Treball de Fi de Grau

Títol

Desarrollo de una plataforma web para cursos interactivos

Autoria

Albert Arrebola Sans

Professorat tutor

Ramon Voces Merayo, Belén Monclús Blanco

Grau

Comunicació Audiovisual	
Periodisme	
Publicitat i Relacions Públiques	
Comunicació Interactiva	x
Comunicació de les Organitzacions	

Tipus de TFG

Projecte	x
Recerca	

Data

Del 3 al 7 de juny de 2024	x
26 de juliol de 2024	

Full resum del TFG

Títol del Treball Fi de Grau:

Català:	Desenvolupament d'una plataforma web per cursos interactius			
Castellà:	Desarrollo de una plataforma web para cursos interactivos			
Anglès:	Development of a web platform for interactive courses			
Autoria:		Albert Arrebola Sans		
Professorat tutor:		Ramon Voces Merayo, Belén Monclús Blanco		
Curs:	2023/24	Grau:	Comunicació Audiovisual	
			Periodisme	
			Publicitat i Relacions Públiques	
			Comunicació Interactiva	x
			Comunicació de les Organitzacions	

Paraules clau (mínim 3)

Català:	web, react, aplicació, desenvolupament, javascript
Castellà:	web, react, aplicación, desarrollo, javascript
Anglès:	web, react, application, development, javascript

Resum del Treball Fi de Grau (extensió màxima 100 paraules)

Català:	Desenvolupament duna plataforma de cursos interactius amb React. L'aplicació compta amb una interfície intuïtiva per als usuaris i un sistema administratiu que permet al creador de cursos gestionar eficientment continguts i usuaris. El projecte combina facilitat dús amb una funcionalitat robusta per oferir una solució integral daprenentatge i administració.
Castellà:	Desarrollo de una plataforma de cursos interactivos con React. La aplicación cuenta con una interfaz intuitiva para los usuarios y un sistema administrativo que permite al creador de cursos gestionar contenidos y usuarios eficientemente. El proyecto combina facilidad de uso con una funcionalidad robusta para ofrecer una solución integral de aprendizaje y administración.
Anglès:	Development of an interactive course platform with React. The application has an intuitive interface for users and an administrative system that allows course creator to manage content and users efficiently. The project combines ease of use with robust functionality to offer a comprehensive learning and administration solution.

Agradecimientos

Este trabajo es el resultado de mucho esfuerzo, mucho tiempo y el apoyo de muchas personas. Me gustaría empezar dando las gracias a la primera que me viene a la cabeza, mi padre. Gracias *papá* por tu conocimiento infinito, tu paciencia y por la pasión que me has transmitido siempre por el desarrollo web. A mi familia, y amigos, por apoyarme siempre en cada paso que doy, por soportar mis quejas, mis soliloquios y por animarme cuando nada salía.

Agradecimiento especial a Pol, muchas gracias amigo por tu tiempo, tus consejos, por ayudarme sin esperar nada a cambio y por los componentes tan bonitos que podréis ver más adelante. Sin tí esto hubiese sido imposible. Y a tí, Pau, por contar conmigo en este y en tantos proyectos y por confiar en mí incluso más de lo que yo lo hago.

Por último, gracias a Ramon y a Belén. A tí Ramon por guiarme en este proyecto, por escucharme, frenarme y aconsejarme en el momento adecuado, pero sobre todo por ser siempre tan positivo. Y Belén, gracias por dedicarme un tiempo que no tienes y por empujarme cuando el viento soplaba en contra.

Índice de contenidos

Índice de figuras

1. Introducción y contexto

En el contexto actual de la educación, las tecnologías digitales han emergido como fuerzas transformadoras, redefiniendo los paradigmas educativos y ampliando significativamente el acceso al conocimiento (Nguyen, 2023). Según un artículo publicado el pasado año por Aristovnik, Karampelas, Umek y Ravšelj en la revista Frontiers, el *e-learning*, en particular, ha surgido como una herramienta poderosa, ofreciendo una alternativa flexible y accesible a la educación tradicional, sobre todo desde la pandemia del Covid-19 (Aristovnik *et al*., 2023). Plataformas en línea, aplicaciones educativas y recursos digitales han democratizado el aprendizaje, permitiendo a individuos de todos los ámbitos de la vida adquirir nuevas habilidades y conocimientos desde la comodidad de sus hogares y accediendo a estas clases de manera asíncrona en muchos casos. Esta revolución educativa ha generado un interés creciente en el desarrollo de aplicaciones y plataformas innovadoras que fomentan el aprendizaje interactivo y personalizado.

En este contexto, se enmarca el presente trabajo, que se centra en la **creación de una aplicación web de cursos interactivos,** diseñada para satisfacer las necesidades educativas contemporáneas y ofrecer una experiencia de aprendizaje enriquecedora y efectiva.

El desarrollo de este proyecto surge a raíz de una necesidad; la empresa Rispot Consulting analizó el mercado y vió que la oferta de cursos online que había era excesivamente cara, muy plana en contenido y con muy poca motivación más allá del precio de estos. Este proyecto pretende responder a la carencia planteada por el cliente en cuestión. La empresa me contrató con el objetivo de desarrollar una plataforma que les permitiera crear cursos personalizados según sus propios criterios, sin depender de terceros y con la capacidad de diferenciarse significativamente de la competencia.

El presente trabajo se estructura en tres capítulos fundamentales (estado de la cuestión, alcance y gestión del proyecto y desarrollo del proyecto). En primer lugar, se abordará la **fase de conocimiento del entorno general del trabajo** desde un prisma teórico.

A continuación, se presentará una explicación genérica de todas las **tecnologías seleccionadas** para la plataforma y se detallarán las razones detrás de cada elección tecnológica. Esta sección proporcionará una comprensión clara de la infraestructura técnica y las soluciones que sustentan la plataforma.

Finalmente, se desarrollará un proceso narrativo que explicará **cómo se han implementado estas tecnologías en cada fase del proyecto**. Esta parte del trabajo abordará el desarrollo de la aplicación, el diseño, la implementación y las pruebas realizadas para asegurar la calidad y funcionalidad de la plataforma.

2. Objetivos

Partiendo de esta necesidad, Rispot Consulting contrató mis servicios para realizar esta plataforma donde dar soporte a sus cursos interactivos relacionados en este caso con el sector de la publicidad y el comercio electrónico mediante Amazon.

El objetivo principal del proyecto era crear una aplicación con una interfaz intuitiva que permitiera a los usuarios finales acceder y disfrutar de cursos personalizados. Al mismo tiempo, se buscaba diseñar e implementar un sistema administrativo integral. Este sistema debía facilitar al creador de cursos la gestión eficiente de contenidos y usuarios. Además, se priorizó desarrollar una plataforma escalable, capaz de expandirse fácilmente para incorporar más cursos y usuarios en el futuro.

Existen una serie de objetivos secundarios o complementarios que fueron completamente necesarios para la realización de la plataforma. Estos incluyeron la investigación de las tecnologías disponibles, la selección de aquellas que mejor se ajustaban al proyecto, el proceso de familiarización con su funcionamiento y la planificación detallada del diseño y desarrollo del proyecto en su totalidad.

3. Estado de la cuestión

Una vez definidos los objetivos, es necesario ver qué hay actualmente que nos permita desarrollar este proyecto. Esta parte abordará el ya introducido contexto desde cuatro puntos de vista distintos, respondiendo a **cuatro preguntas**: ¿de qué trata la plataforma? (*e-learning*), ¿qué elementos hay en una aplicación como la que aborda este trabajo?, ¿cuántas maneras hay para construirla? (*stacks* tecnológicos) y, por último, ¿cómo se organiza el desarrollo de una aplicación como esta? (metodología).

3.1 El concepto e-learning

Según el experto en tecnología educativa Elliot Masie, "el *e-learning* es el uso de la tecnología de redes para diseñar, entregar, seleccionar, gestionar y ampliar el aprendizaje" (Masie, 1999 en Ispring, 2023).

Desde la primera vez que Elliot pronunció esas palabras en la conferencia de TechLearn, (Ispring, 2023) hasta hoy, este concepto ha ido tomando forma y adaptándose a las nuevas generaciones. Actualmente, cuando hablamos de *e-learning* nos estamos refiriendo a cualquier formación online en cualquier dispositivo digital. Desde la pandemia del Covid-19, es un modelo de aprendizaje que ha cogido mucha fuerza y relevancia en el día a día de la educación (Aristovnik *et al.*, 2023).

Características y tipos

Según un artículo publicado en el portal *ispring.es* por Colman (2023) las características del *e-learning* a nivel general son los siguientes:

1. **Flexibilidad**: El *e-learning* se adapta fácilmente a las necesidades de cada organización, permitiendo una formación personalizable y adaptable a los horarios y requerimientos individuales.
2. **Seguimiento**: Las herramientas de *e-learning* proporcionan funciones para monitorear el progreso y desarrollo de los estudiantes de manera cercana y precisa.

3. **Comodidad**: Al no estar limitado por horarios o ubicaciones físicas, el *e-learning* ofrece la conveniencia de poder aprender y enseñar desde cualquier lugar y en cualquier momento, facilitando la participación y la accesibilidad.

4. **Canales de comunicación**: Facilita la interacción entre profesores y estudiantes a través de diversos canales de comunicación, promoviendo la participación y permitiendo establecer vínculos educativos sólidos, incluso fuera de los horarios de enseñanza.

5. **Recursos de diseño**: Las plataformas de *e-learning* ofrecen herramientas para el diseño de cursos y programas de capacitación de manera sencilla y atractiva, permitiendo a los educadores estructurar y presentar los contenidos de forma efectiva sin necesidad de tener conocimientos técnicos avanzados en diseño y elaboración de materiales educativos.

Respecto a la **tipología**, no todos los cursos de aprendizaje online siguen la misma estructura y metodología. Dependiendo de las necesidades se han ido desarrollando diferentes. La misma revista *Ispring* (2023) recoge en su artículo estas 4 tipologías:

1. **Sincrónico**: Esta modalidad incluye la interacción en tiempo real entre maestros y estudiantes, similar a las clases presenciales pero a través de videoconferencias. Facilita una comunicación directa y dinámica durante el proceso de enseñanza.

2. **Asincrónico**: En esta modalidad, los estudiantes pueden acceder y participar en contenido educativo a su propio ritmo a través de plataformas en línea. Esto les permite adaptar sus horarios de estudio según sus necesidades individuales, brindando una mayor flexibilidad en el proceso de aprendizaje.

3. ***M-learning***: Se refiere al aprendizaje a distancia utilizando dispositivos móviles, como teléfonos inteligentes y tabletas. Esta modalidad permite a los estudiantes acceder al contenido educativo en cualquier momento y lugar, haciendo que el aprendizaje sea altamente accesible y conveniente.

4. ***B-learning*:** Es una modalidad híbrida que combina la enseñanza presencial tradicional con actividades realizadas en plataformas de aprendizaje en línea. Este enfoque permite integrar distintos métodos de enseñanza, aprovechando las ventajas de ambos entornos para ofrecer una experiencia de aprendizaje más completa y versátil.

3.2 Características de una aplicación

La plataforma de cursos interactivos de Rispot, no deja de ser una aplicación web que contempla una parte visual y una parte interna, donde se encuentra la base de datos que almacena, actualiza, controla y gestiona toda la información que se mostrará, o no, en la interfaz web. A continuación en base a mi formación universitaria y experiencia se hace una descripción de las **fases o características** que considero comportan una **aplicación** de este estilo desde un prisma relativamente genérico:

1. **Desarrollo *frontend*:** Esta fase se enfoca en la parte visual y la interacción del usuario con la aplicación. Habitualmente se utilizan tecnologías como HTML (*HyperText Markup Language*), CSS (*Cascading Sheet Styles*) y *JavaScript*, junto con frameworks como React, Angular o Vue.js, para crear la interfaz de usuario y la lógica interactiva. Los componentes de la interfaz se diseñan y se implementan para proporcionar una experiencia de usuario intuitiva y atractiva.
2. **Diseño y experiencia de usuario (UX/UI)**: Se presta atención al diseño de la interfaz de usuario para garantizar una experiencia fluida y atractiva para los usuarios. Se aplican principios de diseño centrados en el usuario y se realizan pruebas de usabilidad para obtener retroalimentación y mejorar la experiencia del internauta.
3. **Funcionalidades de la aplicación**: Se desarrollan las funcionalidades específicas de la aplicación, como la gestión de contenido, la interacción del usuario, las características de personalización y cualquier otra característica única que agregue valor al producto. En el caso de una plataforma de cursos, esto incluiría la presentación del contenido del curso, cuestionarios, seguimiento del progreso del usuario, etc.
4. **Gestión de Datos (*backend)***: Se establece una arquitectura de *backend* para manejar la lógica de negocio, la autenticación de usuarios, el almacenamiento de datos y cualquier otra operación necesaria para que la aplicación funcione correctamente. Se pueden utilizar tecnologías como Node.js, Python/Django, Java/Spring, entre otras, junto con bases de datos como PostgreSQL, MySQL o MongoDB. Si la arquitectura es externa se conoce como *"backend as a service".*

5. **Integración de recursos externos**: Se exploran y se integran recursos externos y servicios de terceros para ampliar las funcionalidades de la aplicación. Esto puede incluir APIs (*Application Programming Interface*) servicios como pagos, autenticación, almacenamiento de archivos, entre otros, así como el uso de librerías externas para agregar características específicas.

6. **Optimización y seguridad**: Se optimiza el rendimiento de la aplicación y se implementan medidas de seguridad para proteger los datos del usuario y prevenir posibles vulnerabilidades. Esto incluye la optimización del código, la gestión de la seguridad en la capa de aplicación y la aplicación de buenas prácticas de seguridad en el almacenamiento y transmisión de datos.

7. **Pruebas y depuración**: Se realizan pruebas exhaustivas de la aplicación para identificar y corregir errores. Esto incluye pruebas unitarias, de integración y de aceptación, así como pruebas de rendimiento y seguridad. La depuración se realiza para solucionar cualquier problema que surja durante las pruebas.

8. **Despliegue y mantenimiento**: Se planifica y se ejecuta el despliegue de la aplicación en un entorno de producción. Esto puede implicar la configuración de servidores, la gestión de dominios y certificados SSL (*Secure Sockets Layer*), entre otros aspectos. Además, se establecen estrategias de mantenimiento a largo plazo para garantizar que la aplicación siga siendo funcional y segura, incluyendo actualizaciones regulares, monitoreo del rendimiento y resolución proactiva de problemas.

3.3 *Stacks* tecnológicos

En 1957, John W. Backus creó Fortran, el primer lenguaje de programación diseñado específicamente para realizar cálculos numéricos y científicos de manera eficiente. Con la aparición de Internet en la década de 1990, se produjo un rápido incremento en la creación de nuevos lenguajes de programación, acelerando significativamente el ritmo de desarrollo tecnológico hasta la actualidad (Gómez, 2023).

Hace no más de 30 años, las opciones para construir una aplicación eran limitadas a una única metodología. En contraste, hoy en día existen numerosas tecnologías y caminos diversos para

alcanzar el mismo objetivo. La agrupación de estas tecnologías y lenguajes se denomina *stack*, y a continuación se comentan algunos de los más populares en el sector en 2024 (Inaba, 2023). Es importante señalar que, debido a la extensión del presente TFG, no se abordarán todos los lenguajes, aunque se proporcionará más adelante una explicación detallada de los lenguajes y *frameworks* seleccionados para este trabajo.

MEAN: Utiliza MongoDB como base de datos NoSQL, Express.js como *framework* de *backend*, Angular como *framework* de *frontend*, y Node.js como entorno de ejecución de JavaScript en el servidor. Angular, mantenido por Google, ofrece una estructura robusta y utiliza TypeScript para una sintaxis más estricta y robusta, lo que lo hace ideal para aplicaciones de gran escala y complejidad (Pineda, 2017).

MERN: Este *stack* reemplaza Angular con React.js como el *framework* de *frontend*, manteniendo MongoDB, Express.js y Node.js. React.js, desarrollado por Facebook, se centra en componentes reutilizables y ofrece una gran flexibilidad y libertad para los desarrolladores. Esto hace que el stack MERN sea popular para el desarrollo rápido de aplicaciones web interactivas y escalables (Parada, 2023).

MEVN: Es similar al anterior, pero utiliza Vue.js en lugar de React.js como *framework* de *frontend*. Además de MongoDB, Express.js y Node.js, Vue.js ofrece una curva de aprendizaje suave y una sintaxis intuitiva, lo que lo convierte en una opción popular para desarrolladores que prefieren trabajar con Vue.js y desean construir aplicaciones web modernas y escalables (Solera, 2020).

LAMP: Es un acrónimo que representa Linux (sistema operativo), Apache (servidor web), MySQL (base de datos relacional) y PHP (lenguaje de programación del lado del servidor). Aunque PHP es el componente principal del *backend*, también se puede utilizar Python o Perl (Kyle, 2023).

Serverless: Esta arquitectura prescinde de servidores físicos y se basa en servicios en la "nube" para la lógica de *backend*. Se pueden utilizar servicios como AWS Lambda, API Gateway, DynamoDB, Firebase o Supabase (para la base de datos), y un *framework* de *frontend* como React o Vue.js para desarrollar la aplicación (Roberts, 2018).

JAMstack: JAMstack es una arquitectura moderna que se basa en JavaScript, APIs y archivos *markdown*. Utiliza tecnologías estáticas como JavaScript para el *frontend* (por ejemplo React o Vue.js), servicios de terceros o APIs para la lógica del *backend*, y generadores de sitios estáticos como Gatsby o Next.js para crear y desplegar el contenido estático del sitio (Haider, 2021).

Ruby on Rails: Ruby on Rails es un *framework* de desarrollo web que utiliza el lenguaje de programación Ruby. Es conocido por su enfoque en la productividad y la convención sobre configuración. Se integra con una variedad de bases de datos, como PostgreSQL o MySQL, y es compatible con tecnologías *frontend* como JavaScript, HTML y CSS (Łata, 2023).

Django Stack: Django es un framework de desarrollo web de alto nivel para Python. Proporciona una arquitectura sólida para la construcción rápida de aplicaciones web seguras y escalables. Django se integra bien con bases de datos como PostgreSQL, MySQL o SQLite, y es compatible con tecnologías *frontend* como JavaScript, HTML y CSS (Lobo, 2023).

3.4 Metodologías de desarrollo de software

Las metodologías de desarrollo de *software* han evolucionado desde los sistemas tradicionales de 1960 hasta las opciones modernas que conocemos hoy en día (Gómez, 2023). Su finalidad es proporcionar una estructura para que los equipos de desarrolladores trabajen eficientemente, facilitando la comunicación interna y externa, optimizando tiempos, costos y reduciendo riesgos (Gamboa y Arreaga, 2018).

Existen diversas metodologías de desarrollo de software, cada una con sus propias características y ventajas. Una metodología de desarrollo de *software* es un marco de trabajo utilizado para estructurar, planificar y controlar el proceso de desarrollo de un proyecto

tecnológico. Tiene por objetivo establecer un enfoque sistemático en el desarrollo informático (GooApps, 2022). Según la revista GooApps encontramos la definición y las cinco mejores metodologías de desarrollo:

1. **Agile**: Se centra en la satisfacción del usuario y en la flexibilidad durante el desarrollo, dividido en *sprints* cortos con iteraciones y *feedback* continuo.
2. **SCRUM**: Basada en el marco *agile*, utiliza *sprints* para abordar cambios y se enfoca en la colaboración y el compromiso del equipo.
3. **Cascada**: Un método lineal y secuencial donde las etapas se estructuran de forma fija, adecuado para proyectos con requisitos claros y predecibles.
4. **Lean**: Maximiza la productividad reduciendo actividades poco productivas y enfocándose en el valor para el usuario, con una mejora continua del proceso.
5. **Rapid Application Development (RAD)**: Busca obtener resultados finales en poco tiempo mediante prototipos y pruebas iterativas, adecuado para proyectos con alta participación del cliente y equipos altamente cualificados.

Cada metodología tiene sus propias ventajas e inconvenientes, y la elección depende de los requisitos y características específicas de cada proyecto.

4. Alcance del proyecto

En desarrollos de este tipo es muy importante saber muy bien qué es lo que se necesita, hasta donde se puede llegar y sobre todo qué se tiene que poder hacer y qué no en el producto final. Definir unos objetivos a cumplir, los casos de uso, y las fases de desarrollo ha sido imprescindible y es lo que conforma lo que me gusta llamar: alcance del proyecto.

4.1 Fases del proyecto

Para la consecución de la aplicación se definieron una serie de fases o "metas" que han sido el hilo conductor en todo momento y han marcado las etapas necesarias para obtener el resultado final.

1. Definición del proyecto y requisitos básicos.
2. Definición de las versiones.
3. Planificación del proyecto.
4. Diseño de la interfaz.
5. Maquetación del *frontend*.
6. Programación del *backend*.
7. Integración del *frontend* y *backend* en la aplicación.
8. Análisis de usabilidad.
9. Despliegue en producción.

4.2 Requisitos

Los requisitos son elementos fundamentales que describen las características, funcionalidades y restricciones que debe cumplir el sistema para satisfacer las necesidades de los usuarios y cumplir con los objetivos del proyecto. En este proyecto se distingue entre requisitos funcionales y requisitos no funcionales.

4.2.1. Requisitos funcionales

Los requisitos funcionales definen las acciones específicas que el sistema debe ser capaz de realizar, así como las interacciones con los usuarios y otros sistemas.

4.2.1.1. Casos de uso

Estos requisitos se basan en los casos de uso identificados previamente y describen cómo el sistema debe comportarse en diferentes situaciones. Los casos de uso son escenarios detallados que representan las diferentes acciones que los usuarios pueden realizar en la aplicación y cómo el sistema responde a estas acciones.

Teniendo en cuenta esto, se definen los casos de uso para los tres roles que existirán en la plataforma. En primer lugar, encontramos el **"rol administrador"**, que tendrá el control total, accesos para gestionar y administrar todos los usuarios y aspectos de la aplicación. Podrá crear nuevos cursos y gestionar cualquier módulo y lección dentro de estos.

Por otro lado existirá el **"rol usuario registrado"**. Este será un rol con muy pocos permisos, limitado a iniciar sesión con sus credenciales, ver su perfil y realizar los cursos como consumidor.

Finalmente es necesario contemplar el **"rol usuario no registrado"** que pertenece a esa persona que llega al sitio web o plataforma pero no tiene ningún acceso. En este caso estará aún más limitado que el anterior. A continuación se muestran tres tablas con los casos de usos detallados en función del rol mencionado.

Rol	**Casos de Uso**
ADMIN	• Iniciar y cerrar sesión • Ver cualquier usuario (perfil) • Eliminar cualquier usuario • Editar cualquier usuario • Administrar roles y accesos • Crear cursos y módulos • Editar cursos y módulos • Eliminar cursos y módulos • Relacionar módulos con cursos

	● Relacionar usuarios con cursos ● Agregar, editar y eliminar contenido de cualquier curso ○ Video, actividades, materiales, exámenes, etc ● Preguntas diarias y consejos diarios (CRUD)

Fig.1: Casos de uso usuario administrador

Fuente: Elaboración propia.

Rol	Casos de Uso
REGISTRADO	● Iniciar y cerrar sesión ● Ver catálogo de cursos ● Ver sus cursos ● Ver y realizar contenido al que tiene acceso ● Ver y editar su perfil

Fig.2: Casos de uso usuario registrado

Fuente: Elaboración propia.

Rol	Casos de Uso
NO REGISTRADO	● Registrarse ● Ver catálogo de cursos

Fig.3: Casos de uso usuario no registrado

Fuente: Elaboración propia.

4.2.2. Requisitos no funcionales

Los requisitos no funcionales son aquellos que describen las características y restricciones del sistema que no están relacionadas directamente con sus funciones específicas. Estos requisitos abordan aspectos como el rendimiento, la seguridad, la escalabilidad y la usabilidad del sistema (AltexSoft, 2023).

Esta plataforma *e-learning* requiere contemplar estos tres ejes desde las siguientes estrategias:

1. **Usabilidad y accesibilidad**: Diseñar una interfaz intuitiva, sencilla, con contrastes claros, y con información estructurada correctamente en la pantalla. Uso de menús que separan los apartados, mensajes de errores y éxito para que el usuario sepa en todo momento qué ocurre mientras navega.
2. **Escalabilidad**: La escalabilidad en esta aplicación se centra en dos objetivos claros: poder generar más de un curso de manera efectiva y funcional, y por otra parte dar acceso a multitud de usuarios y no solo a unos pocos.
3. **Rendimiento**: Este último es probablemente una característica en causa y consecuencia de las dos anteriores. Un mejor rendimiento mejora la usabilidad potencialmente, pero una mayor escalabilidad puede generar pérdida de rendimiento en muchas ocasiones. El objetivo es encontrar el equilibrio, manteniendo una relación coherente, programando con buenas prácticas y siendo lo más efectivo posible.

5. Gestión del proyecto

En un proyecto de esta envergadura, especialmente en el ámbito de la tecnología y el desarrollo de *software*, resulta inviable completar el trabajo sin una gestión adecuada del proyecto y una planificación meticulosa. Este apartado tiene como objetivo principal detallar las decisiones y estrategias adoptadas en términos de planificación, organización y metodologías empleadas, así como los enfoques técnicos específicos en programación que he elegido.

Se destacarán las metodologías de gestión de proyectos utilizadas para asegurar una ejecución eficiente y ordenada. Además, se explicarán las decisiones tomadas respecto a las tecnologías y lenguajes de programación, justificando su selección con base en su relevancia y eficacia para los objetivos del proyecto.

5.1 Metodología

Para desarrollar la aplicación en cuestión se han usado lo que se conocen como metodologías ágiles, un concepto de gestión de proyectos que implica la división del proyecto en fases, con unos tiempos definidos para cada fase y una continua evaluación y revisión en tiempo prácticamente real para asegurar la consecución exitosa del proyecto (Atlassian, 2021). Como sostiene Medina (2021), "Actualmente las empresas más competitivas del mercado, como Google, Amazon o Spotify, trabajan con métodos ágiles".

Pero dentro del paraguas de las metodologías ágiles (*agile* en inglés), hay muchas opciones que "beben de la misma fuente". En mi proyecto he usado la que se conoce como *Scrumban,* una combinación entre la metodología Scrum y la metodología Kanban. Scrum por un lado, tiene como fundamento las entregas parciales y regulares, ejecutadas en tiempos de no más de 3 o 4 semanas donde cada iteración o entrega debe ser un resultado completo y tangible. Luego, según los roles se realizan las revisiones correspondientes y la planificación de la siguiente iteración, entrega o *sprint*, un concepto que se usará mucho a lo largo de este trabajo (Hurtado, 2023).

Para la plataforma de cursos se definieron los siguientes *sprints:*

Fig.4: *Sprints* del proyecto en Github Projects

Fuente: Elaboración propia.

Si bien son tareas muy genéricas, su función principal es agrupar pequeños procesos en etiquetas. Cada sprint es una fase ciertamente genérica del proceso y en cada etiqueta el equipo de desarrollo decide qué tareas incluye en cada sprint, y se le indica una fecha de consecución o *deadline.* Estas tareas, en el contexto de metodologías ágiles toman el nombre de "historias". Las historias definidas para este proyecto se detallarán en el apartado 5.3. *Backlog* de producto.

Por otro lado encontramos la metodología Kanban, inventada y desarrollada por el ingeniero Japonés Taiichi Ohno en 1940 y que etimológicamente significa Kàn (señal visual) y Ban (tablero), ha ido evolucionando mucho hasta adaptarse al desarrollo de software. En un tablero, una "tarjeta Kanban" representa una tarea o historia, y esta historia avanza a través de las etapas del trabajo a medida que se finaliza (Laoyan, 2024).

Para este proyecto, he utilizado la popular aplicación de control de versiones GitHub (Gustavo, 2023), la cual me ha permitido almacenar todos los avances en el desarrollo del código. Además, GitHub soporta una interfaz Scrumban que ha facilitado la agrupación y

relación de todos los avances con los sprints de manera ordenada y visual mediante el uso de tarjetas. En mi caso, he creado las columnas: "To Do" (por hacer), "In Progress" (en progreso), "Reviewing" (en revisión) y "Done" (completado).

