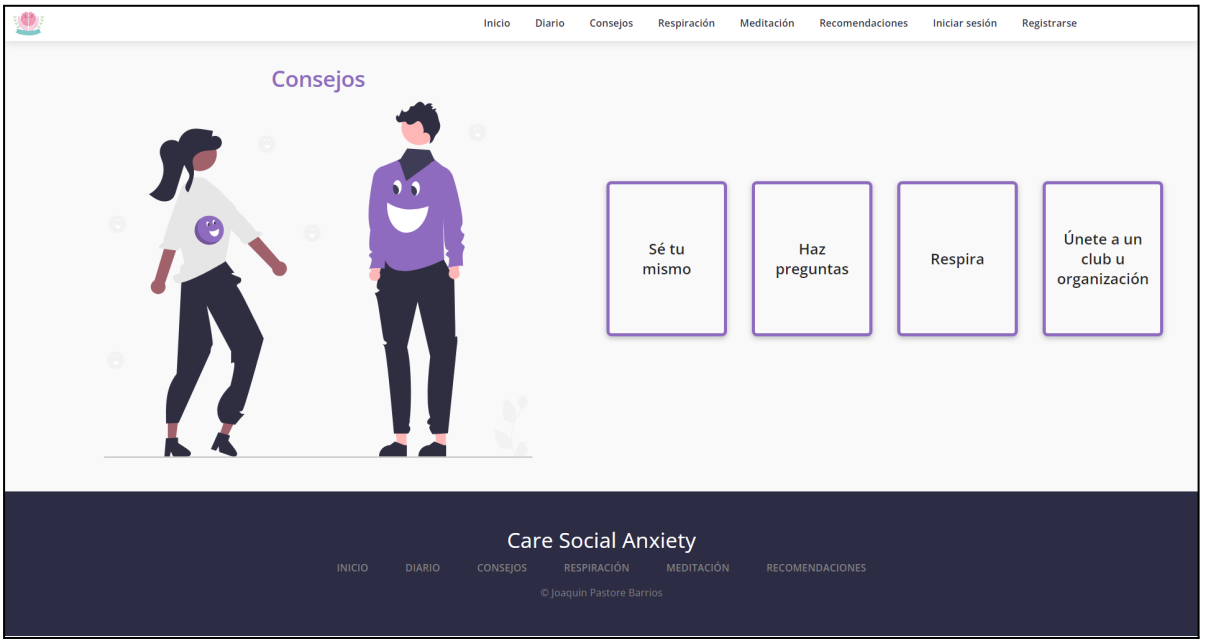


Ilustración 17: *Mockup* de la sección *Consejos* tras el cambio de diseño

Fuente: Propia

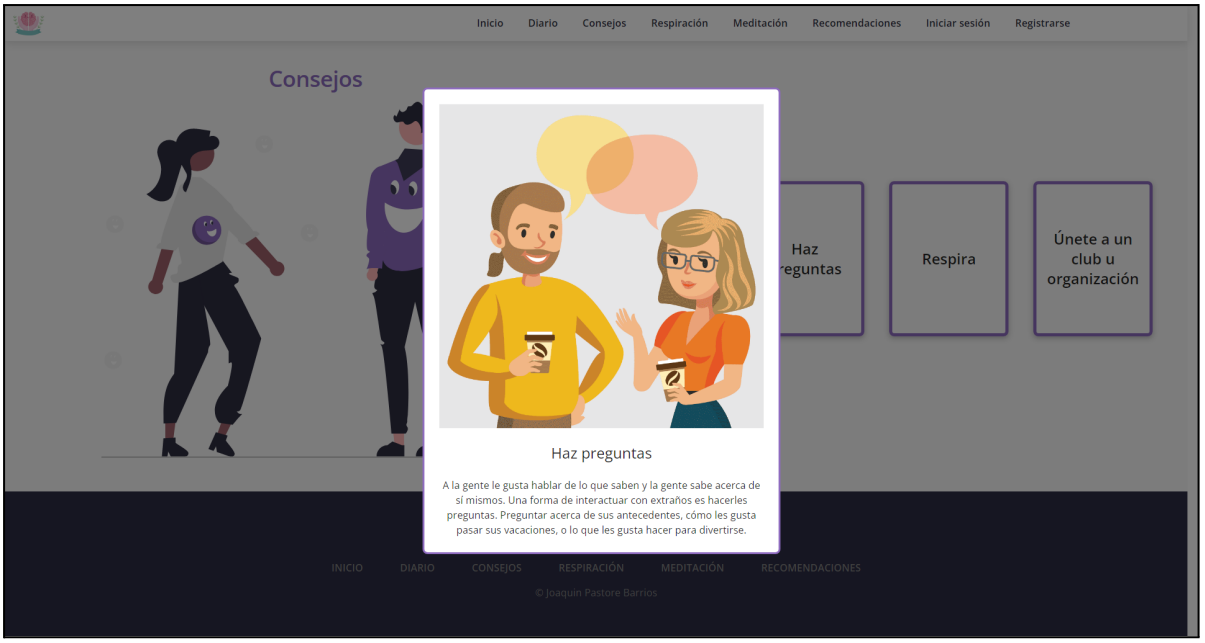


Ilustración 18: Previsualización de un recurso con su imagen y descripción

Fuente: Propia

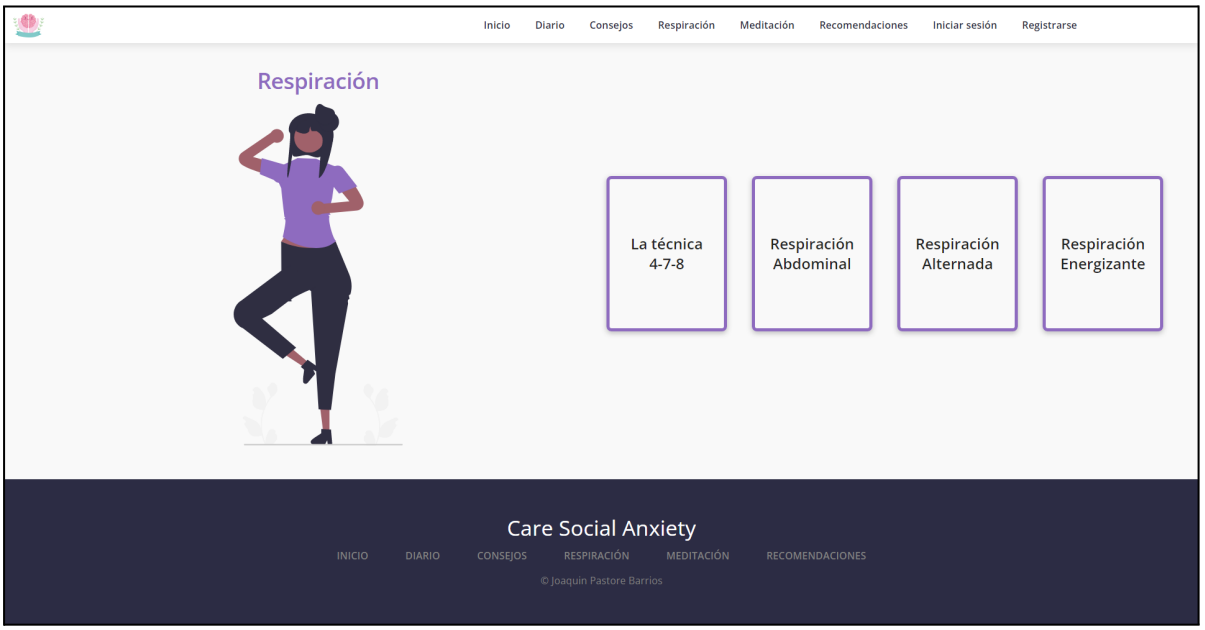


Ilustración 19: *Mockup* de la sección *Respiración* tras el cambio de diseño

Fuente: Propia

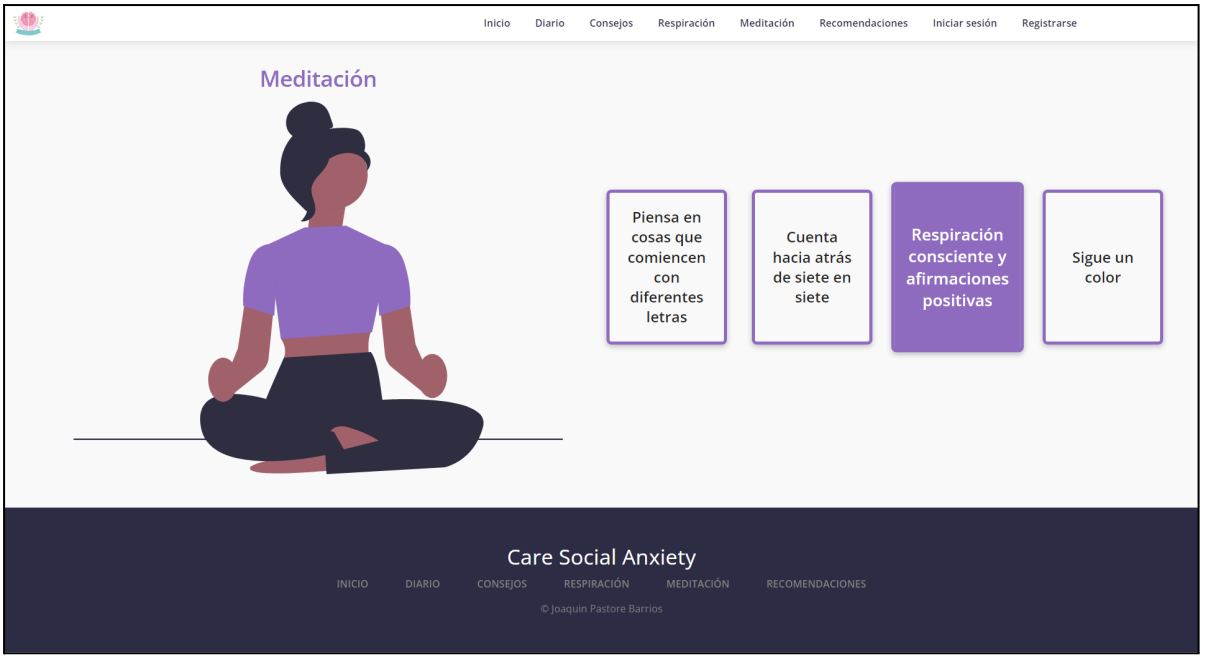


Ilustración 20: *Mockup* de la sección *Meditación* tras el cambio de diseño

Fuente: Propia

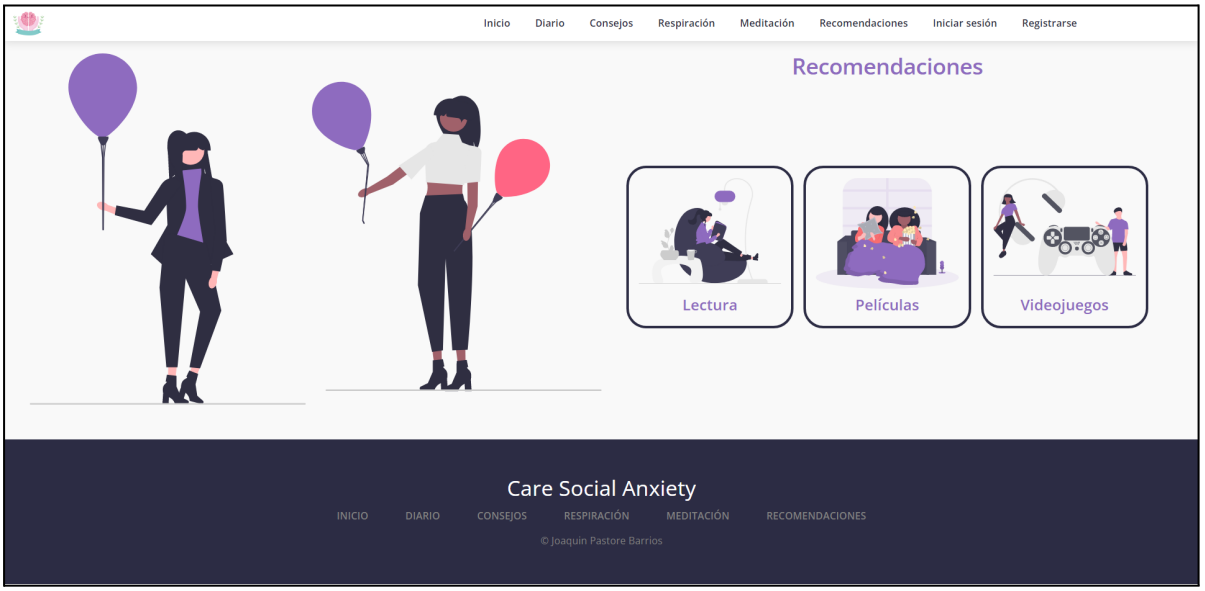


