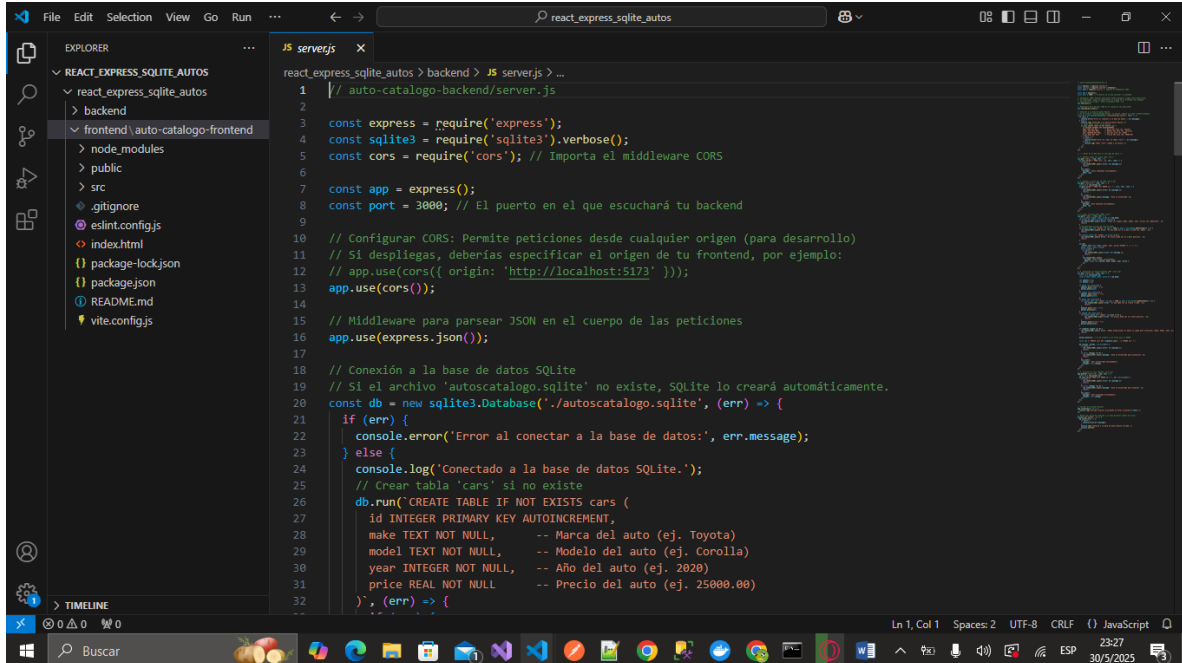
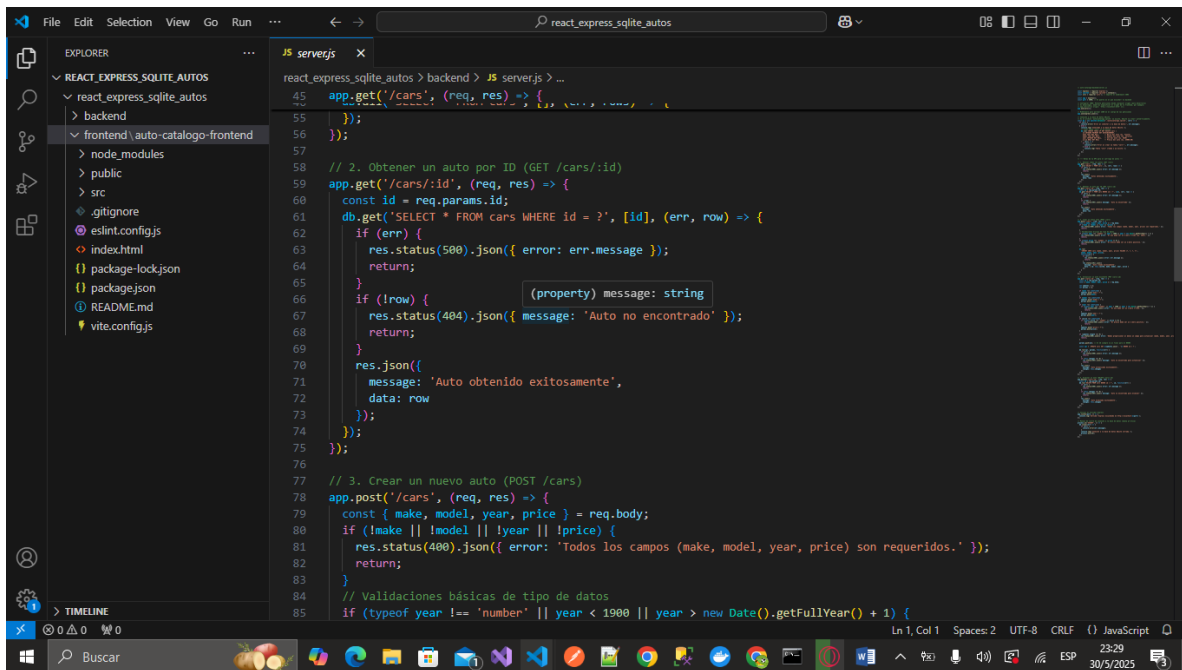