De esta manera, si las tareas están bien definidas en "To Do", a medida que se avanza en la construcción de la aplicación y el desarrollo del código, estas tarjetas se desplazan hasta la columna de "completado" como muestra la figura 5. Esto permite controlar en todo momento la evolución del proyecto y el estado de cada pequeño desarrollo realizado.

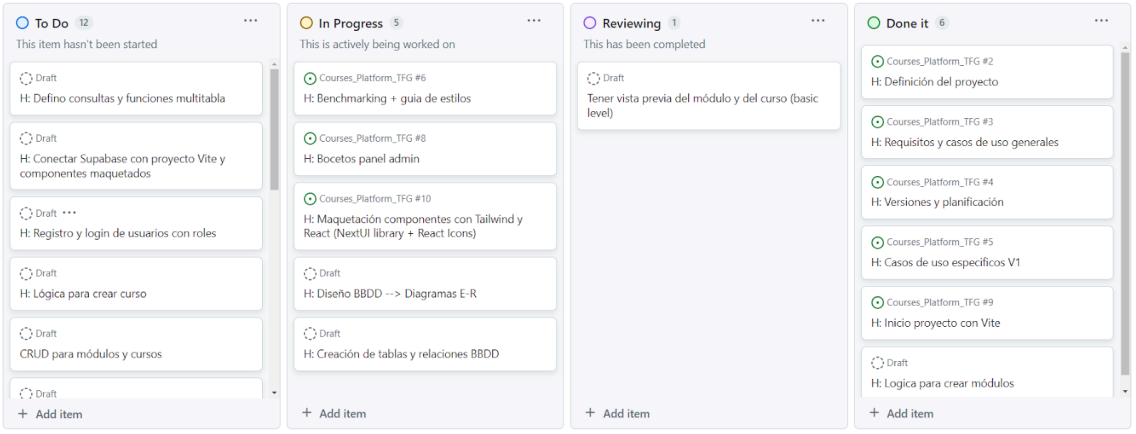

Fig.5: Tablero Kanban de Github Projects

Fuente: github.com (20/03/2024).

5.2 Entorno de desarrollo y tecnologías utilizadas

Para desarrollar una aplicación como esta hoy en día, hay multitud de caminos, existen decenas de lenguajes y modos de enfocar este proyecto. Estos caminos se conocen como stacks tecnológicos (desarrollados en la sección 3.3 del presente trabajo).

La intención ahora es la de explicar y detallar qué camino he escogido para la plataforma, qué tecnologías he usado, porqué he escogido estas y no otras y el funcionamiento principal de cada una de ellas para entender el nexo con el proyecto que nos ocupa.

Esta aplicación web se basa en la arquitectura *serverless* usando react como *framework* de *frontend* y Supabase como servicio *online* de *backend.*

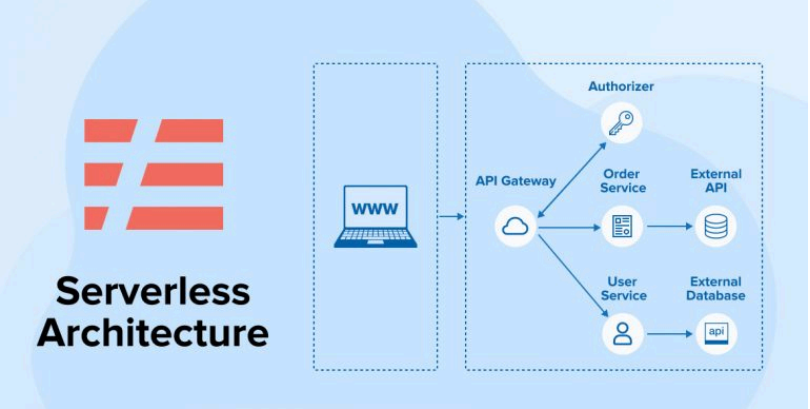

Fig.6: Esquema arquitectura *serverless*
Fuente: tatvasoft.com (17/03/2023).

Una aplicación web se puede dividir en dos grandes componentes: el *frontend* (parte visual donde interactúa el usuario) y el *backend* (parte interna, que incluye bases de datos y *endpoints* donde se configura la arquitectura del desarrollo). La arquitectura *serverless* (sin servidor en inglés) tiene como objetivo liberar a los desarrolladores de la responsabilidad de gestionar y mantener servidores, así como de los problemas relacionados con la infraestructura de *backend*, tales como las capacidades físicas, los límites y las complejidades. Esto les permite concentrarse en la funcionalidad principal de las aplicaciones (Shah, 2023).

En resumen, la página web que se visualiza en el navegador utiliza servicios de una API alojada en la "nube" que proporciona una infraestructura completa para el desarrollo. Esta API ofrece funcionalidades equivalentes a una "cocina bien equipada": suministro de ingredientes, un catálogo de recetas almacenadas, equivalentes a sistemas para gestionar y compartir información como bases de datos, autenticación y almacenamiento. Esta disposición permite a los desarrolladores concentrarse en la implementación y mejora de funcionalidades sin la necesidad de ocuparse de tareas fundamentales como la adquisición de recursos o la creación de infraestructura desde cero.

Para la plataforma de cursos interactivos, decidí usar esta arquitectura para poder centrarme en el desarrollo interactivo y funcional de la aplicación, olvidándome de toda la gestión de usuarios, *endpoints*, rutas, modelado de datos y sesiones. Existen una amplia gama de tecnologías *serverless*, pero en este caso se optó por **Supabase** por los siguientes motivos:

Supabase sigue el principio de "*backend as a service*" (*backend* como servicio en inglés) y su objetivo es permitir la creación de una base de datos de forma muy intuitiva desde un panel de control, enfocándose únicamente en definir las tablas y sus relaciones para desarrollar la aplicación. "Ellos" se encargan de gestionar el servicio de manera totalmente escalable, permitiéndote centrarte en el desarrollo frontal, visual e interactivo de tu aplicación (Murillo, 2022).

¿Qué cosas se pueden hacer con Supabase?, ¿por qué he escogido esta y no otras?, y por último; ¿cómo se ha implantado Supabase en el presente proyecto?... son las preguntas que trataré de responder a continuación:

¿Qué cosas se pueden hacer con Supabase?

Según un artículo de la revista digital Medium, permite:

- Escuchar cambios de las tablas tanto de manera asíncrona como en tiempo real.
- Hacer consultas a las tablas.
- Realizar CRUDs (*Create, Read, Update, Delete*).
- Gestión de usuarios de tu aplicación (ofrece servicio de roles, sesiones, permisos y autenticación con Google, Github, Apple y muchas otras plataformas externas).
- Interactuar con tu base de datos PostgreSQL desde una interfaz gráfica intuitiva.

¿Qué ventajas tiene?¿por qué he usado esta y no otra?

1. Fácil de usar: Supabase ofrece una interfaz intuitiva que simplifica la creación y gestión de bases de datos. Esto permite configurar rápidamente las tablas y relaciones necesarias para mi plataforma.
2. Basado en PostgreSQL: Al utilizar PostgreSQL, una base de datos reconocida por su robustez y confiabilidad, Supabase asegura que mi proyecto se beneficie de un rendimiento y seguridad sólidos. PostgreSQL es una tecnología madura y ampliamente adoptada, lo que me daba confianza en la estabilidad de mi aplicación (González, 2024).
3. Autenticación y autorización integradas: Supabase incluye herramientas integradas para la gestión de usuarios, lo cual es esencial para una plataforma de cursos

interactivos. Esto me permitió implementar fácilmente funciones de registro, inicio de sesión y control de acceso, garantizando que solo los usuarios autorizados pudieran acceder a ciertos contenidos.

Estas características combinadas me permitieron centrarme en desarrollar la funcionalidad principal de mi aplicación sin preocuparme por la complejidad de la infraestructura.

¿Cómo se ha usado *Supabase* en el proyecto?

Este servicio proporciona diversas opciones de desarrollo, entre las cuales he utilizado principalmente cuatro funcionalidades clave: gestión de usuarios (incluyendo autenticación, roles y sesiones), base de datos (para la gestión de tablas), almacenamiento de archivos multimedia (*storage*) y funciones avanzadas para consultas que involucran múltiples tablas (*Remote Procedure Calls*). Los detalles específicos sobre esta arquitectura *backend* se detallan en el sprint 3: Programación del *backend*.

Una vez explorada la estructura interna (o "trasera") del proyecto, es relevante analizar las tecnologías empleadas para desarrollar la interfaz visual e interactiva de la aplicación. Como se ha mencionado anteriormente, estas tecnologías abarcan un espectro variado; sin embargo, en este caso específico, opté por utilizar el *framework* React, respaldado por JavaScript ES6, junto con la biblioteca de estilos Tailwind CSS y Pol-UI.

> "React, es una popular biblioteca de JavaScript creada por Instagram que permite reutilizar fácilmente fragmentos de código al dividirlos en componentes. Estos componentes son funciones de JavaScript que devuelven HTML mediante una sintaxis llamada **JSX** (Javascript XML), lo que simplifica el desarrollo. Tailwind CSS, por otro lado, ofrece numerosas clases para construir rápidamente sitios web atractivos y responsivos. A diferencia de otros marcos como Bootstrap, que proporcionan estilos predefinidos para componentes, Tailwind adopta un enfoque funcional con clases reutilizables" (Andrade, 2023).

HTML y CSS juegan un papel fundamental en este contexto. HTML, o *HyperText Markup Language*, define la estructura básica y los elementos de una página web, como textos, imágenes y enlaces. CSS, *Cascading Style Sheets*, complementa HTML al controlar el diseño y la presentación visual de estos elementos, aplicando estilos como colores, fuentes y márgenes para mejorar la apariencia y la usabilidad del sitio web.

Una práctica muy habitual es la de usar librerías externas junto al *core* de React, en la aplicación que se presenta, las dos que quiero destacar son **React Router Dom**, para el sistema de enrutamiento de las páginas y **Pol-UI**, una librería de componentes atractivos y funcionales desarrollados a medida de la que se hablará más adelante.

Según palabras de Mauricio García en la revista digital Medium, se entiende enrutamiento como "el proceso de mantener sincronizada la URL con el contenido de una aplicación, permitiéndonos controlar el flujo de datos de la aplicación" (Garcia, 2021).

La combinación de React, HTML, CSS, Tailwind CSS y Pol-UI proporciona un marco robusto para desarrollar una aplicación web moderna, integrando la funcionalidad dinámica de JavaScript con la estructura y el estilo de HTML y CSS de manera eficiente y efectiva. El siguiente esquema muestra las tecnologías usadas para el desarrollo:

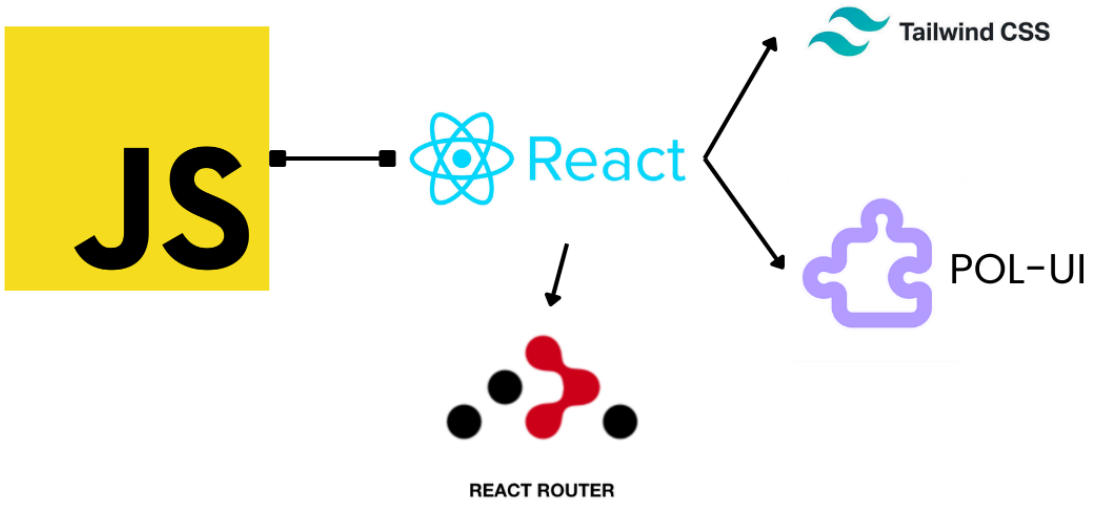

Fig. 7: Tecnologías *front-end* del proyecto

Fuente: Elaboración propia.

Para cada una de estas tecnologías el objetivo es poder responder: qué es, para qué sirve y cómo lo he usado en mi proyecto web.

El enfoque será deductivo, partiendo de lo general hacia lo particular, empezando con JavaScript y acabando con Pol-UI.

JavaScript ES6

¿Qué es?

JavaScript es un lenguaje de programación ampliamente utilizado para crear y controlar contenido dinámico en páginas web que desde 1995 está en todos los navegadores (Coppola, 2023). Permite agregar interactividad, como actualizaciones en tiempo real, animaciones, y validaciones de formularios. Es esencial para el desarrollo web moderno y se ejecuta en el navegador, lo que lo hace ideal para mejorar la experiencia del usuario en sitios y aplicaciones web (O'Grady, 2023). Desde hace años se ha convertido en uno de los lenguajes más utilizados (Loreto, 2024) gracias a su versatilidad incorporando soporte con *backend* gracias a **NodeJS** y la multitud de frameworks como el ya comentado React u otros como Angular o Vue.

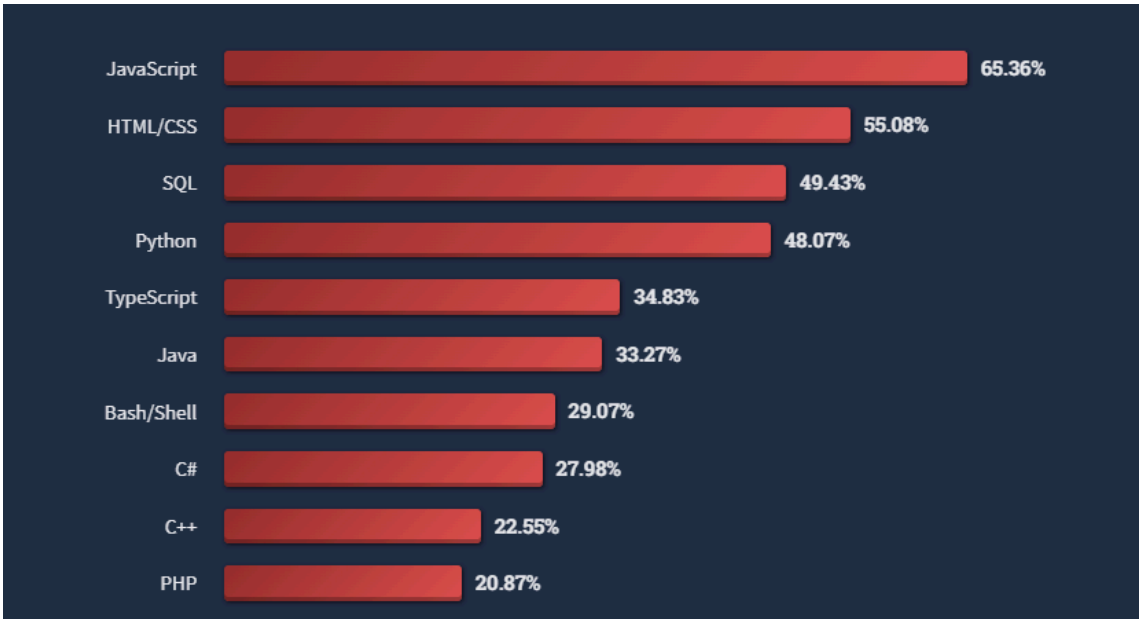

Fig. 8: Gráfico porcentaje (%) de uso de lenguajes de programación
Fuente: survey.stackoverflow.co (01/04/2024).

¿Para qué sirve JavaScript?

JavaScript es una herramienta fundamental en el desarrollo web, utilizada extensamente para añadir interactividad y mejorar la experiencia del usuario. Inicialmente concebido para el *frontend*, JavaScript ha evolucionado para ser también empleado en el *backend* (W3Techs, 2024), permitiendo a los desarrolladores realizar operaciones CRUD y manejar la lógica del servidor. Según *W3Techs*, más del 90% de los sitios web incorporan JavaScript, consolidándose como la tecnología líder en el desarrollo web.

¿Cómo se ha usado en el proyecto?

En mi proyecto web, he implementado JavaScript a través de React para construir la interfaz de usuario y administrar componentes utilizando archivos con extensión .jsx, que combinan JavaScript y HTML de manera eficiente. Además, he empleado JavaScript para gestionar la lógica del cliente y conectarme con la base de datos, utilizando las capacidades de Supabase para integrar datos de manera efectiva en la aplicación.

React JS

¿Qué es?

React JS es una biblioteca de JavaScript desarrollada por Facebook que se utiliza para construir interfaces de usuario. Es especialmente útil para crear aplicaciones web de una sola página (SPAs) mediante la construcción de componentes reutilizables (Deshpande, 2023). Esta infografía de EDteam muestra a la perfección el funcionamiento básico de React y su sintaxis:

Fig. 9: Infografía sintaxis de React

Fuente: ed.team (02/04/2024).

¿Para qué sirve?

React sirve para desarrollar interfaces de usuario dinámicas y eficientes. Facilita la creación de componentes interactivos que actualizan y renderizan eficientemente datos cambiantes en la pantalla (Kinsta, 2022). Utiliza un Virtual DOM para optimizar las actualizaciones de la interfaz, lo que mejora significativamente el rendimiento de las aplicaciones web. React ofrece una gran cantidad de funciones propias, lo que se denominan *hooks*. Estas son "funciones de Javascript que permiten crear/acceder al estado y a los ciclos de vida de React" (Huet, 2023). React proporciona tres *hooks* básicos que cubren las necesidades habituales del ciclo de vida de los componentes.

1. Use State

   El *hook* "useState" permite manejar el estado en componentes funcionales. Devuelve un valor de estado y una función para actualizarlo (Ayebola, 2023).

1. Use Effect

   El *hook* "useEffect" permite agregar efectos secundarios, como la manipulación del DOM, solicitudes de datos, y otras operaciones después de que el componente se ha renderizado. Se ejecutará cada vez que se cargue el componente o cambie alguno de los parámetros que este recibe (Ayebola, 2023).

2. Use Context

   El *hook* "useContext" permite acceder a valores de contexto en componentes funcionales, facilitando el paso de datos entre componentes sin necesidad de pasar "props" manualmente (Ayebola, 2023). Ayuda mucho para la renderización dinámica de componentes, primero se define un archivo que hará de "baúl" donde guardar las variables y sus estados.

¿Cómo se ha usado en el proyecto?

En el desarrollo de la plataforma de cursos, he utilizado la componentización de React para estructurar visualmente la interfaz y los *hooks* para manejar dinámicamente la lógica y la interacción con la base de datos.

Desde esta perspectiva, se destacan claramente el funcionamiento y las ventajas que React ofrece en un desarrollo web escalable y reutilizable. Este enfoque permite reducir considerablemente el tiempo necesario para construir aplicaciones.

No obstante, aunque React permite crear pantallas de manera rápida, es fundamental contar con un método para conectar y permitir la navegación entre estas pantallas. Para abordar esta necesidad, se consideraron dos librerías externas:: Next.js y React Router DOM. En este proyecto, se decidió emplear React Router DOM por los siguientes motivos: la curva de aprendizaje era menor respecto otras como Next Js, los requisitos de la aplicación no requerían de almacenamiento en caché, la principal ventaja de Next JS, y por último, su amplia documentación y compatibilidad en lo que a React se refiere.

React Router DOM

¿Qué es?

"React Router DOM es una librería que permite agregar enrutamiento a una aplicación web de React" (Valera, 2023).

¿Para qué sirve?

Para explicarlo de una manera sencilla: imaginemos que estamos organizando una casa con varias habitaciones. Cada habitación es una página diferente. El enrutamiento es como el plano de la casa que indica cómo moverse de una habitación a otra. Cuando alguien entra por la puerta principal (la página de inicio), necesita saber cómo llegar al dormitorio, la cocina o el baño (otras páginas). El enrutamiento se asegura de que haya un camino claro desde la entrada hasta cualquier habitación que quieran visitar.

El sentido de esta librería no es más que el de conectar los componentes, su diseño y su lógica entre sí para que el usuario pueda navegar entre estos de una manera fluida y coherente. Normalmente se ubica en el componente "padre" de la aplicación (App.jsx) y usa un sistema de rutas con herencia. A continuación se muestra un ejemplo de *router* sencillo explicado por Duque Valera en su artículo:

```jsx
import React from 'react';
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import Navbar from './Navbar';
import HomePage from './HomePage';
import AboutPage from './AboutPage';
import ContactPage from './ContactPage';
import NotFoundPage from './NotFoundPage';

function App() {
  return (
    <BrowserRouter>
      <Navbar />
      <Routes>
        <Route path="/" element={<HomePage />} />
        <Route path="/about" element={<AboutPage />} />
        <Route path="/contact" element={<ContactPage />} />
        <Route path="*" element={<NotFoundPage />} />
      </Routes>
    </BrowserRouter>
  );
}

export default App;
```

Fig. 10: Fragmento de código ejemplo de *router*
Fuente: linkedin.com (11/04/2024).

Los *imports* son componentes maquetados y que existen en el entorno de nuestra aplicación. A continuación se indican las rutas, siempre con la misma estructura, dentro del componente "Routes" indicamos cada una de las rutas que queremos que haya, con el componente "hijo" "Route" decidimos qué componente se mostrará (*element*) dependiendo de lo que se escriba en la url (*path*).

En el ejemplo, si el usuario escribe: mi-aplicacion/contact el contenido que se mostrará será el del componente "ContactPage".

Este es un *router* muy sencillo de ejemplo. En el *sprint* 2: Maquetación del *frontend* se muestra en detalle el *router* interno creado para la plataforma de cursos interactivos.

¿Cómo se ha usado en el proyecto?

Partiendo de la anterior analogía con las habitaciones de la casa, mi aplicación equivale a una finca (de nombre "**App**") con una entrada o jardín común y dos casas: una casa principal y una para los invitados. El *router* se ha diseñado en función del rol, siendo los usuarios administradores los que tienen llave de ambas casas y pueden entrar a todas las habitaciones y los invitados como los usuarios, que solo pueden entrar a su casa que es más pequeña y tiene muchas menos habitaciones.

Finalmente mi aplicación está envuelta por una capa de seguridad detallada en el sprint 6. Roles, sesiones y seguridad. El objetivo de esta capa de seguridad es que ni los invitados, ni personas ajenas a la casa puedan entrar en la casa principal sin permiso, rodeándola con un una baya llamada ***Private Route*** que solo se puede superar si el código de acceso es de administrador.

Tailwind CSS

¿Qué es?

Este *framework* de *frontend* está cogiendo mucha fuerza y popularidad entre los desarrolladores por su alta flexibilidad, buena documentación y soporte con lo más nuevo de JavaScript.. Tailwind no deja de ser una librería con clases de CSS agrupadas qué permite desde el propio HTML de una aplicación dar formato y estilos a los componentes sin necesitar ninguna línea de código CSS (Vergara, 2023).

¿Para qué sirve?

Su uso se puede aplicar en muchas facetas del desarrollo de un componente web. Desde crear la estructura o *layout* de la página mediante el sistema de rejillas *grid* o *flex* hasta estilos como una sombra, un color de un botón o el tamaño de un texto. En la siguiente infografía diseñada por Alfredo Mendoza publicada en Twitter se detallan estas características de Tailwind, donde se destacan: la construcción con clases, la responsividad, las pseudoclases, y las clases customizables que estas nos aportan.

Fig. 11: Infografía características de Tailwind

Fuente: twitter.com/AlfredoMenCap (11/04/2024).

¿Cómo se ha usado en el proyecto?

Son muchas las funciones que Tailwind CSS aporta a una aplicación como la mía, he utilizado Tailwind CSS en mi aplicación de cursos de React para diseñar y estilizar componentes de manera eficiente.

Esta librería me ha permitido aplicar estilos directamente en el código, utilizando clases predefinidas que facilitan el desarrollo y mantienen un diseño coherente en toda la aplicación. Acelerando el proceso de desarrollo y asegurando que la interfaz de usuario fuese atractiva y responsiva.

Pol UI

¿Qué es?

Según la propia documentación del sitio oficial, Pol-UI se define como "una bonita y funcional biblioteca de UI para React y Next.js" (Gubau, 2024). El lenguaje de base es TypeScript.

Cabe destacar que el creador de esta librería es Pol Gubau Amores un compañero de universidad y desarrollador *frontend* cuyo objetivo era construir una serie de componentes atractivos, funcionales y totalmente reutilizables y adaptativos para cualquier tipo de plataforma web.

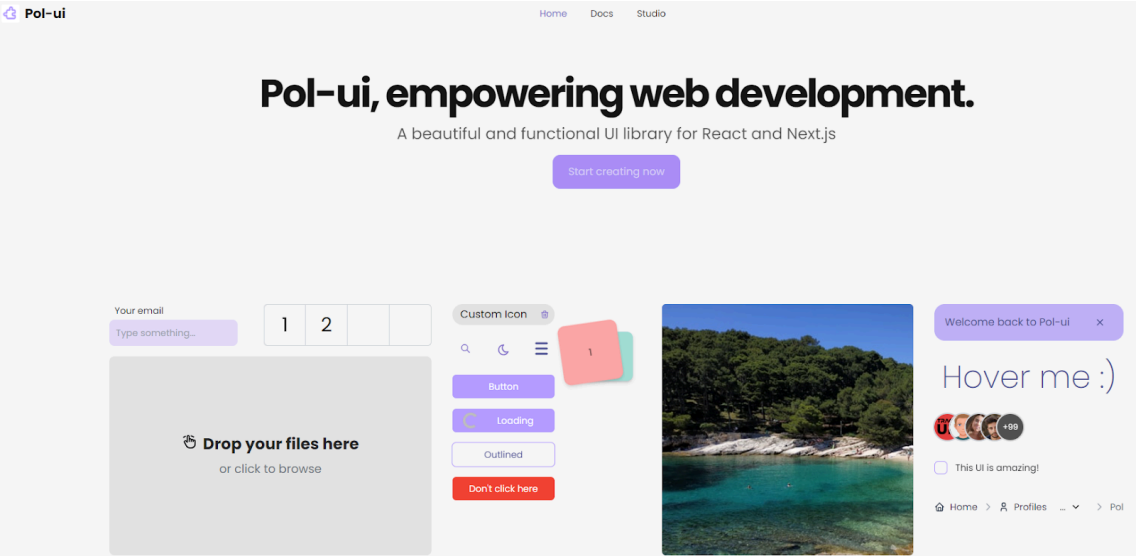

Fig. 12: Página principal web oficial de Pol-UI
Fuente: ui.polgubau.com (18/04/2024).

¿Para qué sirve?

A pesar de la notable eficiencia de Tailwind en comparación con el enfoque convencional de CSS, el proceso de maquetación manual y la programación individual de cada componente resultan extremadamente exigentes. Gracias a bibliotecas de componentes como Pol-UI, podemos mitigar este considerable costo de tiempo, permitiéndonos obtener una variedad de elementos desde simples botones hasta menús completos.

Estas bibliotecas no solo ofrecen funcionalidades interactivas predefinidas, sino que también permiten una personalización estilística que asegura la integración perfecta de los elementos con el proyecto en cuestión.

¿Cómo se ha usado en el proyecto?