Ilustración 21: *Mockup* de la sección *Recomendaciones* tras el cambio de diseño

Fuente: Propia

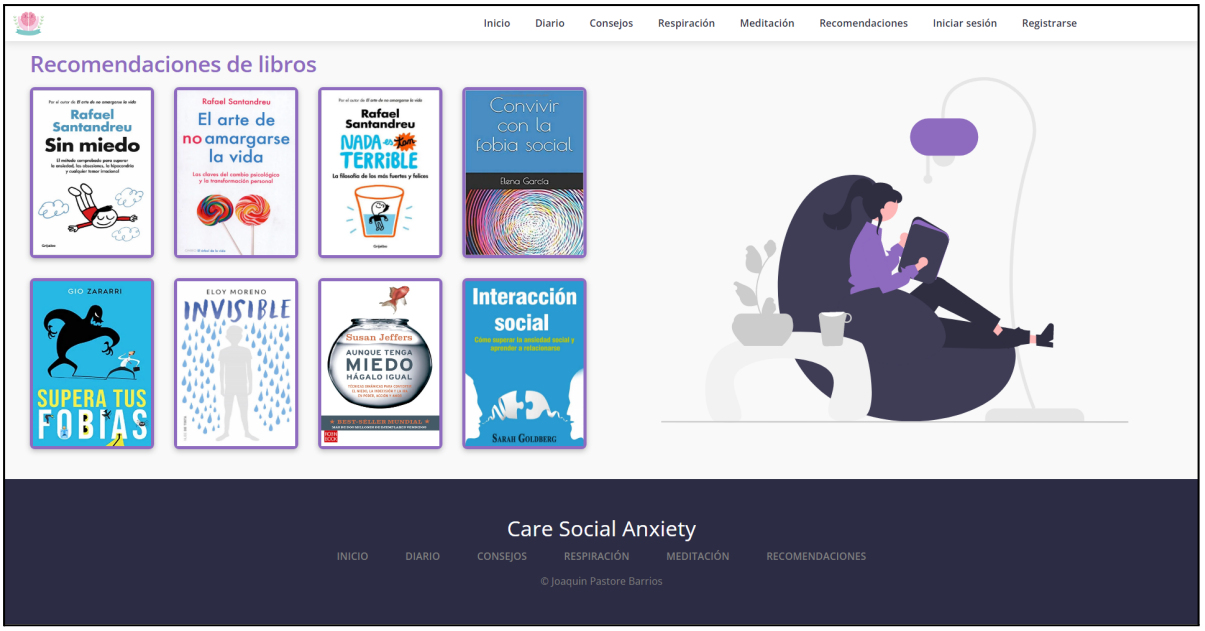


Ilustración 22: *Mockup* de la sección *Recomendaciones de libros* tras el cambio de diseño

Fuente: Propia

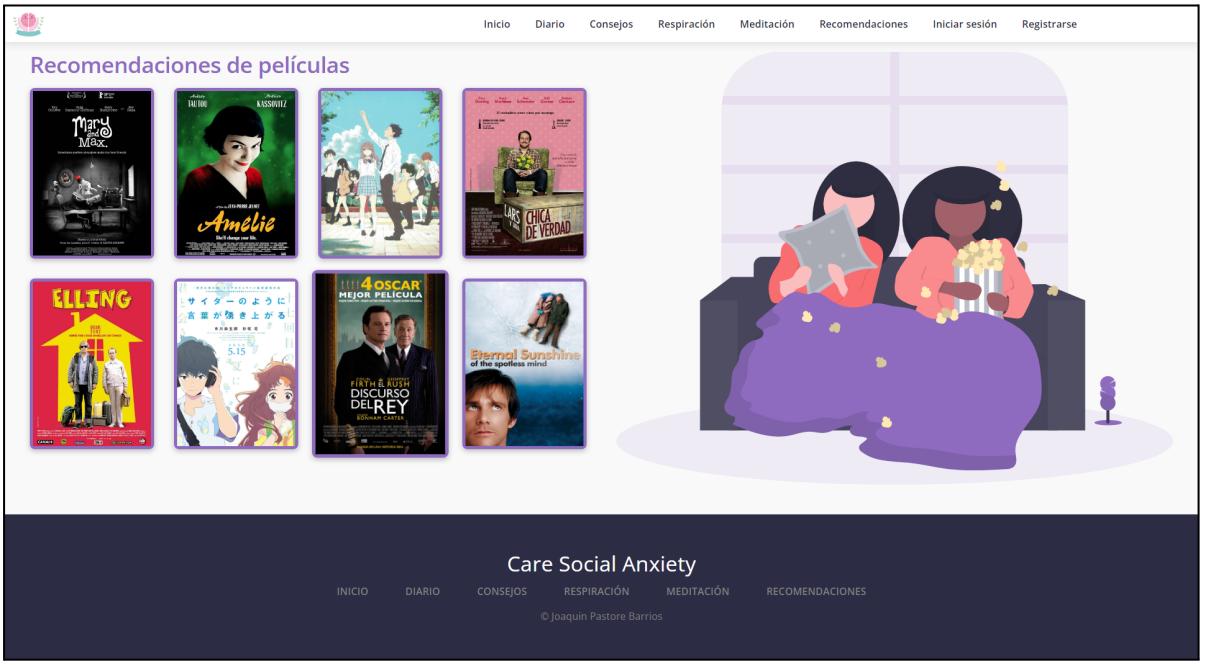


Ilustración 23: *Mockup* de la sección *Recomendaciones de películas* tras el cambio de diseño

Fuente: Propia

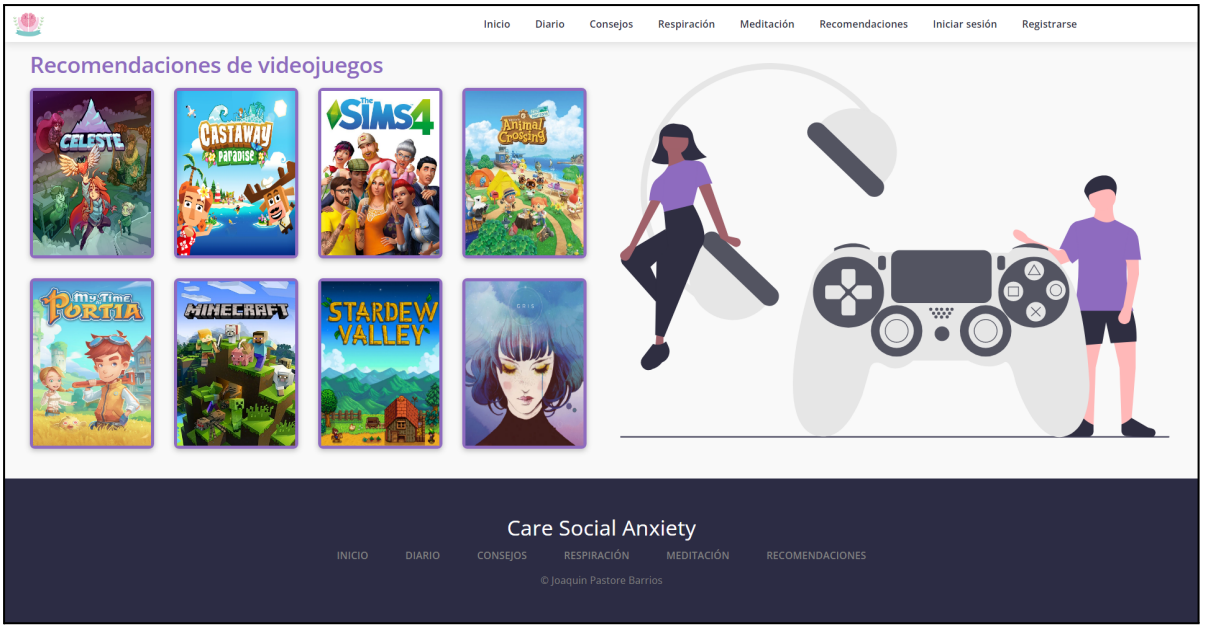


Ilustración 24: *Mockup* de la sección *Recomendaciones de videojuegos* tras el cambio de diseño

