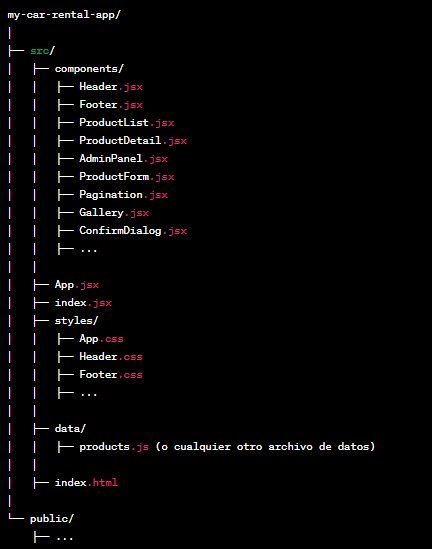




**SPRINT 1:**  
Desarrollar la estructura básica de un sitio web de alquiler de autos.

Debo agregar funcionalidades de registro, visualización y eliminación de productos.



**Historias de Usuario:**

Cero: Creación del proyecto. Instalar Node.js y crear React App “my-car-rental-app”.

Primero: Colocar encabezado. Como usuario, quiero un encabezado que muestre el logo de la aplicación y una barra para facilitar la navegación en el sitio.

1. El header debe ocupar el 100% de la pantalla en todas las páginas de la aplicación.
2. El header debe estar fijo en la parte superior de la página, incluso al hacer scroll, para que siempre sea visible y accesible para el usuario.
3. El header debe ser consistente en todas las páginas de la aplicación y debe ser fácil de usar y entender para el usuario.
4. El header debe ser optimizado para diferentes dispositivos y resoluciones de pantalla.
5. Dentro del header debe existir un bloque alineado a la izquierda que contenga el logotipo y el lema de la empresa.
6. Al hacer clic en el logotipo o el lema, el usuario debe ser redirigido a la página principal de la aplicación. Dentro del header debe existir un bloque alineado a la derecha que contenga dos botones: “Crear cuenta” e “Iniciar sesión” (Sin funcionalidad).

**Archivo: Header.jsx**

import React from 'react';

import { Link } from 'react-router-dom';

const Header = () => {

return (

<header className="header">

<div className="logo">

<Link to="/">Mi Empresa de Alquiler de Autos</Link>

</div>

<div className="user-actions">

<Link to="/register">Crear cuenta</Link>

<Link to="/login">Iniciar sesión</Link>

</div>

</header>

);

};

export default Header;

Segundo: Definir el cuerpo del sitio. Como administrador, quiero que el color de fondo del sitio web esté en armonía con la identidad de marca de la empresa para mejorar la estética y la coherencia visual de la aplicación.

1. El body debe tener un color de background que corresponda a la identidad de marca.
2. El body debe ocupar el 100 % del alto de la pantalla.
3. El body debe ser optimizado para diferentes dispositivos y resoluciones de pantalla.
4. Se debe visualizar las tres secciones o bloques: buscador, categorías y recomendaciones de los productos.

**Archivo : Body.jsx**

import React from 'react';

const Body = () => {

const bodyStyle = {

background: '#007acc', // Color de fondo personalizado

minHeight: '100vh', // Ocupa el 100% del alto de la pantalla

display: 'flex',

flexDirection: 'column',

alignItems: 'center',

};

return (

<div style={bodyStyle}>

{/\* Contenido de las secciones \*/}

<div className="search-section">

{/\* Contenido del buscador \*/}

</div>

<div className="categories-section">

{/\* Contenido de las categorías \*/}

</div>

<div className="recommendations-section">

{/\* Contenido de las recomendaciones \*/}

</div>

</div>

);

};

export default Body;

Tercero: Registrar producto. Como administrador, quiero poder agregar nuevos productos para mantener actualizado el catálogo de productos disponibles.

1. El producto debe visualizarse en el listado de productos. *(Relacionada con historia #4)*
2. El panel de administración debe contener un botón "Agregar producto".
3. La página de "Agregar producto" debe incluir campos para ingresar información relevante del producto, como nombre, descripción e imagen.
4. Se debe poder subir una o más imágenes del producto.
5. Se debe poder guardar el producto y este debe ser agregado correctamente a la base de datos del sitio.
6. Si se intenta agregar un producto con un nombre que ya existe, se debe mostrar un mensaje de error indicando que el nombre ya está en uso.

