



## ***Desarrollo full stack***

### ***Proyecto Final***

<b>Nombre:</b> Frida Sarahi Garza Gálvez Pedro Francisco De León Salazar Paola Guadalupe Urdiales Luna Abraham Isaí Garza Sánchez Brandon Alan Carrión Morales Carolina Monserrat Hermosillo Castañeda	<b>Matricula:</b> al07002961 al03103352 al03107338 al03109910 al03108850 al02938037
<b>Profesor:</b> Erick Bethuel Estrada Andrade	
<b>Fecha:</b> 1 de marzo de 2025	

#### **Descripción**

En esta fase final, los aprendedores deberán expandir y completar la aplicación full stack creada en la primera fase. Implementarán funcionalidades avanzadas en el backend y frontend, añadirán una capa de seguridad utilizando OAuth o JWT, y realizarán el despliegue completo de la aplicación en un entorno de producción en la nube. Finalmente, deberán integrar APIs externas (REST o GraphQL), realizar pruebas y garantizar que la aplicación esté lista para su uso.

#### **Objetivo**

Implementar funcionalidades avanzadas en el backend y frontend, utilizando autenticación y autorización seguras con OAuth o JWT. Integrar una API externa en el proyecto y realizar el despliegue completo de la aplicación en un entorno en la nube. Realizar pruebas que garanticen su correcto funcionamiento y faciliten la depuración de posibles errores.

# Introducción

En nuestra entrega del proyecto final de la materia Desarrollo Full Stack, empezando con nuestra empresa HERMCODE donde buscamos el ofrecer una pagina a los usuarios donde tengan la posibilidad de conocer y ampliar su aprendizaje en areas de promagación, dando asi una pagina con un diseño visualmente agradable junto a una funcionalidad y usabilidad de buen nivel a los usuarios que nos buscamos dirigir. Esperamos en esta entrega poder hacer uso de varios conocimientos abarcados a lo largo de la materia para poder cumplir con nuestro objetivo.

## Reporte final

### Frida Sarahi Garza Gálvez

Cargo: Lead

En la realización del proyecto, nos repartimos el trabajo por cargos, dando así que yo porte el cargo de líder junto al de Dev empeñado en Frontend, como líder mi trabajo consistió en generar un acuerdo con el equipo para tomar la decisión de que trabajo entregado en el avance del proyecto podríamos usar de base, y con ello hecho, se dé comienzo con la creación de los diversos requerimientos en Jira para los cargos como UI/UX, CI/CD, Dev Backend y QA. Cada cargo apoyara con la realización de su diagrama individual junto a un reporte individual donde detallen su cumplimiento de los requerimientos asignados. Siendo que mi cargo de líder se empeñe en la gestión de los entregables y el avance de cada integrante del equipo y por parte del cargo de Dev Frontend, yo doy apoyo en la realización de los HTML y la generación de estilos que se me son dados por UI/UX.

A la par de esto, se busca que el equipo logre generar una buena comunicación de forma presencial en conjunto con online para así poder cumplir en tiempo y forma la entrega del proyecto final del curso presente. *Link de requerimientos de Jira: [Excel de requerimientos de Jira](#)*

### Requerimientos

Los requerimientos que yo maneje en el cargo de Líder fueron los siguientes:

- *Crear Board en Jira:* Se debe crear en Jira el Board donde se deberá monitorear el seguimiento de los requerimientos de cada uno de los puestos.
- *Añadimient o en el Board:* Se asegura que todos los integrantes del equipo estén en el proyecto de Jira, en la cual debieron generar una cuenta para hacer posible el ingreso.
- *Creación de Stories/Requerimientos:* Se son agregados y creados los requerimientos en el Board de una forma en la que todos los compañeros puedan acceder a sus Stories correspondientes.
- *Creación de base para reportes individuales:* Se genera un reporte base para que los integrantes del equipo puedan generar un reporte con el mismo modelo para generar un reporte completo y conciso.
- *Asignación de Issues al rol correspondiente:* Se debe trabajar en la asignación de Issues generado por el QA para asignarlo al rol correspondiente para que se logre la corrección o el manejo necesario.
- *Seguimiento de las Stories/Requerimientos:* Se busca tomar seguimiento de cada uno de los Stories de los integrantes para asegurar un trabajo continuo.

- *Manejo de dudas en los integrantes:* Se está al pendiente de las necesidades o dudas que vayan surgiendo en el equipo a lo largo del proyecto.
- *Creación y gestión del diagrama de actividades:* Se debe generar un diagrama de actividades que se empeñe en abordar cada trabajo o actividad realizada por los integrantes y como se engloban con los demás.
- *Gestionar entregable de reportes y reporte final:* Asegurarse el obtener los reportes individuales de cada uno de los cargos, asimismo, el asegurar la creación del reporte final, el Excel con los requerimientos manejados en Jira, el video de funcionalidad y la entrega final.

Dichos requerimientos fueron tratados a lo largo de la creación de dicho proyecto, siendo unos finalizados en conjunto con los otros roles.

## Realización

Tras mostrar los requerimientos que abarcan mi cargo de Lead ahora, comenzaré a detallar como los lleve a cabo cada uno de ellos. Dando una base sólida y concisa de mi parte de trabajo a lo largo del proyecto final.

### Elección de proyecto

Antes de poder dar comienzo a mis requerimientos, tuvimos que ponernos de acuerdo en la selección de uno de los dos proyectos que manejábamos tras la mezcla de nuevos integrantes al equipo; siendo un equipo creado el 12 de Febrero con seis integrantes. Para la selección del proyecto, nos empeñamos en el tema y el diseño que se manejaba, es por ello, que se dio elección del proyecto de Brandon Carrión y Carolina Hermosillo, ofreciéndonos una empresa empeñada en cursos para programadores, que lleva consigo un tema relevante, diseño atractivo y mejores posibilidades de implementar lo que se nos es requerido.

### Creación Board en Jira

Mi primer requerimiento conlleva la creación del Board en Jira, donde nos empeñamos en manejar un tablero el cual deberá contener todas las Stories que sean creadas. Donde me aseguro del manejo de ciertos datos en diversas secciones en las Stories, manejando diversos datos como el *Titulo del requerimiento, la descripción, a la persona que se le es asignada, las personas que deberán aprobarlo, la fecha en la que se dio comienzo y la fecha en la que finalizo, la Epic donde es asignada, el equipo que se hará cargo junto al creador de dichas Stories.*

Proyectos / Proyecto Final - DFS / KAN-8 / KAN-5

Implementación de autenticación

**Descripción**  
Se debe implementar un JWT a las operaciones CREATE, UPDATE y DELETE de los cursos, donde solo los usuarios con cargo ADMIN puedan hacer uso. Tanto Admin como Regular pueden leer los cursos que se manejan en la página principal.

**Incidencias secundarias**  
KAN-52 Implementar una autenticación al Admin/Regular  
KAN-55 Funciones y permisos para Admin o Regular

**Actividad**  
Todo Comentarios Historial Registro de actividad

Añadir un comentario... ¡Tiene buena pinta! ¿Necesitas ayuda? Esto está bloqueado... ¿Puedes aclararlo...? Esto va por bue...

Consejo de expertos: pulsa M para comentar

Finalizada ✓ Listo

**Detalles**

Persona asignada: PEDRO FRANCISCO DE LEON SALAZAR  
Asignarme a mí

Approvers: Brandon Alan Carrion Mor...  
Frida Sarahi Garza Gálvez

Actual start: 18 feb 2025, 10:00

Actual end: 27 feb 2025, 19:00

Principal: KAN-8 Dev's

Team: DEV

Sprint: Tablero Sprint

Desarrollo: Crear rama, Crear confirmación

Informador: Frida Sarahi Garza Gálvez

Podemos ver que a la par podremos manejar alrededor de varias incidencias secundarias que portaran la misma necesidad de datos para poder llevar un mejor control de como lo estuvimos manejando.

Con esa base de datos en el Board, ahora podemos avanzar al siguiente requerimiento, tras la correcta creación del Board en Jira.

Para acceder al Board en Jira, puede ingresar dando click en el siguiente enlace:

[Link acceso a Jira](#)

### Añadimient o en el Board de Jira

Mi segundo requerimiento conllevo la búsqueda de que todos los integrantes del equipo junto al profesor es que tengan acceso al Board en Jira, donde podrán tener acceso a los requerimientos que se les fueron asignados.

Veamos en la siguiente imagen como es que se fue correctamente posible el acceso de mis compañeros de equipo en el Board en Jira. Podremos observar que hay siete integrantes y uno de ellos es nuestro profesor para que fuera posible el monitoreo del trabajo, así mismo, podremos ver la última vez en la que estuvieron activos en el Board.

The screenshot shows the Jira 'Users' management page. At the top, it displays 'Administrador / proyecto-final-equipo' and 'Usuarios'. Below that, it says 'Gestiona el acceso a productos de todos los usuarios de tu organización.' There are three counts: 'Usuarios en total' (9), 'Usuarios activos' (7), and 'Administradores' (1). A search bar and a 'Filtros (1)' button are present. The main table lists seven users with their names, email addresses, last active date, status, and an 'Acciones' (Actions) column with three dots. The users are: Abraham Isaí Garza Sánchez, Brandon Alan Carrion Morales, Carolina Hermosillo, ERICK BETHUEL ESTRADA ANDRADE, Frida Sarahi Garza Gálvez, PAOLA GUADALUPE URDIALES LUNA, and PEDRO FRANCISCO DE LEÓN SALAZAR.

Usuario	Última vez activo	Estado	Acciones
Abraham Isaí Garza Sánchez ai03109910@tecmilenio.mx	28 feb 2025	Activo	...
Brandon Alan Carrion Morales ai03108850@tecmilenio.mx	28 feb 2025	Activo	...
Carolina Hermosillo ai02938037@tecmilenio.mx	28 feb 2025	Activo	...
ERICK BETHUEL ESTRADA ANDRADE erick_estrada@tecmilenio.mx	25 feb 2025	Activo	...
Frida Sarahi Garza Gálvez ADMINISTRADORES DE LA ORG... ai07002961@tecmilenio.mx	28 feb 2025	Activo	...
PAOLA GUADALUPE URDIALES LUNA ai03107338@tecmilenio.mx	01 mar 2025	Activo	...
PEDRO FRANCISCO DE LEÓN SALAZAR ai03103352@tecmilenio.mx	01 mar 2025	Activo	...

Por último, con los integrantes en el Board, me es posible crear y manejar los equipos basados en los roles que se tienen, y dichos equipos serán de ayuda para la asignación de requerimientos.

The screenshot shows the Jira 'Equipos' (Teams) management page. At the top, it displays 'Personas y equipos / Equipos' and buttons for 'Gestionar usuarios', 'Crear equipo', and 'Añadir personas'. Below that is a search bar and a 'Filtrar por miembros del equipo' (Filter by team members) button. The page displays five teams as cards: 'CI/CD' (with member AS), 'DEV' (with members FG and PS), 'LEAD' (with member FG), 'QA' (with member BM), and 'UI/UX' (with members CH and PL).

Notemos como hay un equipo para cada rol, donde solo dos de dichos equipos portan dos integrantes, siendo el de DEV y UI/UX. Con la definición de los equipos y el añadimient o de los integrantes al Board, ahora podemos empezar con la creación de los requerimientos.

## Creación de Stories/Requerimientos

Para la creación de los requerimientos fue necesario el manejar desde las necesidades del avance del proyecto hasta las necesidades del proyecto final, se crearon alrededor de 110 Stories, las cuales abarcaron el trabajo de cada uno de los roles.

Para las UI/UX, se abarcaron alrededor de 14 requerimientos, dando así un abarque completo de lo que realizaran en el proyecto final. A continuación, una muestra de los requerimientos que fueron asignados:

<input checked="" type="checkbox"/> KAN-114 Reporte individual de Carolina	UI/UX	FINALIZADA ✓	CH
<input checked="" type="checkbox"/> KAN-9 Genera interacciones dinámicas en sus páginas	UI/UX	FINALIZADA ✓	CH
<input checked="" type="checkbox"/> KAN-28 Asegura diseño en Mobil y Web para sus páginas	UI/UX	FINALIZADA ✓	CH
<input checked="" type="checkbox"/> KAN-59 Generación de página RegularCursos	UI/UX	FINALIZADA ✓	PL
<input checked="" type="checkbox"/> KAN-29 Concuerdan con tipo de diseño	UI/UX	FINALIZADA ✓	PL
<input checked="" type="checkbox"/> KAN-7 Trabajan en conjunto en Figma	UI/UX	FINALIZADA ✓	PL
<input checked="" type="checkbox"/> KAN-23 Asegura diseño en Mobil y Web para sus páginas	UI/UX	FINALIZADA ✓	PL
<input checked="" type="checkbox"/> KAN-26 Genera interacciones dinámicas en sus páginas	UI/UX	FINALIZADA ✓	PL
<input checked="" type="checkbox"/> KAN-22 Generación de página Register	UI/UX	FINALIZADA ✓	PL
<input checked="" type="checkbox"/> KAN-21 Generación de página Login	UI/UX	FINALIZADA ✓	PL
<input checked="" type="checkbox"/> KAN-118 Generación de pagina Cursos	UI/UX	FINALIZADA ✓	PL
<input checked="" type="checkbox"/> KAN-112 Especificación de manejo entre paginas	UI/UX	FINALIZADA ✓	PL
<input checked="" type="checkbox"/> KAN-24 Reporte individual de Paola	UI/UX	FINALIZADA ✓	PL
<input checked="" type="checkbox"/> KAN-27 Generación de página AdminCursos	UI/UX	FINALIZADA ✓	CH
<input checked="" type="checkbox"/> KAN-25 Generación de página Principal	UI/UX	FINALIZADA ✓	CH

Para el CI/CD, se abarcaron alrededor de 6 requerimientos, dando así un abarque completo de lo que debería realizar en el proyecto final, dichos requerimientos en su mayoría son abarcados a lo largo del proyecto, solo siendo finalizados en su mayoría a las ultimas horas de la última modificación en el repositorio de GitHub. A continuación, una muestra de los requerimientos que se les fueron asignados:

<input checked="" type="checkbox"/> KAN-34 Seguimiento de Issues	CI/CD	FINALIZADA ✓	AS
<input checked="" type="checkbox"/> KAN-39 Creación de repositorio	CI/CD	FINALIZADA ✓	AS
<input checked="" type="checkbox"/> KAN-33 Seguimiento de PRs	CI/CD	EN CURSO ▾	AS
<input checked="" type="checkbox"/> KAN-85 Despliegue completo en la nube	CI/CD	FINALIZADA ✓	AS
<input checked="" type="checkbox"/> KAN-39 Reporte Individual de Abraham	CI/CD	FINALIZADA ✓	AS
<input checked="" type="checkbox"/> KAN-32 Creación de nomenclatura	CI/CD	FINALIZADA ✓	AS
<input checked="" type="checkbox"/> KAN-31 Acceso al repositorio	CI/CD	FINALIZADA ✓	AS

Para los DEV, se abarcaron alrededor de 46 requerimientos, pero debido a la gran amplitud que manejan, se deberá dar solo una muestra limitada de algunos de ellos que se encuentran en el Backlog.

<input checked="" type="checkbox"/> KAN-57 Generación de Middleware para manejar errores y validaciones	DEV'S	FINALIZADA ✓	PS
<input checked="" type="checkbox"/> KAN-36 Generación de pruebas de seguridad	DEV'S	FINALIZADA ✓	PS
<input checked="" type="checkbox"/> KAN-35 Generación de pruebas unitarias para la robustez de la aplicación	DEV'S	FINALIZADA ✓	PS
<input checked="" type="checkbox"/> KAN-70 Generación de Middleware para rutas protegidas	DEV'S	FINALIZADA ✓	PS
<input checked="" type="checkbox"/> KAN-37 Generación de pruebas de usabilidad	DEV'S	FINALIZADA ✓	PS
<input checked="" type="checkbox"/> KAN-38 Generación de pruebas de aceptación	DEV'S	FINALIZADA ✓	PS
<input checked="" type="checkbox"/> KAN-51 Implementación de autenticación	DEV'S	FINALIZADA ✓	PS
<input checked="" type="checkbox"/> KAN-6 Subir archivos a GitHub	DEV'S	FINALIZADA ✓	PS
<input checked="" type="checkbox"/> KAN-55 Creación de READ para cursos registrados	DEV'S	FINALIZADA ✓	PS
<input checked="" type="checkbox"/> KAN-71 Manejo de filtro en operaciones CRUD	DEV'S	FINALIZADA ✓	PS
<input checked="" type="checkbox"/> KAN-56 Creación de DELETE para cursos registrados	DEV'S	FINALIZADA ✓	PS
<input checked="" type="checkbox"/> KAN-54 Creación de CREATE para cursos registrados	DEV'S	FINALIZADA ✓	PS
<input checked="" type="checkbox"/> KAN-64 Creación de HTML AdminCursos	DEV'S	FINALIZADA ✓	PS
<input checked="" type="checkbox"/> KAN-119 Creación de HTML Cursos	DEV'S	FINALIZADA ✓	PS
<input checked="" type="checkbox"/> KAN-63 Creación de HTML Mi Aprendizaje	DEV'S	FINALIZADA ✓	PS
<input checked="" type="checkbox"/> KAN-61 Creación de HTML Login	DEV'S	FINALIZADA ✓	PS
<input checked="" type="checkbox"/> KAN-60 Creación de HTML Principal	DEV'S	FINALIZADA ✓	PS

Para el QA, se abarcaron alrededor de 22 requerimientos, junto a que él pueda tener la posibilidad de crear los Issues en Jira, específicamente en uno de sus requerimientos. A continuación, podremos ver una muestra de los requerimientos que se les fueron asignados:

The screenshot shows two Jira boards side-by-side. The left board is titled 'Backlog' and lists several requirements (KAN-86, KAN-115, KAN-84, KAN-89, KAN-87) all marked as 'FINALIZADA' (Completed). The right board is titled 'Incidentes secundarias' and lists secondary requirements (KAN-88 through KAN-102), also all marked as 'FINALIZADA'. Both boards have a green header bar indicating 100% completed.

Requirement ID	Description	Status
KAN-86	Creación de Issues	FINALIZADA
KAN-115	Reporte Individual de Brandon	EN CURSO
KAN-84	Creación de Matriz de Req vs Pruebas	FINALIZADA
KAN-89	Realización de tester UI/UX	FINALIZADA
KAN-87	Realización de tester solución	FINALIZADA
KAN-88	Comprobar registro de nuevo usuario	FINALIZADA
KAN-103	Funcionalidad de botón Añadir curso en Regular	FINALIZADA
KAN-111	Funcionalidad de botones en barra superior	FINALIZADA
KAN-106	Funcionalidad del buscador de cursos en Regular	FINALIZADA
KAN-105	Muestra de lista de cursos registrados en Regular	FINALIZADA
KAN-104	Funcionalidad de botón Eliminar curso en Regular	FINALIZADA
KAN-101	Comprobar ingreso de usuario Regular	FINALIZADA
KAN-97	Comprobar ingreso de usuario Admin	FINALIZADA
KAN-98	Funcionalidad de botón Eliminar curso en Admin	FINALIZADA
KAN-99	Funcionalidad de botón Crear curso en Admin	FINALIZADA
KAN-100	Muestra de cursos para el usuario Admin	FINALIZADA
KAN-102	Funcionalidad del buscador para cursos en Admin	FINALIZADA

Debido a la amplitud de los requerimientos, no podemos reportar cada uno de ellos en el archivo, es por ello, que se dará un Excel con todo mayormente detallado adjunto en la entrega del proyecto, junto a que estará entre los documentos guardados en el repositorio.

Sabemos que cada rol tiene requerimientos que dan un apoyo fundamental en la creación final del proyecto, para así ofrecer un proyecto acorde a la rúbrica que se nos es dada. Por eso, para poder reportar cada requerimiento abarcado, se deberá crear un reporte individual de cada puesto.

#### *Formato base para el reporte individual*

Uno de los requerimientos que vi necesarios el abarcálos, fue el generar un formato base del reporte individual, para así facilitar la implementación de los reportes en el reporte final, ofreciendo coherencia tanto en sus reportes individuales como en el final. Siendo así que sea fácil el entendimiento en caso de que los otros cargos requieran información de lo hecho por alguno de los roles.

A continuación, una muestra de cómo fue hecho el formato:

**Reporte individual**

**Nombre**  
- Rol  
- Rol al que apoya

*Información básica de lo realizó en su rol y si es que dio apoyo a alguno de los roles.*

**Requerimientos**  
Dar una muestra ya sea en lista o tabla los requerimientos que se tienen en su puesto, abarcando el nombre y descripción concisa de lo que debía realizar.

**Realización**  
Dar una explicación detallada de lo que se realizó para cada uno de los requerimientos; se deben mostrar capturas de muestra donde se confirma que se realizó el requerimiento.

**Inconvenientes/Problemas y Solución**  
En caso de que haya ocurrido un problema con alguno de los requerimientos, detalla lo que se tuvo que hacer ya sea solo o en conjunto de un compañero para solucionarlo.

Podremos notar que el formato conlleva la definición del nombre de la persona, el rol y si es que apoya a otro rol, junto a una breve descripción de lo que realizó, para así después dar comienzo con el detalle de cuáles fueron sus requerimientos, y con ello, después poder dar una muestra de la realización de los requerimientos que se le fueron asignados; dando, por último, que si surgieron problemas el comentarlos y decir cómo fueron solucionados.

Para que los integrantes tuvieran acceso a dicho formato, se fue incluido en una carpeta especialmente creada para resguardar los archivos que sean manejados en el proyecto final en el repositorio en

GitHub; teniendo que crear una Branch que se llame “LEAD-AñadirYEliminarArchivos”, en la cual aseguro que se contenga una breve explicación de lo que se añadirá en el repositorio.

### Asignación de Issues al rol correspondiente

En este requerimiento trabajo en conjunto con el QA, el cual se encarga de generar el Issues para después serme asignado y yo tome el control respecto quien debería empeñarse en manejarlo. Es por ello, que continuación mostrare como el requerimiento del QA, conlleva subtareas las cuales el crea y después yo me encargo de asignar al rol correspondiente.

Estos son algunos de los Issues que maneje para la asignación de rol dependiendo la necesidad de como corregirlo:

**Error ficheros- 02**

**Descripción**  
Error, no se encuentra node, así también no se especifica la ruta donde inicia el proyecto en scripts

**Actividad**

Añadir un comentario...  
¡Tiene buena pinta! ¿Necesitas ayuda? Esto está bloqueado... Consejo de expertos: pulsa M para comentar

Frida Sarahi Garza Gálvez hace 4 días  
@PEDRO FRANCISCO DE LEON SALAZAR se le es asignado este Issues, corrección necesaria.  
Editar · Eliminar ·

Brandon Alan Carrion Morales hace 4 días

T-Dep-	Instalar depend	Se descarg	Abri r la	Se esp	No se instalo	Incorec	
02							

Finalizada ✓ Listo

**Detalles**

Persona asignada: PEDRO FRANCISCO DE LEON SALAZAR... Asignarme a mí

Principal: KAN-86 Creación de Issues

Approvers: Brandon Alan Carrion Mor... Frida Sarahi Garza Gálvez

Actual start: Ninguno

Actual end: 25 feb 2025, 23:30

Team: QA

Sprint: Tablero Sprint

Desarrollo: Crear rama · Crear confirmación

Informador: Brandon Alan Carrion Morales

Podemos ver en la imagen como tras leer lo que conlleva el Issues, yo comento la persona que se es asignada para después asignarlo en la sección correspondiente, y manejando a la par la fecha en la que fue solucionado.

**Error UIX -02**

**Descripción**  
Editar descripción

**Archivos adjuntos**

image-202502\_645.png 28 feb 2025, 05:26 pm  
image-202502\_656.png 28 feb 2025, 05:26 pm  
image-202502\_652.png 28 feb 2025, 05:26 pm

**Actividad**

Añadir un comentario...  
¡Tiene buena pinta! ¿Necesitas ayuda? Esto está bloqueado... Consejo de expertos: pulsa M para comentar

Frida Sarahi Garza Gálvez 28 de febrero de 2025, 17:40  
Se me es asignado y deberá ser manejado lo más pronto posible  
Editas Eliminar ·

Finalizada ✓ Listo

**Detalles**

Persona asignada: Frida Sarahi Garza Gálvez

Principal: KAN-86 Creación de Issues

Approvers: Brandon Alan Carrion Mor...

Actual start: 28 feb 2025, 17:15

Actual end: 28 feb 2025, 17:40

Team: QA

Sprint: Tablero Sprint

Desarrollo: Crear rama · Crear confirmación

Informador: Brandon Alan Carrion Morales

Automatización: Ejecuciones de regla

Creado 28 de febrero de 2025, 17:26 Configurar

Notamos que, aunque se me sean asignadas algunas de esos Issues, debo especificar que son para mí, y especificar tanto el día y hora en la que comencé como en la que lo resolví.

**Error vista admin -02**

**Descripción**  
Editar descripción

**Archivos adjuntos**

image-202502\_200.png 27 feb 2025, 08:02 am

**Actividad**

Añadir un comentario...  
¡Tiene buena pinta! ¿Necesitas ayuda? Esto está bloqueado... Consejo de expertos: pulsa M para comentar

Frida Sarahi Garza Gálvez 27 de febrero de 2025, 9:30  
@PEDRO FRANCISCO DE LEON SALAZAR se asegura que en la base de datos haya una limitación y lo maneja en el CSS.

Finalizada ✓ Listo

**Detalles**

Persona asignada: PEDRO FRANCISCO DE LEON SALAZAR... Asignarme a mí

Principal: KAN-86 Creación de Issues

Approvers: Brandon Alan Carrion Mor...

Actual start: 27 feb 2025, 10:00

Actual end: 27 feb 2025, 1:00

Team: QA

Sprint: Tablero Sprint

Desarrollo: Crear rama · Crear confirmación

Informador: Brandon Alan Carrion Morales

Automatización: Ejecuciones de regla

Creado 27 de febrero de 2025, 8:01 Configurar

Para la asignación de Issues, debo aclarar que sin las especificaciones del QA sería complicado el entender que área es la que requiere atención, siendo que la descripciones e imágenes implementadas dan

Debido a los múltiples Issues creados, solo abarque tres de los cuales demuestran como fue el formato, que podemos observar en la primera imagen como el QA da entrega de una tabla con las descripciones e imágenes necesarias.

Se crearon y manejaron alrededor de 12 Issues a lo largo del proyecto, siendo mayormente de asignados a los DEV correspondientes, ya sean de Backend o Frontend .

## Seguimiento de los Stories/Requerimientos

En mi sexto requerimiento, me asegure del estar al pendiente del avance de cada uno de los integrantes con sus requerimientos correspondientes, siendo esto abarcado desde el 17 de Febrero hasta el 1 de Marzo del presente año, en donde yo comprobaba de forma contante de forma presencial y por Jira el avance que iban dando cada uno de los integrantes.

A continuación, podremos ver en las siguientes imágenes una muestra de cómo las personas asignadas se encargan de poner en revisión para así yo después empeñarme en la afirmación de que fue finalizada de forma correcta.

### UI/UX

The image contains two side-by-side screenshots of Jira software interfaces, each showing a story card for a UI/UX task.

**Screenshot 1: Trabajan en conjunto en Figma**

**Description:** Se debe tener un Figma donde se pueda mostrar cada página con sus diseños correspondientes.

**Activity:** Todo Comentarios Historial Registro de actividad

Log entries:

- FG Frida Sarahi Garza Gálvez ha cambiado el Estado 1 de marzo de 2025, 8:04  
EN REVISIÓN → DONE
- FG Frida Sarahi Garza Gálvez ha actualizado: Resolución 1 de marzo de 2025, 8:04  
Nada → Done
- PL PAOLA GUADALUPE URDIALES LUNA ha actualizado: Rank 28 de febrero de 2025, 23:18  
Nada → Ranked lower
- PL PAOLA GUADALUPE URDIALES LUNA ha cambiado el Estado 28 de febrero de 2025, 23:18  
DONE → EN REVISIÓN

**Details:**

- Personas asignadas: PAOLA GUADALUPE URDIALES LUNA, Asignarme a mí
- Aplicadores: Carolina Hermosillo
- Fecha de inicio: 18 feb 2025, 10:30
- Fecha de finalización: 19 feb 2025, 7:30
- Principal: KAN-9 UI/UX
- Equipo: UI/UX
- Sprint: Tablero Sprint
- Desarrollo: Crear rama, Crear confirmación
- Informador: Frida Sarahi Garza Gálvez

**Screenshot 2: Asegura diseño en Mobil y Web para sus páginas**

**Description:** Creación de dinámicas sencillas para dar mayor interacción al usuario, como con botones o imágenes.

**Activity:** Todo Comentarios Historial Registro de actividad

Log entries:

- FG Frida Sarahi Garza Gálvez ha actualizado: Rank 28 de febrero de 2025, 10:27  
Nada → Ranked higher
- FG Frida Sarahi Garza Gálvez ha cambiado el Estado 28 de febrero de 2025, 10:27  
EN REVISIÓN → DONE
- FG Frida Sarahi Garza Gálvez ha actualizado: Resolución 28 de febrero de 2025, 10:27  
Nada → Done
- FG Frida Sarahi Garza Gálvez ha actualizado: Actual end 28 de febrero de 2025, 10:27  
Nada → 28 feb 2025, 10:30

**Details:**

- Personas asignadas: Carolina Hermosillo, Asignarme a mí
- Aplicadores: Frida Sarahi Garza Gálvez, PAOLA GUADALUPE URDIALES LUNA
- Fecha de inicio: 18 feb 2025, 10:00
- Fecha de finalización: 28 feb 2025, 10:30
- Principal: KAN-9 UI/UX
- Equipo: UI/UX
- Sprint: Tablero Sprint
- Desarrollo: Crear rama, Crear confirmación
- Informador: Frida Sarahi Garza Gálvez

### CI/CD

The image shows a single screenshot of a Jira story card for a CI/CD task.

**Description:** Se le brinda a los integrantes del equipo y al profesor el acceso al repositorio creado.

**Archivos adjuntos:** image-202502\_600.png, image-202502\_798.png

**Activity:** Todo Comentarios Historial Registro de actividad

Log entries:

- FG Frida Sarahi Garza Gálvez ha cambiado el Estado 20 de febrero de 2025, 7:31  
EN REVISIÓN → DONE
- FG Frida Sarahi Garza Gálvez ha actualizado: Resolución 20 de febrero de 2025, 7:31  
Nada → Done
- FG Frida Sarahi Garza Gálvez ha actualizado: Actual end 20 de febrero de 2025, 7:31  
16 feb 2025, 20:30 → 20 feb 2025, 7:20
- FG Frida Sarahi Garza Gálvez ha cambiado el Estado 20 de febrero de 2025, 7:30  
IN PROGRESS → EN REVISIÓN

**Details:**

- Personas asignadas: Abraham Isai Garza Sánchez, Asignarme a mí
- Aplicadores: Frida Sarahi Garza Gálvez
- Fecha de inicio: 16 feb 2025, 20:00
- Fecha de finalización: 20 feb 2025, 7:20
- Principal: KAN-12 CI/CD
- Equipo: CI/CD
- Sprint: Tablero Sprint
- Desarrollo: Crear rama, Crear confirmación
- Informador: Frida Sarahi Garza Gálvez
- Automatización: Ejecuciones de regla

Historial de cambios:

- Creado 18 de febrero de 2025, 10:16
- Actualizado 20 de febrero de 2025, 7:31
- Resuelto 20 de febrero de 2025, 7:31

## Dev

The image contains two side-by-side screenshots of Jira boards. The left board is titled 'Subir archivos a GitHub' and the right board is titled 'Implementación de CSS'. Both boards show a 'Historial' tab with activity logs and a 'Finalizada' status bar at the top.

**Subir archivos a GitHub:**

- Descripción:** Se deben subir los archivos que se crearon en el avance del proyecto.
- Actividad:** Shows several log entries from users like Frida Sarahi Garza Gálvez and PEDRO FRANCISCO DE LEON SALAZAR.
- Detalles:** Shows assignee (Pedro Francisco de Leon Salazar), approvers (Abraham Isai Garza Sánchez, Frida Sarahi Garza Gálvez), start date (18 feb 2025, 0:00), end date (18 feb 2025, 8:30), and team (DEV).

**Implementación de CSS:**

- Descripción:** Se asegura de concordar el diseño CSS a los HTML que fueron creados con anterioridad para el Login, Principal, Register, RegularCursos y AdminCursos.
- Archivos adjuntos:** Captura de pantalla .png (27 feb 2025, 10:24 am).
- Incidencias secundarias:** KAN-69 Interacciones dinámicas.
- Actividad:** Shows log entries from Frida Sarahi Garza Gálvez.
- Detalles:** Shows assignee (Frida Sarahi Garza Gálvez), approvers (Brandon Alan Carrón Morales, Frida Sarahi Garza Gálvez), start date (21 feb 2025, 7:00), end date (28 feb 2025, 10:40), and team (DEV).

## QA

The image shows a single Jira board titled 'Creación de Matriz de Req vs Pruebas'. It has a 'Historial' tab with activity logs and a 'Finalizada' status bar at the top.

