

Test Node JS

Eduardo Armando Nava Correa

Se incluyen en este documento los elementos de la solución a la prueba siguiente (cambiando PHP por NodeJS para la capa BackEnd):

Ejercicio:

Se requiere crear un sistema de control de inventario de automóviles en el cual únicamente se podrá mostrar y agregar vehículos al sistema. En este ejercicio solo se usarán dos tipos de vehículos (sedan y motocicletas), diferenciando número de llantas y potencia de motor. Con esto se busca categorizar el tipo de vehículo usando abstracción, herencia, etc.

Puntos:

- Repositorio del ejercicio.
- Añadir colección de APIs a postman.

PHP:

- Creación de APIs para consumo de frontend (framework indistinto).
- Aplicar principios SOLID (Usar clases, abstracción, herencia, interfaces, polimorfismo).
- Manejo de Auth para consumo de APIs.
- Inserción de datos a BD relacional.

Opcional:

- Uso de métodos mágicos.

JS:

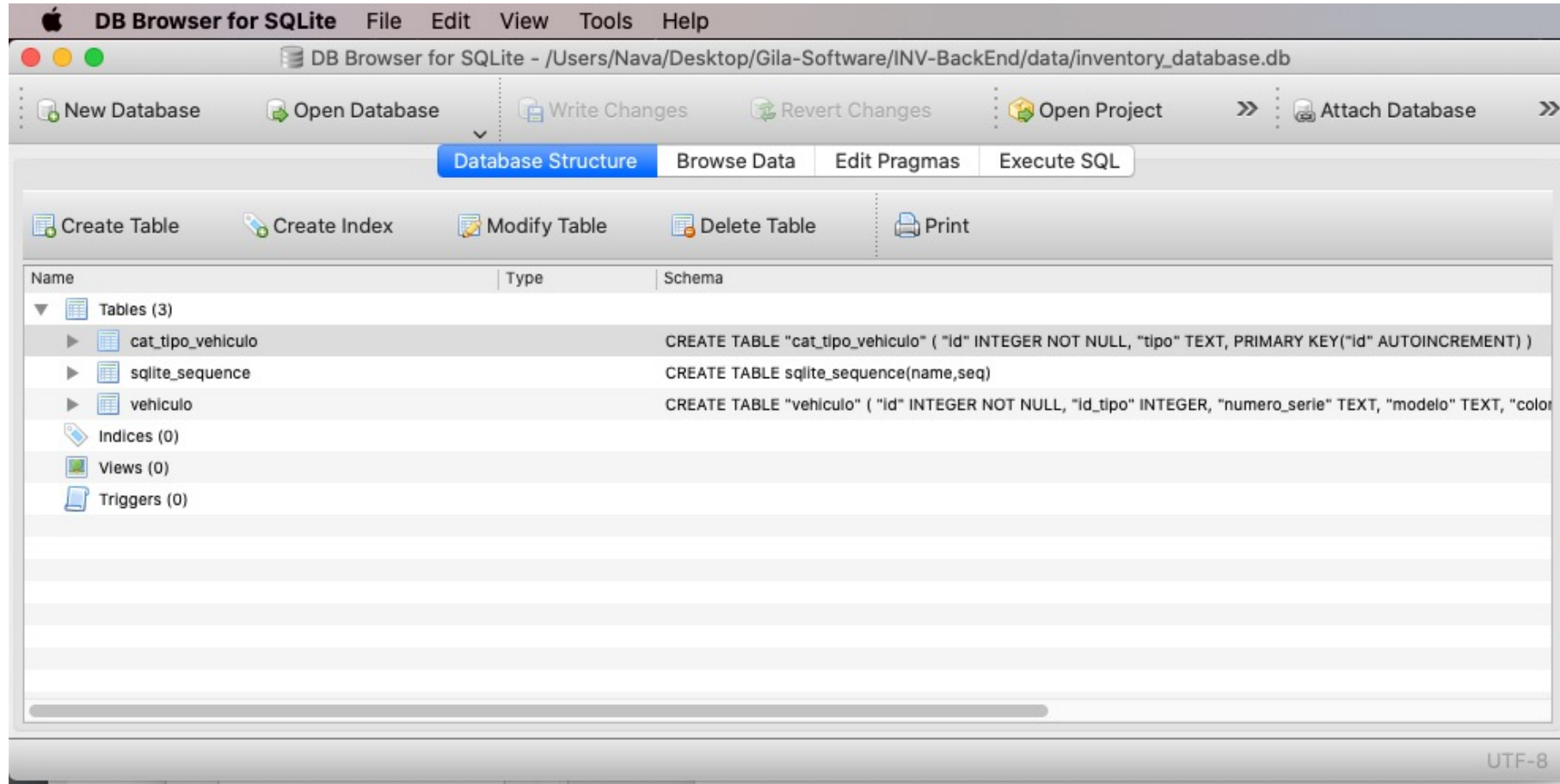
- Se puede añadir framework opcional.
- Login.
- Manejo de sesión (se puede usar framework o herramientas que lo ayuden al desarrollo).
- Funciones básicas para el uso de APIs (crear y mostrar registros).
- Uso de promesas, async, arrow functions.
- Manejo de clases.

Opcional:

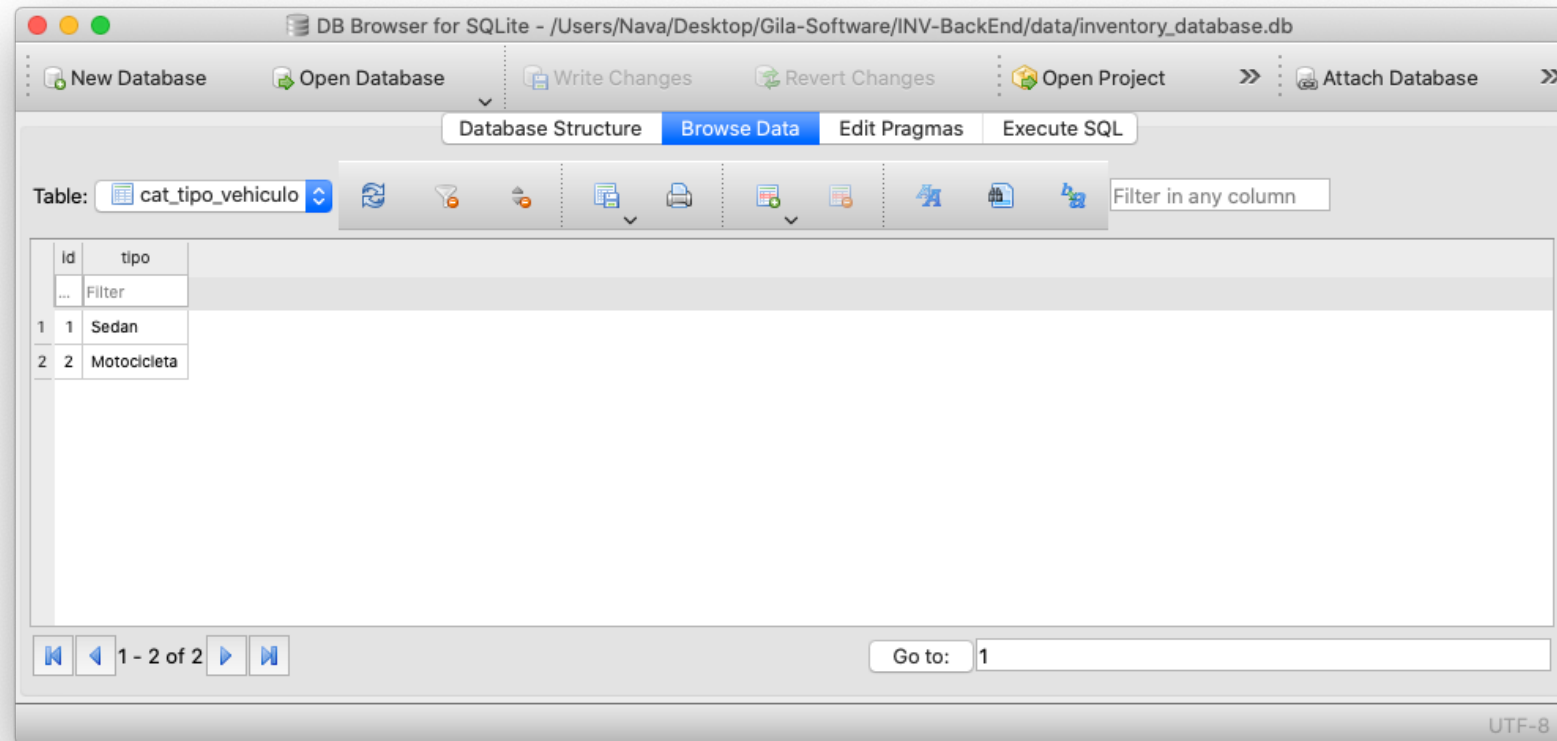
- Validaciones en inputs de formulario.