Hay una gran variedad de bibliotecas que ya cuentan con componentes montados, las más conocidas son Material UI (desarrollada por Google) y Next UI (Devink, 2024), pero Pol-UI me aportaba una serie de ventajas que las otras no:

- **Diseño atractivo**: El diseño visual de los componentes de Pol-UI eran mucho más estéticos y encajaban mejor con los prototipos de la plataforma en comparación con Material UI.
- **Más cantidad de componentes**: Pol-UI era la opción con más componentes desarrollados para usar.
- **Compatibilidad con HTML nativo**: Si bien a nivel estético Next UI era una buena competencia de Pol-UI, acabé rechazando su uso por la poca compatibilidad y los problemas que me daba al usar propiedades propias en lugar de adaptarse al HTML nativo.
- **Diseño a medida:** Por último, al ser un proyecto transversal con Pol, un factor decisivo fué la colaboración ciertamente simbiótica con Pol, el podía usar mi proyecto para comprobar el correcto funcionamiento de sus componentes, testearlos, mejorar aspectos tanto de diseño como de funcionalidad y ver cómo se plasman su biblioteca en un proyecto web real. Esto me daba a mí la oportunidad de poder pedir los componentes necesarios para mi aplicación, ajustando los componentes en base a los requisitos que la plataforma requería.

Entorno de desarrollo, gestor de paquetes y control de versiones

La programación del código web necesita sostenerse sobre 3 ejes; un entorno de desarrollo en servidor, un gestor de paquetes para incorporar librerías externas de manera sencilla y segura, y por último un *software* para controlar las versiones, los avances y los errores de manera ordenada y eficaz. Hay varias tecnologías en el mercado que hacen esto, pero las más conocidas y las más óptimas para mi *stack* son las siguientes.

1. Vite

   Como entorno de desarrollo.Según la propia documentación oficial de *Vite*, es una herramienta que hace que desarrollar sitios web modernos sea más rápido y fácil. Tiene dos partes importantes (Hernandez, 2022):

   - **Servidor de desarrollo**: Cuando estás trabajando en tu proyecto, este servidor te permite ver los cambios que haces casi instantáneamente. Por ejemplo, si modificas algo en tu código, puedes ver cómo afecta a tu página web sin tener que "recargar" manualmente.
   - **Comando de compilación**: Cuando terminas tu proyecto y estás listo para publicarlo en internet, Vite lo prepara para que funcione de la mejor manera posible. Optimiza todo el código para que los sitios web sean rápidos y eficientes cuando los visiten las personas.

   En resumen, Vite hace que sea más rápido desarrollar y lanzar sitios web modernos, mejorando la experiencia tanto para los desarrolladores como para los usuarios finales.

2. NPM (Node Package Manager)

   Como gestor de paquetes. Según la página oficial de NPM, podría definirse como una tienda de aplicaciones para programadores que usan Node.js. Te ayuda a instalar, compartir y manejar las herramientas que necesitas para tu proyecto. Con NPM, puedes buscar y descargar muchas herramientas útiles que otros programadores han creado y compartido de forma gratuita. Esto facilita agregar nuevas funciones y herramientas a tus propios proyectos sin tener que crear todo desde cero. Las operaciones comunes con NPM incluyen:

   - **npm install:** Instalar dependencias del archivo package.json.
   - **npm install <paquete>**: Instalar un paquete específico y añadirlo como dependencia.
   - **npm run <script>**: Ejecutar scripts definidos en package.json, como scripts de construcción, pruebas y desarrollo.

3. Git

   Para el control de versiones. Mijacobs (2023) habla de git como un sistema de control de versiones distribuido que permite a los desarrolladores rastrear cambios en su código fuente a lo largo del tiempo. Git facilita la colaboración, permitiendo que múltiples desarrolladores trabajen en paralelo, integren sus cambios y gestionen versiones del código de manera eficiente.

   Las operaciones básicas con Git incluyen:

   - **git init:** Inicializar un nuevo repositorio Git.
   - **git clone <url>**: Clonar un repositorio existente.
   - **git add <archivo>**: Añadir cambios a la zona de preparación (staging area).
   - **git commit -m "mensaje"**: Confirmar los cambios en el repositorio.
   - **git push**: Enviar los cambios confirmados a un repositorio remoto.
   - **git pull**: Actualizar el repositorio con los cambios del repositorio remoto.

   Para gestionar las versiones y repositorios de una manera más clara, se creó la interfaz web **Github**, que permite realizar las mismas operaciones pero desde un sitio web más usable y accesible para el desarrollador.

5.3 *Backlog* de producto (historias)

GitHub, además de ofrecer soporte visual para el control de versiones con Git, cuenta con un servicio gratuito llamado GitHub Projects. Esta herramienta permite planificar trabajos tecnológicos como el presente, basándose en la metodología Scrumban mencionada anteriormente. Con GitHub Projects, puedes definir las tareas de tu proyecto y asociarlas a un *sprint*, siguiendo su evolución mediante un tablero Kanban. Esto aporta un gran valor, especialmente, porque al estar integrada con Git, cada cambio en el código puede asociarse a estas historias, controlando así de una manera meticulosa el avance del proyecto.

A continuación se muestra este *backlog* de producto (tareas) pensado para la plataforma de cursos interactivos de Rispot Consulting.

Sprint	Historias
***SPRINT* 0** Documentación y planificación	1. Definición del proyecto 2. Requisitos y casos de uso generales 3. Versiones y planificación 4. Casos de uso específicos (versión 1) 6. Estilos y requisitos del cliente de diseño
***SPRINT* 1** Diseño interfaz	1. Bocetos pantallas panel admin 2. Bocetos pantallas panel usuario 3. Tests usabilidad (I) 4. Crear *wireframes* de las interfaces (admin y usuario) 5. Tests usabilidad (II) 6. Incorporar *feedback* de tests de usabilidad en *mockup*
***SPRINT* 2** Maquetación *front-end*	1. Inicio proyecto con Vite 2. Maquetación de componentes con Tailwind y React. - Pol UI + React Icons 3. Crear componentes estáticos. 4. Responsividad
***SPRINT* 3** Tablas y BBDD *backend*	1. Diagrama E-R 2. Creación de tablas y relaciones en Supabase 3. Definir consultas y funciones multitabla previsibles 4. Crear usuario de prueba

***SPRINT* 4** Funcionalidades panel administrador	1. Conectar Supabase con proyecto Vite y componentes maquetados. 2. Pruebas de integración. 3. CRUD para cursos. 4. CRUD para módulos. 5. Vista previa (nivel básico) de módulo y curso. 6. CRUD para detalle módulo. - Video, acciona, examen, numeral, quiz, material. - Ordenar elementos módulos. - Buckets Supabase. 7. CRUD Preguntas diarias y consejos diarios
***SPRINT* 5** Funcionalidades panel usuario	1. Registro usuario. 2. Login usuario. 3. Roles de usuario. 4. Pantallas iniciales usuario. 5. Asociar usuarios a cursos. 6. Pantalla del contenido del curso (contenido). 7. Lógica para visualizar contenido estático. 8. Lógica para hacer contenido interactivo (actividades y examen). - Preguntas diarias y consejos diarios. 9. Relacionar usuario-curso (evolución). 10. Poder ver y editar información del perfil.

SPRINT 6 Roles, accesos y seguridad	1. Gestionar usuarios de un curso. Aceptar, expulsar... 2. Control de acceso con rol (rutas). 3. Control de tablas y privacidad en Supabase.
SPRINT 7 Pruebas y análisis de usabilidad II	1. Repaso de todas las funcionalidades. 2. Ajustar fallos de programación. 3. Pruebas de rendimiento. 4. Ajustar fallos de rendimiento. 5. Test de usabilidad final.
***SPRINT* 8** Despliegue a producción	1. Configuración del entorno de producción. 2. Despliegue a producción (Vercel. 3. Configuración de CI/CD para despliegue continuo.
SPRINT 9 Validación y pruebas	1. Pruebas en producción. 2. Corrección de errores y *bugs* detectados en producción.

Fig. 13: Tabla s*prints* y *backlog* de producto

Fuente: Elaboración propia.

5.4 Gestión económica y recursos

El desarrollo de la plataforma de cursos, se realizó sin una motivación económica inicial. Sin embargo, para comprender las posibles implicaciones económicas, es crucial considerar los costos de desarrollo, mantenimiento y la propuesta de valor para las instituciones educativas.

Costes del proyecto

Aunque el desarrollo de la plataforma se realizó de manera gratuita he invertido tiempo y esfuerzo para su creación. La plataforma cuenta con más de 700 horas de desarrollo para el

código principal y entre 20 y 40 horas para la planificación y diseño. En total, esto suma 760 horas de trabajo de codificación.
Suponiendo que utilizamos el salario promedio de un desarrollador de software en España, que según los datos anteriores es de aproximadamente 2.375€ al mes, equivalente a 14,62€ por hora.

Dado que la plataforma ha requerido 760 horas de trabajo, podemos calcular el costo total del desarrollo de la siguiente manera: 760 horas x 14,62€/hora = **11.115,20€**
Este sería el costo estimado del desarrollo de la plataforma si consideramos el salario promedio de un desarrollador en España. Es importante recordar que este cálculo es una estimación y puede variar dependiendo de factores como la complejidad del desarrollo, la experiencia del desarrollador y otros costos asociados al proyecto.
Respecto a los recursos humanos en este caso la viabilidad es muy clara ya que solo consta de un desarrollador, desde el punto de vista de recursos materiales se puede aumentar el coste si tenemos en cuenta un equipo informático óptimo con un mínimo aproximado de 800 a 1000 euros.

5.5 Gestión del riesgo

Una vez identificadas todas las tareas necesarias para completar el proyecto de manera exitosa, es crucial emplear herramientas efectivas como el diagrama de Gantt para planificar y asignar adecuadamente el tiempo requerido para cada una. Este enfoque visual asegura que los plazos no se extiendan innecesariamente, lo cual es fundamental para evitar pérdidas económicas en el ámbito empresarial, aunque en este proyecto específico no se maneje riesgo financiero directo. Establecer plazos claros para cada subobjetivo es igualmente esencial.

El diagrama de Gantt proporciona un marco temporal ideal desde las etapas iniciales del proyecto, si bien es necesario reconocer que factores como la limitación de tiempo, falta de conocimientos específicos y otros aspectos externos pueden afectar la planificación inicial y extender los períodos previstos (Stsepanets , 2024).

En este proyecto particular, el diagrama de Gantt no se ha detallado para cada tarea individual debido a su duración relativa y los riesgos identificados. En su lugar, se ha adoptado un enfoque más general, estableciendo plazos para cada *sprint* (agrupación de tareas concretas). A continuación se muestra un diagrama realizado con *excel* para el proyecto. La planificación de las tareas se dividió por semanas, siendo el plazo más corto de una semana y el plazo más largo de ocho semanas para el desarrollo funcional del panel de usuario y su conexión con base de datos.

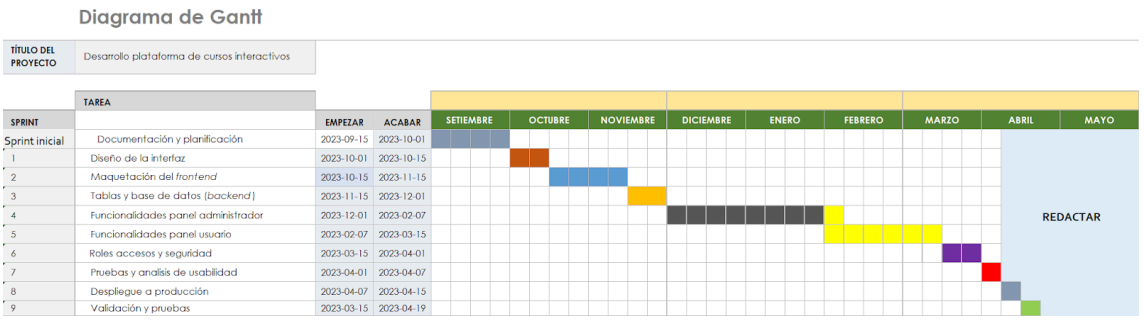

Fig. 14: Diagrama de Gantt del proyecto

Fuente: Elaboración propia.

6. Desarrollo y ejecución del proyecto

La intención en este punto del trabajo y con todas las tecnologías y herramientas contextualizadas, es explicar el proceso desde un enfoque narrativo. Comenzando desde el planteamiento del proyecto el 28 de agosto de 2023 hasta el día de hoy.

Para mantener un hilo discursivo coherente, seguiré los *sprints* construidos hasta ahora, destacando los desarrollos más importantes y los problemas más notables encontrados en cada uno de estos nueve pasos.

Sprint 0: Definición del proyecto y requisitos básicos y planificación

La plataforma surge de la necesidad comercial de la empresa Rispot Consulting, propiedad de Pau Cardús Fillat, compañero de universidad que me encargó el presente desarrollo.

Las bases de todo el proyecto parten de sus necesidades comerciales, sus intereses y su conocimiento del sector. El primer paso era plasmar todo esto, conocer los requisitos del cliente. Se realizaron diversas reuniones donde se plantearon bocetos para definir las partes conceptuales del desarrollo.

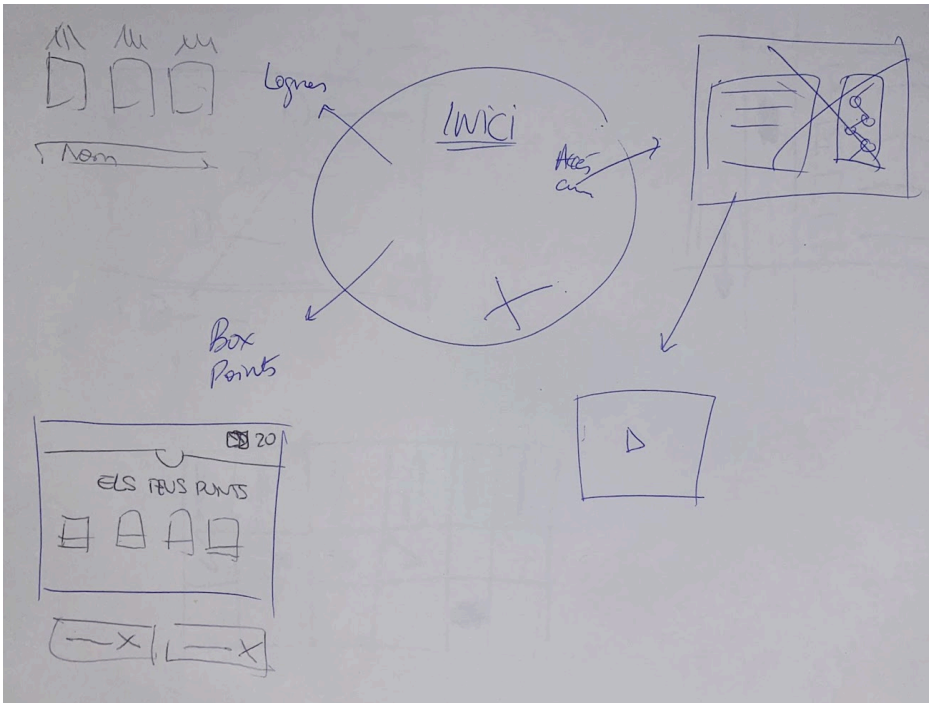

Fig. 15: Bocetos conceptuales de la plataforma
Fuente: Elaboración propia.

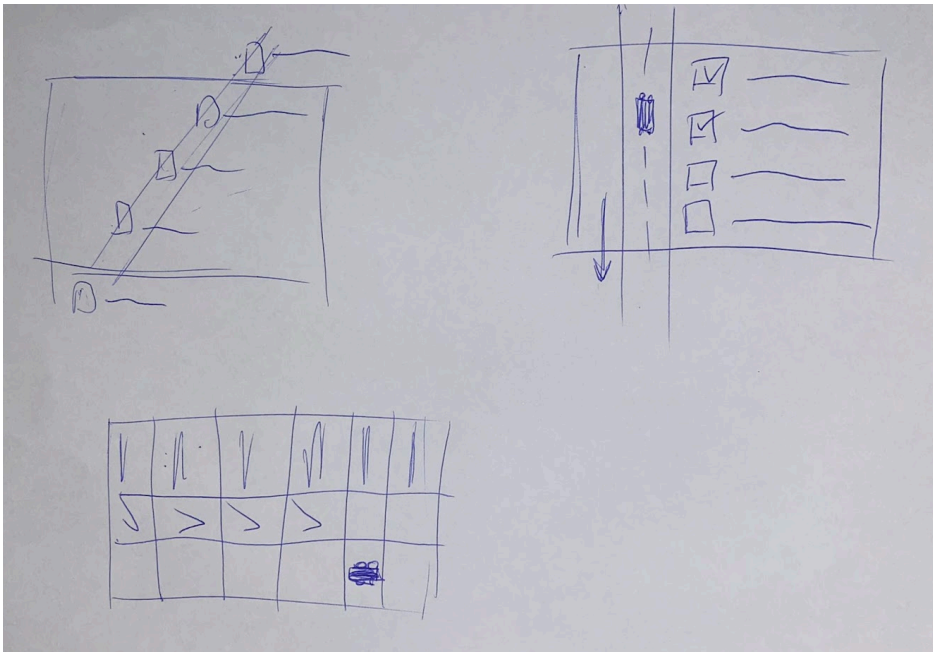

Fig. 16: Bocetos diseño de la plataforma
Fuente: Elaboración propia.

Después de varios bocetos y varias reuniones conseguimos establecer una idea de lo que queríamos tener en el proyecto. El siguiente paso fué acotar las funcionalidades planteando versiones para adaptarse al tiempo del que se disponía.

Este proyecto responde a la primera versión, con un panel de administración funcional y acceso a usuarios que puedan consumir el curso. Conceptos planteados como la gamificación con ránkings, premios y rachas se reservaron para el desarrollo de una segunda versión de la plataforma.

Con las versiones ya estudiadas se trasladaron los casos de uso en papel, la intención era ver todo lo que podía hacer un usuario al entrar en la plataforma, desde el punto de vista del administrador y del usuario base del curso.

A continuación se muestra el diagrama UML para representar estos casos de uso que se han detallado en la sección: 4.2.1.1. Casos de uso.

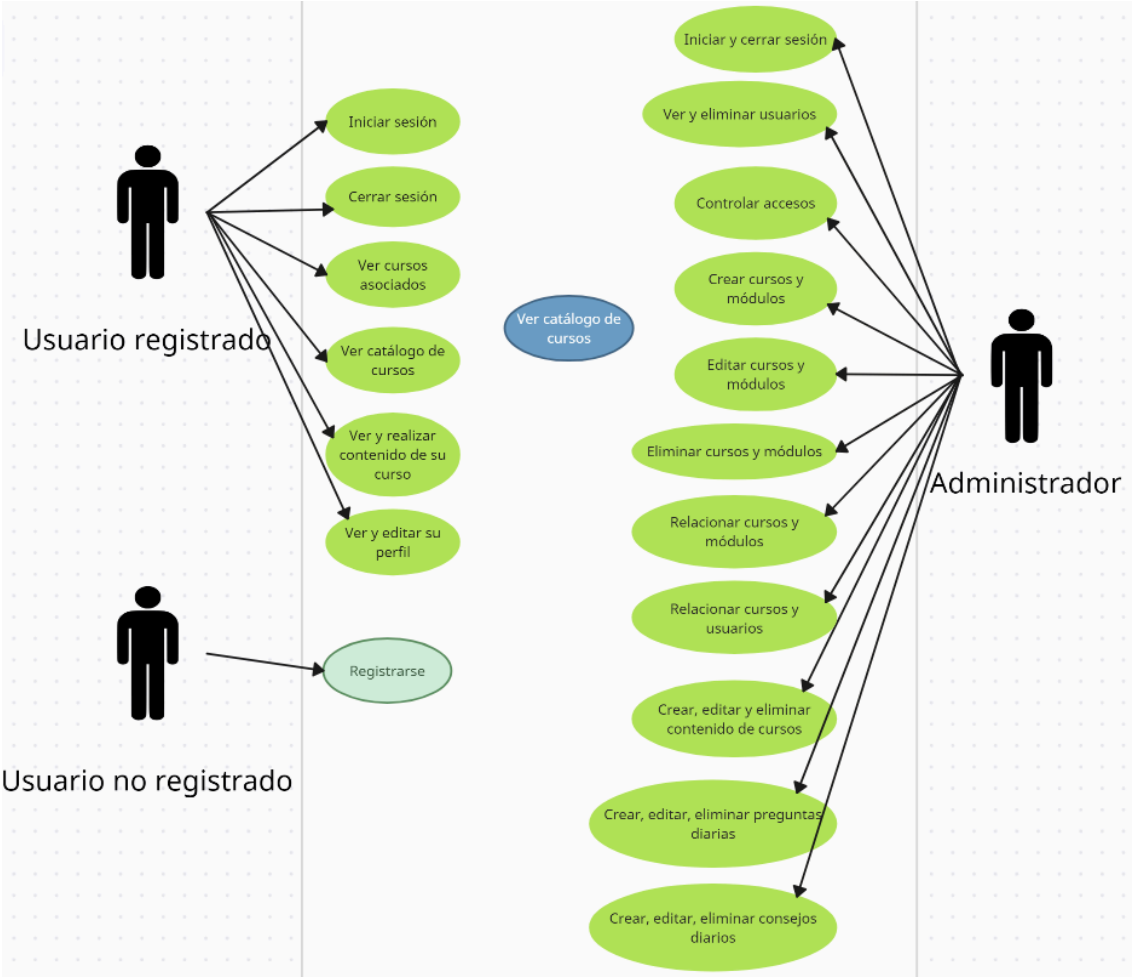

Fig. 17: Diagrama UML. Casos de uso.
Fuente: Elaboración propia

El siguiente paso, como desarrollador individual, consistió en planificar el proyecto desde una perspectiva logística y temporal, utilizando la metodología ágil Scrumban y GitHub Projects, basándome en toda la información y requisitos proporcionados por el cliente

Sprint 1: Diseño de interfaz

El cliente tenía una visión muy clara de cómo quería el diseño del panel de usuario pero no tenía nada definido para el panel de administración , donde me dió total libertad para diseñar adaptándome a los imprevistos técnicos. En este caso omití el paso de los bocetos de bajo nivel y pasé directamente a definir una plataforma con cierto detalle para poder realizar tests de usabilidad coherentes.

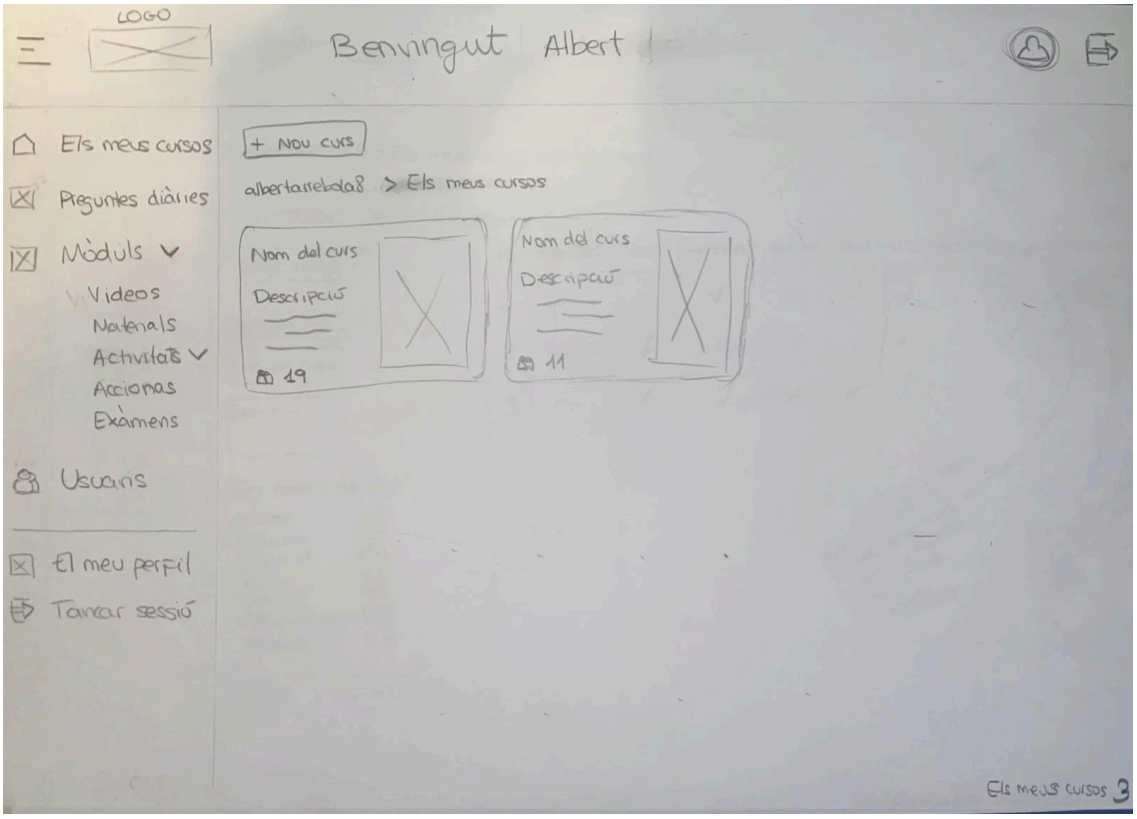

Fig. 18: Boceto pantalla principal administrador

Fuente: Elaboración propia.

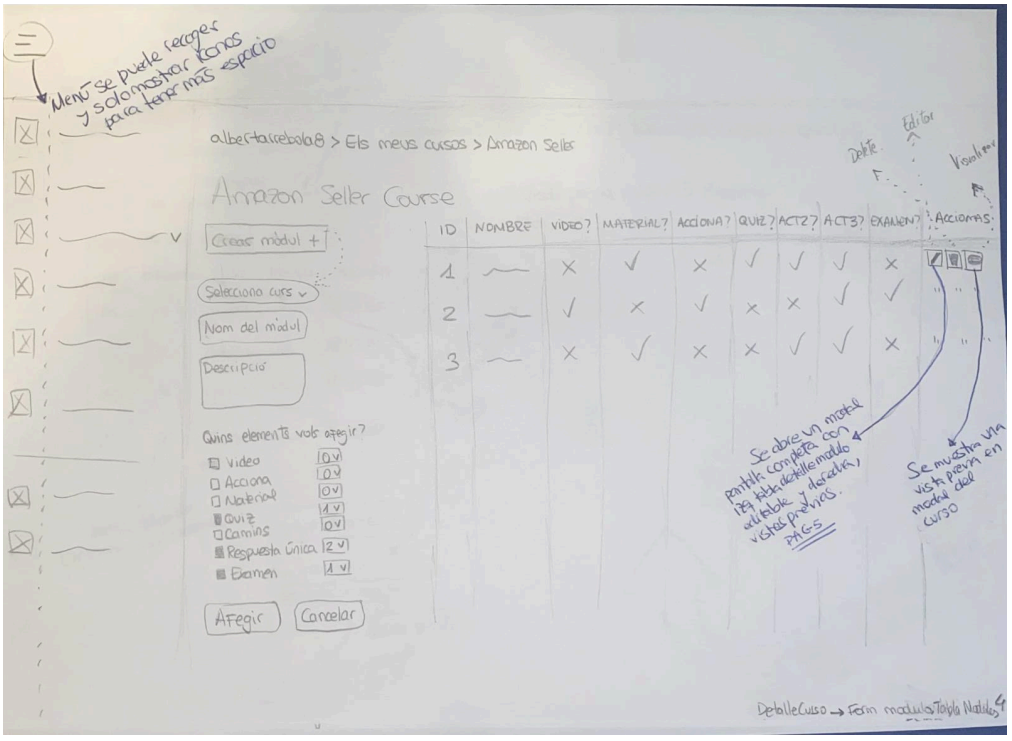

Fig. 19: Boceto pantalla de curso

Fuente: Elaboración propia.