Fuente: Propia

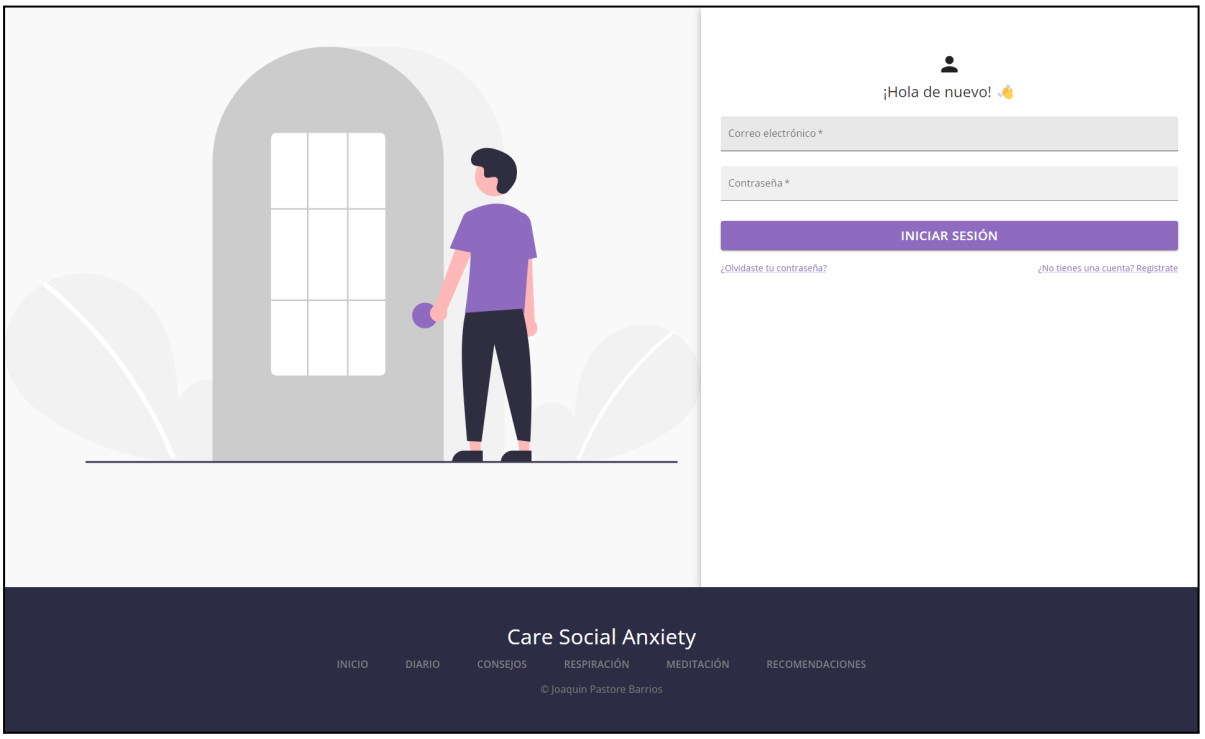


Ilustración 25.: *Mockup* del inicio de sesión

Fuente: Propia

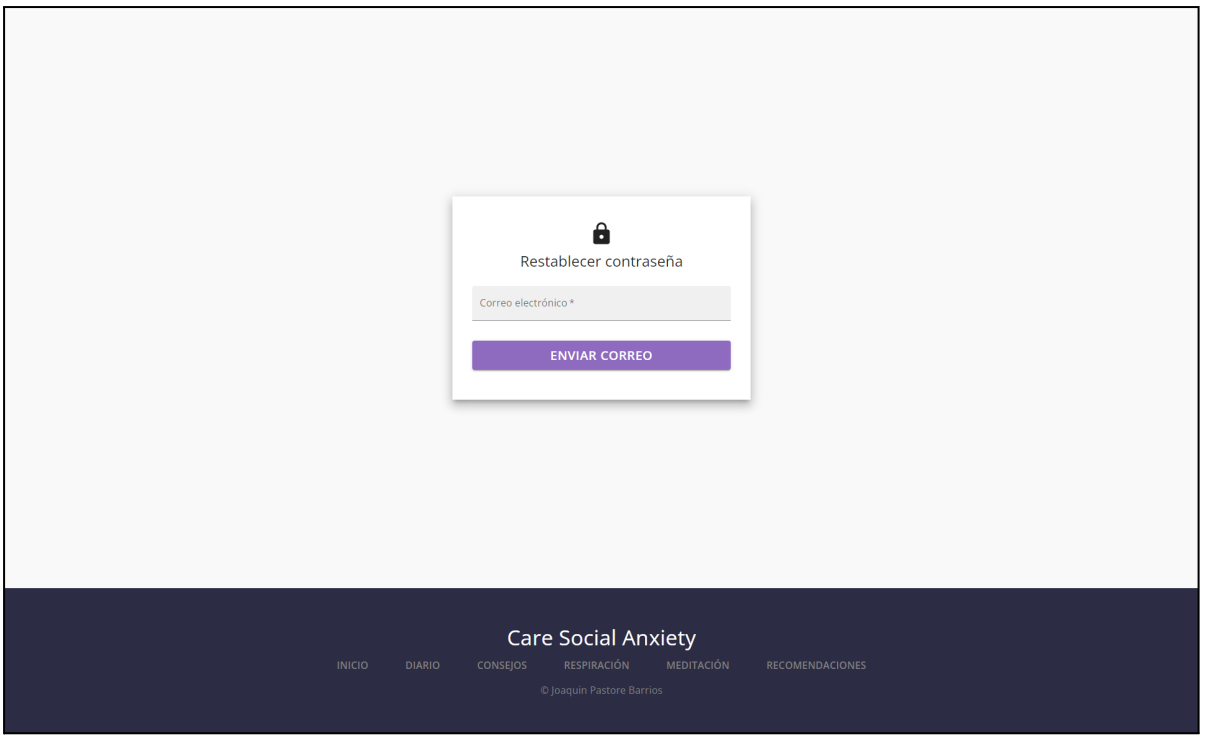


Ilustración 26: *Mockup* de establecer nueva contraseña rellenando el correo electrónico