Nota: No es necesario concluir todos los puntos descritos, sin embargo, si se espera tener funcionalidades completas.

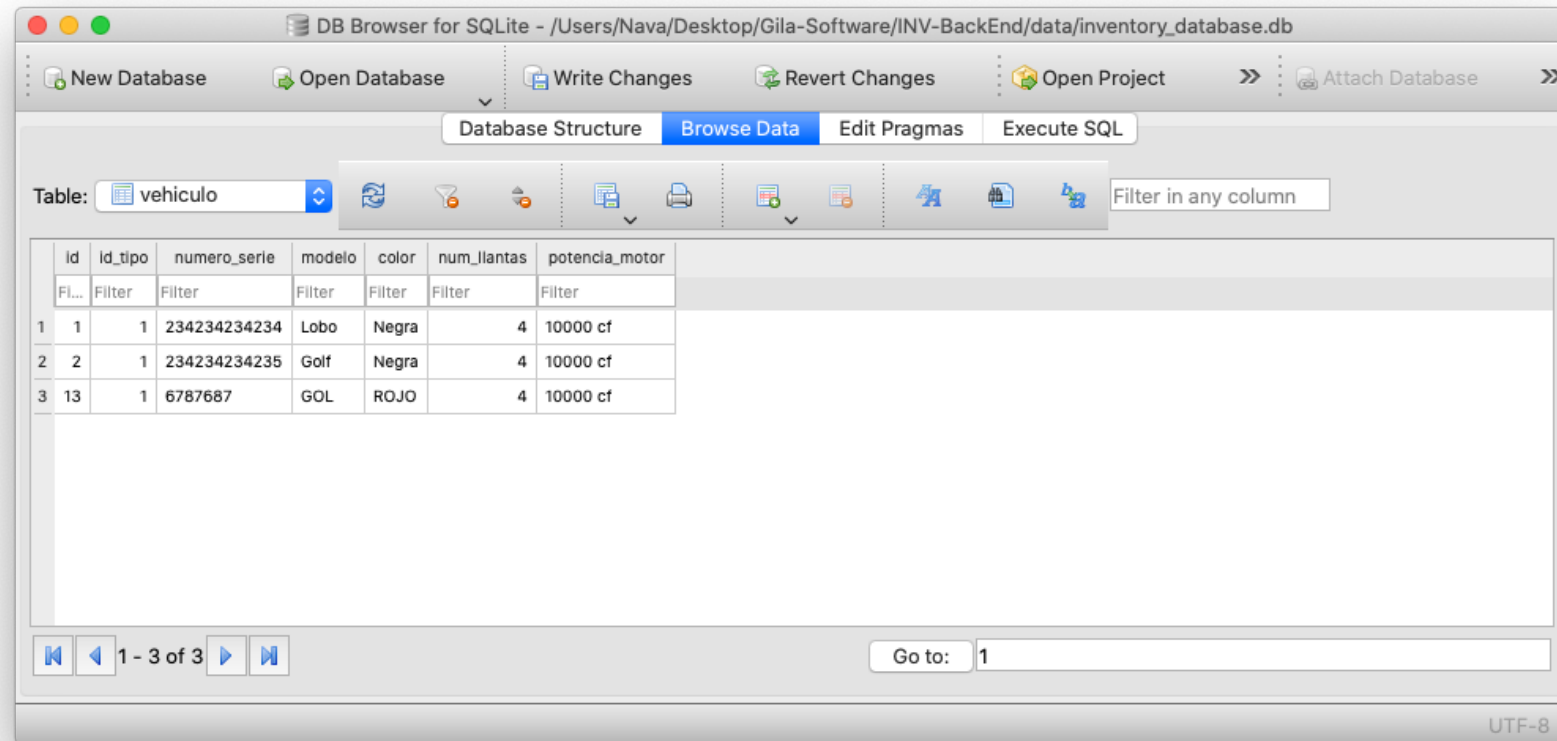
Se creo la base de datos en SQLite con 2 tablas principales.



La tabla cat_tipo_vehiculo se usa actualmente solo para la consulta, no para el llenado de el listado de selección en el formulario.



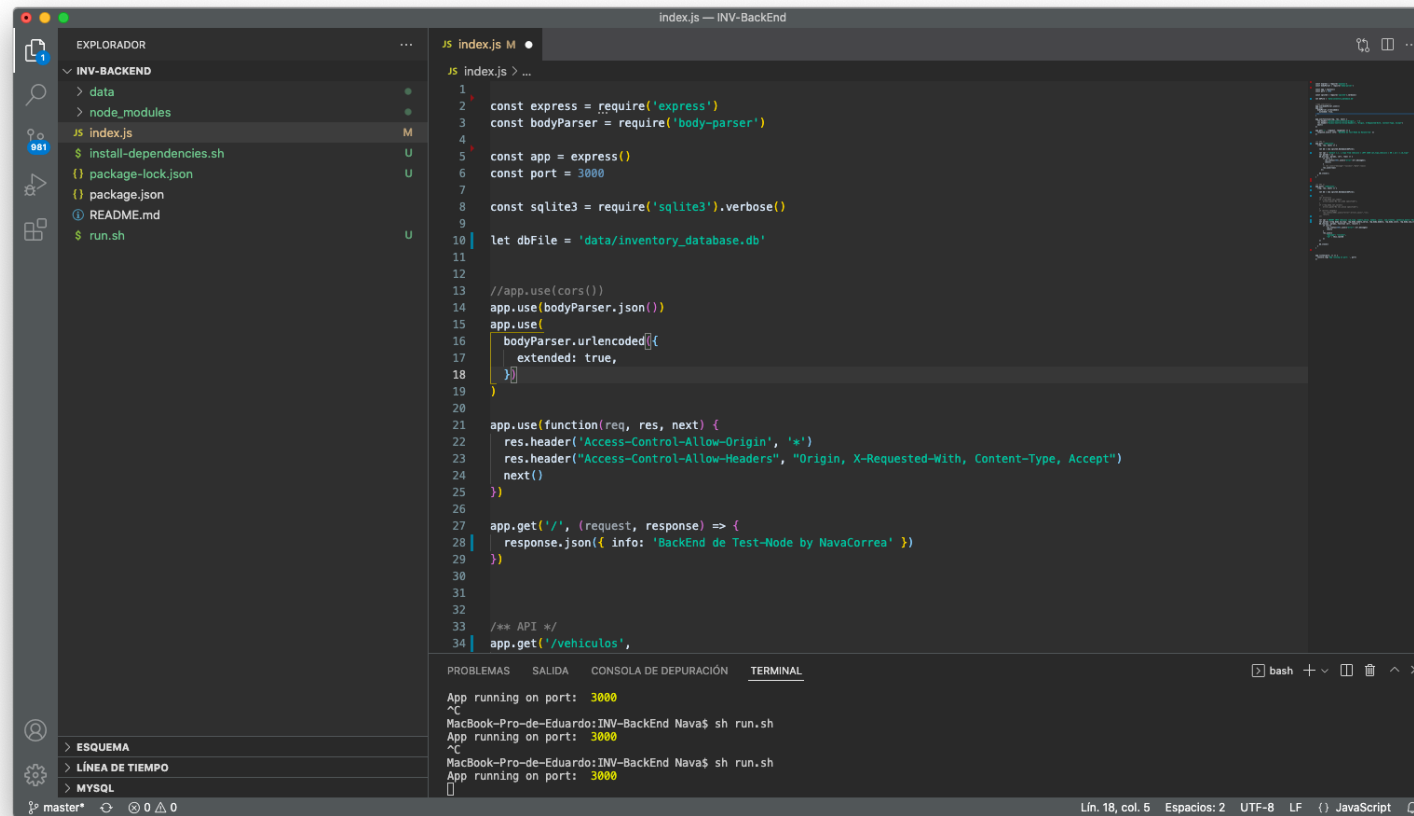
La tabla vehiculo contiene tanto Autos como Motocicletas.