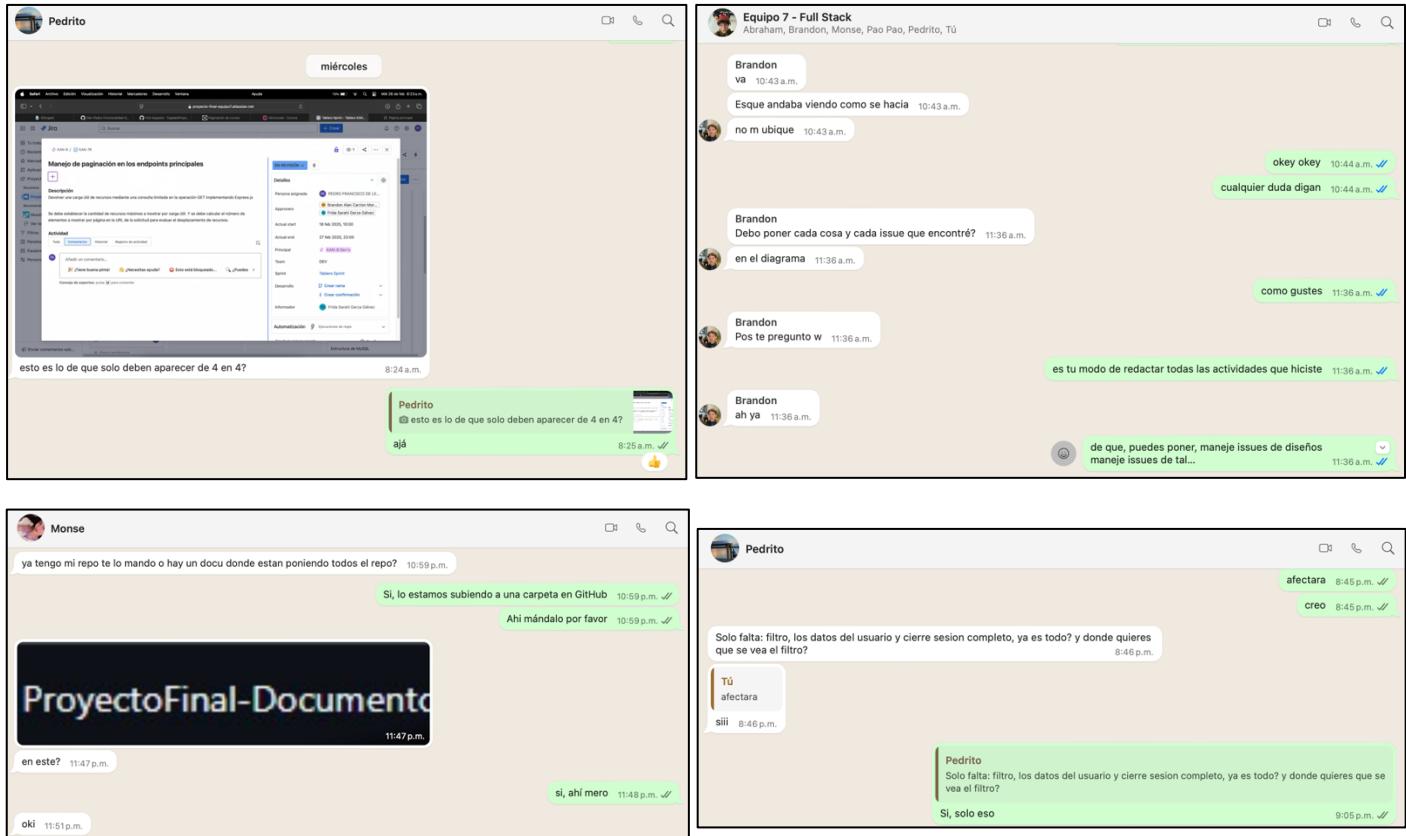
- Descripción:** Editar descripción.
- Actividad:** Shows log entries from users like Brandon Alan Carrón Morales and Frida Sarahi Garza Gálvez.
- Detalles:** Shows assignee (Brandon Alan Carrón Morales), approver (Frida Sarahi Garza Gálvez), start date (21 feb 2025, 7:00), end date (28 feb 2025, 19:30), principal (KAN-11 QA), team (QA), sprint (Tablero Sprint), and informer (Frida Sarahi Garza Gálvez).
- Automatización:** Shows execution details for a rule.

Podemos observar cómo me encargo en algunos requerimientos el asignar la fecha y hora que termino junto que tras revisarlo poder darlo en un estado de finalizado. Este proceso fue llevado a lo largo de todos los requerimientos que se crearon y abarcaron en el proyecto. Asimismo, en caso de querer ver más a detalle dicho seguimiento, podrán comprobarlo en el Board de Jira.

## Manejo de dudas en los integrantes

A lo largo de todo el proyecto, me di como tarea el tratar las dudas de los integrantes que fueran surgiendo a lo largo de la realización de sus requerimientos, dichas dudas podían ser tratadas de manera presencial y por WhatsApp.

Algunas muestras de dichas dudas tratadas por vía WhatsApp:

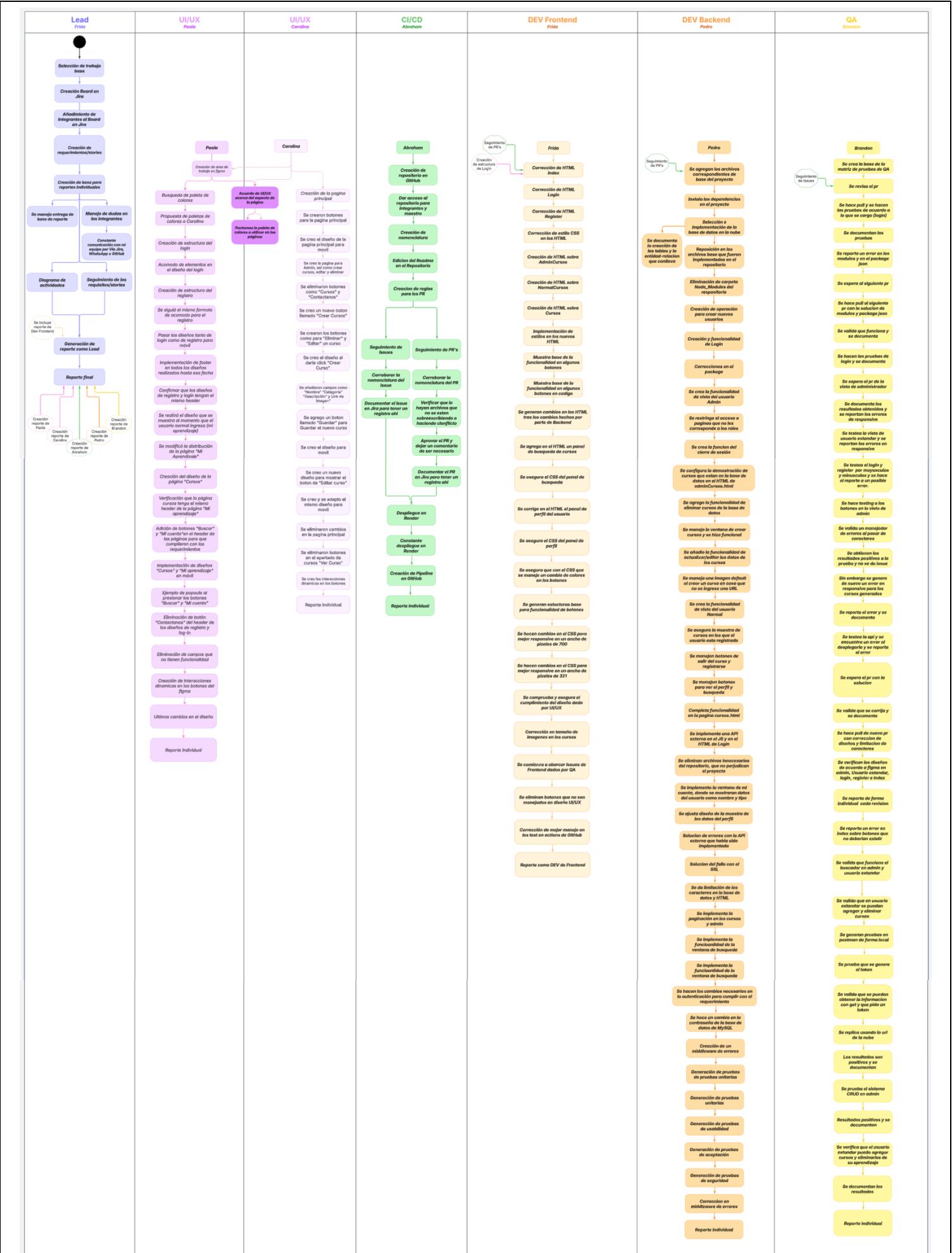


Creo tener la fortuna de portar un equipo con buena comunicación y confianza respecto a cualquier problema o inconveniente que surgiera a lo largo de la realización del proyecto final.

## Creación y gestión del diagrama de actividades

Mi último requerimiento como Lead trata acerca de la creación y gestión del diagrama de actividades de cada rol que se maneja en el equipo, siendo así que tenga un diagrama donde se muestre la actividad que conllevó cada integrante, siendo así un diagrama altamente extenso debido a las múltiples actividades que llevan como equipo en el proyecto final.

Debido a que buscamos un buen abarque respecto a cada una de las actividades que abarcó un rol se tomó por apoyo el que cada cargo creará su diagrama, siendo así que una vez ellos me entreguen dicho diagrama yo pudiera hacer la implementación correcta a un diagrama final con todos los roles implementados ofreciendo orden y coherencia respecto a la inicialización de sus actividades.



Gracias a este diagrama podemos comprender que tras la creación de los requerimientos por parte del iee los demás roles pudieron empezar a realizar sus actividades que fueran complementarias al proyecto final, a la par que podemos notar como ciertas secciones son complementadas con otros compañeros de equipo, un buen ejemplo son las UI/UX las cuales tuvieron que ponerse de acuerdo para poder trabajar en conjunto el diseño; y que por parte del CI/CD el cómo él fue parte del monitoreo en el repositorio de github, el cual se enlazaba con las actividades que realizaban los DEV y el QA.

Este diagrama llega a ser de gran apoyo ya que nos dio una idea mayormente visual respecto a cuál era nuestro orden y cómo es que al final del día nos terminamos complementando.

Si desea ver de forma más detallada dicho diagrama puede ingresar en el siguiente link:

[Link de diagrama de actividades](#)

## Paola Guadalupe Urdiales Luna

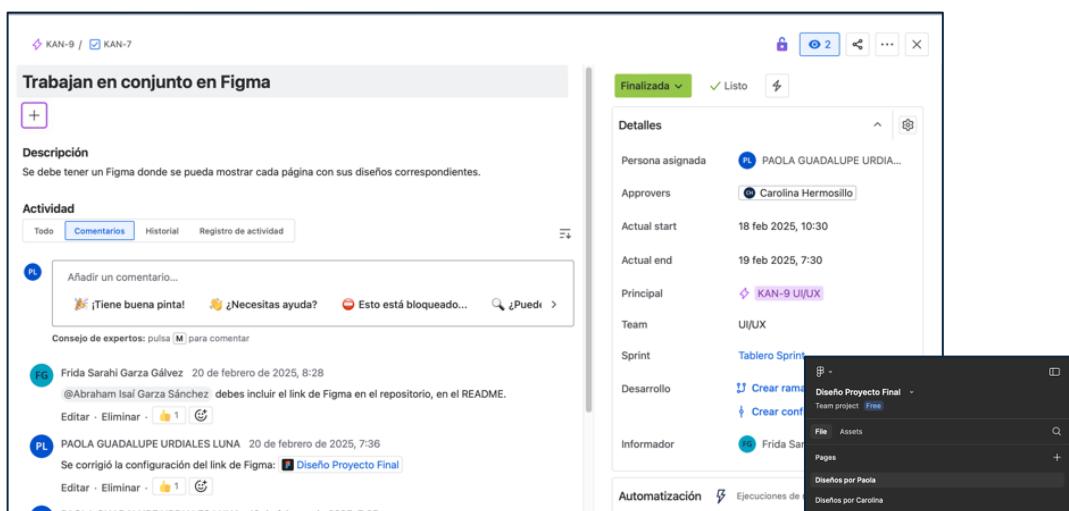
Cargo: UI/UX

En la entrega final del proyecto, desempeñé el rol de UI/UX en conjunto con mi compañera Carolina. El diseño de la página web fue distribuido en dos partes, la interfaz del admin y la interfaz del usuario común, en mi caso las páginas que diseñé fueron las del usuario común, las cuales fueron: página de log-in, página de registro, página de los cursos a los que el usuario está inscrito (“Mi aprendizaje”) y la página de los cursos disponibles para que el usuario pueda obtener (“Cursos”). Algunas de las tareas con las que trabajé fueron: la elección de la paleta de colores (en conjunto con mi compañera Carolina), el trabajo colaborativo en Figma, la definición de la estructura visual, modificaciones, integración de acciones dinámicas en los botones, etc. Todo el diseño fue creado con el seguimiento de los requerimientos que la Lead definió, esto me ayudó mucho a llevar un orden de los aspectos faltantes en mis diseños y que queden lo más completos y eficientes posibles.

### Realización de requerimientos

#### Primer requerimiento

Durante la primera semana de trabajo en el proyecto, la primera tarea que se realizó fue realizar un área de trabajo en Figma para el trabajo colaborativo con mi compañera. Figma es un espacio de trabajo que nos facilitó demasiado el desarrollo de las páginas en paralelo, ya que cada una tenía su espacio de trabajo y así mismo ambas teníamos acceso al trabajo de la otra, para cualquier aclaración de colores o formato de los elementos de la interfaz.



## Segundo requerimiento

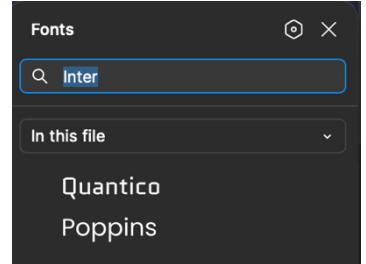
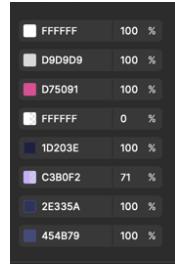
Lo próximo que realizamos fue elegir una paleta de colores que quedara bien con la temática del sitio web.

The screenshot shows a Jira ticket interface. The title is "Concuerdan con tipo de diseño". The description states: "Los encargados de UI/UX deben definir en conjunto un tipo de paleta de colores, tipo de letra, manejo de imágenes, entre múltiples necesidades del diseño." The activity panel shows several comments from users PAOLA GUADALUPE URDIALES LUNA, Carolina Hermosillo, and Frida Sarahi Garza Gálvez. The details panel shows the ticket is "Finalizada" with approvers PAOLA GUADALUPE URDIALES LUNA, Carolina Hermosillo, and KAN-B UI/UX. It was started on 18 feb 2025, 10:00 and ended on 25 feb 2025, 10:00. The principal is KAN-B UI/UX, and the team is UI/UX. The sprint is Tablero Sprint, and the developer is Frida Sarahi Garza Gálvez.

Así como la elección de las tipografías que quedan mejor con la página, las tipografías que se utilizaron fueron:

**Quantico:** Utilizado para los botones de la página tanto del header como del body.

**Poppins:** Tipografía utilizada para el resto de la página como los nombres y descripciones de los cursos que la página ofrece.



El formato de las imágenes se le fue dando en el transcurso de los días, según lo que parecía mejor para el acomodo de la página, siempre cuidando que el acomodo de los botones, página y textos de los cursos mantuvieran un aspecto agradable para el usuario.

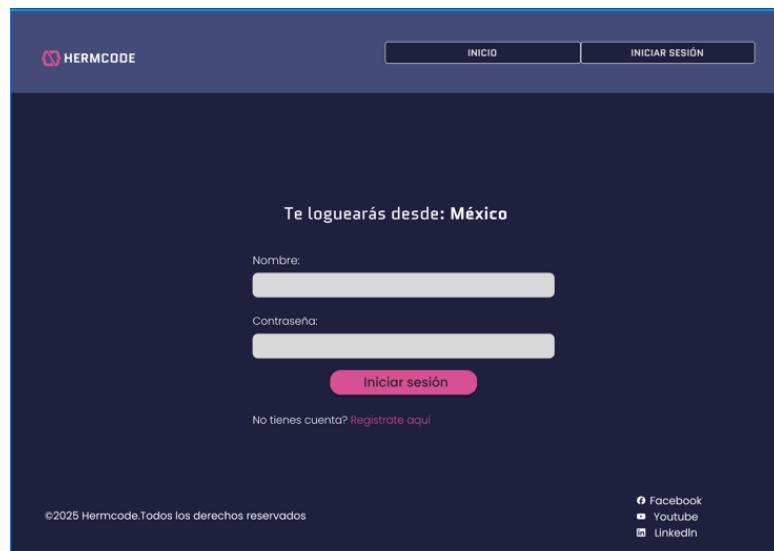
## Tercer requerimiento

Ya con los elementos como la tipografía y la paleta de colores definidos se comenzó a trabajar en el desarrollo de la página de ingreso, la cual se compone de una cabecera con el nombre de la empresa

The screenshot shows a Jira ticket interface. The title is "Generación de página Login". The description states: "Debe crear un diseño donde se le pida al usuario el ingreso de un nombre y contraseña como base." The activity panel shows several comments from users PAOLA GUADALUPE URDIALES LUNA, Carolina Hermosillo, and Frida Sarahi Garza Gálvez. The details panel shows the ticket is "Finalizada" with approvers PAOLA GUADALUPE URDIALES LUNA, Carolina Hermosillo, and KAN-B UI/UX. It was started on 18 feb 2025, 10:00 and ended on 22 feb 2025, 15:00. The principal is KAN-B UI/UX, and the team is UI/UX. The sprint is Tablero Sprint, and the developer is Frida Sarahi Garza Gálvez.

"Hermcode" al lado derecho se ingresaron dos botones "Inicio" e "Iniciar sesión", en el cuerpo de esa parte de la aplicación hay dos campos en los que el usuario deberá ingresar el nombre de su cuenta y la contraseña que el trabajador o cliente definió en el registro, lo siguiente es un botón, que al pulsarlo el usuario será redirigido a la página "Mi aprendizaje", debajo de ese botón hay un link en el que el usuario podrá crear una cuenta para poder acceder a la información del sitio, finalmente hay

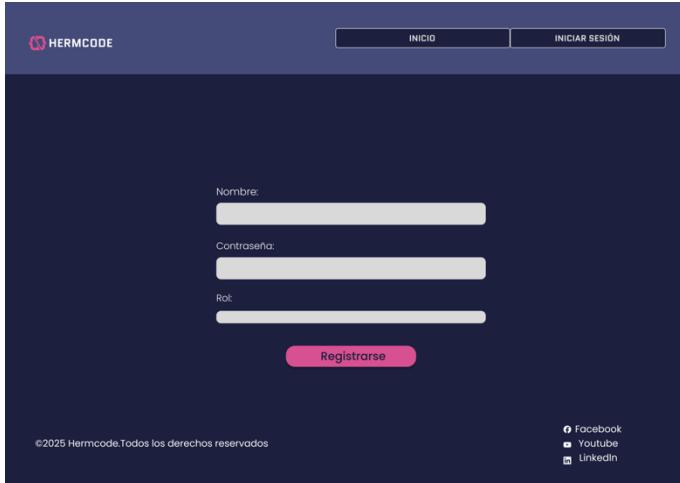
un pie de página en donde está el copyright y los enlaces de las redes sociales.



## Cuarto requerimiento

La distribución de la página de registro es prácticamente el mismo que la del log-in, lo único en lo que hay diferencias es en el campo de rol, en donde el usuario podrá seleccionar el tipo de usuario que será (normal ó admin) y que en la dicha página de registro no cuenta con la muestra del lugar geográfico en donde estas ingresando a la página.

A esta página se puede acceder desde el log-in (en el link de la parte inferior de la aplicación web).



## Quinto requerimiento

Luego, en el requerimiento habla de la generación de una página llamada “Regular cursos” en donde se muestren los cursos en los que el usuario esta registrado, sin embargo, esta actividad tuvo que ser dividida en dos páginas web, una para que el usuario vea en que cursos esta registrado, es decir a cuales cursos el ya tiene acceso (“Mi aprendizaje”) y una segunda página para que se le muestren al usuario los cursos nuevos y disponibles a los que aún no se inscribe (“Cursos”) ambas páginas son muy similares, ya que las dos exponen los nombres de los cursos, en conjunto con la categoría, una pequeña descripción de los mismos y una imagen que haga referencia al lenguaje de programación que se esta ofertando.

## Sexto requerimiento

Todas las páginas realizadas deben tener una base de cómo es que quiere el desarrollador que se vea su aplicación, cuando entras a ella desde un dispositivo con una extensión de pantalla menor al de una PC, se debe tener bien en claro cómo es que están acomodados todos los elementos de la interfaz, es ahí donde UI/UX debe hacer las pruebas de la estructura que deben tener las distintas páginas de la app. Es así como para cada una de las páginas que se diseñaron deben contar con su versión móvil, lo que se realizó en ese requerimiento fue el diseño de las páginas, en una extensión de pantalla menor, en donde se

A screenshot of a Jira ticket interface. The ticket is titled "Asegura diseño en Mobil y Web para sus páginas" and is assigned to "PAOLA GUADALUPE URDIA...". It has three approvers: "Carolina Hermosillo" and "Frida Sarahi Garza Gámez". The ticket is marked as "Finalizada" (Completed). The "Actividad" section shows a comment from "Frida Sarahi Garza Gámez" with a link to a Figma file. The "Detalles" section includes fields for "Actual start" (18 feb 2025, 10:00), "Actual end" (25 feb 2025, 10:00), "Principal" (KAN-9 UI/UX), "Team" (UI/UX), "Sprint" (Tablero Sprint), "Desarrollo" (Crear rama, Crear confirmación), and "Informador" (Frida Sarahi Garza Gámez). The "Automatización" section shows "Ejecuciones de regla".

acomodaron los elementos del sitio, de modo que los pop-outs no se vean empalmados ni que los botones del cuerpo se vean demasiado juntos.

## Séptimo requerimiento

Para una mejor experiencia de usuario, es importante crear interacciones dinámicas en la página web.

A screenshot of a Jira ticket interface. The ticket is titled "Genera interacciones dinámicas en sus páginas" and is assigned to "PAOLA GUADALUPE URDIA...". It has three approvers: "Carolina Hermosillo" and "Frida Sarahi Garza Gámez". The ticket is marked as "Finalizada" (Completed). The "Actividad" section shows a comment from "Frida Sarahi Garza Gámez" with a link to a Figma file. The "Detalles" section includes fields for "Actual start" (18 feb 2025, 10:00), "Actual end" (26 feb 2025, 12:00), "Principal" (KAN-9 UI/UX), "Team" (UI/UX), "Sprint" (Tablero Sprint), "Desarrollo" (Crear rama, Crear confirmación), and "Informador" (Frida Sarahi Garza Gámez). The "Automatización" section shows "Ejecuciones de regla".

Una interacción dinámica se da cuando la interfaz responde a las acciones del usuario (click, hover, scroll, etc.), haciendo que la navegación sea más fluida e intuitiva.

En este caso, la interacción principal que se implementó según la acción del usuario (**hover**) es que, al pasar el cursor sobre un botón, este cambia de color. Además de este cambio visual, al iniciar sesión en la página **Mi Aprendizaje**, al hacer click tanto en el **browser**

como en el botón "**MI CUENTA**", se despliegan pop-outs con el buscador y el perfil del usuario. Esos pequeños detalles hacen una gran diferencia en la experiencia de quien usa la plataforma, porque no solo hacen que la página sea atractiva visualmente, sino también interactiva y dinámica, generando una sensación de sorpresa y fluidez en la navegación.

En la imagen se pueden ver los botones de la cabecera del log-in: el botón de la derecha es como el usuario lo verá la mayor parte del tiempo, mientras que el botón de la izquierda muestra el estado cuando el usuario hace **hover** sobre él.

A screenshot of the HERM CODE website. The header includes the logo, a search bar, and navigation links for "MI APRENDIZAJE", "CURSOS", "MI CUENTA", and "CERRAR SESIÓN". The main content area is titled "Mi aprendizaje". The "MI CUENTA" button is highlighted with a tooltip showing "Frida Sarahi Garza Gámez Estudiante".

Two screenshots of the HERM CODE website. The top screenshot shows the "MI CUENTA" button with a tooltip "Frida Sarahi Garza Gámez Estudiante". The bottom screenshot shows the "MI CUENTA" button with a tooltip "Búscame en el curso aquí".

Así es como se muestran los pop outs en la página al interactuar con el botón "**MI CUENTA**" y el buscador.

## Octavo requerimiento

Finalmente tenemos la especificación de manejos entre páginas, el cual es un camino que ayuda al usuario(para saber como llegar a la página que desea) o bien al desarrollador(para saber en que orden tiene que crear las páginas).

The screenshot shows a task card from KAN-9. The title is "Especificación de manejo entre paginas". The card includes a description: "Deberá dar una especificación de cómo será posible el llegar a diversas páginas, como la de página principal al Login o a la de Register." It has sections for "Actividad" (Comments, History, Activity Log), "Comentarios" (with a text input field), and "Consejo de expertos: pulsar M para comentar". The "Detalles" section shows the following information:

- Personas asignadas: PAOLA GUADALUPE URDIA...
- Aprobadores: Frida Sarahi Garza Gálvez
- Actual start: 26 feb 2025, 10:00
- Actual end: 26 feb 2025, 2:00
- Principal: KAN-9 UI/UX
- Team: UI/UX
- Sprint: Tablero Sprint
- Desarrollo: Crear rama, Crear confirmación
- Informador: Frida Sarahi Garza Gálvez
- Automatización: Ejecuciones de regla

## Flujo de pantallas de tipo usuario:

[https://lucid.app/lucidchart/740ce8b9-6ad5-4802-8cd7-ba1ce19bfc32/edit?viewport\\_loc=139,57,2230,1184,E-hXik3\\_dQgi&invitationId=inv\\_10a8d464-767e-447d-95c5-301de9142f15](https://lucid.app/lucidchart/740ce8b9-6ad5-4802-8cd7-ba1ce19bfc32/edit?viewport_loc=139,57,2230,1184,E-hXik3_dQgi&invitationId=inv_10a8d464-767e-447d-95c5-301de9142f15)

## Carolina Monserrat Hermosillo Castañeda

Cargo: UI/UX

Como parte del trabajo como UI/UX realice la página principal, y la pagina para Admin tanto para vista web y móvil con ayuda de Figma.

### Realización de requerimientos

Para la creación de la página principal añadí un header en el que conlleva el botón de iniciar sesión, el logo de la página y los unscroll de “Cursos” y “Contáctanos”.

Dentro de la misma página esta una breve descripción en la que habla sobre que ofrece nuestra página.

También añadí un apartado que se llama “Cursos en Línea” y muestra algunos de los cursos que ofrecemos en nuestra página, al igual se añadió un apartado para poder contactarnos donde puedes proporcionar tu nombre, correo y el mensaje que quieras enviar, y por último se agregó el pie de página donde muestra los links y el copyright.

#### Vista en Web:

The screenshot shows the desktop version of the Herocode website. At the top is a dark header with the logo, navigation links for "CURSOS" and "CONTACTANDOS", and a "INICIAR SESIÓN" button. Below the header is a section titled "EMPIEZA A APRENDER" with a sub-section about improving programming skills. The main content area features three course cards: "HTML & CSS" (with a person icon), "Introducción a JavaScript" (with a JS icon), and "Fundamentos de Python" (with a Python icon). At the bottom is a "SE PARTE DE NUESTRO EQUIPO!" section with a form for "CONTACTO" (Name, Email, Message) and a "DEJA UNA PREGUNTA" button. The footer contains social media links for Facebook, YouTube, and LinkedIn, and a copyright notice: "©2025 Herocode. Todos los derechos reservados".

#### Vista en Móvil:

Para el diseño en móvil realmente fue el mismo que se utilizó en la vista web.

The screenshot shows three mobile device screens displaying the Herocode website. The first screen is the "Móvil pagina principal" (Mobile main page) showing the header and the "EMPIEZA A APRENDER" section. The second screen is the "Móvil Cursos" (Mobile Courses) showing the three course cards. The third screen is the "Móvil Contac y PDP" (Mobile Contact and About) showing the "SE PARTE DE NUESTRO EQUIPO!" section with a contact form and social media links at the bottom.

Por otro lado, también me encargue de crear un diseño donde se le pueda dar al Administrador el poder crear, editar y eliminar un curso que se vaya a crear en la página.

En la imagen se muestra el cómo se debería de ver al crear un curso, para poder hacer eso se creó un botón llamado “Crear Curso”. Para añadir un nuevo curso deberá darle un Nombre, Categoría, Descripción, así con un link para añadir una imagen, una vez que el Administrador llene los campos este se agregará a la lista de sus cursos creados. También añadí en la parte del header la lupa de “Búsqueda” para buscar el curso que el deseé.

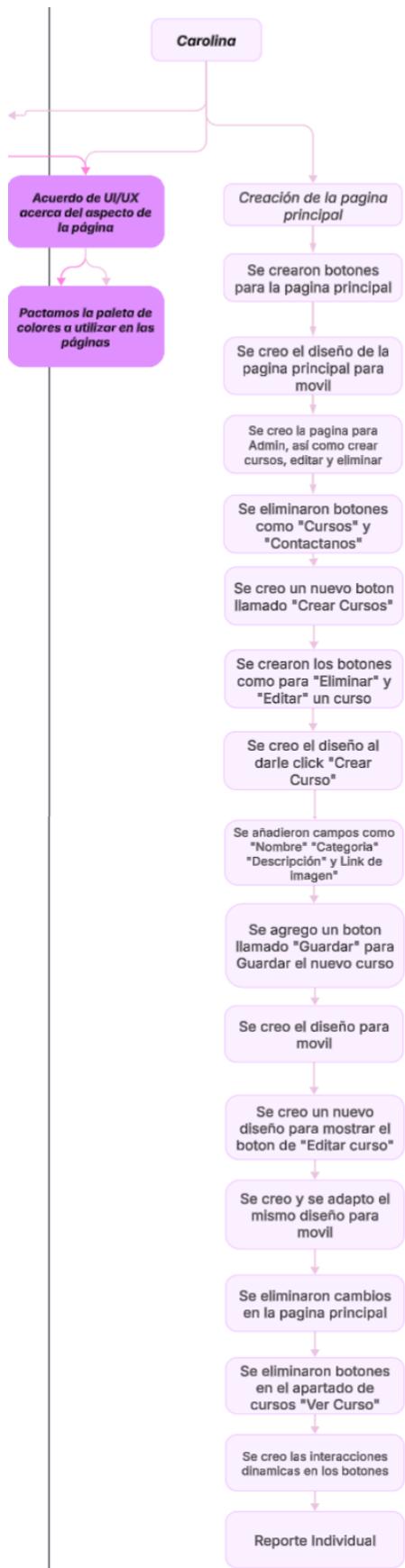
### Vista de “Editar curso”

En este caso, solo el Administrador podrá editar su propio curso, ya sea cambiando el Nombre, la Categoría, la Descripción o cambiando la imagen del curso, nuevamente una vez que haya editado los campos que fueran de su elección, está el botón de “Guardar”.

Como último paso, cree una visión completa del como seria el diseño en móvil:

Como último recurso cree los diseños dinámicos de los botones, así como para las botones de “Iniciar Sesión”, “Cerrar Sesión”, y “Mi Cuenta” tanto para web, así como para móvil.

También realice el diagrama de actividades para UI/UX, de acuerdo a como avanza con los requerimientos que se me solicitaron:



# Abraham Isaí Garza Sánchez

Cargo: CI/CD

El rol que se me asignó se encargó principalmente de llevar a cabo el manejo del Repositorio del Proyecto Final, también se encargó del despliegue de la página por medio de la nube

Link al repositorio: <https://github.com/Yugidar/Proyecto-Final>

## Requerimientos

- Se debía llevar a cabo el seguimiento de Pull Request dentro del repositorio, estos iban siendo documentados a través de imágenes por medio de Jira.

The screenshot shows a Jira interface for tracking Pull Requests. On the left, there's a board titled 'Seguimiento de PRs' with several cards representing different PRs. Each card has a preview image and some text. On the right, there's a detailed view of a specific PR. The 'Detalles' section includes fields for 'Personas asignadas' (Abraham Isaí Garza Sánchez), 'Aprobadores' (Frida Sarahi Garza Gálvez), 'Actual start' (18 feb 2025, 10:00), 'Actual end' (27 feb 2025, 0:00), 'Principal' (KAN-12 CI/CD), 'Team' (CI/CD), 'Sprint' (Tablero Sprint), 'Desarrollo' (options to 'Crear rama' or 'Crear confirmación'), and 'Informador' (Frida Sarahi Garza Gálvez). There's also an 'Automatización' section with a link to 'Ejecuciones de regla'.

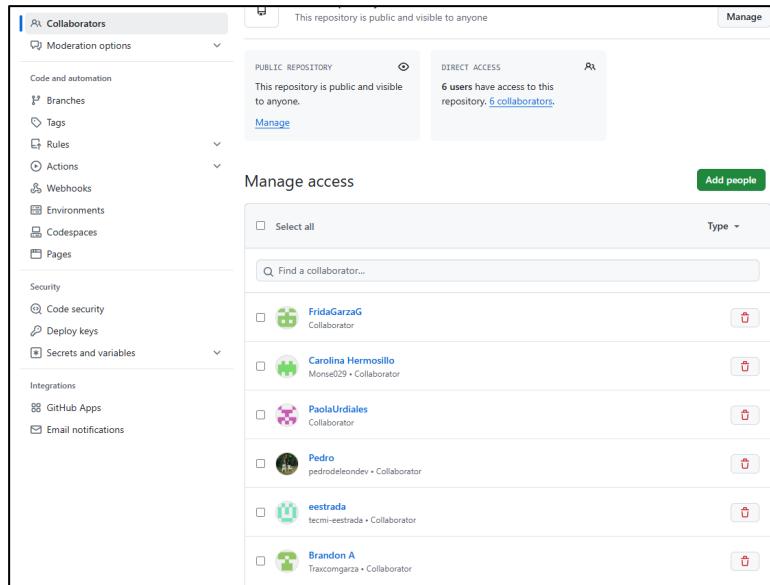
- Se debía llevar a cabo el seguimiento de Issues dentro del Repositorio, estos también fueron documentados a través de imágenes por medio de Jira.

The screenshot shows a Jira interface for tracking issues. On the left, there's a board titled 'Seguimiento de Issues' with several cards representing different issues. Each card has a preview image and some text. On the right, there's a detailed view of a specific issue. The 'Detalles' section includes fields for 'Personas asignadas' (Abraham Isaí Garza Sánchez), 'Aprobadores' (Brandon Alan Carrion Mora and Frida Sarahi Garza Gálvez), 'Actual start' (18 feb 2025, 9:00), 'Actual end' (Ninguno), 'Principal' (KAN-12 CI/CD), 'Team' (CI/CD), 'Sprint' (Tablero Sprint), 'Desarrollo' (options to 'Crear rama' or 'Crear confirmación'), and 'Informador' (Frida Sarahi Garza Gálvez). There's also an 'Automatización' section with a link to 'Ejecuciones de regla'.

