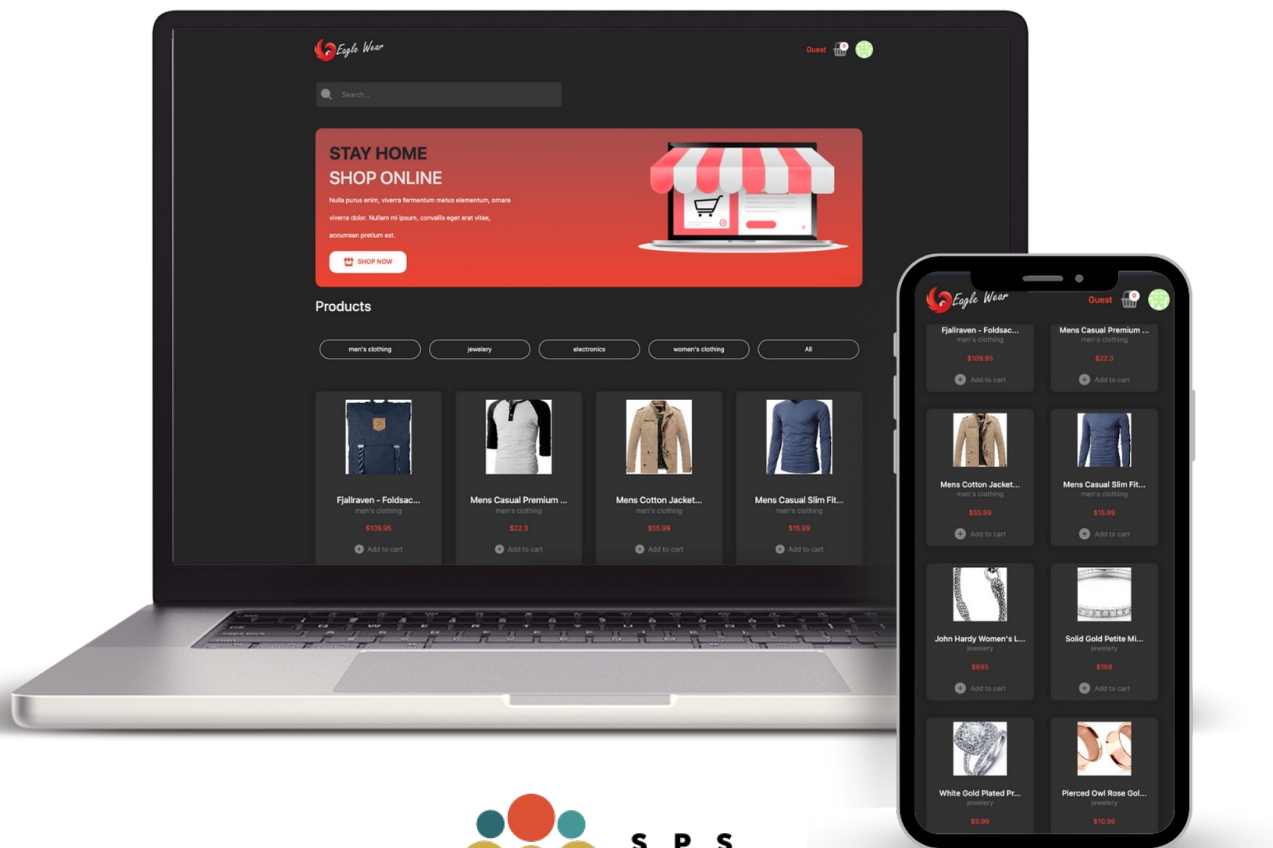


PRUEBA TECNICA

EDDIE ELORZA | DESARROLLADOR



DEPLOY | CODIGO

A continuación podrán acceder al código dando clic en las imagenes



PROBLEMA

La empresa comercializadora Eagle Wear busca obtener un mayor alcance en el mercado de la moda, por lo que dirección ha decidido desarrollar una aplicación web en la que el aspirante deberá realizar los siguientes requerimientos.

¿COMO LO RESUELVO?



Antes de ir directo al código me surgio la duda de
por
donde sería el major camino por empezar para
poder completar el reto, y tuve que hacer un
pequeño diagrama

SOLUCIÓN

Como primer paso fue abstraer cada uno de los puntos desde pensar la parte gráfica como lógica, por lo que tome una hoja y me puse a bocetear la estructura de la página, así mismo a revisar la API y las tecnologías con las que contaba.