The screenshot shows the 'DB Browser for SQLite' application window. The title bar indicates the database path: '/Users/Nava/Desktop/Gila-Software/INV-BackEnd/data/inventory_database.db'. The 'Browse Data' tab is active, displaying the 'vehiculo' table. The table has 7 columns: id, id_tipo, numero_serie, modelo, color, num_llantas, and potencia_motor. There are 3 rows of data. The bottom status bar shows '1 - 3 of 3' and 'Go to: 1'. The encoding is UTF-8.

	id	id_tipo	numero_serie	modelo	color	num_llantas	potencia_motor
	Fl...	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	234234234234	Lobo	Negra	4	10000 cf
2	2	1	234234234235	Golf	Negra	4	10000 cf
3	13	1	6787687	GOL	ROJO	4	10000 cf

Todo el código del Back End esta en un archivo index.js manteniendo la simplicidad del código para esta prueba de concepto. Muestra la referencia a la base de datos SQLite y la implementación de CORS.



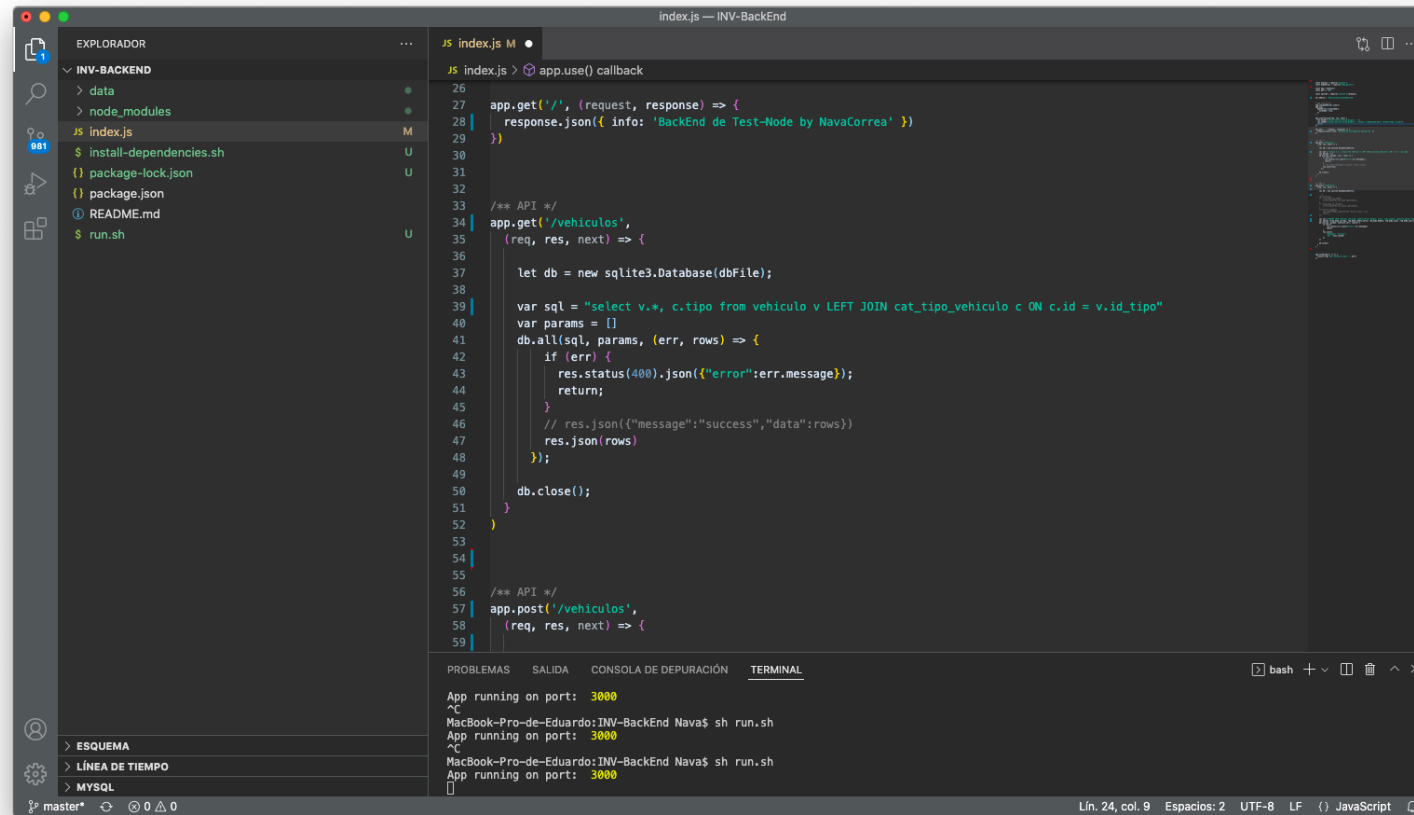
```
1
2 const express = require('express')
3 const bodyParser = require('body-parser')
4
5 const app = express()
6 const port = 3000
7
8 const sqlite3 = require('sqlite3').verbose()
9
10 let dbFile = 'data/inventory_database.db'
11
12
13 //app.use(cors())
14 app.use(bodyParser.json())
15 app.use(
16   bodyParser.urlencoded({
17     extended: true,
18   })
19 )
20
21 app.use(function(req, res, next) {
22   res.header('Access-Control-Allow-Origin', '*')
23   res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept")
24   next()
25 })
26
27 app.get('/', (request, response) => {
28   response.json({ info: 'BackEnd de Test-Node by NavaCorrea' })
29 })
30
31
32
33 /** API */
34 app.get('/vehiculos',
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
App running on port: 3000
^C
MacBook-Pro-de-Eduardo:INV-BackEnd Nava$ sh run.sh
App running on port: 3000
^C
MacBook-Pro-de-Eduardo:INV-BackEnd Nava$ sh run.sh
App running on port: 3000
```

Lín. 18, col. 5 Espacios: 2 UTF-8 LF () JavaScript

Código para el EndPoint con GET, para la consulta de vehiculos.