<https://github.com/leon7A/Crud-Final.git>

APLICACIÓN TERMINADA

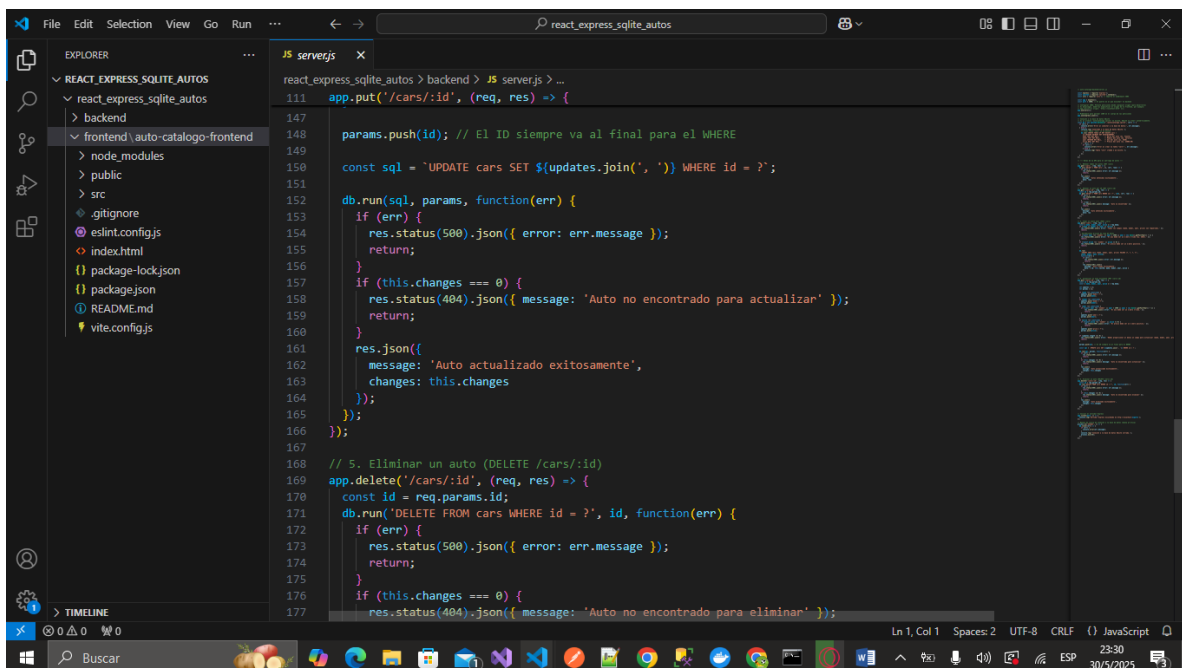


CONEXIÓN CON EL BACK-END Y CREACION DE LA TABLA



```
react_express_sqlite_autos > backend > JS server.js > ...
45 app.get('/cars', (req, res) => {
46   // 1. Obtener todos los autos (GET /cars)
47   db.get('SELECT * FROM cars', (err, row) => {
48     if (err) {
49       res.status(500).json({ error: err.message });
50       return;
51     }
52     if (!row) {
53       res.status(404).json({ message: 'Auto no encontrado' });
54       return;
55     }
56     res.json({
57       message: 'Auto obtenido exitosamente',
58       data: row
59     });
60   });
61 });
62
63 // 2. Obtener un auto por ID (GET /cars/:id)
64 app.get('/cars/:id', (req, res) => {
65   const id = req.params.id;
66   db.get('SELECT * FROM cars WHERE id = ?', [id], (err, row) => {
67     if (err) {
68       res.status(500).json({ error: err.message });
69       return;
70     }
71     if (!row) {
72       res.status(404).json({ message: 'Auto no encontrado' });
73       return;
74     }
75     res.json({
76       message: 'Auto obtenido exitosamente',
77       data: row
78     });
79   });
80 });
81
82 // 3. Crear un nuevo auto (POST /cars)
83 app.post('/cars', (req, res) => {
84   const { make, model, year, price } = req.body;
85   if (!make || !model || !year || !price) {
86     res.status(400).json({ error: 'Todos los campos (make, model, year, price) son requeridos.' });
87     return;
88   }
89   // Validaciones básicas de tipo de datos
90   if (typeof year !== 'number' || year < 1900 || year > new Date().getFullYear() + 1) {
91     res.status(400).json({ error: 'El año debe ser un número entre 1900 y el año actual más uno.' });
92     return;
93   }
94   db.run('INSERT INTO cars (make, model, year, price) VALUES (?, ?, ?, ?)', [make, model, year, price], (err) => {
95     if (err) {
96       res.status(500).json({ error: err.message });
97       return;
98     }
99     res.status(201).json({ message: 'Auto creado exitosamente' });
100   });
101 });
```