- Se debía hacer el despliegue de la aplicación a través de una plataforma como Heroku, AWS, Digital Ocean o Render.

## Realización

Lo primero dentro de la realización del proyecto para el CI/CD fue la creación del Repositorio que fue nombrado “Proyecto-Final”. El paso a seguir fue incluir a los miembros participantes del equipo dentro del repositorio, así como añadir al profesor para que tenga acceso a este y pueda verificar/corroborar los resultados.



Después se realizó la creación del archivo README.txt dentro del repositorio para facilitar la presentación de esté. En esta primera iteración del archivo se incluyeron la nomenclatura que debía ser usada para los Commits y la nomenclatura que debía ser usada para los Branches que se creen dentro del repositorio. También se escribieron los nombres de los integrantes del equipo.

En este paso se tuvo el primer Problema que será abordado dentro de sección de problemas dentro de este mismo documento.

Después de esto se creó una carpeta dentro del repositorio donde se colocarían los documentos necesarios para que los miembros del equipo puedan subir sus respectivos reportes o documentos sin la necesidad de que entren interfieran con la raíz del repositorio. Esta después fue renombrada por el LEAD a “ProyectoFinal-Documentos” donde el LEAD subió un archivo base de cómo se debían realizar los reportes del equipo.

Después de esto, se me fue pedido a través de Jira por el LEAD incluir la nomenclatura para los Issues dentro del repositorio ya que este no había sido escrito anteriormente.

Este fue implementado correctamente y sin problemas, se optó por usar “QA-Tipo de error-Descripción breve (Comentario libre para detallar por si es necesario)” como nomenclatura para los Issues.

A screenshot of the GitHub README file for 'Proyecto-Final'. It contains sections for 'REPOSITORIO PARA EL PROYECTO FINAL DE LA CLASE DESARROLLO FULL STACK', 'NOMENCLATURA PARA COMMITS/PUSHES', 'NOMENCLATURA PARA BRANCHES', and 'NOMENCLATURA PARA ISSUES'. Below these sections, it lists 'Integrantes del equipo' with names: Frida, Paola, Pedro, Carolina, and Brandon. At the bottom, there are 'Editar' and 'Eliminar' buttons. Below the README, there are two Jira comments. The first comment from 'Abraham Isaí Garza Sánchez' says 'Se hicieron los cambios y se agrego la nomenclatura para los issues del QA'. The second comment from 'Frida Sarahi Garza Gálvez' says '@Abraham Isaí Garza Sánchez debes hacer cambios para la nomenclatura de Issues para el QA.' Both comments have 'Editar' and 'Eliminar' buttons.

Después se me fue solicitado por el LEAD incluir el Link del Figma dentro del README del repositorio para facilitar su acceso.

FG Frida Sarahi Garza Gálvez hace 4 días  
@Abraham Isaí Garza Sánchez debes incluir el link de Figma en el repositorio, en el README.  
Editar · Eliminar · 1

PL PAOLA GUADALUPE URDIALES LUNA hace 4 días  
Se corrigió la configuración del link de Figma: Diseño Proyecto Final  
Editar · Eliminar · 1

Finalmente, el README del repositorio se vio de la siguiente manera.

**REPOSITORIO PARA EL PROYECTO FINAL DE LA CLASE DESARROLLO FULL STACK**

**NOMENCLATURA PARA COMMITS/PUSHES**  
Rol-Nombre-Descripción de que se realizó brevemente-Descripción más detallada del cambio

**NOMENCLATURA PARA BRANCHES**  
Rol-Lo Que se está realizando

**NOMECLATURA PARA ISSUES**  
QA-Tipo de error-Descripción breve (Comentario libre para detallar por si es necesario)

**ENLACE AL FIGMA**  
<https://www.figma.com/design/ShzIS5VpCLGj7c4Zpm2WGK/Dise%C3%B1o-Proyecto-Final?node-id=28-2>

**Integrantes del equipo**

Frida Garza  
Paola Urdiales  
Alan Carrion  
Pedro de Leon  
Carolina Hermosillo

Añadido a esto dentro del repositorio se modificaron las reglas de este para que al momento de querer fusionar los Branches a la rama principal este tenga un bloqueo y no pueda realizarse el Merge sin que antes tenga por lo menos un Approve de algún reviewer asignado, esto para evitar problemas al momento de realizar Merge por roles que no sean CI/CD.

CICD-PR  
1 branch rule

Require a pull request before merging  
Require all commits be made to a non-target branch and submitted via a pull request before they can be merged.

Hide additional settings ^

Required approvals  
1

The number of approving reviews that are required before a pull request can be merged.

Dismiss stale pull request approvals when new commits are pushed  
New, reviewable commits pushed will dismiss previous pull request review approvals.

Require review from Code Owners  
Require an approving review in pull requests that modify files that have a designated code owner.

Require approval of the most recent reviewable push  
Whether the most recent reviewable push must be approved by someone other than the person who pushed it.

Require conversation resolution before merging  
All conversations on code must be resolved before a pull request can be merged.

Request pull request review from Copilot [Preview](#)  
Automatically request review from Copilot for new pull requests, if the author has access to Copilot code review.

Allowed merge methods [Preview](#)  
Merge, Squash, Rebase

When merging pull requests, you can allow any combination of merge commits, squashing, or rebasing. At least one option must be enabled.

## Despliegue en la nube

Para el despliegue en la nube se consideró utilizar diversos servicios como AWS, Digital Ocean, Heroku y Render. Al final se optó por utilizar Render debido a su interfaz de fácil uso y que permite desplegar la página directamente desde nuestro repositorio de GitHub sin la necesidad de registrar alguna tarjeta de crédito o débito o de algún tipo de pago. La página hace una configuración inicial basada en los archivos que encuentra en el repositorio para poder desplegarla, también cuenta con otros planes para alojar páginas más pesadas, es decir que necesiten de bases de datos grandes o que tengan mucho tráfico. Una vez seleccionado el repositorio de nuestro proyecto (Proyecto-Final) hizo la configuración inicial basada en los archivos encontrados, que fue aquí donde se formó el segundo error que será abordado en su debida sección.

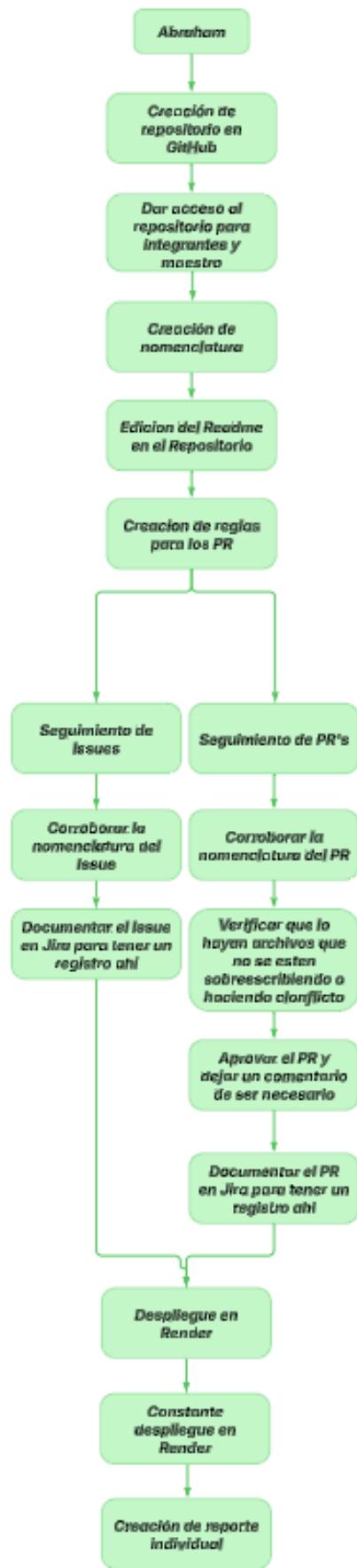
Una vez arreglado el problema la página se desplegó de manera correcta utilizando el siguiente enlace: <https://proyecto-final-07c2.onrender.com/> con la consola y página viéndose de la siguiente manera.

The screenshot shows the homepage of a website named 'HERM CODE'. The header features a logo with a stylized 'H' and 'C', followed by the text 'HERM CODE'. Below the header are navigation links for 'CURSOS' and 'CONTÁCTANOS', and buttons for 'VER CURSOS' and 'INICIAR SESIÓN'. The main content area has a dark blue background with a central illustration of a person wearing headphones and working on a laptop. The text 'EMPIEZA A APRENDER' is displayed above the illustration. Below the illustration, there is a section titled 'CURSOS EN LÍNEA'.

The screenshot shows the Render service dashboard for the repository 'Proyecto-Final'. At the top, it displays the URL 'https://proyecto-final-07c2.onrender.com/' and a message stating 'Suspended by you. You are not billed for suspended services.' Below this, there are two informational messages: 'Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more.' and 'Newer logs are unavailable because the service is suspended.' A large log viewer window is at the bottom, showing a list of log entries. The logs indicate the service was suspended due to inactivity and show a recent message about the service being live again.

Time	Message
Feb 25 08:11:09 AM	at PacketParser.unsocket (/opt/render/project/src/node_modules/mysql/lib/base/connection.js:93:12)
Feb 25 08:11:09 AM	at PacketParser.executeStart (/opt/render/project/src/node_modules/mysql/lib/packet_parser.js:75:16)
Feb 25 08:11:09 AM	at Socket.<anonymous> (/opt/render/project/src/node_modules/mysql/lib/base/connection.js:100:25)
Feb 25 08:11:09 AM	at Socket.emit (node:events:524:28)
Feb 25 08:11:09 AM	at addChunk (node:internal/streams/readable:561:12)
Feb 25 08:11:09 AM	at readableAddChunkPushByMode (node:internal/streams/readable:512:3)
Feb 25 08:11:09 AM	at Readable.push (node:internal/streams/readable:392:5)
Feb 25 08:11:09 AM	code: 'ER_USER_LIMIT_REACHED',
Feb 25 08:11:09 AM	errno: 1226,
Feb 25 08:11:09 AM	sqlState: '42000',
Feb 25 08:11:09 AM	sqlMessage: 'User 'ufvomre2iqltqto' has exceeded the 'max_user_connections' resource (current value: 5)',
Feb 25 08:11:09 AM	sql: undefined
Feb 25 08:11:09 AM	}
Feb 25 08:11:11 AM	>>> Your service is live ▶
Feb 25 08:16:13 AM	>>> Detected service running on port 10000
Feb 25 08:16:13 AM	>>> Docs on specifying a port: <a href="https://render.com/docs/web-services#port-binding">https://render.com/docs/web-services#port-binding</a>

También se realizó el diagrama de actividades para CI/CD dentro de Lucid, basándose en las actividades realizadas y ya documentadas en este documento para verse de la siguiente manera



## Inconvenientes/Problemas y Solución

1. El primer problema fue el formato del archivo README.txt, el cual, aunque se le implementaron Headers para una visualización más placentera, estos no se veían reflejados dentro de la página principal del documento. Después de buscar por la causa de esto, se pudo resolver al cambiar la extensión del archivo, es decir, se cambió de README.txt a README.md.
2. El segundo problema que fue que al momento de querer desplegar la página en la nube con la configuración inicial que te da render arrojaba el siguiente error: "Error: Cannot find module '/opt/render/project/src/index.js'". La suposición inicial fue que no teníamos un archivo llamado index.js dentro del repositorio así que se consulto con el Dev si en realidad teníamos este archivo, después de corroborar que no contábamos con ningún archivo llamado index.js se siguió indagando dentro de los foros de Troubleshooting de Render; tras no dar con el problema se recurrió al uso de una inteligencia artificial para verificar si podía resolver el error. La primer sugerencia fue verificar que la estructura del proyecto fuera la correcta, con package.json fuera de la carpeta de source, tras verificar que la estructura era la correcta lo siguiente a corroborar eran los comandos con los que se estaba iniciando la página. El error radicaba ahí, donde el comando de inicio (Start Command) estaba con un valor predeterminado de algo así como "/src/index.js", este tuvo que ser cambiado a npm start para que la pagina pueda ser desplegada de manera correcta en la nube.
3. El tercer problema se presentó al momento de querer utilizar una API que mostraba el país de donde estabas visitando la página, el problema era que al mostrar el país del que se conectaba el usuario mostraba un valor default. Después de buscar la causa del problema se concluyó que esto se debía a la manera en la que el API forzaba una conexión HTTPS mientras que el despliegue de la página debía utilizar API que utilizaran HTTP, esto se resolvió cambiando a otra API que funcionalmente hacia lo mismo, pero sin la necesidad de que utilizara HTTPS. Al principio se pensó que la raíz del problema era el despliegue en Render, por lo que para hacer troubleshoot del problema también se probó desplegar la página por medio de Vercel (por eso se pueden ver comentarios de Vercel dentro de los PRs), una vez que se comprobó que el problema no era el lugar de donde se estaba desplegando la página sino la API esto se corrigió, pero se optó por conservar Vercel como despliegue secundario en caso de que Render llegue a fallar.

## Frida Sarahi Garza Gálvez

Cargo: Dev Frontend

Como había comentado, yo porte dos cargos, el de Lead y Dev de Frontend, es por ello que ahora detallare lo que realice en dicho cargo, como los HTML y la estilización de los mismos basado en lo brindado por UI/UX.

## Requerimientos

Los requerimientos que yo maneje en el cargo de Líder fueron los siguientes:

- *Creación de HTML Principal:* Se empeñará en desarrollar la página principal de la aplicación basándose en el formato que se es requerido por el UI/UX.
- *Creación de HTML Login:* Se empeñará en desarrollar la página de Login para la aplicación basándose en el formato que se es requerido por el UI/UX.
- *Creación de HTML Register:* Se empeñara en desarrollar la página Register de la aplicación basándose en el formato que se es requerido por el UI/UX.

- *Creación de HTML adminCursos:* Se empeña en generar el HTML donde se manejarán las operaciones que puede abarcar el usuario tipo Admin, donde se maneja las operaciones donde se edita, crea y elimina los cursos que solo el Admin puede manejar.
  - *Creación de HTML Mi Aprendizaje:* Se empeña en generar el HTML donde se manejan las operaciones que puede abarcar el usuario tipo Regular, el cual puede ver los cursos generales los cuales puede eliminar de su lista que se podrá mostrar en la misma página.
  - *Creación de HTML Cursos:* Se empeña en generar el HTML donde se manejan las operaciones en las cuales se puede ver los cursos generales los cuales puede añadir a su lista que se podrá mostrar en la página de Mi Aprendizaje.
  - *Implementación de CSS:* Se asegura de concordar el diseño CSS a los HTML que fueron creados con anterioridad para el Login, Principal, Register, RegularCursos y AdminCursos.

Estos requerimientos solo pudieron empezar tras la confirmación del diseño que es dado por UI/UX.

*Realización*

Mostrando los requerimientos que conllevan mi cargo de Dev, les especificare lo que tuve que realizar para poder cumplir con los diseños que fueron creados.

*HTML Principal*

Para el manejo del HTML de la página principal se tomó como base lo que manejaban en el avance del proyecto por parte de Brandon Carrión y Carolina Hermosillo, siendo manejada solo la eliminación de botones irrelevantes y el manejo de datos en el formulario para el contacto.

En esta sección de código nos empeñamos en especificar el nombre de la página, el icono de la web, los JS que haremos uso, y a la par empezamos con el poder tener un encabezado que contenga unas diversas secciones de la página para la navegación, haciendo solo cambios respecto a la eliminación de unos botones que no estaban integrados en el diseño.

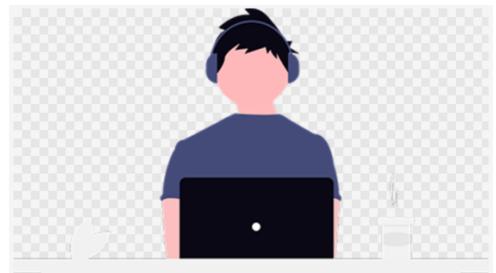
```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Hermcode</title>
        <link rel="website icon" type="png" href="/assets/logo_1.png">
        <!-- Importación de íconos de Font Awesome -->
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css" integrity="sha512-Evv84Mr4kqVGR
        <link rel="stylesheet" href="styles/styles.css">
        <script type="text/javascript" src="js/main.js"></script>
        <script src=""></script>
    </head>
    <body>
        <!-- MENU -->
        <!-- El header es una etiqueta para los encabezados que ayuda a un buen SEO en el navegador -->
        <header class="container py-3 w-100 menu">
            <div class="menu-logo" onclick="redirect(${window.location.origin})">
                
                <p class="text-uppercase quantico-font fw-bold fs-2">hermcode</p>
            </div>

            <!-- Etiqueta de navegación entre las diferentes secciones de la página -->
            <nav class="menu-navbar">
                <a class="text-yellow fw-bold menu-navbar_a" href="#cursos">CURSOS</a>
                <a class="text-yellow fw-bold menu-navbar_a" href="#contacto">CONTÁCTANOS</a>
                <button id="btnLoginPage" class="btn btn-secondary menu-navbar_btn" onclick="redirectLoginPage()">
                    Iniciar Sesión
                </button>
            </nav>
        </header>
    </body>
</html>
```

Por esta sección solo se hicieron cambios en la implementación de la imagen cambiada en el diseño, siendo la imagen donde un desarrollador está frente de una laptop.

```
<!-- MAIN SECTION -->
<section class="main-section">
  <div class="container main-section-container">
    <div class="main-section-message">
      <h1 class="text-uppercase lh-1 main-section-message__h1">Emplea a aprender</h1>
      <p>
        Somos una plataforma diseñada para programadores que quieran mejorar sus habilidades.
        Donde te ofrecemos cursos especializados adaptados a todos los niveles de experiencia.
        También puedes formar parte de nuestro equipo ofreciendo tus propios cursos para los demás.
      </p>
    </div>

    <figure class="w-100">
      <!-- la propiedad alt ayuda en la accesibilidad para las herramientas de voz descriptivas para personas con discapacidad visual -->
      
    </figure>
  </div>
</section>
```



```

<!-- CURSOS -->
<section id="cursos" class="container py-5">
  <h2 class="mb-3 fs-3 text-uppercase">Cursos en línea</h2>

  <!-- Contenedor padre de todos los cards -->
  <div class="cards">
    <!-- Card individual -->
    <div class="card">
      

      <div class="d-flex flex-column align-items-center" style="justify-content: space-between;">
        <div class="mb-2">
          <h3 class="fs-2 mb-1 text-yellow">Introducción a HTML y CSS</h3>
          <p>En este curso, descubrirás cómo estructurar páginas web con HTML y darles estilo con CSS...</p>
        </div>
      </div>
    </div>
    <!-- Card individual -->
    <div class="card">
      
      <div class="mb-2">
        <h3 class="fs-2 mb-1 text-yellow">Introducción a JavaScript</h3>
        <p>El mundo de la programación con JavaScript, el lenguaje clave para el desarrollo web...</p>
      </div>
    </div>
    <!-- Card individual -->
    <div class="card">
      
      <div class="mb-2">
        <h3 class="fs-2 mb-1 text-yellow">Fundamentos de Python</h3>
        <p>En este curso aprenderás sobre variables, estructuras de control, funciones y manejo de datos...</p>
      </div>
    </div>
  </div>
</section>

```

En esta parte del código solo fueron necesarios hacer cambios en las imágenes de uno de los cursos.



Junto a la eliminación de los botones que no tendrían funcionalidad alguna y no eran parte del diseño dado por UI/UX.

Por esta parte de código, mismamente hacemos cambio de imagen y damos una funcionalidad al formulario, que anteriormente no portaba.

```

<section id="contacto" class="container py-5 contacto">
  <div class="contacto-anuncio">
    <h3 class="fs-2 mb-3">SE PARTE DE NUESTRO EQUIPO!</h3>
    
  </div>

  <div class="contacto-form">
    <h2 class="mb-2 fs-2 text-uppercase">Contacto</h2>

    <form class="w-100" onsubmit="preventDefault(event)">
      <!-- Elemento que engloba el input y label -->
      <div class="mb-1 w-100">
        <label class="form-label" for="name">Nombre:</label>
        <input class="form-control fs-1 w-100" type="text" name="name" id="name" placeholder="Ingresa tu nombre">
      </div>
      <!-- Elemento que engloba el input y label -->
      <div class="mb-2 w-100">
        <label class="form-label" for="email">Correo electrónico:</label>
        <input class="form-control fs-1 w-100" type="email" name="email" id="email" placeholder="example@example.com">
      </div>
      <!-- Elemento que engloba el input y label -->
      <div class="mb-2 w-100">
        <label class="form-label" for="textarea">Mensaje:</label>
        <textarea id="textarea" name="textarea" class="form-control w-100" rows="3"></textarea>
      </div>

      <button id="btn-submit" class="btn btn-primary w-100 mt-2">Enviar</button>
    </form>
  </div>
</section>

```



## HTML Login

Para el manejo del HTML del Login mismamente se tomó como base lo que se manejaba en el avance del proyecto por parte de Brandon Carrión y Carolina Hermosillo, siendo muy pocos cambios realizados, como el eliminar las secciones en la barra superior donde salía respecto a Cursos y Contáctanos.

A la par que definir el icono del sitio web por el logo de la empresa.

Siendo también manejados los botones de hacer que vuelvan a la página principal o pues volver a redirigirse a la página de iniciar sesión, y dichas funciones son abarcadas en el archivo JS de la página.

```

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Hermode</title>
  <!-- Importación de iconos de Font Awesome -->
  <link rel="website icon" type="png" href="assets/logo_1.png">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css" integrity="sha512-"/>
  <link rel="stylesheet" href="styles/styles.css">
  <script type="text/javascript" src="js/login.js"></script>
  <script type="text/javascript" src="js/main.js"></script>
  <script src=""></script>
</head>
<body>
  <!-- MENU -->
  <!-- El header es una etiqueta para los encabezados que ayuda a un buen SEO en el navegador -->
  <header class="container py-3 w-100 menu">
    <div class="menu-logo" onclick="redirect(`${window.location.origin}`)">
      
      <p class="text-uppercase quanticoto-font fw-bold fs-2">hermcode</p>
    </div>
    <!-- Etiqueta de navegación entre las diferentes secciones de la página -->
    <nav class="menu-navbar">
      <button id="btnLogin" class="btn btn-secondary menu-navbar__btn" onclick="redirectToPrincipal()">
        | Inicio
      </button>
      <button id="btnLoginPage" class="btn btn-secondary menu-navbar__btn" onclick="redirectToLogin()">
        | Iniciar sesión
      </button>
    </nav>
  </header>

```

## HTML Register

Para la página de registro, use de base el formato del HTML de Login, en donde solo cambio el manejo de datos en el formulario.

```
<section class="main-section">
  <div id="register" class="container main-section-container">
    <form id="registerForm">
      <label for="username">Nombre:</label>
      <input type="text" id="username" name="username" maxlength="15" required>

      <label for="password">Contraseña:</label>
      <input type="password" id="password" name="password" required>

      <label for="role">Selecciona un rol:</label>
      <select id="role" name="role" required>
        <option value="normal" selected>Usuario Normal</option>
        <option value="admin">Administrador</option>
      </select>

      <input type="submit" value="Registrarse" id="inputLogin" class="btn-primary">
    </form>
  </div>
</section>
```

## HTML AdminCursos

Para la creación de la página del administrador, comencé con la definición de la barra superior en el cual contenga los botones para que pueda cerrar sesión, ver la información de su cuenta y a la par poder buscar algún curso en especial, dichos botones tendrán completa funcionalidad gracias al apoyo de Pedro De León.

```
<header class="container py-3 w-100 menu">
  <div class="menu-logo">
    
    <p class="text-uppercase quantico-font fw-bold fs-2">hermcode</p>
  </div>
  <nav class="menu-navbar">
    
    <button class="btn btn-secondary menu-navbar__btn" onclick="redirectToPrincipal()>Cerrar Sesión</button>
    <button id="btnUser" class="btn btn-secondary menu-navbar__btn">Mi cuenta</button>
  </nav>
</header>
```

Luego se genera una sección en donde se pueda notar el cómo manejo un botón en el cual a la hora de seleccionarlo deba mostrar un modal el cual le mostrará un formulario en donde debe ingresar el nombre, la descripción, la categoría y un link de la imagen junto al botón correspondiente de crear; después de ello también pueden ver que hay un contenedor el cual se empeñará en la búsqueda del curso pasado lo que ingrese el usuario. Yo en lo que me

```
<section class="main-section-botónCrear">
  <div class="container main-section-container">
    <button id="btnCrear" class="btn btn-terciary">+ Crear Curso</button>
  </div>
  <!--Contenedor donde se pueden buscar los cursos-->
  <div class="search-box" id="searchBox">
    <label>Busca el curso aquí:</label>
    <input type="text" id="searchInput">
  </div>
</section>
```

encargó es en generar una base en el cual mi compañero Pedro de león pueda hacer uso para poder darle completa funcionalidad en los JS.

Podemos ver que a la hora de presionar el botón de crear se le deberá mostrar al usuario este modal el cual podrá generar un curso y ser guardado en la base de datos.

Pueden ver como el formulario ahora maneja que solo se tenga límite de 15 caracteres y a la par la selección de un rol a la hora de registro y el botón de submit.

Mi compañero Pedro De León se encargó de la funcionalidad total en dicha sección.

```
<!-- MODAL CREAR CURSO -->
<div class="jw-modal" id="modalCrear">
  <div class="jw-modal-body container-form">
    <div class="d-flex justify-content-between align-items-center mb-3 w-100">
      <h2>Crear curso</h2>
      <i class="fa-solid fa-xmark fs-3" onclick="closeModal(document.getElementById('modalCrear'))"></i>
    </div>
    <form id="formCrearCurso">
      <label for="nombre">Nombre:</label>
      <input type="text" class="colorWhite w-100" id="nombreCurso" name="nombre" placeholder="Ingresa nombre del curso" maxlength="30" required>

      <label for="categoria">Categoria:</label>
      <input type="text" class="colorWhite w-100" id="categoriaCurso" name="categoria" placeholder="Ingresa categoria del curso" maxlength="15" required>

      <label for="descripcion">Descripción:</label>
      <textarea id="descripcionCurso" class="colorWhite w-100" name="descripcion" placeholder="Ingresa descripción del curso" maxlength="40" required></textarea>

      <label for="imagen">Imagen (link):</label>
      <input type="url" class="colorWhite w-100" id="imagenCurso" name="imagen" placeholder="Ingresa un link de imagen">

      <button type="submit" id="guardarCurso" class="btn btnCreate">Guardar</button>
    </form>
  </div>
</div>
```

```

<!-- MODAL EDITAR -->


<div class="jw-modal-body container-form">
    <div class="d-flex justify-content-between align-items-center mb-3 w-100">
      <h2>Editar curso</h2>
      <i class="fa-solid fa-xmark fs-3" onclick="closeModal(document.getElementById('modalEdit'))"></i>
    </div>
    <form id="formEditCurso">
      <label for="nombre">Nombre:</label>
      <input type="text" class="colorWhite w-100" id="nombreCursoEdit" required>

      <label for="categoria">Categoría:</label>
      <input type="text" class="colorWhite w-100" id="categoriaCursoEdit" required>

      <label for="descripcion">Descripción:</label>
      <textarea class="colorWhite w-100" id="descripcionCursoEdit" required></textarea>

      <label for="imagen">Imagen (link):</label>
      <input type="url" class="colorWhite w-100" id="imagenCursoEdit" required>

      <button type="submit" id="guardarCursoEdit" class="btn btnCrear">Guardar</button>
    </form>
  </div>


```

Por último tendremos el Tooltip que contiene los datos del perfil, desde el nombre hasta el tipo de usuario.

Otro de los modales que se manejan es el de editar, dicho botón viene dentro de un JS que deberá generar Pedro en el cual el usuario tenga la posibilidad de editar el curso específico, y podrá editar desde el nombre, la descripción, la categoría y la propia imagen.

```

<!-- Tooltip de perfil -->


<div class="perfil-tooltip-content">
    
    <div class="perfilUser__info">
      <h2 class="fw-bold textoh" id="perfilNombre">Nombre</h2>
      <p id="perfilTipoUser">Estudiante</p>
    </div>
  </div>


```

## HTML RegularCursos

Para la creación de la página la cual contenga todos los cursos a los que el usuario esta registrado, empezamos manejando en la barra superior la redirección para el HTML en el que se encuentra y el otro HTML en donde podrá ver a los cursos en los que podría registrarse, y al igual que el usuario administrador tiene la posibilidad de ver sus datos como su nombre y su tipo de usuario en el botón mi cuenta y también la posibilidad de cerrar sesión.

```

<header class="container py-3 w-100 menu">
  <div class="menu-logo">
    
    <p class="text-uppercase quantico-font fw-bold fs-2">hermcode</p>
  </div>
  <nav class="menu-navbar">
    
    <a class="text-yellow fw-bold menu-navbar__a" href="normalCursos.html">MI APRENDIZAJE</a>
    <a class="text-yellow fw-bold menu-navbar__a" href="cursos.html">CURSOS</a>
    <!--Falta agregar que se muestren los datos del usuario-->
    <button id="btnUser" class="btn btn-secondary menu-navbar__btn">
      Mi cuenta
    </button>
    <!--Falta agregar una forma de cerrar sesión, por el momento solo maneja que rediriga a pagina principal-->
    <button class="btn btn-secondary menu-navbar__btn" onclick="redirectToPrincipal()">
      Cerrar Sesión
    </button>
  </nav>
</header>

```

Por otra parte mismamente manejamos un Tooltip el cual mostraría la información del usuario cuando presiona el botón “mi cuenta”, y también manejamos el buscador que sobresale una vez presionamos el icono de la lupa.

Y toda la funcionalidad sería gracias a mi compañero Pedro De León. Aún así podemos ver que hay un contenedor el cual contendrá todos los cursos pero serán mandados desde un JS.

```

<div id="perfilTooltip" class="perfil-tooltip" style="display: none;">
  <div class="perfil-tooltip-content">
    
    <div class="perfilUser__info">
      <h2 class="fw-bold textoh" id="perfilNombre">Nombre</h2>
      <p id="perfilTipoUser">Estudiante</p>
    </div>
  </div>

```

## HTML Cursos

Para el último HTML que maneje se creó casi el mismo tipo de código, en el cual puedes manejar una barra superior con la redirección correspondiente ya sea para el html donde puedes ver los cursos a los que estás suscritos o ver los cursos a los que te puedes inscribir, mismamente portando los botones para poder ver los datos de tu cuenta, el buscador y el cierre de sesión.

En esta sección de código puedes ver cómo tenemos varios contenedores, uno de ellos portando la búsqueda de los cursos, otro manejando la muestra de los datos del usuario y el último el cual conllevará el manejo de la muestra de todos los cursos en la página, siendo así que la mayoría de los HTML creados respecto a la de demostración de cursos contengan el mismo contenedor para la demostración de estos.

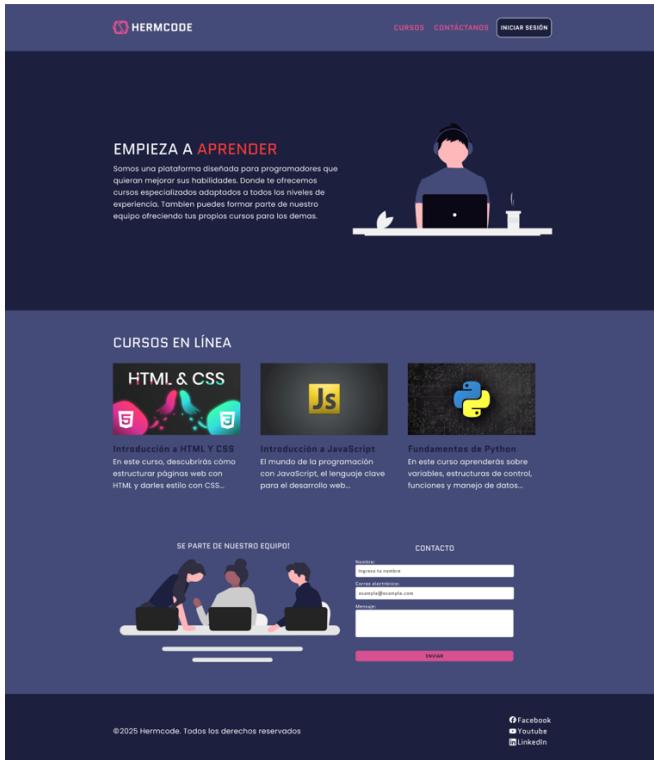
```
<nav class="menu-navbar">
  
  <a class="text-yellow fw-bold menu-navbar_a" href="#">MI APRENDIZAJE</a>
  <a class="text-yellow fw-bold menu-navbar_a" href="#">CURSOS</a>
  <button id="btnUser" class="btn btn-secondary menu-navbar_btn">Mi cuenta</button>
  <button class="btn btn-secondary menu-navbar_btn" id="logoutBtn">Cerrar Sesión</button>
</nav>
```

```
<section id="cursoNormal" class="main-section-normal">
  <h3 class="section-cursosIns fs-5">Cursos</h3>
  <!--Contenedor donde se pueden buscar los cursos-->
  <div class="search-box" id="searchBox">
    <label>Busca tu curso aquí:</label>
    <input type="text" id="searchInput"/>
  </div>
  <!--Contenedor donde se muestran los datos del usuario-->
  <div class="perfil-box" id="perfilContainer" style="display: none;">
    <div class="perfilUser_gen">
      
      <div class="perfilUser_info">
        <h2 class="fw-bold textoH" id="perfilNombre">Nombre</h2>
        <p id="perfilTipoUser">Tipo de usuario</p>
      </div>
    </div>
  </div>
  <div class="container main-section-normal">
    <div class="main-container-cursos">
      <div class="cursos-flex" id="cursos-container"></div>
      <div class="pagination-dots" id="pagination-dots"></div>
    </div>
  </div>
</section>
```

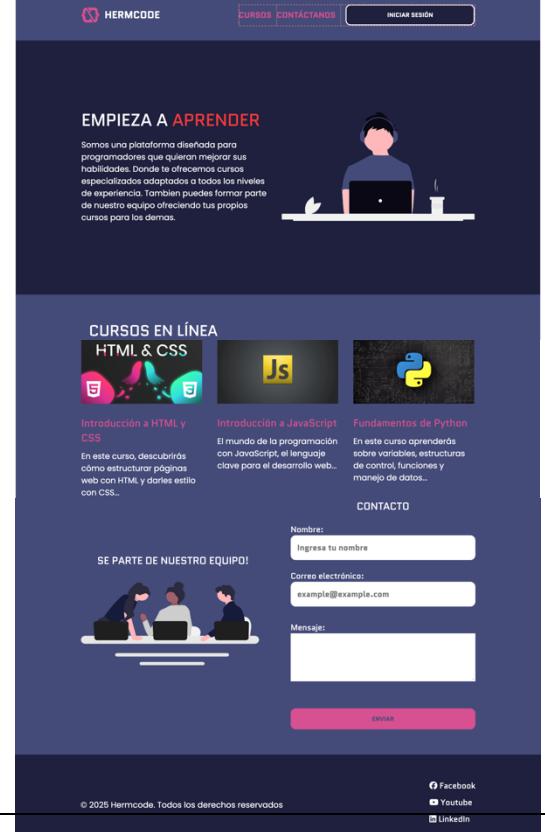
## Implementación del CSS

Por último tenemos la implementación del CSS en los HTML, debido a la gran extensión que se tiene en el CSS se decidió por hacer una muestra comparativa entre lo que se dio en él firma y lo que se entregó como resultado final.

