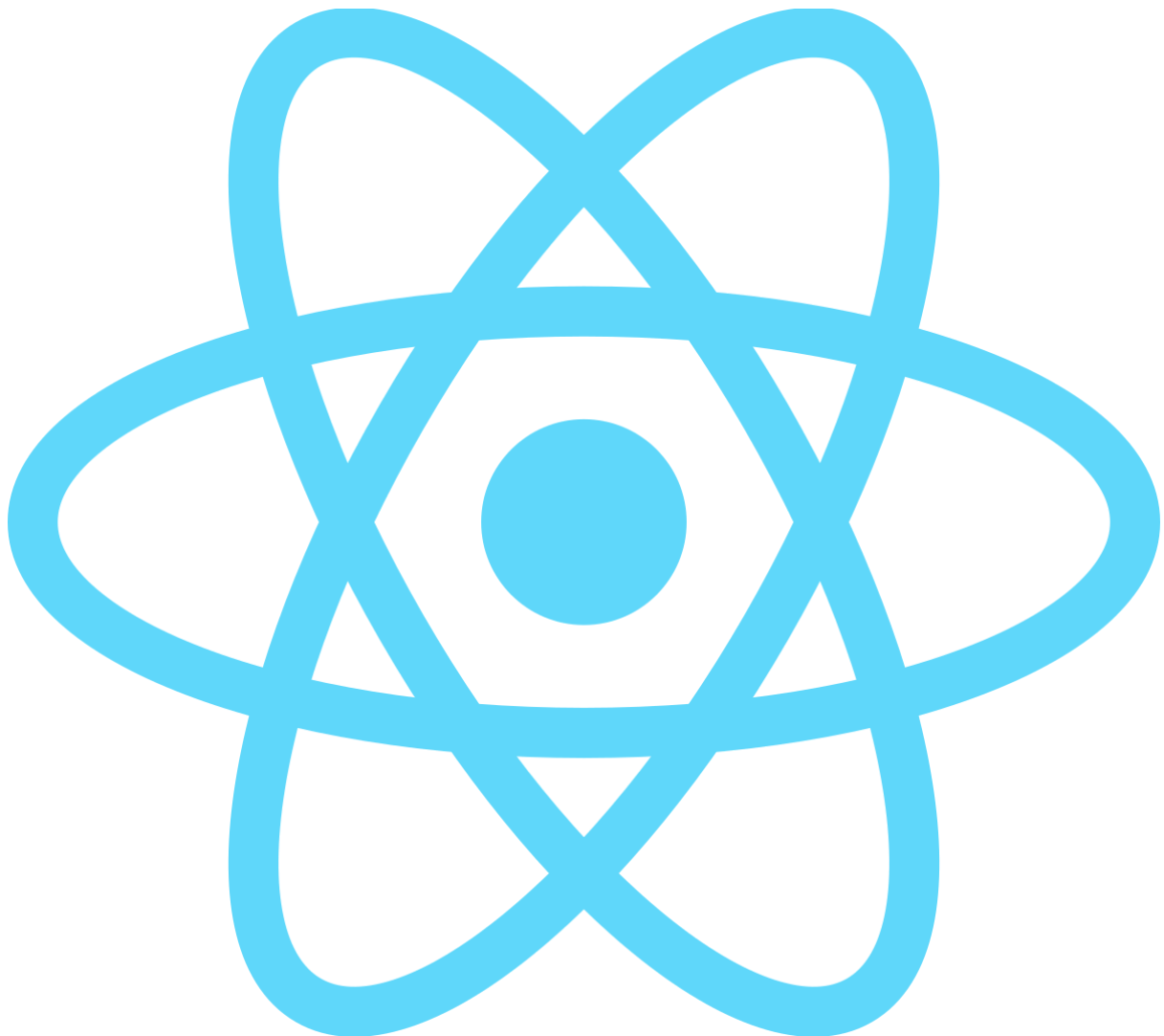


MEMORIA - PART 1 - REACT



ÍNDEX

PARTS MÉS IMPORTANTS DEL PROJECTE	4
COMPLICACIONS	11

EN QUÈ CONSISTEIX AQUEST PROJECTE?