Fuente: Propia

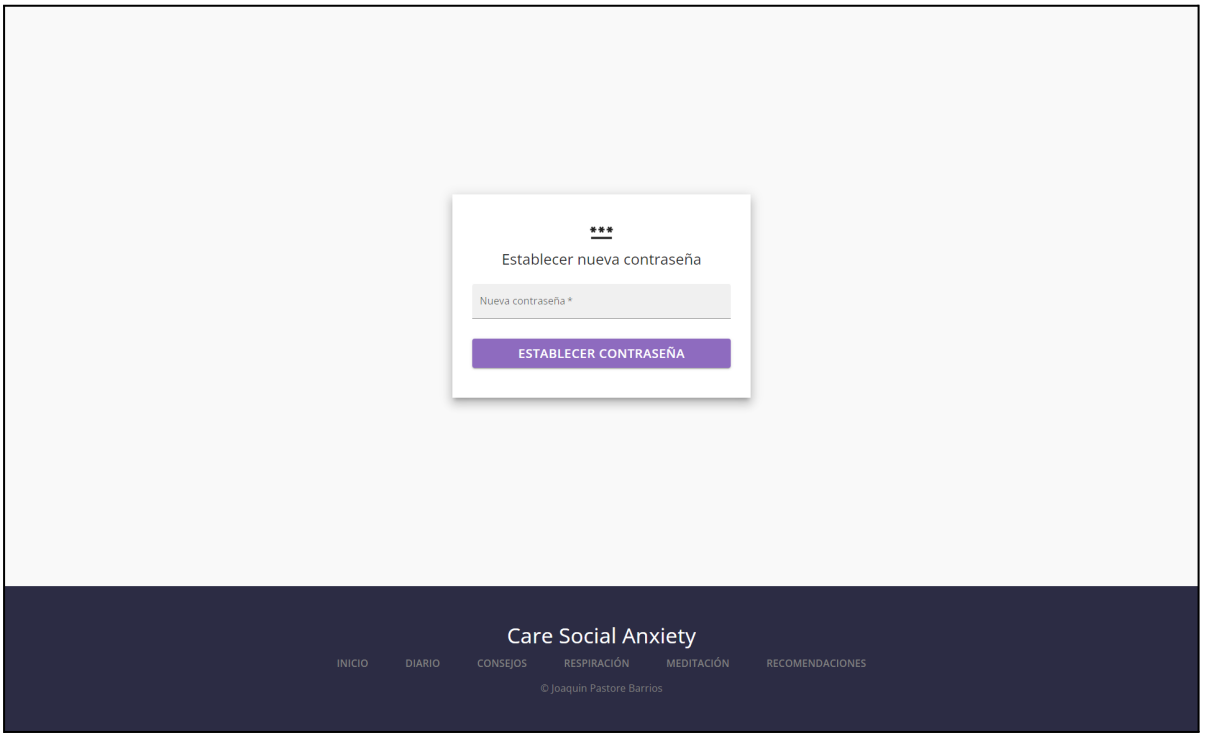


Ilustración 27: *Mockup* de establecer nueva contraseña

Fuente: Propia

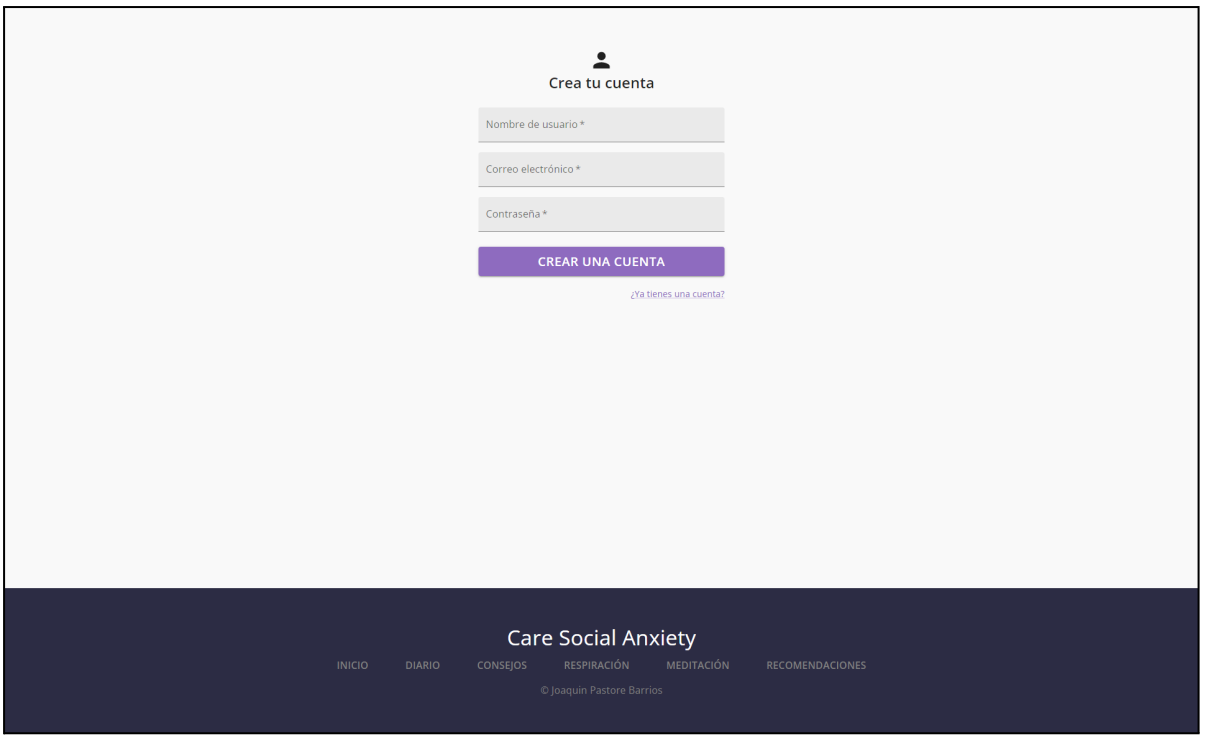


Ilustración 28: *Mockup* del formulario de creación de cuenta

Fuente: Propia

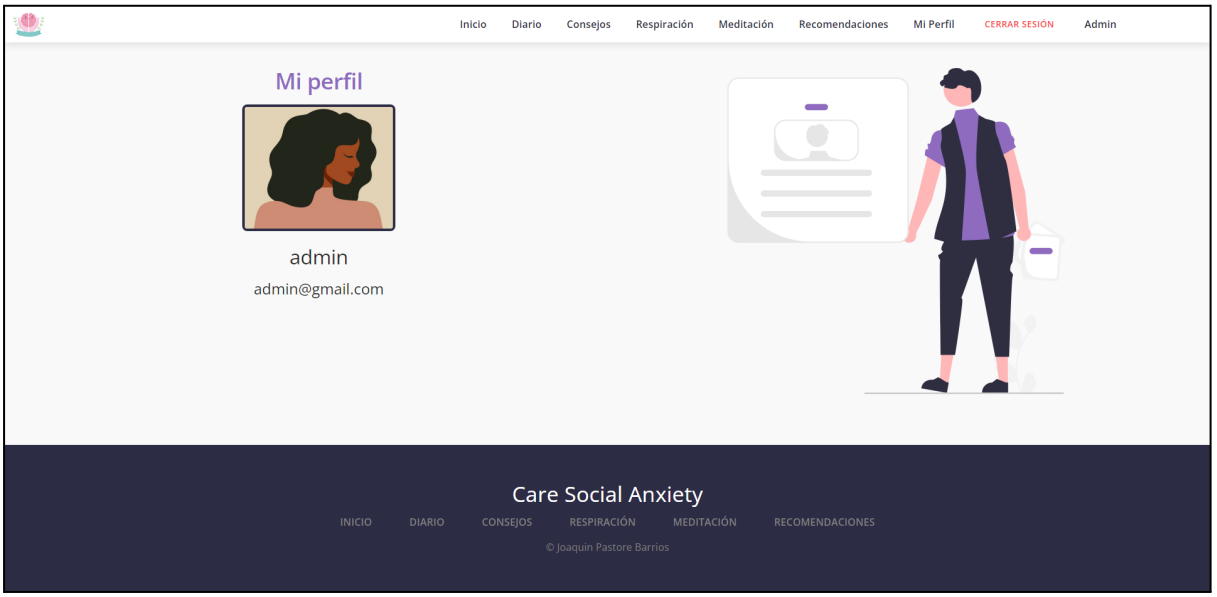


Ilustración 29: *Mockup* de la sección de *Mi Perfil*

Fuente: Propia

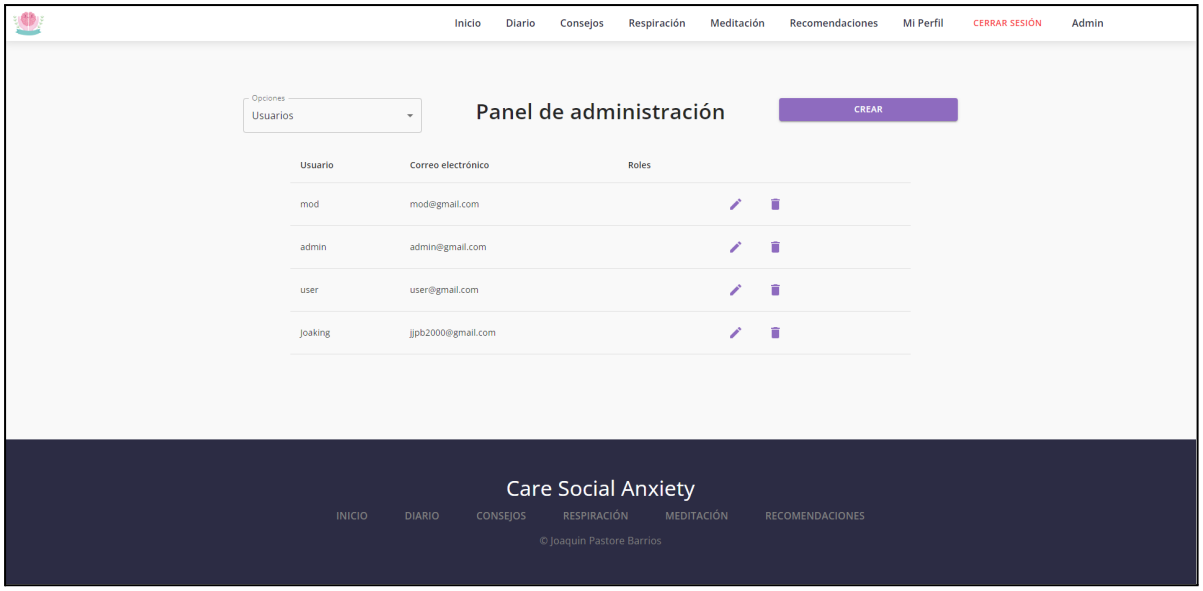


Ilustración 30: *Mockup* de la sección de *Panel de administración*

Fuente: Propia

ANEXO II. Manual de usuario.

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Joaquín Javier Pastore Barrios

TUTORIZADO POR:

Francisco Alexis Quesada Arencibia

Índice general

I. Introducción

El presente manual de usuario tiene como objetivo explicar las funcionalidades y el funcionamiento de esta aplicación web, que se ha diseñado para brindar al usuario herramientas para mejorar su bienestar mental y emocional orientado al trastorno de ansiedad social.

La aplicación ofrece recursos como ejercicios de meditación y relajación, técnicas de respiración, diarios de gratitud y reflexión, y herramientas para controlar el estrés y la ansiedad. Además, cuenta con una sección de seguimiento del progreso, que permite a los usuarios hacer un monitoreo de sus objetivos y ver cómo están mejorando con el tiempo.

Para utilizar la aplicación, el usuario podrá acceder a todas las herramientas disponibles y utilizarlas de manera fácil e intuitiva. Sin embargo, para la funcionalidad del diario personal, si será necesario que el usuario esté registrado.

II. Página principal

La página principal de la aplicación ha sido diseñada para brindar una visión general de la misma y para facilitar la navegación del usuario por las distintas secciones de la plataforma de manera intuitiva.

La introducción en la página principal proporciona una breve descripción de la aplicación y su propósito, lo que ayuda a los usuarios a entender rápidamente de qué se trata la plataforma. Además, el panel con las distintas secciones permite a los usuarios acceder de forma sencilla y rápida a las diferentes funcionalidades de la aplicación.

La sección de historia y misión ofrece a los usuarios una idea clara de los valores y principios que impulsan la aplicación, así como de su compromiso con la mejora del bienestar mental y emocional de las personas.

Por último, la sección de contacto es una manera fácil y conveniente para que los usuarios puedan comunicarse con el equipo detrás de la aplicación en caso de tener alguna pregunta o consulta. De esta forma, se garantiza una atención personalizada y eficiente para los usuarios que requieran ayuda o asistencia en cualquier momento.