**Archivo: ProductForm.jsx**

import React, { useState } from 'react';

const ProductForm = ({ onAddProduct }) => {

const [product, setProduct] = useState({

name: '',

description: '',

image: '',

});

const handleInputChange = (e) => {

const { name, value } = e.target;

setProduct({ ...product, [name]: value });

};

const handleAddProduct = () => {

if (product.name && product.description) {

onAddProduct(product);

setProduct({ name: '', description: '', image: '' });

}

};

return (

<div className="product-form">

<h2>Agregar Producto</h2>

<input

type="text"

name="name"

placeholder="Nombre del Producto"

value={product.name}

onChange={handleInputChange}

/>

<textarea

name="description"

placeholder="Descripción del Producto"

value={product.description}

onChange={handleInputChange}

/>

<input

type="text"

name="image"

placeholder="URL de la Imagen"

value={product.image}

onChange={handleInputChange}

/>

<button onClick={handleAddProduct}>Agregar Producto</button>

</div>

);

};

export default ProductForm;

Cuarto: Visualizar productos en el home. Como usuario, quiero ver una lista de productos aleatorios en el home cuando ingreso por primera vez para obtener referencias de productos que encontraré.

1. Se deben mostrar como máximo 10 productos aleatorios.
2. Los productos no deben repetirse.
3. Se deben distribuir en 2 columnas y 5 filas como máximo.
4. Garantizar que la lista de productos aleatorios sea verdaderamente aleatoria, sin repetir productos y sin seguir un patrón previsible

**Archivo: ProductList.jsx v.1.**  
  
import React from 'react';

const ProductList = ({ products }) => {

return (

<div className="product-list">

<h2>Lista de Productos</h2>

<ul>

{products.map((product, index) => (

<li key={index}>

<h3>{product.name}</h3>

<p>{product.description}</p>

<img src={product.image} alt={product.name} />

</li>

))}

</ul>

</div>

);

};

export default ProductList; **Archivo: ProductList.jsx v.2.**

import React from 'react';

const ProductList = ({ products }) => {

// Función para mezclar aleatoriamente la lista de productos

const shuffleProducts = (array) => {

const shuffledArray = [...array];

for (let i = shuffledArray.length - 1; i > 0; i--) {

const j = Math.floor(Math.random() \* (i + 1));

[shuffledArray[i], shuffledArray[j]] = [shuffledArray[j], shuffledArray[i]];

}

return shuffledArray;

};

// Mezclar aleatoriamente los productos

const shuffledProducts = shuffleProducts(products);

return (

<div className="product-list">

<h2>Lista de Productos Aleatorios</h2>

<ul>

{shuffledProducts.map((product, index) => (

<li key={index}>

<h3>{product.name}</h3>

<p>{product.description}</p>

<img src={product.image} alt={product.name} />

</li>

))}

</ul>

</div>

);

};

export default ProductList;

Quinto: Visualizar detalle de producto. Como usuario, quiero poder ingresar en el detalle del producto para conocer datos básicos de éste.

1. Visualizar un bloque de header el cual deberá cubrir el 100 % del ancho de la pantalla.
2. El título del producto deben estar alineados a la izquierda.
3. La flecha para volver atrás debe estar alineada a la derecha.
4. En el body debe estar el texto descriptivo del producto y sus imágenes.

**Archivo : ProductDetail.jsx**

import React from 'react';

const ProductDetail = ({ product }) => {

return (

<div className="product-detail">

<h2>Detalles del Producto</h2>

<div className="product-info">

<h3>{product.name}</h3>

<p>{product.description}</p>

<img src={product.image} alt={product.name} />

</div>

</div>

);

};

export default ProductDetail;

Sexto : Visualizar galería de imágenes. Como usuario quiero poder visualizar una galería de Imágenes del producto para tener una idea clara de cómo se ve y facilitar mi decisión de reserva.

1. Debe presentar un bloque al 100 % del ancho del contenedor que incluye 5 imágenes.
2. La imagen principal debe estar posicionada en la mitad izquierda del bloque de imágenes.
3. En la versión desktop, una grilla de 2 filas y 2 columnas debe estar en la mitad derecha del bloque donde se visualizarán las 4 imágenes restantes.
4. El bloque debe incluir en su región inferior derecha el texto “Ver más”, el cual, permitirá acceder a un componente para ver todas las imágenes disponibles del producto.
5. La galería debe ser responsiva a los diferentes dispositivos como mobile y tablet.

**Archivo : Gallery.jsx**

import React, { useState } from 'react';

const Gallery = ({ images }) => {

const [selectedImage, setSelectedImage] = useState(images[0]);

const handleImageClick = (image) => {

setSelectedImage(image);

};

return (

<div className="gallery">

<div className="main-image">

<img src={selectedImage} alt="Producto" />

</div>

<div className="thumbnail-images">

{images.map((image, index) => (

<img

key={index}

src={image}

alt={`Producto ${index + 1}`}

onClick={() => handleImageClick(image)}

/>

))}

</div>

</div>

);

};

export default Gallery;

Septima : Panel de administración. Como administrador quiero poder ingresar a las funciones de administración de mi negocio de forma simple para gestionarlo.

