

Proyecto Fin de Ciclo



Alumno: Daniel Verano Moreno
Ciclo Formativo: DAW
Curso: 2022 - 2023
IES Alixar

Índice

Introducción.....	2
Características de los usuarios.....	3
Análisis / comparativa con las alternativas del mercado.....	4
Stack Tecnológico.....	5
Requisitos funcionales.....	6
Diseño de la base de datos.....	7
Entidades de la base de datos.....	8
Prototipado de la aplicación Web.....	10
Servicios API REST.....	14
Manual de despliegue.....	16
Conclusiones del proyecto.....	20

Introducción

La idea del proyecto sería crear una tienda online en la cual los usuarios puedan ver y navegar todos los productos que ofrece la tienda. La tienda estará destinada a ofrecer productos del sector tecnológico, como pueden ser teléfonos móviles, auriculares o cualquier otro dispositivo.

La motivación principal por la que he decidido desarrollar este proyecto es porque siempre he tenido curiosidad de cómo funciona realmente una tienda online, de ver cuáles son todos los procesos que se ejecutan por debajo y en qué orden, es decir, lo que el cliente final no ve, y de cómo interactúan la parte del frontend y del backend.

Además, voy a afrontar este proyecto como un reto personal, ya que voy a usar una tecnología no vista en el curso, y me gusta seguir aprendiendo cosas nuevas y ampliando mis conocimientos, por lo que me apoyaré en varios cursos online relacionados con esta tecnología.

Características de los usuarios

En la aplicación habría dos roles: los clientes y el usuario administrador. El usuario administrador podrá ver una lista de todos los usuarios y productos, y además, tendrá permisos para insertar, actualizar y borrar en ambos modelos.

Como en cualquier tienda, los clientes podrían ver individualmente cada producto, en donde se podrían ver las características de cada producto como puede ser una descripción, valoración y precio. Esto implica que los clientes podrán dar su valoración a un producto que hayan comprado.

Los clientes podrán iniciar sesión y navegar hasta su perfil personal, donde podrán editar y actualizar sus diferentes datos si lo desean.

Por otro lado, los clientes podrán añadir los productos que deseen comprar a un carrito para posteriormente realizar el pago. En el carrito, se podrá ver un resumen de todos los productos que el cliente ha añadido antes de proceder a realizar el pago. Dicho pago se realizará mediante la pasarela de pago Stripe.

Para concluir, si el pago ha sido satisfactorio, los clientes podrán ver un resumen de la factura con todos sus productos comprados.

Análisis / comparativa con las alternativas del mercado

Teniendo en cuenta que la aplicación es una tienda enfocada a productos informáticos y electrónicos, podríamos decir que compite con las siguientes alternativas:

- **PC Componentes:** Es una tienda de informática y electrónica de consumo donde se puede encontrar prácticamente cualquier aparato electrónico. Aunque está especializada en el comercio a través de internet, PC Componentes también dispone de una tienda en Madrid y en Murcia. PC Componentes lleva operando en España y Portugal desde el 2005 y actualmente sigue siendo la tienda referente en informática y todo tipo de componentes y periféricos para ordenadores y portátiles.
- **Coolmod:** Es una tienda de tecnología física y online, con gran repercusión en España, que lleva numerosos años ofreciendo componentes de ordenador y periféricos de todo tipo. Su tienda física se encuentra en Castellón, en la Comunidad Valenciana, aunque también podemos acceder a ella a través de su página web. Coolmod lleva en el negocio desde el 2002 y, al centrarse en tener más proveedores que sus competidores, es considerada como otra gran alternativa.

Si comparamos la aplicación web con las alternativas mencionadas anteriormente, resulta prácticamente imposible competir con ellas, ya que llevan muchos años asentadas en el mercado y con una gran base de clientes fidelizados, además de disponer de numerosas características que no están presentes en la aplicación desarrollada. Para tener un mínimo de posibilidades de competir, haría falta una gran inversión inicial y disponer de un grupo de desarrolladores en lugar de una sola persona.

Stack Tecnológico

Las tecnologías que se usarán para desarrollar este proyecto son las siguientes:

- **Node JS y Express:** Para la parte del servidor (Backend), usaré Node.js, un entorno que trabaja en tiempo de ejecución, de código abierto y multiplataforma que permite a los desarrolladores crear todo tipo de aplicaciones y servicios utilizando el lenguaje Javascript, sin necesidad de tener un navegador web para que el código funcione.
Además, también usaré Express, uno de los framework más populares de Node que proporciona numerosos mecanismos y facilidades a la hora de crear servicios web, como el manejo de diferentes peticiones HTTP según el verbo usado en las rutas, o el procesamiento de esas peticiones incorporando funciones 'middleware' en la tubería del manejo de la petición.
- **React:** Para el lado del cliente (Frontend), he optado por usar React, una librería Javascript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página (SPA). El objetivo de React es ser sencillo, declarativo y fácil de combinar.
- **MongoDB** (Base de datos): MongoDB es un sistema de base de datos no relacional (NoSQL), orientado a documentos y de código abierto. MongoDB guarda estructuras de datos BSON (similar a JSON) con un esquema dinámico, lo que hace que la integración de los datos en muchas aplicaciones sea más fácil y rápida.