### Vista en Figma



### Vista en Resultado Final



## Vista en Figma

The first screenshot shows the 'EMPIEZA A APRENDER' section with a dark background, featuring a person at a desk and text about the platform's purpose.

The second screenshot shows the 'CURSOS EN LÍNEA' section with a purple background, featuring a large 'HTML & CSS' logo and a brief introduction to the course.

The third screenshot shows the 'SE PARE DE NUESTRO EQUIPO!' section with a dark background, featuring three people at a table and a contact form.

## Vista en Resultado Final

The first screenshot shows the 'EMPIEZA A APRENDER' section with a dark background, featuring a person at a desk and text about the platform's purpose.

The second screenshot shows the 'CURSOS EN LÍNEA' section with a purple background, featuring a large 'HTML & CSS' logo and a brief introduction to the course.

The third screenshot shows the 'SE PARE DE NUESTRO EQUIPO!' section with a dark background, featuring three people at a table and a contact form.

## Vista en Figma

The page has a dark blue header with the 'HERMCODE' logo and navigation links for 'INICIO' and 'INICIAR SESIÓN'. The main content area displays a message: 'Te loguearás desde: México'. It contains fields for 'Nombre:' and 'Contraseña:', a 'Iniciar sesión' button, and a link 'No tienes cuenta? Regístrate aquí'. At the bottom are social media links for Facebook, YouTube, and LinkedIn.

## Vista en Resultado Final

The page has a dark blue header with the 'HERMCODE' logo and navigation links for 'INICIO' and 'INICIAR SESIÓN'. The main content area displays a message: 'Te loguearás desde: México'. It contains fields for 'Nombre:' and 'Contraseña:', a 'Iniciar sesión' button, and a link '¿No tienes cuenta? Inicia sesión aquí'. At the bottom are social media links for Facebook, YouTube, and LinkedIn.

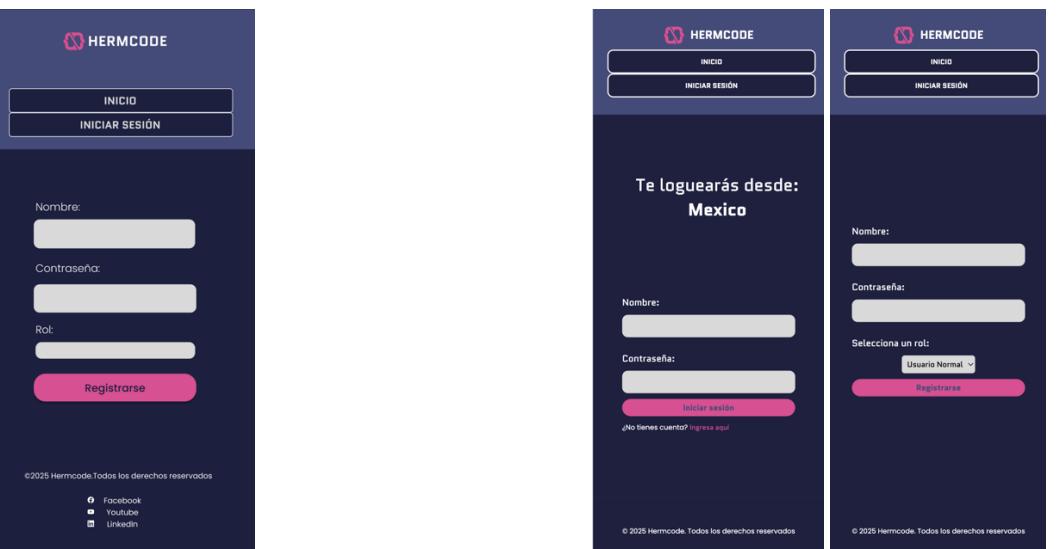
## Vista en Figma

The page has a dark blue header with the 'HERMCODE' logo and navigation links for 'INICIO' and 'INICIAR SESIÓN'. The main content area displays a message: 'Te loguearás desde: México'. It contains fields for 'Nombre:', 'Contraseña:', and 'Rut:', a 'Registrarse' button, and a link 'No tienes cuenta? Regístrate aquí'. At the bottom are social media links for Facebook, YouTube, and LinkedIn.

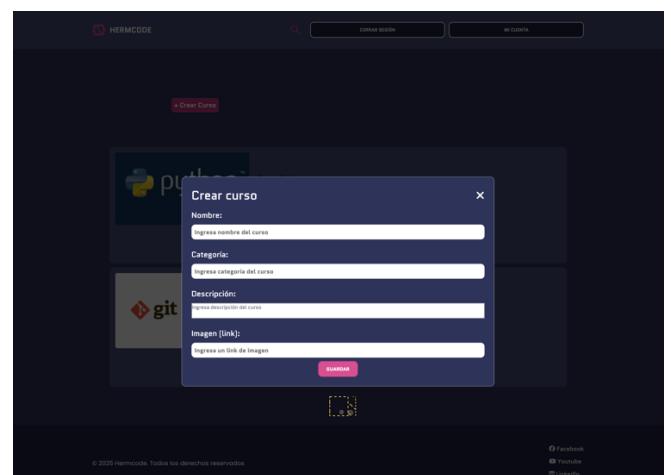
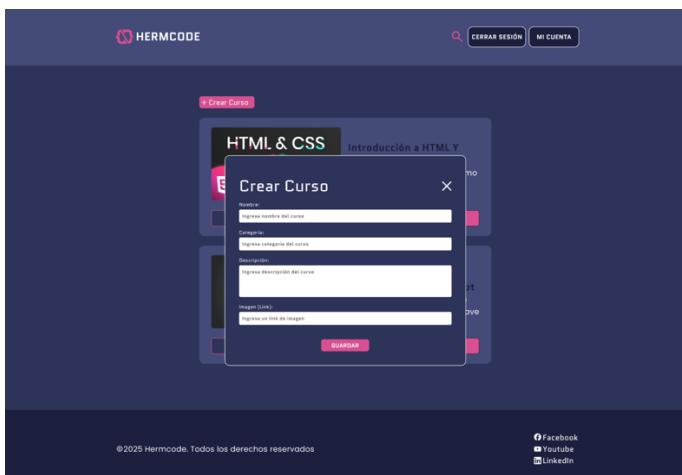
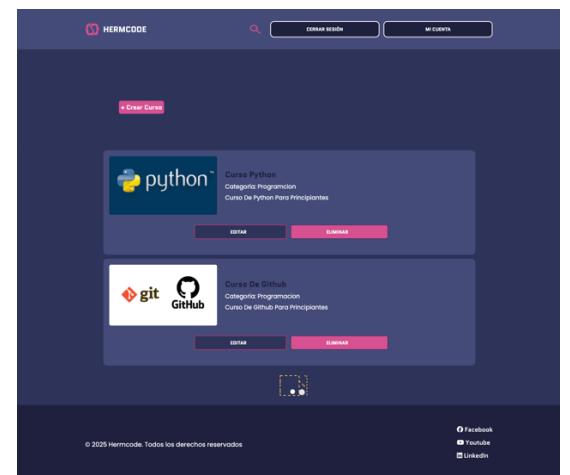
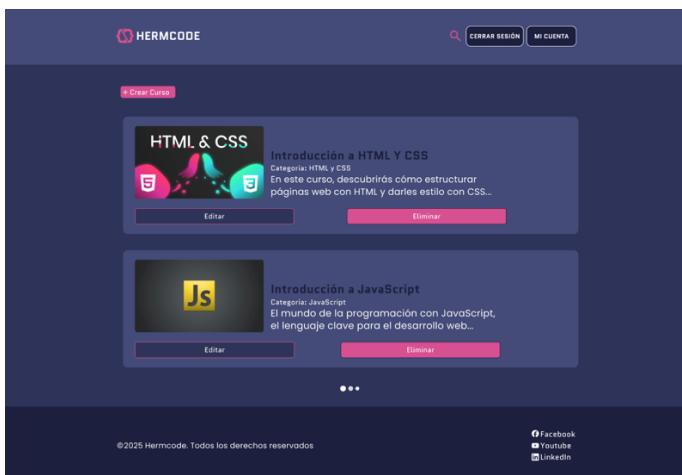
## Vista en Resultado Final

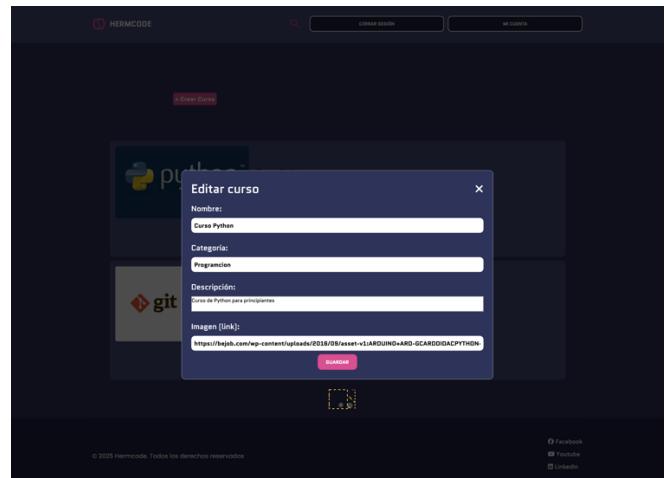
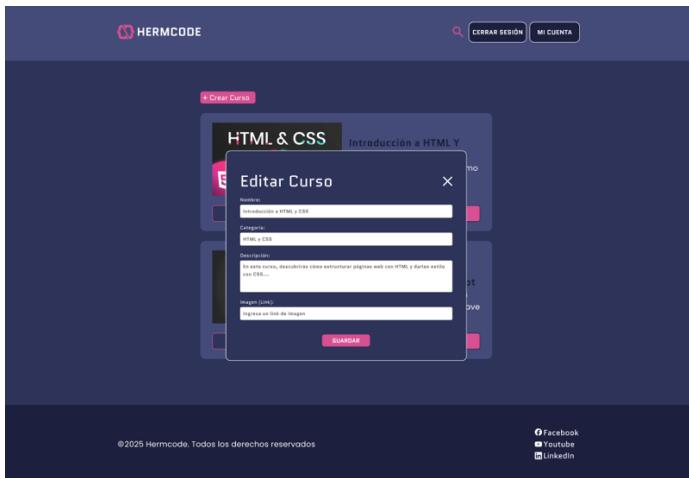
The page has a dark blue header with the 'HERMCODE' logo and navigation links for 'INICIO' and 'INICIAR SESIÓN'. The main content area displays a message: 'Te loguearás desde: México'. It contains fields for 'Nombre:', 'Contraseña:', and 'Selecione un rol:' (with a dropdown menu showing 'Usuario Normal'), a 'Registrarse' button, and a link '¿No tienes cuenta? Inicia sesión aquí'. At the bottom are social media links for Facebook, YouTube, and LinkedIn.

## Vista en Figma

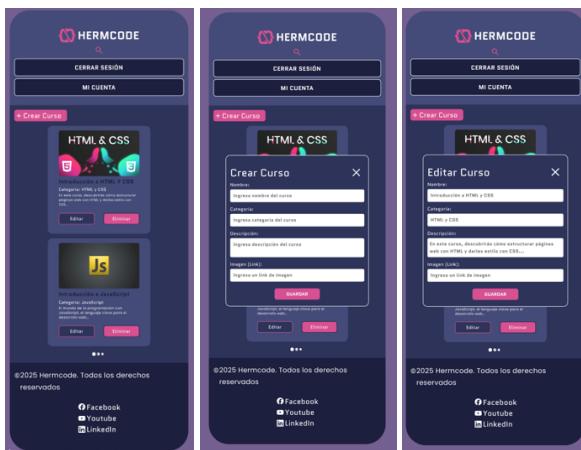


## Vista en Figma

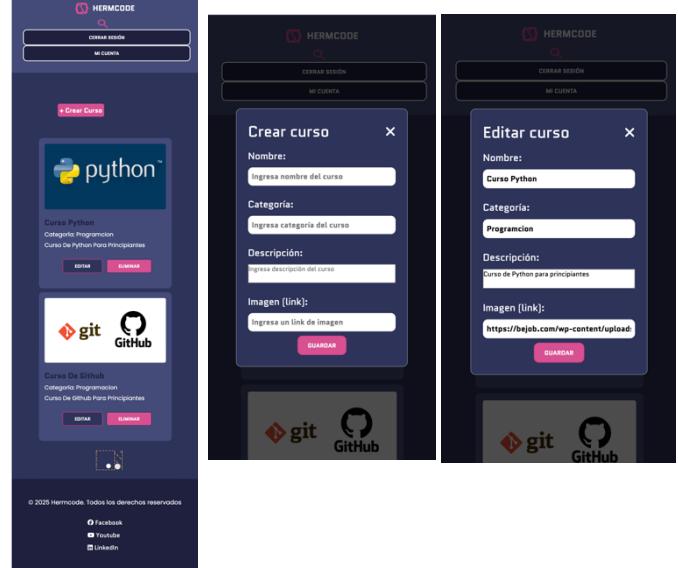




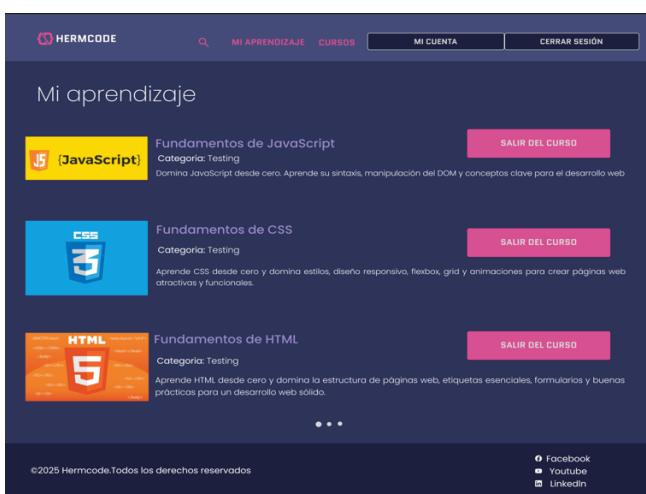
Vista en Figma



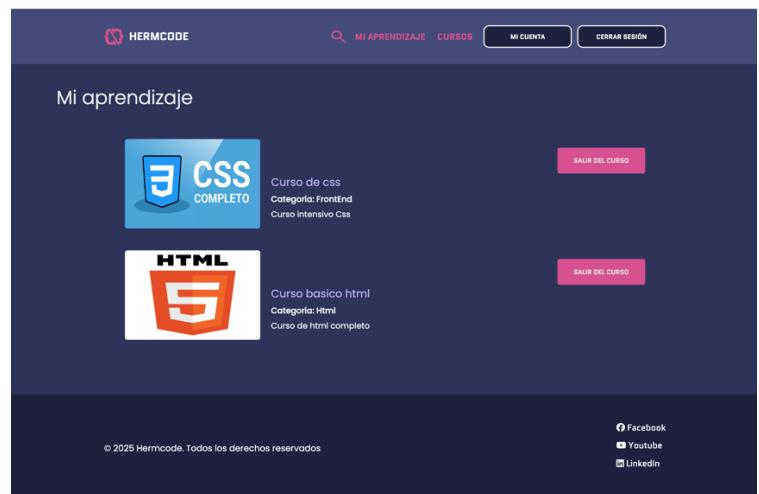
Vista en Resultado Final



Vista en Figma



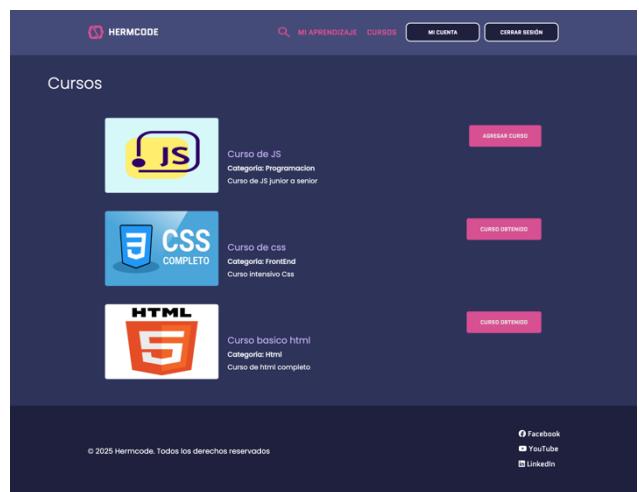
Vista en Resultado Final



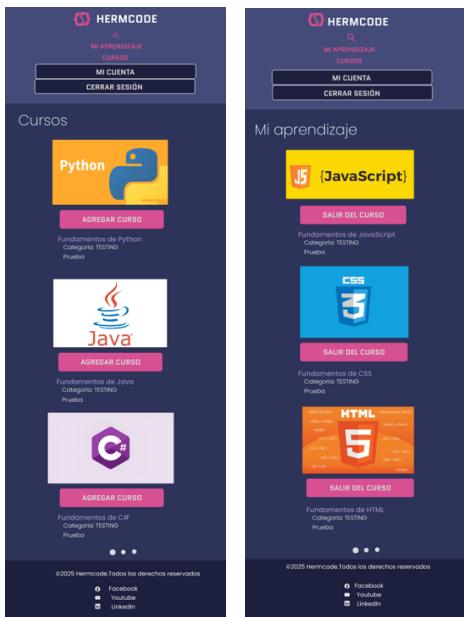
## Vista en Figma



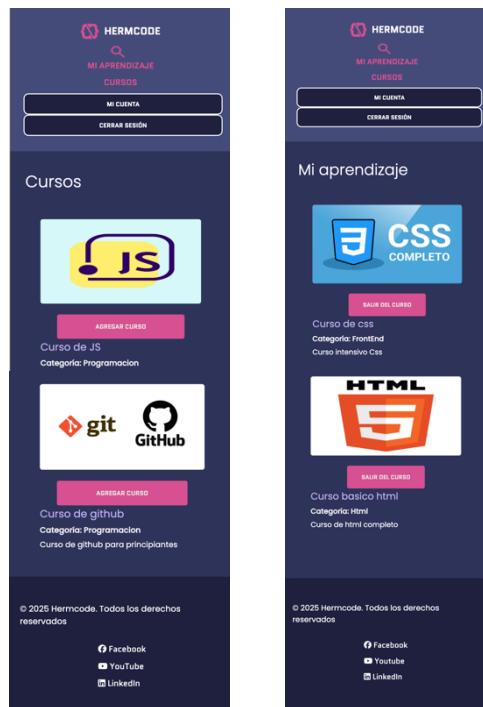
## Vista en Resultado Final



## Vista en Figma



## Vista en Resultado Final



## Inconvenientes/Problemas y Solución

Tal vez uno de los problemas que nos surgieron fueron en conjunto con el CI/CD, ya que a la hora de configurar el Pipeline, hubo problemas en la demostración de la correcta funcionalidad de los test, y a la par del despliegue en render.

Para la corrección completa de dicho Pipeline, el cual confirma solamente el despliegue en Vercel y la funcionalidad total de las pruebas unitarias, a continuación una muestra de cómo quedó el código del Pipeline y su correcto funcionamiento.

```

FridaGarzaG LEAD-Frida-Corrección en las pruebas

Code Blame 31 lines (24 loc) · 606 Bytes Code 65% faster with GitHub Copilot

1 name: Pruebas y Deploy
2
3 on:
4   push:
5     branches:
6       - main
7
8   jobs:
9     deploy:
10       runs-on: ubuntu-latest
11
12       steps:
13         - name: Checkout del repositorio
14           uses: actions/checkout@v4
15
16         - name: Configurar Node.js
17           uses: actions/setup-node@v4
18           with:
19             node-version: 18
20
21         - name: Instalar dependencias
22           run: npm install
23
24         - name: Ejecutar pruebas
25           run: npx jest --detectOpenHandles
26
27         - name: Deploy a Vercel
28           run: npx vercel --prod ${{ secrets.VERCEL_TOKEN }} --yes
29
30         - name: Desplegar en Vercel
31           run: npx vercel --prod ${{ secrets.VERCEL_TOKEN }} --yes

```

Queda aclarar que las pruebas test fallan cuando hay múltiples conexiones en la base de datos, esto se debe a que la base de datos es gratuita no podemos dar una capacidad máxima mayor a 5 y es por eso que en momentos de cuando subíamos la branch había inconvenientes debido a las múltiples conexiones que conllevaba la página en ese momento.

## Pedro Francisco De León Salazar

Cargo: Dev Backend

*Yo como Dev me encargué de todo el Backend de la página, conectando la BD con node, express y demás, para mostrar todo lo que hay en ella dentro del Frontend, también me encargué de hacer posible el CRUD en la BD mediante el Frontend. Me encargue de hacer posible los registros, logins, inscripciones a cursos, creación de cursos, etc.*

### Requerimientos

- **Error login- 03:** Se presentan errores en la conexión para login y register.
- **Error API, despliegue en la nube:** No se especifica la correcta implementación de la API en el entorno de la nube.
- **Error vista admin -02:** Errores detectados en la interfaz de administración.
- **Error vista usuario estándar:** Problemas en la vista del usuario estándar que afectan la experiencia.
- **Error Login Campo usuario -02:** Se encontraron problemas en la validación de mayúsculas y minúsculas en los campos del login.
- **Error Login Campo usuario:** Se creó un usuario de prueba con fallos en la validación de formato, dando resultados incorrectos.
- **Error ficheros- 02:** No se encuentra Node.js correctamente configurado y falta la ruta en scripts.
- **Estructura de MySQL:** Creación de un archivo que defina la estructura de la base de datos en MySQL, incluyendo el código SQL para generar las tablas.
- **Error ficheros:** No se ignoraron correctamente los módulos de Node.js en GitHub.
- **Reporte Individual de Pedro:** Elaboración de un informe sobre los requerimientos trabajados en el proyecto.
- **Implementación del Framework seleccionado:** Configurar rutas dinámicas en el frontend y establecer la gestión de sesión de usuarios.
- **Selección de un Framework:** Elegir un framework frontend (React, Angular o Vue.js) para mejorar la estructura y experiencia del usuario.
- **Integración de la API Externa seleccionada:** Implementación de la API seleccionada, manejando respuestas y validaciones.
- **Seleccionar API Externa:** Elegir una API externa REST o GraphQL que complemente la funcionalidad de la aplicación.
- **Implementación de un controlador de errores:** Incorporar un sistema centralizado para manejar excepciones dentro de la aplicación.
- **Manejo de paginación en los endpoints principales:** Implementar paginación en las consultas de cursos para optimizar el rendimiento.
- **Filtro en DELETE de Cursos para Admin:** Aplicar filtros de seguridad en la eliminación de cursos para que solo los administradores puedan realizarlos.
- **Filtro en UPDATE de Cursos para Admin:** Añadir validaciones en la actualización de cursos, asegurando modificaciones seguras.
- **Filtro en READ de Cursos:** Aplicar restricciones de acceso a cursos según el rol del usuario.

- **Filtro en CREATED de usuarios:** Validar si un usuario ya existe antes de permitir su registro.
- **Manejo de filtro en operaciones CRUD:** Garantizar la extracción de datos específicos mediante filtros en cada operación CRUD.
- **Generación de Middleware para rutas protegidas:** Implementar middleware para proteger rutas con autenticación basada en JWT.
- **Manejo de JavaScript:** Utilizar JavaScript para manipulación del DOM y gestión de peticiones al servidor.
- **Realización de debugging:** Aplicar técnicas de depuración con console.log, node inspect O herramientas de depuración en el IDE.
- **Generación de Middleware para manejar errores y validaciones:** Creación de un middleware que gestione errores de validación, autenticación y autorización.
- **Creación de DELETE para cursos registrados:** Implementar la funcionalidad de eliminación de cursos a los que los usuarios están inscritos.
- **Creación de READ para cursos registrados:** Permitir la visualización de los cursos en los que el usuario se ha registrado.
- **Creación de CREATE para cursos registrados:** Implementar la inscripción de usuarios en cursos con su respectiva fecha de registro.
- **Funciones y permisos para Admin o Regular:** Definir permisos específicos para administradores y usuarios regulares, asegurando accesos adecuados.
- **Implementar una autenticación al Admin/Regular:** Implementar JWT u OAuth para gestionar el acceso según el rol del usuario.
- **Implementación de autenticación:** Aplicar autenticación basada en JWT para restringir acciones en la aplicación.
- **Creación de CREATE para usuarios:** Habilitar la creación de cuentas de usuario con validaciones.
- **Creación de DELETE para cursos:** Permitir la eliminación de cursos en la base de datos únicamente por administradores.
- **Creación de UPDATE para cursos:** Implementar la actualización de cursos con validación de datos de entrada.
- **Creación de READ para cursos:** Habilitar la consulta de cursos almacenados en la base de datos.
- **Creación de CREATE para cursos:** Permitir la creación de nuevos cursos en la base de datos.
- **Creación de servidor en Express.js:** Configurar un servidor básico utilizando Express.js.
- **Configuración de variables de entorno:** Manejar variables sensibles en un archivo de configuración seguro.
- **Creación de base de datos en la nube:** Implementar la base de datos en un servicio en la nube.
- **Instalación de dependencias:** Instalar las dependencias esenciales como Express, JWT, bcrypt y otras necesarias.
- **Selección de base de datos:** Elegir entre MongoDB, MySQL u otra base de datos adecuada para la aplicación.
- **Crear proyecto en Node.js:** Inicializar un nuevo proyecto con Node.js y su estructura base.
- **Generación de pruebas de aceptación:** Desarrollar pruebas para evaluar si la aplicación cumple con las expectativas del usuario.
- **Generación de pruebas de usabilidad:** Medir la facilidad de uso de la aplicación y la interacción del usuario con sus funciones.
- **Generación de pruebas de seguridad:** Validar que el sistema de autenticación y permisos funciona correctamente.

- **Generación de pruebas unitarias para la robustez de la aplicación:** Implementar pruebas unitarias en los módulos clave del sistema.
- **Subir archivos a GitHub:** Realizar la carga y mantenimiento del código fuente en un repositorio en GitHub.

## Realización

- **Selección de base de datos**

Se seleccionó MySQL como base de datos, debido a su gran facilidad al manejar la BD.

- **Creación de base de datos en la nube**

Se creó la base de datos en clever cloud, la cual nos arrojó los siguientes datos para ingresar en ella.

Host	<input type="text" value="bbz7wlsdy5yaf7ocvlva-mysql.services.clever-cloud.com"/>
Database Name	<input type="text" value="bbz7wlsdy5yaf7ocvlva"/>
User	<input type="text" value="ufvomre2iqltqutc"/>
Password	<input type="password" value="JfEy8nMBIVlNIQcuASay"/> 
Port	<input type="text" value="3306"/>

- **Creación de proyecto en Node.js**

Al tener como base un proyecto usado anteriormente en clase, ya estaba instalado NodeJS.

- **Instalación de dependencias**

La única nueva dependencia que instalé fue “mysql2”, para hacer uso de MySQL.

- **Configuración de variables de entorno**

```
const mysql = require('mysql2');

const db = mysql.createPool({
  host: process.env.DB_HOST || 'bbz7w1sdy5yaf7ocvlva-mysql.services.clever-cloud.com',
  user: process.env.DB_USER || 'ufvomre2iqltqutc',
  password: process.env.DB_PASSWORD || 'Y9m3AeTqyxFTL2l0ZPZn',
  database: process.env.DB_NAME || 'bbz7w1sdy5yaf7ocvlva',
  port: process.env.DB_PORT || 3306,
  waitForConnections: true,
  connectionLimit: 10,
  queueLimit: 0
});

db.getConnection((err, connection) => {
  if (err) {
    console.error('Error en la conexión a MySQL:', err);
  } else {
```

```

        console.log('Conectado a MySQL');
        connection.release();
    }
});

module.exports = db;

const app = express();
process.env.JWT_SECRET = process.env.JWT_SECRET || 'secreto_super_seguro';
const PORT = process.env.PORT || 3000;

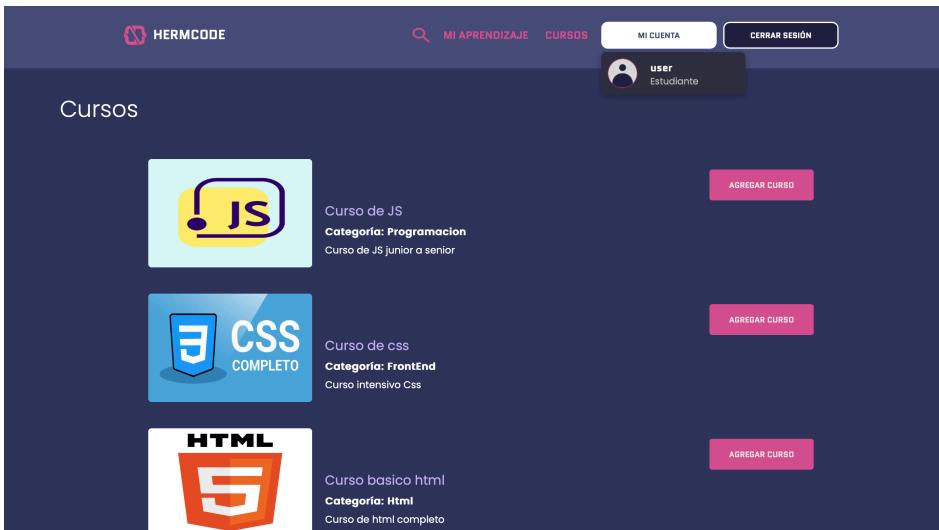
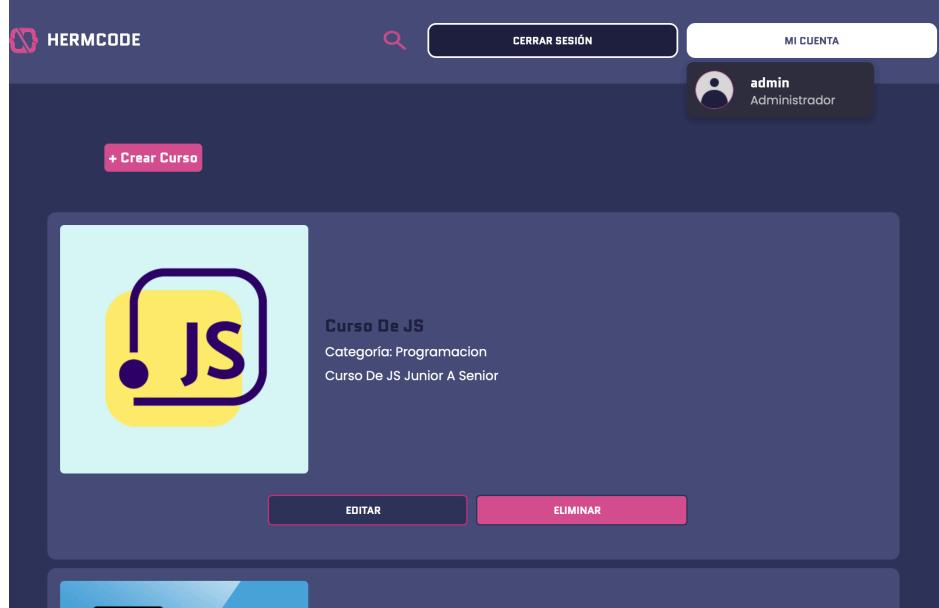
```

- **Creación de servidor en Express.js**

Al tener como base un proyecto usado anteriormente en clase, ya estaba creado el servidor.

- **Implementación de autenticación y autorización para Admin/Regular**

Al ingresar como administrador se le dará el rol de admin, por lo que automáticamente irá a adminCursos.html y el token le dará acceso de admin, pero al ingresar como usuario normal / regular se le dará el rol de normal, por lo que irá a cursos.html y normalCursos.html



- **Generación de middleware para rutas protegidas**

En las siguientes capturas podemos ver como al estar logueado y con token vigente podemos realizar diversas tareas, pero al borrar el token en localStorage se activará la protección y le pedirá al usuario loguearse.

The screenshot shows a web application interface for managing courses. On the left, there's a list of courses with icons, names, categories, and edit/delete buttons. On the right, the Chrome DevTools Network tab displays two fetch requests for 'paginated?page=1' and 'paginated?page=2'. Both requests are successful (status 200) and are identified as 'fetch' type requests initiated by 'adminCursos.js:93'.

This screenshot shows the same browser session after the token has been cleared from localStorage. The application now displays a login form instead of the course management interface, indicating that the user is no longer authenticated.

- **Generación de middleware para manejar errores y validaciones**

En register el usuario no puede dejar campos vacíos.

The screenshot shows a registration form with three fields: 'Nombre', 'Contraseña', and 'Selección un rol'. The 'Nombre' field is empty and has a red border, with an error message 'Completa este campo' above it. The 'Contraseña' field is also empty and has a red border. The 'Selección un rol' dropdown menu is open, showing 'Usuario Normal' as the selected option.

