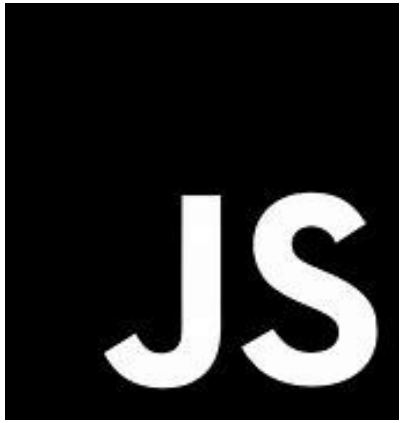
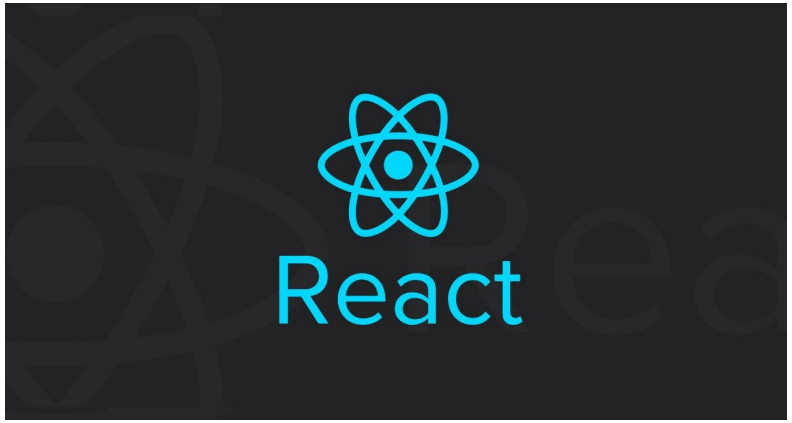
comfama