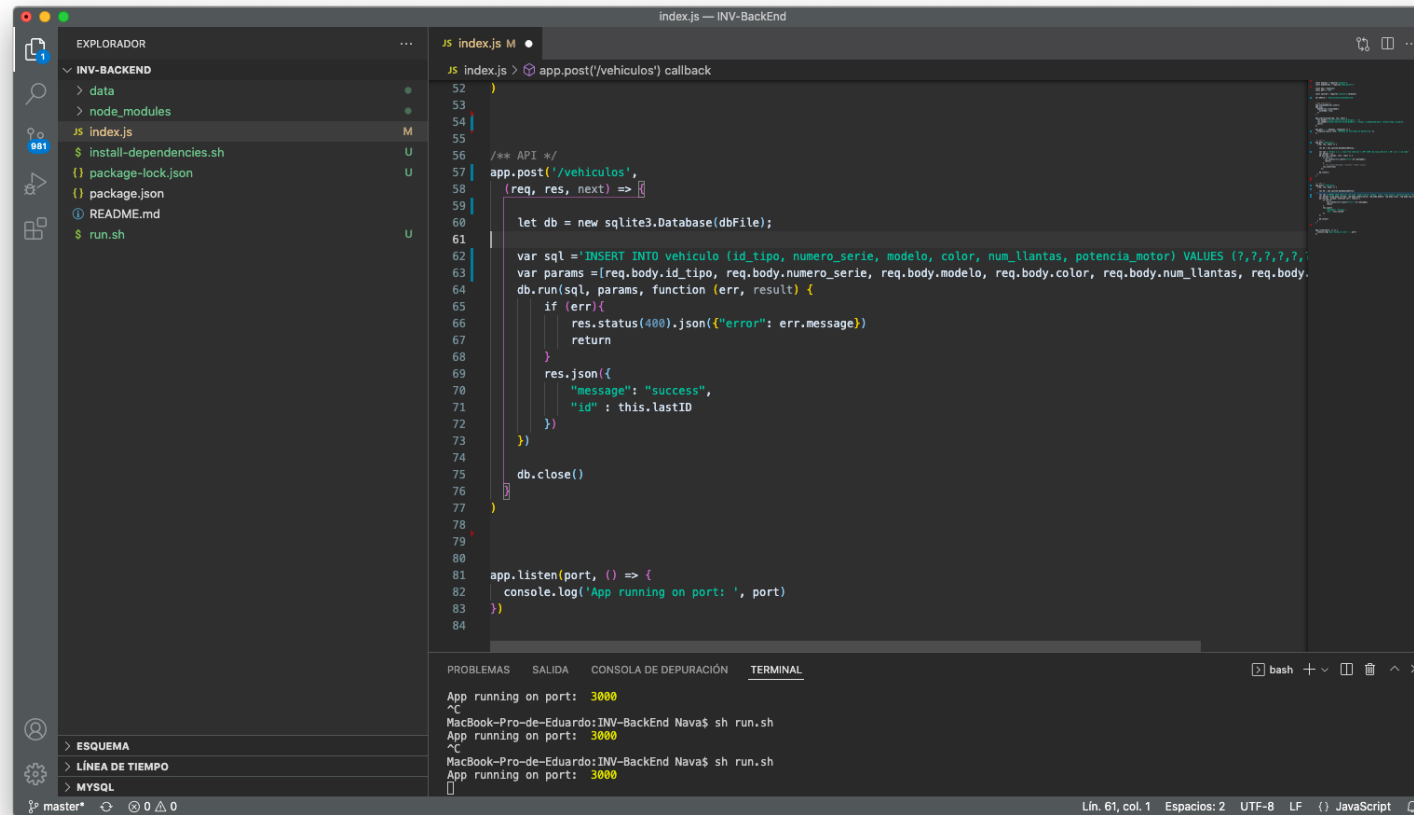
```
index.js — INV-BackEnd
JS index.js M
JS index.js > app.use() callback
26
27 app.get('/', (request, response) => {
28 |   response.json({ info: 'BackEnd de Test-Node by NavaCorrea' })
29 | })
30
31
32
33 /** API */
34 | app.get('/vehiculos',
35 | (req, res, next) => {
36 |
37 |   let db = new sqlite3.Database(dbFile);
38 |
39 |   var sql = "select v.*, c.tipo from vehiculo v LEFT JOIN cat_tipo_vehiculo c ON c.id = v.id_tipo"
40 |   var params = []
41 |   db.all(sql, params, (err, rows) => {
42 |     if (err) {
43 |       res.status(400).json({ "error": err.message });
44 |       return;
45 |     }
46 |     // res.json({ "message": "success", "data": rows })
47 |     res.json(rows)
48 |   });
49 |   db.close();
50 | }
51 | )
52 |
53 |
54 |
55 |
56 /** API */
57 | app.post('/vehiculos',
58 | (req, res, next) => {
59 |
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
App running on port: 3000
^C
MacBook-Pro-de-Eduardo:INV-BackEnd Nava$ sh run.sh
App running on port: 3000
^C
MacBook-Pro-de-Eduardo:INV-BackEnd Nava$ sh run.sh
App running on port: 3000
```

Lín. 24, col. 9 Espacios: 2 UTF-8 LF () JavaScript

Código para el EndPoint con POST, para el alta de vehiculos.