Por otra parte, algunos de los paquetes o funcionalidades a destacar que se usarán en la aplicación son los siguientes:

- **Bcrypt:** Permite encriptar las contraseñas de los usuarios antes de ser almacenadas en la base de datos.
- **JsonWebToken** (JWT): Permite poder identificar a los usuarios que se registren o inicien sesión desde el lado del servidor a la hora de acceder a los diferentes recursos que ofrece la aplicación mediante un token.
- **Multer:** Middleware que permite parsear las imágenes que se suban al servidor antes de ser subidas al proveedor elegido.
- **Cloudinary:** Empresa que ofrece servicios de gestión de imágenes y vídeos basados en la nube.
- **Stripe:** Es una plataforma de procesamiento de pagos por Internet. Mediante su API, los usuarios podrán realizar pagos en la aplicación y comprobar que todos los datos son correctos.
- **Axios:** Es un cliente HTTP basado en promesas para Node.js y navegadores, parecido a XHR y fetch pero con algunas características adicionales.
- **Styled Components:** Es una librería que te permite escribir código CSS en Javascript a la hora de desarrollar y customizar componentes en React.
- **React Admin:** Es una librería desarrollada en ES6, React y Material Design que permite implementar y customizar paneles de administrador en muchas aplicaciones.

Requisitos funcionales

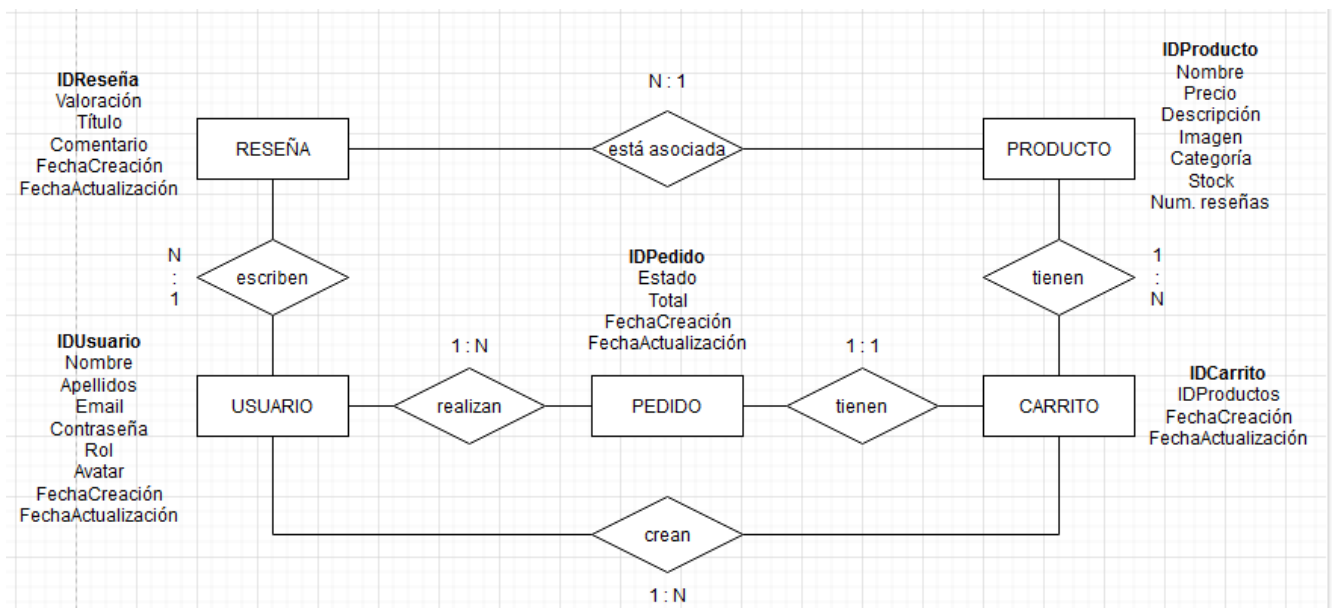
En base a las características de los usuarios definidas en el apartado anterior, los requisitos de la aplicación serían los siguientes:

- RF01: Tanto el usuario administrador como el usuario cliente deberán poder iniciar y cerrar sesión en la aplicación.
- RF02: Un usuario cliente podrá registrarse en el sistema si no tiene una cuenta.
- RF03: El usuario administrador podrá ver un listado de todos los usuarios registrados, así como de todos los productos, reseñas y pedidos que hagan los clientes.
- RF04: El usuario administrador podrá añadir, editar y eliminar usuarios, productos, reseñas y pedidos de la aplicación.
- RF05: El usuario cliente, al iniciar sesión, será redirigido a su dashboard donde podrá ver todos sus datos.
- RF06: El usuario cliente podrá modificar sus datos si lo desea, como su nombre, apellidos, email o contraseña.
- RF07: El usuario cliente podrá cambiar su foto de avatar por una que elija desde su equipo local.
- RF08: El usuario cliente podrá visualizar todos los productos que existen en la tienda.
- RF09: El usuario cliente podrá realizar una búsqueda concreta de productos mediante filtros, como búsqueda por texto, categoría o precio.
- RF10: El usuario cliente podrá visualizar un producto de manera individual, en donde se podrá ver la información de ese producto, como su descripción, valoración y precio.
- RF11: El usuario cliente podrá añadir un producto que desee comprar a un carrito de compra.
- RF12: El usuario cliente podrá realizar un pedido con todos los productos que haya añadido al carrito.
- RF13: El usuario cliente podrá realizar el pago del pedido realizado después de que el sistema compruebe que todos los datos son correctos.

Diseño de la base de datos

En este apartado, se indica el modelo entidad-relación de la base de datos que se usará en la aplicación, de acuerdo a las siguientes premisas:

- Un usuario puede realizar uno o varios pedidos.
- Un usuario puede escribir una o varias reseñas.
- Un usuario puede crear uno o varios carritos.
- Una reseña está asociada a un usuario y a un producto.
- Un pedido está asociado a un carrito y a un usuario.
- Un carrito puede contener uno o varios productos.



Entidades de la base de datos

En este apartado, se indican cuáles son todas las entidades que aparecen en el diseño de la base de datos, así como todas sus propiedades:

USUARIO (USER)

- **Id Usuario**
- Nombre
- Apellidos
- Email
- Contraseña
- Rol
- Avatar
- Fecha Creación
- Fecha Actualización

PRODUCTO (PRODUCT)

- **Id Producto**
- Nombre
- Precio
- Descripción
- Imagen
- Categoría
- Stock
- Número reseñas

PEDIDO (ORDER)

- **Id Pedido**
- Id Carrito
- Id Usuario
- Estado
- Total
- Fecha Creación
- Fecha Actualización

CARRITO (CART)

- **Id Carrito**
- Id Usuario
- Id Pedido
- Lista de Productos
- Fecha Creación
- Fecha Actualización

ITEM CARRITO (CART ITEM)

- Id Producto
- Nombre
- Cantidad
- Precio por Unidad
- Subtotal
- Imagen

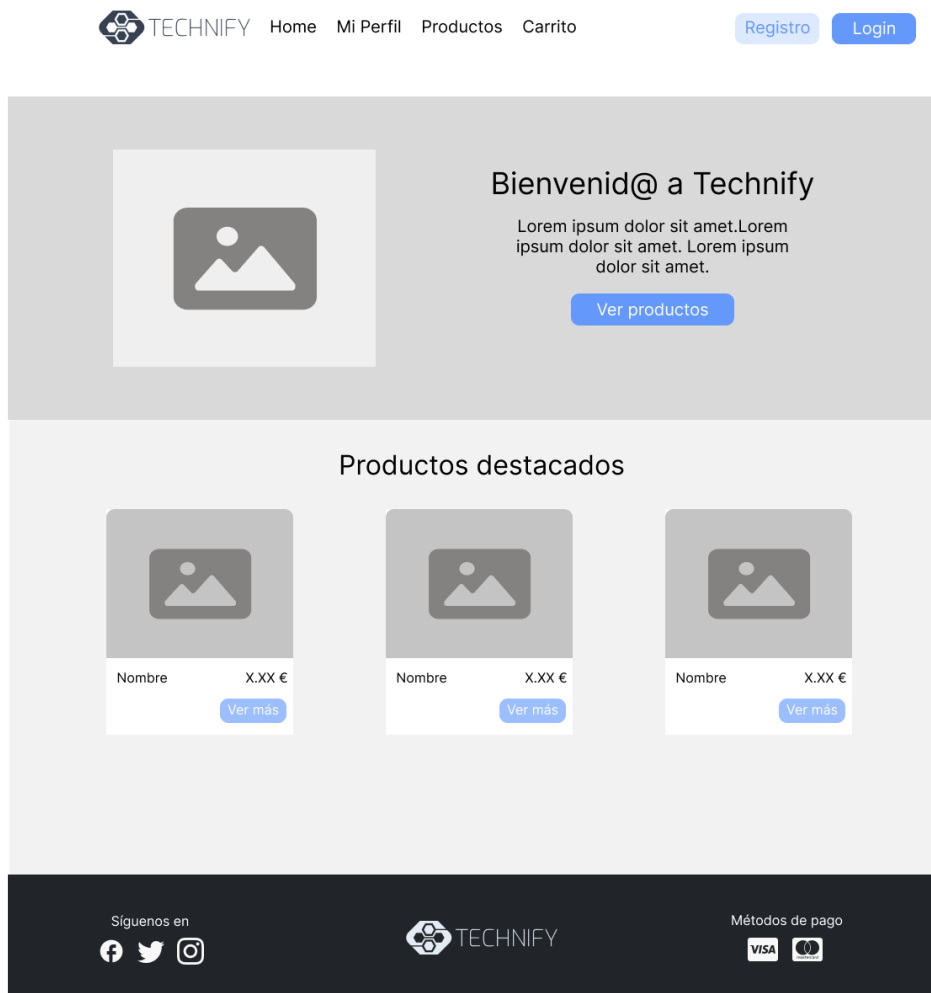
RESEÑA (REVIEW)