1. Debe existir una url /administración para acceder al panel
2. Se visualiza un menú con todas las funciones desarrolladas para la administración.
3. No debe ser responsive. Si se desea ingresar al panel desde un dispositivo móvil, debe mostrarse un mensaje que no está disponible.

**Archivo: AdminPanel.jsx**

import React from 'react';

const AdminPanel = () => {

return (

<div className="admin-panel">

<h2>Panel de Administración</h2>

<ul>

<li>

<a href="/admin/add-product">Agregar Producto</a>

</li>

<li>

<a href="/admin/list-products">Lista de Productos</a>

</li>

<li>

<a href="/admin/delete-product">Eliminar Producto</a>

</li>

</ul>

</div>

);

};

export default AdminPanel;

Octavo: Listar Productos: Como administrador, quiero poder visualizar una lista de productos disponibles para poder gestionarlos con facilidad.

1. En el panel de administración debe existir un botón "Lista de productos".
2. En la página “Lista de productos” se deben listar todos los productos disponibles en el sitio.
3. Debe existir las columna: “Id”, “Nombre” y “Acciones”

**Archivo : AdminPanel.jsx**

import React from 'react';

const AdminPanel = () => {

return (

<div className="admin-panel">

<h2>Panel de Administración</h2>

<ul>

<li>

<a href="/admin/add-product">Agregar Producto</a>

</li>

<li>

<a href="/admin/list-products">Lista de Productos</a>

</li>

<li>

<a href="/admin/delete-product">Eliminar Producto</a>

</li>

</ul>

</div>

);

};

export default AdminPanel;

**Archivo : ProductListAdmin.jsx**

import React from 'react';

const ProductListAdmin = ({ products }) => {

return (

<div className="product-list-admin">

<h2>Lista de Productos (Administración)</h2>

<ul>

{products.map((product, index) => (

<li key={index}>

<h3>{product.name}</h3>

<p>{product.description}</p>

<img src={product.image} alt={product.name} />

</li>

))}

</ul>

</div>

);

};

export default ProductListAdmin;

Noveno : Colocar Pie de Página. Como usuario, quiero un footer en la aplicación para poder mostrar información relevante de la empresa.

1. Se debe crear un footer que ocupe el 100% del ancho de la pantalla y que esté ubicado en el pie de página en todas las páginas de la aplicación.
2. Dentro del footer, debe existir un bloque alineado a la izquierda que contenga el isologotipo de la empresa, el año y el copyright correspondiente.
3. El footer debe ser optimizado para diferentes dispositivos y resoluciones de pantalla.
4. El isologotipo, el año y el copyright deben ser legibles y estar diseñados de acuerdo con la identidad visual de la empresa.

**Archivo : Footer.jsx**

import React from 'react';

const Footer = () => {

const currentYear = new Date().getFullYear();

return (

<footer className="footer">

<div className="footer-content">

<div className="footer-logo">

<img src="/logo.png" alt="Logo de la Empresa" />

</div>

<div className="footer-info">

<p>&copy; {currentYear} Mi Empresa de Alquiler de Autos</p>

</div>

</div>

</footer>

);

};

export default Footer;

Decimo : Paginar productos. Como usuario, quiero ver resultados limitados en el home para ver el contenido de forma ordenada.

1. Dividir el listado de productos en páginas de no más de 10 productos.
2. Mostrar un contador de páginas funcional que permita la navegación entre las mismas.
3. Mostrar un botón o enlace para que el usuario pueda ir hacia atrás, hacia delante o al inicio del listado de productos.

**Archivo : Pagination.jsx**  
import React from 'react';

const Pagination = ({ itemsPerPage, totalItems, paginate }) => {

const pageNumbers = [];

for (let i = 1; i <= Math.ceil(totalItems / itemsPerPage); i++) {

pageNumbers.push(i);

}

return (

<nav>

<ul className="pagination">

{pageNumbers.map((number) => (

<li key={number} className="page-item">

<a onClick={() => paginate(number)} href="!#" className="page-link">

{number}

</a>

</li>

))}

</ul>

</nav>

);

};

export default Pagination;

Undecimo : Eliminar producto. Como administrador, quiero poder eliminar un producto para mantener una lista actualizada y precisa de los productos disponibles.

1. En el listado de productos (Panel de administración) por cada producto debe existir la acción “Eliminar producto”.
2. Al presionar “Eliminar producto” debe aparecer un mensaje de confirmación.
3. Si se acepta la eliminación del producto el mismo debe eliminarse en la base de datos.
4. El producto eliminado no debe mostrarse en listado de productos.
5. Si no se acepta la acción de eliminar producto, no se deben realizar cambios.