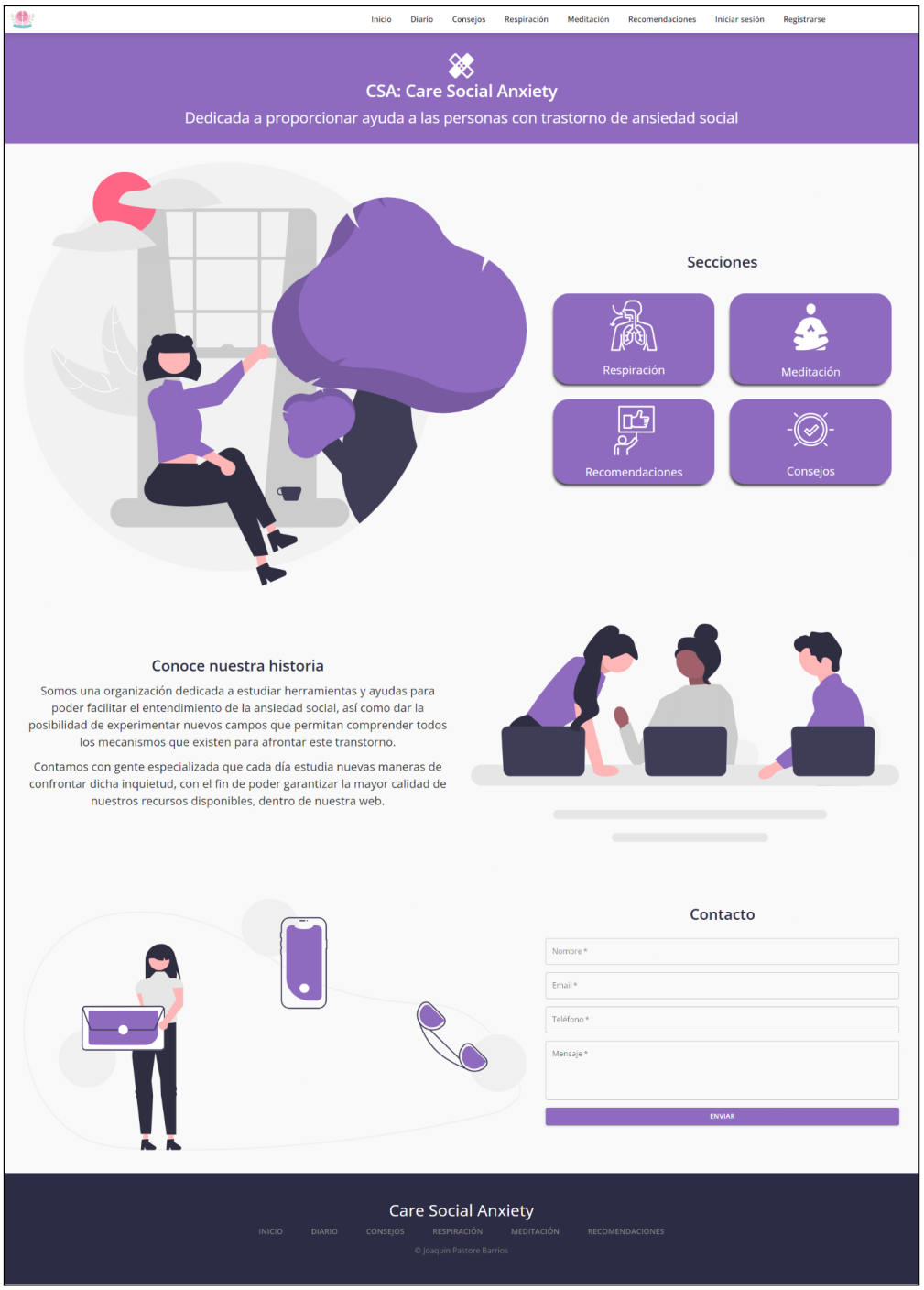


Ilustración 1: Página principal

Fuente: Propia

III. Diario personal

La sección del diario personal de la aplicación web solo está disponible para los usuarios registrados en la plataforma. En esta sección, podrás crear diarios y notas, y visualizar tus datos representados mediante gráficas.

Para agregar un nuevo diario, simplemente se debe presionar en el botón "CREAR NUEVO DIARIO" y aparecerá una ventana emergente para escribir el título del diario. Una vez que se haya escogido un título, aparecerá un mensaje indicando que la acción se ha completado y el diario se mostrará en pantalla. También aparecerán dos botones, uno de color azul para editar el título del diario y otro de color rojo para eliminarlo. Es importante destacar que solo los usuarios registrados pueden acceder a esta sección y crear sus propios diarios.

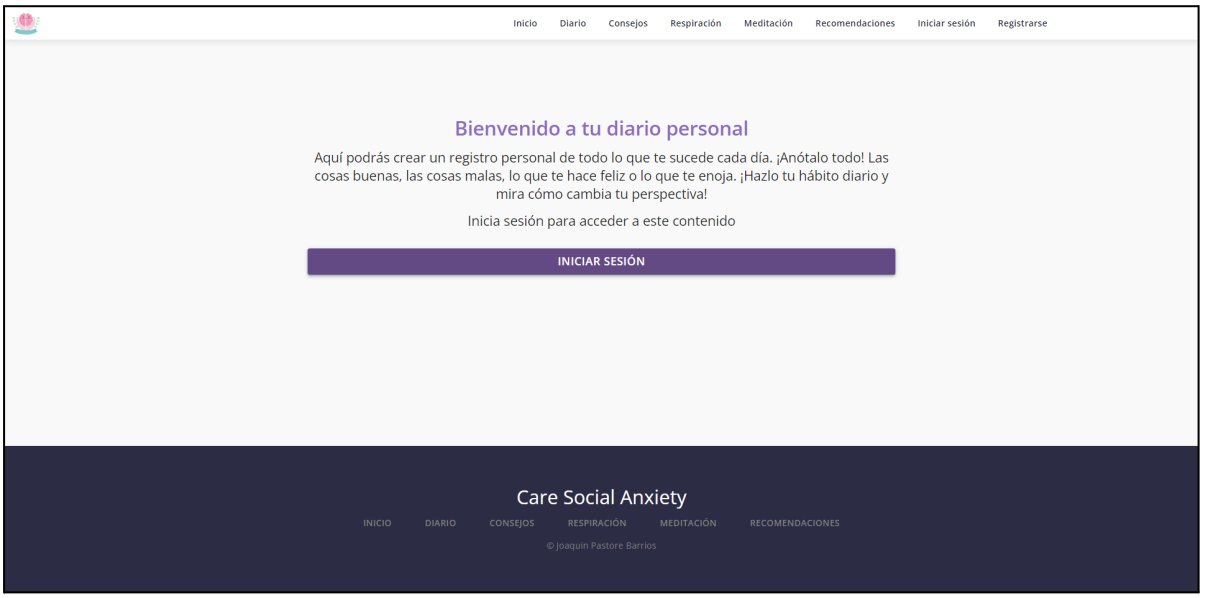


Ilustración 2: Sección diario personal sin iniciar sesión

Fuente: Propia

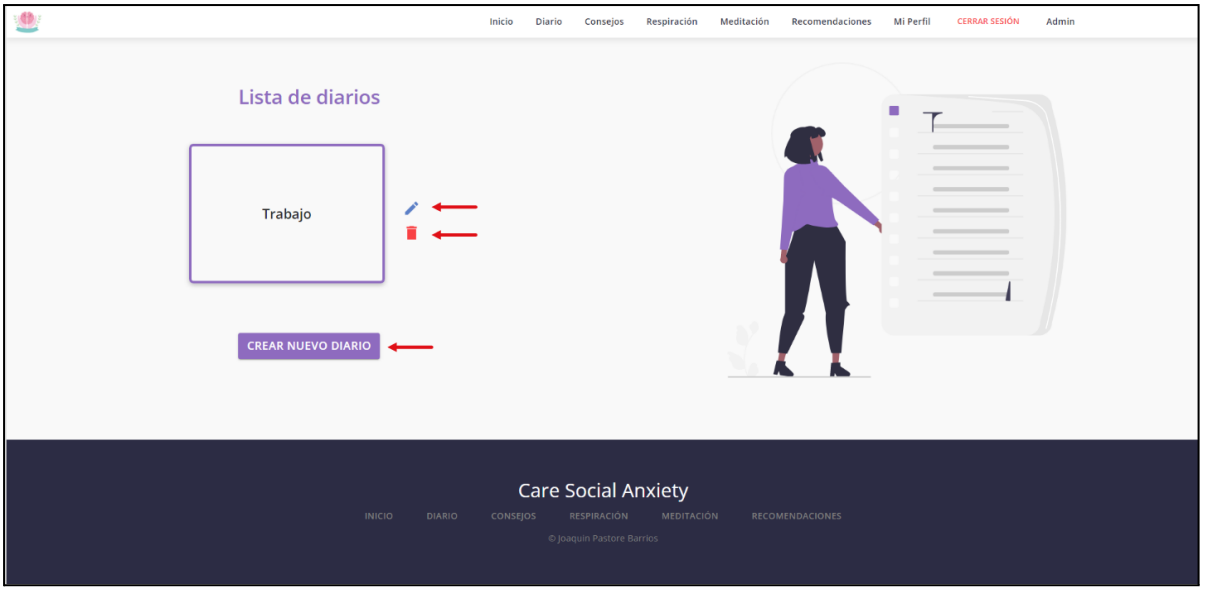


Ilustración 3: Sección diario personal usuario registrado

Fuente: Propia

Una vez que se haya creado un diario en la sección correspondiente, simplemente se debe pulsar en él para acceder a su sección de creación de notas. En este apartado, se encuentra un calendario para seleccionar la fecha en la que se desea crear la nota, así como los campos correspondientes para escribir la nota y seleccionar la emoción asociada a ella.

Una vez completado los campos, la nota aparecerá en la parte superior derecha de la pantalla, y a su vez dos gráficas en la parte inferior, exponiendo los días asociados a las emociones y el porcentaje total para cada una de ellas. También aparecerán dos botones debajo de la nota, uno de color azul para editar la nota y otro de color rojo para eliminarla.

Es importante destacar que la sección de notas solo está disponible dentro de un diario específico y que cada nota creada se asocia automáticamente con la fecha correspondiente. De esta manera, podrás visualizar las notas y emociones en diferentes días y fechas, lo que te permitirá un mejor seguimiento y control sobre tu bienestar emocional.