- **Id Reseña**
- Valoración (1–5)
- Título
- Comentario
- Id Usuario
- Id Producto
- Fecha Creación
- Fecha Actualización

Prototipado de la aplicación Web

En este apartado se muestra un prototipo de las diferentes vistas que tendría la aplicación una vez desarrollada. La herramienta usada para realizar el prototipado ha sido Figma.

Home



Login



TECHNIFY

[Home](#)

[Mi Perfil](#)

[Productos](#)

[Carrito](#)

[Registro](#)

[Login](#)

[Home / Login](#)

Login

Email

Contraseña

Login

Síguenos en



TECHNIFY

Métodos de pago



Register



TECHNIFY

[Home](#)

[Mi Perfil](#)

[Productos](#)

[Carrito](#)

[Registro](#)

[Login](#)

[Home / Register](#)

Registro

Nombre

Apellidos

Email

Contraseña

Registrarse

Síguenos en




TECHNIFY

Métodos de pago




Dashboard

TECHNIFY

[Home](#) [Mi Perfil](#) [Productos](#) [Carrito](#)

[Logout](#)

Home / Mi Perfil



Pepe
Pérez Pérez

Editar perfil

Nombre

Pepe

Apellidos

Pérez Pérez




Email


pepe@gmail.com

Contraseña



Actualizar mi perfil

Síguenos en




TECHNIFY

Métodos de pago



Productos

TECHNIFY

[Home](#) [Mi Perfil](#) [Productos](#) [Carrito](#)

[Registro](#) [Login](#)

Home / Productos

Filtrar por:


Texto

Categoría


Precio

Valoraciones


Borrar filtros




Nombre X.XX €
Ver más




Nombre X.XX €
Ver más




Nombre X.XX €
Ver más



Nombre X.XX €
Ver más







Nombre X.XX €
Ver más





Nombre X.XX €
Ver más

Síguenos en



TECHNIFY

Métodos de pago



Producto Individual

TECHNIFY

HomeMi PerfilProductosCarrito

RegistroLogin

Home / Productos / Título Producto

Volver atrás

Título Producto

Valoración: ★★★★★

Precio: XX.XX €

Lorem ipsum dolor sit amet consectetur adipiscing elit. lusto repudiandae temporibus autem unde aliquid quo, deserunt neque veniam dolores alias doloribus quidem tempora officia, magni officis earum, cupiditate illum repellendus pariatur fugiat. ipsam velit atque, odio laboriosam impedit et consequuntur eum ab explicabo modi dolore reprehenderit, enim quidem! Maxime, illo.

Disponibilidad

Empresa

Cantidad: - 1 +

Añadir al carrito

Síguenos en

TECHNIFY

Métodos de pago

VISA

Carrito

TECHNIFY

HomeMi PerfilProductosCarrito

Logout

Home / Carrito

Producto	Precio	Cantidad	Subtotal
Producto 1	XX.XX €	2	XX.XX €
Producto 2	XX.XX €	1	XX.XX €
Producto 3	XX.XX €	3	XX.XX €

Seguir comprando

Limpiar carrito

Subtotal : XX.XX €

Gastos de envío : XX.XX €

Total : XX.XX €

Realizar pedido

Síguenos en

TECHNIFY

Métodos de pago

VISA

Servicios API REST

A continuación, se van a mostrar todos las rutas definidas que ofrece la API REST para cada uno de los modelos de la aplicación:

USUARIO		
Ruta	Método	Descripción
/users	GET	Devuelve todos los usuarios
/users/:id	GET	Devuelve un usuario identificado por su ID
/updateUser	PATCH	Modifica uno o varios campos de un usuario identificado por su ID, que se obtiene por middleware
/updatePassword	PATCH	Actualiza la contraseña de un usuario por una nueva si la antigua es correcta
/users/:id	DELETE	Elimina un usuario de la BD identificado por su ID
/auth/register	POST	Registra a un nuevo usuario en la BD
/auth/login	POST	Inicia sesión en la aplicación si el usuario existe en la BD y la contraseña es correcta

PRODUCTO		
Ruta	Método	Descripción
/products	GET	Devuelve todos los productos
/products/:id	GET	Devuelve un producto identificado por su ID
/products	POST	Crea un nuevo producto en la BD
/products/:id	PATCH	Modifica uno o varios campos de un producto identificado por su ID
/products/:id	DELETE	Elimina un producto de la BD identificado por su ID

RESEÑA		
Ruta	Método	Descripción
/reviews	GET	Devuelve todas las reseñas
/reviews/:id	GET	Devuelve una reseña identificada por su ID
/reviews	POST	Crea una nueva reseña asociada a un usuario y un producto en la BD
/reviews/:id	PATCH	Modifica uno o varios campos de una reseña identificada por su ID
/reviews/:id	DELETE	Elimina una reseña de la BD identificada por su ID

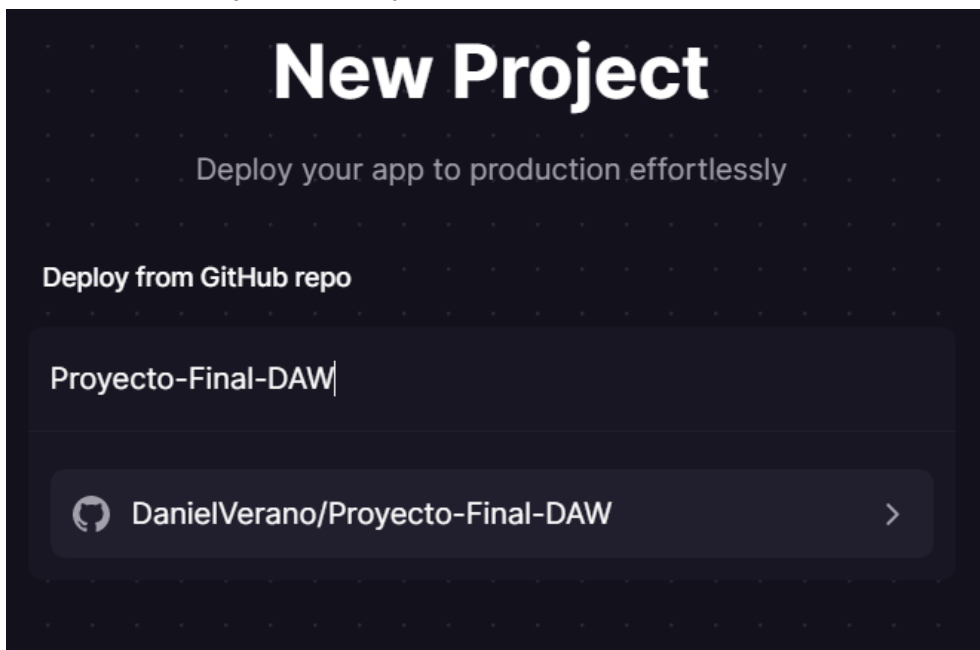
PEDIDO		
Ruta	Método	Descripción
/orders	GET	Devuelve todos los pedidos
/orders/:id	GET	Devuelve un pedido identificado por su ID
/orders	POST	Crea un nuevo pedido asociado a un usuario y a un carrito de compra en la BD
/orders/:id	PATCH	Modifica uno o varios campos de un pedido identificado por su ID
/orders/:id	DELETE	Elimina un pedido de la BD identificado por su ID
/createPaymentIntent	POST	Devuelve una clave que será utilizada para identificar al usuario antes de pagar con la API de Stripe si los datos proporcionados son correctos