Software Development Challenge

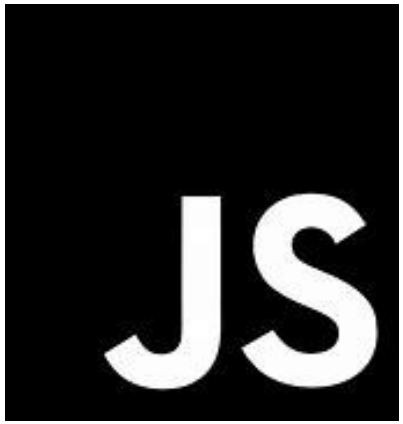
Ximena Vasco

1. Front-end technologies

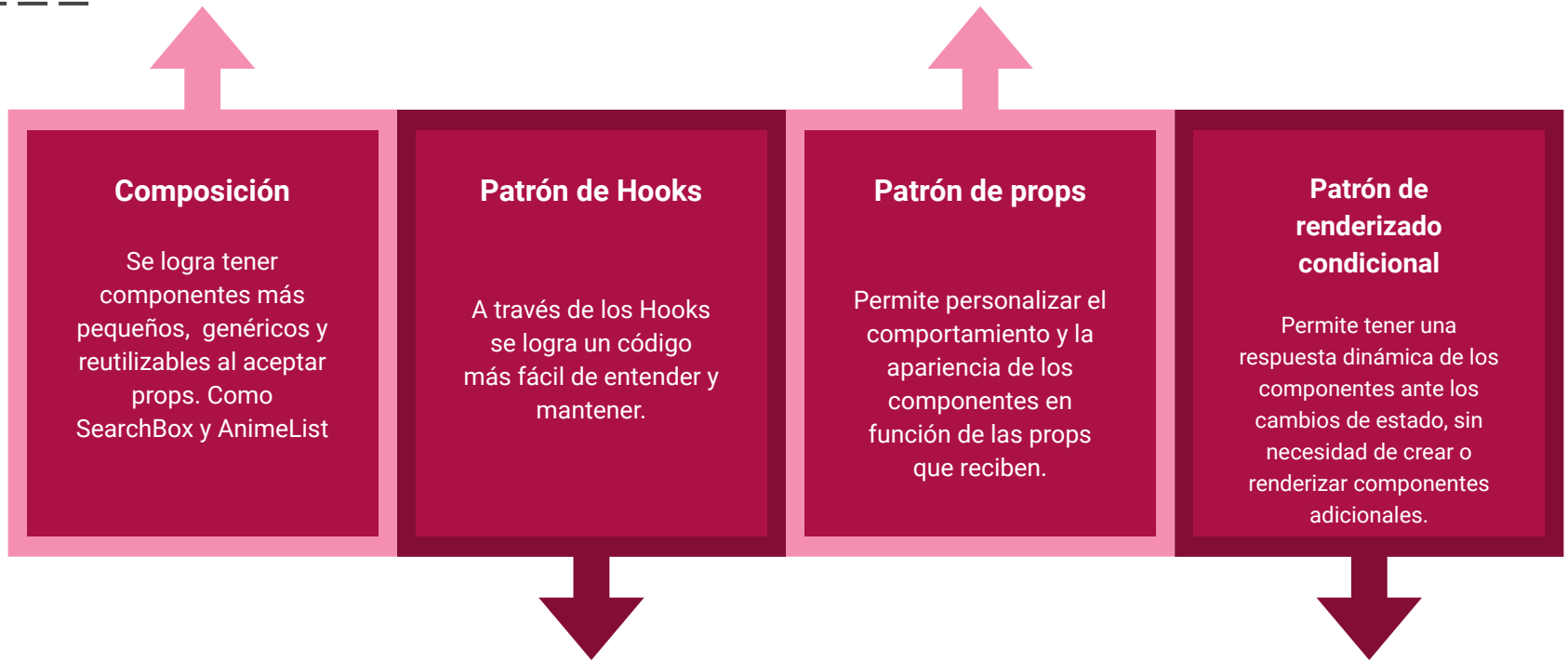




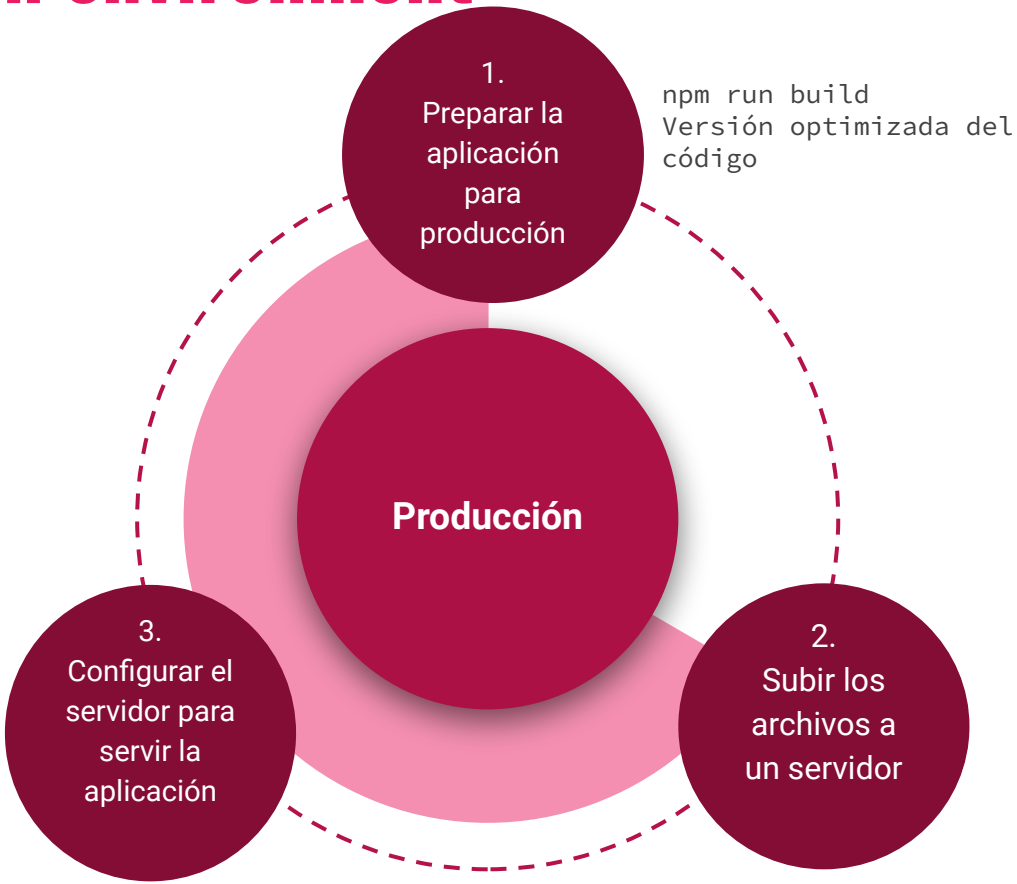
2. Back-end technologies



3. Design Pattern



4. Production environment



5. Data persistence

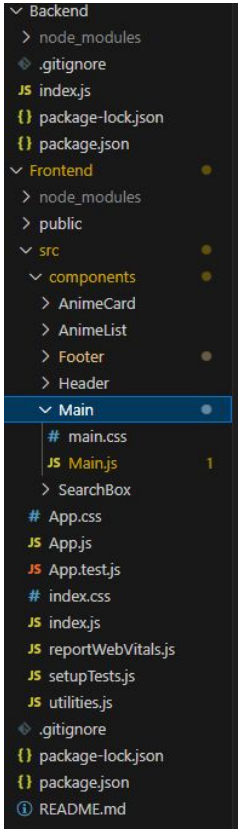
6. Relevant points (1/5)

Estructura de Carpetas

Dos Directorios principales:

1. ***Backend***: contiene un archivo principal, index.js, el cual contiene el servidor a través de express y los endpoints necesarios.