PRIMEROS PASOS

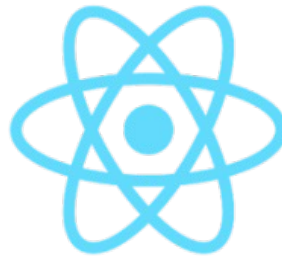
- Búsqueda de diferentes UI para tener una referencia, al momento de desarrollar el Frontend.
- Crear y estructurar archivos con React
- Dividir tareas
- Diseño de wireframes
- Desarrollo de diagrama de flujo

TECNOLOGÍAS Y HERRAMIENTAS

Herramientas y tecnologías con las que se dio solución a la prueba técnica



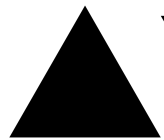
Firebase



React



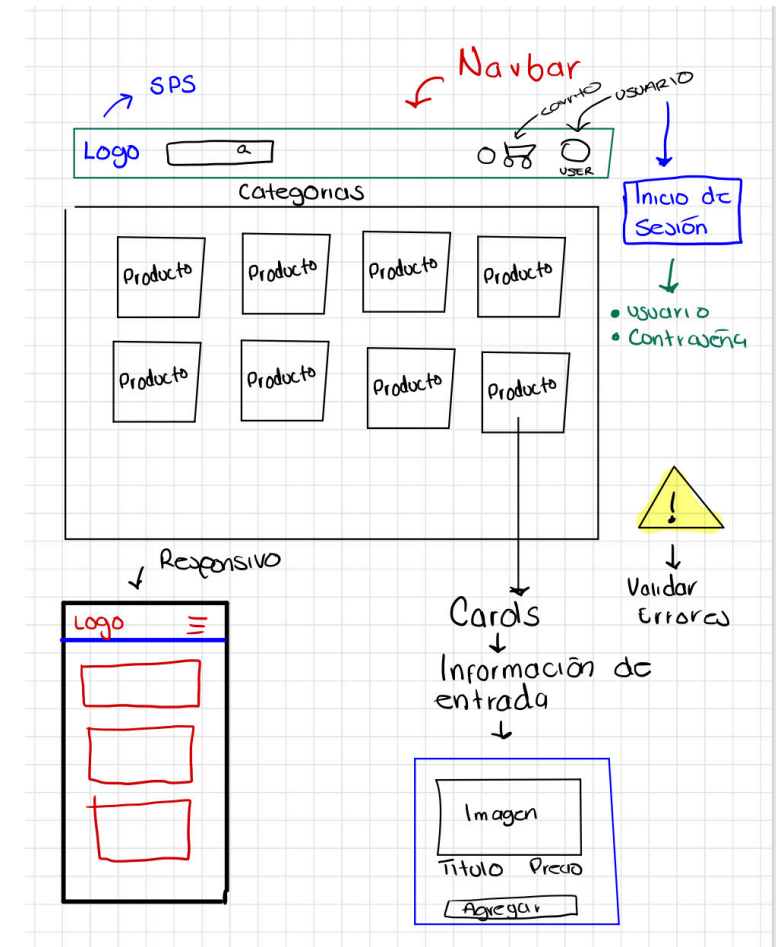
Bootstrap



Vercel



WIREFRAMES



CONSUMO DE APIS

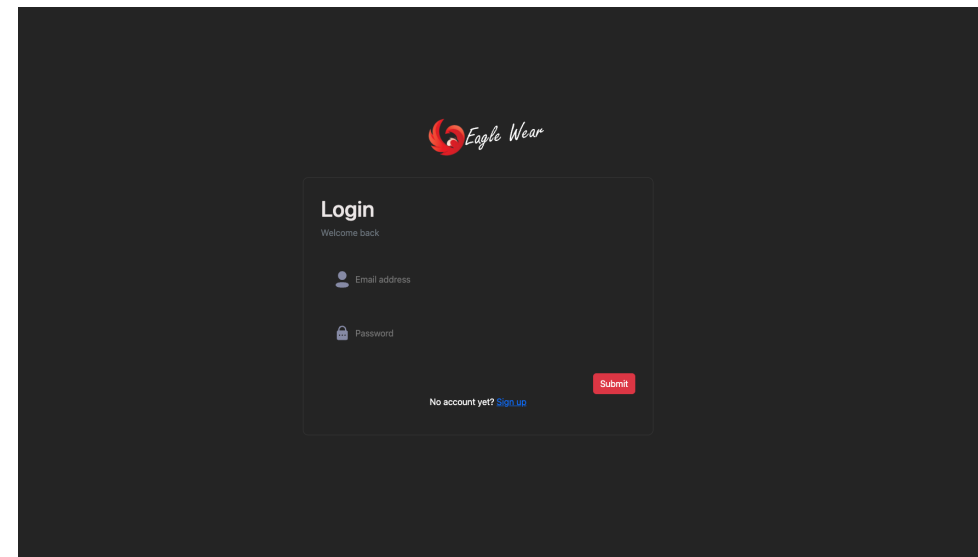
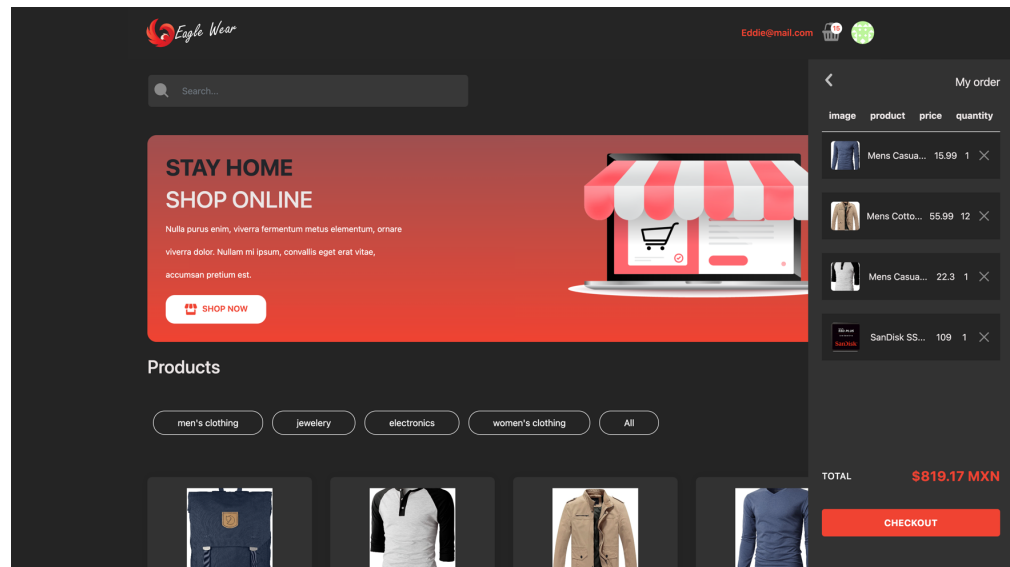
- Durante el periodo de abstracción de ideas me percate de que debería tener en cuenta una base de datos para iniciar sesión, una base de datos para guardar los productos de acuerdo al id del usuario ya logueado, y consumir el api de productos.