Manual de despliegue

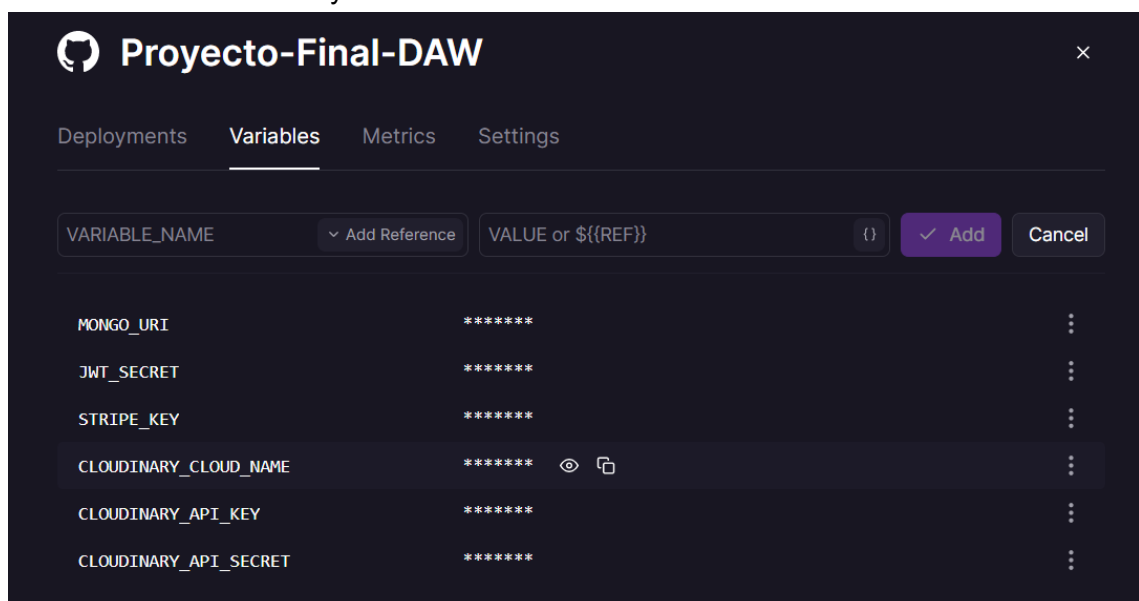
A continuación, se va a explicar cómo realizar el despliegue de la aplicación utilizando Github Pages para la parte del cliente y Railway para la parte del servidor. Primero empezaremos desplegando la API, ya que el frontend depende de ella:

Despliegue del servidor

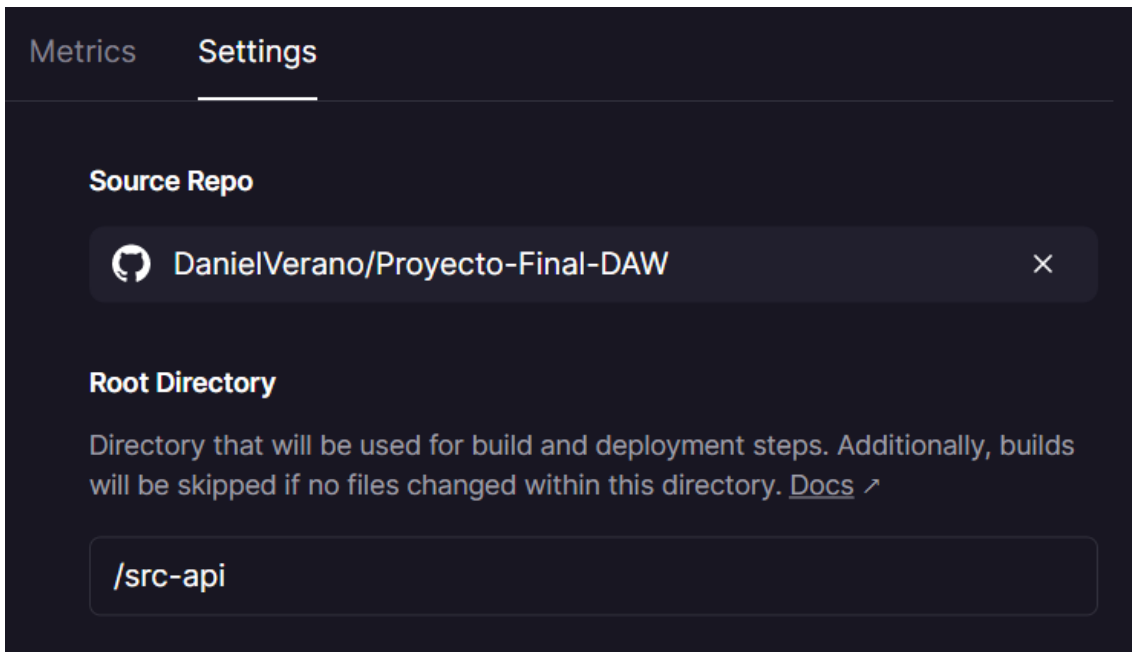
1. Iniciamos sesión en nuestra cuenta de Railway, y desde el dashboard nos vamos a:
Create a New Project > Deploy from Github Repo > Seleccionamos el repositorio utilizado