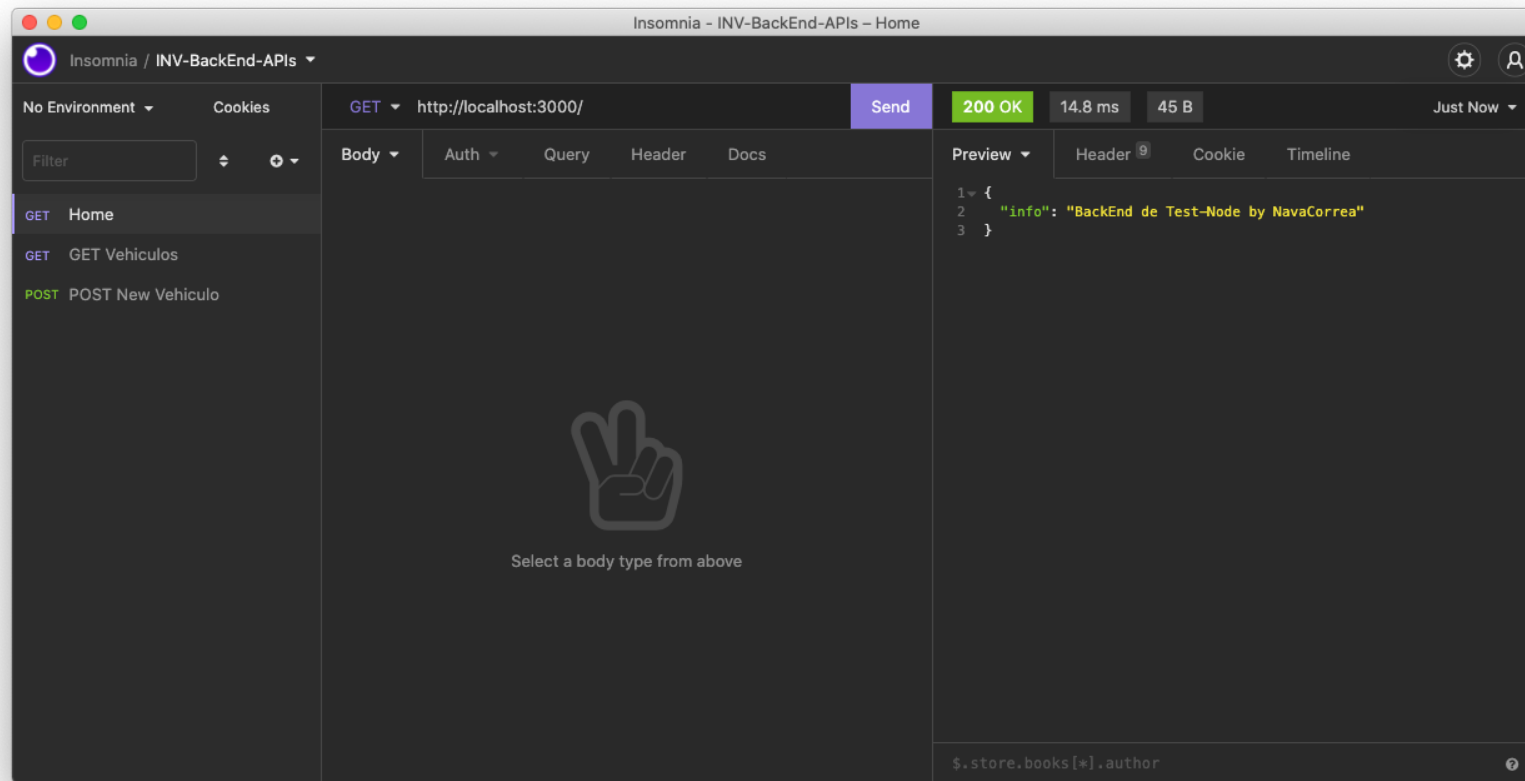
```
index.js — INV-BackEnd
JS index.js M
JS index.js > app.post('/vehiculos') callback
52
53
54
55
56 /** API */
57 app.post('/vehiculos',
58   (req, res, next) => {
59
60     let db = new sqlite3.Database(dbFile);
61
62     var sql = 'INSERT INTO vehiculo (id_tipo, numero_serie, modelo, color, num_llantas, potencia_motor) VALUES (7,7,7,7,7,7)';
63     var params = [req.body.id_tipo, req.body.numero_serie, req.body.modelo, req.body.color, req.body.num_llantas, req.body.potencia_motor];
64     db.run(sql, params, function (err, result) {
65       if (err) {
66         res.status(400).json({ "error": err.message });
67         return;
68       }
69       res.json({
70         "message": "success",
71         "id": this.lastID
72       });
73     });
74     db.close();
75   }
76 )
77
78
79
80
81 app.listen(port, () => {