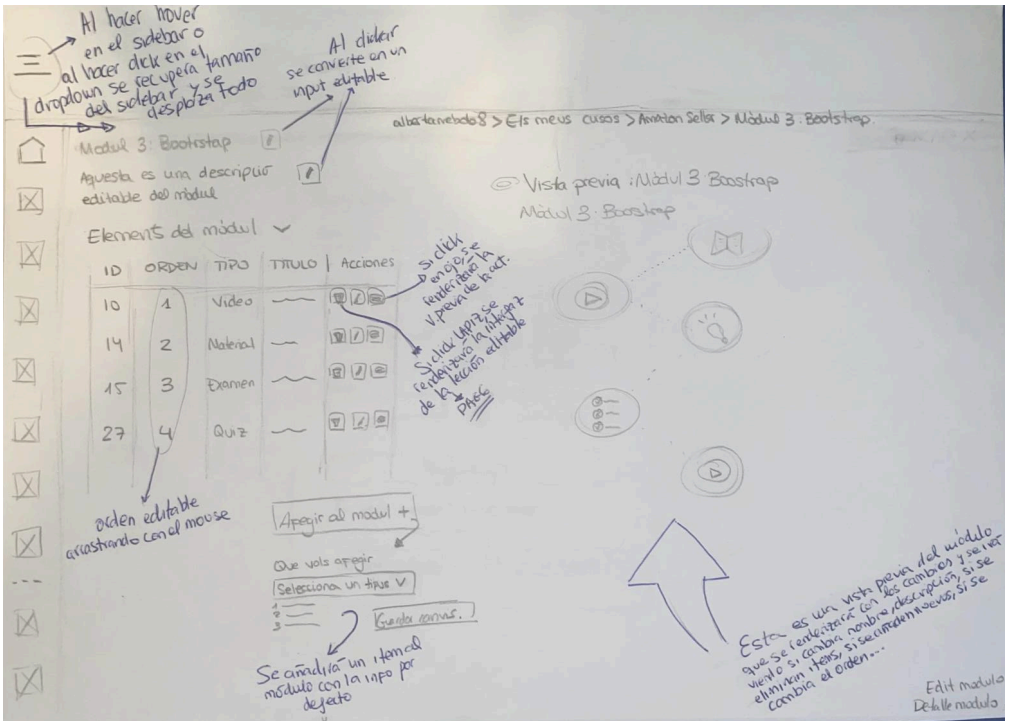

Fig. 20: Boceto pantalla de módulo

Fuente: Elaboración propia.

También se plantearon los bocetos para editar los contenidos dentro del módulo. Por extensión, se muestra solo el prototipo para gestionar el contenido de un examen:

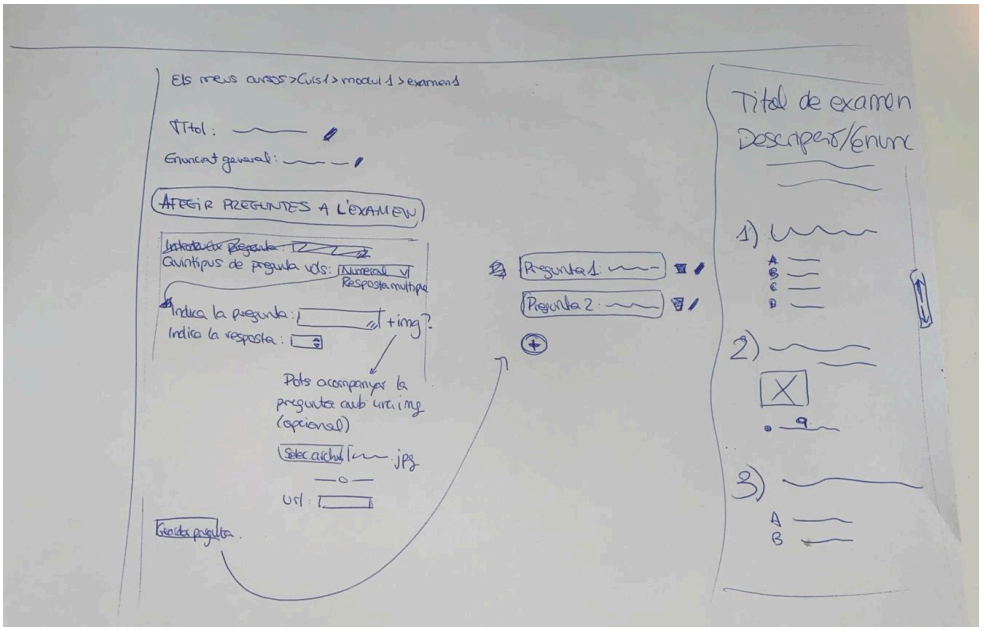

Fig. 21: Boceto pantalla de examen (administrador)

Fuente: Elaboración propia

Para el panel de administrador, al ser un diseño totalmente propio, era necesario realizar un primer test de usabilidad de primer nivel sobre los bocetos para evitar problemas estructurales de diseño y comprobar que se podían realizar todos los casos de uso planteados para ese rol.

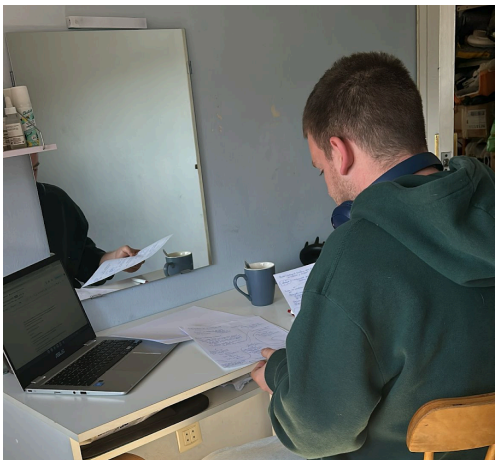

Fig. 22: Fotografía realización test de usabilidad
Fuente: Elaboración propia.

Estos primeros tests fueron muy importantes porque me obligaron a cambiar el planteamiento que había hecho sobre la página de creación de módulos y sobre todo para las de generar exámenes y actividades. El hecho de hacerlos en este *sprint* y no hacia el final, me hizo ganar mucho tiempo ya que aún no había empezado a maquetar ni programar nada.

Una vez claros los resultados, era el momento de maquetar la parte visual de la aplicación, partiendo de los mismos bocetos del test o del *mockup* recibido por parte del cliente.

Fig. 23: Prototipo del cliente para pantalla del curso
Fuente: Elaboración del cliente (08/05/2024).

Este prototipo de alto nivel ya contenía estructura, colores, formas y estilos con cierto nivel de detalle. Con estos diseños ya establecidos, era el momento de comenzar a maquetar la plataforma.

Sprint 2: Maquetación del *frontend*

El enfoque consistía en replicar los diseños utilizando contenido estático, dado que aún no se contaba con una base de datos implementada para consumir información dinámica de usuarios o cursos. Por esta razón, los textos y gráficos fueron creados ficticiamente con el único propósito de visualizar cómo se vería la interfaz en pleno funcionamiento.

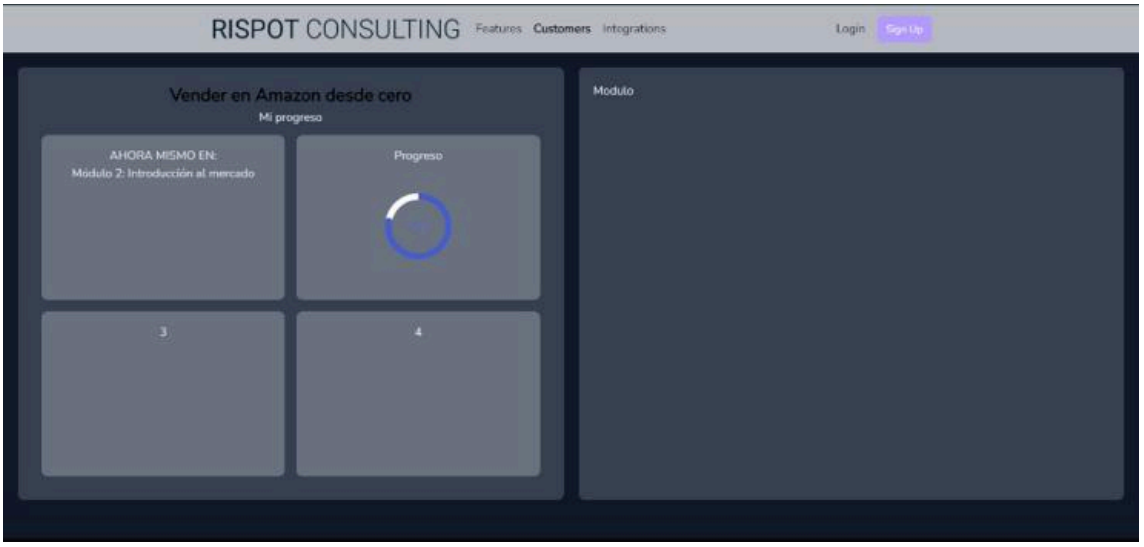

Fig. 24: Captura de pantalla durante el proceso de maquetación
Fuente: Elaboración propia.

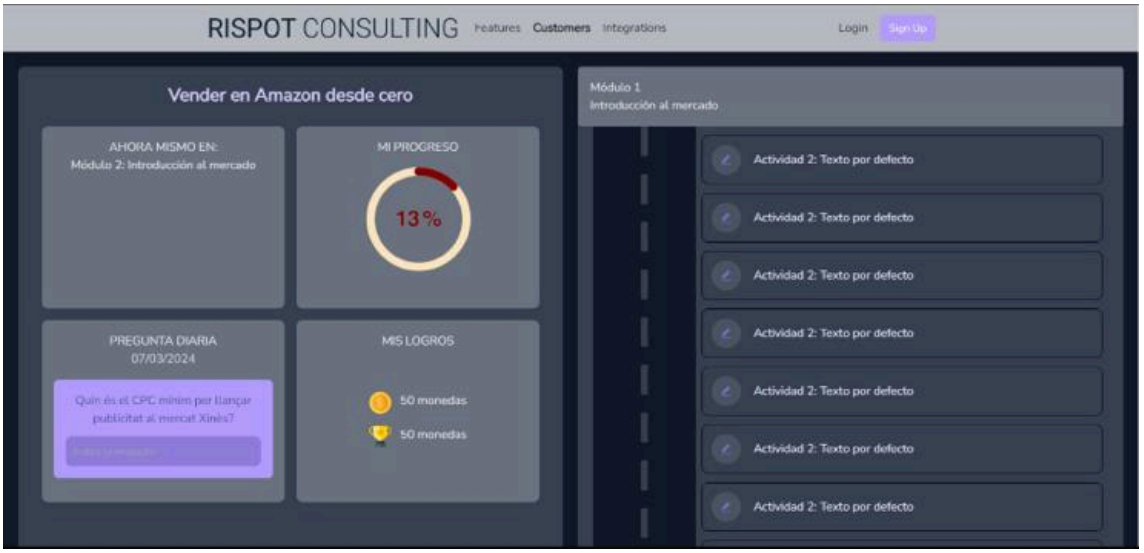

Fig. 25: Captura de pantalla (2) durante el proceso de maquetación
Fuente: Elaboración propia.

Maquetar todas las pantallas una por una utilizando solo HTML y CSS hubiera sido impracticable. Por lo tanto, opté por utilizar las bibliotecas de React, Tailwind CSS, React Router DOM y Pol-UI para desarrollar esta parte del proyecto, aprovechando sus ventajas para ser lo más eficiente posible. A continuación se detalla la función que ha cumplido cada una de estas tecnologías en el proyecto:

El uso de React en la aplicación

La función principal de React que utilicé fue la componentización, la cual me permitió maquetar los componentes globales y reutilizarlos en múltiples pantallas. Este enfoque estructural se intenta plasmar en el siguiente esquema de "cajas" diseñado para destacar los componentes desarrollados para la plataforma de cursos.

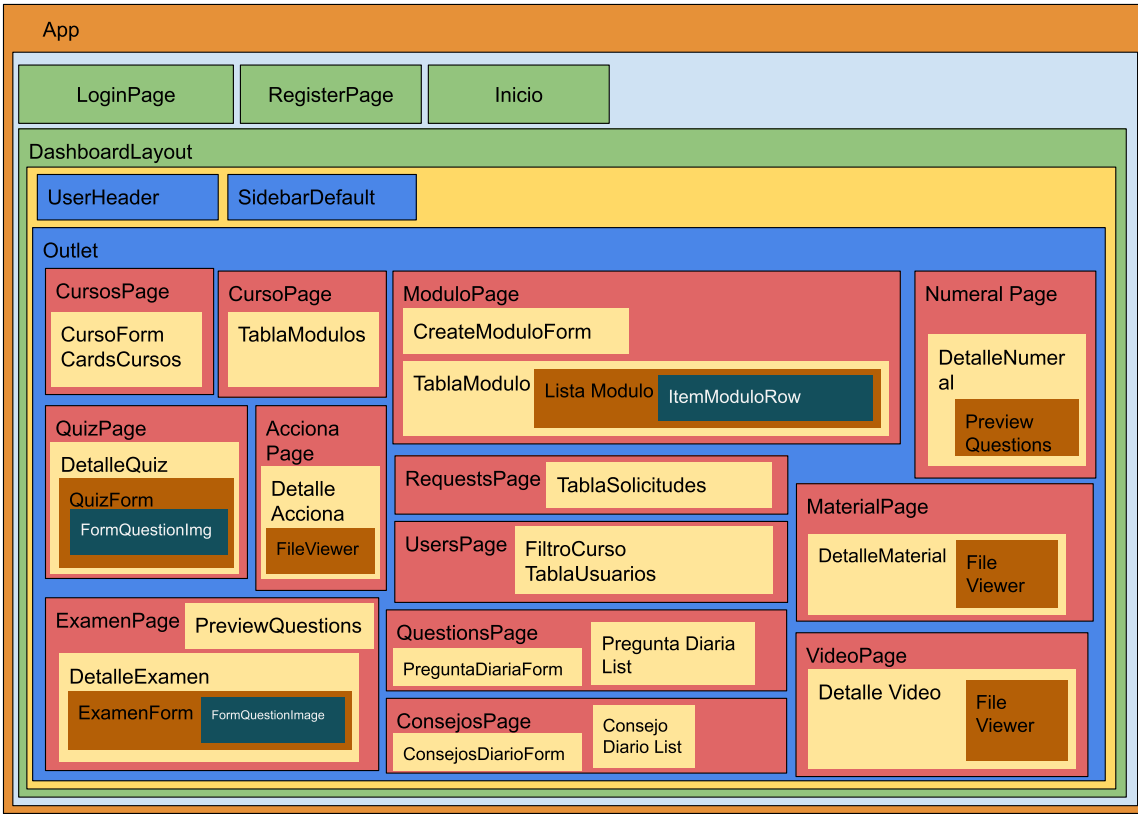

Fig. 26: Esquema de componentes para administrador

Fuente: Elaboración propia.

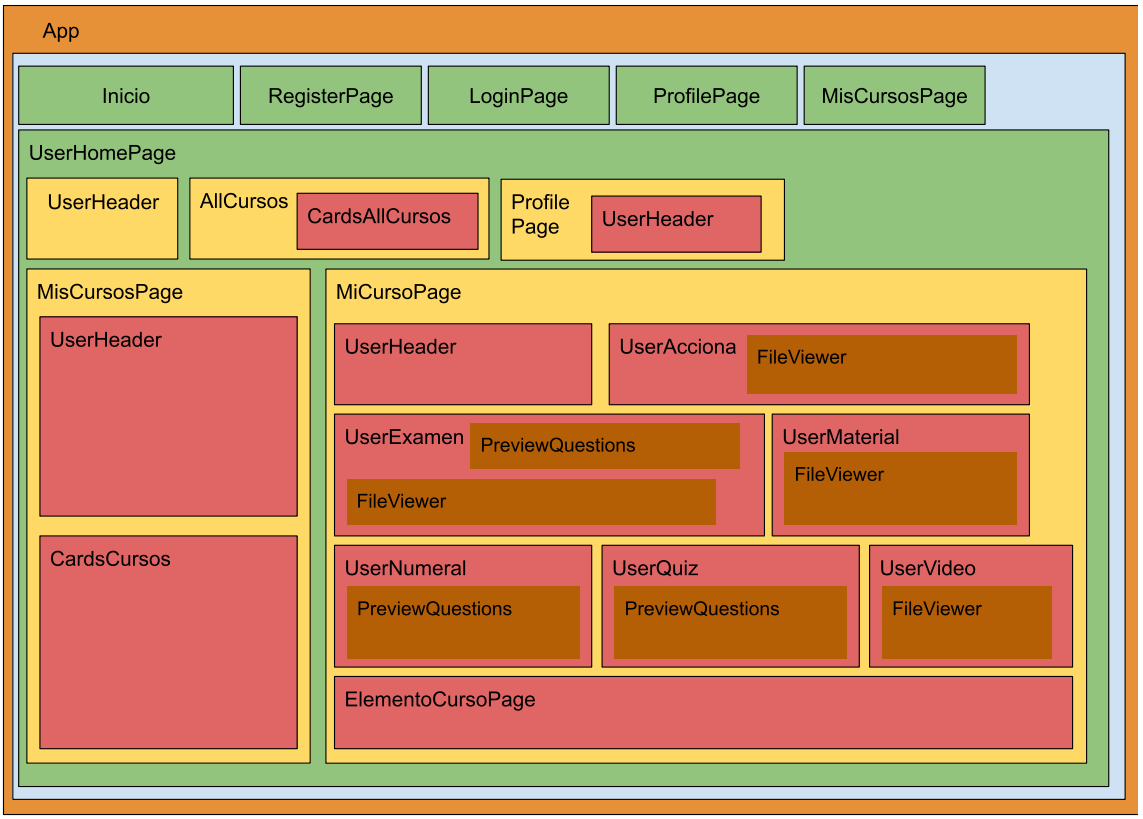

Fig. 27: Esquema de componentes para usuario

Fuente: Elaboración propia.

Una vez establecida la estructura de componentización que he desarrollado para organizar la plataforma web, es el momento de observar cómo esta teoría se materializa en la práctica. A continuación se presenta un ejemplo de los primeros niveles de componentes en la vista del curso, diseñados para un rol administrador.

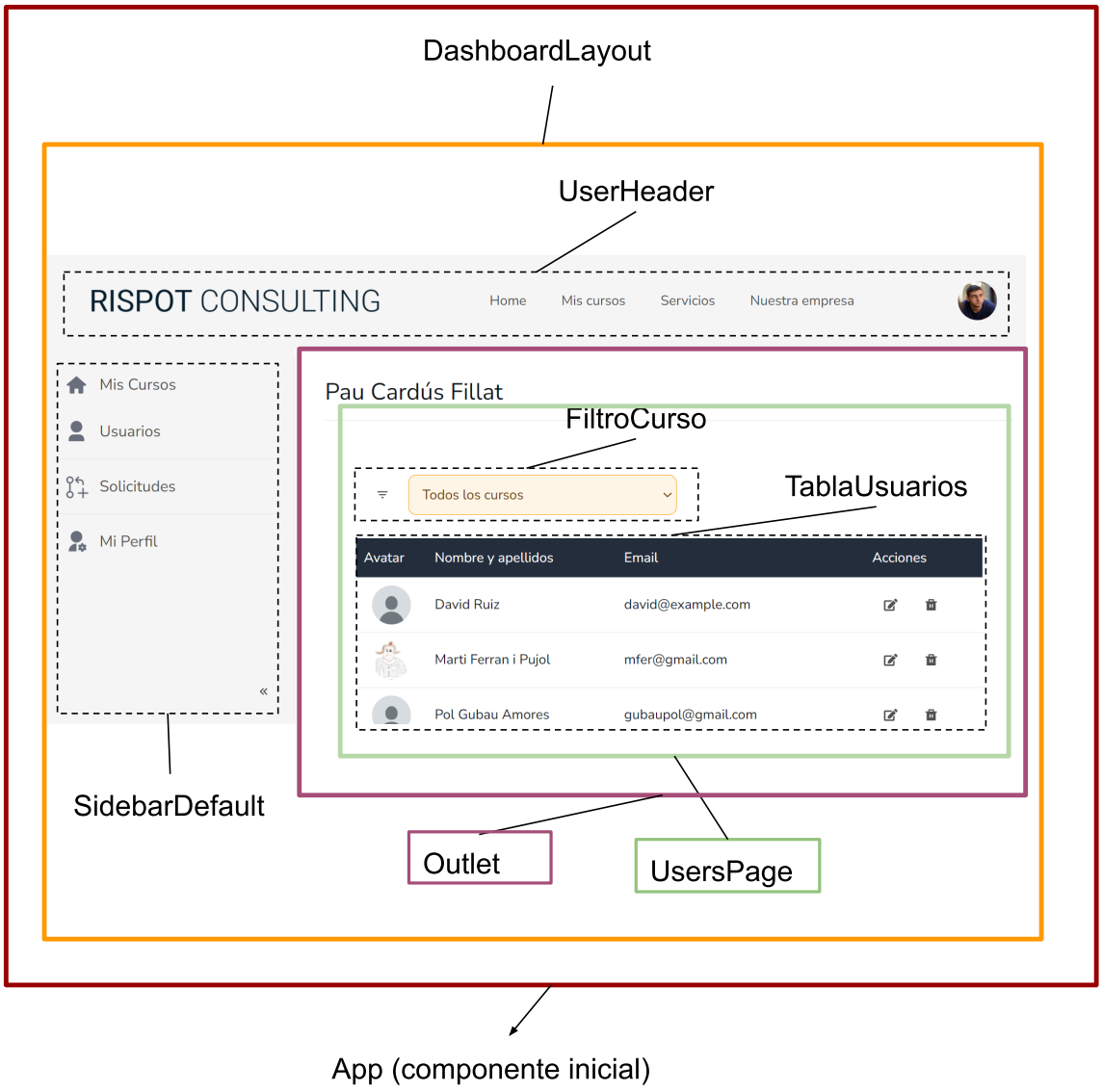

Fig. 28: Esquema de componentes
Fuente: Elaboración propia.

Esta vista corresponde a la interfaz que un usuario administrador visualizará al acceder a la página y componente de gestión de usuarios, denominada "UsersPage". El contenedor principal en color granate representa el componente base "App", sobre el cual se estructuran todos los demás elementos. Dentro de este componente base, encontramos el contenedor en color naranja "DashboardLayout", que incluye un menú lateral fijo ("SidebarDefault"), un encabezado de página con un menú ("UserHeader"), y un componente nativo de React ("Outlet") marcado en color morado. Este último componente se utiliza para renderizar dinámicamente los componentes "hijos" según la interacción del usuario.

En este caso específico, se muestra la página "UsersPage", la cual incluye un componente para filtrar cursos denominado "FiltroCurso" y una tabla ("TablaUsuarios") para visualizar registros y realizar acciones correspondientes.

```jsx
// UsersPage.jsx
import React, { useState } from "react";
import TablaUsuarios from "../../components/TablaUsuarios";
import FiltroCurso from "../../components/Filtros/FiltroCurso";

function UsersPage() {
  const [cursoSeleccionado, setCursoSeleccionado] = useState("");

  return (
    <div className="p-8">
      <FiltroCurso onChange={setCursoSeleccionado} />
      <TablaUsuarios cursoSeleccionado={cursoSeleccionado} />
    </div>
  );
}

export default UsersPage;
```

Fig. 29: Fragmento de código componente “UsersPage”
Fuente: Elaboración propia.

La figura anterior resume y ejemplifica el papel de React en mi aplicación mediante la componentización. En términos de programación, sigue el concepto de "cajas" mencionado anteriormente, donde cada componente se encapsula como una unidad independiente. En el contexto de la vista de la página "UsersPage", se integran los componentes del filtro de cursos y la tabla de usuarios, mostrando cómo estos se organizan y combinan dentro de la estructura general de la aplicación.

El uso de Tailwind en la aplicación

Tailwind está integrado en todos los componentes junto con el código HTML, y su utilización fue completamente simultánea con el uso de React y la maquetación de los componentes. A continuación se presentan dos ejemplos del uso de Tailwind en mi código para ilustrar cómo he aplicado esta librería:

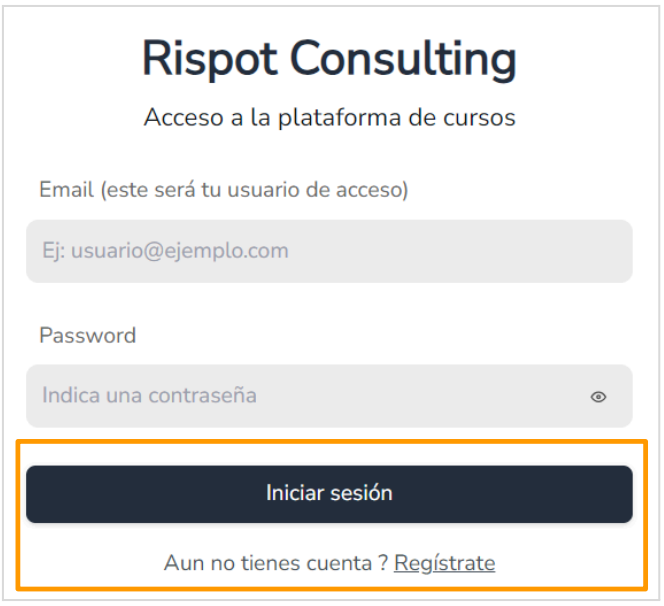

Fig. 30: Componente "Login"
Fuente: Elaboración propia.

```
  <Button
    className="w-full bg-[#232f3e] hover:bg-[#202e55] text-white"
    type="submit"
  >
    Iniciar sesión
  </Button>
</form>
<div className="flex flex-col items-center justify-center mt-4 gap-4">
  <Link className="text-sm text-gray-600" to="/register">
    Aun no tienes cuenta ? <span className="underline">Regístrate</span>
  </Link>
</div>
```

Fig. 31: Fragmento de código componente "Login"
Fuente: Elaboración propia.

En estas líneas se puede observar claramente la aplicación de las clases de Tailwind para la estructura, alineación de elementos y estilos. Para comprender mejor la contribución e impacto de Tailwind, se detallan las clases utilizadas, su equivalencia con el CSS tradicional y su función específica.

CLASE TAILWIND	EQUIVALENCIA CSS	FUNCIÓN
w-full	width: 100%	Ocupar el máximo ancho respecto al “padre”
bg-[#232f3e]	background-color: #232f3e	Poner el colo de fondo de color #232f3e
hover: bg-[#202e55]	background-color: #202e55	al pasar el ratón por encima del elemento cambiar el color de fondo a: #202e55
text-white	color: white	Color del texto en blanco
flex flex-col	display:flex flex-direction: column	Indica que habrá un comportamiento de rejilla, ordena los elementos en formato columna uno debajo de otro
items-center	align-items: center	Alinea los elementos “hijos” en el centro verticalmente
justify-center	justify-content:center	Alinea los elementos “hijos” en el centro horizontalmente
gap-4	gap: 1rem	Indica el espaciado que habrá entre los elementos “hijos” (4rem)
mt-4	margin-top: 1rem	Marca el margen superior que habrá encima del elemento que tenga esta clase
text-sm	font-size: 14px	Da un tamaño de 14px al texto que contenga esta clase
text-gray-600	color: #4B5563	El color de este texto será gris pero con una tonalidad media. (tailwind tiene de 100 a 900 siendo este último el más oscuro)

Fig. 32: Tabla comparativa clases Tailwind y CSS

Fuente: Elaboración propia.

Véase otro ejemplo más complejo de la construcción visual del componente “Inicio”:

```jsx
const Inicio = () => {
  return (
    <div className="flex flex-col  bg-gray-200 min-h-screen">
      <section className="flex bg-gray-200 flex-col justify-start ">
        {/* Primera fila con el logo de Rispot */}
        <div className="p-8">
          <img
            src="https://rispot.com/wp-content/uploads/2023/12/cropped-logo_possible3.png"
            alt="Rispot Logo"
            className="p-4 w-[60%] md:w-[20%]"
          />
        </div>

        {/* Segunda fila con dos columnas */}
        <div className="grid md:grid-cols-[50%,40%] w-screen items-center justify-center grid-cols-1">
          {/* Columna izquierda con título, párrafo y botones */}
          <div className="flex flex-col gap-8 p-12">
            <div classname="">
              <h1 className="text-5xl  lg:text-7xl  font-bold mb-8 ">
                Plataforma para cursos <br /> interactivos
              </h1>
              <p className="text-gray-500 text-xl ">
                Aquí aprenderás lo fundamental sobre el negocio de Amazon <br />{" "}
                de una forma única en el mercado
              </p>
            </div>
            <div className="flex gap-2 ">
              {" "}
              <Link to="/home">
                <Button className="px-8" variant="outline">
                  Explora Cursos
                </Button>{" "}
              </Link>
              <Link to="/login">
                <Button className="px-8" href="/login">
                  Iniciar sesión
```

Fig. 33: Fragmento de código componente “Inicio”

Fuente: Elaboración propia.