L'objectiu d'aquest projecte individual és crear una aplicació client amb React i Vite basada en la temàtica pròpia utilitzada al projecte anterior. Inicialment, les dades s'emmagatzemen en un fitxer JSON i es gestionaran mitjançant un context global.

Generarem un JSON amb cotxes en el meu cas ja que puc escollir una temàtica per el meu projecte, ho farem en formes de context en React.

Utilitzarem un menú interactiu amb React Router de React per poguer navegar entre pàgines.

Farem una barra de navegació amb Link.

Comptarem amb la pàgina principal que serà un catàleg y que mostrarà els elements en forma de carta.

Cada element tindrà un botó d'informació detallada sobre l'element, la vista detallada de cada element contindrà informació detallada, uns botons per modificar y eliminar aquest producte, a més de poguer fer reserves y marcat com usats aquest elements.

Els formularis utilitzarem useForm de React per gestionar els formularis, on aquest formularis contindran, inputs de tipus text, de números, select/options, checkbox amb opcions, Radio Button per posar característica concreta i més.

Estructura dels components → Catalog on mostra l'array d'elements en forma de targeta → Item card → Mostra la info bàsica d'un element amb un botó per accedir a la vista → ItemDetail → Mostra la informació completa amb opcions per modificar, eliminar o interactuar amb l'element.

PARTS MÉS IMPORTANTS DEL PROJECTE

Dins d'app s'afegeix el react router dom per poguer controlar les routes y poguer fer navegació dins dels nostres components, a més envoltam tots els components en el nostre context per poguer controlar i heretar els cotxes

```
Codeium: Refactor | Explain | Generate JSDoc | X
function App() {
  return (
    <CocheProvider>
      <Router>
        <Header />
        [/* Agregamos rutas para nuestra web */]
        <Routes>
          <Route path="/" element={<Catalog />} />
          <Route path="/catalog" element={<Catalog />} />
          <Route path="/sobre-nosotros" element={<SobreNosotros />} />
          <Route path="/formulario" element={<Formulario />} />
        </Routes>
      </Router>
    </CocheProvider>
  );
}
```

Declaració del JSON i el context, en aquest cas del cotxe

- Funcions claus → agregarCoche, eliminarCoche, editarCoche

```
Codeium: Refactor | Explain | Generate JSDoc | X
const agregarCoche = (nuevoCoche) => {
  setCoches([...coches, { ...nuevoCoche, id: coches.length + 1, reservado: false }]);
};

Codeium: Refactor | Explain | Generate JSDoc | X
const eliminarCoche = (id) => {
  setCoches(coches.filter((coche) => coche.id !== id));
};

Codeium: Refactor | Explain | Generate JSDoc | X
const editarCoche = (cocheEditado) => {
  setCoches(
    coches.map((coche) =>
      coche.id === cocheEditado.id ? cocheEditado : coche
    )
  );
};
```

Afegirem a agregarCoche el nou cotxe utilitzant el setter afegint al JSON el següent cotxe

Eliminarem el cotxe amb `eliminarCoche` recollint l'id del JSON actualitzant i utilitzant un condicional per el filter

Editarem el cotxe amb `editarCoche` recollint tots els cotxes amb el mètode `map` i fent un operador ternari amb el condicional del cotxe editable

Envoltem tot el context en el provider perquè puguin manipular tota la informació tot aquell componen que estigui envolt en aquest context (per això s'afegeix `children`)

```
//Colocamos el contexto para que puedan heredarlo los hijos para que puedan manipular las funciones, coches, etc
return (
  <CocheContext.Provider value={{ coches, eliminarCoche, editarCoche, agregarCoche }}>
    {children}
  </CocheContext.Provider>
);
};
export { CocheContext, CocheProvider };
```

En `catalog` simplement amb `map` tornem tots els cotxes i ho afegim a `ItemCard` que es on controlem tots els cotxes

```
Codeium: Refactor | Explain | Generate JSDoc | X
export const Catalog = () => {
  const { coches } = useContext(CocheContext);

  return (
    <div>
      <h2>Coches históricos</h2>
      <div className="listaCoches">
        { /* Recorremos nuestros coches con map */ }
        {coches.map((coche) => (
          <ItemCard key={coche.id} coche={coche} />
        ))}
      </div>
    </div>
  );
};
```