En adminCursos, el administrador no puede dejar ningún campo vacío, EXCEPTO el campo de la url de la imagen, ya que si no carga una imagen aparecerá una imagen por default, pero si ingresa una url con un formato incorrecto también le pedirá ingresar una url correcta

**Crear curso**

Nombre:

Categoría:

Descripción:

Imagen [link]:

**Crear curso**

Nombre:

Categoría:

Descripción:

Imagen [link]:

También al intentar dejar un campo vacío en editar curso nos marcará dicho error.

**Editar curso**

Nombre:

Categoría:

Descripción:

Imagen [link]:

**GUARDAR**

- **Implementación de un controlador de errores**

A continuación se verán dos ejemplos donde se manejan los errores.

localhost:3000 dice  
Debes iniciar sesión primero.

Clave	Valor
binance- <a href="http://localhost:3000">http://localhost:3000</a>	0
ethereum- <a href="http://localhost:3000">http://localhost:3000</a>	{"chainId": "0x1"}
isWhitelisted	false
logLevel	SILENT
name	admin
trustCacheTimestamp	{"timestamp": 1740864302581}

(timestamp: 1740864302581)

("error": "Ruta no encontrada")

## • Manejo de JavaScript y debugging

A continuación se verá como se manejan el debugging, y en diversas partes del código se ven las respuestas en el console.log

```
Cursos inscritos en el backend: [
  {
    id_user_course: 69,
    id_course: 2,
    title: 'Curso de JS',
    description: 'Curso de JS junior a senior',
    category: 'Programacion',
    image_url: 'https://www.cursoenvideo.com/wp-content/uploads/bb-plugin/cache/javascript-circle-dc92b56e539139ec2bf42ebf8393803c-5d48cb37edbef.jpg'
  }
]
```

## • Creación de CRUD para usuarios:

- **CREATE**

Campos de la tabla antes de agregar el nuevo usuario.

	<b>id_user</b>	<b>username</b>	<b>password</b>	<b>role</b>
	1	admin	\$2b\$10\$s38i0LfroHRGcK8VzW1Zq.7phNyIXV7...	admin
	2	user	\$2b\$10\$THsRu4cmJ.2w4lc7Qmxlmukshw/wEs...	normal
	3	FridaNormal	\$2b\$10\$dnNnjuNRLyPnUftxgSfte4vE4fhblrWO...	normal
	4	FridaAdmin	\$2b\$10\$MCkB/OU9zVgxu9aEiE/D.u3VP117S...	admin
	5	admin2	\$2b\$10\$UVsJpkcsn1z4.pwuJzceQkYJh74Av3...	admin
	6	admin22	\$2b\$10\$ijry97GPPwG9HERV.ViheOSbAd.NCC...	normal
	7	test	\$2b\$10\$/BZbMxAooaf8osNV/lqmt.0VYGD0f8...	normal
	34	testing	\$2b\$10\$Gnylra9.9INi8/LYO91avelDov3l0bzeCc...	normal
	51	test22	\$2b\$10\$WqU1DxyuFQjN7O2ZHsMDLO8GdRG...	admin
	52	test333	\$2b\$10\$1.hPx33UxySjrFXmdkrFUuzgbGhBEU...	normal
	53	admin444	\$2b\$10\$IQMzPIXEKmjlsLa6XtO99le82p3epnaV...	admin
	54	admin4444	\$2b\$10\$es8VvWyEB6x3e5SQxPye0eyr11etg2...	normal
	85	123	\$2b\$10\$1OUVZ/OcBDelwLPE2JBKR.ctLgsNjx...	normal
	90	prueba	\$2b\$10\$KNpxUNjmmfn0DiPf7rmi/O1K6nqtBeT...	normal

Nombre:

Contraseña:

Selecciona un rol:

**Registrarse**

Registro exitoso

[Cerrar](#)

	<b>id_user</b>	<b>username</b>	<b>password</b>	<b>role</b>
	1	admin	\$2b\$10\$s38i0LfroHRGcK8VzW1Zq.7phNyIXV7...	admin
	2	user	\$2b\$10\$THsRu4cmJ.2w4lc7Qmxlmukshw/wEs...	normal
	3	FridaNormal	\$2b\$10\$dnNnjuNRLyPnUftxgSfte4vE4fhblrWO...	normal
	4	FridaAdmin	\$2b\$10\$MCkB/OU9zVgxu9aEiE/D.u3VP117S...	admin
	5	admin2	\$2b\$10\$UVsJpkcsn1z4.pwuJzceQkYJh74Av3...	admin
	6	admin22	\$2b\$10\$ijry97GPPwG9HERV.ViheOSbAd.NCC...	normal
	7	test	\$2b\$10\$/BZbMxAooaf8osNV/lqmt.0VYGD0f8...	normal
	34	testing	\$2b\$10\$Gnylra9.9INi8/LYO91avelDov3l0bzeCc...	normal
	51	test22	\$2b\$10\$WqU1DxyuFQjN7O2ZHsMDLO8GdRG...	admin
	52	test333	\$2b\$10\$1.hPx33UxySjrFXmdkrFUuzgbGhBEU...	normal
	53	admin444	\$2b\$10\$IQMzPIXEKmjlsLa6XtO99le82p3epnaV...	admin
	54	admin4444	\$2b\$10\$es8VvWyEB6x3e5SQxPye0eyr11etg2...	normal
	85	123	\$2b\$10\$1OUVZ/OcBDelwLPE2JBKR.ctLgsNjx...	normal
	90	prueba	\$2b\$10\$KNpxUNjmmfn0DiPf7rmi/O1K6nqtBeT...	normal
	91	pedro	\$2b\$10\$.d4WEhZmQZX2kXO5nMC.CKN4o8...	normal

## • Creación de CRUD para cursos registrados:

- **CREATE**
- **READ**
- **UPDATE**
- **DELETE**

El admin podrá crear un nuevo curso mediante el botón de Crear Curso y le aparecerá la siguiente ventana.

**Crear curso**

Nombre:

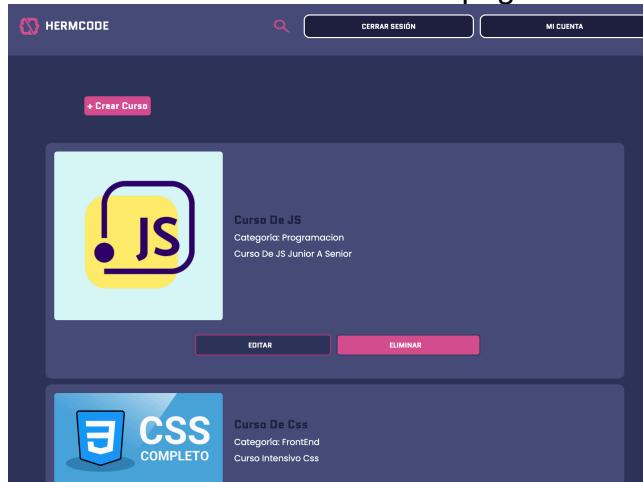
Categoría:

Descripción:

Imagen (link):

**GUARDAR**

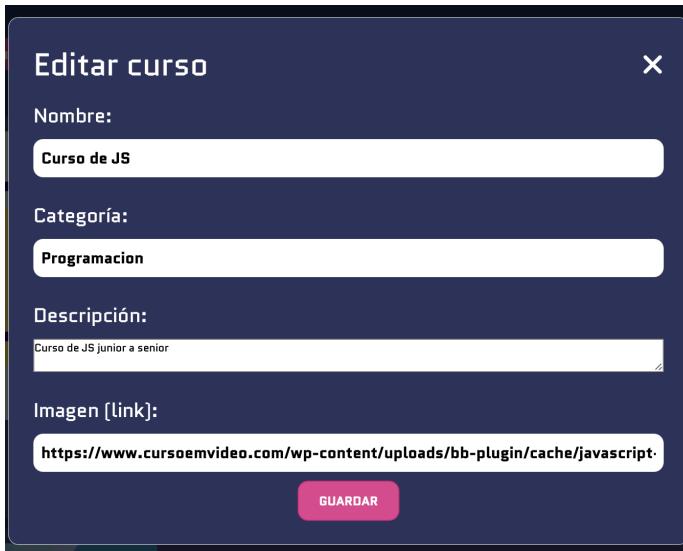
El admin podrá ver los cursos creados mediante su página.



El admin podrá editar o eliminar mediante dos botones que aparecen en cada curso.



Al darle clic a Editar aparecerá la siguiente ventana donde el usuario podrá editar cualquier dato.



Nombre:  
Curso de JS

Categoría:  
Programacion

Descripción:  
Curso de JS junior a senior

Imagen [link]:  
<https://www.cursoenvideo.com/wp-content/uploads/bb-plugin/cache/javascript->

GUARDAR

- **Manejo de filtros en operaciones CRUD:**

- **Filtro en READ de cursos**
- **Filtro en UPDATE de cursos para Admin**
- **Filtro en DELETE de cursos para Admin**

Se implementó una ventana de búsqueda para que el usuario pueda reconocer fácilmente los cursos a los que esta inscrito (READ), o actualizar/eliminar si es admin (DELETE), al igual que puede salir de un curso si es normal/regular (DELETE).

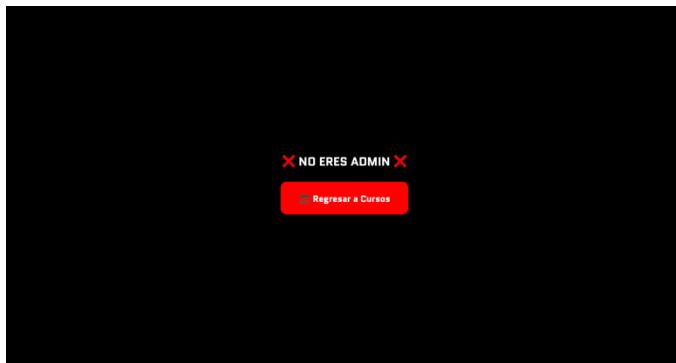
The screenshot shows a course card for "Curso De JS". The card includes a yellow icon with a blue "JS" logo, the title "Curso De JS", the category "Programacion", and the subtitle "Curso De JS Junior A Senior". Below the card are two buttons: "EDITAR" and "ELIMINAR". At the top of the page, there is a search bar with the placeholder "Busca el curso aquí:" and a magnifying glass icon. Other top navigation items include "CERRAR SESIÓN" and "MI CUENTA". A pink button "+ Crear Curso" is located in the top left corner.

The screenshot shows a list of courses under the heading "Cursos". It displays two course cards: "Curso de JS" (with a yellow icon) and "Curso de java" (with a red icon). Each card provides details such as the title, category ("Programacion"), subtitle, and a "CURSO OBTENIDO" or "AGREGAR CURSO" button. The top navigation bar includes "HERM CODE", "MI APRENDIZAJE", "CURSOS", "MI CUENTA", and "CERRAR SESIÓN". A search bar with the placeholder "Busca tu curso aquí:" is also present.

The screenshot shows a section titled "Mi aprendizaje". It features a course card for "Curso de JS" with a yellow icon, including the title, category, and subtitle. A pink button "SALIR DEL CURSO" is visible. The top navigation bar includes "HERM CODE", "MI APRENDIZAJE", "CURSOS", "MI CUENTA", and "CERRAR SESIÓN". A search bar with the placeholder "Busca tu curso aquí:" is also present.

- **Funciones y permisos para Admin o Regular**

Si un admin intenta acceder a cursos.html o normalCursos.html le aparecerá que no tiene ese rol, al igual que si un usuario normal / regular intenta acceder a adminCursos.html le aparecerá el mismo mensaje.



- **Selección e integración de la API externa seleccionada**

Se escogió una API que extrae el país donde está ubicado el usuario, para después mostrarlo en la página de login y decirle desde que país se está logueando.

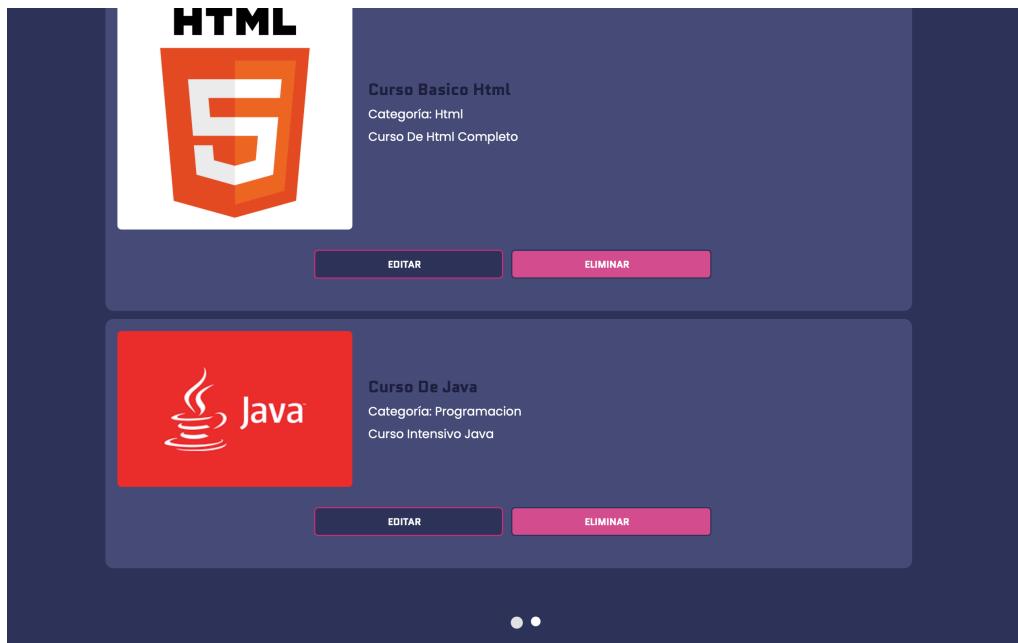


- **Implementación del framework seleccionado**

Se seleccionó el framework React.

- **Manejo de paginación en los endpoints principales**

En adminCursos se mostrarán los cursos de 4 en 4 y se cambiará de página mediante los puntos.



En las demás páginas es scroll infinito, pero cargará de 4 en 4 para no saturar la carga de la página, por lo que hasta que el usuario no llegue al último curso no se mostrarán más de 4.

- **Estructura de MySQL**

Hay 3 tablas, la de usuario, curso y donde queda registrado en que curso se inscribió el usuario.

```

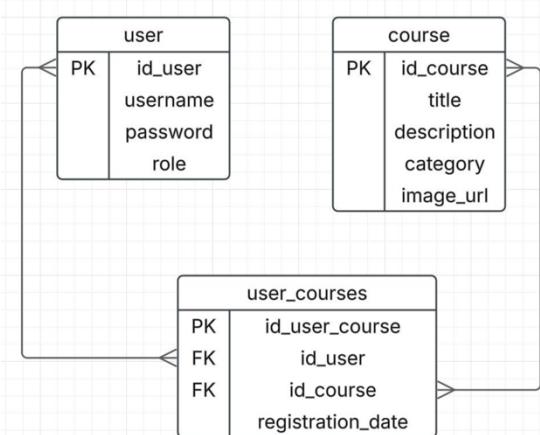
CREATE TABLE user (
    id_user INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(15) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    role ENUM('admin', 'normal') NOT NULL DEFAULT 'normal'
);
CREATE TABLE course (
    id_course INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(30) NOT NULL,
    description TEXT NOT NULL,
    category VARCHAR(15) NOT NULL,
    image_url VARCHAR(500) NOT NULL DEFAULT './assets/default.png'
);
CREATE TABLE user_courses ( id_user_course INT AUTO_INCREMENT PRIMARY KEY,
    id_user INT NOT NULL, id_course INT NOT NULL,
    registration_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT unique_user_course UNIQUE (id_user, id_course),
    FOREIGN KEY (id_user) REFERENCES user(id_user) ON DELETE CASCADE,
    FOREIGN KEY (id_course) REFERENCES course(id_course) ON DELETE CASCADE
);

```

id_user	username	password	role

id_course	title	description	category	Image_url

id_user_course	id_user	id_course	registration_date



- Generación de pruebas para la robustez de la aplicación:

- Pruebas de aceptación
- Pruebas de usabilidad
- Pruebas de seguridad
- Pruebas unitarias

Trabajé en la implementación de diferentes pruebas para asegurar que la aplicación funcione correctamente en todos los aspectos. Primero, hice las pruebas de aceptación, que básicamente se trataban de verificar si la aplicación cumplía con lo que se esperaba. Probé los flujos más importantes: registrar usuarios, iniciar sesión, acceder a los cursos. Cosas así. Me aseguré de que todo lo que un usuario real haría en la plataforma estuviera bien implementado y sin errores visibles. Algunas pruebas pasaron sin problemas, otras me obligaron a corregir detalles que no había notado antes.

Después me enfoqué en las pruebas de usabilidad. Aquí el reto fue ver qué tan fácil era moverse dentro de la aplicación. Navegué por cada sección, revisé botones, formularios, y traté de pensar como alguien que usa la plataforma por primera vez. Encontré cosas que parecían obvias para mí, pero que para alguien sin experiencia en el sistema podían ser confusas. Ajusté algunas cosas, como la disposición de los elementos y la claridad de los mensajes. Hacer esto me hizo ver que la funcionalidad es importante, pero si la gente no entiende cómo usar la app, no sirve de mucho.

Luego pasé a las pruebas de seguridad, donde el objetivo era ver qué tan protegida estaba la aplicación contra posibles ataques. Probé cosas como ingresar contraseñas incorrectas muchas veces, modificar tokens JWT para intentar acceder a recursos restringidos y validar si los datos sensibles estaban realmente protegidos. Encontré algunos puntos débiles que arreglé rápidamente, porque no quería que hubiera ninguna vulnerabilidad que pusiera en riesgo la información de los usuarios. Este tipo de pruebas me hizo pensar en lo fácil que es olvidar la seguridad cuando uno está enfocado solo en que "funcione".

Por último, hice las pruebas unitarias para validar que cada módulo del código hiciera lo que debía hacer sin depender del resto del sistema. Aquí usé herramientas como Jest y Supertest para verificar cosas como la autenticación, la gestión de cursos y el manejo de la base de datos. Algunas pruebas fallaron porque había pequeños errores que no noté en el desarrollo, pero gracias a esto pude corregirlos antes de que llegaran a producción. Todo este proceso me dejó claro que probar el software no es solo buscar errores, sino asegurarse de que la aplicación sea confiable, estable y segura para los usuarios.

```
pedrodeleon@Laptop-de-Pedro test % npx jest test.test.js
  console.log
    Servidor corriendo en http://localhost:3000
      at Server.log (src/app.js:58:13)
    console.log
      Conectado a MySQL
        at log (src/config/db.js:18:17)
    console.log
      Conexión a la base de datos establecida correctamente
        at log (src/app.js:42:17)

  PASS  ./test.test.js
    ✓ Pruebas de aceptación del usuario
      ✓ Debe permitir a un usuario inscribirse en un curso (233 ms)
      ✓ Debe permitir a un usuario salir de un curso (157 ms)
      ✓ Debe rechazar eliminación de cursos por un usuario normal (5 ms)
    ✘ Autenticación de usuarios
      ✓ Debe permitir iniciar sesión con credenciales correctas (174 ms)
      ✓ Debe rechazar credenciales incorrectas (149 ms)
    ✘ Gestión de permisos según rol
      ✓ Debe permitir acceso a admin para crear cursos (85 ms)
      ✓ Debe rechazar acceso a usuarios normales para crear cursos (2 ms)
    ✘ Pruebas de manejo de tokens en JWT
      ✓ Debe rechazar acceso sin token (2 ms)
      ✓ Debe rechazar token inválido (1 ms)
      ✓ Debe aceptar un token válido (152 ms)
      ✓ Debe manejar la expiración del token correctamente (6 ms)
    ✘ Pruebas de usabilidad
      ✓ Debe cargar los cursos correctamente con paginación (153 ms)
      ✓ Debe permitir buscar cursos con filtros (79 ms)

Test Suites: 1 passed, 1 total
Tests:       13 passed, 13 total
Snapshots:  0 total
Time:        3.167 s, estimated 4 s
Ran all test suites matching /test.test.js/i.
 Jest did not exit one second after the test run has completed.

  "This usually means that there are asynchronous operations that weren't stopped in your tests. Consider running Jest with '--detectOpenHandles' to troubleshoot this issue."
```

## Subir archivos a GitHub

Dev-Pedro-Subir archivos-Se subieron los archivos base #4

Merged · Yugidar merged 1 commit into main from Dev-SubirArchivos · last week

Conversation 1 · Commits 1 · Checks 0 · Files changed 31 · +6,974 -0

pedrodeleondev commented 2 weeks ago

No description provided.

pedrodeleondev requested review from FridaGarzaG and Yugidar 2 weeks ago

pedrodeleondev self-assigned this 2 weeks ago

Yugidar approved these changes last week

FridaGarzaG approved these changes last week

FridaGarzaG left a comment

Reviewers: FridaGarzaG, Yugidar

Assignees: pedrodeleondev

Labels: None yet

Projects: None yet

Milestone: No milestone

Development: Successfully merging this pull request may close these issues.

## Inconvenientes/Problemas y Solución

### Error Login Campo usuario -02 & Error Login Campo usuario

El campo de usuario no es case sensitive y únicamente detecta si existe el usuario o no en la BD.

### Error ficheros- 02

Se modificó el código para que esto sea posible.

```
const express = require('express');
const path = require('path');
const cors = require('cors');
const db = require('./config/db');
const errorHandler = require('./middleware/errorHandler'); // Middleware de errores agregado

const app = express();
process.env.JWT_SECRET = process.env.JWT_SECRET || 'secreto_super_seguro';
const PORT = process.env.PORT || 3000;

app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// Rutas
const authRoutes = require('./routes/authRoutes');
const courseRoutes = require('./routes/courseRoutes');

app.use('/auth', authRoutes);
app.use('/courses', courseRoutes); // Ruta de cursos

app.use(express.static(path.join(__dirname, '../public')));

app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, '../public/index.html'));
});

app.get('/login', (req, res) => {
  res.sendFile(path.join(__dirname, '../public/login.html'));
});

app.get('/register', (req, res) => {
  res.sendFile(path.join(__dirname, '../public/register.html'));
});

// Verificar conexión a MySQL
db.getConnection((err, connection) => {
  if (err) {
    console.error("Error al conectar con la base de datos:", err.message);
  } else {
    console.log("Conexión a la base de datos establecida correctamente");
    connection.release(); // Liberar conexión
  }
});
```

```
// Middleware para manejar rutas inexistentes (404)
app.use((req, res, next) => {
  const error = new Error("Ruta no encontrada");
  error.statusCode = 404;
  next(error); // Pasar error al middleware global
});

// Middleware para manejo de errores global
app.use(errorHandler); // Aquí se usa el middleware de errores

app.listen(PORT, () => {
  console.log(`Servidor corriendo en http://localhost:${PORT}`);
});

module.exports = app;

{
  "name": "proyecto-final",
  "version": "1.0.0",
  "description": "Proyecto final de la clase Desarrollo Full Stack",
  "main": "index.js",
  "scripts": {
    "start": "node src/app.js",
    "dev": "nodemon src/app.js",
    "test": "jest --coverage"
  },
  "dependencies": {
    "axios": "^1.7.9",
    "bcryptjs": "^3.0.2",
    "cors": "^2.8.5",
    "dotenv": "^16.4.7",
    "express": "^4.21.2",
    "express-validator": "^7.2.1",
    "helmet": "^8.0.0",
    "jsonwebtoken": "^9.0.2",
    "morgan": "^1.10.0",
    "mysql2": "^3.12.0",
    "nodemon": "^3.0.2",
    "passport": "^0.6.0",
    "passport-jwt": "^4.0.1",
    "passport-local": "^1.0.0"
  },
  "devDependencies": {
    "jest": "^29.7.0",
    "supertest": "^6.3.4"
  },
  "jest": {
    "testEnvironment": "node"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/Yugidar/Proyecto-Final.git"
  }
}
```

```
{
  "keywords": [],
  "author": "",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/Yugidar/Proyecto-Final/issues"
  },
  "homepage": "https://github.com/Yugidar/Proyecto-Final#readme"
}
```

## Error login- 03

Solamente era un problema de la nube.

## Error API, despliegue en la nube

Se cambió la API.

```
document.addEventListener('DOMContentLoaded', async () => {
  async function obtenerPais() {
    try {
      const response = await
fetch('https://api.ipgeolocation.io/ipgeo?apiKey=b38be2d1e68b4bb0b7c3d9f33fee6ca9');
      const data = await response.json();

      // Verificar que la API devolvió datos correctos
      if (data && data.country_name) {
        const loginCountryElement = document.getElementById('loginChangeCountry');
        if (loginCountryElement) {
          loginCountryElement.innerText = data.country_name;
        }
      } else {
        console.error('No se pudo detectar el país');
      }
    } catch (error) {
      console.error('Error obteniendo la ubicación:', error);
    }
  }

  obtenerPais();
});
```

## Error vista admin -02 & Error vista usuario estándar

Se les puso un límite de caracteres a los campos.

```
form id="formCrearCurso">
  <label for="nombre">Nombre:</label>
  <input type="text" class="colorWhite w-100" id="nombreCurso" name="nombre"
    placeholder="Ingresa nombre del curso" maxlength="30" required>

  <label for="categoria">Categoría:</label>
  <input type="text" class="colorWhite w-100" id="categoriaCurso" name="categoria"
    placeholder="Ingresa categoría del curso" maxlength="15" required>

  <label for="descripcion">Descripción:</label>
  <textarea id="descripcionCurso" class="colorWhite w-100" name="descripcion">
```

```

placeholder="Ingresa descripción del curso" maxlength="40" required></textarea>

<label for="imagen">Imagen (link):</label>
<input type="url" class="colorWhite w-100" id="imagenCurso" name="imagen"
placeholder="Ingresa un link de imagen">

<button type="submit" id="guardarCurso" class="btn btnCrear">Guardar</button>
</form>
```

```

<form id="registerForm">
    <label for="username">Nombre:</label>
    <input type="text" id="username" name="username" maxlength="15" required>

    <label for="password">Contraseña:</label>
    <input type="password" id="password" name="password" required>

    <label for="role">Selecciona un rol:</label>
    <select id="role" name="role" required>
        <option value="normal" selected>Usuario Normal</option>
        <option value="admin">Administrador</option>
    </select>

    <input type="submit" value="Registrarse" id="inputLogin" class="btn-primary">
</form>
```

## Error ficheros

Se bloqueo node modules en gitignore

```

.gitignore
1 # Ignora archivos del sistema Mac
2 .DS_Store
3
4 # Ignora la carpeta node_modules
5 node_modules
6
7 # Ignora el archivo README
8 README.txt
```

## Código explicado

/public/js/adminCursos.js

El código gestiona cursos en una plataforma web, permitiendo buscarlos, editarlos y eliminarlos. Primero, configura variables para la paginación y la búsqueda en tiempo real. Si el usuario escribe en el buscador, se filtran los cursos por título o categoría. Si no hay nada escrito, se cargan los cursos de manera paginada. También se activa el scroll infinito: cuando el usuario llega al final de la página, se cargan más cursos automáticamente. Un detalle útil para evitar que la página se sienta lenta o pesada.

Para la edición y eliminación de cursos, el código maneja eventos en los botones correspondientes. Al eliminar un curso, se pide confirmación antes de enviar la petición al servidor. Si la eliminación es exitosa, la lista de cursos se actualiza sin necesidad de recargar toda la página. Para editar, los datos

del curso se rellenan en un formulario dentro de un modal. Una vez que el usuario guarda los cambios, se envía una actualización al servidor y se refresca la lista. Todo directo. Sin pasos innecesarios.

También hay un control de acceso. Se revisa el rol del usuario en el localStorage. Si no es "admin", la pantalla se bloquea por completo con un mensaje que impide la interacción. Además, en unos segundos se redirige automáticamente a otra página. No se puede hacer nada más. Esta medida evita que personas sin permisos puedan gestionar los cursos o realizar cambios que no les corresponden. Una manera sencilla de mantener la seguridad sin complicaciones.

Por último, la sesión del usuario se valida constantemente. Se extrae el token almacenado y se revisa si ha expirado. Si está vencido, el usuario es desconectado y redirigido a la pantalla de inicio de sesión. Además, cada 30 segundos se vuelve a comprobar. Si la sesión sigue activa, todo bien. Si no, se elimina el acceso. Así se evita que alguien permanezca en la plataforma sin autorización por mucho tiempo. Una seguridad simple, pero efectiva.

```
document.addEventListener("DOMContentLoaded", async function () {
    let currentPage = 1;
    let totalPages = 1;
    let editingCourseId = null;
    let allCourses = [];
    let filteredCourses = [];
    let isFiltering = false;
    let displayedCourses = 10;

    const paginationDots = document.getElementById("pagination-dots");
    const searchBar = document.getElementById("searchBox");
    const toggleSearch = document.getElementById("toggleSearch");

    /** 🔎 **Abrir y cerrar buscador** */
    if (toggleSearch && searchBar) {
        toggleSearch.addEventListener("click", function () {
            searchBar.style.display = searchBar.style.display === "block" ? "none" : "block";
        });
    }

    document.addEventListener("click", function (event) {
        if (!searchBar.contains(event.target) && event.target !== toggleSearch) {
            searchBar.style.display = "none";
        }
    });
}

/** 🔎 **Buscar cursos en tiempo real** */
document.getElementById("searchInput").addEventListener("input", async function () {
    const searchTerm = this.value.toLowerCase();
    if (searchTerm) {
        if (allCourses.length === 0) await fetchAllCourses();
        isFiltering = true;
        paginationDots.innerHTML = "";

        filteredCourses = allCourses.filter((curso) =>
            curso.title.toLowerCase().includes(searchTerm) ||

```

```
        curso.category.toLowerCase().includes(searchTerm)
    );

displayedCourses = 10;
renderCursos(filteredCourses.slice(0, displayedCourses));

window.onscroll = function () {
    if (window.innerHeight + window.scrollY >= document.body.offsetHeight - 100) {
        displayedCourses += 5;
        renderCursos(filteredCourses.slice(0, displayedCourses));
    }
};

} else {
    isFiltering = false;
    filteredCourses = [];
    currentPage = 1;
    fetchCourses(currentPage);
    window.onscroll = null;
}

});

async function fetchAllCourses() {
try {
    const token = localStorage.getItem("token");
    if (!token) {
        alert("Debes iniciar sesión primero.");
        window.location.href = "login.html";
        return;
    }

    const response = await fetch("/courses/all", {
        headers: { Authorization: `Bearer ${token}` },
    });

    if (!response.ok) {
        throw new Error("Error al obtener todos los cursos");
    }

    const data = await response.json();
    allCourses = data.courses;
} catch (error) {
    console.error("Error al obtener todos los cursos:", error);
}
}

async function fetchCourses(page) {
    if (isFiltering) return;

    try {
        const token = localStorage.getItem("token");
        if (!token) {
            alert("Debes iniciar sesión primero.");
        }
    }
}
```

```
        window.location.href = "login.html";
        return;
    }

    const response = await fetch(`/courses/paginated?page=${page}`, {
        headers: { Authorization: `Bearer ${token}` },
    });

    if (!response.ok) {
        throw new Error("Error al obtener cursos");
    }

    const data = await response.json();
    renderCursos(data.courses);
    totalPages = data.totalPages;
    renderPaginationDots();
} catch (error) {
    console.error("Error al obtener los cursos:", error);
}
}

function renderCursos(cursos) {
    const container = document.getElementById("cursos-container");
    if (!container) return;

    container.innerHTML = ``;

    cursos.forEach((curso) => {
        const cursoDiv = document.createElement("div");
        cursoDiv.classList.add("curso");

        cursoDiv.innerHTML = `
            <div class="cursoConten">
                <div class="contenidoCurAdm">
                    
                    <div class="textAdm">
                        <h3 id="textoP">${curso.title}</h3>
                        <p id="textoP">Categoría: ${curso.category}</p>
                        <p id="textoH">${curso.description}</p>
                    </div>
                </div>
            </div>

            <div class="botones">
                <button class="btn btn-secondary btn-edit" id="btnEdit"
                    data-id="${curso.id_course}"
                    data-title="${curso.title}"
                    data-category="${curso.category}"
                    data-description="${curso.description}"
                    data-image="${curso.image_url}">Editar</button>

                <button id="btnElim" class="btn btn-danger btn-delete" data-
id="${curso.id_course}">Eliminar</button>
            
```

```
        </div>
    </div>
    `;
    container.appendChild(cursoDiv);
});

asignarEventos();
}

function asignarEventos() {
    document.querySelectorAll(".btn-delete").forEach(button => {
        button.addEventListener("click", function () {
            const id = this.getAttribute("data-id");
            eliminarCurso(id);
        });
    });
}

document.querySelectorAll(".btn-edit").forEach(button => {
    button.addEventListener("click", function () {
        const id = this.getAttribute("data-id");
        const title = this.getAttribute("data-title");
        const category = this.getAttribute("data-category");
        const description = this.getAttribute("data-description");
        const image = this.getAttribute("data-image");

        editarCurso(id, title, category, description, image);
    });
});
}

async function eliminarCurso(id) {
    const confirmacion = confirm("¿Estás seguro de que deseas eliminar este curso?");
    if (!confirmacion) return;

    try {
        const token = localStorage.getItem('token');
        if (!token) {
            alert('Debes iniciar sesión primero.');
            window.location.href = 'login.html';
            return;
        }

        const response = await fetch(`/courses/${id}`, {
            method: 'DELETE',
            headers: { 'Authorization': `Bearer ${token}` }
        });

        if (!response.ok) {
            throw new Error('Error al eliminar el curso');
        }
    }

    alert("Curso eliminado correctamente");
}
```

```
    fetchCourses(currentPage);
} catch (error) {
  console.error("Error al eliminar curso:", error);
  alert("Hubo un error al eliminar el curso");
}
}

// MODAL CREAR CURSO
const modalCrear = document.getElementById("modalCrear");
const btnCrear = document.getElementById("btnCrear");

if (btnCrear) {
  btnCrear.onclick = function () {
    openModal(modalCrear);
  };
}

document.getElementById("formCrearCurso").addEventListener("submit", async function (event) {
  event.preventDefault();

  const title = document.getElementById("nombreCurso").value;
  const category = document.getElementById("categoriaCurso").value;
  const description = document.getElementById("descripcionCurso").value;
  const image_url = document.getElementById("imagenCurso").value ||
    'https://upload.wikimedia.org/wikipedia/commons/1/14/No_Image_Available.jpg';

  if (!title || !category || !description) {
    alert("Todos los campos son obligatorios, excepto la imagen.");
    return;
  }

  try {
    const token = localStorage.getItem("token");

    const response = await fetch("/courses", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
        "Authorization": `Bearer ${token}`
      },
      body: JSON.stringify({ title, category, description, image_url })
    });

    if (!response.ok) {
      throw new Error("Error al crear el curso");
    }

    alert("Curso creado exitosamente");
    closeModal(modalCrear);
    fetchCourses(1);
  } catch (error) {
    console.error("Error al crear curso:", error);
    alert("Hubo un error al crear el curso");
  }
})
```

```

    }
});

// FUNCIÓN PARA EDITAR CURSO
function editarCurso(id, title, category, description, image) {
    editingCourseId = id;
    document.getElementById("nombreCursoEdit").value = title;
    document.getElementById("categoriaCursoEdit").value = category;
    document.getElementById("descripcionCursoEdit").value = description;
    document.getElementById("imagenCursoEdit").value = image;
    openModal(document.getElementById("modalEdit"));
}

document.getElementById("formEditCurso").addEventListener("submit", async function (event) {
    event.preventDefault();

    const title = document.getElementById("nombreCursoEdit").value;
    const category = document.getElementById("categoriaCursoEdit").value;
    const description = document.getElementById("descripcionCursoEdit").value;
    const imageUrl = document.getElementById("imagenCursoEdit").value;

    try {
        const token = localStorage.getItem("token");

        const response = await fetch(`/courses/${editingCourseId}`, {
            method: "PUT",
            headers: {
                "Content-Type": "application/json",
                "Authorization": `Bearer ${token}`
            },
            body: JSON.stringify({ title, category, description, imageUrl })
        });

        if (!response.ok) {
            throw new Error("Error al actualizar el curso");
        }

        alert("Curso actualizado correctamente");
        closeModal(document.getElementById("modalEdit"));
        fetchCourses(1);
    } catch (error) {
        console.error("Error al actualizar curso:", error);
        alert("Hubo un error al actualizar el curso");
    }
});

function renderPaginationDots() {
    paginationDots.innerHTML = "";
    if (isFiltering) return;

    for (let i = 1; i <= totalPages; i++) {
        let dot = document.createElement("span");