SE HACE LA CONSULTA HACIA LA TABLA CON LO AUTOS DEL CATALOGO



```
react_express_sqlite_autos > backend > JS server.js > ...
111 app.put('/cars/:id', (req, res) => {
112   // 4. Actualizar un auto (PUT /cars/:id)
113   const { id } = req.params;
114   const { make, model, year, price } = req.body;
115   const updates = [];
116   if (make) updates.push(`make='${make}'`);
117   if (model) updates.push(`model='${model}'`);
118   if (year) updates.push(`year='${year}'`);
119   if (price) updates.push(`price='${price}'`);
120   const sql = `UPDATE cars SET ${updates.join(', ')} WHERE id = ?`;
121   db.run(sql, [id], (err) => {
122     if (err) {
123       res.status(500).json({ error: err.message });
124       return;
125     }
126     if (this.changes === 0) {
127       res.status(404).json({ message: 'Auto no encontrado para actualizar' });
128       return;
129     }
130     res.json({
131       message: 'Auto actualizado exitosamente',
132       changes: this.changes
133     });
134   });
135 });
136
137 // 5. Eliminar un auto (DELETE /cars/:id)
138 app.delete('/cars/:id', (req, res) => {
139   const id = req.params.id;
140   db.run('DELETE FROM cars WHERE id = ?', [id], (err) => {
141     if (err) {
142       res.status(500).json({ error: err.message });
143       return;
144     }
145     if (this.changes === 0) {
146       res.status(404).json({ message: 'Auto no encontrado para eliminar' });
147       return;
148     }
149     res.status(200).json({ message: 'Auto eliminado exitosamente' });
150   });
151 });
```

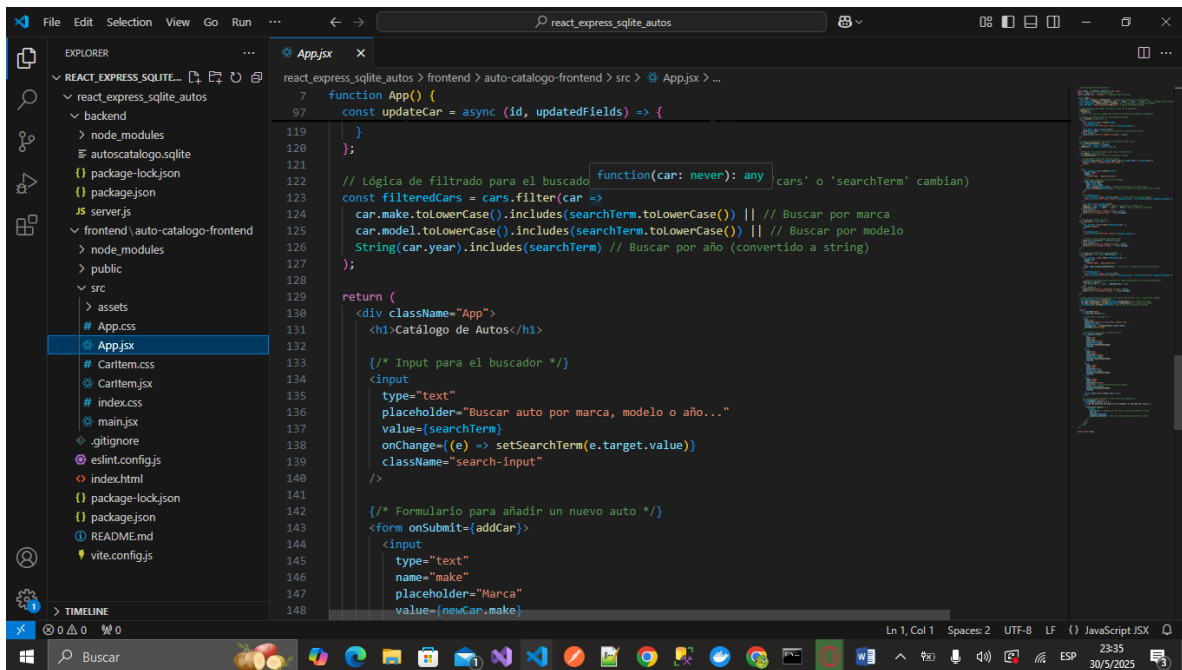
SE HACE EL UPDATE Y EL DELETE DE LOS AUTOS AGREGADOS AL CATALOGO DE AUTOS

```
1 // auto-catalogo-frontend/src/App.jsx
2
3 import React, { useState, useEffect } from 'react';
4 import './App.css'; // Estilos generales
5 import CarItem from './CarItem'; // Componente para cada auto
6
7 function App() {
8   const [cars, setCars] = useState([]); // Estado para almacenar la lista de autos
9   const [newCar, setNewCar] = useState({ make: '', model: '', year: '', price: '' }); // Estado para el formulario
10  const [searchTerm, setSearchTerm] = useState(''); // Estado para el término de búsqueda
11  const API_URL = 'http://localhost:3000/cars'; // URL de tu backend de autos
12
13  // Hook useEffect para cargar los autos al inicio de la aplicación
14  useEffect(() => {
15    fetchCars();
16  }, []); // El array vacío asegura que se ejecute solo una vez al montar el componente
17
18  // Función para obtener todos los autos desde el backend
19  const fetchCars = async () => {
20    try {
21      const response = await fetch(API_URL);
22      if (!response.ok) {
23        throw new Error('HTTP error! status: ${response.status}');
24      }
25      const data = await response.json();
26      setCars(data.data); // Actualiza el estado con los datos de los autos
27    } catch (error) {
28      console.error("Error al obtener los autos:", error);
29    }
30  };
31
32  // Maneja los cambios en los inputs del formulario "Añadir Auto"
```