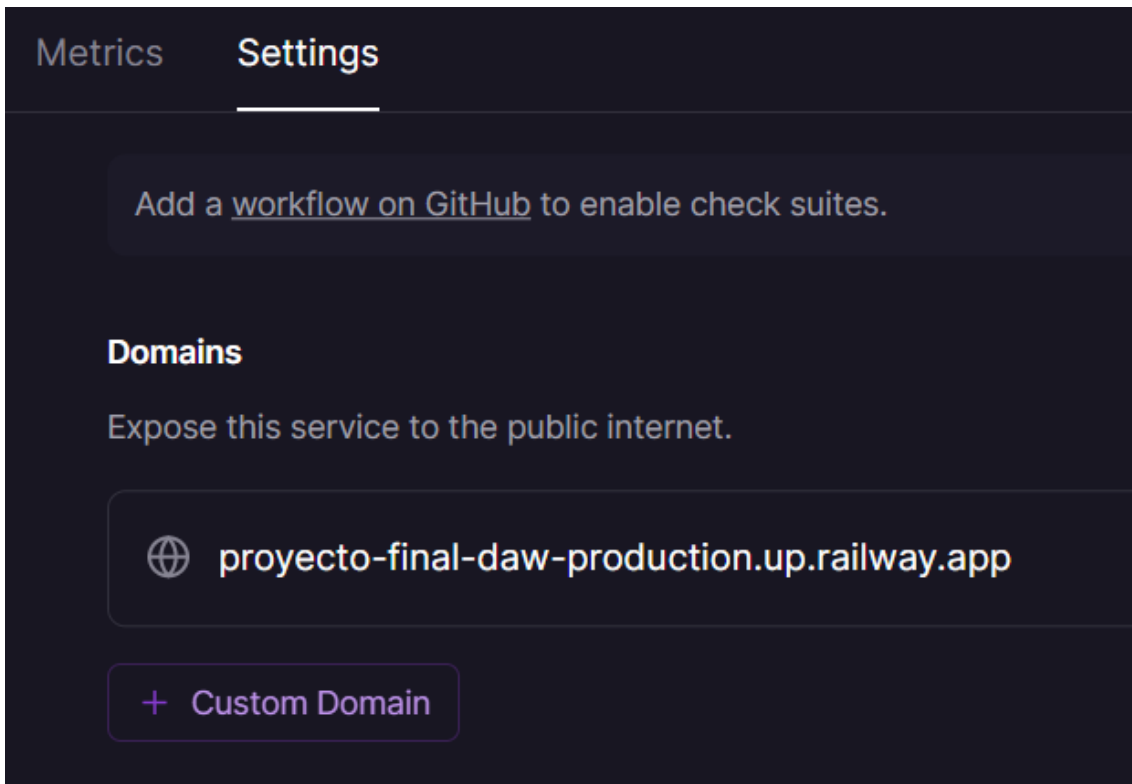
2. El siguiente paso es añadir las variables de entorno que usa la aplicación. En el siguiente menú, clicamos en Add variables y las añadimos.



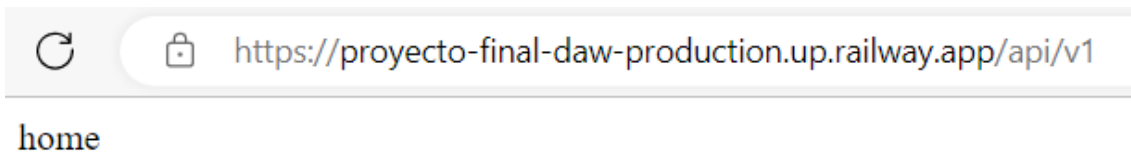
3. Para este repositorio, es necesario especificar la ruta donde se encuentra todo el código del servidor. Para ello, nos vamos a la pestaña Settings, y en Root Directory ponemos `/src-api`



4. Por último, sólo nos queda exponer el servicio a Internet. Desde la misma pestaña anterior (Settings), en la sección Environment, pulsamos en Generate New Domain, la cual nos generará un dominio para poder acceder a la API. Esta URL es la que utilizaremos también en el despliegue del frontend para indicar la dirección de la misma.



5. Para asegurar, accedemos a la URL anterior y realizamos una petición para comprobar que la API está funcionando correctamente:



Despliegue del Frontend

1. En primer lugar, para desplegar la aplicación necesitamos instalar el paquete `gh-pages`. Para ello, desde la raíz del proyecto *src-frontend*, abrimos un nuevo terminal y ejecutamos el siguiente comando: `npm install gh-pages --save-dev`

```
user@dvm-vbox:~/repos/pfc-prueba/Proyecto-Final-DAW/src-frontend$ npm install gh-pages --save-dev
```