**Archivo: ProductListAdmin.jsx**  
import React from 'react';

const ProductListAdmin = ({ products, onDeleteProduct }) => {

const handleDelete = (product) => {

onDeleteProduct(product);

};

return (

<div className="product-list-admin">

<h2>Lista de Productos (Administración)</h2>

<ul>

{products.map((product, index) => (

<li key={index}>

<h3>{product.name}</h3>

<p>{product.description}</p>

<img src={product.image} alt={product.name} />

<button onClick={() => handleDelete(product)}>Eliminar</button>

</li>

))}

</ul>

</div>

);

};

export default ProductListAdmin;

**BACKLOG DE TAREAS PENDIENTES:**

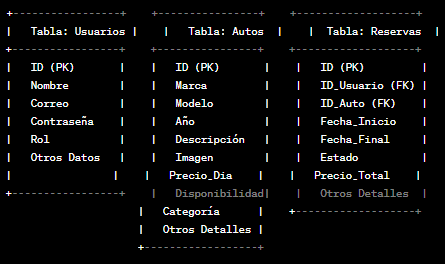
1. Configurar la Base de Datos:  


Tabla de Usuarios:

Campos: ID (Clave primaria), Nombre, Correo Electrónico, Contraseña, Rol (Administrador o Usuario), Otros datos de perfil.

Relaciones: Puede haber una relación uno a muchos con la tabla de Reservas si un usuario puede realizar múltiples reservas.

Tabla de Autos:

Campos: ID (Clave primaria), Marca, Modelo, Año, Descripción, Imagen, Precio por Día, Disponibilidad, Categoría, Otros detalles del automóvil.

Relaciones: Puede haber relaciones uno a muchos con la tabla de Reservas si un automóvil puede ser reservado varias veces.

Tabla de Reservas:

Campos: ID (Clave primaria), ID del Usuario (Clave foránea), ID del Auto (Clave foránea), Fecha de Inicio, Fecha de Finalización, Estado de la Reserva, Precio Total, Otros detalles de la reserva.

Relaciones: Puede tener una relación con las tablas de Usuarios y Autos.

2. Conexión a la Base de Datos:

Establecer una conexión al DBMS utilizando el controlador JDBC correspondiente. Esto implica proporcionar la URL de conexión, credenciales de usuario y otras configuraciones necesarias.

3. Crear Controladores en JAVA para Agregar, Eliminar Productos.

Desarrolla la lógica para acceder y manipular los datos de la base de datos.

**Archivo: ProductController.java**

public class ProductController {

public void deleteProduct(int productId) {

// Conexión a la base de datos

Connection connection = // Obtener conexión a la base de datos

// Consulta SQL para eliminar el producto

String sql = "DELETE FROM productos WHERE id = ?";

try (PreparedStatement statement = connection.prepareStatement(sql)) {

statement.setInt(1, productId);

statement.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

// Manejo de errores

}

}

}

4. Utilizar SpringBoot para exponer endpoints RESTful que permita interactuar con el frontend.

5. Creación de Test.

NOTAS:

**Estructura de Carpetas:**

1. **src:** Directorio principal de código fuente.
   * **main:** Código fuente principal de la aplicación.
     + **java:** Directorio para archivos Java.
       - **com.tuempresa:** Paquete principal de tu aplicación.
         * **controller:** Contendrá los controladores, como **ProductController**.
         * **model:** Contendrá las clases de modelo, como **Product**.
         * **repository:** Contendrá las interfaces de repositorio JPA, como **ProductRepository**.
         * **service:** Contendrá servicios, si es necesario.
     + **resources:** Directorio para archivos de recursos.
       - **static:** Puedes poner aquí recursos estáticos, como archivos CSS, JavaScript, imágenes, etc.
       - **templates:** Puedes colocar plantillas HTML aquí si estás utilizando Spring MVC.
       - **application.properties:** Archivo de configuración de Spring Boot.
   * **test:** Directorio para pruebas unitarias y de integración.
2. **pom.xml:** Archivo de configuración de Maven si estás utilizando Maven para gestionar las dependencias de tu proyecto.

**Diagrama de Componentes:**

Puedes utilizar un diagrama de componentes para representar las diferentes partes de tu aplicación:

1. **Controlador (ProductController):** Representa la capa de controladores que maneja las solicitudes del frontend. Contiene los endpoints RESTful para listar, agregar y eliminar productos.
2. **Servicio (ProductService, si es necesario):** Puede ser una capa intermedia entre el controlador y el repositorio si necesitas lógica adicional antes de acceder a la base de datos.
3. **Repositorio (ProductRepository):** Encargado de interactuar con la base de datos y proporcionar métodos para realizar operaciones de base de datos, como guardar, eliminar y buscar productos.
4. **Base de Datos SQL:** Representa la base de datos que almacena la información de los productos.
5. **Frontend (React):** Representa la parte de frontend de tu aplicación, que se comunicará con los endpoints RESTful del controlador.