ESPECIFICA LA RUTA Y EL PUERTO EN EL CUAL FUNCIONA EL DOMINIO DE LA PAGINA

```
7 function App() {
39   const addCar = async (e) => {
40     return;
41   };
42
43   try {
44     const response = await fetch(API_URL, {
45       method: 'POST',
46       headers: {
47         'Content-Type': 'application/json',
48       },
49       // Envía los datos del nuevo auto al backend
50       body: JSON.stringify({
51         make: newCar.make,
52         model: newCar.model,
53         year: parseInt(newCar.year), // Asegúrate de enviar el año como número
54         price: parseFloat(newCar.price) // Asegúrate de enviar el precio como número flotante
55       })),
56     );
57
58     if (!response.ok) {
59       const errorData = await response.json(); // Intenta leer el error del backend
60       throw new Error("HTTP error! status: ${response.status} - ${errorData.error || response.statusText}");
61     }
62
63     const data = await response.json();
64     setCars([...cars, data.data]); // Añade el nuevo auto al estado local
65     setNewCar({ make: '', model: '', year: '', price: '' }); // Limpia el formulario
66   } catch (error) {
67     console.error("Error al añadir el auto:", error);
68     alert("Error al añadir el auto: " + error.message);
69   }
70 }
71
72 // Maneja los cambios en los inputs del formulario "Añadir Auto"
```

LE MANDA LOS DATOS DEL FRONT AL BACK DEPENDIENDO DE LA CONSULTA QUE SE REQUIERA HACER



```
7 function App() {
97   const updateCar = async (id, updatedFields) => {
119   }
120 };
121
122 // Lógica de filtrado para el buscador function(car: never): any {cars' o 'searchTerm' cambian)
123 const filteredCars = cars.filter(car =>
124   car.make.toLowerCase().includes(searchTerm.toLowerCase()) || // Buscar por marca
125   car.model.toLowerCase().includes(searchTerm.toLowerCase()) || // Buscar por modelo
126   String(car.year).includes(searchTerm) // Buscar por año (convertido a string)
127 );
128
129 return (
130   <div className="App">
131     <h1>Catálogo de Autos</h1>
132
133     /* Input para el buscador */
134     <input
135       type="text"
136       placeholder="Buscar auto por marca, modelo o año..."
137       value={searchTerm}
138       onChange={(e) => setSearchTerm(e.target.value)}
139       className="search-input"
140     />
141
142     /* Formulario para añadir un nuevo auto */
143     <form onSubmit={addCar}>
144       <input
145         type="text"
146         name="make"
147         placeholder="Marca"
148         value={newCar.make}
```

Lógica de filtrado para el buscador