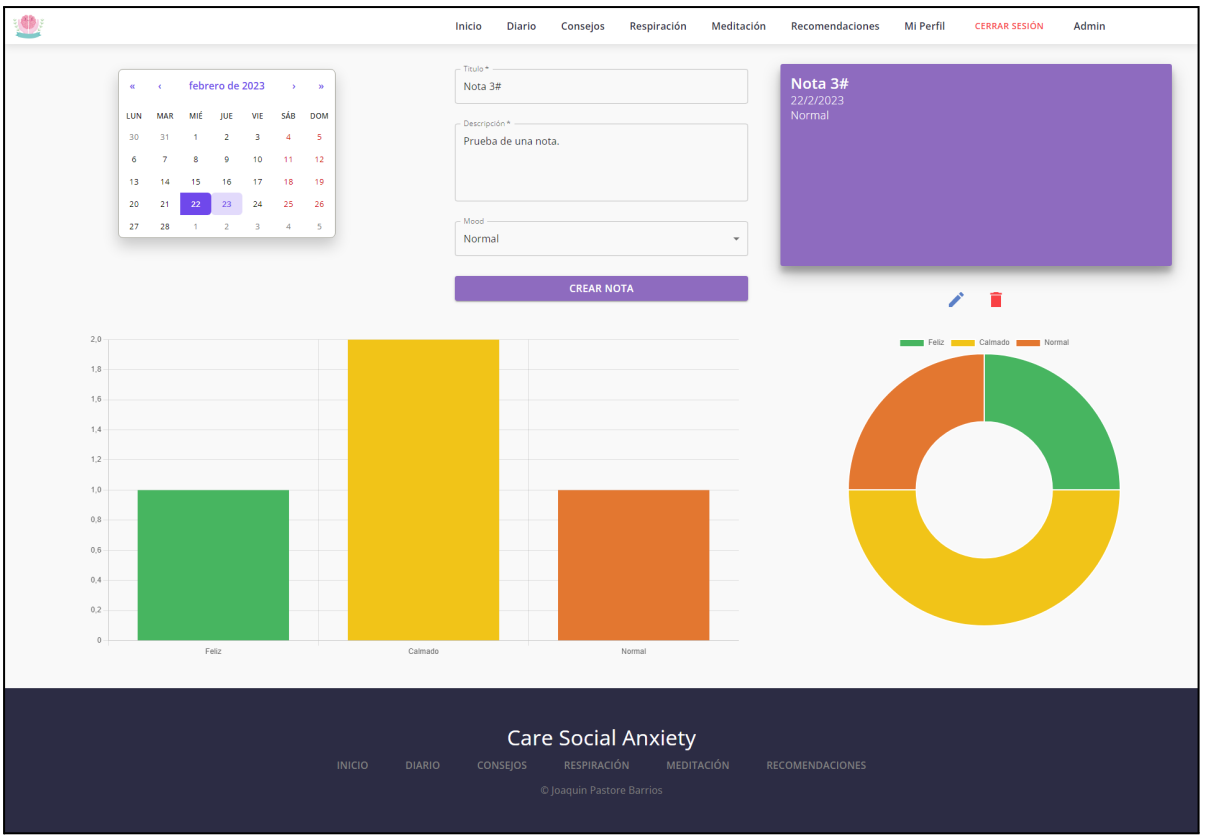


Ilustración 4: Sección diario personal - creación de notas

Fuente: Propia

IV. Secciones de información

La sección de consejos, de respiración, de meditación y la de recomendaciones están diseñadas para brindar a los usuarios diferentes recursos que pueden ayudarles a mejorar su bienestar mental y emocional. En cada sección, se exhiben una serie de tarjetas que contienen información sobre dicho recurso en cada ocasión.

Cuando se selecciona una tarjeta en concreto, se abrirá un modal con información adicional sobre el tema en cuestión. Por ejemplo, en la sección de consejos se mostrará más información sobre cómo aplicar el consejo en la vida cotidiana, mientras que en la sección de meditación se explicará cómo realizar la meditación en cuestión. Estas secciones se han diseñado para ser intuitivas y fáciles de usar, por lo que los usuarios pueden acceder rápidamente a los recursos sin la necesidad de estar registrados.

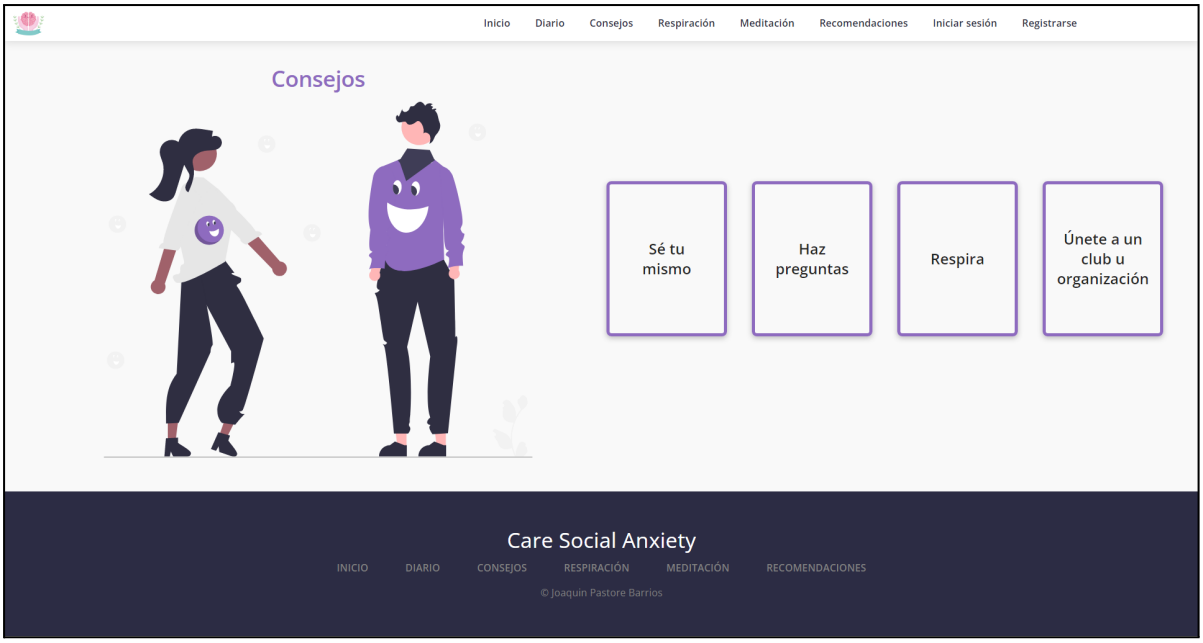


Ilustración 5: Sección de consejos

Fuente: Propia

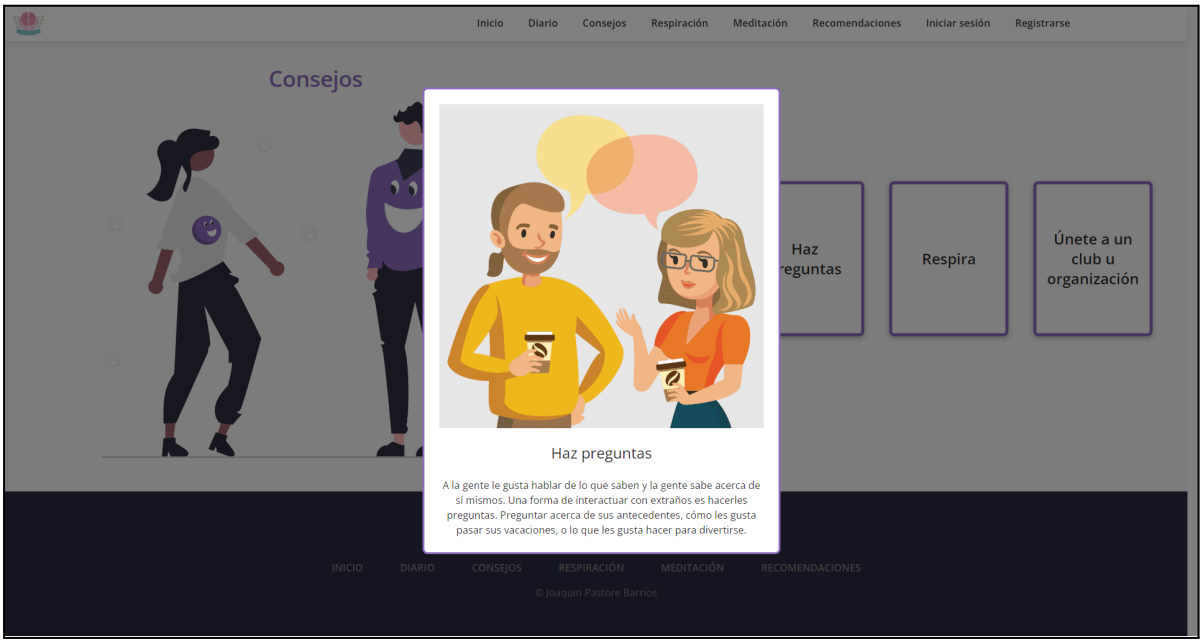


Ilustración 6: Sección consejos - Modal con información adicional

Fuente: Propia

V. Registro

Para acceder a la funcionalidad principal de la aplicación, el diario personal, es necesario registrarse. Al hacer clic en el botón "Registrarse", se abrirá una página con un formulario que deberá completarse con los datos requeridos: nombre de usuario, correo electrónico y contraseña. Una vez completado el formulario, se debe hacer clic en el botón "Registrarse" para crear una cuenta.

Además, los usuarios registrados también podrán recibir actualizaciones y notificaciones sobre nuevas funcionalidades y actualizaciones de la aplicación, así como tener acceso a soporte técnico en caso de tener algún problema o duda sobre su uso.