Si revisamos los fragmentos de código anteriores, se observan elementos como <Button/> no explicados en el trabajo. A primera vista, podría parecer una etiqueta para un botón estándar de HTML, que se escribiría como <button/> en minúsculas. Sin embargo, la primera letra en mayúscula indica que se trata de un componente. No se ha mencionado explícitamente en el esquema de "cajas" porque es una combinación de ambos conceptos: posee la estructura y funcionalidad básica de un botón HTML, pero también es un componente predefinido con estilos y comportamientos específicos. Aquí es donde entra en juego la biblioteca Pol-UI.

El uso de Pol-UI en la aplicación

Los componentes de Pol-UI se han utilizado ampliamente en la plataforma de cursos interactivos. Estos componentes van desde botones simples hasta menús completos con todas

las funcionalidades e interactividad integradas. El proceso general consistió en seleccionar el componente deseado para la web, consultar la documentación correspondiente para implementarlo y, si era necesario, añadir propiedades adicionales (tanto visuales como funcionales) conocidas como *props*.

El primer paso fué importar la biblioteca de componentes en el entorno de trabajo utilizando el gestor de paquetes npm (*Node Package Manager*). El comando necesario para instalar Pol-UI es: **npm install pol-ui**. Una vez ejecutado este comando en el terminal, el entorno estará configurado para reconocer y utilizar los componentes de Pol-UI mediante importaciones.

A continuación se presentan algunos ejemplos de uso de la biblioteca Pol-UI, ordenados de menor a mayor complejidad:

Ejemplo 1: Elementos de formulario

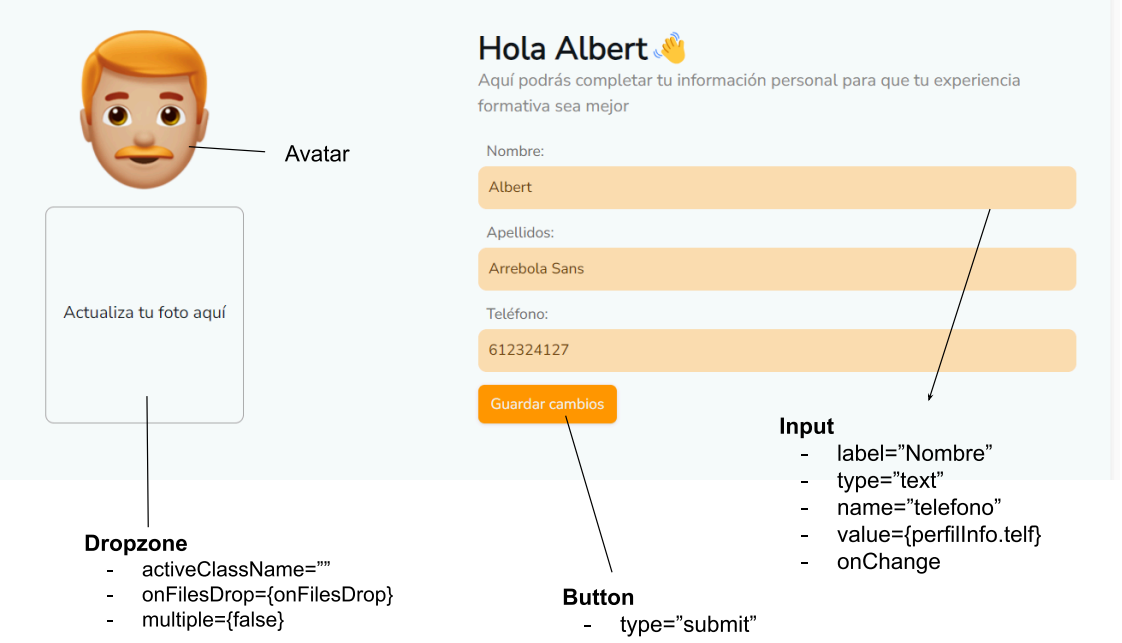

Fig. 34: Componentes Pol-UI en la página "mi-perfil"
Fuente: Elaboración propia.

Para este formulario hemos necesitado importar de POL-UI los componentes *Button, DropZone, FileList, Avatar, Input* y *toast.* Cada uno incorpora alguna propiedad o característica que aporta un valor o funcionalidad a los componentes naturales.

```
import { Button, Dropzone, FileList, Input, Avatar, toast } from "pol-ui";
```

Fig. 35: Fragmento código importación componentes Pol-UI

Fuente: Elaboración propia.

```
<Dropzone
  activeClassName="bg-neutral-200"
  onFilesDrop={onFilesDrop}
  multiple={false} // Cambiado a false para aceptar solo un archivo
  className="bg-transparent border border-neutral-400 border-solid
>
  <h2>Actualiza tu foto aquí</h2>
  {/* Texto actualizado para reflejar la restricción */}
  <FileList files={files} setFiles={setFiles} />
  {/* Muestra la lista de archivos seleccionados */}
</Dropzone>
</div>
<div>
  <h2 className="font-bold text-3xl">Hola {perfilInfo.nombre}👋</h2>
  <p className="mb-5 text-secondary-700">···
  </p>

  <Input
    label="Nombre:" ···
  />
  <Input
    label="Apellidos:" ···
  />
  <Input
    label="Teléfono:"
    type="text"
    name="telefono"
    value={perfilInfo.telefono}
    onChange={handleChange}
    className="mb-3"
  />

  <Button
    type="submit"
    className="col-span-2 w-fit bg-primary text-gray-100"
```

Fig. 36: Fragmento código uso de componentes POL-UI en “mi-perfil”

Fuente: Elaboración propia.

Ejemplo 2: Menú lateral

La biblioteca Pol-UI resultó invaluable para implementar componentes más elaborados como el menú lateral (*sidebar* en inglés). En la siguiente imagen se puede apreciar el uso del componente Sidebar, el cual incluye componentes "sidebarItem" como sus elementos hijos individuales del menú. Este componente ofrece propiedades como *"icon"*, que facilita la adición de iconos de manera sencilla, y un atributo "href" para utilizar cada elemento del menú como un enlace a otras partes de la página.

En la parte inferior se muestra un botón con un icono, que aunque no es un componente separado, está programado internamente dentro del Sidebar. Este botón gestiona las transiciones de apertura y cierre del menú de manera limpia y atractiva, mejorando así la responsividad del sitio web.

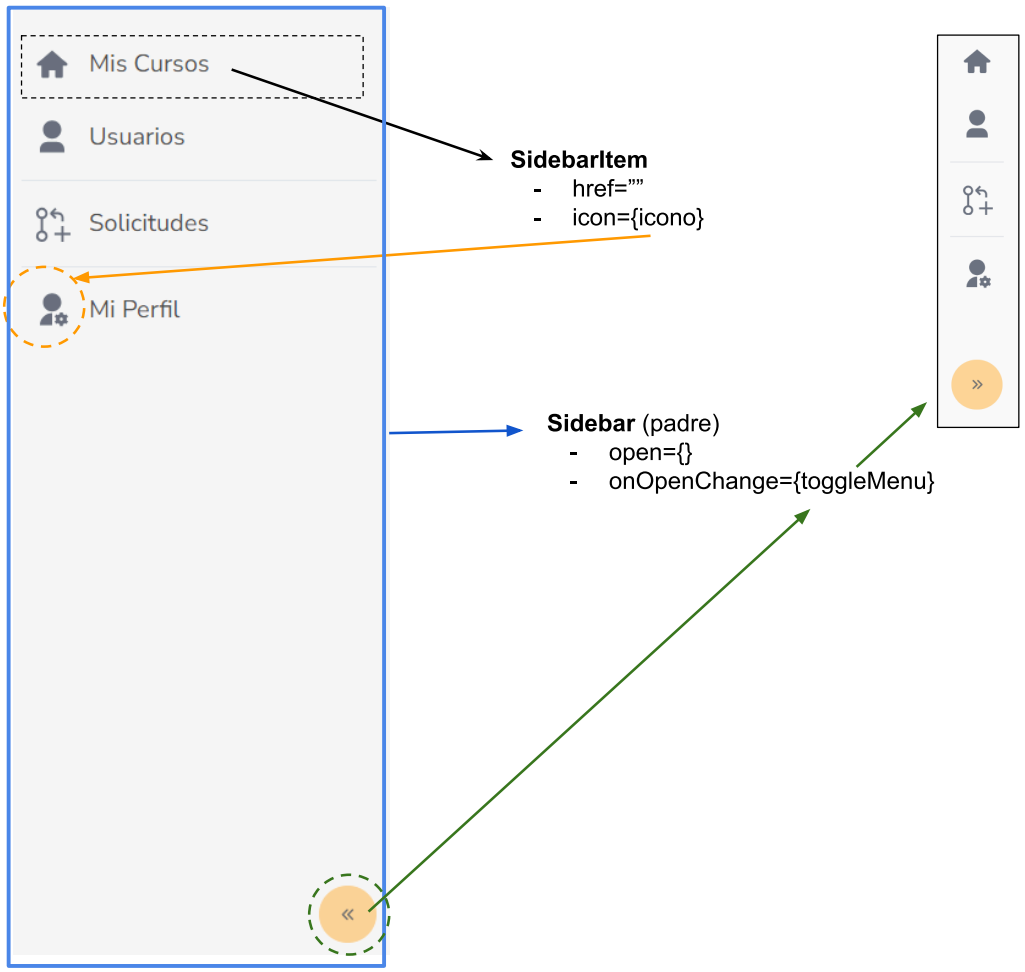

Fig. 37: Esquema componente Sidebar (Pol-UI)

Fuente: Elaboración propia.

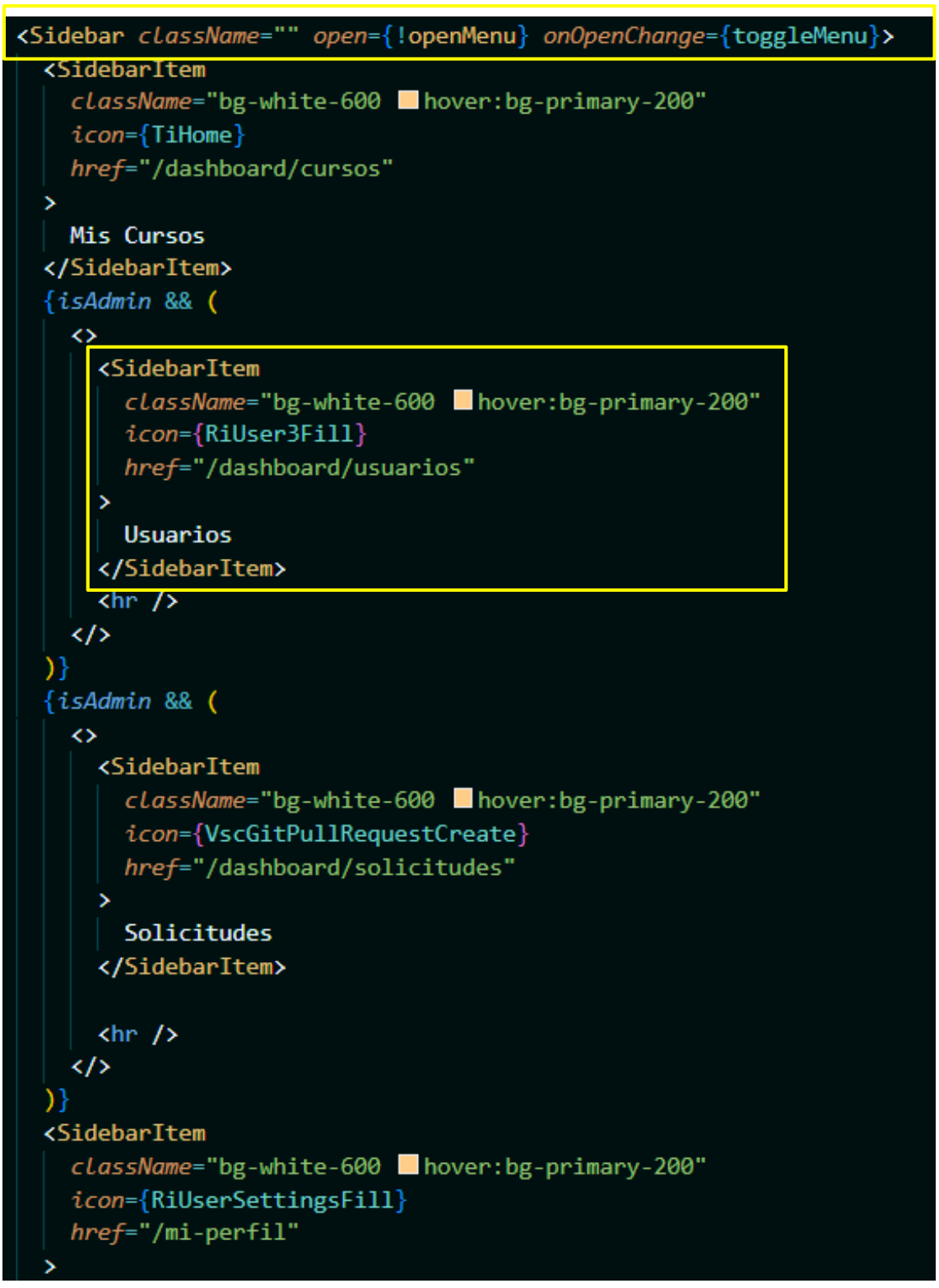

Fig. 38: Fragmento código componente Sidebar (Pol-UI)

Fuente: Elaboración propia.

Ejemplos como los mencionados son representativos del desarrollo visual de la aplicación de cursos interactivos, proporcionando interactividad y estética a cada uno de los componentes que integran la plataforma en su conjunto.

Para completar el *sprint*, era necesario integrar todas las partes previamente desarrolladas. Aunque cada sección estaba avanzada en cuanto a maquetación, funcionaban de manera independiente y no estaban conectadas entre sí. Fue en este punto cuando implementé el enrutador (*router* en inglés).

El uso de React Router en la aplicación

Una vez montados los componentes por separado utilizando las tecnologías mencionadas, el siguiente paso era diseñar el enrutamiento de la aplicación. Esto implica crear todas las rutas y direcciones posibles que un usuario podría seguir para visualizar las diferentes páginas en el navegador.

```
function App() {
  return (
    <AppProvider>
      <Router>
        <UserProvider>
          <GlobalProvider>
            <NextUIProvider>
              <div className="App ">
                <Routes>
                  {/* RUTAS PRIVADAS */}
                  <Route element={<DashboardLayout />} path="dashboard/">
                    {/* La ruta a continuación es index, por lo que coge el mismo path
                    <Route index element={<CursosPage />} />
                    <Route
                      path="usuarios"
                      element={
                        <PrivateRoute roles={["admin"]}>
                          <UsersPage />
                        </PrivateRoute>
                      }
                    />
                    <Route
                      path="cursos"
                      element={
                        <PrivateRoute roles={["admin"]}>
                          <CursosPage />
                        </PrivateRoute>
                      }
                    />
```

Fig. 39: Fragmento de código de router en el componente "App"

Fuente: Elaboración propia.

El objetivo de esta sección, más que mostrar el enrutador íntegro, es entender cómo funcionan las rutas en la aplicación de cursos interactivos que nos ocupa. Para ilustrar esto, veamos un fragmento de código similar al ejemplo teórico, destacando algunos elementos clave. En la función App(), que representa el componente principal de la aplicación (marcado en rojo), se renderiza todo el contenido. Después de configurar todos los proveedores necesarios para el proyecto (marcados en amarillo), nos encontramos con el componente "Routes". Este componente contiene todas las rutas de la plataforma de cursos.
Para la explicación práctica, consideremos la primera ruta. Aquí tenemos dos niveles de rutas: una principal que corresponde a la página de inicio (home), indicada en verde para especificar el parámetro de búsqueda en la URL, y un nivel inferior donde se definen los componentes específicos que se mostrarán, marcados en rosa junto al elemento "element", que determina qué componente se mostrará en esa ruta.

En relación al "PrivateRoute", su función detallada se explicará más adelante en la sección titulada "Sprint 6: Roles, sesiones y seguridad".

Sprint 3: Programación del *backend*

Una vez que la aplicación tenía cierta forma, se procedió a preparar el backend. Como se ha mencionado a lo largo del trabajo, se usó un backend as a service (BaaS). Esto significa que no tuve que diseñar las rutas y la arquitectura interna para la gestión de información. El servicio Supabase ya ofrecía toda una infraestructura robusta y escalable que me permitió centrarme únicamente en crear las tablas necesarias con sus relaciones y atributos característicos.

Configuración del entorno en Supabase

Supabase ofrece dos grandes entornos (*schemas*) para los desarrolladores. El entorno principal llamado "public" es donde se crean las tablas con la información dinámica necesaria para la aplicación en cuestión. Este entorno es el núcleo de la base de datos, donde gestionamos toda la información relacionada con los cursos, inscripciones y otros datos relevantes.

Además del entorno “public”, Supabase proporciona un entorno exclusivo para la gestión de usuarios, conocido como “auth”. A diferencia del entorno “public”, el entorno “auth” ya viene con una serie de tablas preconfiguradas, como la tabla users. Esta tabla facilita la gestión de inicios de sesión, registros, roles y el encriptamiento de contraseñas, permitiendo un manejo seguro y eficiente de la autenticación de usuarios.

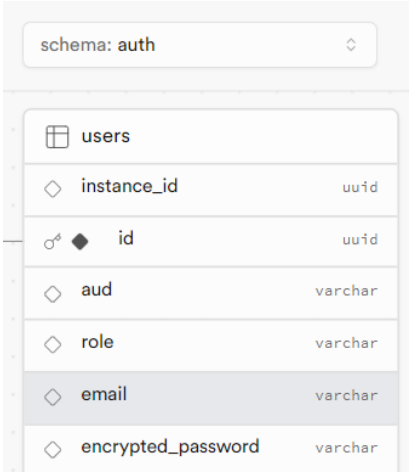

Fig. 40: Tabla users *schema auth*
Fuente: supabase.com (24/05/2024).

Entorno o *schema* “public”

En el entorno “public”, se crearon varias tablas esenciales para la aplicación, incluyendo:

- **Tabla perfiles**: Esta tabla amplía la información del usuario almacenada en auth.users. Incluye campos adicionales específicos de la aplicación, como teléfono, apellidos o un avatar como imagen de perfil.
- **Tabla “usuario_en_curso”**: Registra las inscripciones de los usuarios a los cursos. Incluye campos como id, “usuario_id”, “curso_id”, “modulo_id” y estado, estableciendo las relaciones necesarias entre usuarios y cursos.
- **Tabla “curso_usuario”**: Guarda qué usuarios que hay en cada curso, las solicitudes de estos en cada uno de los cursos y el *status* de estas (aceptada, en curso o solicitada).

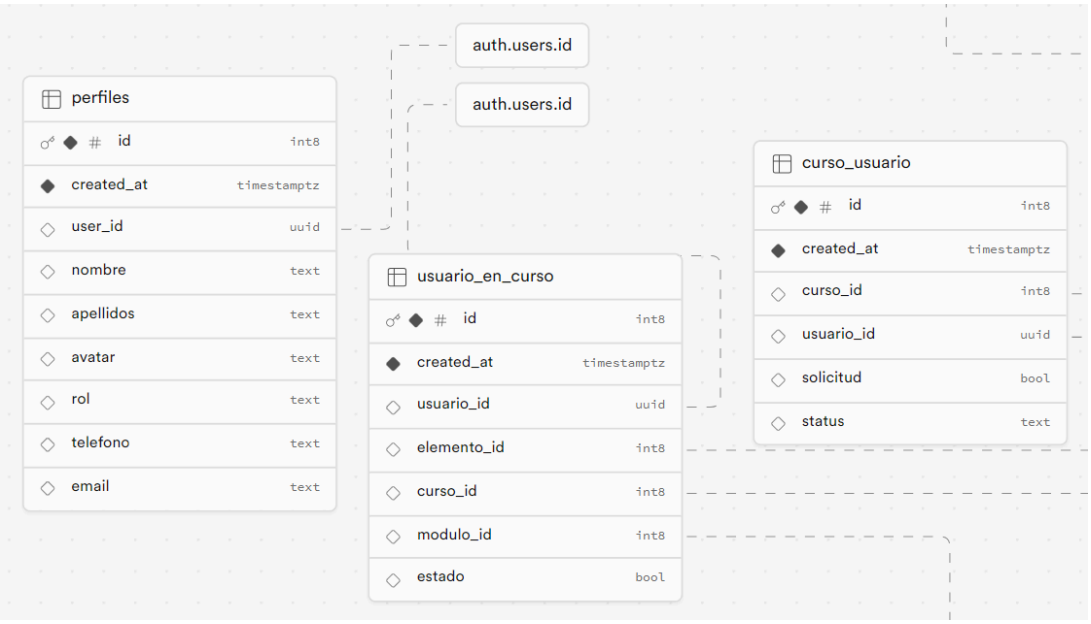

Fig.41: Diagrama E-R Supabase (I)
Fuente: supabase.com (24/05/2024).

La integración entre los esquemas public y "auth" se realiza mediante la relación de claves foráneas. Por ejemplo, la propiedad "usuario_id" en las tablas del esquema "public" se relaciona con el *id* en la tabla "auth.users", asegurando una referencia consistente y segura entre los datos de usuarios y cursos. Esto permite una gestión eficiente de las inscripciones y la creación de cursos, manteniendo la integridad y seguridad de los datos de los usuarios.

Detalle de tablas de cursos y módulos (primera capa de profundidad)

A continuación, se presenta la estructura de las tablas que componen la primera capa de cursos y módulos:

- **Tabla curso**: Almacena la información básica de cada curso. Incluye campos como id, nombre, descripción, imagen, precio, duración y instructor.
- **Tabla módulo**: Almacena la información de los módulos que componen cada curso. Incluye campos como id, nombre, descripción, imagen y varios indicadores booleanos (verdadero o falso) cómo "have_video", "have_examen"y otros para identificar los tipos de elementos que contiene cada módulo.

- **Tabla “curso_modulo”**: Relaciona los cursos con los módulos que los componen. Incluye campos como “curso_id” y “modulo_id”.

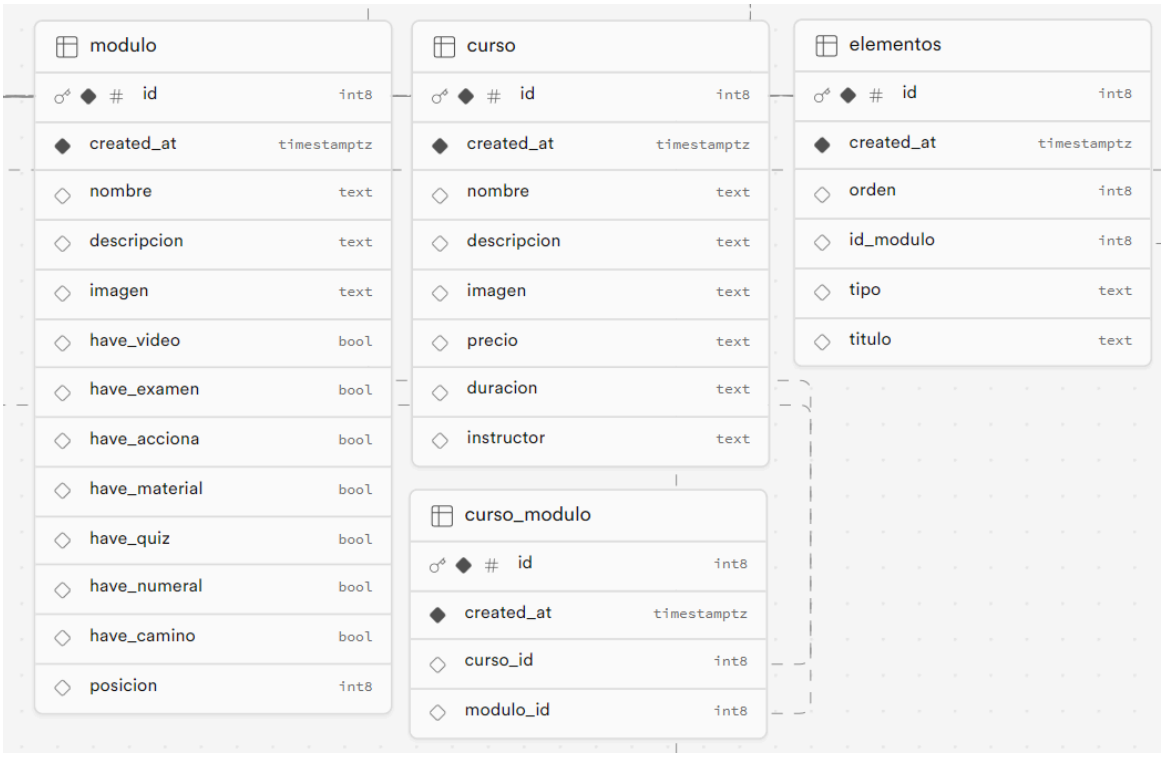

Fig.42: Diagrama E-R Supabase (II)
Fuente: supabase.com (24/05/2024).

Detalles de los elementos y lecciones (segunda capa de profundidad)

Dentro de los módulos, se encuentran las lecciones que pueden ser de diferentes tipos. Las tablas correspondientes son:

- **Tabla elementos**: Es una tabla central que agrupa todos los elementos (lecciones) dentro de los módulos. Incluye campos como id, orden,” id_modulo”, tipo (video, examen, material, etc.) y título.
- **Tabla video**: Contiene información sobre los videos. Incluye campos como id, “url_video”, duración, etc.
- **Tabla examen**: Contiene información sobre los exámenes. Incluye campos como id, preguntas, duración, etc.
- **Tabla material**: Contiene información sobre el material adicional. Incluye campos como id, “url_material”, “tipo_material”, etc.

- **Tabla acciona**: Contiene información sobre actividades prácticas. Incluye campos como id, descripción, duración, etc.
- **Tabla numeral**: Contiene información sobre numerales importantes. Incluye campos como id, contenido, etc.
- **Tabla quiz**: Contiene información sobre quizzes. Incluye campos como id, preguntas, duración, etc.

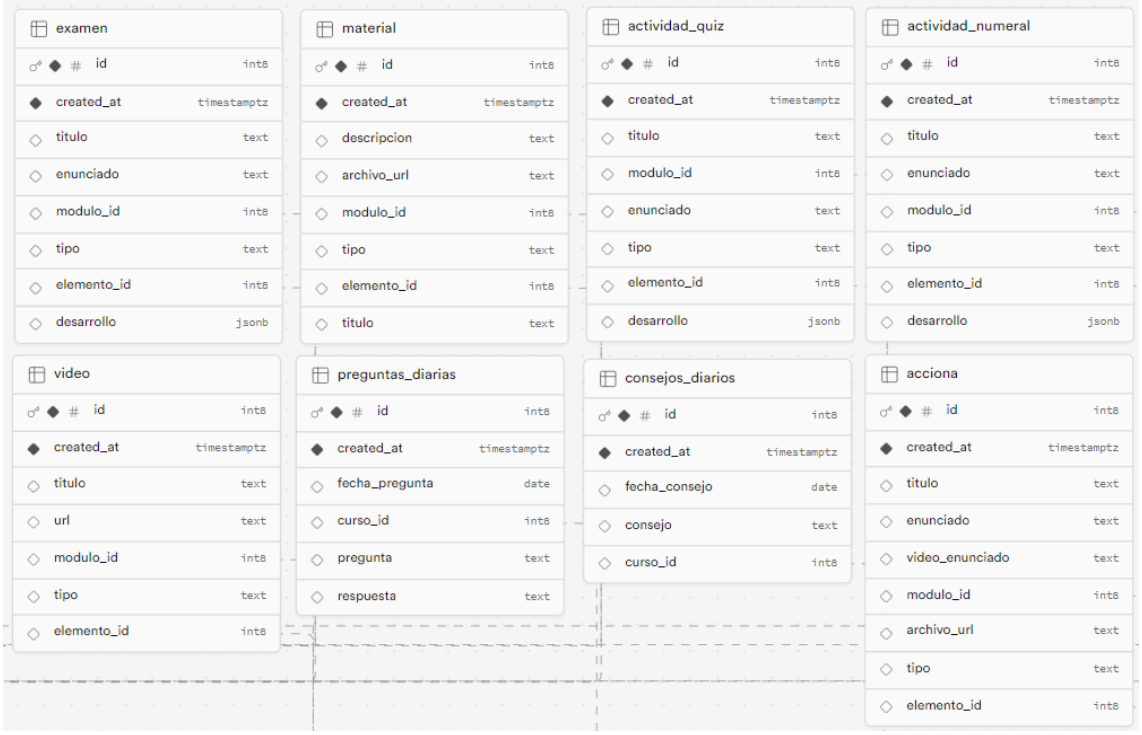

Fig.43: Diagrama ER Supabase (III)
Fuente: supabase.com (24/05/2024).