82   console.log('App running on port: ', port)
83 })
84
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

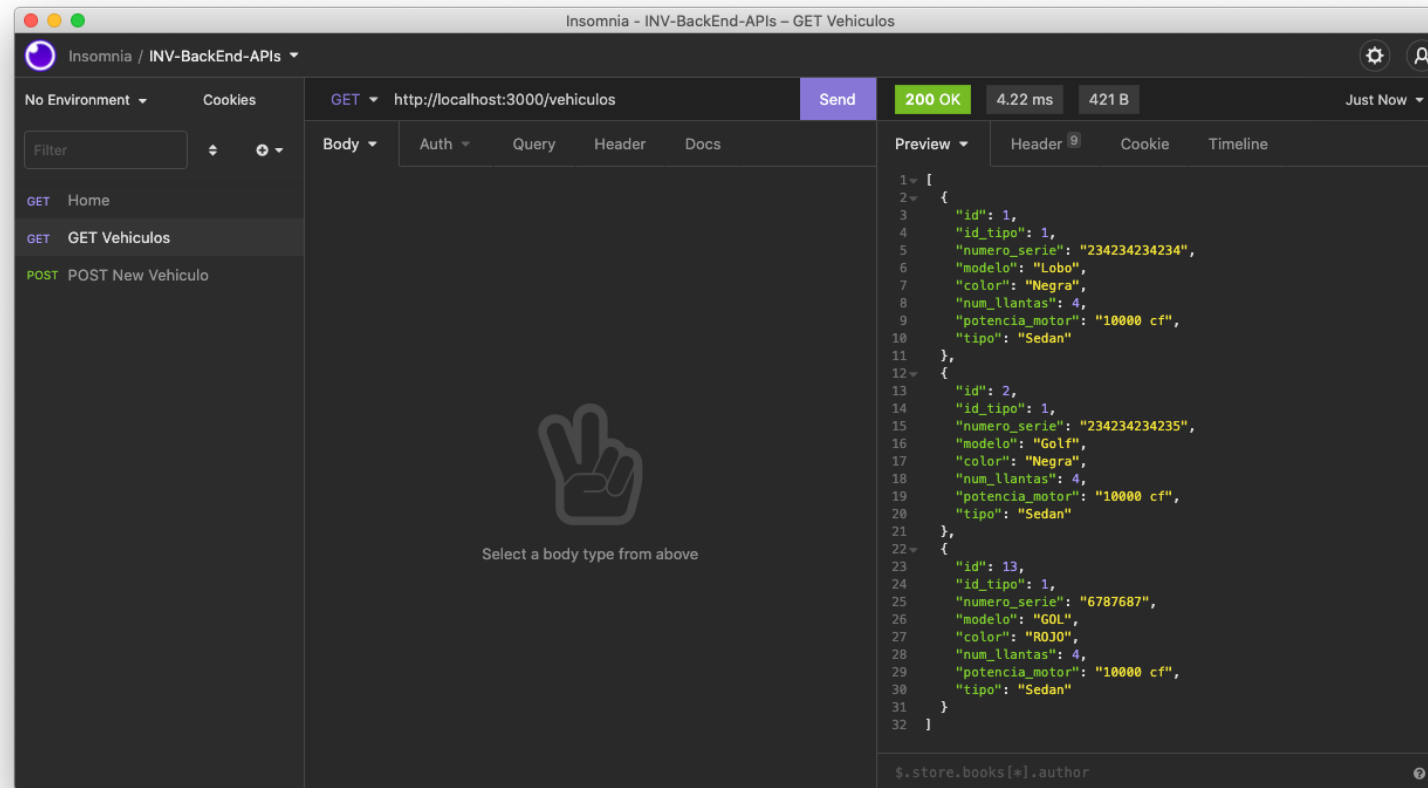
```
bash
App running on port: 3000
^C
MacBook-Pro-de-Eduardo:INV-BackEnd Nava$ sh run.sh
App running on port: 3000
^C
MacBook-Pro-de-Eduardo:INV-BackEnd Nava$ sh run.sh
App running on port: 3000
```

Lín. 61, col. 1 Espacios: 2 UTF-8 LF () JavaScript

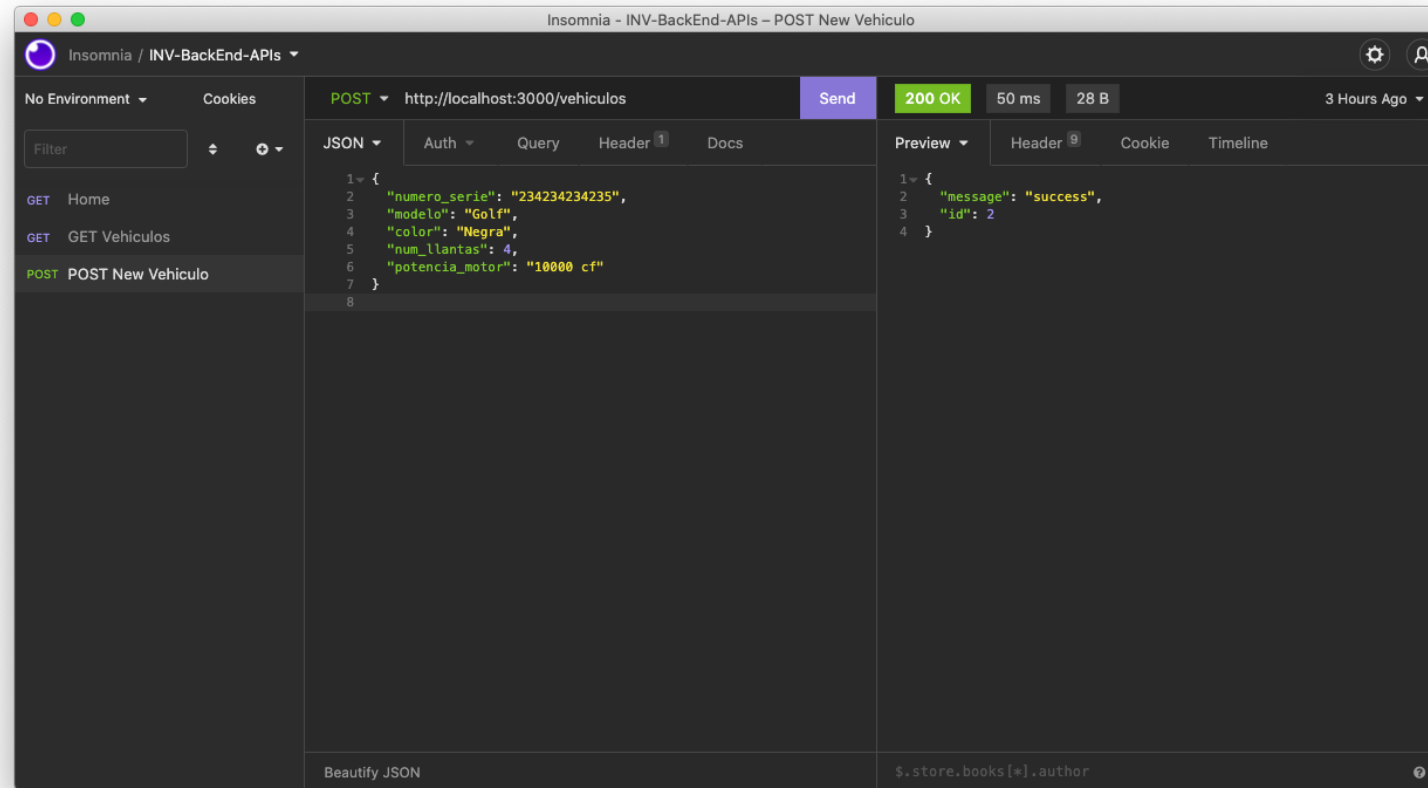
Prueba desde Insomnia del Endpoint (Home). Se incluye en el código la colección de requests en formato JSON (Postman/Insomnia).



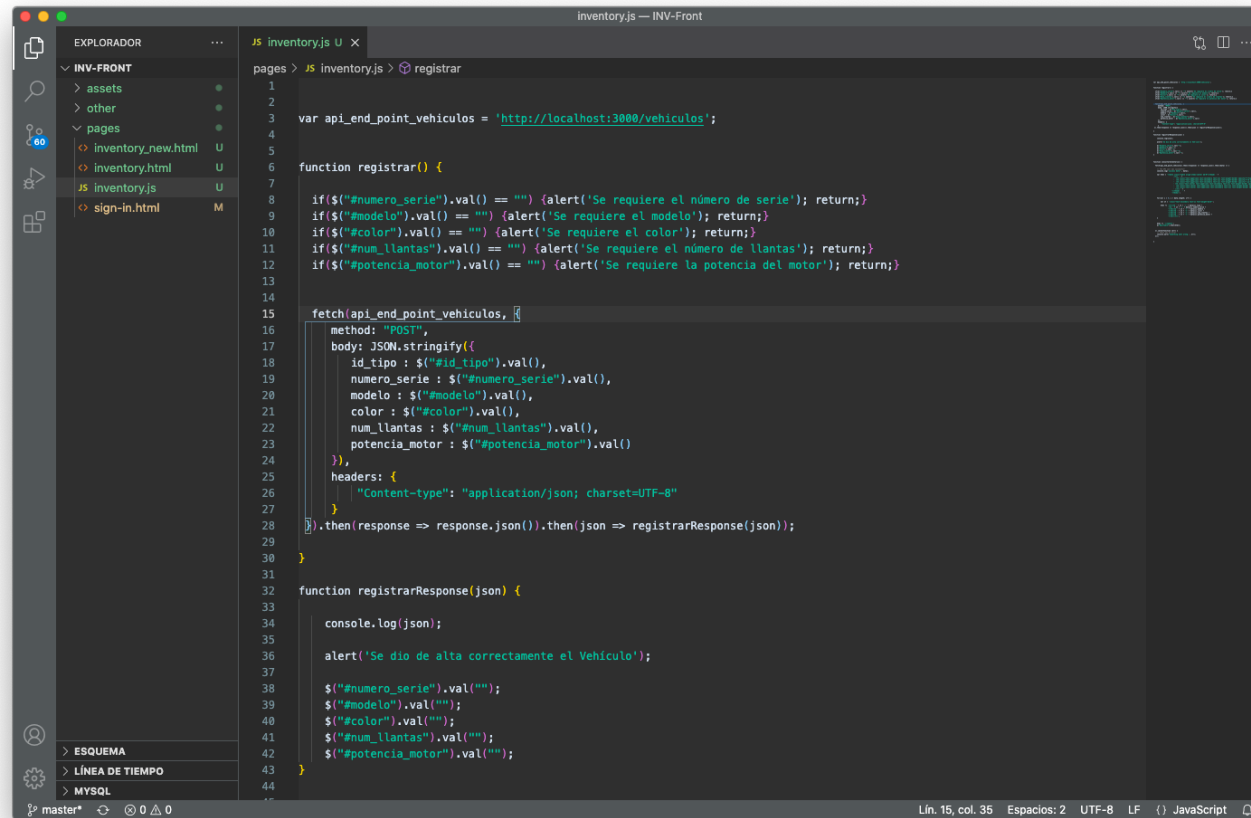
Prueba desde Insomnia del Endpoint (GET).



Prueba desde Insomnia del Endpoint (POST).

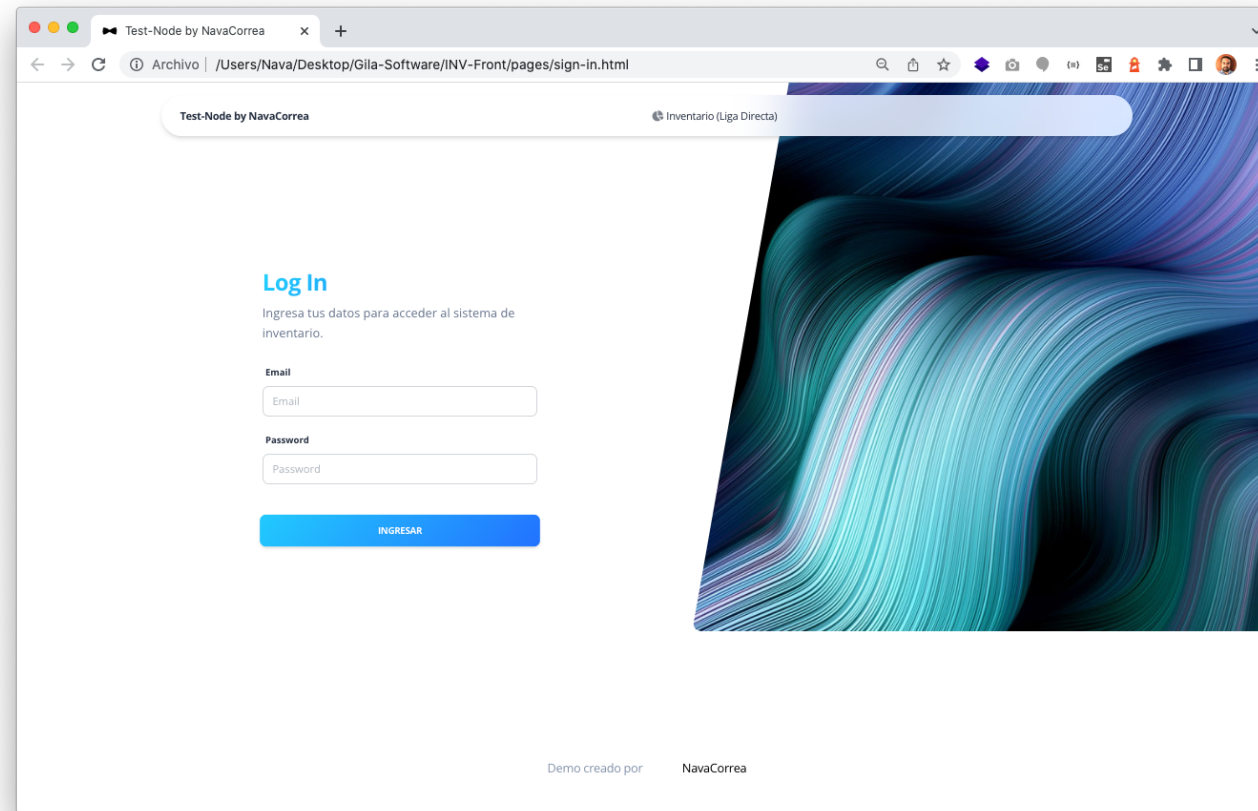


Código del Front End para la invocación de la API de alta de vehículos.

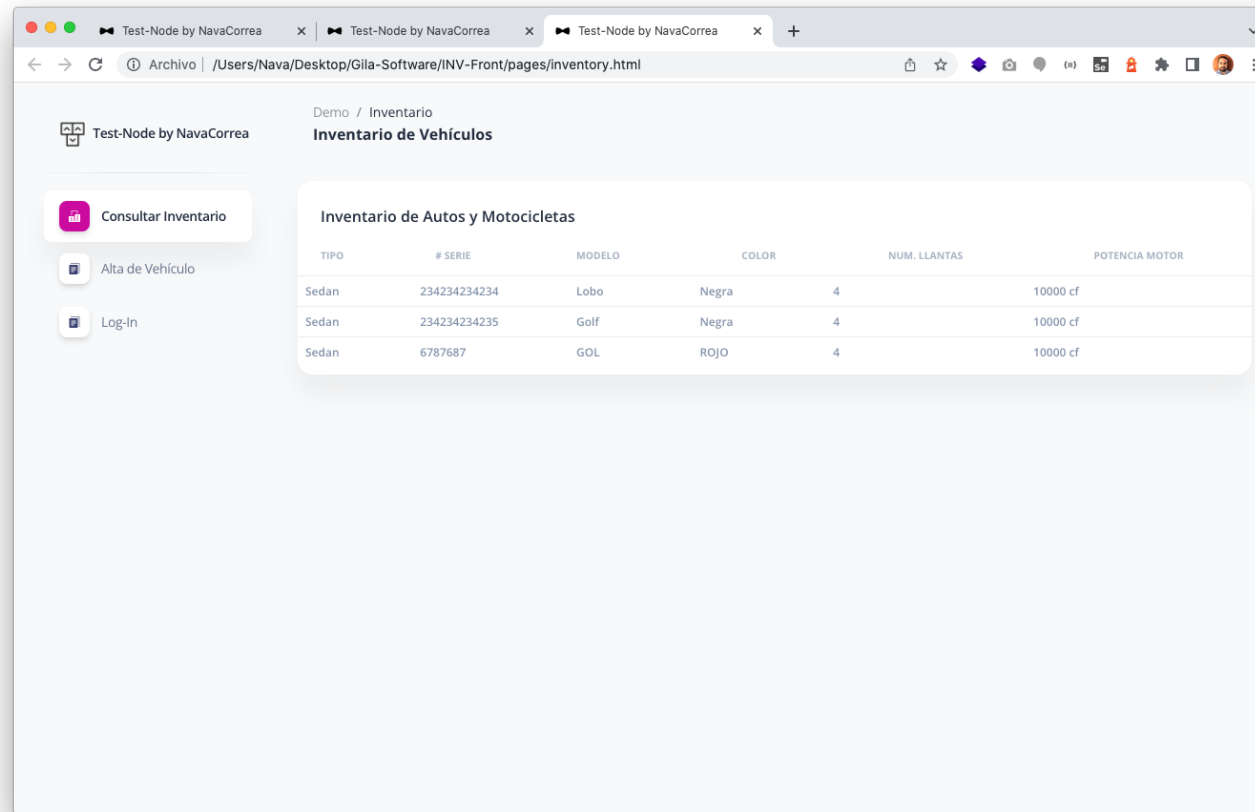