```

```
dot.classList.add("dot");
if (i === currentPage) {
    dot.classList.add("active");
}
dot.addEventListener("click", () => {
    currentPage = i;
    fetchCourses(currentPage);
});
paginationDots.appendChild(dot);
}

fetchCourses(currentPage);

function openModal(modal) {
    modal.classList.add("open");
    document.body.classList.add("jw-modal-open");
}

function closeModal(modal) {
    modal.classList.remove("open");
    document.body.classList.remove("jw-modal-open");
}
});

document.addEventListener("DOMContentLoaded", function () {
    const userRole = localStorage.getItem("role");

    if (userRole !== "admin") {
        // Crear un div que bloquee toda la pantalla
        const overlay = document.createElement("div");
        overlay.style.position = "fixed";
        overlay.style.top = "0";
        overlay.style.left = "0";
        overlay.style.width = "100vw";
        overlay.style.height = "100vh";
        overlay.style.backgroundColor = "black";
        overlay.style.color = "white";
        overlay.style.display = "flex";
        overlay.style.flexDirection = "column";
        overlay.style.justifyContent = "center";
        overlay.style.alignItems = "center";
        overlay.style.fontSize = "2rem";
        overlay.style.fontWeight = "bold";
        overlay.style.zIndex = "9999";
        overlay.innerHTML = `
            ✗ NO ERES ADMIN ✗
            <br>
            <button id="btnRegresar" style="
                margin-top: 20px;
                padding: 15px 30px;
                font-size: 1.5rem;
                font-weight: bold;
            >
```

```
color: white;
background-color: red;
border: none;
border-radius: 10px;
cursor: pointer;
">  Regresar a Cursos</button>
';

// Agregar el overlay al body
document.body.appendChild(overlay);

// Bloquear interacciones (opcional)
document.body.style.overflow = "hidden";

// Agregar evento al botón para regresar a cursos
document.getElementById("btnRegresar").addEventListener("click", function () {
    window.location.href = "cursos.html";
});

// Redirigir automáticamente después de 3 segundos
setTimeout(() => {
    window.location.href = "cursos.html";
}, 3000);
}

};

document.addEventListener("DOMContentLoaded", function () {
    const token = localStorage.getItem("token");

    if (token) {
        const decoded = JSON.parse(atob(token.split(".")[1])); // Decodifica el token
        const currentTime = Math.floor(Date.now() / 1000); // Tiempo actual en segundos

        if (decoded.exp < currentTime) {
            // Token vencido, eliminar y redirigir al index
            localStorage.removeItem("token");
            localStorage.removeItem("role");
            alert("Tu sesión ha expirado. Inicia sesión nuevamente.");
            window.location.href = "index.html";
        } else {
            // Configurar verificación en intervalos de tiempo (cada 30 segundos)
            setInterval(() => {
                const now = Math.floor(Date.now() / 1000);
                if (decoded.exp < now) {
                    localStorage.removeItem("token");
                    localStorage.removeItem("role");
                    alert("Tu sesión ha expirado. Inicia sesión nuevamente.");
                    window.location.href = "index.html";
                }
            }, 30000); // Se ejecuta cada 30 segundos
        }
    }
});
```

## /public/js/cursos.js

El código maneja la carga y visualización de cursos en una plataforma, aplicando paginación infinita. Al cargar la página, inicializa variables para el control de los cursos, evitando duplicados y asegurando que no se carguen datos innecesarios. Si el usuario llega al final de la página, se solicita automáticamente más contenido, siempre y cuando no se esté filtrando. También se implementa un buscador en tiempo real que filtra los cursos por título o categoría, actualizando la vista sin recargar la página.

El sistema permite a los usuarios inscribirse en cursos con un solo clic. Si el usuario ya está inscrito, el botón cambia de estado para reflejarlo y se desactiva para evitar múltiples inscripciones. Además, si el usuario intenta acceder a un curso ya obtenido, se muestra un mensaje de confirmación. Todos los cursos inscritos por el usuario se almacenan en una lista para que el sistema pueda verificar su estado sin necesidad de recargar constantemente los datos desde el servidor.

Para la seguridad, se incluye un bloqueo de acceso basado en roles. Si un administrador intenta ver la sección de usuarios regulares, la pantalla se cubre con un mensaje de restricción y, tras unos segundos, se redirige automáticamente a la página de administración. De la misma manera, se revisa si el usuario tiene una sesión activa. Si el token ha expirado, se eliminan los datos del usuario y se redirige a la pantalla de inicio de sesión. Además, se realiza una verificación periódica cada 30 segundos para garantizar que nadie se quede dentro con un token vencido.

El código también gestiona la visibilidad del buscador. Se oculta automáticamente cuando el usuario hace clic fuera de él, mejorando la experiencia de navegación. También evita que la página cargue cursos cuando se está filtrando activamente, evitando conflictos en la visualización. Todo esto hace que la aplicación sea fluida, interactiva y segura para los usuarios.

```
document.addEventListener("DOMContentLoaded", async function () {
  let lastCourseId = 0;
  const itemsPerPage = 4;
  let allCourses = [];
  let isFiltering = false;
  let isLoading = false;
  let allCoursesFetched = false;

  const cursosContainer = document.getElementById("cursos-container");
  const searchInput = document.getElementById("searchInput");

  async function fetchCourses(lastId) {
    if (isLoading || isFiltering) return; // 🔍 No cargar más si está filtrando
    isLoading = true;

    try {
      const token = localStorage.getItem('token');
      if (!token) {
        alert('Debes iniciar sesión primero.');
        window.location.href = 'login.html';
        return;
      }
    
```

```
const response = await fetch('/courses/load-more?lastCourseId=${lastId}&limit=${itemsPerPage}', {
  headers: { 'Authorization': `Bearer ${token}` }
});

if (!response.ok) {
  throw new Error('Error al obtener cursos');
}

const data = await response.json();

// ♦ Evitar agregar cursos duplicados
const newCourses = data.courses.filter(curso =>
  !allCourses.some(existing => existing.id_course === curso.id_course)
);

if (newCourses.length > 0) {
  allCourses = [...allCourses, ...newCourses];
  const userCourses = await fetchUserCourses();
  appendCursos(newCourses, userCourses);
  lastCourseId = parseInt(newCourses[newCourses.length - 1].id_course);
}
} catch (error) {
  console.error("Error al obtener los cursos:", error);
} finally {
  isLoading = false;
}
}

async function fetchAllCourses() {
  if (allCoursesFetched) return allCourses;

  try {
    const token = localStorage.getItem('token');
    const response = await fetch('/courses/all', {
      headers: { 'Authorization': `Bearer ${token}` }
    });

    if (!response.ok) {
      throw new Error("Error al obtener todos los cursos");
    }

    const data = await response.json();
    allCoursesFetched = true;
    return data.courses;
  } catch (error) {
    console.error("Error al obtener todos los cursos:", error);
    return [];
  }
}
```

```
async function fetchUserCourses() {
  try {
    const token = localStorage.getItem('token');
    const response = await fetch('/courses/user-courses', {
      headers: { 'Authorization': `Bearer ${token}` }
    });

    if (!response.ok) {
      throw new Error("Error al obtener los cursos del usuario");
    }

    const data = await response.json();
    return new Set(data.courses.map(course => parseInt(course.id_course)));
  } catch (error) {
    console.error("Error al obtener cursos del usuario:", error);
    return new Set();
  }
}

function renderCursos(cursos, userCourses) {
  cursosContainer.innerHTML = ""; // 🗑️ Limpiar contenedor solo en búsqueda
  appendCursos(cursos, userCourses);
}

function appendCursos(cursos, userCourses) {
  cursos.forEach(curso => {
    const cursoDiv = document.createElement("div");
    cursoDiv.classList.add("cursoNormal");

    const isEnrolled = userCourses.has(parseInt(curso.id_course));
    const actionText = isEnrolled ? "Curso Obtenido" : "Agregar Curso";
    const disabledAttr = isEnrolled ? "disabled" : "";

    cursoDiv.innerHTML = `
      <div class="cursoConten">
        <div class="contenidoCurNor">
          
          <div class="textNor">
            <button class="botonAgre btn-course-action" id="btnAgre" data-id="${curso.id_course}" data-enrolled="${isEnrolled}" ${disabledAttr}>
              ${actionText}
            </button>
            <h3 id="textoP">${curso.title}</h3>
            <p id="textoP" class="fw-bold">Categoría: ${curso.category}</p>
            <p id="textoP">${curso.description}</p>
          </div>
        </div>
      </div>
    `;
    cursosContainer.appendChild(cursoDiv);
  });
}
```

```
document.querySelectorAll(".btn-course-action").forEach(button => {
  button.addEventListener("click", async function () {
    const courseId = parseInt(this.getAttribute("data-id"));
    const enrolled = this.getAttribute("data-enrolled") === "true";

    if (!enrolled) {
      await inscribirseCurso(courseId);
      this.textContent = "Curso Obtenido";
      this.setAttribute("data-enrolled", "true");
      this.disabled = true;
    } else {
      ingresarCurso(courseId);
    }
  });
});

async function inscribirseCurso(courseId) {
  try {
    const token = localStorage.getItem("token");
    const response = await fetch(`/courses/enroll/${courseId}`, {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
        "Authorization": `Bearer ${token}`
      }
    });

    if (!response.ok) {
      throw new Error("Error al inscribirse en el curso");
    }

    alert("Te has inscrito correctamente en el curso.");
    const userCourses = await fetchUserCourses();
    renderCursos(allCourses, userCourses);
  } catch (error) {
    console.error("Error al inscribirse en el curso:", error);
    alert("Hubo un error al inscribirse en el curso.");
  }
}

function ingresarCurso(courseId) {
  alert(`Ingresando al curso con ID: ${courseId}`);
}

searchInput.addEventListener("input", async function () {
  const searchTerm = this.value.toLowerCase().trim();

  if (searchTerm) {
    if (!allCoursesFetched) {
      allCourses = await fetchAllCourses();
    }
  }
})
```

```
isFiltering = true;
cursosContainer.innerHTML = ""; // ⚡ Solo limpiar si estamos filtrando

const filteredCourses = allCourses.filter((curso) =>
  curso.title.toLowerCase().includes(searchTerm) ||
  curso.category.toLowerCase().includes(searchTerm)
);

const userCourses = await fetchUserCourses();
renderCursos(filteredCourses, userCourses);
} else {
  isFiltering = false;
  cursosContainer.innerHTML = "";
  lastCourseId = allCourses.length > 0 ? allCourses[allCourses.length - 1].id_course : 0;
  renderCursos(allCourses, await fetchUserCourses()); // ⚡ Restaurar sin duplicados
}
});

window.addEventListener("scroll", function () {
  if (window.innerHeight + window.scrollY >= document.body.offsetHeight - 100) {
    if (!isFiltering && !isLoading) {
      fetchCourses(lastCourseId);
    }
  }
});

fetchCourses(lastCourseId);
});

document.getElementById("toggleSearch").addEventListener("click", function () {
  let searchBar = document.getElementById("searchBox");

  if (searchBar.style.display === "none") {
    searchBar.style.display = "block";
    document.getElementById("searchInput").focus();
  } else {
    searchBar.style.display = "none";
  }
});

document.addEventListener("click", function (event) {
  let searchBar = document.getElementById("searchBox");
  let toggleImage = document.getElementById("toggleSearch");

  if (!searchBar.contains(event.target) && event.target !== toggleImage) {
    searchBar.style.display = "none";
  }
});
document.addEventListener("DOMContentLoaded", function () {
  const userRole = localStorage.getItem("role");
```

```

if (userRole === "admin") {
    // Crear pantalla de bloqueo
    const overlay = document.createElement("div");
    overlay.style.position = "fixed";
    overlay.style.top = "0";
    overlay.style.left = "0";
    overlay.style.width = "100vw";
    overlay.style.height = "100vh";
    overlay.style.backgroundColor = "black";
    overlay.style.color = "white";
    overlay.style.display = "flex";
    overlay.style.flexDirection = "column";
    overlay.style.justifyContent = "center";
    overlay.style.alignItems = "center";
    overlay.style.fontSize = "2rem";
    overlay.style.fontWeight = "bold";
    overlay.style.zIndex = "9999";
    overlay.innerHTML = `

        ✗ SOLO USUARIOS REGULARES PUEDEN VER ESTO ✗

        <br>
        <button id="btnRegresarAdmin" style="

            margin-top: 20px;
            padding: 15px 30px;
            font-size: 1.5rem;
            font-weight: bold;
            color: white;
            background-color: red;
            border: none;
            border-radius: 10px;
            cursor: pointer;
        >  Ir a Administración</button>

    `;
}

// Agregar el overlay al body
document.body.appendChild(overlay);

// Bloquear interacciones (opcional)
document.body.style.overflow = "hidden";

// Agregar evento al botón para regresar a adminCursos.html
document.getElementById("btnRegresarAdmin").addEventListener("click", function () {
    window.location.href = "adminCursos.html";
});

// Redirigir automáticamente después de 3 segundos
setTimeout(() => {
    window.location.href = "adminCursos.html";
}, 3000);
}
});
```

```

document.addEventListener("DOMContentLoaded", function () {
  const token = localStorage.getItem("token");

  if (token) {
    const decoded = JSON.parse(atob(token.split(".")[1])); // Decodifica el token
    const currentTime = Math.floor(Date.now() / 1000); // Tiempo actual en segundos

    if (decoded.exp < currentTime) {
      // Token vencido, eliminar y redirigir al index
      localStorage.removeItem("token");
      localStorage.removeItem("role");
      alert("Tu sesión ha expirado. Inicia sesión nuevamente.");
      window.location.href = "index.html";
    } else {
      // Configurar verificación en intervalos de tiempo (cada 30 segundos)
      setInterval(() => {
        const now = Math.floor(Date.now() / 1000);
        if (decoded.exp < now) {
          localStorage.removeItem("token");
          localStorage.removeItem("role");
          alert("Tu sesión ha expirado. Inicia sesión nuevamente.");
          window.location.href = "index.html";
        }
      }, 30000); // Se ejecuta cada 30 segundos
    }
  }
});
```

## /public/js/geo.js

Este código obtiene el país del usuario a través de su dirección IP y lo muestra en la página. Cuando la página se carga, se ejecuta la función obtenerPais(), que realiza una solicitud a una API de geolocalización. Si la respuesta es válida y contiene el nombre del país, este se inserta en un elemento HTML con el ID loginChangeCountry.

Para garantizar que la actualización se realice correctamente, primero se verifica si el elemento con el ID loginChangeCountry existe en el DOM antes de intentar modificar su contenido. Si el país no se puede detectar, se muestra un mensaje de error en la consola, evitando que el código falle silenciosamente.

El uso de try...catch es clave aquí. Si la solicitud a la API falla (por ejemplo, debido a un problema de conexión o un error en la clave de la API), el catch captura el error y lo muestra en la consola. Esto evita que la ejecución del resto del código se interrumpa.

Este enfoque es útil para personalizar la experiencia del usuario según su ubicación. Sin embargo, depende de un servicio externo, lo que significa que si la API deja de estar disponible o la clave expira, la funcionalidad no podrá ejecutarse correctamente.

```

document.addEventListener('DOMContentLoaded', async () => {
  async function obtenerPais() {
    try {
      const response = await
fetch('https://api.ipgeolocation.io/ipgeo?apiKey=b38be2d1e68b4bb0b7c3d9f33fee6ca9');
      const data = await response.json();
```

```

    // Verificar que la API devolvió datos correctos
    if (data && data.country_name) {
        const loginCountryElement = document.getElementById('loginChangeCountry');
        if (loginCountryElement) {
            loginCountryElement.innerText = data.country_name;
        }
    } else {
        console.error('No se pudo detectar el país');
    }
} catch (error) {
    console.error('Error obteniendo la ubicación:', error);
}
}

obtenerPais();
});

```

### /public/js/login.js

Este código maneja el proceso de inicio de sesión en la aplicación. Se activa cuando el usuario envía el formulario de inicio de sesión (loginForm). Primero, evita que la página se recargue usando event.preventDefault(). Luego, obtiene el nombre de usuario y la contraseña ingresados en los campos del formulario.

Después, se envía una solicitud POST al endpoint /auth/login, enviando las credenciales en formato JSON. Si la respuesta del servidor es exitosa (response.ok), se guarda el token de autenticación y el rol del usuario en localStorage. Esto permite que el usuario mantenga su sesión activa sin necesidad de volver a autenticarse en cada acción.

El código también maneja la redirección según el rol del usuario. Si el usuario tiene el rol de "admin", es enviado a adminCursos.html, mientras que los usuarios regulares son redirigidos a cursos.html. Si el inicio de sesión falla, se muestra un mensaje de error basado en la respuesta del servidor.

Este método asegura un flujo de autenticación básico y eficiente. Sin embargo, no valida los campos antes de enviarlos, lo que significa que se podrían enviar datos vacíos. Para mejorar la experiencia del usuario, se podría agregar una verificación previa en el frontend antes de realizar la solicitud.

```

document.getElementById('loginForm').addEventListener('submit', async (event) => {
    event.preventDefault();
    const username = document.getElementById('username').value;
    const password = document.getElementById('password').value;

    const response = await fetch('/auth/login', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ username, password })
    });

    const data = await response.json();
    if (response.ok) {

```

```

    alert('Inicio de sesión exitoso');
    localStorage.setItem('token', data.token);
    localStorage.setItem('role', data.role); // Guardamos el rol en el almacenamiento local

    if (data.role === 'admin') {
        window.location.href = 'adminCursos.html';
    } else {
        window.location.href = 'cursos.html';
    }
} else {
    alert(data.error);
}
});

```

## /public/js/main.js

Este código maneja la gestión de cursos en una aplicación web. Implementa un sistema donde los usuarios pueden agregar, editar y eliminar cursos, además de marcar su estado como "En curso" o "Terminado". Los datos de los cursos se almacenan en localStorage, asegurando que la información se mantenga persistente entre sesiones. Al cargar la página, se inicializa una instancia de GestorDeCursos, que se encarga de renderizar los cursos guardados y gestionar sus cambios dinámicamente.

Cada curso es representado mediante la clase Curso, que encapsula sus propiedades y métodos. El GestorDeCursos administra una lista de estos objetos y proporciona métodos para agregar, actualizar y eliminar cursos. Al modificar un curso, se actualiza automáticamente en la interfaz y en localStorage, asegurando coherencia en la visualización. Además, se agregan eventos a los botones de edición y eliminación de cada curso, permitiendo la interacción sin necesidad de recargar la página.

También se maneja la autenticación del usuario. Se guarda la información de la sesión en localStorage, y dependiendo del estado de la sesión, se ajustan los elementos de la interfaz, como el botón de login. Si un usuario intenta acceder sin estar autenticado, es redirigido a la página de inicio de sesión. Además, se proporciona una funcionalidad de cierre de sesión que elimina el token de autenticación y redirige al usuario al login.

Para la gestión de la interfaz, se implementan modales que permiten agregar o actualizar cursos sin salir de la página. Estos modales se activan y cierran dinámicamente, mejorando la experiencia del usuario. También se incluyen eventos para evitar que los modales se cierren accidentalmente al hacer clic fuera de ellos. Todo esto hace que la aplicación sea más intuitiva, interactiva y fácil de usar.

```

let sessionData
let btnLoginElement
let userDisplayElement
let gestorDeCursos

class Curso {
    constructor(id, autor, nombreCurso, terminado) {
        this._id = id;
        this._autor = autor;
        this._nombreCurso = nombreCurso;
        this._onBtnLoginClick = terminado ?? false;
    }
}

```

```
// Getters
get id() {
    return this._id;
}

get autor() {
    return this._autor;
}

get nombreCurso() {
    return this._nombreCurso;
}

get terminado() {
    return this._terminado;
}

// Setters
setAutor(nuevoAutor) {
    this._autor = nuevoAutor;
}

setNombreCurso(nuevoNombre) {
    this._nombreCurso = nuevoNombre;
}

// Método para alternar estado de terminado
toggleTerminado() {
    this._terminado = !this._terminado;
}

}

class GestorDeCursos {
    constructor(cursos) {
        this.cursos = cursos.map(curso => (new Curso(
            curso.id,
            curso.autor,
            curso.nombreCurso,
            curso.terminado
        )) ?? []);
        this.renderListaCursos();
    }

    setCursosToLocalStorage() {
        const cursosLimpios = this.cursos.map(curso => ({
            id: curso.id,
            autor: curso.autor,
            nombreCurso: curso.nombreCurso,
            terminado: curso.terminado
        }));
        localStorage.setItem('cursos', JSON.stringify(cursosLimpios))
    }
}
```

```
agregarCurso(autor, nombreCurso) {
    const id = this.cursos.length
        ? this.cursos[this.cursos.length - 1].id + 1
        : this.cursos.length + 1
    const curso = new Curso(id, autor, nombreCurso)
    this.cursos.push(curso)
    this.setCursosToLocalStorage()
    this.renderListaCursos()
}

eliminarCurso(id) {
    this.cursos = this.cursos.filter(curso => curso.id !== id)
    this.setCursosToLocalStorage()
    this.renderListaCursos()
}

actualizarCurso({ id, autor, nombreCurso }) {
    const curso = this.cursos.find(curso => curso.id === id);

    if (!curso) {
        console.error(`Curso con ID ${id} no encontrado`);
        return;
    }

    if (autor) curso.setAutor(autor);
    if (nombreCurso) curso.setNombreCurso(nombreCurso);
    this.setCursosToLocalStorage();
    this.renderListaCursos();
    alert("Curso actualizado correctamente");
}

onEstadoChange(id) {
    const curso = this.cursos.find(curso => curso.id === id);
    if (curso) {
        curso.toggleTerminado();
        this.setCursosToLocalStorage()
        this.renderListaCursos()
    } else {
        console.error(`Curso con ID ${id} no encontrado`);
    }
}

renderListaCursos() {
    this.clearContainer('listaCursos')
    const listaCursos = document.getElementById('listaCursos')
    this.cursos.forEach((curso, index) => {
        const card = document.createElement("div");
        card.classList.add('lista-cursos-card')
        card.innerHTML =
            <div class="d-flex mb-1">
```

```

<p class="lista-cursos-card__label quantico-font text-uppercase me-1">ID:</p>
<P class="lista-cursos-card__id quantico-font text-uppercase">#${curso.id}</P>
</div>

<div class="d-flex mb-1">
    <p class="lista-cursos-card__label me-1 quantico-font text-uppercase">Autor:</p>
    <P class="lista-cursos-card__id quantico-font text-uppercase">${curso.autor}</P>
</div>

<div class="w-100 d-flex justify-content-between mb-1">
    <div class="w-100">
        <p class="lista-cursos-card__label">Nombre del curso:</p>
        <h3 class="fs-3 wrap-break-word ">${curso.nombreCurso}</h3>
    </div>
</div>

<div class="w-100 mb-3">
    <p class="lista-cursos-card__label">Estado:</p>
    <select
        name="estado-${curso.id}"
        id="estado-${curso.id}"
        class="btn btn-secondary bg-strong-blue w-100"
    >
        <option value="false" ${curso.terminado ? "" : "selected"}>En curso</option>
        <option value="true" ${curso.terminado ? "selected" : ""}>Terminado</option>
    </select>
</div>

<div class="w-100 d-flex">
    <button id="editar-${curso.id}" class="btn btn-secondary w-100 me-2 d-flex align-items-center justify-content-center">
        <i class="fa-solid fa-pencil me-1 fs-1"></i>
        <span>Editar</span>
    </button>
    <button id="eliminar-${curso.id}" class="btn btn-danger w-100 d-flex align-items-center justify-content-center">
        <i class="fa-solid fa-x me-1 fs-1"></i>
        <span>Eliminar</span>
    </button>
</div>
`;

listaCursos.appendChild(card);

document.getElementById(`estado-${curso.id}`).addEventListener('change', () => {
    this.onEstadoChange(curso.id)
})
document.getElementById(`editar-${curso.id}`).addEventListener('click', () => {
    onClickEditarCurso({
        id: curso.id,
        autor: curso.autor,
        nombreCurso: curso.nombreCurso
    })
})

```

```
        })
    })
    document.getElementById(`eliminar-${curso.id}`).addEventListener('click', () => {
        this.eliminarCurso(curso.id)
    })
});
```

```
clearContainer(id) {
    document.getElementById(id).innerHTML = ""
}
```

```
}
```

```
function handleEditClick(event) {
    event.preventDefault();

    const id = document.getElementById('curso-id').value;
    const autor = document.getElementById('owner-name').value;
    const nombreCurso = document.getElementById('curso-name').value;

    if (!autor || !nombreCurso) {
        alert('Los campos "Autor" y "Nombre del curso" no pueden estar vacíos');
        return;
    }

    gestorDeCursos.actualizarCurso({
        id: Number(id),
        autor,
        nombreCurso
    });

    closeModal();
}
```

```
function setUserDisplayElement() {
    userDisplayElement = document.getElementById('usernameDisplay')
    if (!userDisplayElement) return
    const { username } = sessionData
    userDisplayElement.innerHTML = username
}
```

```
function onCursosInit() {
    const cursosBody = document.getElementById('cursosBody')
    if (!cursosBody) return;
    if (!sessionData) {
        redirect(`${window.location.origin}`)
    }
    setUserDisplayElement();
    const cursos = JSON.parse(localStorage.getItem('cursos')) || [];
    gestorDeCursos = new GestorDeCursos(cursos);
}
```

```
function setBtnLoginValue(htmlElement) {
  if (!sessionData) return
  if (!htmlElement) return
  const { username, email } = sessionData
  if (username && email) htmlElement.innerHTML = 'VER CURSOS'
}

// Obtiene la información de la sesión activa del localStorage
function setSessionData(sessionValues) {
  if (sessionValues) localStorage.setItem('session', JSON.stringify(sessionValues));
  sessionData = JSON.parse(localStorage.getItem('session'))
}

// open modal by id
function openModal(id) {
  document.getElementById(id).classList.add('open');
  document.body.classList.add('jw-modal-open');
}

// close currently open modal
function closeModal(event) {
  document.querySelector('.jw-modal.open').classList.remove('open');
  document.body.classList.remove('jw-modal-open');
}

function onLoginSubmit(event) {
  event.preventDefault()
  const formValues = Object.values(event.target).reduce((obj, field) => { obj[field.name] = field.value; return obj }, {})
  if (Object.values(formValues).some(el => !el)) return

  setSessionData({
    username: formValues["login-username"],
    email: formValues["login-email"]
  })

  event.target.reset()
  setBtnLoginValue(btnLoginElement)

  redirect(`${window.location.origin}/cursos.html`)
  closeModal()
}

function onAgregarCursoSubmit(event) {
  event.preventDefault();
  const formValues = Object.values(event.target).reduce((obj, field) => { obj[field.name] = field.value; return obj }, {})
  if (Object.values({ autor: formValues["owner-name"], nombreCurso: formValues["curso-name"], }))
```

```
}).some(el => !el)) return

event.target.reset()

if (formValues['curso-id']) {
  gestorDeCursos.actualizarCurso({
    id: Number(formValues['curso-id']),
    autor: formValues["owner-name"],
    nombreCurso: formValues["curso-name"],
  })
}

if (!formValues['curso-id']) {
  gestorDeCursos.agregarCurso(formValues["owner-name"], formValues["curso-name"])
}
closeModal()
}

function onBtnLoginClick() {
  if (sessionData) {
    redirect(`${window.location.origin}/pages/cursos.html`)
    return
  }
  openModal('loginModal');
}

function redirectToLogin() {
  window.location.href = "login.html";
}

function redirectToPrincipal() {
  window.location.href = "index.html";
}

function onClickEditarCurso({ id, autor, nombreCurso }) {
  openModal('agregarCursoModal');

  document.getElementById('agregarCursoModalTitle').innerHTML = 'Actualiza el curso'
  document.getElementById('curso-id').value = id
  document.getElementById('owner-name').value = autor
  document.getElementById('curso-name').value = nombreCurso
  document.getElementById('agregarCursoSubmitBtn').value = 'Actualizar curso'
  document.getElementById('agregarCursoSubmitBtn').addEventListener('click', handleEditClick);
}

function onClickAgregarCurso() {
  openModal('agregarCursoModal')

  document.getElementById('agregarCursoModalTitle').innerHTML = 'Agrega un curso'
  document.getElementById('curso-id').value = null
  document.getElementById('owner-name').value = ""
  document.getElementById('curso-name').value = ""
}
```

```

        document.getElementById('agregarCursoSubmitBtn').value = 'Aregar curso'
    }

function redirect(url) {
    window.location.href = url
}

function preventDefault(event) {
    event.preventDefault()
    return false
}

document.addEventListener('DOMContentLoaded', () => {
    btnLoginElement = document.getElementById('btnLogin')

    setSessionData(sessionData);
    setBtnLoginValue(btnLoginElement);
    onCursosInit()

    window.addEventListener('load', function () {
        // close modals on background click
        document.addEventListener('click', event => {
            if (event.target.classList.contains('jw-modal')) {
                closeModal();
            }
        });
    });
});

document.addEventListener('DOMContentLoaded', () => {
    const logoutButton = document.getElementById('btnLogout');

    if (logoutButton) {
        logoutButton.addEventListener('click', () => {
            localStorage.removeItem('token'); // Eliminar el token
            localStorage.removeItem('role'); // Eliminar el rol del usuario
            alert('Sesión cerrada correctamente.');
            window.location.href = 'login.html'; // Redirigir al login
        });
    }
});

```

### /public/js/miperfil.js

Este código maneja la visualización de un tooltip con la información del usuario cuando se hace clic en el botón de perfil (btnUser). Al cargarse la página, se asignan eventos para controlar la apertura y cierre del tooltip, asegurando que solo se muestre cuando el usuario haga clic en el botón y se oculte si se hace clic fuera de él.

Cuando el usuario hace clic en btnUser, se llama a la función mostrarPerfil(), que obtiene los datos del usuario desde el backend utilizando un fetch a la ruta /auth/user. Para ello, se extrae el token de autenticación desde localStorage y se envía en los encabezados de la solicitud. Si la respuesta es válida,

se actualizan los elementos HTML con el nombre y el rol del usuario. Si no hay token, se alerta al usuario y se le redirige a la página de inicio de sesión.

El tooltip se posiciona dinámicamente debajo del botón de usuario. Se calcula la posición exacta del botón usando `getBoundingClientRect()`, ajustando la ubicación del tooltip con `scrollY` y `scrollX` para asegurar que siempre se alinee correctamente. Además, si el tooltip ya está visible, el clic en el botón lo oculta, permitiendo un comportamiento de alternancia.

Finalmente, se implementa un evento global en `document` para detectar clics fuera del tooltip. Si el usuario hace clic en cualquier parte de la página que no sea el botón o el tooltip mismo, este se oculta automáticamente. De esta manera, se garantiza una experiencia fluida y que el tooltip no permanezca abierto cuando no es necesario.

```
document.addEventListener("DOMContentLoaded", function () {
  const btnUser = document.getElementById("btnUser");
  const perfilTooltip = document.getElementById("perfilTooltip");

  if (btnUser) {
    btnUser.addEventListener("click", async function (event) {
      event.stopPropagation(); // Evita que el clic cierre el tooltip inmediatamente
      await mostrarPerfil(); // Obtiene los datos del usuario y los muestra

      // Calcula la posición del botón y coloca el tooltip debajo
      const rect = btnUser.getBoundingClientRect();
      perfilTooltip.style.top = `${rect.bottom + window.scrollY + 5}px`; // Justo debajo del botón
      perfilTooltip.style.left = `${rect.left + window.scrollX}px`; // Alineado con el botón
      perfilTooltip.style.display = (perfilTooltip.style.display === "none") ? "block" : "none";
    });
  }

  // Ocultar el tooltip si se hace clic fuera de él
  document.addEventListener("click", function (event) {
    if (!perfilTooltip.contains(event.target) && event.target !== btnUser) {
      perfilTooltip.style.display = "none";
    }
  });
});

// Función para obtener y mostrar los datos del usuario
async function mostrarPerfil() {
  try {
    const token = localStorage.getItem("token");
    if (!token) {
      alert("Debes iniciar sesión primero.");
      window.location.href = "login.html";
      return;
    }

    const response = await fetch("/auth/user", {
      headers: { Authorization: `Bearer ${token}` }
    });
  }
}
```

```

if (!response.ok) {
    throw new Error("Error al obtener la información del usuario");
}

const userData = await response.json();

document.getElementById("perfilNombre").innerText = userData.username;
document.getElementById("perfilTipoUser").innerText =
    userData.role === "admin" ? "Administrador" : "Estudiante";
} catch (error) {
    console.error("Error al obtener el perfil del usuario:", error);
}
}

```

## /public/js/normalcursos.js

Este código maneja la carga y visualización de los cursos en los que un usuario está inscrito. Al iniciar la página, se obtiene la lista de cursos del usuario autenticado, almacenándolos en una variable para permitir filtrado y paginación infinita. Se cargan los cursos en bloques de cuatro, y al hacer scroll, se añaden más automáticamente, siempre y cuando no se esté filtrando o en medio de una carga.