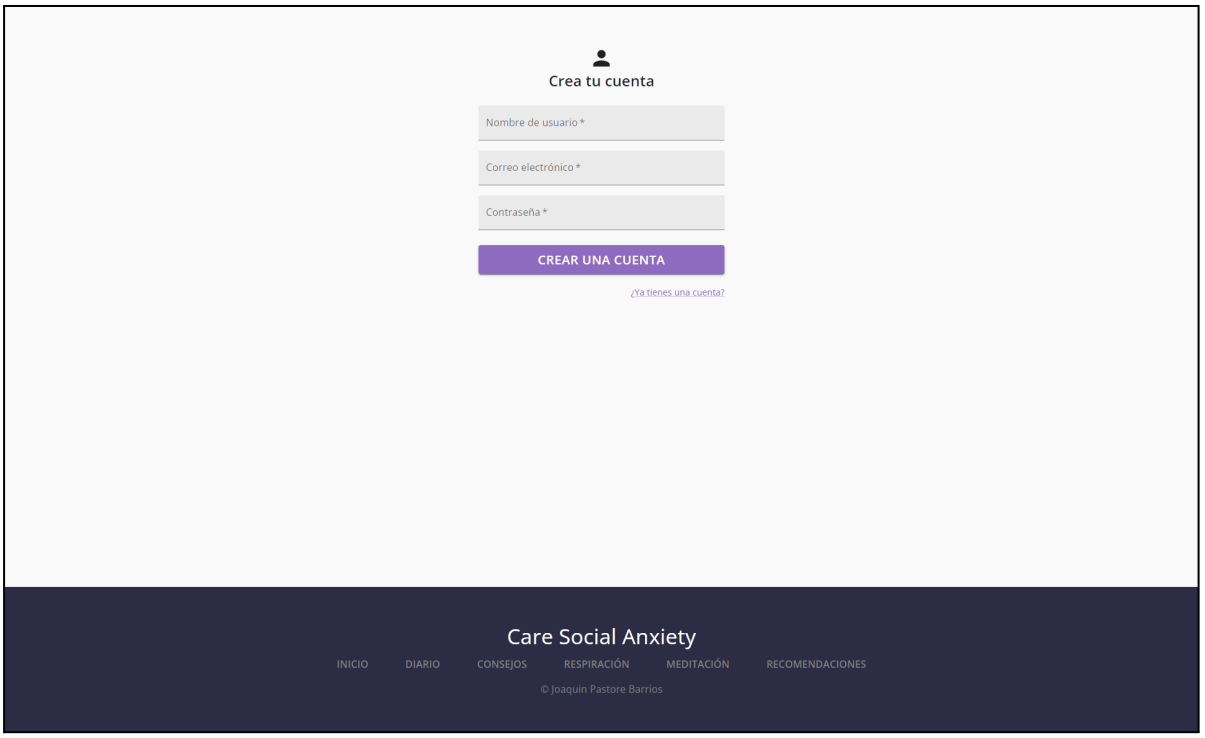


Ilustración 6: Sección creación de cuenta

Fuente: Propia

VI.Iniciar sesión

Para iniciar sesión en la aplicación, el usuario debe introducir su correo electrónico y contraseña previamente registrados. Tras completar el formulario, deberá apretar el botón de “INICIAR SESIÓN”, que le redireccionará a la página principal de la aplicación. Además, tras realizar dicha acción, aparecerá disponible la sección del diario personal y la de mi perfil. Por otra parte, si se ha olvidado la contraseña, se puede seleccionar la opción "¿Olvidó su contraseña?" y realizar todo el proceso para restablecerla. Por otra parte, si el usuario no tiene una cuenta, debe seleccionar la opción "Registrarse" que le llevará a la sección de registro para crear una nueva cuenta.

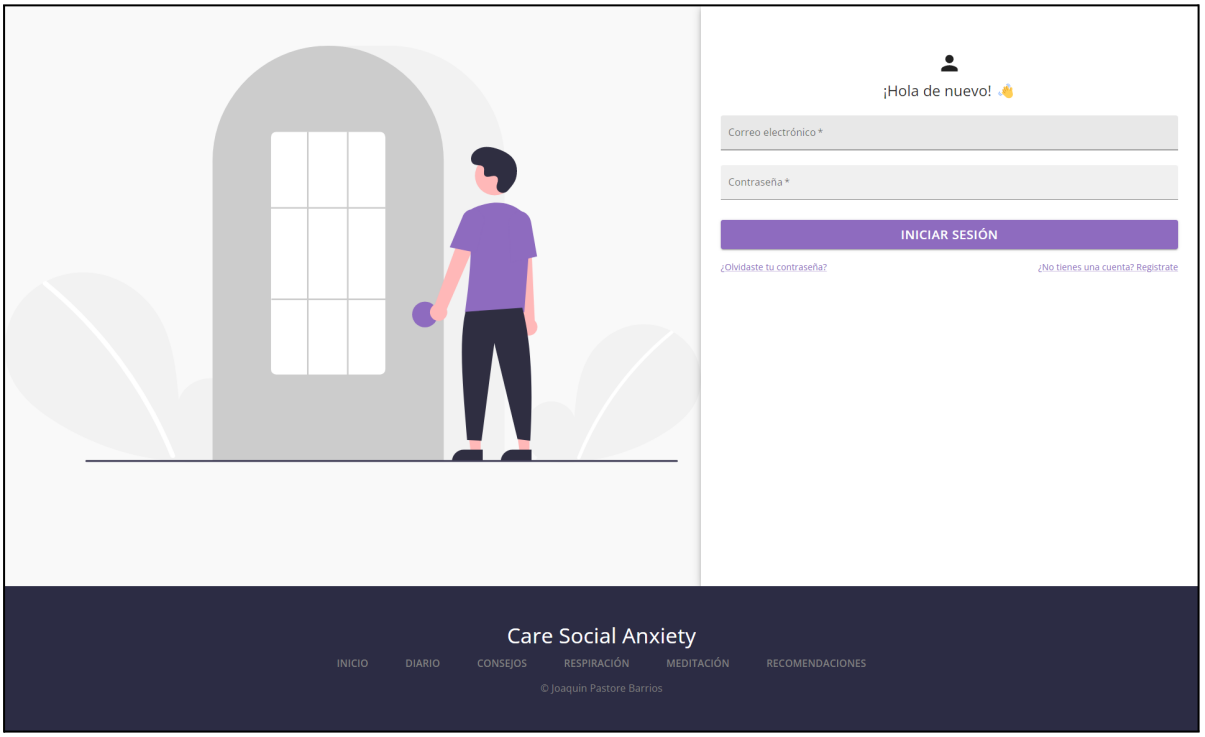


Ilustración 7: Sección iniciar sesión

Fuente: Propia

Este proceso se empieza en colocar el correo electrónico asociado a la cuenta. A continuación, se recibe un correo donde se explica a la dirección donde tiene que dirigirse el usuario. Por último, en esta sección se debe colocar la nueva contraseña que se desea establecer. Si la acción ha sido completada con éxito, aparecerá un mensaje de confirmación.