Para clarificar toda esta información y tablas, veamos una situación en nuestra plataforma donde se explique cómo se vinculan las tablas mencionadas.

Imaginemos a Carlos, quien se registra en nuestra plataforma como usuario (auth.users, perfiles) . Él decide inscribirse en el curso "Vender en Amazon desde Cero" (curso, “usuario_en_curso, “curso_usuario”). Este curso se organiza en módulos como "Introducción a Amazon como plataforma de ventas" y "Optimización de listados" (“curso_modulo”, elementos), cada uno con lecciones específicas como recursos en pdf y

videos tutoriales (material, video). Una vez vista la estructura de la base de datos veamos como se conecta con la aplicación.

src > supabase > JS supabaseClient.js > ...

```
1   import { createClient } from "@supabase/supabase-js";
2
3   const supabaseUrl = import.meta.env.VITE_SUPABASE_URL;
4   const supabaseAnonKey = import.meta.env.VITE_SUPABASE_ANON_KEY;
5   const storageBucket = import.meta.env.VITE_STORAGE_BUCKET;
6
7   export const supabase = createClient(
8     supabaseUrl,
9     supabaseAnonKey,
10    storageBucket
11  );
```

Fig.44: Fragmento de código de configuración de supabase en el proyecto
Fuente: Elaboración propia.

El fragmento de código anterior muestra cómo se incorporó la *suite* Supabase en la aplicación. A continuación mediante capturas de pantalla de la interfaz y con pequeños ejemplos de código propio se busca hacer un recorrido por los 4 ejes principales de Supabase relacionados con la plataforma.

1. Gestión de usuarios

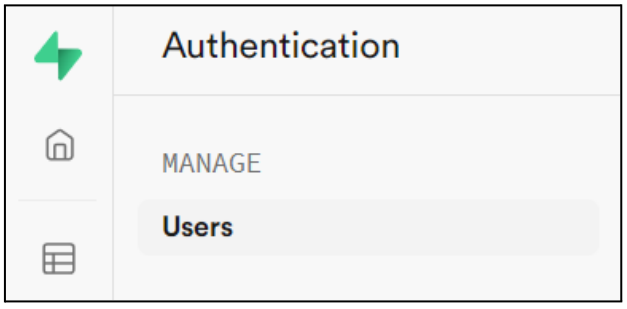

Fig.45: Captura de pantalla sección *Authentication* de Supabase
Fuente: supabase.com (17/05/2024).

Email	Phone	Provider	Created	Last Sign In	User UID
gubaupol@gmail.com	-	Email	21 Apr, 2024 11:48	22 Apr, 2024 10:07	d7cb9570...
mfer@gmail.com	-	Email	17 Apr, 2024 20:59	17 Apr, 2024 21:04	96819f62...

Fig. 46: Tabla de *users* en Supabase
Fuente: supabase.com (17/05/2024).

Supabase ofrece internamente una infraestructura que permite crear usuarios de una manera muy sencilla, es capaz de guardar la información de su sesión, aplicarle un rol, añadir y quitar permisos a un usuario. Para este sprint, únicamente se definieron usuarios de prueba para poder desarrollar de una manera más cómoda. Los roles, sesiones y permisos y accesos a la plataforma se emplazaron al sprint 5: Funcionalidades panel admin

2. Gestión de tablas y base de datos

Esta es la funcionalidad central o base que tiene como objetivo almacenar los datos de una manera estructurada en tablas para su posterior uso en la aplicación dependiendo de la lógica que se aplique.

Fig. 47: Captura de pantalla sección *Table Editor* de Supabase
Fuente: supabase.com (17/05/2024).

Aunque los detalles específicos sobre el contenido de las tablas, su estructura y las relaciones entre ellas se desarrollarán más exhaustivamente en el sprint 3 - Programación del backend, en este punto se puede observar cómo he utilizado el editor de tablas para crear y definir las diferentes entidades. Este proceso incluyó la determinación de los campos y la información que deseaba almacenar en cada tabla.

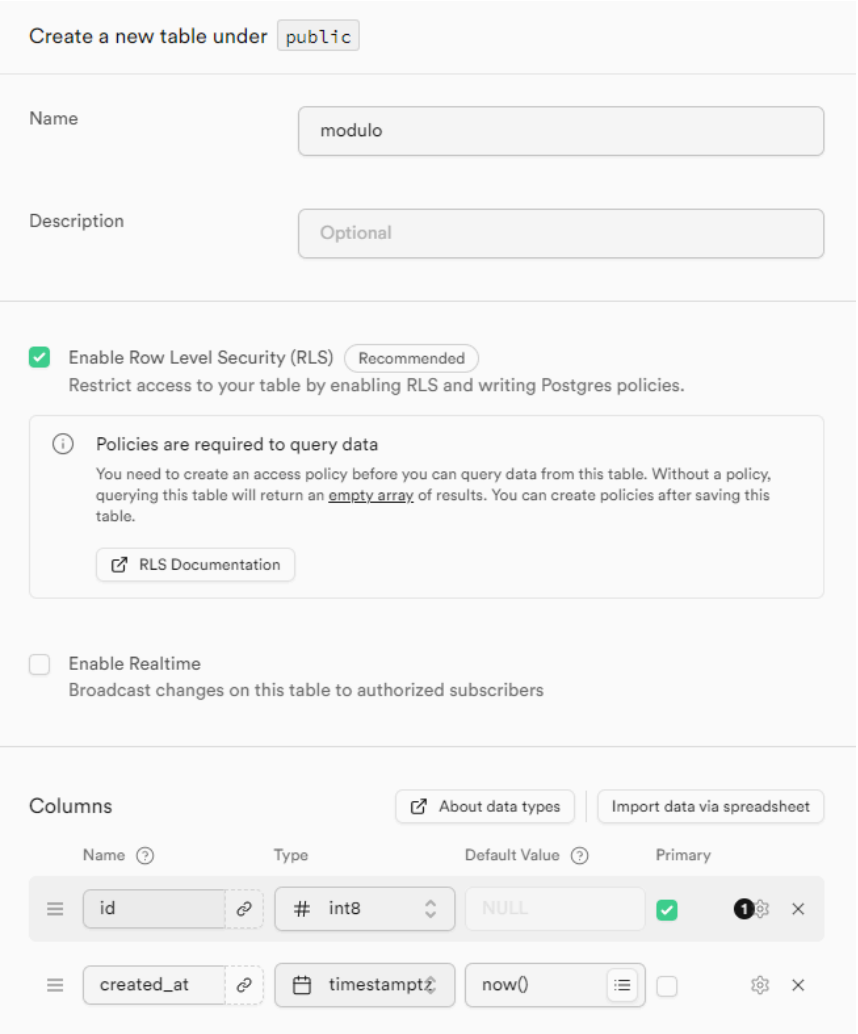

Fig. 48: Interfaz creación de nueva tabla en Supabase
Fuente: supabase.com (17/05/2024).

En esta captura de pantalla de la interfaz de Supabase, tomada durante la creación de la tabla "módulo", se puede apreciar claramente una de las ventajas fundamentales de utilizar esta herramienta. La interfaz proporciona una experiencia completa y sencilla que facilita significativamente la creación de tablas, la adición de campos, la configuración y otros detalles relacionados. Esta facilidad de uso contrasta notablemente con la complejidad de crear tablas manualmente utilizando lenguaje Postgre SQL.

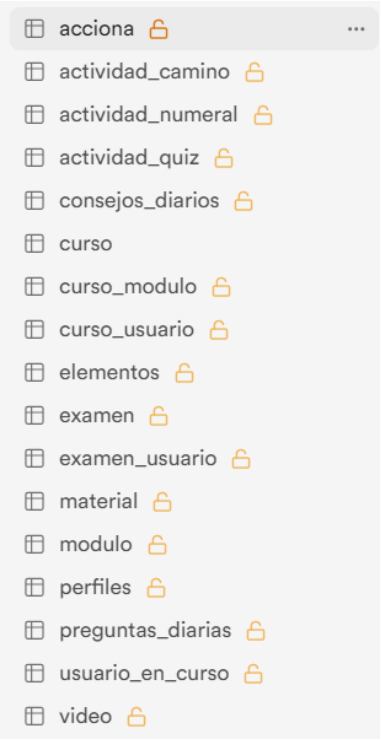

Fig. 49: Lista de tablas del proyecto en Supabase
Fuente: supabase.com (17/05/2024).

Gracias a toda la infraestructura proporcionada por Supabase y a la información almacenada en estas tablas, es posible acceder utilizando las funciones que el propio servicio nos ofrece. Dentro del editor de tablas, se encuentra disponible una documentación detallada para cada tabla, la cual describe las funciones disponibles para manipular los datos almacenados en ellas.

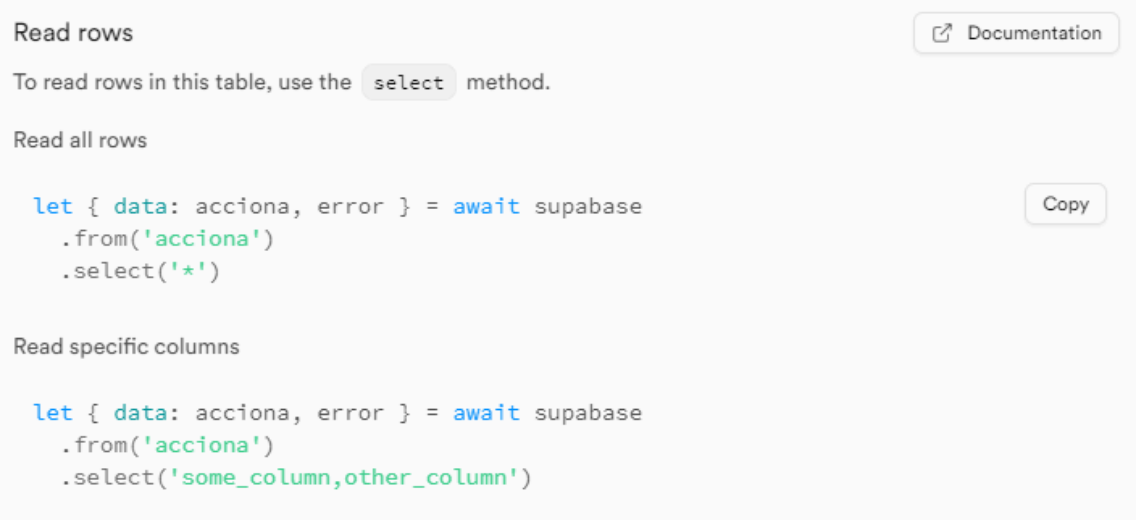

Fig. 50: Documentación de funciones para la tabla "Acciona"
Fuente: supabase.com (17/05/2024).

3. Almacenamiento de medios (*storage)*

Este servicio "back as a service" también ofrece un sistema de almacenamiento basado en AWS (Amazon Web Services), que permite alojar contenido digital como imágenes, videos, archivos y documentos en un servidor dedicado y organizado. Esta funcionalidad está accesible a través de la misma interfaz de Supabase, que proporciona funciones específicas para la gestión y manipulación de estos archivos. El proceso es simple: se crea un bucket (cubo o balde), dentro del cual se pueden almacenar y gestionar archivos según las necesidades del usuario, incluyendo operaciones como recuperación, eliminación y actualización.

A continuación se detallan los buckets creados para mi proyecto, junto con algunos fragmentos de código que ilustran cómo acceder a ellos utilizando las funciones proporcionadas por Supabase.

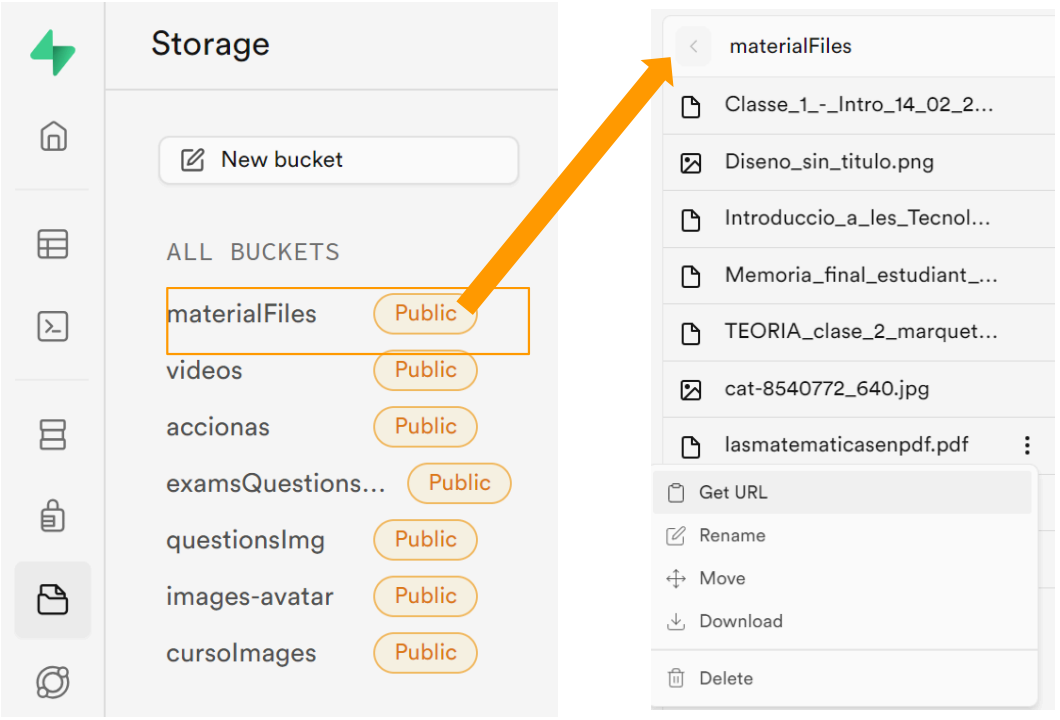

Fig. 51: *Buckets* del proyecto de Supabase
Fuente: supabase.com (17/05/2024).

4. Funciones de la base de datos

Para operaciones sencillas como leer tablas, actualizar registros, eliminarlos o insertarlos se puede usar las funciones básicas que hemos visto en el punto de gestión de tablas pero en algunas ocasiones son necesarias operaciones más complejas, consultas que combinan diferentes tablas, subconsultas... y para eso supabase no tiene métodos propios. Lo que sí nos permite es crear funciones SQL y ponerles nombre para luego desde el código "llamarlas" y poder ejecutar así estas consultas más complejas.

INVOKE FUNCTION

```
let { data, error } = await supabase
  .rpc('obtener_cursos_usuario', {
    userid
  })
if (error) console.error(error)
else console.log(data)
```

Fig. 52: Documentación de la función "obtener_curso_usuario()"
Fuente: supabase.com (17/05/2024).

La documentación de la función nos muestra como se "llaman" a dichas funciones. Se muestra un ejemplo de la función **obtener_cursos_usuario()**, que combina las tablas "curso" y "curso_usuario".

Name	Arguments	Return type	Security
calcula_progreso_curso	cursoid bigint, usuarioid uuid	TABLE(progreso integer)	Invoker
insertar_elementos_curso_usuari	usuario_id uuid, curso_id_par...	void	Invoker
obtener_cursos_usuario	userid uuid	TABLE(curso_id bigint, nombre text, descripcion text, imagen text)	Invoker
obte_cursos_usuarios	userid uuid	TABLE(curso_id bigint, nombre text, descripcion text, imagen text, instructor text, duracion text, precio text)	Invoker

Definition

The language below should be written in `plpgsql` .

```
BEGIN
    RETURN QUERY
    SELECT c.id, c.nombre, c.descripcion, c.imagen, c.instructor, c.
    duracion, c.precio
    FROM curso c
    JOIN curso_usuario cu ON c.id = cu.curso_id
    WHERE cu.usuario_id = userid and solicitud = 'True';
END;
```

Fig. 53: Definición de la función "obtener_cursos_usuario()"

Fuente: supabase.com (17/05/2024).

Sprint 4: Funcionalidades panel administración

Después de avanzar significativamente en la maquetación de la mayoría de los componentes con contenido estático y de establecer la base de datos con la información inicial, llegó el momento de integrar estos elementos para comenzar a implementar las funcionalidades "reales".

El uso de React ha sido fundamental para gestionar el estado de la aplicación y actualizar dinámicamente el contenido sin necesidad de recargar la página. Para lograr este comportamiento, he utilizado los *hooks* que me han permitido conectar los componentes previamente maquetados con datos reales procedentes de la base de datos mediante JavaScript.

Durante la fase inicial del desarrollo, me enfoqué en realizar pruebas para mostrar información dinámica de las tablas en mi aplicación. El objetivo principal era familiarizarme con las tecnologías y los hooks de React, dejando de lado el estilo visual para centrarme en la funcionalidad. Estas pruebas consistieron en implementar formularios simples para insertar información en Supabase, así como tablas para mostrar y capturar la información recuperada de la base de datos.

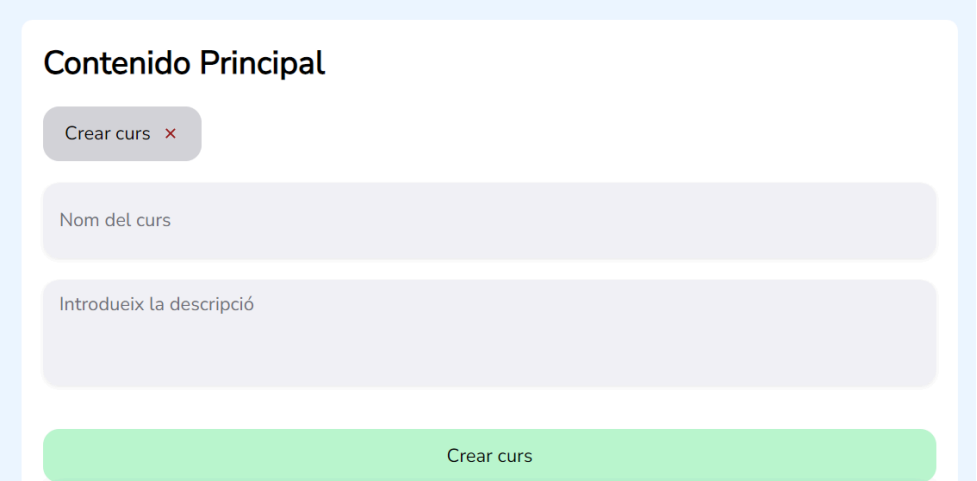

Fig. 54: Formulario de pruebas para insertar curso
Fuente: Elaboración propia.

Al superar las pruebas con éxito y ya familiarizado con las tecnologías empecé a conectar los prototipos reales maquetados con React usando javascript y los 3 *hooks* principales explicados en apartados anteriores. A continuación se muestra con ejemplos propios el uso que he hecho de estos *hooks* y el uso que he hecho de las tablas y funciones de supabase diseñadas en el *sprint* anterior.

Ejemplo de "Use State"

```
const [loading, setLoading] = useState(false);
const handleDeleteConsejo = async (consejoId) => {
  try {
    setLoading(true);
```

Fig. 55: Fragmento de código ejemplo *use State*
Fuente: Elaboración propia.

En este fragmento de código se muestra el funcionamiento de "*useState*", iniciando el estado "loading" qué es el que marca la aparición o no de la ruedita que indica que está cargándose algún contenido. Esta empieza en falso (sin aparecer) y cuando se llama a la función "handleDeleteConsejo()" mientras espera a que se realice la lógica se pone el *loader* en verdadero para que el usuario sepa que se está ejecutando el proceso.

Ejemplo de "Use Effect"

```
useEffect(() => {
  const fetchConsejos = async () => {
    try {
      setLoading(true);
      const { data, error } = await supabase
        .from("consejos_diarios")
        .select("*")
        .eq("curso_id", Number(id));

      if (error) {
        throw new Error(error.message);
      }

      setConsejosData(data);
    } catch (error) {
      handleError(error, "Error al obtener los consejos de Supabase");
    } finally {
      setLoading(false);
    }
  };

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

Fig. 56: Fragmento de código ejemplo *use Effect*
Fuente: Elaboración propia.

Este fragmento se encuentra dentro del componente "ConsejosDiariosList", y el "*useEffect*" en este caso lo que hace es definir y ejecutar la función "fetchConsejos()" cuyo objetivo es leer (mediante la función .select() de supabase ya explicada) todos los registros de la tabla "consejos_diarios" que tengan un id concreto.

Cada vez que se renderiza el componente "ConsejosDiariosList" el "*useEffect*" actuará y obtendrá los registros de la tabla de supabase.

Ejemplo de "Use Context"

Un ejemplo de "useContext" puede ser este componente llamado "UserContext" que será nuestro baúl donde se guarda la información del usuario (guardandola en el estado *user*) junto con datos del perfil (guardándola en el estado "perfilInfo"). De esta manera desde cualquier lugar de la aplicación, siempre que hagamos referencia al "UserContext" podremos acceder a la información sin tener que pasar ninguna variable.

```
export const UserContext = createContext();

export const UserProvider = ({ children }) => {
  const [user, setUser] = useState(() => {
    const storedUser = localStorage.getItem("user");
    return storedUser ? JSON.parse(storedUser) : null;
  });
  const [perfilInfo, setPerfilInfo] = useState(() => {
    const storedPerfilInfo = localStorage.getItem("perfilInfo");
    return storedPerfilInfo ? JSON.parse(storedPerfilInfo) : null;
  });
```

Fig. 57: Fragmento de código ejemplo de *use Context*
Fuente: Elaboración propia.

A continuación se muestra otro fragmento esta vez del componente "UserHomePage", donde se necesita saber el rol del usuario que está navegando para dirigirlo a la página de "mis-cursos" o a la página de "login". El rol se encuentra en el perfil del usuario es por eso que para poder saberlo, es necesario acceder al estado perfilInfo, que ya hemos visto donde se encuentra definido.

Gracias a este *hook* podemos acceder a ese "baúl" y desde allí conocer el rol.

```
import { UserContext } from "../../UserContext";
import { useContext } from "react";

const UserHomePage = () => {
  const { perfilInfo } = useContext(UserContext);

  return (
    <div className="">
      <section className="flex h-full gap-4 ">
        <div className="w-full">
          <UserHeader />
          <div className="space-y-2 p-8">
            <a href={perfilInfo && perfilInfo.rol ? "mis-cursos" : "login"}>
```

Fig. 58: Fragmento de código ejemplo importación del contexto
Fuente: Elaboración propia.

Para crear un entorno dinámico y funcional en el panel de administración, combiné las capacidades de React con las herramientas proporcionadas por Supabase, aprovechando sus

funcionalidades esenciales para consumir y gestionar datos en las tablas, utilizar su sistema de autenticación integrado y diseñar la lógica para el almacenamiento de archivos.

Inicialmente, desarrollé la interfaz completa utilizando las funciones básicas necesarias para interactuar con la base de datos. Esto incluyó la implementación de operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en las tablas configuradas a través del editor de tablas mencionado anteriormente.

Véase algún ejemplo de lectura e inserción de registros de tablas en el propio código:

Ejemplo de lectura de la tabla "acciona"

```
.supabase.from("nombre de la tabla").select("campos que quieras leer")
```

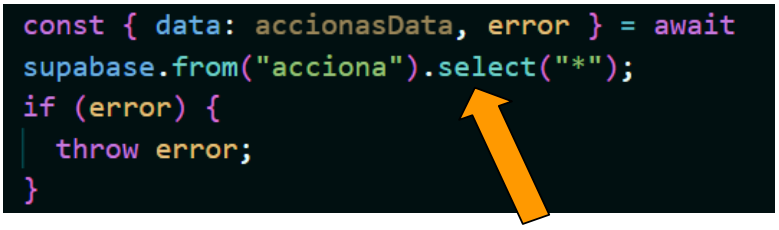

Fig.59: Fragmento de código para seleccionar elementos de la tabla "acciona"
Fuente: Elaboración propia.

Ejemplo de inserción de la tabla "acciona"