En `ItemCard` mostrem tots els cotxes en forma de carta, on tindrem per reservar, editar cotxes i eliminar-ho del JSON declarem 3 hooks i un context per heretar les funcions (**No he fet `detailCard` ja que ho he fet amb el botó de mostrar més informació**)

```
Codeium: Refactor | Explain | Generate JSDoc | X
export const ItemCard = ({ coche }) => {
  const { eliminarCoche, editarCoche } = useContext(CocheContext);
  const [mas, setMas] = useState(false);
  const [editando, setEditando] = useState(false);
  const [formData, setFormData] = useState({ ...coche });
```

Aquestes son les principals funcions on podem veure que poguem actualitzar el nostre cotxe i poguem reservar-ho

```
//Función booleana para cuando den al botón mostrar más cambie el estado
Codeium: Refactor | Explain | X
const mostrarMas = () => {
  setMas(!mas);
};

//Actualiza el estado cuando el usuario edita los datos del coche

Codeium: Refactor | Explain | Generate JSDoc | X
const handleChange = (e) => {
  const { name, value } = e.target;
  setFormData({
    ...formData,
    [name]: value,
  });
};

//Envía los datos a la función editar coche y los cambia


Codeium: Refactor | Explain | Generate JSDoc | X
const handleSubmit = (e) => {
  e.preventDefault();
  editarCoche(formData);
  setEditando(false);
};

//Cambia el estado del coche y modifica su estado de reserva a true
Codeium: Refactor | Explain | X
const toggleReservado = () => {
  const cocheActualizado = { ...formData, reservado: !formData.reservado };
  editarCoche(cocheActualizado);
  setFormData(cocheActualizado);
};
```

Amb aquest botó inclueim la eliminació del cotxe per id


```
<button onClick={() => eliminarCoche(coche.id)}>Eliminar</button>
```

Així es com es veu al catàleg:




Ford Model T (1927)
Color: Negro
Precio: 30000 €

ReservarMostrar más




Chevrolet Bel Air (1957)
Color: Celeste
Precio: 60000 €

ReservarMostrar más




Volkswagen Beetle (Escarabajo) (1967)
Color: Rojo
Precio: 20000 €

ReservarMostrar más




Porsche 911 (1973)
Color: Plata
Precio: 100000 €

ReservarMostrar más




Jaguar E-Type (1965)
Color: Blanco
Precio: 150000 €

ReservarMostrar más



Ferrari 250 GTO (1962)
Color: Rojo
Precio: 70000000 €

ReservarMostrar más



Chevrolet Bel Air (1957)
Color: Celeste
Precio: 60000 €

ReservarMostrar menos

Kilometraje: 30000 km
Motor: 4.6L V8
Transmisión: Automática
Combustible: Gasolina
Características: Pintura bicolor, Molduras cromadas, Asientos de banco
Descripción: El Bel Air de 1957 es un ícono de la era dorada de los automóviles estadounidenses. Con su diseño elegante y potente motor V8, es un clásico muy buscado por los coleccionistas.

EditarEliminar

ChevroletBel Air1957Celeste60000300004.6L V8AutomáticaGasolina

Pintura bicolor,Molduras cromadas,Asientos de banco

El Bel Air de 1957 es un ícono de la era dorada de los automóviles estadounidenses. Con su diseño elegante y potente motor V8, es un clásico muy buscado por los coleccionistas.