Cada curso se muestra con su información y un botón para salir del curso. Si el usuario decide abandonarlo, se envía una solicitud DELETE al servidor para eliminar su inscripción. Si la solicitud es exitosa, el curso se elimina de la vista sin necesidad de recargar la página. Además, antes de ejecutar la acción, se muestra un mensaje de confirmación para evitar eliminaciones accidentales.

También se implementa un sistema de búsqueda en tiempo real. Si el usuario escribe en el campo de búsqueda, se filtran los cursos por título o categoría, y solo se muestran los que coincidan con la consulta. Si el campo de búsqueda se vacía, se restablece la lista de cursos a su estado original, permitiendo nuevamente la paginación infinita. El buscador se puede abrir o cerrar con un botón, y si el usuario hace clic fuera de él, se oculta automáticamente.

El código incluye medidas de seguridad y control de acceso. Si el usuario es un administrador, se bloquea la vista con un mensaje que indica que solo los usuarios regulares pueden acceder, redirigiéndolo después de unos segundos. Además, se verifica constantemente la validez del token de autenticación. Si el token expira, se elimina la sesión y se redirige al usuario a la página de inicio de sesión, asegurando que solo usuarios autenticados tengan acceso a la información.

```

document.addEventListener("DOMContentLoaded", async function () {
    let lastCourseIndex = 0;
    const itemsPerPage = 4;
    let allUserCourses = [];
    let isFiltering = false;
    let isLoading = false;

    const cursosContainer = document.getElementById("cursos-container");
    const searchInput = document.getElementById("searchInput");

    async function fetchUserCourses() {
        if (isLoading || isFiltering) return;
        isLoading = true;

        try {
            ...
        } catch (error) {
            ...
        }
    }

    ...
}

```

```
const token = localStorage.getItem('token');
if (!token) {
  alert('Debes iniciar sesión primero.');
  window.location.href = 'login.html';
  return;
}

const response = await fetch('/courses/user-courses', {
  headers: { 'Authorization': `Bearer ${token}` }
});

if (!response.ok) {
  throw new Error('Error al obtener los cursos');
}

const data = await response.json();
console.log("✅ Cursos obtenidos:", data.courses);

allUserCourses = data.courses;
renderNextCourses(); // Cargar los primeros 4 cursos
} catch (error) {
  console.error("Error al obtener los cursos:", error);
} finally {
  isLoading = false;
}
}

function renderNextCourses() {
  if (lastCourseIndex >= allUserCourses.length) return;

  const nextCourses = allUserCourses.slice(lastCourseIndex, lastCourseIndex + itemsPerPage);
  lastCourseIndex += itemsPerPage;
  appendCursos(nextCourses);
}

function appendCursos(cursos) {
  cursos.forEach(curso => {
    const cursoDiv = document.createElement("div");
    cursoDiv.classList.add("cursoNormal");
    cursoDiv.id = `curso-${curso.id_user_course}`;

    cursoDiv.innerHTML =
      `

<div class="contenidoCurNor">
          
          <div class="textNor">
            <button class="botonDetele btn-course-action" id="btnDelete" data-id="${curso.id_user_course}">
              Salir del curso
            </button>
            <h3>${curso.title}</h3>
            <p class="fw-bold">Categoría: ${curso.category}</p>

`;
```

```
        <p id="textoP">${curso.description}</p>
      </div>
    </div>
  </div>
`;
cursosContainer.appendChild(cursoDiv);
});

document.querySelectorAll(".botonDelete").forEach(button => {
  button.addEventListener("click", function () {
    const id_user_course = this.getAttribute("data-id");
    salirDelCurso(id_user_course);
  });
});
}

async function salirDelCurso(id_user_course) {
  try {
    const token = localStorage.getItem('token');
    if (!token) {
      alert('Debes iniciar sesión primero.');
      window.location.href = 'login.html';
      return;
    }

    const confirmar = confirm("¿Estás seguro de que quieres salir del curso?");
    if (!confirmar) return;

    const response = await fetch(`/courses/user-courses/${id_user_course}`, {
      method: 'DELETE',
      headers: { 'Authorization': `Bearer ${token}` }
    });

    if (!response.ok) {
      throw new Error('Error al salir del curso');
    }

    alert('Has salido del curso correctamente.');
    document.getElementById(`curso-${id_user_course}`).remove();
  } catch (error) {
    console.error("Error al salir del curso:", error);
    alert('Hubo un error al intentar salir del curso.');
  }
}

searchInput.addEventListener("input", function () {
  const searchTerm = this.value.toLowerCase().trim();

  if (searchTerm) {
    isFiltering = true;
    cursosContainer.innerHTML = "";
  }
});
```

```

const filteredCourses = allUserCourses.filter(curso =>
  curso.title.toLowerCase().includes(searchTerm) ||
  curso.category.toLowerCase().includes(searchTerm)
);

appendCursos(filteredCourses);
} else {
  isFiltering = false;
  cursosContainer.innerHTML = "";
  lastCourseIndex = 0;
  renderNextCourses();
}
});

window.addEventListener("scroll", function () {
  if (window.innerHeight + window.scrollY >= document.body.offsetHeight - 100) {
    if (!isFiltering && !isLoading) {
      renderNextCourses();
    }
  }
});
};

document.getElementById("toggleSearch").addEventListener("click", function () {
  let searchBar = document.getElementById("searchBox");

  if (searchBar.style.display === "none" || searchBar.style.display === "") {
    searchBar.style.display = "block";
    document.getElementById("searchInput").focus();
  } else {
    searchBar.style.display = "none";
  }
});
};

document.addEventListener("click", function (event) {
  let searchBar = document.getElementById("searchBox");
  let toggleImage = document.getElementById("toggleSearch");

  if (!searchBar.contains(event.target) && event.target !== toggleImage) {
    searchBar.style.display = "none";
  }
});
};

fetchUserCourses();
});

document.addEventListener("DOMContentLoaded", function () {
  const userRole = localStorage.getItem("role");

  if (userRole === "admin") {
    // Crear pantalla de bloqueo
    const overlay = document.createElement("div");
    overlay.style.position = "fixed";
    overlay.style.top = "0";
  }
});

```

```

overlay.style.left = "0";
overlay.style.width = "100vw";
overlay.style.height = "100vh";
overlay.style.backgroundColor = "black";
overlay.style.color = "white";
overlay.style.display = "flex";
overlay.style.flexDirection = "column";
overlay.style.justifyContent = "center";
overlay.style.alignItems = "center";
overlay.style.fontSize = "2rem";
overlay.style.fontWeight = "bold";
overlay.style.zIndex = "9999";
overlay.innerHTML = `

    X SOLO USUARIOS REGULARES PUEDEN VER ESTO X

    <br>
    <button id="btnRegresarAdmin" style="

        margin-top: 20px;
        padding: 15px 30px;
        font-size: 1.5rem;
        font-weight: bold;
        color: white;
        background-color: red;
        border: none;
        border-radius: 10px;
        cursor: pointer;
    >`← BACK Ir a Administración</button>

`;
}

// Agregar el overlay al body
document.body.appendChild(overlay);

// Bloquear interacciones (opcional)
document.body.style.overflow = "hidden";

// Agregar evento al botón para regresar a adminCursos.html
document.getElementById("btnRegresarAdmin").addEventListener("click", function () {
    window.location.href = "adminCursos.html";
});

// Redirigir automáticamente después de 3 segundos
setTimeout(() => {
    window.location.href = "adminCursos.html";
}, 3000);
}

document.addEventListener("DOMContentLoaded", function () {
    const token = localStorage.getItem("token");

    if (token) {
        const decoded = JSON.parse(atob(token.split(".")[1])); // Decodifica el token
        const currentTime = Math.floor(Date.now() / 1000); // Tiempo actual en segundos
    }
});

```

```

if (decoded.exp < currentTime) {
    // Token vencido, eliminar y redirigir al index
    localStorage.removeItem("token");
    localStorage.removeItem("role");
    alert("Tu sesión ha expirado. Inicia sesión nuevamente.");
    window.location.href = "index.html";
} else {
    // Configurar verificación en intervalos de tiempo (cada 30 segundos)
    setInterval(() => {
        const now = Math.floor(Date.now() / 1000);
        if (decoded.exp < now) {
            localStorage.removeItem("token");
            localStorage.removeItem("role");
            alert("Tu sesión ha expirado. Inicia sesión nuevamente.");
            window.location.href = "index.html";
        }
    }, 30000); // Se ejecuta cada 30 segundos
}
});

```

### /public/js/register.js

Este código gestiona el registro de nuevos usuarios en la aplicación. Se activa cuando el usuario envía el formulario de registro (registerForm). Primero, evita que la página se recargue automáticamente usando event.preventDefault(). Luego, captura los valores ingresados en los campos de nombre de usuario, contraseña y rol, asegurándose de que la información esté lista para ser enviada al servidor.

Antes de enviar los datos, se imprime un mensaje en la consola con la información capturada, lo que ayuda a depurar errores si algo sale mal. Luego, se realiza una solicitud POST al endpoint /auth/register, enviando los datos en formato JSON al backend. Si la respuesta es exitosa (response.ok), se muestra un mensaje de alerta indicando que el registro fue exitoso y el usuario es redirigido a la página de inicio de sesión.

Si la respuesta del servidor indica un error, se muestra un mensaje de alerta con el error devuelto por el backend. Si no hay un mensaje específico, se muestra un mensaje genérico indicando que hubo un problema en el registro. Este manejo de errores permite al usuario recibir retroalimentación clara en caso de que ocurra algún problema durante el proceso.

El código es funcional, pero podría mejorarse agregando validaciones previas en el frontend para evitar el envío de datos vacíos o incorrectos. También podría deshabilitar el botón de envío mientras se procesa la solicitud, evitando que el usuario envíe múltiples peticiones accidentalmente. Estas mejoras optimizarían la experiencia del usuario y harían la interfaz más robusta.

```

document.getElementById('registerForm').addEventListener('submit', async (event) => {
    event.preventDefault();
    const username = document.getElementById('username').value;
    const password = document.getElementById('password').value;
    const role = document.getElementById('role').value;

    console.log('Datos enviados:', { username, password, role }); // Depuración
}

```

```

const response = await fetch('/auth/register', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({ username, password, role })
});

const data = await response.json();
if (response.ok) {
  alert('Registro exitoso');
  window.location.href = 'login.html';
} else {
  alert(data.error || 'Error en el registro');
}
});

```

## /src/config/db.js

Este código configura la conexión a una base de datos MySQL alojada en Clever Cloud. Usa mysql2 y un pool de conexiones para manejar múltiples solicitudes sin crear y cerrar conexiones constantemente. Esto hace que la aplicación sea más eficiente y rápida. Clever Cloud genera credenciales aleatorias, por eso los nombres de usuario y contraseñas son largos y complicados. No es algo que el desarrollador elija. Simplemente así funciona el servicio.

La configuración incluye varios parámetros importantes. Primero, el host, que define dónde está la base de datos. Luego, el usuario y la contraseña, que permiten autenticarse. También está el database, que indica qué base de datos se usará. El port, que normalmente es 3306, pero podría cambiar si es necesario. Además, hay configuraciones como connectionLimit, que evita que demasiadas conexiones sobrecarguen el servidor, y queueLimit, que controla cuántas solicitudes pueden esperar si todas las conexiones están ocupadas.

Después de configurar el pool, se hace una prueba. Se obtiene una conexión con db.getConnection(). Si todo va bien, el programa imprime "Conectado a MySQL". Si hay un error, lo muestra en la consola con detalles. Así, si algo falla, se puede ver qué está pasando. La conexión se libera de inmediato para que pueda ser reutilizada por otra solicitud. Esto es clave para evitar bloqueos o que la base de datos se quede sin conexiones disponibles.

Por último, el objeto db se exporta con module.exports = db. Esto permite que otros archivos de la aplicación usen la conexión sin necesidad de definirla nuevamente. Básicamente, cualquier parte del código que necesite acceder a la base de datos solo tiene que importar db y listo. Todo está centralizado en un solo archivo. Más ordenado, más fácil de mantener.

```

const mysql = require('mysql2');

const db = mysql.createPool({
  host: process.env.DB_HOST || 'bbz7w1sdy5yaf7ocvlva-mysql.services.clever-cloud.com',
  user: process.env.DB_USER || 'ufvomre2iqltqutc',
  password: process.env.DB_PASSWORD || 'Y9m3AeTqyxFTL2l0ZPZn',
  database: process.env.DB_NAME || 'bbz7w1sdy5yaf7ocvlva',
  port: process.env.DB_PORT || 3306,
  waitForConnections: true,
  connectionLimit: 10,
});

```

```

    queueLimit: 0
});

db.getConnection((err, connection) => {
  if (err) {
    console.error('Error en la conexión a MySQL:', err);
  } else {
    console.log('Conectado a MySQL');
    connection.release();
  }
});

module.exports = db;

```

## /src/controllers/authController.js

Este código maneja la autenticación y gestión de usuarios en una aplicación web utilizando MySQL, JSON Web Tokens (JWT) y bcrypt para el cifrado de contraseñas. Se definen tres funciones principales: register, login y getUserProfile. Cada una se encarga de un aspecto diferente del sistema de usuarios.

En la función register, se recibe un nombre de usuario, una contraseña y un rol. Primero, se valida que los campos no estén vacíos y que el rol sea válido. Luego, se verifica si el usuario ya existe en la base de datos. Si no, se cifra la contraseña con bcrypt.hash() antes de almacenarla en la base de datos, lo que mejora la seguridad. Finalmente, se envía una respuesta de éxito o un error en caso de fallos en el proceso.

En la función login, se busca al usuario en la base de datos y se compara la contraseña ingresada con la almacenada usando bcrypt.compare(). Si las credenciales son correctas, se genera un **JWT** con una duración de 30 minutos. Este token incluye información clave como el ID del usuario, su rol y un identificador único del token. Si todo es correcto, el token y el rol del usuario se devuelven en la respuesta, permitiendo al cliente autenticarse en futuras solicitudes.

La función getUserProfile obtiene el perfil del usuario autenticado. Usa el id\_user extraído del token para buscar en la base de datos y devuelve su nombre de usuario y rol. Si no se encuentra el usuario, se envía un error 404. En caso de problemas con la base de datos, se devuelve un error 500. Este método asegura que solo usuarios autenticados puedan acceder a su información.

En general, este código implementa un sistema de autenticación robusto con verificación de credenciales, cifrado de contraseñas y seguridad basada en tokens. Sin embargo, podría mejorarse con una mejor gestión de errores y registros de actividad para detectar intentos de acceso sospechosos. También sería recomendable invalidar tokens en el servidor cuando el usuario cierre sesión, evitando que sigan siendo válidos después de su expiración.

```

const db = require('../config/db');
const jwt = require('jsonwebtoken');
const bcrypt = require('bcryptjs');

exports.register = async (req, res) => {
  const { username, password, role } = req.body;

  try {

```

```
if (!username || !password || !role) {
  return res.status(400).json({ error: 'Todos los campos son obligatorios' });
}

if (!['admin', 'normal'].includes(role)) {
  return res.status(400).json({ error: 'Rol no válido' });
}

const [existingUser] = await db.promise().query('SELECT * FROM user WHERE username = ?', [username]);
if (existingUser.length > 0) {
  return res.status(400).json({ error: 'El usuario ya existe' });
}

const hashedPassword = await bcrypt.hash(password, 10);
await db.promise().query('INSERT INTO user (username, password, role) VALUES (?, ?, ?)', [
  username, hashedPassword, role
]);

res.status(201).json({ message: 'Usuario registrado exitosamente' });

} catch (err) {
  res.status(500).json({ error: 'Error en el servidor', details: err.message });
}
};

exports.login = async (req, res) => {
  const { username, password } = req.body;

  try {
    const [user] = await db.promise().query('SELECT * FROM user WHERE username = ?', [username]);
    if (user.length === 0) {
      return res.status(400).json({ error: 'Usuario no encontrado' });
    }

    const isMatch = await bcrypt.compare(password, user[0].password);
    if (!isMatch) {
      return res.status(400).json({ error: 'Contraseña incorrecta' });
    }

    if (!process.env.JWT_SECRET) {
      return res.status(500).json({ error: 'Clave JWT no configurada' });
    }

    const issuedAt = Math.floor(Date.now() / 1000); // Hora de emisión (en segundos)
    const expiresAt = issuedAt + 1800; // Expira en media hora

    const tokenPayload = {
      id_user: user[0].id_user,
      role: user[0].role,
      iat: issuedAt,
```

```

    exp: expiresAt,
    token_id: `${user[0].id_user}-${issuedAt}` // Identificador único del token
};

const token = jwt.sign(tokenPayload, process.env.JWT_SECRET);

res.status(200).json({ token, role: user[0].role });

} catch (err) {
  res.status(500).json({ error: 'Error en el servidor', details: err.message });
}
};

exports.getUserProfile = async (req, res) => {
  try {
    const userId = req.user.id_user;

    const [user] = await db.promise().query(
      "SELECT username, role FROM user WHERE id_user = ?",
      [userId]
    );

    if (user.length === 0) {
      return res.status(404).json({ error: "Usuario no encontrado" });
    }

    res.status(200).json(user[0]);
  } catch (error) {
    console.error("Error al obtener perfil:", error);
    res.status(500).json({ error: "Error interno del servidor" });
  }
};

```

### /src/controllers/courseController.js

Este código maneja la gestión de cursos en la aplicación, incluyendo la obtención, creación, actualización, eliminación y la inscripción de usuarios en los cursos. Se conecta a una base de datos MySQL utilizando db.promise().query() para realizar consultas de manera asíncrona. Todas las respuestas se envían en formato JSON, asegurando que el frontend pueda interpretarlas fácilmente.

La función getPaginatedCourses permite obtener cursos con paginación. Usa los parámetros page y limit para devolver una cantidad específica de cursos y calcular cuántas páginas hay en total. Mientras que loadMoreCourses implementa un sistema de carga progresiva, donde solo se recuperan cursos con un id\_course mayor al último obtenido. Esto permite una experiencia más fluida sin sobrecargar el servidor con datos innecesarios.

Para gestionar la relación entre usuarios y cursos, getUserCourses obtiene los cursos en los que un usuario está inscrito. enrollInCourse permite a un usuario inscribirse en un curso, verificando antes si ya está inscrito para evitar duplicados. leaveCourse hace lo contrario: permite salir de un curso eliminando la relación en la base de datos. Si el usuario no está inscrito, devuelve un error.

Las funciones `createCourse`, `updateCourse` y `deleteCourse` permiten administrar los cursos. `createCourse` inserta un nuevo curso en la base de datos, asegurando que todos los campos obligatorios estén completos. `updateCourse` modifica la información de un curso existente, verificando que el curso realmente exista antes de actualizarlo. `deleteCourse` elimina un curso de la base de datos, pero solo si realmente está registrado. Si no se encuentra, devuelve un error 404.

Cada función maneja los errores de forma específica. Si ocurre un problema en la base de datos o si falta algún dato, se devuelve un mensaje de error con detalles. También se registran errores en la consola para facilitar la depuración. Esto ayuda a identificar rápidamente fallos en el código o problemas con la base de datos sin afectar la experiencia del usuario.

```
const db = require('../config/db');

exports.getPaginatedCourses = async (req, res) => {
  try {
    const page = parseInt(req.query.page) || 1;
    const limit = 4; // Cursos por página
    const offset = (page - 1) * limit;

    const [courses] = await db.promise().query(
      'SELECT id_course, title, description, category, image_url FROM course LIMIT ? OFFSET ?',
      [limit, offset]
    );

    const [total] = await db.promise().query('SELECT COUNT(*) AS total FROM course');
    const totalPages = Math.ceil(total[0].total / limit);

    res.status(200).json({
      courses,
      totalPages,
      currentPage: page
    });
  } catch (error) {
    console.error("Error en getPaginatedCourses:", error);
    res.status(500).json({ error: 'Error al obtener los cursos', details: error.message });
  }
};

exports.loadMoreCourses = async (req, res) => {
  try {
    const lastCourseId = parseInt(req.query.lastCourseId) || 0;
    const limit = parseInt(req.query.limit) || 4;

    let query = `
      SELECT id_course, title, description, category, image_url
      FROM course
    `;

    let params = [];

    if (lastCourseId > 0) {
```

```

        query += ` WHERE id_course > ? `;
        params.push(lastCourseld);
    }

    query += ` ORDER BY id_course ASC LIMIT ? `;
    params.push(limit);

    const [courses] = await db.promise().query(query, params);

    res.status(200).json({
        courses
    });

} catch (error) {
    console.error("✖ Error en loadMoreCourses:", error);
    res.status(500).json({ error: "Error al obtener los cursos", details: error.message });
}
};

exports.getUserCourses = async (req, res) => {
try {
    const userId = req.user.id_user;

    // 🔔 Obtener todos los cursos inscritos SIN paginación
    const [courses] = await db.promise().query(`

        SELECT uc.id_user_course, c.id_course, c.title, c.description, c.category, c.image_url
        FROM user_courses uc
        JOIN course c ON uc.id_course = c.id_course
        WHERE uc.id_user = ?
    `, [userId]);

    console.log("Cursos inscritos en el backend:", courses);

    res.status(200).json({ courses });

} catch (error) {
    console.error("Error en getUserCourses:", error);
    res.status(500).json({ error: "Error al obtener los cursos del usuario", details: error.message });
}
};

exports.enrollInCourse = async (req, res) => {
try {
    const userId = req.user?.id_user;
    const courseld = req.params.id;

    if (!userId || !courseld) {
        return res.status(400).json({ error: "Faltan datos para la inscripción" });
    }

    const [existingEnrollment] = await db.promise().query(`

        SELECT *
        FROM user_courses
        WHERE id_user = ?
        AND id_course = ?
    `, [userId, courseld]);
}

```

```

    'SELECT * FROM user_courses WHERE id_user = ? AND id_course = ?',
    [userId, courseld]
);

if (existingEnrollment.length > 0) {
    return res.status(400).json({ error: "⚠ Ya estás inscrito en este curso" });
}

await db.promise().query(
    'INSERT INTO user_courses (id_user, id_course) VALUES (?, ?)',
    [userId, courseld]
);

res.status(201).json({ message: "Inscripción exitosa" });

} catch (error) {
    console.error("Error en enrollInCourse:", error);
    res.status(500).json({ error: "Error al inscribirse en el curso", details: error.message });
}
};

exports.leaveCourse = async (req, res) => {
try {
    console.log("Usuario autenticado:", req.user);
    const userId = req.user?.id_user;
    const courseld = req.params.id_course;

    if (!userId || !courseld) {
        return res.status(400).json({ error: "Falta el ID del usuario o del curso" });
    }

    const [result] = await db.promise().query(`DELETE FROM user_courses WHERE id_user = ? AND id_course = ?`, [userId, courseld]);

    if (result.affectedRows === 0) {
        return res.status(404).json({ error: "⚠ No estás inscrito en este curso" });
    }

    res.status(200).json({ message: "Has salido del curso correctamente" });

} catch (error) {
    console.error("Error en leaveCourse:", error);
    res.status(500).json({ error: "Error al salir del curso", details: error.message });
}
};

exports.createCourse = async (req, res) => {
try {
    const { title, description, category, image_url } = req.body;

```

```

if (!title || !description || !category) {
    return res.status(400).json({ error: 'Todos los campos son obligatorios' });
}

const [result] = await db.promise().query(
    'INSERT INTO course (title, description, category, image_url) VALUES (?, ?, ?, ?)',
    [title, description, category, image_url]
);

res.status(201).json({ message: 'Curso creado correctamente', courseId: result.insertId });

} catch (error) {
    console.error("Error en createCourse:", error);
    res.status(500).json({ error: 'Error al crear el curso', details: error.message });
}
};

exports.updateCourse = async (req, res) => {
    try {
        const { id } = req.params;
        const { title, category, description, image_url } = req.body;

        if (!title || !category || !description || !image_url) {
            return res.status(400).json({ error: "⚠️ Todos los campos son obligatorios" });
        }

        const query = `
            UPDATE course
            SET title = ?, category = ?, description = ?, image_url = ?
            WHERE id_course = ?
        `;
        const values = [title, category, description, image_url, id];
        const [result] = await db.promise().query(query, values);

        if (result.affectedRows === 0) {
            return res.status(404).json({ error: "⚠️ Curso no encontrado" });
        }

        res.json({ message: "Curso actualizado correctamente" });

    } catch (error) {
        console.error("Error en updateCourse:", error);
        res.status(500).json({ error: "Error interno del servidor", details: error.message });
    }
};

exports.deleteCourse = async (req, res) => {
    try {
        const { id } = req.params;
    }

```

```

const [course] = await db.promise().query('SELECT * FROM course WHERE id_course = ?',
[id]);
if (course.length === 0) {
    return res.status(404).json({ error: '⚠️ Curso no encontrado' });
}

await db.promise().query('DELETE FROM course WHERE id_course = ?', [id]);

res.status(200).json({ message: 'Curso eliminado correctamente' });

} catch (error) {
    console.error("Error en deleteCourse:", error);
    res.status(500).json({ error: 'Error al eliminar el curso', details: error.message });
}
};

```

### /src/middleware/authenticate.js

Este código define un **middleware de autenticación y autorización** en Node.js utilizando jsonwebtoken (JWT). Su objetivo es verificar que los usuarios estén autenticados y, opcionalmente, restringir el acceso según su rol. Se exporta como una función que recibe un parámetro opcional (roleRequired), permitiendo que algunas rutas requieran un rol específico, como "admin".

Cuando se ejecuta, el middleware primero revisa si el encabezado Authorization está presente y tiene el formato correcto (Bearer <token>). Si falta o es inválido, se responde con un **401 - Acceso denegado**. Luego, extrae el token y verifica si la clave secreta (JWT\_SECRET) está configurada en las variables de entorno. Si no lo está, devuelve un **500 - Error interno del servidor**.

Si el token es válido, se decodifica con jwt.verify() y se almacena la información del usuario en req.user, permitiendo que otros middleware o controladores accedan a sus datos. Si la ruta requiere un rol específico (roleRequired no es null), se compara con el rol del usuario en el token. Si no coincide, se devuelve un **403 - Acceso prohibido**.

El código maneja distintos tipos de errores de autenticación de JWT de forma detallada. Si el token ha expirado, responde con **440 - Sesión expirada**. Si el token es inválido o está corrupto, devuelve **498 - Token inválido**. Si el token aún no es válido (NotBeforeError), indica que hay que esperar con **425 - Token aún no válido**. Para otros errores inesperados, usa **520 - Error desconocido**.

Este middleware es esencial para proteger rutas que requieren autenticación, asegurando que solo los usuarios autorizados accedan a ellas. Además, su manejo detallado de errores ayuda a proporcionar respuestas más claras al cliente, mejorando la experiencia del usuario y la seguridad de la aplicación.

```

const jwt = require('jsonwebtoken');

module.exports = (roleRequired = null) => {
    return (req, res, next) => {
        try {
            const authHeader = req.header('Authorization');
            if (!authHeader || !authHeader.startsWith('Bearer ')) {
                return res.status(401).json({ error: 'Acceso denegado, token no proporcionado' });
            }
        }
    };
};

```

```

const token = authHeader.split(' ')[1];
if (!process.env.JWT_SECRET) {
  return res.status(500).json({ error: 'Error interno del servidor, clave JWT no configurada' });
}

const verified = jwt.verify(token, process.env.JWT_SECRET);
req.user = verified;

// Verificar si el usuario tiene el rol adecuado
if (roleRequired && req.user.role !== roleRequired) {
  return res.status(403).json({ error: 'Acceso prohibido, rol no autorizado' });
}

next();
} catch (err) {
  if (err instanceof jwt.TokenExpiredError) {
    return res.status(440).json({ error: 'Sesión expirada, inicia sesión nuevamente' }); // 440
  Login Time-out (usado en algunos servidores)
  }
  if (err instanceof jwt.JsonWebTokenError) {
    return res.status(498).json({ error: 'Token inválido o corrupto' }); // 498 Invalid Token (no
estándar, pero usado por algunos servicios)
  }
  if (err instanceof jwt.NotBeforeError) {
    return res.status(425).json({ error: 'Token aún no es válido, espera hasta su activación' }); // 425 Too Early (cuando algo se usa antes de tiempo)
  }

  return res.status(520).json({ error: 'Error de autenticación desconocido' }); // 520 Unknown
Error (utilizado por Cloudflare y otros servicios)
}
};

},

```

### /src/middleware/errorHandler.js

Este código define un **middleware de manejo de errores** para una aplicación en Node.js con Express. Se exporta como una función que recibe cuatro parámetros: err (el objeto de error), req (la solicitud), res (la respuesta) y next (para pasar el control al siguiente middleware si es necesario). La presencia de estos cuatro parámetros indica que Express lo reconoce como un **manejador de errores global**.

Cuando ocurre un error en cualquier parte de la aplicación, este middleware captura el error y lo registra en la consola con `console.error()`, lo que ayuda a depurar problemas en el servidor. Luego, determina el código de estado HTTP a devolver: si el error tiene un `statusCode` definido, lo usa; de lo contrario, asigna el valor 500, que representa un **error interno del servidor**.

El mensaje de error también se personaliza. Si el error tiene un `message`, lo usa para dar detalles más específicos. Si no, se devuelve un mensaje genérico indicando un problema interno. Finalmente, se envía la respuesta en formato JSON con un objeto que contiene la clave `error`, permitiendo al frontend interpretar el fallo de manera estructurada.

Este middleware es útil porque centraliza el manejo de errores. En lugar de escribir lógica de manejo de errores en cada ruta, cualquier error no controlado es capturado aquí. Además, se puede mejorar agregando más detalles en la respuesta, como un stack trace en entornos de desarrollo o códigos de error personalizados según el tipo de problema.

```
module.exports = (err, req, res, next) => {
  console.error("Error en el servidor:", err);

  let statusCode = err.statusCode || 500;
  let message = err.message || "Error interno del servidor";

  res.status(statusCode).json({ error: message });
};
```

## /src/routes/authRoutes.js

Este código configura las rutas de autenticación en una aplicación Node.js utilizando **Express**. Define un **router** con tres rutas principales: registro de usuario, inicio de sesión y obtención del perfil del usuario autenticado. Se importa el controlador de autenticación (authController), que contiene la lógica para manejar estas solicitudes, y el middleware authenticate, que protege ciertas rutas.

La primera ruta POST /register permite registrar un usuario. Llama a authController.register, que valida los datos y almacena al usuario en la base de datos con una contraseña cifrada. La segunda, POST /login, gestiona el inicio de sesión, verificando las credenciales y generando un **JSON Web Token (JWT)** para que el usuario pueda autenticarse en futuras solicitudes.

La tercera ruta GET /user es diferente: requiere autenticación. Para acceder a ella, el usuario debe enviar un token válido en el encabezado Authorization. Se usa el middleware authenticate(), que valida el token y extrae la información del usuario antes de pasar la solicitud a authController.getUserProfile, que devuelve los datos del usuario autenticado.

Este enfoque modulariza las rutas de autenticación, manteniendo el código organizado y fácil de extender. Al separar el controlador y el middleware, la aplicación se vuelve más mantenible y segura, evitando duplicaciones de código y asegurando que solo los usuarios autenticados puedan acceder a información protegida.

```
const express = require('express');
const router = express.Router();
const authController = require('../controllers/authController');
const authenticate = require("../middleware/authenticate");

router.post('/register', authController.register);
router.post('/login', authController.login);

router.get('/user', authenticate(), authController.getUserProfile);

module.exports = router;
```

## /src/routes/courseRoutes.js

Este código define las rutas para la gestión de cursos en una aplicación **Node.js con Express**. Se importa el courseController, que maneja la lógica de cada operación, y el middleware authenticate, que protege las rutas restringidas. También se importa la conexión db para ejecutar consultas directas a la base de datos cuando es necesario.

Las rutas permiten diferentes operaciones sobre los cursos. GET /paginated devuelve los cursos en formato paginado, mientras que GET /load-more carga más cursos progresivamente. Ambas requieren autenticación, asegurando que solo usuarios registrados accedan a la información. GET /user-courses obtiene los cursos en los que el usuario está inscrito, verificando su identidad mediante el token.

Las rutas protegidas con authenticate('admin') están reservadas para administradores. Estas incluyen POST / para crear un curso, PUT /:id para actualizarlo y DELETE /:id para eliminarlo. Esto garantiza que solo usuarios con el rol adecuado puedan modificar los cursos, evitando accesos no autorizados.

Además, POST /enroll/:id permite a los usuarios inscribirse en un curso, mientras que DELETE /user-courses/:id\_user\_course les permite darse de baja. Esta última consulta verifica que el usuario realmente esté inscrito antes de eliminar la relación en la base de datos. Si no lo está, devuelve un error 404.