```
inventory.js — INV-Front
EXPLORADOR
INV-FRONT
  assets
  other
  pages
    inventory_new.html U
    inventory.html U
    JS inventory.js U
    sign-in.html M
  JS inventory.js X
pages > JS inventory.js > registrar
1
2
3 var api_end_point_vehiculos = 'http://localhost:3000/vehiculos';
4
5
6 function registrar() {
7
8   if($("#numero_serie").val() == "") {alert('Se requiere el número de serie'); return;}
9   if($("#modelo").val() == "") {alert('Se requiere el modelo'); return;}
10  if($("#color").val() == "") {alert('Se requiere el color'); return;}
11  if($("#num_llantas").val() == "") {alert('Se requiere el número de llantas'); return;}
12  if($("#potencia_motor").val() == "") {alert('Se requiere la potencia del motor'); return;}
13
14
15  fetch(api_end_point_vehiculos, {
16    method: "POST",
17    body: JSON.stringify({
18      id_tipo : ($("#id_tipo").val()),
19      numero_serie : ($("#numero_serie").val()),
20      modelo : ($("#modelo").val()),
21      color : ($("#color").val()),
22      num_llantas : ($("#num_llantas").val()),
23      potencia_motor : ($("#potencia_motor").val())
24    }),
25    headers: {
26      "Content-type": "application/json; charset=UTF-8"
27    }
28  }).then(response => response.json()).then(json => registrarResponse(json));
29
30 }
31
32 function registrarResponse(json) {
33
34   console.log(json);
35
36   alert('Se dio de alta correctamente el Vehículo');
37
38   $("#numero_serie").val("");
39   $("#modelo").val("");
40   $("#color").val("");
41   $("#num_llantas").val("");
42   $("#potencia_motor").val("");
43
44 }
```

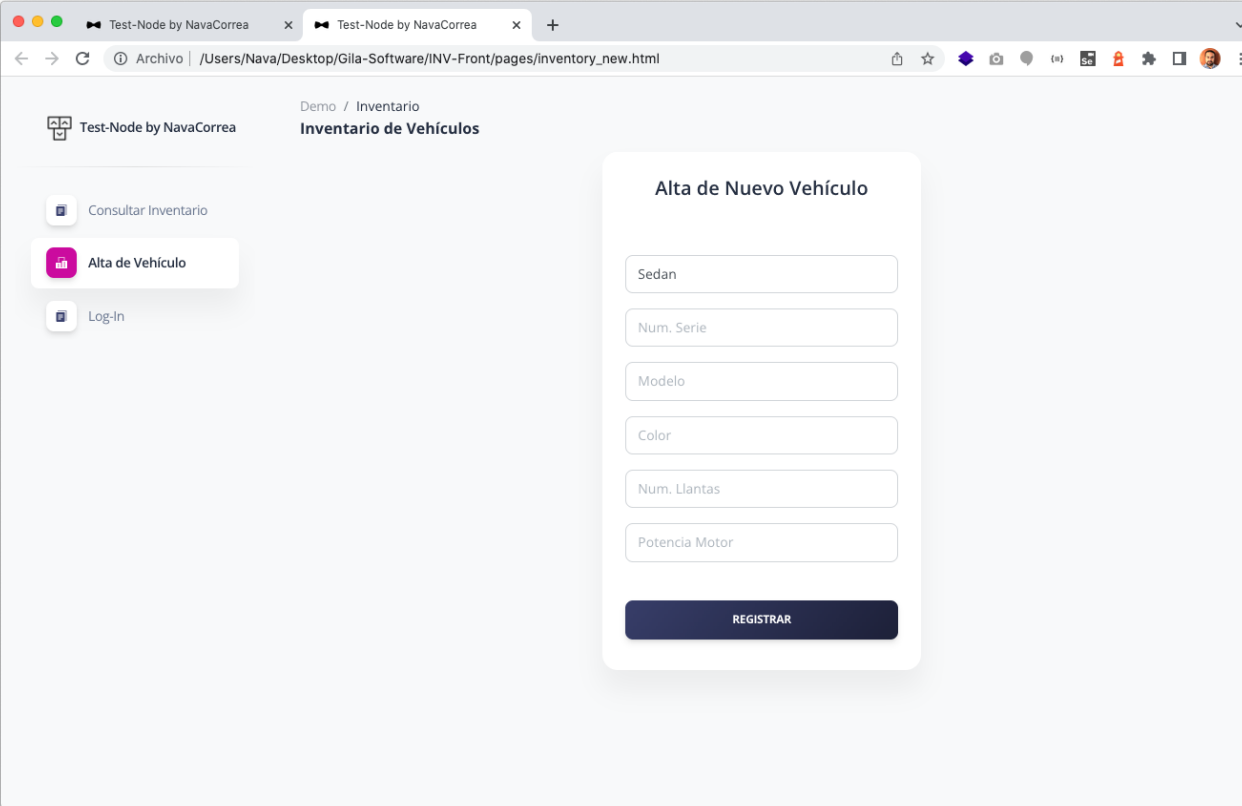

Interfaz gráfica del Login (solo maquetado, no tiene funcionalidad con API en esta versión).



Interfaz gráfica de la Consulta del Inventario.



Formulario de Alta.

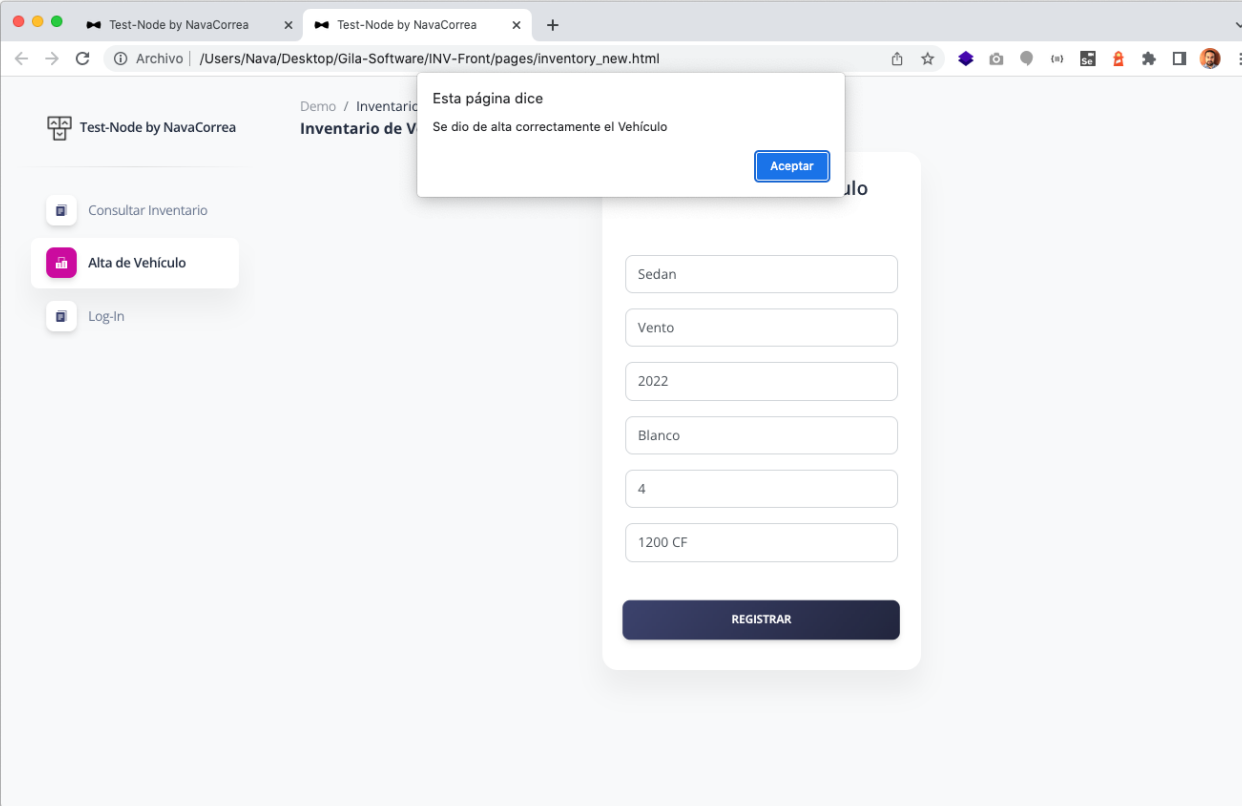


The screenshot displays a web browser window with two tabs, both labeled "Test-Node by NavaCorrea". The address bar shows the file path: `/Users/Nava/Desktop/Gila-Software/INV-Front/pages/inventory_new.html`. The page header includes a logo for "Test-Node by NavaCorrea" and a breadcrumb trail: "Demo / Inventario" followed by "Inventario de Vehículos".

On the left side, there is a vertical menu with three items: "Consultar Inventario", "Alta de Vehículo" (highlighted with a pink background), and "Log-In".