Guardar

He fet amb link en el component Header per poguer navegar entre pàgines

```
Codeium: Refactor | Explain | Generate JSDoc | X
export const Header = () => {
  return (
    <header>
      <nav>
        <ol>
          { /* Agregamos los link para que funcionen y conecten con los routes */ }
          <li><Link to="/">Inicio</Link></li>
          <li><Link to="/catalog">Lista de productos</Link></li>
          <li><Link to="/formulario">Agregar</Link></li>
          <li><Link to="/sobre-nosotros">Sobre nosotros</Link></li>
        </ol>
      </nav>
    </header>
  );
};
```

Página sobre nosotros que explico una mica la idea de l'empresa

```
import React from 'react';
//Página sobre nosotros
Codeium: Refactor | Explain | X
const SobreNosotros = () => {
  return (
    <div className="container">
      <h1>Sobre Nosotros</h1>
      <p>
        ¡Bienvenidos a Nico Mesa Classics! Somos una empresa apasionada por los coches clásicos, y estamos aquí para ayudarte a encontrar el vehículo de tus sueños.
      </p>
      <h2>Nuestra historia</h2>
      <p>
        La empresa está dirigida por Nico Mesa, un chico apasionado de los coches clásicos con años de experiencia en el sector. En Nico Mesa Classics, no solo encontrarás una
      </p>
      <h2>Nuestro objetivo</h2>
      <p>
        Nuestro objetivo es ofrecer una experiencia personalizada para nuestros clientes, garantizando que cada coche que pase por nuestras manos sea una joya única y bien cuidada.
      </p>
      <button className="cta">Explora nuestro catálogo</button>
    </div>
  );
};

export default SobreNosotros;
```

En el formulari afegim 1 context el del nostre cotxe i afegim un hook per controlar el form i les dades del nostre formulari, en els comentaris afegeixo la informació necessària per explicar el que faig en el codi


```
// Creamos un evento para cambiar los valores
Codeium: Refactor | Explain | X
const handleChange = (e) => {
  //Eventos manejados con target
  const { name, value, type, checked } = e.target;
  //Si la casilla reservado es marcada actualiza el valor correspondiente en formData (campo reservado)
  if (type === 'checkbox') {
    if (name === 'reservado') {
      setFormData({
        ...formData,
        [name]: checked
      });
    } else {
      //Variable auxiliar para marcar el tipo de combustible
      const tiposCombustible = formData.tipo_combustible;
      //Si marcan los tipos de combustible agrega el valor al formData y si no los elimina (marca vacío)
      if (checked) {
        tiposCombustible.push(value);
      } else {
        const index = tiposCombustible.indexOf(value);
        if (index > -1) {
          tiposCombustible.splice(index, 1);
        }
      }
      //Llamamos al setter para actualizar el tipo de combustible de formData
      setFormData({
        ...formData,
        tipo_combustible: tiposCombustible
      });
    }
  } else {
    setFormData({
      ...formData,
      [name]: value,
    });
  }
};
```

```
//Creamos un evento para cambiar las imagenes y actualizamos su estado
Codeium: Refactor | Explain | X
const handleImageChange = (e) => {
  const archivo = e.target.files[0];
  if (archivo) {
    const imagenCoche = `public/images/${archivo.name}`;
    setFormData({
      ...formData,
      imagen: imagenCoche
    });
  }
};

//Creamos un evento para enviar los datos a la funcion agregar coche de nuestro contexto
Codeium: Refactor | Explain | Generate JSDoc | X
const handleSubmit = (e) => {
  e.preventDefault();
  agregarCoche({
    ...formData,
    //Paso los valores a integer o float
    año: parseInt(formData.año),
    precio: parseFloat(formData.precio),
    kilometraje: parseInt(formData.kilometraje),
    //Divide la cadena recortando cada elemento del parámetro
    características: formData.caracteristicas.split(',').map((item) => item.trim()),
    reservado: formData.reservado
  });
  //Setteo el formData para seguir agregando nuevos coches una vez de envíe
  setFormData({
    marca: '',
    modelo: '',
    año: '',
    color: '',
    precio: '',
    kilometraje: '',
    motor: '',
    transmision: '',
    tipo_combustible: [],
    características: '',
    descripcion: '',
    imagen: '',
    reservado: false
  });
};
```

COMPLICACIONS

Controlar els formularis, amb la ajuda del github de Jorge he pogut controlar i millorar el maneig dels formularis i poguer comprendre com funciona els formularis en react

Una vegada havia fet el codi sencer no em funciona per el provider, pero investigant era perquè no ho estava heretant amb ningú i ja em va funcionar