```
.supabase.from("nombre de la tabla").insert([información a insertar])
```

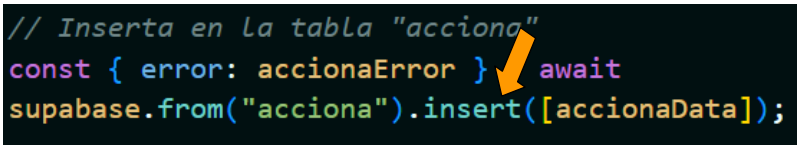

Fig. 60: Fragmento de código para insertar elementos de la tabla "acciona"
Fuente: Elaboración propia

Para habilitar estas funcionalidades esenciales en el panel de administración, desarrollé diversas funciones que permiten el acceso y la gestión de datos en las tablas correspondientes. Implementé operaciones completas de CRUD para cursos, módulos y lecciones, lo que incluyó la capacidad de crear, editar y eliminar estos elementos directamente desde el panel administrativo.

Inicialmente, los cursos estaban configurados con contenido estático, como textos e imágenes con URLs externas proporcionadas por los administradores. Sin embargo, para cumplir con

los requisitos del cliente de manejar contenido multimedia de manera directa desde la plataforma, procedí a configurar y conectar los mencionados *buckets* de almacenamiento de Supabase.

En términos de implementación de código , utilicé la API de almacenamiento de Supabase para interactuar con los buckets. Mediante supabase.storage.from("nombre del bucket"), pude ejecutar acciones como cargar archivos (.upload) y obtener la URL pública de archivos almacenados (.getPublicUrl). A continuación se presenta un ejemplo con fragmentos de código ilustrativos:

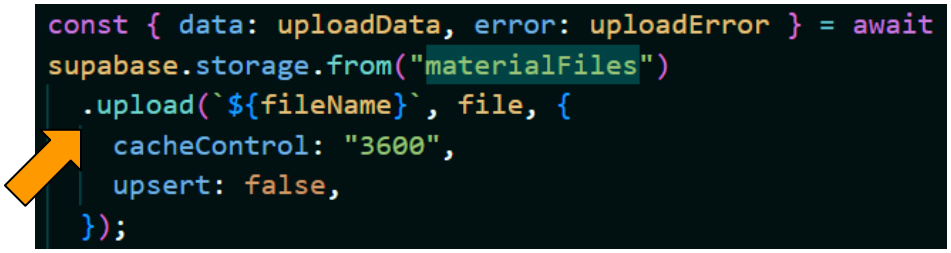

Fig. 61: Fragmento de código para subir archivos al *bucket* "materialFiles"
Fuente: Elaboración propia.

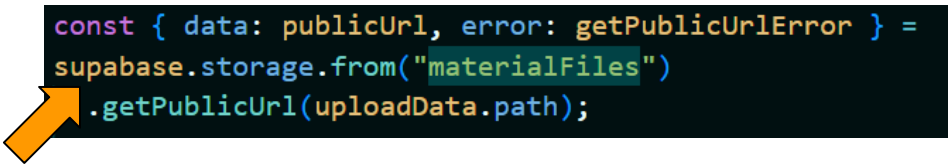

Fig. 62: Fragmento de código para obtener url del *bucket* "materialFiles"
Fuente: Elaboración propia.

En este ejemplo, hemos explorado el funcionamiento del *bucket* "materialFiles", destinado a almacenar diversos materiales que los administradores de cursos pueden añadir a sus módulos, como archivos PDF, videos, presentaciones y otros formatos. Los *buckets* configurados anteriormente en el sprint anterior siguen el mismo patrón de uso.

En esta fase del proyecto, la plataforma había alcanzado la capacidad de facilitar tanto la creación y gestión de contenido estático como la carga y descarga dinámica de archivos. Con un notable retraso con respecto a la planificación inicial, el *sprint* 4 concluía.

Sprint 5: Funcionalidades panel usuario

El siguiente paso fue iniciar la implementación del sistema de gestión de usuarios, abordando el registro y el inicio de sesión en la plataforma. Supabase proporciona un servicio de autenticación integral, discutido previamente en el *sprint* de *backend*, que utilicé para administrar el acceso de los usuarios. A continuación, se presentan ejemplos de cómo se realizó el registro y el inicio de sesión en el código.

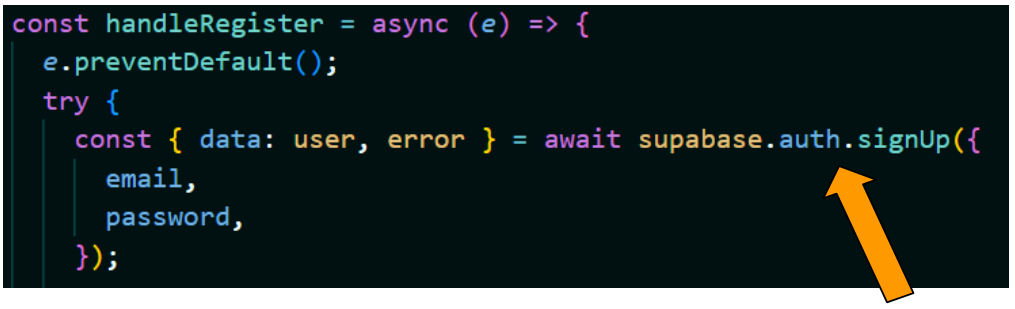

Fig.63: Función para el registro de usuarios

Fuente: Elaboración propia.

La función **auth.signUp()** ya es intrínseca de supabase, el ya sabe que tiene que registrar a un usuario nuevo en la plataforma, guarda sus credenciales, le asigna un id y almacena todo para la posterior gestión de ese usuario. (sesión, último acceso, rol... etc)

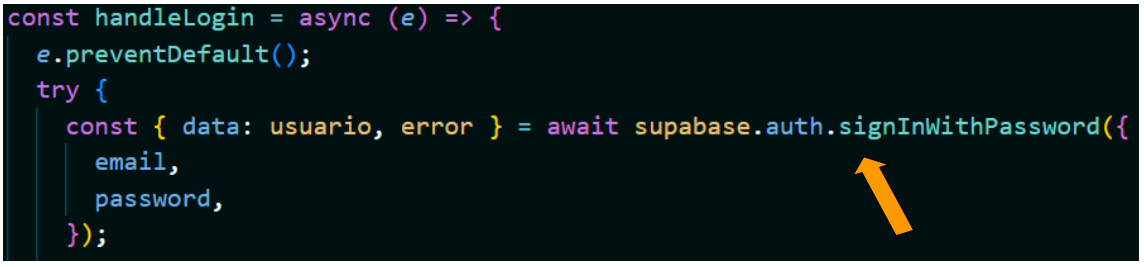

Fig.64: Función para el inicio de sesión de usuarios

Fuente: Elaboración propia.

Otro ejemplo es el método **auth.signInWithPassword()** proporcionado por Supabase, el cual gestiona el inicio de sesión de un usuario registrado previamente con apenas 2 líneas de código. Este servicio maneja la seguridad, las sesiones y toda la gestión de usuarios de manera transparente, permitiéndonos enfocarnos completamente en el uso funcional de la aplicación.

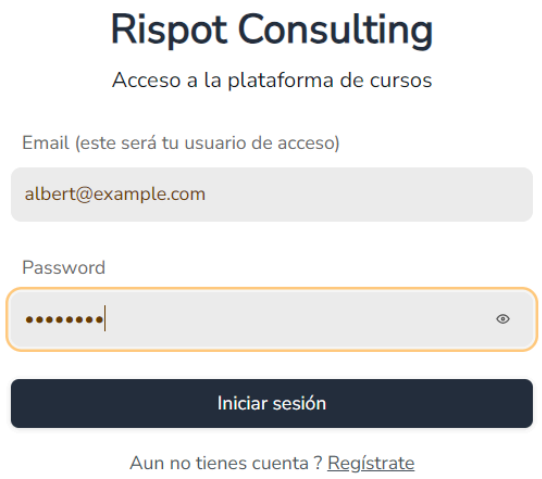

Fig. 65: Formulario login
Fuente: Elaboración propia.

Para conectar eficientemente las tablas de cursos y usuarios en la plataforma web de cursos interactivos para Rispot Consulting, Supabase ofrece la capacidad de crear funciones que facilitan estas operaciones. Por ejemplo, en el *sprint* de *backend* se implementó la función "obtener_cursos_usuario()" para obtener los cursos asociados a un usuario específico. Otras funciones como "calcular_progreso_curso()" o "insertar_elemento_curso_usuario()" también permiten gestionar interacciones entre usuarios y cursos.

A continuación se presenta un ejemplo del uso de la función para obtener cursos pertenecientes a un usuario:

```
const fetchCursos = async () => {
  try {
    const { data: cursosData, error } = await supabase.rpc(
      "obtener_cursos_usuario",
      {
        userid,
      }
    );
```

Fig. 66: Fragmento de código para utilizar la función “obtener_cursos_usuario”
Fuente: Elaboración propia.

Llegados a este punto, ya pude desarrollar el panel de usuario al completo. Con la información de los cursos, de los usuarios y las funciones RPC que me permitían conectar ambas, implementé el código necesario para unir el prototipo maquetado en el *sprint* 2 con el contenido del curso totalmente dinámico y ligado al usuario y su evolución.

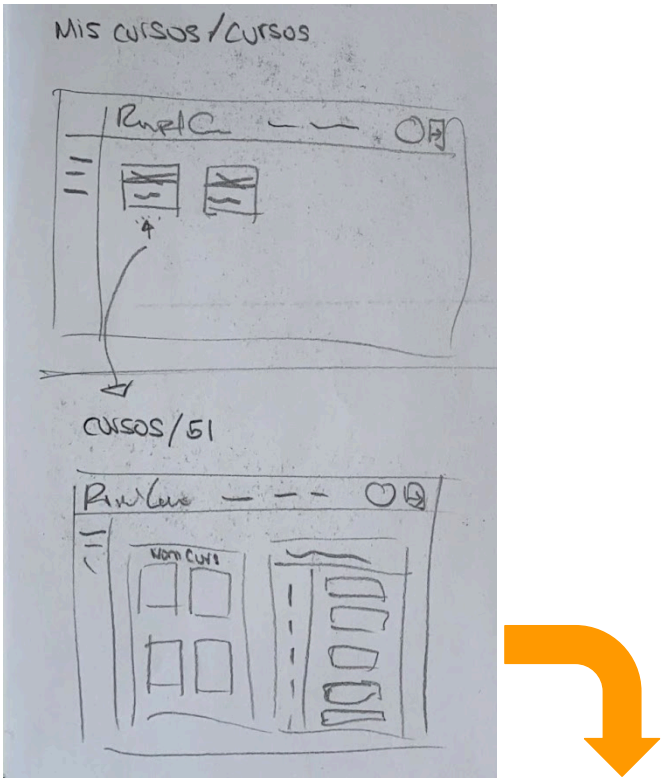

Fig. 67: Bocetos pantallas de inicio *user*
Fuente: Elaboración propia.

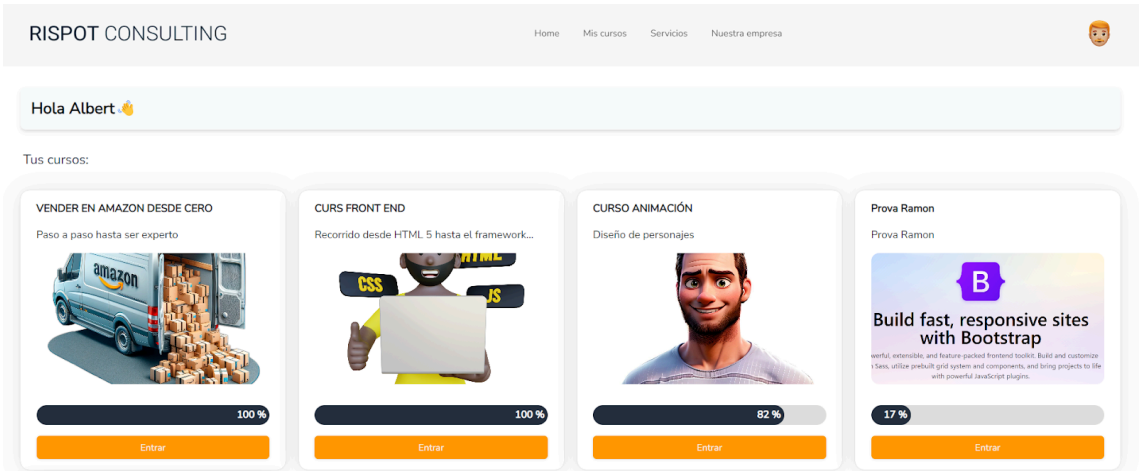

Fig. 68: Vista user pantalla "mis-cursos"
Fuente: Elaboración propia.

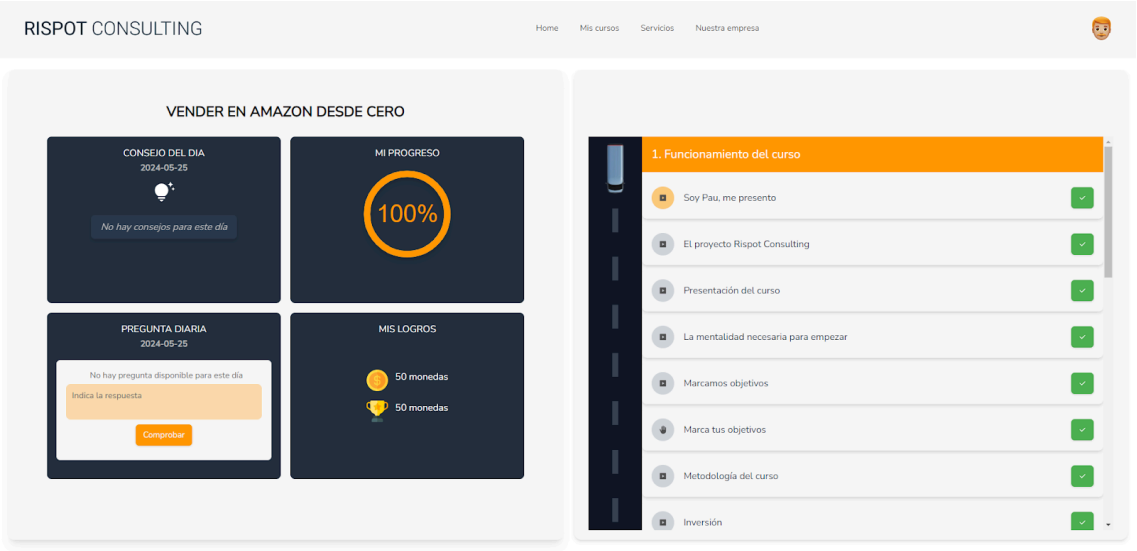

Fig. 69: Vista general *user* curso "Vender en Amazon desde cero"

Fuente: Elaboración propia.

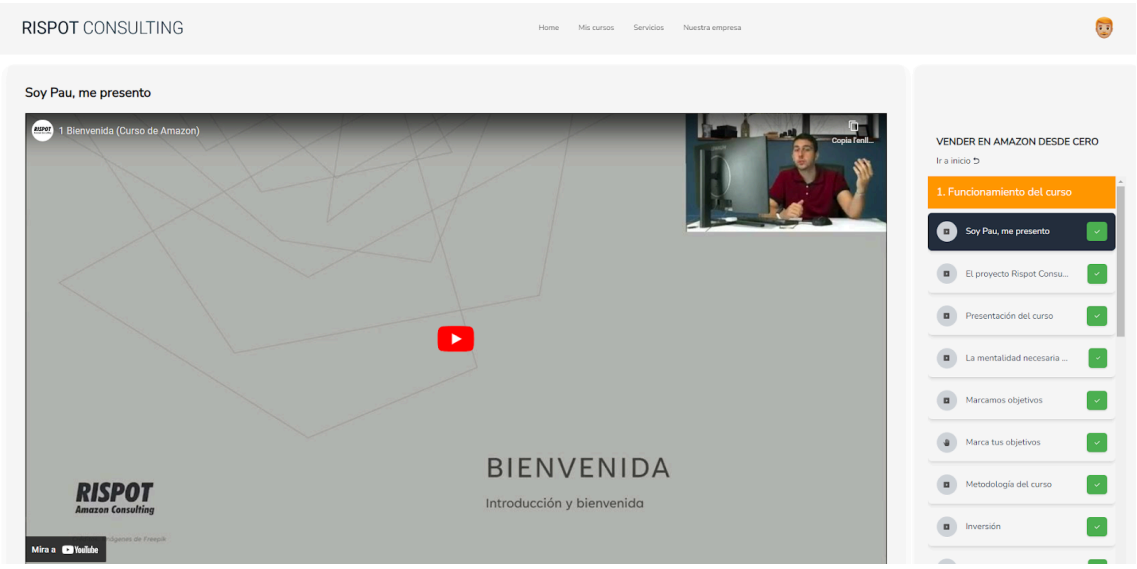

Fig. 70: Primera lección del curso "Vender en Amazon desde Cero"

Fuente: Elaboración propia.

El *sprint* 5 terminaba con los objetivos y casos de uso de la versión uno cubiertos con éxito pero con un serio desajuste respecto a la planificación temporal del proyecto.

Sprint 6: Roles, sesiones, seguridad

Llegando al final del desarrollo de las funcionalidades tanto para el administrador como para el usuario, el siguiente paso consistía en asociar cada uno de estos roles a los usuarios para comenzar a establecer permisos. El servicio de *backend* utilizado es capaz de almacenar la sesión del usuario junto con su información, como el rol asignado, desde el momento en que inicia sesión con su contraseña en el formulario de login hasta que decide cerrar sesión. Esta capacidad me permitió definir una serie de permisos basados en roles para proteger adecuadamente el proyecto.

En términos de seguridad, el proyecto se estructura en dos niveles distintos: un nivel de seguridad interna implementado dentro de Supabase y un nivel dirigido hacia los usuarios que acceden al sitio desde un navegador.

El primer nivel se basa en los permisos y roles ya mencionados. Hay 3 posibilidades según el tipo de usuario que seas:

1. Soy un usuario sin sesión o sin cuenta y quiero entrar a la página inicial y quiero informarme de la plataforma, ver qué catálogo de cursos hay y registrarse o iniciar sesión.

   En este caso no hay seguridad, los únicos componentes a los que puedes acceder son los que no estén rodeados por un "Private Route" (Inicio, *Home*, *Login* y Registro)

```
{/* Rutas públicas */}
<Route element={<Inicio />} path=""></Route>
<Route path="/home" element={<UserHomePage />} />
<Route path="login" element={<LoginPage />} />
<Route path="register" element={<RegisterPage />} />
```

   Fig. 71: Fragmento código rutas públicas
   Fuente: Elaboración propia.

2. Soy un usuario con sesión iniciada y quiero acceder a ver tanto el catálogo de cursos como el contenido de los cursos a los que estoy registrado.

En este caso puedes acceder a los componentes sin seguridad y a los que estén rodeados con un “Private Route” pero sólo si contempla el rol “usuario”.

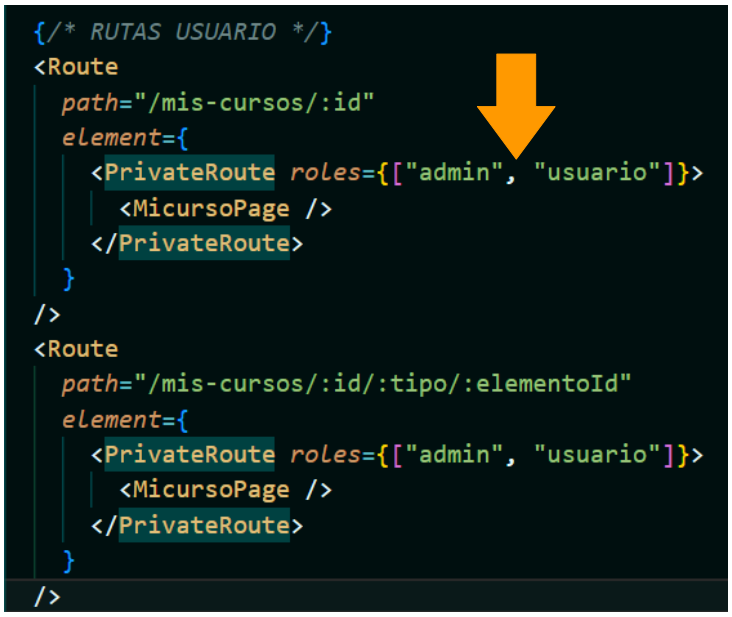

Fig. 72: Fragmento código rutas de *usuario*
Fuente: Elaboración propia.

3. Soy usuario administrador, quiero crear cursos nuevos y añadir o cambiar contenido de ellos, también quiero poder gestionar usuarios.

   En este caso tienes acceso a todas las rutas de la aplicación. Tus rutas están rodeadas de un “Private Route” que únicamente contemplan rol administrador para que solo tú puedas entrar. Ejemplos son la página para crear y editar un material o un examen:

```
<Route
  path=":id/modulo/:moduleId/material/:elementoId"
  element={
    <PrivateRoute roles={["admin"]}>
      <MaterialPage />
    </PrivateRoute>
  }
/>
<Route
  path=":id/modulo/:moduleId/examen/:elementoId"
  element={
    <PrivateRoute roles={["admin"]}>
      <ExamenPage />
    </PrivateRoute>
  }
/>
<Route
```

Fig. 73: Fragmento código rutas de *admin*
Fuente: Elaboración propia.

La segunda capa, o capa interna, la implementé usando la infraestructura que el propio servicio de supabase ofrecía, asegurando el acceso a las tablas con los roles que supabase ya conocía de otras tablas. De esta manera, si alguien conseguía superar la metafórica baya mencionada en apartados anteriores, existiría una segunda capa que aunque el usuario entrase a la habitación, *supabase* sabría que esa persona no tiene permisos y vaciaría todas las habitaciones, es decir, podría ver la página pero sin ningún tipo de información dinámica en ellas. Para hacer esto, creé políticas de nivel de seguridad en las tablas como la del ejemplo.

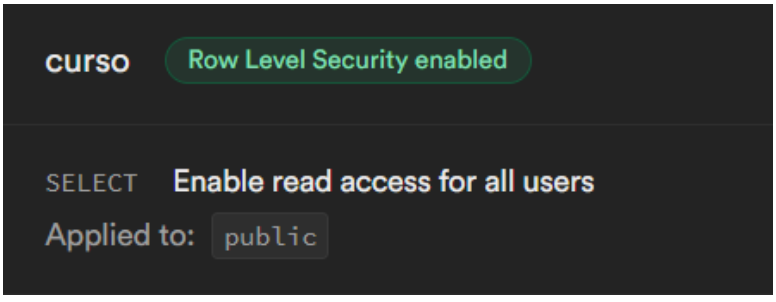

Fig. 74: Política de privacidad de tabla curso
Fuente: supabase.com (20/05/2024).

En este caso se muestra la política: “enable read access for all users” asociada a la tabla curso, esta política está diseñada para que todos los usuarios puedan leer información de la tabla pero permita operaciones de tipo inserción, actualizar o eliminación de registros

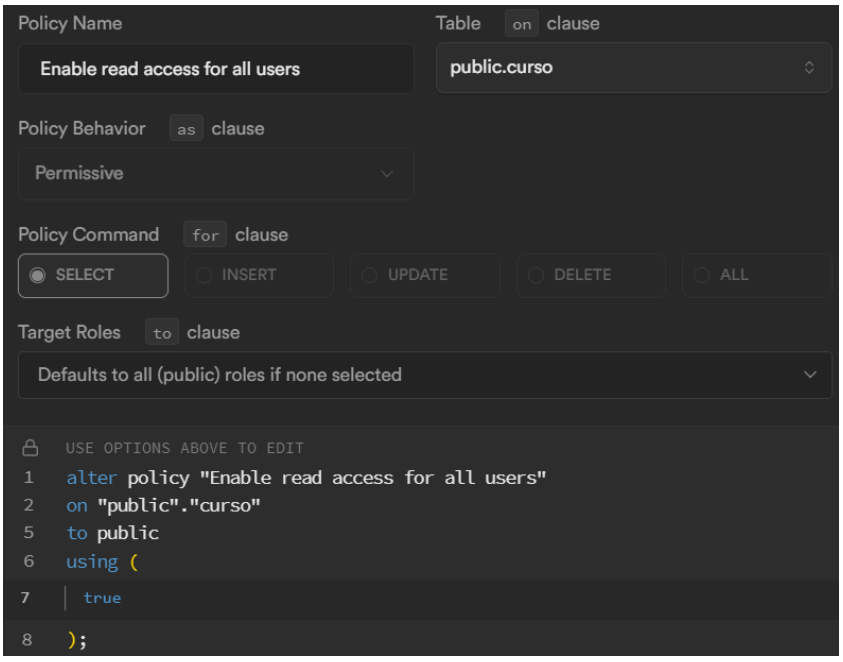

Fig. 75: Política de permiso tabla curso Supabase
Fuente: supabase.com (20/05/2024).

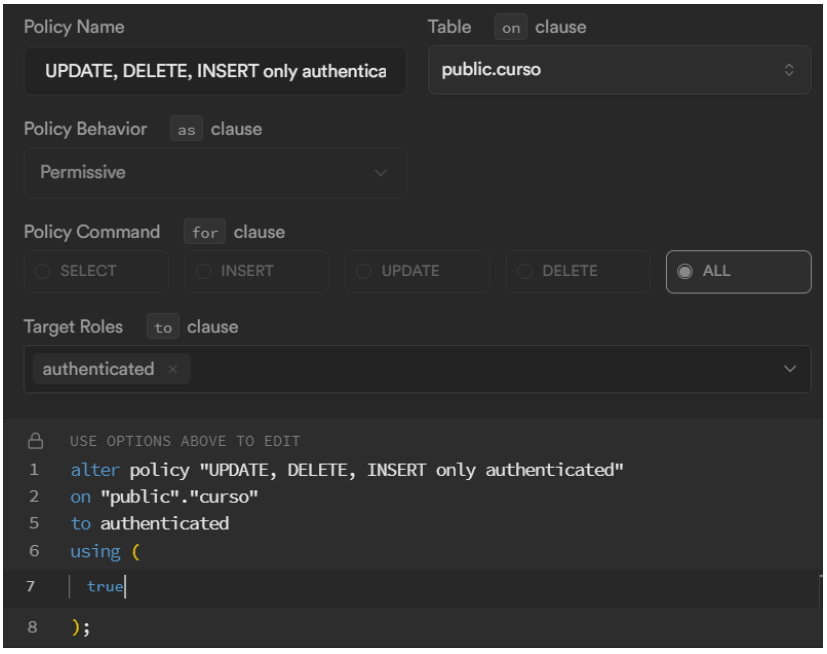

Fig. 76: Politica de permiso Supabase
Fuente: supabase.com (20/05/2024).

Al combinar ambas políticas, si el primer nivel de seguridad no funciona, y algún usuario sin permiso intenta modificar la base de datos, actúa esta capa saltando un error interno y avisa de que no tienes el rol necesario para hacer dicha operación.

Sprint 7: Pruebas y análisis de usabilidad

Con toda la aplicación completada y el *sprint* 6 finalizado, la siguiente fase consistió en realizar las últimas pruebas, tests de usuario y ajustes finales antes de desplegar el proyecto en producción.

Para llevar a cabo esta tarea, utilicé la herramienta User Brain, una aplicación con soporte de extensión en Google que permite realizar tests de usuario en el navegador. Esta herramienta facilita la definición de tareas y el monitoreo del comportamiento del usuario al enfrentarse a dichas tareas. Entre las funcionalidades de User Brain que utilicé se incluyen la grabación de audio durante el test, el seguimiento de los movimientos del ratón y la medición de la duración de cada tarea. El primer paso fue definir las tareas a realizar, y para ello decidí separarlas en tareas específicas para el administrador y tareas específicas para el usuario base.

Tareas test de usuario administrador

- ☐ **Tarea 1**: Inicia sesión con las credenciales: *user*: test / *password*: 12345
- ☐ **Tarea 2**: Crea un curso de nombre : Curso test + "tu nombre"
- ☐ **Tarea 3**: Crea un módulo con el nombre Modulo test + "tu nombre" que contenga 5 videos, un examen y una lección "acciona"
- ☐ **Tarea 4**: Sube un video en alguna de las 5 lecciones creadas y coloca ese vídeo en la última posición del módulo.
- ☐ **Tarea 5**: Añade una actividad de tipo quiz al módulo y crea una pregunta con respuestas (libre).
- ☐ **Tarea 6**: Accede a la tabla de solicitudes para ver si hay nuevos usuarios con interés en el curso.
- ☐ **Tarea 7**: Modifica tu perfil agregando o cambiando alguno de los campos.
- ☐ **Tarea 8**: Elimina el curso creado
- ☐ **Tarea 9**: Cierra tu sesión

Tareas test de usuario base

- ☐ **Tarea 1**: Regístrate e inicia sesión
- ☐ **Tarea 2**: Edita tu perfil.
- ☐ **Tarea 3**: Solicita un curso
- ☐ **Tarea 4**: Accede a él (cuando te den permiso) y realiza alguno de los exámenes.
- ☐ **Tarea 5**: Marca el examen como terminado.
- ☐ **Tarea 6**: Completa la pregunta diaria
- ☐ **Tarea 7**: Cierra sesión

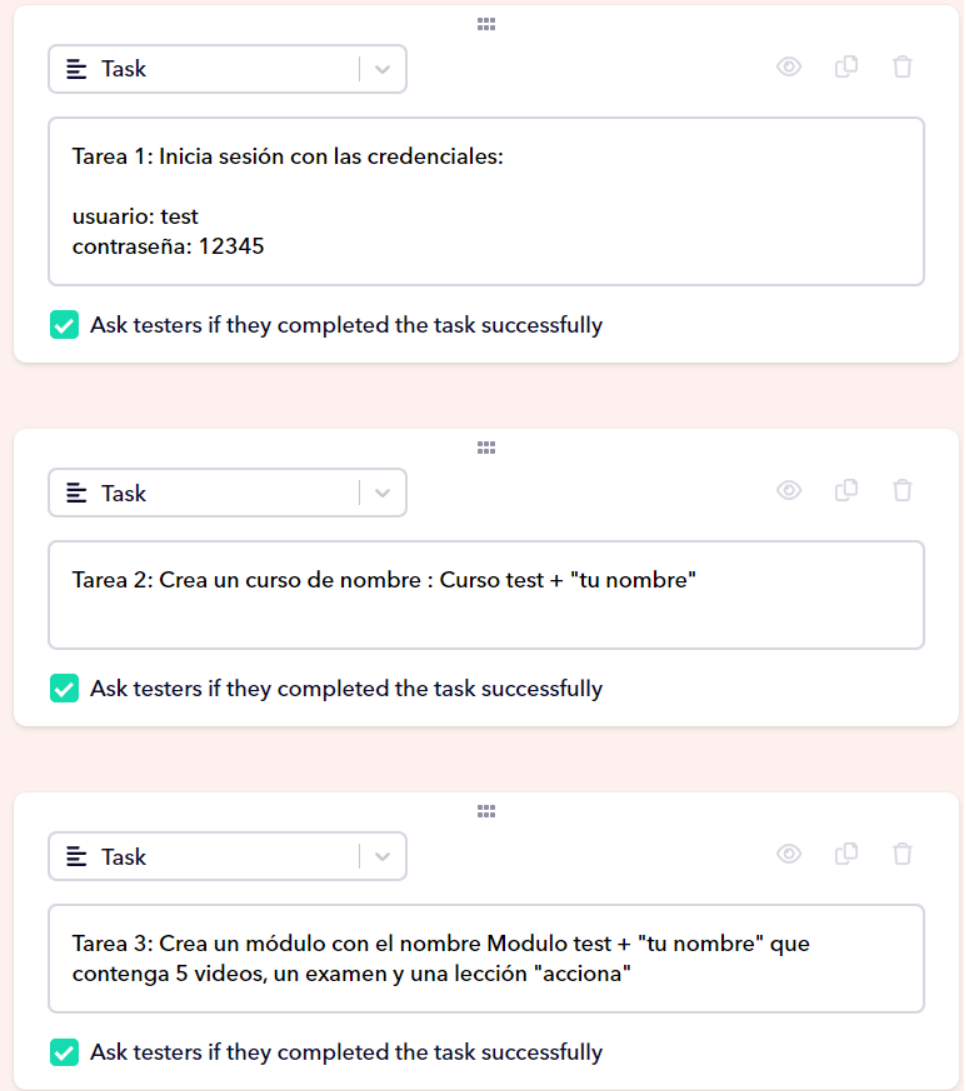

Fig. 77: Interfaz creación de tareas Userbrain

Fuente: userbrain.com (20/05/2024).

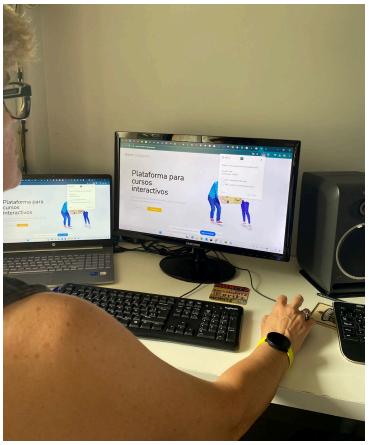

Fig. 78: Test usuario Userbrain *(tarea 1)*

Fuente: Elaboración propia.

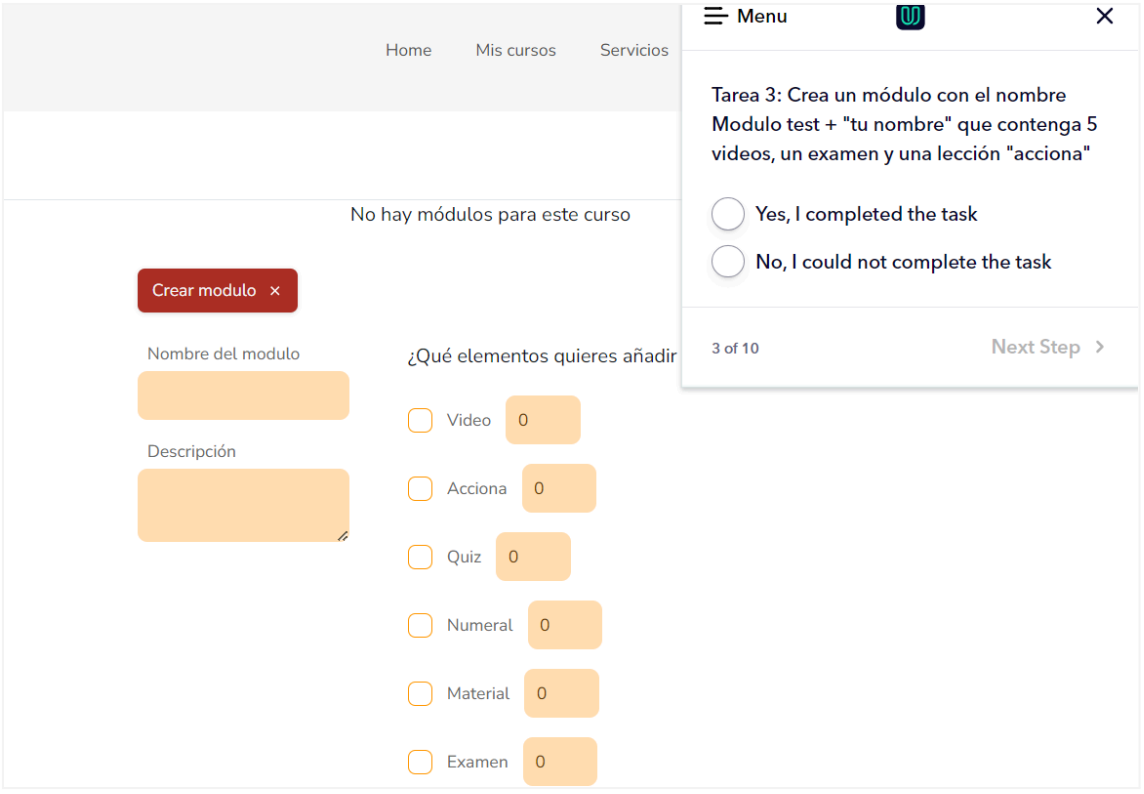

Fig. 79: Tarea 3 test usuario (admin)
Fuente: Elaboración propia.

Los *tests* fueron muy importantes para ver cómo una persona externa entendía la aplicación, sobre todo la parte de usuario que será la más frecuentada. Una vez ajustados los detalles obtenidos de los tests realizados con user brain "subí" a producción la plataforma. Pero, ¿qué es "subir" a producción?

Sprint 8: Despliegue en producción

Durante todo el proyecto se ha mencionado el entorno de desarrollo, configurado mediante Vite y NPM, como se detalló en apartados anteriores. El proyecto sólo tenía validez dentro de dicho entorno y en el propio ordenador. Por ello, al llegar a la fase final y con las pruebas ya realizadas, el siguiente paso consistió en cambiar este entorno de desarrollo por uno de producción.

Según un artículo de la revista de ESIC Business & Marketing School, "subir" a producción, o también conocido como despliegue de una aplicación web, "es la acción de presentar o hacer pública por primera vez una aplicación o página web a través de determinadas herramientas.

Este término proviene del inglés con la palabra *deploy*" (ESIC Business & Marketing School, 2023).

La herramienta que utilicé para este paso fue Vercel, "una plataforma para que los desarrolladores de *frontend* construyan, desplieguen y compartan sus proyectos con el mundo. Está diseñada para facilitar y acelerar el paso del concepto a la producción" (Aplyca, 2023). El proceso resultó muy cómodo y rápido gracias a que Vercel soporta Github, permitiendo enlazar ambas aplicaciones y simplemente desplegar el repositorio donde se encontraban los últimos cambios con la aplicación ya prácticamente definitiva. Finalmente, la plataforma se publicó con el dominio temporal: *"courses-platform-tfg.vercel.app"*.

Sprint 9: Validación y pruebas

Esta fase del proyecto, aunque a menudo pasada por alto, es crucial. El trabajo no termina con el despliegue. Frecuentemente, surgen problemas como archivos de configuración que no se "suben" correctamente, mensajes de depuración visibles solo en el entorno de desarrollo y otros inconvenientes similares, problemas con los que yo también me encontré. Finalmente, procedí a solucionar todos esos fallos y desajustes para asegurar el correcto funcionamiento y la usabilidad tanto para el usuario básico como para el administrador, logrando así el lanzamiento satisfactorio de la primera versión de la aplicación.

7. Valoración de los resultados y futuras ampliaciones

El proyecto ha culminado con el lanzamiento exitoso de la primera versión de la plataforma de cursos interactivos para Rispot Consulting. Esta versión ofrece una experiencia de usuario funcional y fluida tanto para administradores como para usuarios finales. Aunque el resultado es mejorable en algunos aspectos, y el código puede optimizarse, la plataforma cumple con la necesidad inicial, permitiendo la creación y gestión de cursos interactivos de manera totalmente funcional.

7.1 Objetivos y funcionalidades implementadas

A continuación, se señalan las dos grandes caras o ejes sobre los que gira esta aplicación: la experiencia de usuario y la gestión de contenidos. Estos aspectos desde un punto de vista general son los que conforman esta plataforma.

Experiencia de usuario:

La plataforma ha sido diseñada para proporcionar una interfaz intuitiva y fácil de usar tanto para administradores como para usuarios finales. Los principales objetivos en cuanto a la experiencia de usuario incluían:

- **Interfaz intuitiva:** La interfaz de usuario ha sido desarrollada con un enfoque en la simplicidad y la usabilidad, facilitando la navegación y el acceso a las diversas funcionalidades de la plataforma.
- **Funcionalidades básicas operativas:** Las funciones esenciales como la creación, edición y eliminación de cursos, módulos y lecciones están completamente operativas, permitiendo a los usuarios finales y administradores gestionar el contenido de manera eficiente.

Gestión de contenidos:

La plataforma también incluye un robusto sistema de gestión de contenidos, esencial para administradores y creadores de cursos. Las funcionalidades clave en esta área son:

- **Gestión de usuarios:** Los administradores tienen la capacidad de gestionar usuarios, asignar roles y permisos, lo que permite una administración granular y eficiente.
- **Subida de contenido multimedia:** Los administradores pueden subir y gestionar diversos tipos de contenido multimedia, enriqueciendo así los cursos y mejorando la experiencia de aprendizaje.
- **Integración con Supabase:** La plataforma está integrada con Supabase, lo que permite un control eficiente de bases de datos y almacenamiento de archivos. Esta integración asegura una gestión de datos segura y rápida, optimizando el rendimiento de la plataforma.

7.2 Conocimientos adquiridos

Durante el desarrollo de esta aplicación, he adquirido una variedad de conocimientos y habilidades técnicas que han sido fundamentales para el éxito del proyecto. Estos aprendizajes no solo han enriquecido mi experiencia profesional, sino que también han ampliado significativamente mi conjunto de habilidades en el ámbito del desarrollo web y la gestión de proyectos.

Desarrollo *frontend*

- **React**: He adquirido una base sólida en React, explorando conceptos nuevos para mí como los *hooks* y la componentización. Esto me permitió desarrollar una interfaz de usuario dinámica y eficiente para la plataforma de cursos interactivos.
- **Tailwind CSS**: He ampliado mis bases o de Tailwind CSS para el diseño y la estilización de componentes, aprovechando su enfoque basado en clases y su capacidad para crear interfaces *responsive* de manera efectiva.
- **JavaScript ES6**: He profundizado en la sintaxis de JavaScript moderno, utilizando características como *arrow functions*, *destructuring arrays*, y *async/await* para escribir código más limpio y funcional.

Metodologías y herramientas

- **Metodologías ágiles**: He implementado metodologías ágiles, como Scrum, para gestionar el desarrollo iterativo del proyecto, facilitando la colaboración efectiva, la priorización de tareas.
- **GitHub Projects**: He usado GitHub Projects como herramienta de gestión de tareas y seguimiento del progreso del proyecto, he podido familiarizarme con la metodología Scrumban y el uso más completo de la interfaz en lo que a planificación se refiere.

Backend y gestión de datos

- **Supabase**: He interado Supabase como una plataforma *backend-as-a-service* para la gestión eficiente de bases de datos. He podido entender cómo funciona este tipo de infraestructura y me ha servido para familiarizarme con un stack cada vez más utilizado.

Librerías y herramientas adicionales

- **Librería Pol-UI**: He conocido una nueva librería de componentes muy bien documentada, ha sido un gran descubrimiento para desarrollos *frontend* posteriores.
- **Userbrain**: He conocido la herramienta Userbrain para realizar pruebas de usuario y recopilar *feedback* directo de manera online y muy efectiva. Muy intuitiva y útil para futuros proyectos.

Estos conocimientos no solo fueron aplicados durante el desarrollo de la aplicación, sino que también representan un punto de partida sólido para seguir explorando y aprendiendo en el campo del desarrollo web. Cada herramienta y tecnología mencionada ha contribuido de manera significativa al éxito del proyecto, proporcionando soluciones efectivas y optimizando el proceso de desarrollo en general.

7.3 Aspectos mejorables

Desde una reflexión crítica a partir del desarrollo de este TFG, quisiera destacar que, aunque la primera versión de la plataforma es funcional, hay varios aspectos que podrían beneficiarse de mejoras principalmente relacionadas con la optimización del código y la planificación y gestión del proyecto como se especifica a continuación:

Optimización del código

- **Rendimiento y mantenibilidad:** Aunque el código actual cumple con su propósito, existen áreas donde se puede optimizar para mejorar el rendimiento general y la mantenibilidad del sistema. Esto incluye la refactorización de ciertos módulos para hacerlos más eficientes y fáciles de mantener.
- **Escalabilidad:** Con futuras expansiones en mente, es crucial que el código esté optimizado para soportar un mayor volumen de usuarios y datos sin comprometer el rendimiento.

Planificación y gestión del proyecto

- **Conocimiento y recursos:** La planificación del proyecto enfrentó desafíos, principalmente debido a la falta de conocimiento en ciertas áreas, lo que provocó desajustes temporales. Estos desajustes han afectado la eficiencia en algunas partes del trabajo, sugiriendo la necesidad de una mejor gestión del tiempo y los recursos en futuros proyectos.
- **Gestión del tiempo:** Para futuras iteraciones, es esencial mejorar la gestión del tiempo y establecer cronogramas más realistas que consideren posibles contingencias y la curva de aprendizaje asociada con nuevas tecnologías.

Futuras ampliaciones

El resultado del presente trabajo responde a una primera versión funcional, pero existen varias áreas identificadas para futuras mejoras y ampliaciones:

- **Mejora del rendimiento interno:** Se planifica una revisión y optimización del código existente para mejorar el rendimiento y asegurar que la plataforma pueda manejar un mayor número de usuarios y cursos sin problemas.
- **Gamificación:** Se contempla la incorporación de elementos gamificables como *ránkings*, logros y premios. Estos elementos no solo aumentarán la motivación y el compromiso de los usuarios, sino que también pueden mejorar significativamente la experiencia de aprendizaje y diferenciarse de la competencia.
- **Características avanzadas:** En futuras versiones, se planea añadir características avanzadas como analíticas de aprendizaje, con gráficas de evolución e integraciones con otras plataformas educativas y herramientas de colaboración para enriquecer aún más la oferta educativa.

8. Conclusiones y valoración personal

Este proyecto ha representado una fase muy importante en mi formación como desarrollador web, comenzando como una idea remota hace más de siete meses y culminando con el exitoso lanzamiento de la primera versión de la plataforma de cursos interactivos para Rispot Consulting. Durante este tiempo, he podido interiorizar y aplicar ampliamente los conocimientos adquiridos a lo largo de mis cuatro años de formación universitaria.

Me gustaría destacar sobre todo la transversalidad y la colaboración fundamental que han marcado este proceso. Trabajar codo a codo con mi compañero Pol Gubau ha sido crucial; hemos podido retroalimentarnos constantemente, lo que ha enriquecido cada aspecto del proyecto. Además, la estrecha colaboración con Pau Cardús, fundador de Rispot Consulting, ha proporcionado una perspectiva empresarial valiosa y ha facilitado la alineación del desarrollo técnico con las necesidades del cliente.

El presente TFG ha resultado ser una valiosa oportunidad para ampliar mis habilidades en tecnologías esenciales como React, las cuales he tenido que aprender desde cero para cumplir con los requisitos del proyecto. Han sido más de 760 intensas y gratificantes horas dedicadas simultáneamente al desarrollo y a la autoformación. Esta experiencia no solo ha mejorado significativamente mis habilidades en programación y diseño, sino que también me ha permitido abordar proyectos complejos con mayor confianza y eficiencia. Hacer un trabajo de estas características de manera individual ha supuesto un reto personal mayúsculo, no solo por todo el proceso de formación y familiarización, sino, sobre todo por la planificación del proyecto, y el cumplimiento de todos los plazos. El valor, en mí opinión, no radica únicamente en el resultado final de la plataforma, ya que, a comparación con proyectos similares del sector, sigue siendo una versión muy primaria. El verdadero valor que ha aportado en mi formación académica y personal es el recorrido que he tenido que seguir, acercándome a las metodologías, herramientas y tecnologías más punteras en el sector del desarrollo web actualmente.

Para concluir, este proyecto no solo ha sido una prueba de mis habilidades técnicas y de gestión, sino también una experiencia de crecimiento personal y profesional inestimable. He tenido la oportunidad de enfrentar y superar desafíos complejos, aprender nuevas tecnologías y metodologías, y colaborar con profesionales talentosos. La plataforma desarrollada para Rispot Consulting es un testimonio de la dedicación, el esfuerzo y la pasión invertidos en cada etapa del proceso. Aunque este es solo el comienzo y queda mucho por mejorar y expandir, estoy seguro de que las lecciones aprendidas y las habilidades adquiridas en este proyecto serán fundamentales para mi futura carrera como desarrollador web. Agradezco profundamente a todos los que han contribuido a este logro y espero con entusiasmo los próximos desafíos y oportunidades que se presenten.

9. Referencias bibliográficas

Andrade, T. (2023, 5 junio). React y Tailwind para desarrollo web. Listopro Community. https://community.listopro.com/react-y-tailwind-para-desarrollo-web/ [Consultado el 24 de marzo de 2024]

Aplyca. (2023, 26 enero). Vercel: desarrollar, previsualizar, enviar - *Aplyca Tecnología SAS*. https://www.aplyca.com/blog/blog-que-es-vercel-desarrollar-previsualizar-enviar [Consultado el 21 de abril de 2024]

Aristovnik, A., Karampelas, K., Umek, L., & Ravšelj, D. (2023). Impact of the COVID-19 pandemic on online learning in higher education: A bibliometric analysis. *Frontiers In Education, 8*. https://doi.org/10.3389/feduc.2023.1225834 [Consultado el 17 de marzo de 2024]

AltexSoft. (2023, 30 diciembre). Nonfunctional Requirements in Software Engineering: Examples, Types, Best Practices. *AltexSoft*. https://www.altexsoft.com/blog/non-functional-requirements/ [Consultado el 01 de abril de 2024]

Ayebola, J. (2023, 4 diciembre). How to Use React Hooks – useEffect, useState, and useContext Code Examples. freeCodeCamp.org. https://www.freecodecamp.org/news/react-hooks-useeffect-usestate-and-usecontext/ [Consultado el 03 de abril de 2024]

B, Gustavo (2023, 10 enero). Qué es GitHub y cómo empezar a usarlo. *Tutoriales Hostinger*. https://www.hostinger.es/tutoriales/que-es-github [Consultado el 01 de abril de 2024]

CodeX. (2023, 27 noviembre). Full Stack Web Development with Python and Django (Part 7/7) [Vídeo]. *YouTube*. https://www.youtube.com/watch?v=jQoTvL0-ndw [Consultado el 17 de marzo de 2024]

Colman, H. (2023, 27 noviembre). E-learning: Qué es y cómo funciona, beneficios del e-learning. Blog de E-learning. https://www.ispring.es/blog/what-is-elearning [Consultado el 23 de abril de 2024]

Coppola. M. (2023, 12 julio). ¿Qué es Javascript y para qué sirve? *Hubspot*. https://blog.hubspot.es/website/que-es-javascript [Consultado el 25 de marzo de 2024]

Cortés P, Y. (2017, 5 noviembre). Qué es Stack MEAN: desarrollo full-stack en JavaScript. Platzi. https://platzi.com/blog/que-es-mean-full-stack-javascript/ [Consultado el 17 de marzo de 2024]

Deshpande, C. (2023, 5 octubre). The Best Guide to Know What Is React. *Simplilearn.com*. https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs [Consultado el 02 de abril de 2024]

Devink. (2024, 8 enero). Material UI y Next UI: Catálogos de Componentes para Diseño Web Moderno. https://www.linkedin.com/pulse/material-ui-y-next-cat%C3%A1logos-de-componentes-para-dise%C3%B1o-2lmlf/ [Consultado el 18 de abril de 2024]

EDteam - ¿Cómo es la sintaxis de React.js? (s. f.-c). En Español Nadie Te Explica Mejor. https://ed.team/comunidad/como-es-la-sintaxis-de-react-js [Consultado el 2 de abril de 2024]

ESIC Business & Marketing School. (2023, julio). Despliegue de aplicaciones web: en qué consiste. *ESIC*. https://www.esic.edu/rethink/tecnologia/despliegue-aplicaciones-web-c [Consultado el 21 de abril de 2024]

Userbrain - Fast, simple, affordable user testing. (s. f.). https://userbrain.com/ [Consultado el 23 de mayo de 2024]

Garcia, M. (2021, 31 diciembre). React — React Router - Mauricio Garcia - Medium. *Medium*. https://mauriciogc.medium.com/react-react-router-db391b3def2a [Consultado el 24 de marzo de 2024]

GitHub: Let's build from here. (2024). GitHub. https://github.com/

Gómez, P. (2023, 31 mayo). Historia de la programación: ¿qué es y cómo ha evolucionado con los años? - DevCamp. *DevCamp*. https://devcamp.es/historia-de-la-programacion-que-es-y-como-ha-evolucionado-con-los-anos/ [Consultado el 01 de mayo de 2024]

González, C. (2024, 15 abril). Descubre las ventajas de PostgreSQL. *LinkedIn*. https://www.linkedin.com/pulse/descubre-las-ventajas-de-postgresql-c%C3%A9sar-gonz%C3%A1lez-yuoue/ [Consultado el 25 de marzo de 2024]

GooApps. (2022, 27 octubre). Las 5 mejores metodologías de desarrollo de software. *GooApps®*. https://gooapps.es/2022/10/27/las-5-mejores-metodologias-de-desarrollo-de-software/ [Consultado el 23 de abril de 2014]

Gubau, P (s. f.). *Pol-ui - A beautiful and functional UI library for React and Next.js*. https://ui.polgubau.com/ [Consultado el 18 de abril de 2024]

Haider, R. (2021, 8 julio). What is Jamstack and why should you be using it? *CloudBytes.dev*. https://cloudbytes.dev/snippets/what-is-jamstack-and-why-should-you-be-using-it [Consultado el 17 de marzo de 2024]

Hernández, J. (2022, 30 abril). Qué es Vite y cómo funciona más rápido que webpack - Jaime Hernández - Medium. *Medium*. https://devjaime.medium.com/que-es-vite-y-como-funciona-m%C3%A1s-rapido-que-webpack-132f45efa4e4 [Consultado el 21 de abril de 2024]

Hoyos, S. (2018, 8 junio). ¿Qué es JSX? - Simon Hoyos - Medium. *Medium*. https://medium.com/@simonhoyos/qu%C3%A9-es-jsx-95006a2f94f9 [Consultado el 01 de mayo de 2024]

Huet, P. (2023, 18 abril). React Hooks: Qué son y qué problemas solucionan. *OpenWebinars.net*. https://openwebinars.net/blog/react-hooks-que-son-y-que-problemas-solucionan/ [Consultado el 03 de abril de 2024]

Hurtado, J. S. (2023, 15 febrero). Metodología Scrum: Qué es y cómo utilizarla para acometer proyectos. *Thinking For Innovation*. https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/ [Consultado el 23 de abril de 2014]

Inába, S. (2023, 17 abril). Stacks tecnológicos populares. *Soluciones Inába*. https://www.inabaweb.com/stacks-tecnologicos-populares/ [Consultado el 22 de abril de 2024]

Kinsta. (2022, 19 diciembre). ¿Qué es React.js? Un Vistazo a la Popular Biblioteca de JavaScript. *Kinsta®*. https://kinsta.com/es/base-de-conocimiento/que-es-react-js/ [Consultado el 03 de abril de 2024]

Kyle, P. (2023, 28 septiembre). What is a LAMP Stack? *Liquid Web*. https://www.liquidweb.com/kb/what-is-a-lamp-stack/ [Consultado el 25 de marzo de 2024]

Laoyan, S. (2024, 14 febrero). Scrumban: lo mejor de dos metodologías ágiles [2024] • Asana. *Asana*. https://asana.com/es/resources/scrumban [Consultado el 25 de marzo de 2024]

Lata, P. (2023, 14 abril). What is the Ruby on Rails stack? *rail-way-men*. https://blog.railwaymen.org/ruby-on-rails-stack [Consultado el 21 de abril de 2024]

Lobo, D. (2023, 3 abril). Django, la herramienta Full Stack para desarrollar aplicaciones web rápidas. *International Business School*. https://eiposgrados.com/blog-desarrollo-web-fullstack/django-herramienta-full-stack-para-desarrollar-aplicaciones-web-rapidas/ [Consultado el 24 de marzo de 2024]

Loreto, A. M. (2024, 15 enero). 10 razones para aprender JavaScript. *Hack a boss*. https://www.hackaboss.com/blog/10-razones-para-aprender-javascript [Consultado el 01 de mayo de 2024]

Medina, D. (2021, 19 octubre). "Las empresas que apuestan por metodologías ágiles gestionan sus proyectos de forma más flexible y eficaz". *UNIR*. https://www.unir.net/ingenieria/revista/metodologias-agiles-daniel-medina/ [Consultado en 23 de abril de 2014]

Murillo, F. (2022, 6 enero). Supabase: Una alternativa a Firebase con el poder de SQL. *Medium*. https://fabricioism.medium.com/supabase-una-alternativa-a-firebase-con-el-poder-de-sql-d94f9e804ec3 [Consultado el 24 de marzo de 2024]

Mijacobs. (2023, 5 octubre). ¿Qué es Git? - Azure DevOps. *Microsoft Learn*. https://learn.microsoft.com/es-es/devops/develop/git/what-is-git [Consultado el 21 de abril de 2024]

Nguyen, D. (2023, 28 junio). How advanced technologies are transforming education | *MIT Open Learning*. https://openlearning.mit.edu/news/how-advanced-technologies-are-transforming-education [Consultado el 17 de marzo de 2024]

Stack overflow. (s. f.-b). https://survey.stackoverflow.co/ [Consultado el 1 de abril de 2024]

Supabase | the open source Firebase alternative. (s. f.). Supabase. https://supabase.com/

O'Grady, B. (2023, 23 octubre). What is JavaScript and What is it Used for? - *Code Institute IE. Code Institute IE.*

https://codeinstitute.net/global/blog/what-is-javascript-and-why-should-i-learn-it/ [Consultado el 15 de abril de 2024]

Parada, M. (2023, 14 abril). MERN Stack: Qué es y qué ventajas ofrece. *OpenWebinars.net*. https://openwebinars.net/blog/mern-stack-que-es-y-que-ventajas-ofrece/ [Consultado el 18 de abril de 2024]

Roberts, M. (2018, 22 mayo). Serverless architectures. *Martinfowler.com*. https://martinfowler.com/articles/serverless.html [Consultado el 23 de abril de 2014]

Shah, V. (2023, 14 julio). Serverless: An emerging software architecture. *TatvaSoft Blog*. https://www.tatvasoft.com/blog/serverless-an-emerging-software-architecture/ [Consultado el 17 de marzo de 2024]

Solera, J. M. (2020, 4 julio). ¿Qué es MERN Stack? *Javier Martínez Solera*. https://jmsolera.com/que-es-mern/ [Consultado el 01 de mayo de 2024]

Stsepanets, A., & Stsepanets, A. (2024, 26 enero). Qué es y cómo hacer un diagrama de Gantt. *Gantt Chart GanttPRO Blog*. https://blog.ganttpro.com/es/guia-completa-para-los-diagramas-de-gantt/ [Consultado el 22 de abril de 2024]

Valera, J. J. V. (2023, 8 mayo). React Router DOM V6, React.js. *LinkedIn*. https://www.linkedin.com/pulse/react-router-dom-v6-reactjs-junior-javier-duque-valera/ [Consultado el 01 de mayo de 2024]

Vergara, S. (2023, 27 noviembre). ¿Por qué Tailwind CSS es tan popular? Blog ITDO - Agencia de Desarrollo Web, APPs y Marketing En Barcelona. https://www.itdo.com/blog/por-que-tailwind-css-es-tan-popular/ [Consultado el 11 de abril de 2024]

W3Techs (2024, mayo). *W3Techs*. https://w3techs.com/technologies/details/cp-javascript [Consultado el 01 de abril de 2024]

X.com. (s. f.). X (Formerly Twitter). https://twitter.com/alfredoMendoza [Consultado el 11 de abril de 2024]

Zumba Gamboa, J. P., & León Arreaga, C. A. (2018). Evolución de las metodologías y modelos utilizados en el desarrollo de software. *INNOVA Research Journal, 3*(10), 20-33. https://dialnet.unirioja.es/descarga/articulo/6777227.pdf [Consultado el 01 de mayo de 2024]