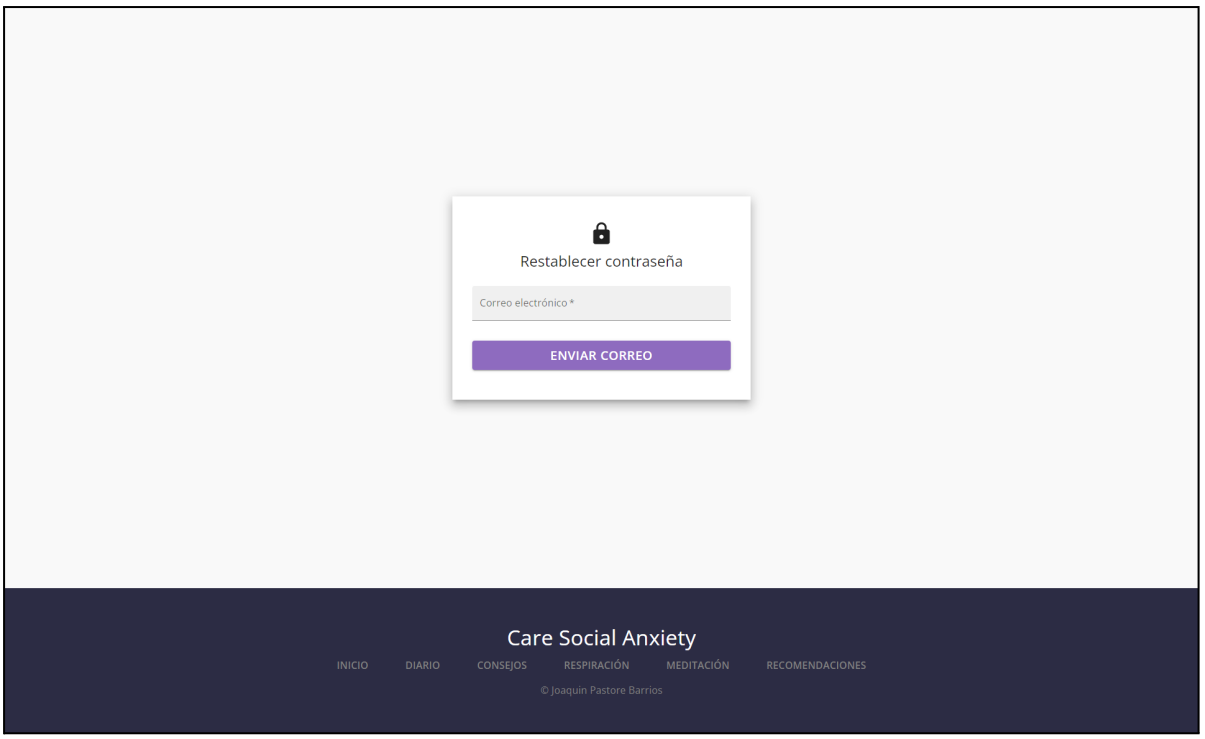


Ilustración 8: Sección Restablecer contraseña

Fuente: Propia

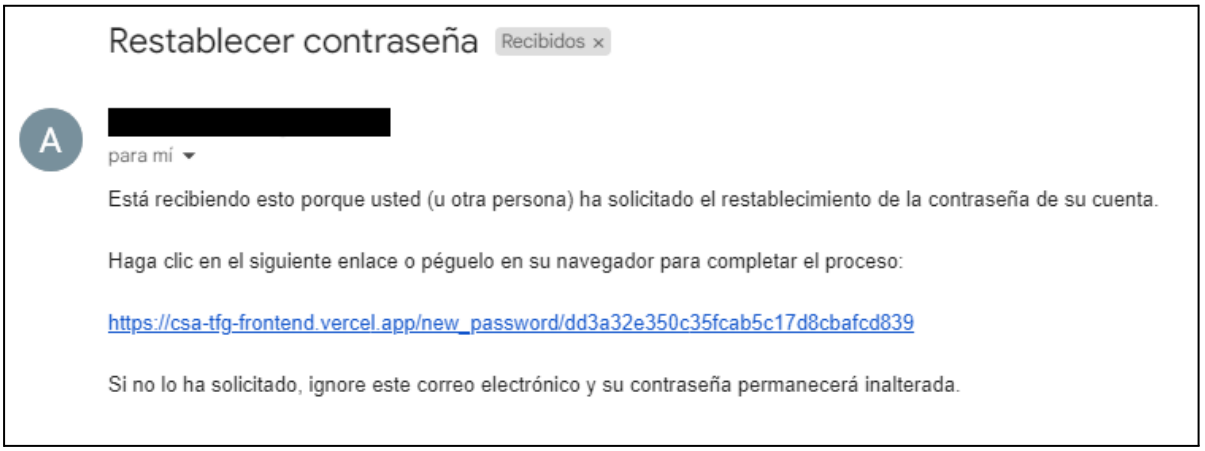


Ilustración 9: Correo explicativo del proceso de restablecimiento de contraseña

Fuente: Propia

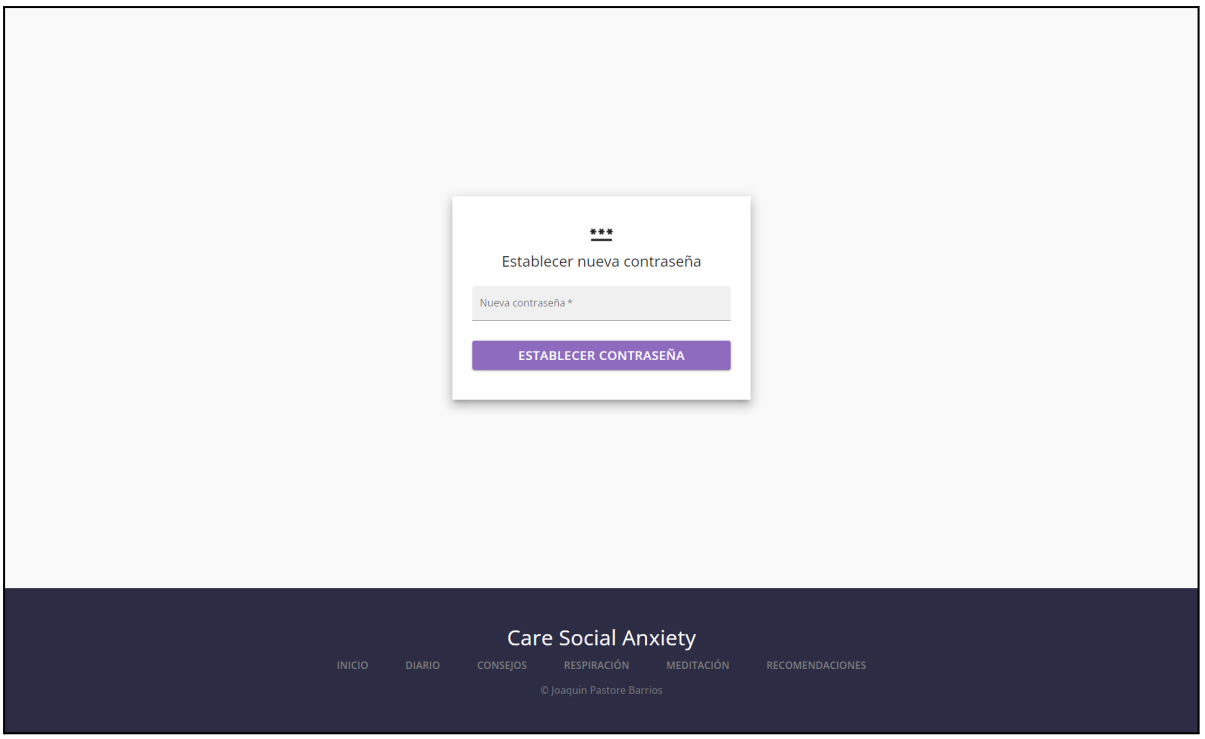


Ilustración 10: Correo explicativo del proceso de restablecimiento de contraseña

Fuente: Propia

Una vez iniciada sesión, el usuario tendrá acceso a todas las funcionalidades de la aplicación web disponibles para su cuenta, como el diario personal, el acceso a su perfil y en caso de ser un usuario administrador/moderador, aparecerá disponible la sección de administración de recursos de la aplicación. Si se desea cerrar sesión, se debe seleccionar la opción correspondiente en el menú desplegable de la cuenta.

VII.Panel de administrador y moderador

El panel de administrador y moderador está diseñado para ser utilizado por los miembros del equipo encargados de supervisar y gestionar la aplicación. Aquí se localizan una serie de herramientas y funcionalidades para ayudar en dicha tarea. Una de las funciones más importantes del panel de administrador y moderador es la posibilidad de revisar y moderar el contenido exhibido para los usuarios.

Desde esta sección, se podrá revisar y eliminar cualquier contenido inapropiado, que incumpla las normas de la aplicación o esté desactualizado, así como crear nuevos recursos

para garantizar la actualización recurrente en la aplicación. Cabe destacar que el administrador y moderador en cuanto a los recursos comparten los mismos permisos, excluyendo la gestión de usuarios, que solo queda relevada a los usuarios administradores.

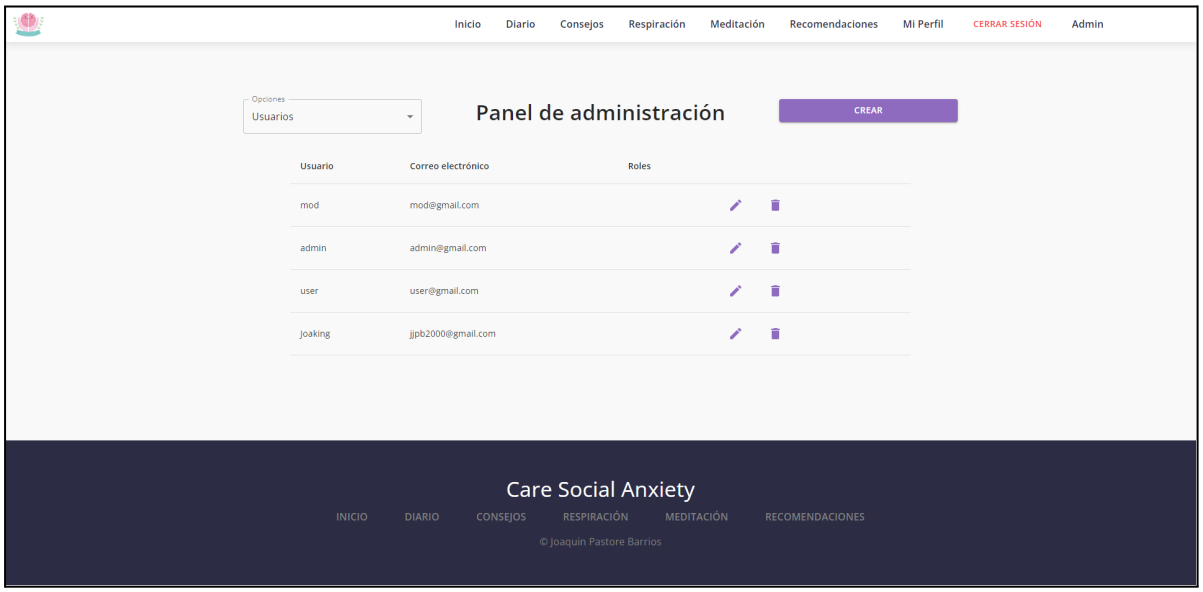


Ilustración 10: Correo explicativo del proceso de restablecimiento de contraseña

Fuente: Propia

Todos los recursos comparten la misma funcionalidad de creación, actualización y eliminación. Para seleccionar qué recursos se quieren moderar se dispone de un selector en la parte izquierda de la sección. En la parte derecha se encuentra el botón de creación, que despliega una ventana emergente con un formulario con los campos requeridos para crear el recurso. Tras crear dicho recurso aparecerá un mensaje de confirmación y estará disponible en la sección correspondiente a su categoría y en la tabla de recursos. Para editar sigue el mismo esquema de antes y con el mismo icono, este botón lanzará una ventana emergente con los campos correspondientes y un botón de actualizar. Por último, el botón de eliminar sigue también con el mismo icono y cumple con la funcionalidad correspondiente.

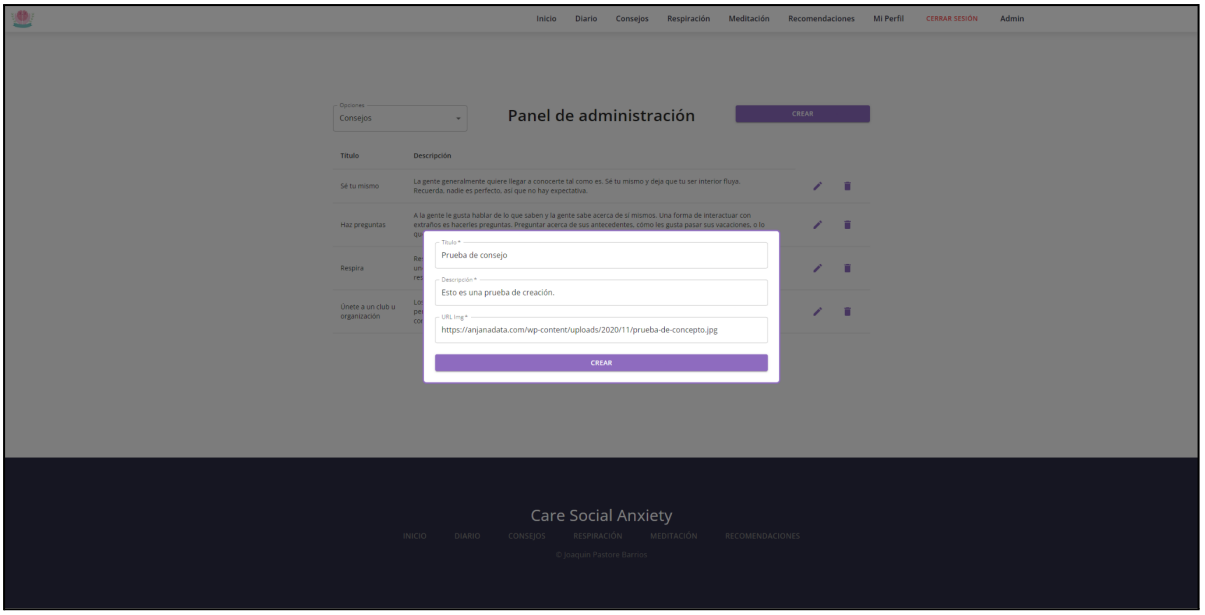


Ilustración 11: Ventana emergente de creación de un recurso

Fuente: Propia