Por último, GET /all obtiene todos los cursos sin paginación, útil para mostrar la lista completa sin restricciones. Todas las operaciones incluyen manejo de errores, asegurando respuestas claras en caso de problemas. Este enfoque modular y seguro permite una administración eficiente de los cursos dentro de la plataforma.

```
const express = require('express');
const router = express.Router();
const courseController = require('../controllers/courseController');
const authenticate = require('../middleware/authenticate');
const db = require('../config/db');

router.get('/paginated', authenticate(), courseController.getPaginatedCourses);
router.post('/', authenticate('admin'), courseController.createCourse);
router.delete('/:id', authenticate('admin'), courseController.deleteCourse);
router.put('/:id', authenticate('admin'), courseController.updateCourse);
router.get('/load-more', authenticate(), courseController.loadMoreCourses);
router.get('/user-courses', authenticate(), courseController.getUserCourses);
router.post('/enroll/:id', authenticate(), courseController.enrollInCourse);

// ♦ **Corrección en la eliminación de inscripción**
router.delete('/user-courses/:id_user_course', authenticate(), async (req, res) => {
  try {
    const userId = req.user.id_user;
    const { id_user_course } = req.params;

    const [result] = await db.promise().query(
      'DELETE FROM user_courses WHERE id_user_course = ? AND id_user = ?',
      [id_user_course, userId]
    );

    if (result.affectedRows === 0) {
```

```

        return res.status(404).json({ error: "No estás inscrito en este curso" });
    }

    res.status(200).json({ message: "Has salido del curso correctamente" });

} catch (error) {
    console.error("Error al salir del curso:", error);
    res.status(500).json({ error: "Error al salir del curso", details: error.message });
}
});

router.get('/all', authenticate(), async (req, res) => {
try {
    const [courses] = await db.promise().query(
        'SELECT id_course, title, description, category, image_url FROM course'
    );
    res.status(200).json({ courses });
} catch (error) {
    console.error("Error al obtener todos los cursos:", error);
    res.status(500).json({ error: "Error al obtener los cursos", details: error.message });
}
};

module.exports = router;

```

## /src/app.js

Este código define las rutas para la gestión de cursos en una aplicación **Node.js con Express**. Se organizan de manera que cada acción sobre los cursos tenga su propia ruta específica, y se protege el acceso con autenticación y roles cuando es necesario.

Para obtener cursos, hay varias rutas. GET /paginated devuelve los cursos en formato paginado, cargando una cantidad limitada por solicitud. GET /load-more permite una carga progresiva, útil para implementar desplazamiento infinito. GET /all obtiene todos los cursos sin restricciones de paginación. GET /user-courses filtra solo los cursos en los que el usuario está inscrito, asegurando que cada usuario vea solo su contenido.

Las rutas protegidas por authenticate('admin') están reservadas para administradores. Estas incluyen POST / para crear un curso, PUT /:id para actualizarlo y DELETE /:id para eliminarlo. Esto impide que usuarios sin permisos puedan modificar o eliminar cursos, asegurando el control total por parte de administradores.

La inscripción en cursos se maneja con POST /enroll/:id, que verifica que el usuario no esté ya inscrito antes de registrar su participación. Si el usuario quiere salir de un curso, DELETE /user-courses/:id\_user\_course permite eliminar la relación en la base de datos, asegurando que el usuario solo pueda darse de baja de cursos en los que realmente esté inscrito.

Cada una de estas rutas maneja los errores de manera específica. Si falta información, si un usuario intenta realizar una acción no permitida o si hay un fallo en la base de datos, se envían respuestas claras con códigos de error adecuados. Todo esto hace que la API sea segura, organizada y fácil de escalar.

```
const express = require('express');
const path = require('path');
const cors = require('cors');
const db = require('./config/db');
const errorHandler = require('./middleware/errorHandler'); // Middleware de errores agregado

const app = express();
process.env.JWT_SECRET = process.env.JWT_SECRET || 'secreto_super_seguro';
const PORT = process.env.PORT || 3000;

app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// Rutas
const authRoutes = require('./routes/authRoutes');
const courseRoutes = require('./routes/courseRoutes');

app.use('/auth', authRoutes);
app.use('/courses', courseRoutes); // Ruta de cursos

app.use(express.static(path.join(__dirname, '../public')));

app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, '../public/index.html'));
});

app.get('/login', (req, res) => {
  res.sendFile(path.join(__dirname, '../public/login.html'));
});

app.get('/register', (req, res) => {
  res.sendFile(path.join(__dirname, '../public/register.html'));
});

// Verificar conexión a MySQL
db.getConnection((err, connection) => {
  if (err) {
    console.error("Error al conectar con la base de datos:", err.message);
  } else {
    console.log("Conexión a la base de datos establecida correctamente");
    connection.release(); // Liberar conexión
  }
});

// Middleware para manejar rutas inexistentes (404)
app.use((req, res, next) => {
  const error = new Error("Ruta no encontrada");
  error.statusCode = 404;
  next(error); // Pasar error al middleware global
});
```

```
// Middleware para manejo de errores global
app.use(errorHandler); // Aquí se usa el middleware de errores

app.listen(PORT, () => {
  console.log(`Servidor corriendo en http://localhost:${PORT}`);
});

module.exports = app;
```

### /test/test.test.js

Este código está diseñado para probar la API de autenticación y gestión de cursos. Antes de ejecutar las pruebas, limpia la base de datos eliminando usuarios, cursos y relaciones previas. Luego, registra dos usuarios: uno con rol de administrador y otro con permisos normales. Ambos inician sesión y se guardan sus tokens para utilizarlos en las pruebas. También se crea un curso para validar la funcionalidad de inscripción y gestión.

Las pruebas verifican que un usuario sin privilegios de administrador pueda inscribirse en un curso y, si lo desea, darse de baja. Sin embargo, si intenta eliminar un curso, la API lo rechaza con un **403 - Acceso prohibido**. Esto garantiza que solo los administradores puedan gestionar los cursos, evitando modificaciones no autorizadas.

En cuanto a la seguridad, se evalúan los tokens JWT. Si un usuario intenta acceder sin autenticarse, la API responde con un **401 - No autorizado**. Si el token es inválido, devuelve un **498 - Token corrupto**. Si el token ha expirado, se envía el mensaje **Sesión expirada, inicia sesión nuevamente**. Todo esto impide accesos no permitidos y refuerza la protección de los datos.

Por último, se comprueba que la paginación de los cursos funcione correctamente y que la búsqueda devuelva los resultados adecuados. Al finalizar las pruebas, se eliminan los datos de prueba y se cierra la conexión con la base de datos. Esto garantiza que cada ejecución comience en un entorno limpio y sin residuos de pruebas anteriores.

```
const request = require('supertest');
const app = require('../src/app');
const db = require('../src/config/db');

const testUser = {
  username: 'testuser',
  password: 'TestPass123',
  role: 'admin',
};

let token;
let normalUserToken;
let courseid;
let userCourseid;

beforeAll(async () => {
  await db.promise().query('DELETE FROM user_courses');
```

```

    await db.promise().query('DELETE FROM user WHERE username IN (?, ?)', [testUser.username, 'user1']);
    await db.promise().query('DELETE FROM course WHERE title IN (?, ?)', ['Curso de Node.js', 'Curso de Vue.js']);

    await request(app).post('/auth/register').send(testUser);
    const loginRes = await request(app).post('/auth/login').send({
        username: testUser.username,
        password: testUser.password,
    });
    token = loginRes.body.token;

    await request(app).post('/auth/register').send({
        username: 'user1',
        password: 'UserPass123',
        role: 'normal',
    });
    const normalLoginRes = await request(app).post('/auth/login').send({
        username: 'user1',
        password: 'UserPass123',
    });
    normalUserToken = normalLoginRes.body.token;

    const courseRes = await request(app)
        .post('/courses')
        .set('Authorization', `Bearer ${token}`)
        .send({
            title: 'Curso de Node.js',
            description: 'Aprende Node.js desde cero',
            category: 'Desarrollo Web',
            image_url: 'https://example.com/nodejs.png',
        });

    courseld = courseRes.body.courseld;
});

describe('✓ Pruebas de aceptación del usuario', () => {
    test('Un usuario normal debe poder inscribirse en un curso', async () => {
        const res = await request(app)
            .post(`/courses/enroll/${courseld}`)
            .set('Authorization', `Bearer ${normalUserToken}`);

        expect(res.status).toBe(201);
        expect(res.body.message).toBe('Inscripción exitosa');

        const [checkEnrollment] = await db.promise().query(
            'SELECT id_user_course FROM user_courses WHERE id_course = ?',
            [courseld]
        );
        expect(checkEnrollment.length).toBeGreaterThan(0);

        userCourseld = checkEnrollment[0].id_user_course;
    });
});

```

```

});
```

```

test('Debe permitir a un usuario salir de un curso', async () => {
  const res = await request(app)
    .delete(`/courses/user-courses/${userCourseId}`)
    .set('Authorization', `Bearer ${normalUserToken}`);

  expect(res.status).toBe(200);
  expect(res.body.message).toBe('Has salido del curso correctamente');

  const [checkEnrollment] = await db.promise().query(
    'SELECT * FROM user_courses WHERE id_user_course = ?',
    [userCourseId]
  );
  expect(checkEnrollment.length).toBe(0);
});
```

```

test('Debe rechazar eliminación de cursos por un usuario normal', async () => {
  const res = await request(app)
    .delete(`/courses/${courseId}`)
    .set('Authorization', `Bearer ${normalUserToken}`);

  expect(res.status).toBe(403);
});
```

```

describe('🔒 Autenticación de usuarios', () => {
  test('Debe permitir iniciar sesión con credenciales correctas', async () => {
    const res = await request(app).post('/auth/login').send({
      username: testUser.username,
      password: testUser.password,
    });
    expect(res.status).toBe(200);
    expect(res.body.token).toBeDefined();
  });

  test('Debe rechazar credenciales incorrectas', async () => {
    const res = await request(app).post('/auth/login').send({
      username: testUser.username,
      password: 'wrongpassword',
    });
    expect(res.status).toBe(400);
  });
});
```

```

describe('🛡️ Gestión de permisos según rol', () => {
  test('Debe permitir acceso a admin para crear cursos', async () => {
    const res = await request(app)
      .post('/courses')
      .set('Authorization', `Bearer ${token}`)
      .send({});
```

```
title: 'Curso de Vue.js',
description: 'Aprende Vue.js',
category: 'Frontend',
image_url: 'https://example.com/vue.png',
});
expect(res.status).toBe(201);
});

test('Debe rechazar acceso a usuarios normales para crear cursos', async () => {
const res = await request(app)
.post('/courses')
.set('Authorization', `Bearer ${normalUserToken}`)
.send({
title: 'Curso de React.js',
description: 'Aprende React.js',
category: 'Frontend',
image_url: 'https://example.com/react.png',
});
expect(res.status).toBe(403);
});
});

describe('🔑 Pruebas de seguridad en tokens JWT', () => {
test('Debe rechazar acceso sin token', async () => {
const res = await request(app).get('/courses/paginated');
expect(res.status).toBe(401);
});

test('Debe rechazar token inválido', async () => {
const res = await request(app)
.get('/courses/paginated')
.set('Authorization', 'Bearer invalidtoken');
expect(res.status).toBe(498);
});

test('Debe aceptar un token válido', async () => {
const res = await request(app)
.get('/courses/paginated')
.set('Authorization', `Bearer ${token}`);
expect(res.status).toBe(200);
});

test('Debe manejar la expiración del token correctamente', async () => {
const expiredToken = require('jsonwebtoken').sign(
{ id_user: 1, role: 'admin', exp: Math.floor(Date.now() / 1000) - 10 },
process.env.JWT_SECRET
);

const res = await request(app)
.get('/courses/paginated')
.set('Authorization', `Bearer ${expiredToken}`);
});
```

```

expect(res.status).toBe(440);
expect(res.body.error).toBe('Sesión expirada, inicia sesión nuevamente');
});

});

describe('📊 Pruebas de usabilidad', () => {
  test('Debe cargar los cursos correctamente con paginación', async () => {
    const res = await request(app)
      .get('/courses/paginated?page=1')
      .set('Authorization', `Bearer ${token}`);
    expect(res.status).toBe(200);
    expect(res.body.courses).toBeDefined();
  });

  test('Debe permitir buscar cursos con filtros', async () => {
    const res = await request(app)
      .get('/courses/load-more?lastCourseId=0&limit=5')
      .set('Authorization', `Bearer ${token}`);
    expect(res.status).toBe(200);
    expect(Array.isArray(res.body.courses)).toBe(true);
  });
});

afterAll(async () => {
  await db.promise().query('DELETE FROM user_courses');
  await db.promise().query('DELETE FROM user WHERE username IN (?, ?)', [testUser.username, 'user1']);
  await db.promise().query('DELETE FROM course WHERE title IN (?, ?)', ['Curso de Node.js', 'Curso de Vue.js']);
  db.end();
});

```

```

pedroleon@Laptop-de-Pedro test % npx jest test.test.js
  ✓ Pruebas de aceptación del usuario
    ✓ Un usuario normal debe poder inscribirse en un curso (233 ms)
    ✓ Debe permitir a un usuario salir de un curso (157 ms)
    ✓ Debe mostrar una lista de cursos por un usuario normal (5 ms)
  ❌ Autenticación de usuarios
    ✕ Debe permitir iniciar sesión con credenciales correctas (174 ms)
    ✕ Debe rechazar credenciales incorrectas (149 ms)
  ❌ Gestión de permisos según rol
    ✕ Debe permitir acceso a admin para crear cursos (85 ms)
    ✕ Debe permitir acceso a administradores normales para crear cursos (2 ms)
  ✓ Pruebas de seguridad en tokens JWT
    ✕ Debe rechazar acceso sin token (2 ms)
    ✕ Debe rechazar token inválido (1 ms)
    ✕ Debe aceptar un token válido (152 ms)
    ✕ Debe manejar la expiración del token correctamente (6 ms)
  📊 Pruebas de rendimiento
    ✕ Debe cargar los cursos correctamente con paginación (153 ms)
    ✕ Debo permitir buscar cursos con filtros (79 ms)

Test Suites: 1 passed, 1 total
Tests:       13 passed, 13 total
Snapshots:  0 added, 0 updated, 0 removed
Time:        0:00:1.167 s, estimated 4 s
 Ran all test suites matching /test.test.js/i.
Jest did not exit one second after the test run has completed.

!This usually means that there are asynchronous operations that weren't stopped in your tests. Consider running Jest with '--detectOpenHandles' to troubleshoot this issue.

```

# Brandon Alan Carrión Morales

Cargo: QA

Mi trabajo principal es testear el producto, asegurándome que cumpliera los estándares de calidad. En esas pruebas se encuentran pruebas de seguridad, de diseño, validación y también de documentar todo en una matriz y hacerle saber al equipo los problemas que se iban presentando.

Link de Matriz vs Pruebas: [QA-Caso De Prueba.xlsx](#)

## Requerimientos

Se debía crear una matriz de pruebas en la cual se documentaba los resultados de las pruebas que realizaba.

The screenshot shows a Jira interface for creating a test case. The title bar says "Creación de Matriz de Req vs Pruebas". The main area has sections for "Descripción" and "Actividad". The "Actividad" section includes tabs for "Todo", "Comentarios" (which is selected), "Historial", and "Registro de actividad". Below these tabs is a text input field with placeholder text "Añadir un comentario...". There are also several small icons and buttons. On the right side, there is a detailed view panel titled "Detalles" which contains information such as "Persona asignada: Brandon Alan Carrión Morales", "Approvers: Frida Sarahi Garza Gálvez", "Actual start: 21 feb 2025, 7:00", and "Principal: KAN-11 QA". At the bottom of the screen, a message from "Brandon Alan Carrión Morales" is visible.

Al tener issues se debían reportar en jira.

The screenshot shows a Jira interface for creating an issue. The title bar says "Creación de Issues". The main area has sections for "Descripción" and "Incidencias secundarias". The "Descripción" section contains a note about CI/CD assignment. The "Incidencias secundarias" section lists several issues with status "FINALIZADA": "KAN-91 Error ficheros", "KAN-125 Error API despliegue en la nube", "KAN-124 Error vista admin -02", and "KAN-123 Error vista usuario estandar". On the right side, there is a detailed view panel titled "Detalles" which contains information such as "Persona asignada: Brandon Alan Carrión Morales", "Approvers: Frida Sarahi Garza Gálvez", "Actual start: 18 feb 2025, 10:00", and "Principal: KAN-11 QA".

Testear el producto con el fin de encontrar problemas.

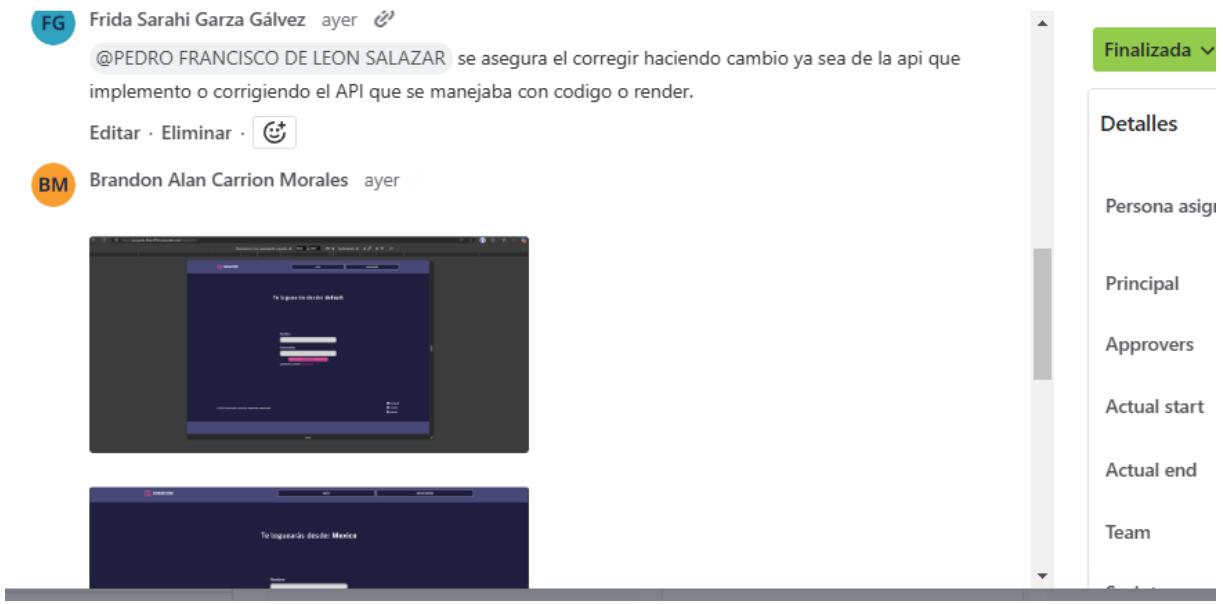
The screenshot shows a Jira interface for "Realización de tester solución". The main area has sections for "Descripción" and "Incidencias secundarias". The "Descripción" section contains a note about security, usability, and acceptance checks. The "Incidencias secundarias" section lists one issue: "KAN-88 Comprobar registro de nuevo usuario" with status "FINALIZADA". On the right side, there is a detailed view panel titled "Detalles" which contains information such as "Persona asignada: Brandon Alan Carrión Morales", "Approvers: Frida Sarahi Garza Gálvez", "Actual start: 21 feb 2025, 7:00", and "Principal: KAN-11 QA".

## Realización

Se debía crear una matriz de pruebas en las cuales se debían documentar los resultados de las pruebas ya sea negativas o positivas, para este requerimiento me encargue de separar las pruebas las positivas cumpliendo lo que se debía, como por ejemplo que los botones funcionaran, que se pudieran registrar etc. En los negativos me encargue de hacer pruebas que intentaran hacer fallar el sistema, como usar VPN para validar el api, llenar todos los campos para ver si presentaban error etc.

Al tener Issues se debían reportar en jira, dando una descripción del problema y cargando el nombre de la prueba, resultados esperado, actuales y subiendo pruebas, con el fin de que pudieran reconocer y replicar el error, de ahí mi lead se encargaría de asignar el error a la persona correspondiente, así yo puedo validar que este corregido en el siguiente Pull Request que se encargue de corregirlo.

Un ejemplo de Issue que trabajé fue con la API, ya que en local no se había problema, pero al utilizarla en la nube existía problema debido a que se utilizaba SSL en https y nos impide la comunicación sin una key. Gracias a esto mi equipo de trabajo pudo reconocer el error y solucionarlo y así poder entregar un producto con mejor calidad.



FG Frida Sarahi Garza Gálvez ayer

@PEDRO FRANCISCO DE LEON SALAZAR se asegura el corregir haciendo cambio ya sea de la api que implemento o corrigiendo el API que se manejaba con código o render.

Editar · Eliminar · ⚡

BM Brandon Alan Carrion Morales ayer

Finalizada

Detalles

Persona asignada

Principal

Approvers

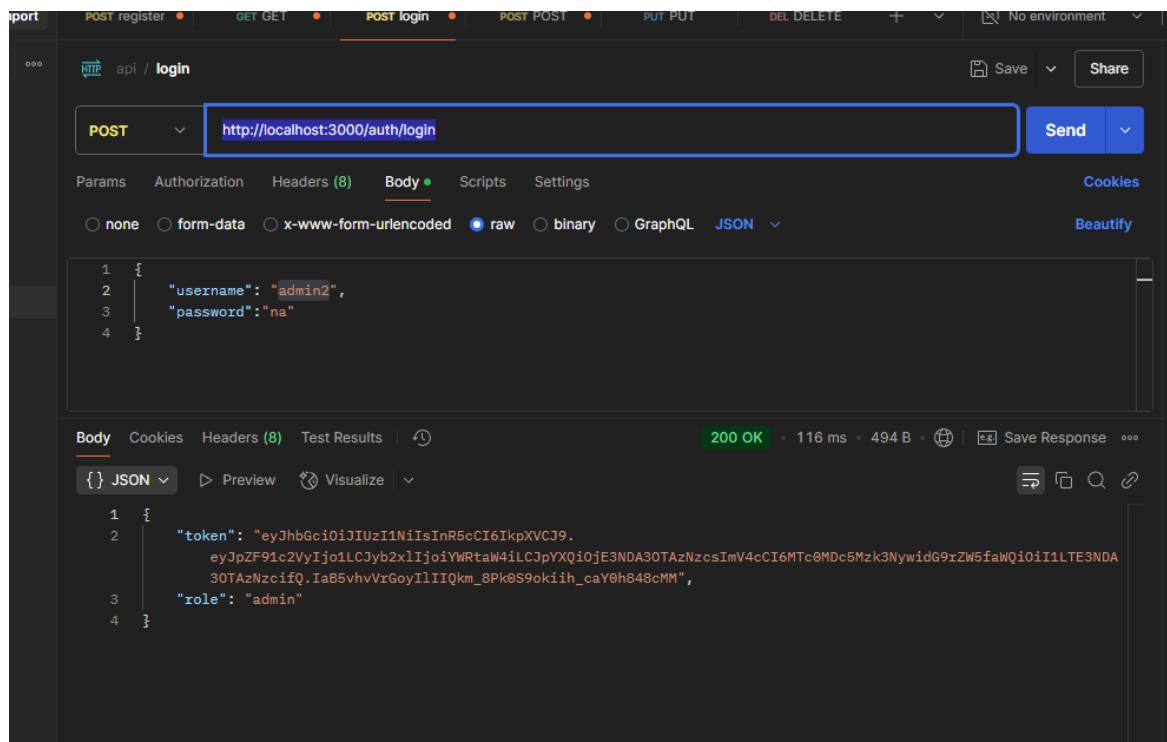
Actual start

Actual end

Team

También se dejaban fotografías para mostrar el error y que sea reconocido más fácilmente, de ahí se le encargaba a la persona responsable, en este caso era del Dev y él se encargó de implementar otro API que manejara https.

En una parte de mis pruebas me encargué de probar el jwt, usando Postman para ello utilicé la ruta de login: <http://localhost:3000/auth/login>



POST register ● GET GET ● POST login ● POST POST ● PUT PUT ● DEL DELETE + No environment

HTTP api / login

POST http://localhost:3000/auth/login Send

Params Authorization Headers (8) Body Scripts Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "username": "admin2",
3   "password": "na"
4 }
```

Body Cookies Headers (8) Test Results 200 OK 116 ms 494 B Save Response

{ } JSON Preview Visualize

```
1 {
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZFR1c2VyIjo1LCJyb2xlIjoiYWRtaW4iLCJpYXQiOjE3NDA3OTAzNzcsImV4cCI6MTc0MDc5Mzk3NywidG9rZW5faWQiOjI1LTE3NDA3OTazNzcihfQ.IaB5vhvRgoyIIIIQkm_8Pk0S9okihh_caY0h848cMM",
3   "role": "admin"
4 }
```

En esta captura se imprime el token y el rol, dado que yo creé ese usuario como administrador se imprime su rol.

My Workspace

api / GET

GET http://localhost:3000/courses/all

Headers (7)

Key	Value	Description
<input checked="" type="checkbox"/> Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ...	
Key	Value	Description

Body Cookies Headers (8) Test Results

```

29
  "id_course": 12,
  "title": "Cuzco Python",
  "description": "Curso de Python para principiantes",
  "category": "Programacion",
  "image_url": "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTwah68EQxlnu0Fag1TRvQAxFFBmxgjya8XLd&s"
30
31
32
33
34
35
36
  
```

200 OK • 59 ms • 1.33 KB • Save Response

Validación de que el token funcionara, se otorgo el token generado y se obtuvieron todos los cursos con su información. El token tambien funciona para la web, tome la url de render y agrege la ruta que obtiene todos los cursos

POST register • GET GET • POST login • POST POST • PUT PUT • DEL DELETE • + • No environment

api / GET

GET https://proyecto-final-07c2.onrender.com/courses/all

Headers (7)

Key	Value	Description
<input checked="" type="checkbox"/> Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ...	
Key	Value	Description

Body Cookies Headers (15) Test Results

```

4
  "id_course": 2,
  "title": "js",
  "description": "js\n",
  "category": "js",
  "image_url": "https://www.cursoenvideo.com/wp-content/uploads/bb-plugin/cache/javascript-circle-dc92b56e539139ec2bf42ebf8393803c-5d48cb37edbef.jpg"
5
6
7
8
9
10
11
12
13
  
```

200 OK • 213 ms • 1.01 KB • Save Response

Aquí genere otro token y valide que se requiriera para poder visualizar los cursos

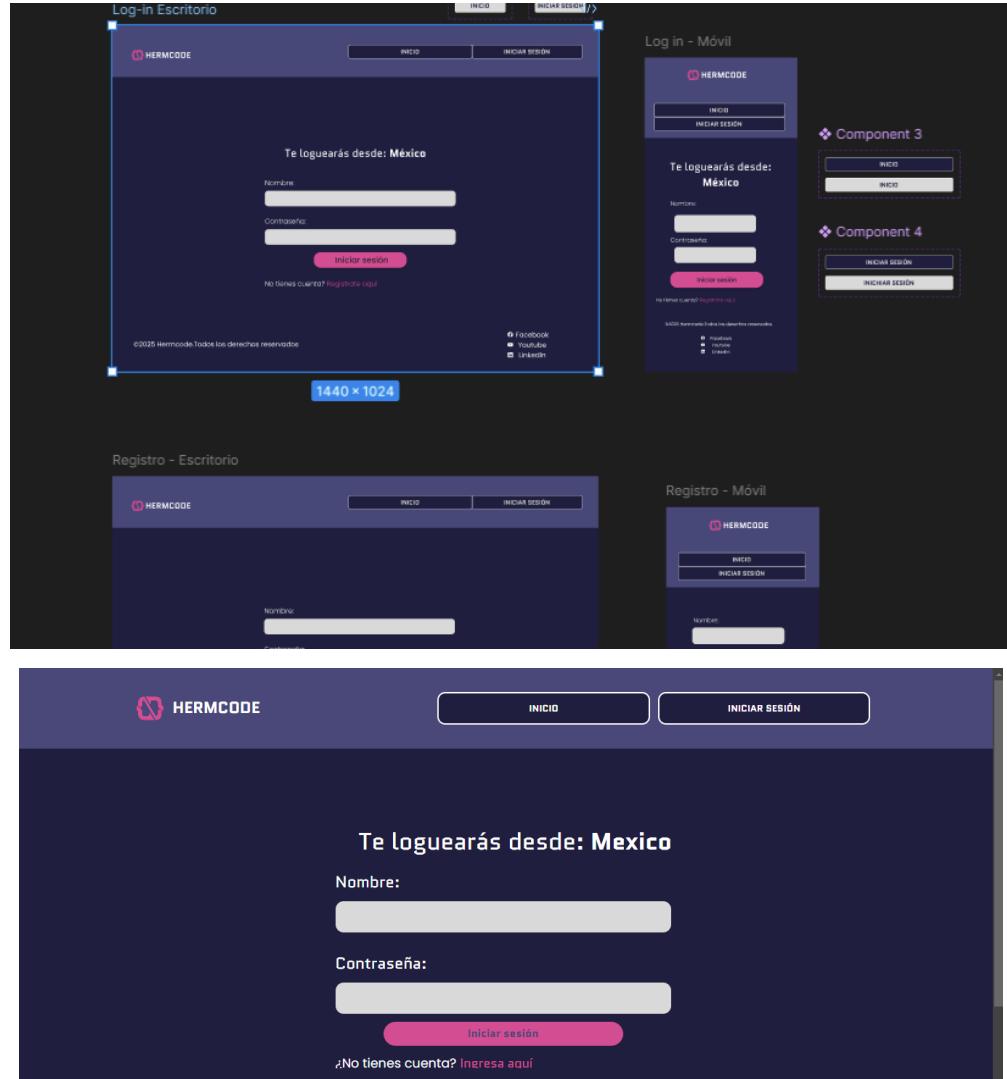
Al desactivar el token vemos que no tengo acceso.

El archivo con los endpoint se subirá en GitHub, en la carpeta de documentación.

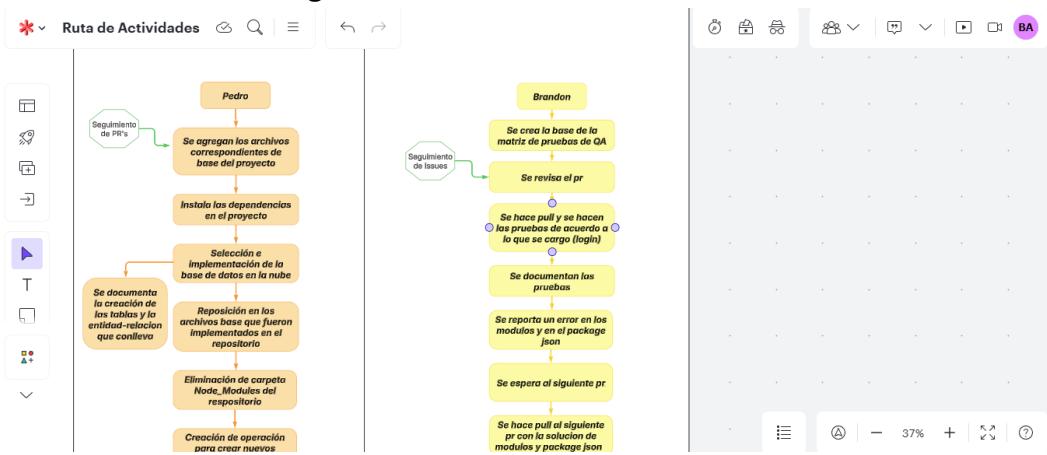
Hablando un poco mas de los test que se debian realizar para el backend, estos fueron algunos en los que debía asegurarme que funcionaran correctamente, pero tambien me tome la tarea de intentar hacerlos fallar, tanto que generara error y asegurarme que no se pudiera guardar, así como que se mostrara correctamente, estas fueron algunas imágenes de los errores.

Este error se presentó en las primeras etapas de desarrollo y consistía en que al sobreponerse, el contenedor daba más espacio del que debía y no se podían usar las funciones. Los errores que más se presentaron fueron problema con responsive, debido a que la mayoría de operaciones que se trabajaron con back end funcionaban correctamente, pero no se mostraban debido a un error en la forma de acomodar los elementos.

También me encargue de asegurarme que se siguiera el diseño de acuerdo a figma y así no tener un diseño mal implementado y tener mejor calidad en nuestro trabajo.



Muchas veces me encontré con errores que no se seguía el diseño, pero gracias a que pude reportar estas incidencias se lograron solucionar.



Para finalizar tuve que crear mi diagrama de actividades, aquí trate de no alargarlo mucho, pero siendo conciso de lo que trabajé.

Finalmente, la elaboración de mi reporte y carga de archivos a él repositorio

Adjunto mis archivos en caso de que no aparezcan en el repositorio.

- POSTMAN: [https://utmedu-my.sharepoint.com/:u/g/personal/al03108850\\_tecmilenio\\_mx/Eduj7qOra0dGoPcm-SoPYTgBbTjcSNlyuFSKc7nkG-DX7A?e=kjSHdy](https://utmedu-my.sharepoint.com/:u/g/personal/al03108850_tecmilenio_mx/Eduj7qOra0dGoPcm-SoPYTgBbTjcSNlyuFSKc7nkG-DX7A?e=kjSHdy)
- Diagrama caso de prueba: [https://utmedu-my.sharepoint.com/:x/g/personal/al03108850\\_tecmilenio\\_mx/ESeNyyatmGhPhg-iBZrTTVUBTLVJ12W4ykikwjEtL6plA?e=2tGAGK](https://utmedu-my.sharepoint.com/:x/g/personal/al03108850_tecmilenio_mx/ESeNyyatmGhPhg-iBZrTTVUBTLVJ12W4ykikwjEtL6plA?e=2tGAGK)

Link del repositorio

<https://github.com/Yugidar/Proyecto-Final.git>

Link del video

[Video muestrario de funcionalidad de página](#)

Link de los requerimientos

[Link de requerimientos de Jira](#)

## Conclusión

Creemos que este proyecto estuvo muy cargado pero a la vez fue de mucho aprendizaje respecto a nuestra organización como equipo, desde la organización en los tiempos, las actividades, los roles y nuestra forma de comunicación fueron un punto clave para poder lograr la finalización de este proyecto que abarcó dos semanas; puso a prueba nuestra dedicación y perseverancia, respecto a la búsqueda de poder aprender y conocer más allá de lo que se nos ha podido enseñar en clase en tan poco tiempo, creemos que nuestra búsqueda constante de resoluciones nos han llegado a formar un equipo con la suficiente confianza en buscar apoyo mutuo. Ese trabajo nos ha enseñado lo que lleva la vida laboral, y que estamos en la búsqueda de aprender y crecer para poder llegar a una buena organización para este tipo de trabajos.