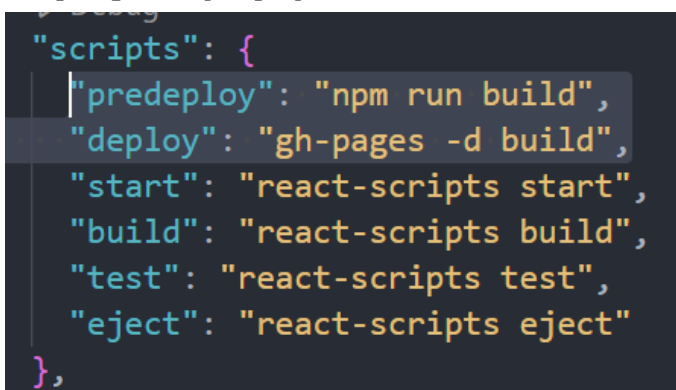
2. Una vez instalado el paquete, antes de realizar el despliegue es necesario realizar algunas modificaciones en el proyecto. La primera de ellas es, en el archivo *package.json*, añadir la propiedad *homepage*, que hace referencia a la URL donde estaría alojado la aplicación, con la siguiente nomenclatura:

```
"homepage": "https://<nombre_usuario>.github.io/<nombre_repo>"
```



3. La siguiente modificación es añadir los siguientes scripts al objeto *script*, también en el *package.json*:

```
"predeploy": "npm run build",
"deploy": "gh-pages -d build",
```



4. La última modificación a realizar es añadir la propiedad *basename* con el valor del nombre del repositorio en Github al router que estamos usando para navegar entre las distintas vistas de la aplicación. De esta manera le estamos indicando al router cuál es la ruta raíz de la aplicación una vez desplegada. Para ello, nos vamos al archivo App.js en src/App.js y añadimos la propiedad al router:

```
const App = () => {  
  return (  
    <BrowserRouter basename="/Proyecto-Final-DAW">  
      <div className="container">  
        <Navbar />  
      </div>  
    </BrowserRouter>  
  )  
}
```

5. Una vez realizadas las modificaciones, tenemos que añadir la URL de la API anterior en el archivo .env, para añadirla como variable de entorno. Para este proyecto se añadiría de la siguiente manera:

```
JS App.js M {} package.json M .env U X  
pfc-prueba > Proyecto-Final-DAW > src-frontend > .env  
1 REACT_APP_API_URL=https://proyecto-final-daw-production.up.railway.app
```

6. Ya podemos realizar el despliegue de la aplicación. Ejecutamos el siguiente comando desde la carpeta *src-frontend*. Cuando termine, se creará una nueva rama en el repositorio de Github llamada *gh-pages* con todos los archivos HTML, CSS Y JS de la aplicación.

npm run deploy

```
• user@dvm-vbox:~/repos/pfc-prueba/Proyecto-Final-DAW/src-frontend$ npm run deploy  
  > frontend@0.1.0 predeploy  
  > npm run build
```

7. Para finalizar, desde el repositorio de Github, nos vamos a Settings > Pages, y lo configuramos para que despliegue desde la rama *gh-pages* y desde la carpeta raíz:

Build and deployment

Source

Deploy from a branch ▾

Branch

Your GitHub Pages site is currently being built from the *gh-pages* branch. [Learn more.](#)

gh-pages ▾

/ (root) ▾

Save

Learn how to [add a Jekyll theme](#) to your site.

Conclusiones del proyecto

Realizar este proyecto me ha servido mucho para seguir aprendiendo conceptos de programación y buenas prácticas, además de aprender a desenvolverse en un entorno más realista, ya que así es como considero que se aprende realmente a programar, es decir, cuando nos enfrentamos a un problema que no sabemos inicialmente cómo resolver, no nos queda otra que investigar por nuestra cuenta, ya sea leyendo la documentación oficial o preguntando a otras personas o compañeros.

Por otro lado, he aprovechado el desarrollo de este proyecto para aprender una tecnología que no había usado hasta ahora, pero que llevaba tiempo con curiosidad por usarla, es el caso de React, utilizada para el desarrollo del lado del cliente. Con ayuda de la documentación oficial y viendo muchos vídeos relacionados, he aprendido bastante sobre ella, y creo que ha sido una decisión acertada la elección de usarla para el desarrollo.

En cuanto a las posibles mejoras de la aplicación, se podrían seguir implementando muchas más funcionalidades, pero, al fin y al cabo, teniendo en cuenta que disponemos de un plazo fijo para entregar el proyecto, es imposible abordarlas todas y tenemos que ceñirnos a los requisitos definidos en la fase inicial del proyecto. Algunas de las mejoras que podrían implementarse son:

- **Paginación:** A medida que los productos ofrecidos por la aplicación crezcan, sería conveniente añadir una funcionalidad de paginación en la vista de todos los productos, para que no carguen todos a la vez.
- **Mejor diseño:** El diseño de la página quizás sea demasiado simple y podría ser mejorable en ciertos aspectos, ya que se ha priorizado más la parte del desarrollo e implementación de las funcionalidades.