The main content area features a white card titled "Alta de Nuevo Vehículo". This card contains six text input fields stacked vertically, each with a placeholder label: "Sedan", "Num. Serie", "Modelo", "Color", "Num. Llantas", and "Potencia Motor". Below these fields is a dark blue button with the text "REGISTRAR" in white capital letters.

Prueba de Alta desde el Front End (UI).



The screenshot shows a web browser window with two tabs, both titled "Test-Node by NavaCorrea". The address bar displays the file path: `/Users/Nava/Desktop/Gila-Software/INV-Front/pages/inventory_new.html`. The page content includes a sidebar on the left with the logo "Test-Node by NavaCorrea" and three menu items: "Consultar Inventario", "Alta de Vehículo" (highlighted with a pink background), and "Log-In". The main area shows a breadcrumb "Demo / Inventario" and a heading "Inventario de V". A modal form for vehicle registration is displayed, containing the following fields: "Sedan", "Vento", "2022", "Blanco", "4", and "1200 CF". A dark blue button labeled "REGISTRAR" is at the bottom of the form. A white alert box with a blue border is overlaid on the form, containing the text "Esta página dice" and "Se dio de alta correctamente el Vehículo", with a blue "Aceptar" button.

Test-Node by NavaCorrea

Demo / Inventario

Inventario de V

Consultar Inventario

Alta de Vehículo

Log-In

Esta página dice

Se dio de alta correctamente el Vehículo

Aceptar

Sedan

Vento

2022

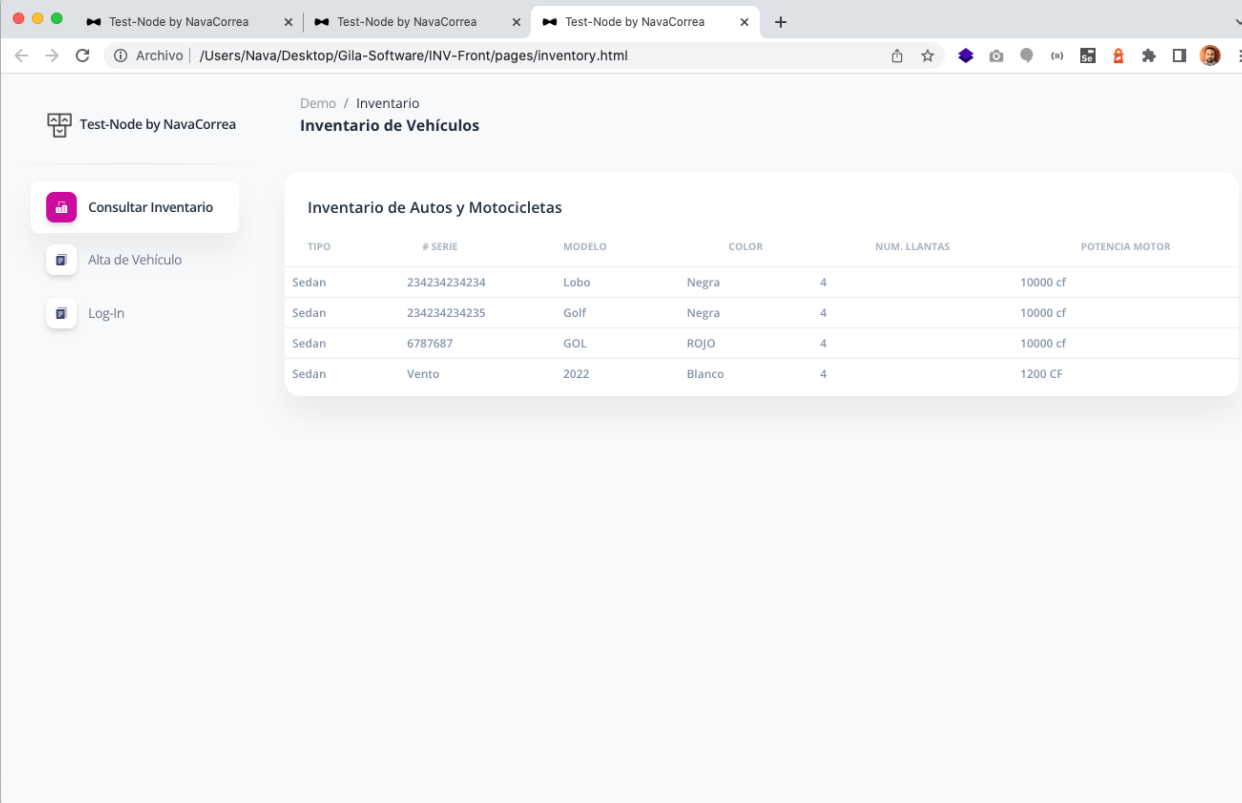
Blanco

4

1200 CF

REGISTRAR

Inventario Actualizado



The screenshot shows a web browser window with three tabs, all labeled 'Test-Node by NavaCorrea'. The address bar shows the URL: /Users/Nava/Desktop/Gila-Software/INV-Front/pages/inventory.html. The page content includes a sidebar with three buttons: 'Consultar Inventario' (highlighted in pink), 'Alta de Vehículo', and 'Log-In'. The main area displays a table titled 'Inventario de Autos y Motocicletas' with the following data:

TIPO	# SERIE	MODELO	COLOR	NUM. LLANTAS	POTENCIA MOTOR
Sedan	234234234234	Lobo	Negra	4	10000 cf
Sedan	234234234235	Golf	Negra	4	10000 cf
Sedan	6787687	GOL	ROJO	4	10000 cf
Sedan	Vento	2022	Blanco	4	1200 CF

Gracias

<https://datidev.com/Nava/CV-Armando-Nava-2022-es.pdf>