¿POR QUE DE ESTA MANERA?

- Al ver que el problema sugería que por cada usuario que ingrese debería poder guardar sus productos en el carrito.
- La mejor opción era que crear una base de datos con Firebase Authentication que me podría guardar mis usuarios y a la vez generarme un token que así mismo podría tener un login y un singup.
- Después de investigar, pude visualizar que necesitaba otro endpoint que me actualizará esa información de acuerdo a como se irían agregando los productos a mi orden, esos datos se guardarían en mi base de datos obteniendo el token del usuario, entonces cada que inicie sesión esos datos deben compararse con el token y obtener la información guardada.

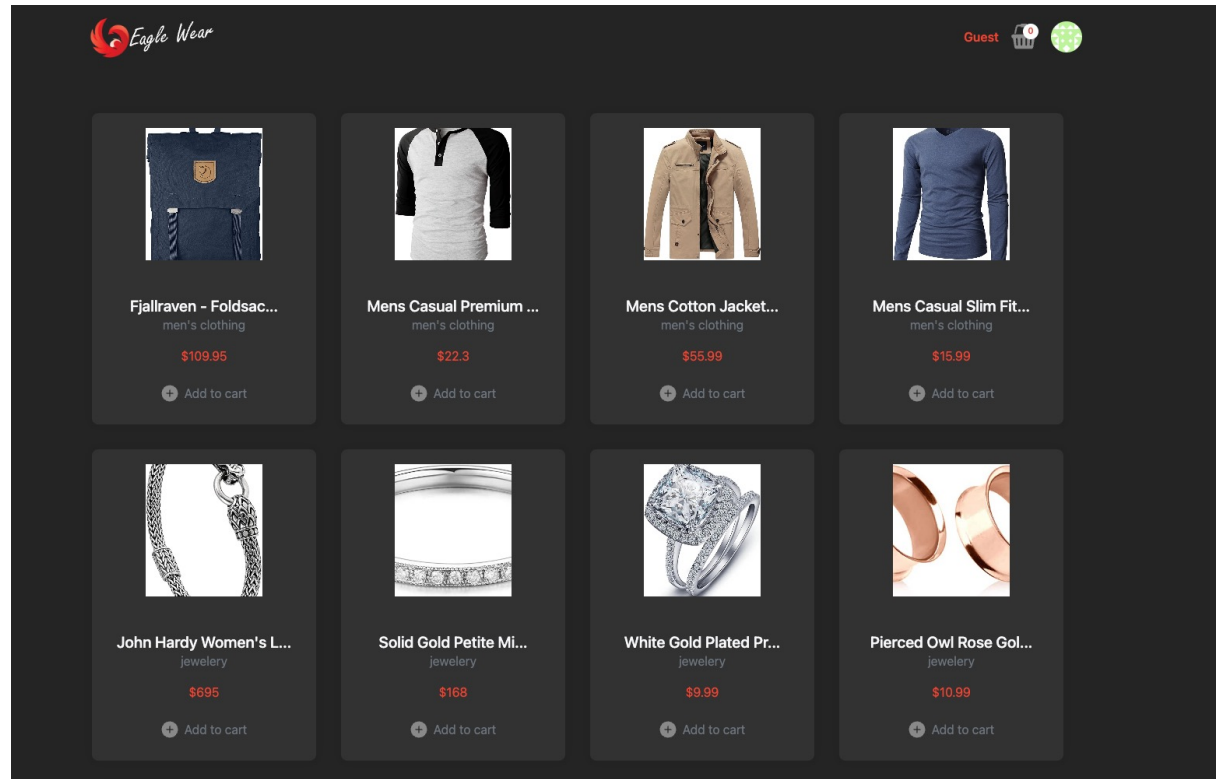
DESARROLLO DE INTERFACES

- Antes de desarrollar las funcionalidades lógicas se diseñaron y desarrollaron las interfaces con herramientas como bootstrap y scss, se incluyo y se modifico el logo propuesto, se desarrollo una interfaz de login funcional.



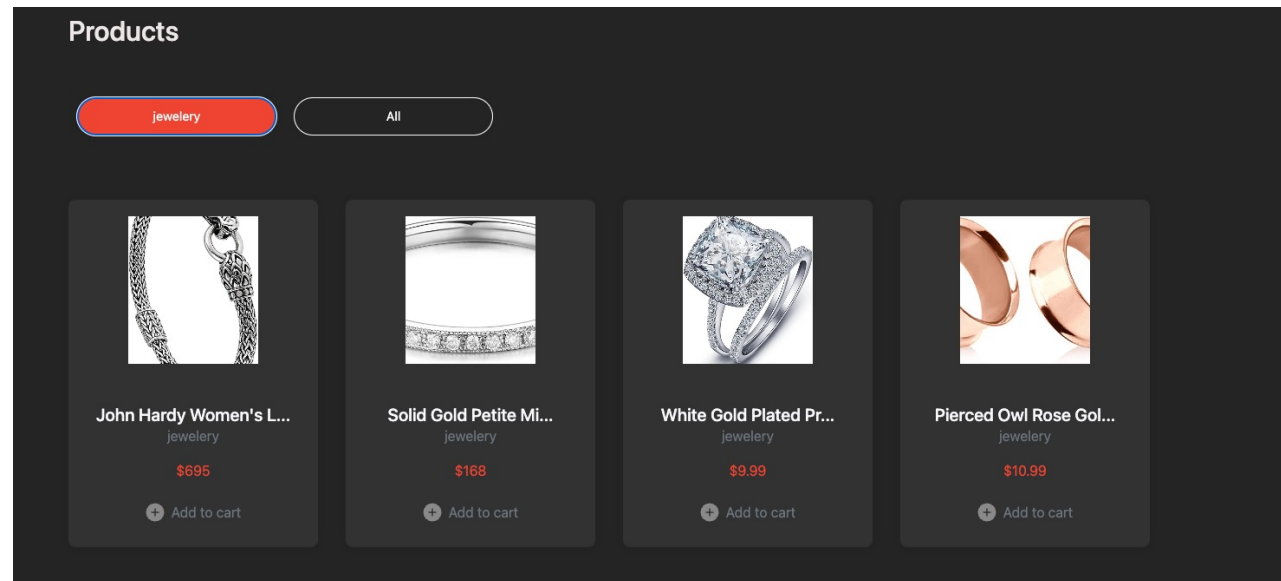
MOSTRAR CATÁLOGO

- Ya teniendo el endpoint de productos, se mapearon en la interfaz.