2. ***Frontend***: Dentro del cual se encuentra “src”, que contiene una carpeta “Componentes”, la cual aloja un directorio correspondiente a cada componente usado y éste a su vez contiene el archivo .js con la lógica y el .css con el estilo. Adicional, hay un archivo “utilities.js”, el cual contiene los fetch.



6. Relevant points (2/5)

```
function Main() {
  const [search, setSearch] = useState("Naruto");
  const [animeData, setAnimeData] = useState();
  const [error, setError] = useState(null);
  const [isLoading, setIsLoading] = useState(false);
  const [scoreAvg, setScoreAvg] = useState([]);
```

Estados

Dentro del componente Main, se definen los estados de la app:

1. ***Search***: criterio de búsqueda definido por el usuario.
2. ***animeData***: Información obtenida de la Api externa, ligada al criterio de búsqueda.
3. ***scoreAvg***: Información obtenida de la Api interna, ligada a la información del anime.
4. ***error***: estado de error durante la carga de la información.
5. ***isloading***: estado de carga de la información.

6. Relevant points (3/5)

Fetch

1. Petición de la información del anime definido por el usuario a la Api externa. La lógica de la función getAnimeData está alojada en utilities.
2. Petición a la Api interna del promedio del score de las diferentes temporadas del anime. La Api funciona en el servidor alojado en el puerto 3001

```
const getAnimeDataList = async () => {
  setIsLoading(true);
  try {
    const data = await getAnimeData(search);
    setAnimeData(data);
  } catch (error) {
    setError(error);
  } finally {
    setIsLoading(false);
  }
};

useEffect(() => {
  getAnimeDataList();
}, []);
```

```
useEffect(() => {
  if (animeData) {
    const scoresList = animeData.map((anime) => anime.score);

    getScoresAvg(scoresList)
      .then((data) => {
        setScoreAvg(data);
      })
      .catch((error) => {
        console.log(error);
      });
  }
}, [animeData]);
```

6. Relevant points (4/5)

Archivo “utilities.js”

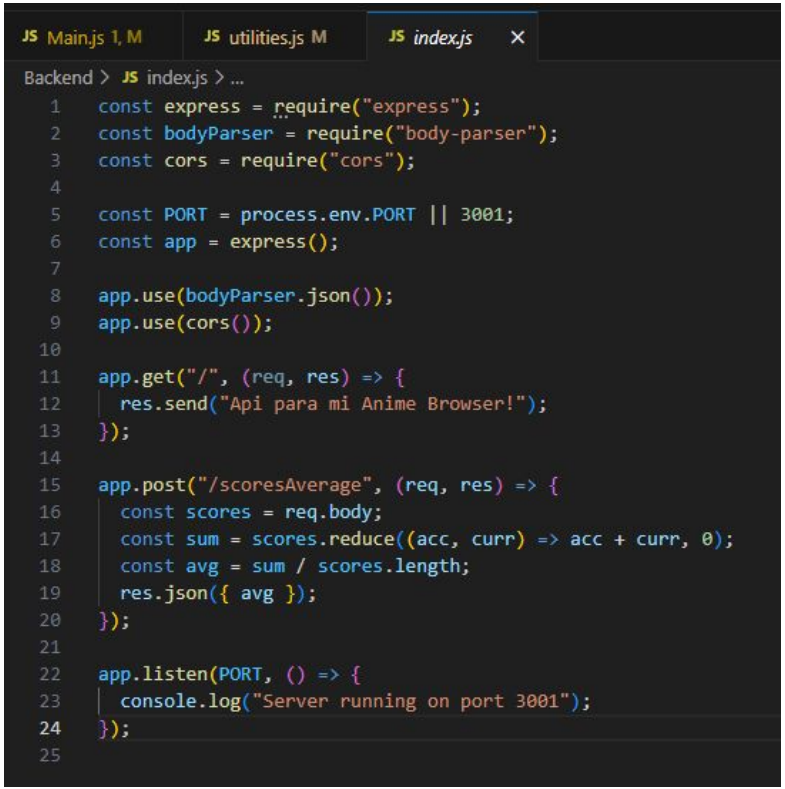
1. Este archivo aloja la lógica de las peticiones realizadas a la Api externa e interna, para el correcto funcionamiento de la App.

6. Relevant points (5/5)

Servidor y endpoints

En el archivo index.js del directorio Backend se encuentra:

1. El servidor creado e iniciado en el puerto 3001, a través de express.
2. El middleware que permite parsear la información que llega desde el front.
3. La configuración de cors.
4. el endpoint ***/scoresAverage***, que calcula el promedio de los scores de las diferentes temporadas del anime.



7. Aspects with more difficulties

Middleware	Congruencia de la información entre el front y el back.
Codigo Modular	Modularizar el código, con el fin de lograr mejor comprensión para el lector y mejor mantenimiento y/o modificación en el futuro.
Errores y estado de carga	Lograr capturar los errores y manejar el estado de carga con el fin de dar una mejor experiencia al usuario.

Gracias!