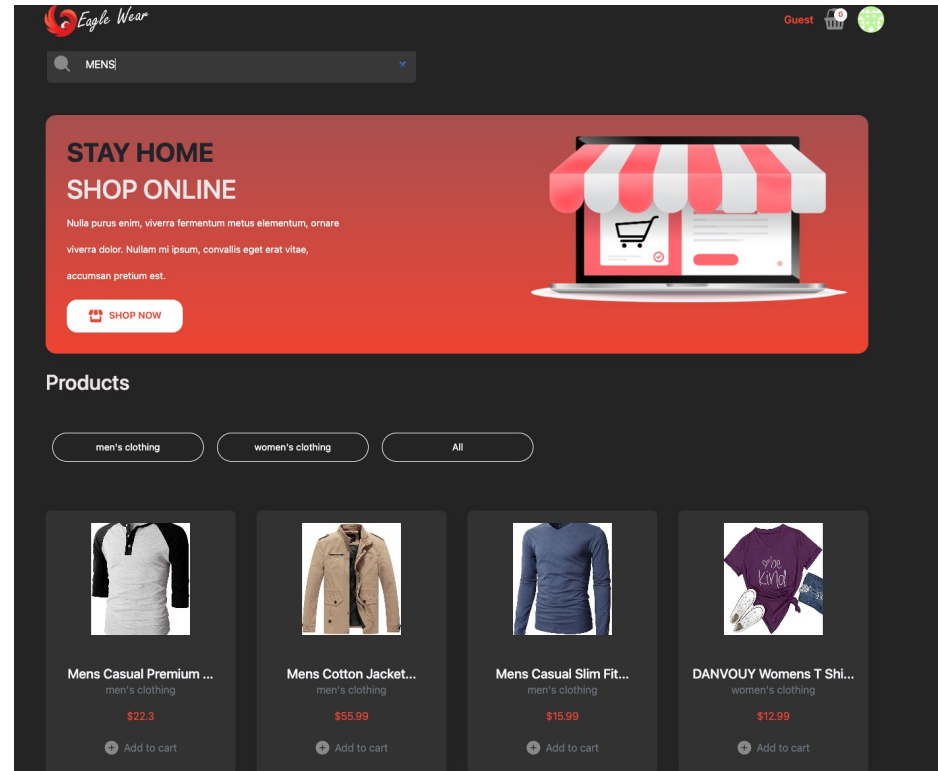
FILTRADO DE PRODUCTOS POR BOTÓN

- Primero agrupe las categorías de los productos y las guarde en un arreglo para después imprimirlas en una lista en la interfaz.
- Después genere un filtrado que al dar clic en una categoría me mostrará solo los productos que coincidieran con ese evento.



FILTRADO DE PRODUCTOS POR BÚSQUEDA

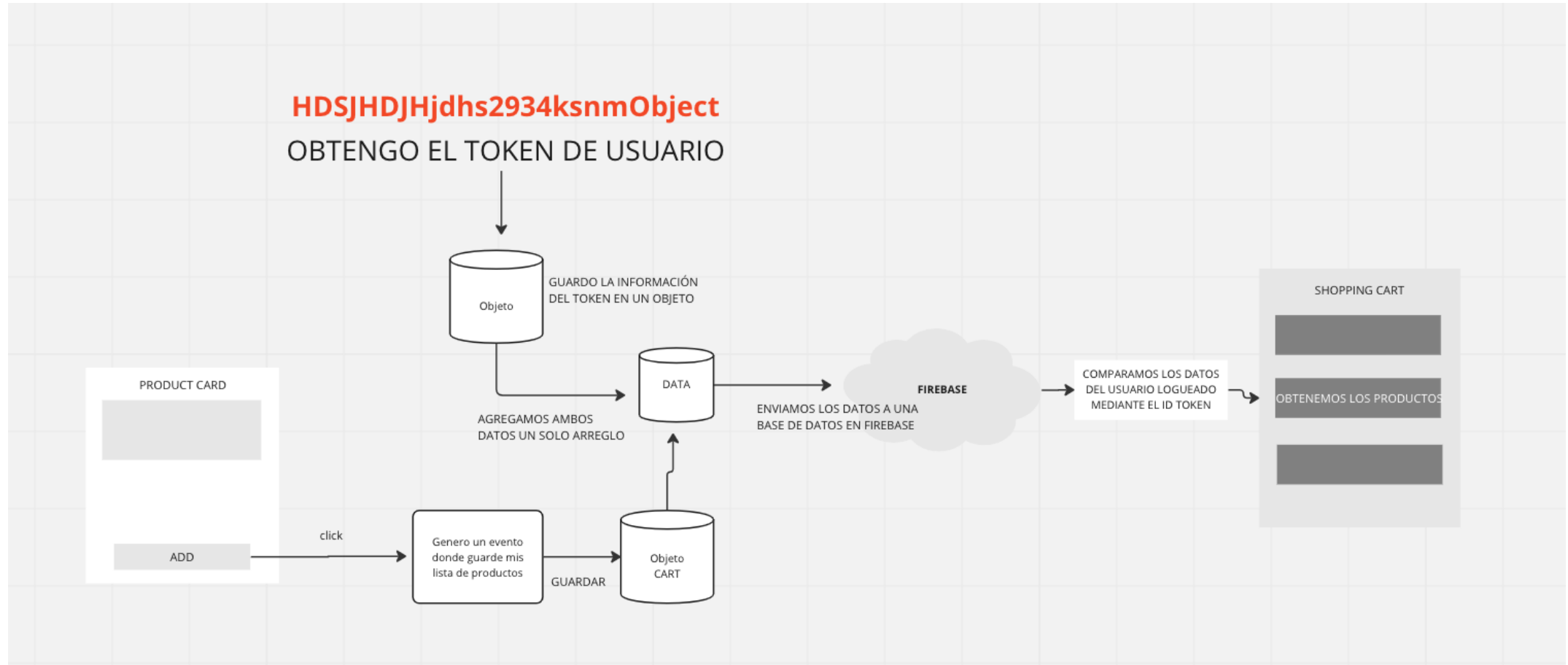
- Para poder generar este resultado la forma de resolver este fue obteniendo los resultado evento del input y filtrarlos por el titulo de los productos.



MANEJO DE CARRITO DE COMPRAS

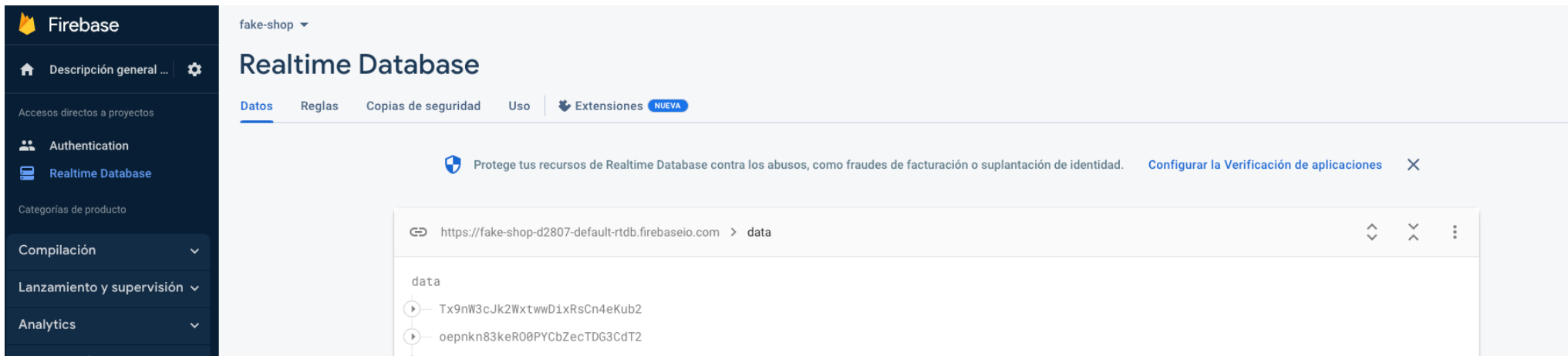
- Esta fue una de las partes mas complicadas de abstraer, ya que no solamente era agregar los productos, sino relacionarlos con cada usuario que inicie sesión y tenga guardado los productos en su carrito.
- La primera solución llamemosla versión 1, fue guardar los productos dentro de un objeto e irlos imprimiendo en el carrito, guardandose mediante el local storage, el tema a discutir fue, este objeto guardado en local no se va a relacionar con un solo usuario.

ABSTRACCIÓN



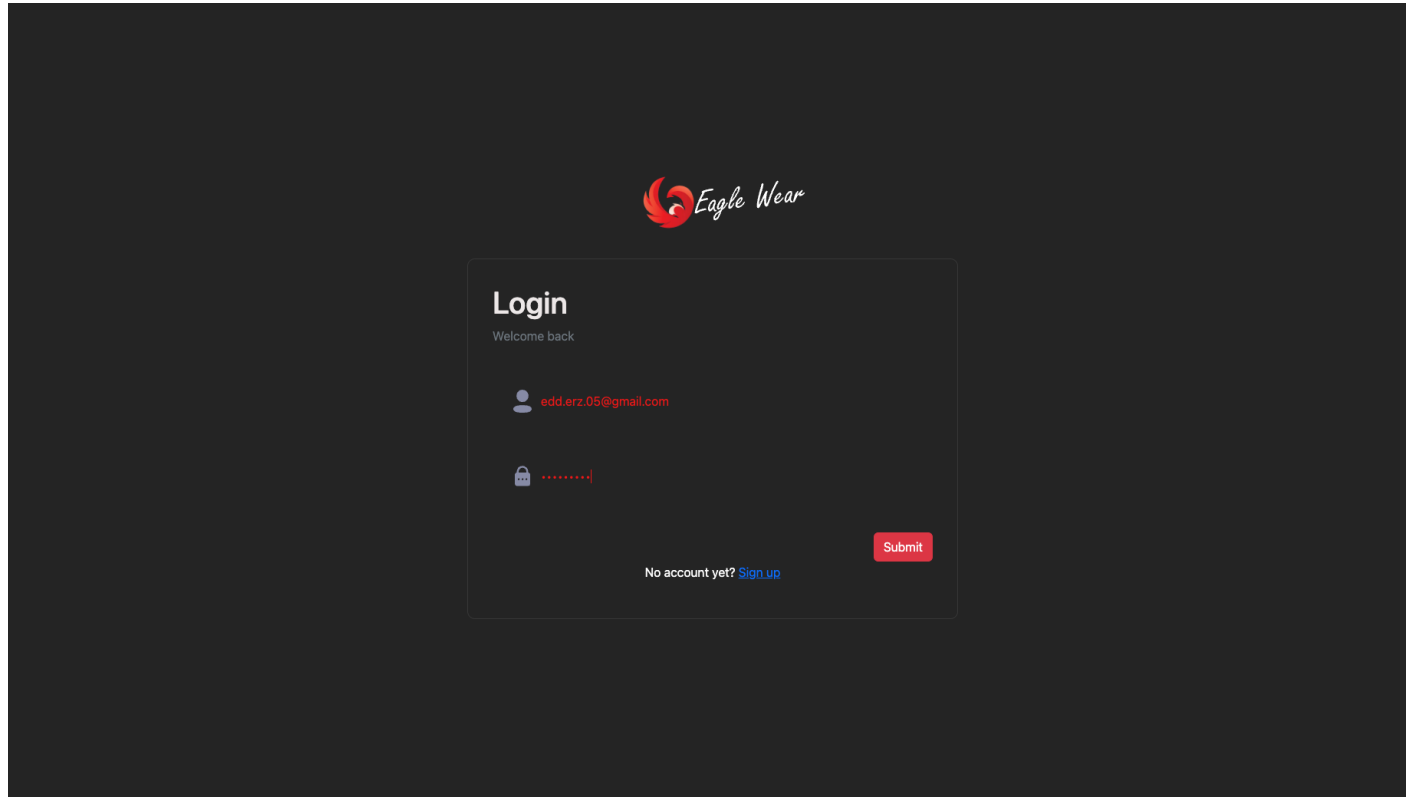
MANEJO DE CARRITO DE COMPRAS

- Solución 2 teniendo en cuenta la investigación sobre iniciar sesión con firebase authentication la cual me iba a dar un token para el usuario, decidí enfocarme en utilizar firebase como herramienta, utilizando su función de crear una base de datos para guardar y relacionar los datos que yo fuera almacenando en mi carrito de compras



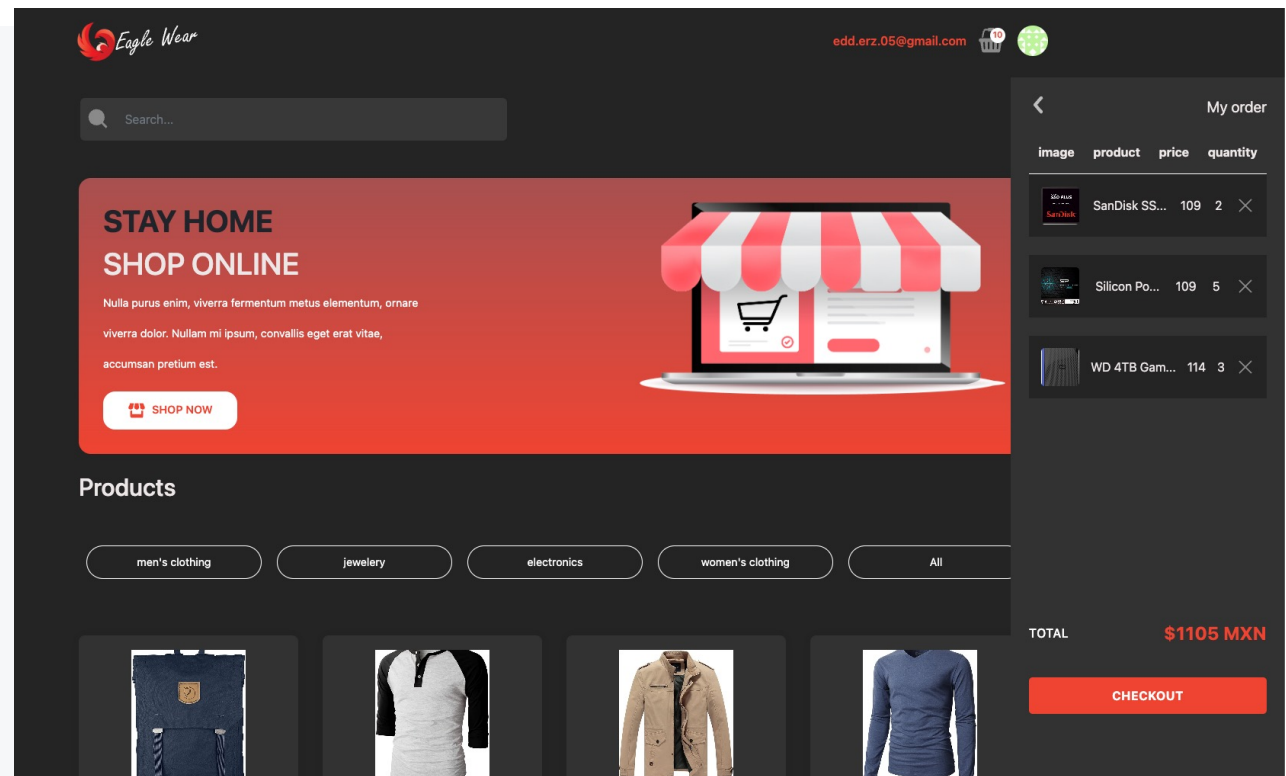
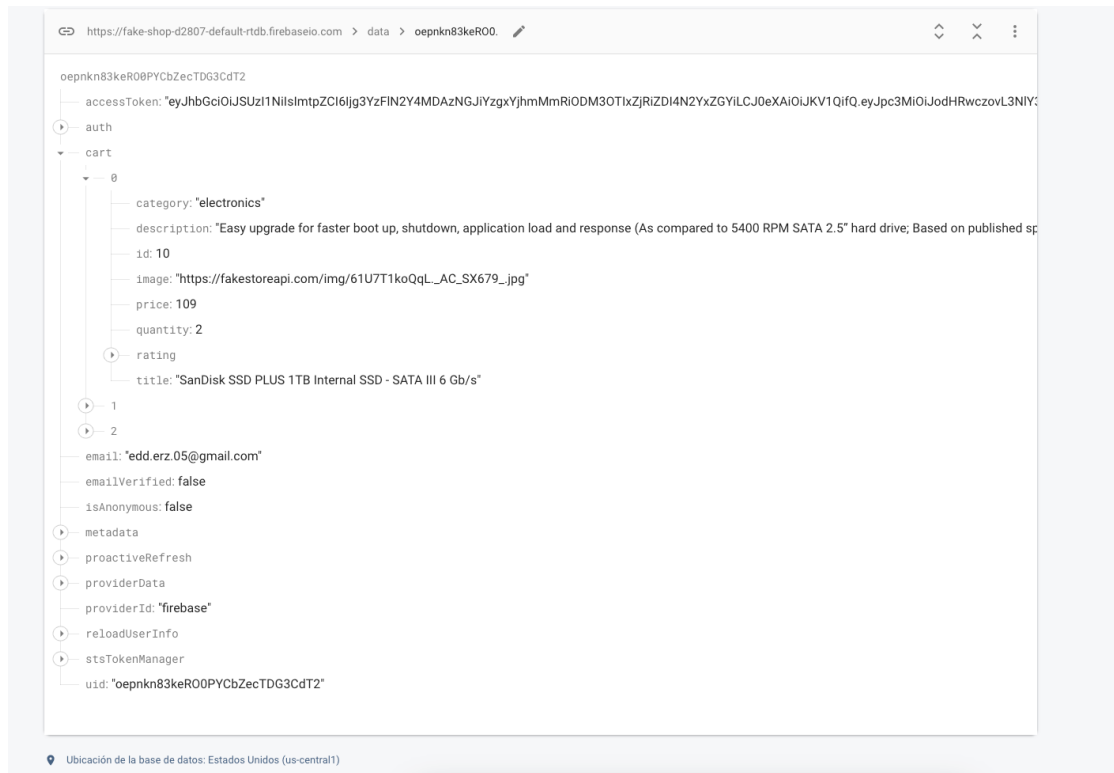
MANEJO DE CARRITO DE COMPRAS / LOGIN

Al registrarme estoy generando un token, el cual me sirve para inciar sesión

A screenshot of a login page for 'Eagle Wear'. The page has a dark gray background. At the top center is the 'Eagle Wear' logo, which consists of a red circular icon with a white eagle head inside, followed by the text 'Eagle Wear' in a white script font. Below the logo is a white-bordered box containing the login form. The form has the title 'Login' in bold white text, followed by the subtitle 'Welcome back' in a smaller white font. There are two input fields: the first is for the email address, with the placeholder text 'edd.erz.05@gmail.com' in red; the second is for the password, with a red lock icon and a series of red dots. To the right of the password field is a red 'Submit' button. At the bottom of the form, there is a link that says 'No account yet? Sign up' in white text, where 'Sign up' is a blue hyperlink.

MANEJO DE CARRITO DE COMPRAS

Entrando a la pagina principal se puede observar el correo electronico que esta logueando y sus productos cargados. Estos productos estan guardados en la base de datos, y mediante un endpoint se leen y se muestran en el carrito de compras.



MANEJO DE CARRITO DE COMPRAS / RESULTADOS

- Como mencionaba realizar esta parte me ayudo a saber hay muchas soluciones y formas de abstraer el problema, los resultados de esta segunda versión y uno de los retos fue poder dar solución al estado en el que los productos se fueran actualizando en tiempo real al momento de agregarlos al carrito.

API Y MANEJO DE ERRORES

Para esta solución fue enviar un mensaje de alerta en cada petición a la información donde le mencione al usuario hay un error para obtener la información

```
const getData = async (id) => {  
  const apiURL = id ? `${URL_BASE}` : URL_BASE;  
  try {  
    const response = await fetch(apiURL);  
    const data = await response.json();  
    return data;  
  } catch (error) {  
    alert("Fetch Error", error);  
  }  
}  
  
const getUserById = async (id) => {  
  const apiURL = id ? `${URL_USER}/${id}/.json` : URL_USER;  
  try {  
    const response = await fetch(apiURL);  
    const data = await response.json();  
    return data;  
  } catch (error) {  
    alert("Fetch Error", error);  
  }  
}
```

MUCHAS GRACIAS POR TU ATENCIÓN

- Esperando que haya podido cumplir con el objetivo de dar solución a este reto, me gustaría recibir el feedback sobre puntos de mejora y sin mas que decir muchas gracias por la oportunidad de realizar esta prueba que tuvo muchos retos que ayudan a mejorar mi proceso de solución de problemas