# Estructura App Parte Diario

Parte Diario

-- assets

-----EstructuraApp.docx

-----images

--------spf\_favicon.png

-- css

---- styles.css

-- js

---- app.js

-- node\_modules

-- .env

-- .gitignore

-- index.html

-- package.json

-- server.js

|  |
| --- |
| styles.css  body {  font-family: Arial, sans-serif;  background-color: #f8f9fa;  margin: 0;  padding: 0;  }  .card-header.bg-primary {  background-color: #007bff !important;  }  .info {  font-size: 14px;  color: #6c757d;  }  .pagination button {  cursor: pointer;  }  #loader img {  width: 50px;  } |

|  |
| --- |
| app.js  const apiUrl = 'http://localhost:5000/api/agentes';  let agentesData = [];  let currentPage = 1;  let rowsPerPage = 5;  let totalPages = 0;  document.addEventListener('DOMContentLoaded', () => {  cargarDatos();  });  async function cargarDatos() {  const searchTerm = document.getElementById('search').value.trim(); // Obtener el término de búsqueda  mostrarCargando(true);  try {  const response = await fetch(`${apiUrl}?search=${encodeURIComponent(searchTerm)}`);  const data = await response.json();  agentesData = data; // Guardar los datos recibidos de la API  currentPage = 1; // Reiniciar la paginación con cada nueva búsqueda  renderizarTabla();  } catch (error) {  console.error('Error al cargar los datos:', error);  alert('Error al cargar los datos');  } finally {  mostrarCargando(false);  }  }  function renderizarTabla() {  const tbody = document.querySelector('#agentesTableBody');  const paginationDiv = document.getElementById('pageButtons');  const infoDiv = document.getElementById('info');  totalPages = Math.ceil(agentesData.length / rowsPerPage);  const start = (currentPage - 1) \* rowsPerPage;  const end = currentPage \* rowsPerPage;  const paginatedData = agentesData.slice(start, end);  tbody.innerHTML = '';  paginatedData.forEach(agente => {  const row = document.createElement('tr');  row.innerHTML = `  <td>${agente.id}</td>  <td>${agente.grado || ''}</td>  <td>${agente.nombre || ''}</td>  <td>${agente.credencial || ''}</td>  <td>${agente.cuil || ''}</td>  <td>${agente.sector || ''}</td>  <td>  <button onclick="editarAgentes(${agente.id})" class="btn btn-sm btn-warning">Editar</button>  <button onclick="eliminarAgentes(${agente.id})" class="btn btn-sm btn-danger">Eliminar</button>  </td>  `;  tbody.appendChild(row);  });  actualizarPaginacion();  }  function actualizarPaginacion() {  const paginationDiv = document.getElementById('pageButtons');  paginationDiv.innerHTML = '';  for (let i = 1; i <= totalPages; i++) {  const button = document.createElement('button');  button.textContent = i;  button.onclick = () => cambiarPagina(i);  button.className = 'btn btn-outline-primary rounded-pill me-2';  if (i === currentPage) button.classList.add('active');  paginationDiv.appendChild(button);  }  }  function cambiarPagina(page) {  if (page >= 1 && page <= totalPages) {  currentPage = page;  renderizarTabla();  }  }  function cambiarRegistrosPorPagina() {  rowsPerPage = parseInt(document.getElementById('rowsPerPage').value);  currentPage = 1;  renderizarTabla();  }  function mostrarCargando(mostrar) {  const loader = document.getElementById('loader');  if (loader) {  loader.style.display = mostrar ? 'block' : 'none';  }  }  document.getElementById('agentesForm').addEventListener('submit', async (e) => {  e.preventDefault();  if (!validarFormulario()) return;  const id = document.getElementById('id').value;  const grado = document.getElementById('grado').value;  const nombre = document.getElementById('nombre').value;  const credencial = document.getElementById('credencial').value;  const cuil = document.getElementById('cuil').value.replace(/-/g, ''); // Eliminar guiones  const sector = document.getElementById('sector').value;  const method = id ? 'PUT' : 'POST';  const url = id ? `${apiUrl}/${id}` : apiUrl;  try {  const response = await fetch(url, {  method,  headers: { 'Content-Type': 'application/json' },  body: JSON.stringify({ grado, nombre, credencial, cuil, sector })  });  if (response.ok) {  limpiarFormulario();  cargarDatos();  } else {  const errorData = await response.json();  alert(`Error: ${errorData.error}`);  }  } catch (error) {  console.error(error);  alert('Error al comunicarse con el servidor');  }  });  async function editarAgentes(id) {  try {  const response = await fetch(`${apiUrl}/${id}`);  if (!response.ok) throw new Error(`Error al obtener el registro con ID ${id}`);  const agente = await response.json();    document.getElementById('id').value = agente.id || '';  document.getElementById('grado').value = agente.grado || '';  document.getElementById('nombre').value = agente.nombre || '';  document.getElementById('credencial').value = agente.credencial || '';  document.getElementById('cuil').value = agente.cuil || '';  document.getElementById('sector').value = agente.sector || '';  } catch (error) {  console.error(error);  alert(`Error: ${error.message}`);  }  }  async function eliminarAgentes(id) {  if (confirm('¿Estás seguro de eliminar este registro?')) {  const response = await fetch(`${apiUrl}/${id}`, { method: 'DELETE' });  if (response.ok) {  cargarDatos();  } else {  alert('Error al eliminar el registro');  }  }  }  function limpiarFormulario() {  document.getElementById('id').value = '';  document.getElementById('grado').value = '';  document.getElementById('nombre').value = '';  document.getElementById('credencial').value = '';  document.getElementById('cuil').value = '';  document.getElementById('sector').value = '';  }  function validarFormulario() {  const grado = document.getElementById('grado').value.trim();  const nombre = document.getElementById('nombre').value.trim();  const credencial = document.getElementById('credencial').value.trim();  const cuil = document.getElementById('cuil').value.trim();  const sector = document.getElementById('sector').value.trim();  if (!grado || !nombre || !credencial || !cuil || !sector) {  alert('Todos los campos son obligatorios.');  return false;  }  return true;  } |

|  |
| --- |
| .env  DB\_HOST=bjam2g6lhokq9u5ev092-mysql.services.clever-cloud.com  DB\_PORT=3306  DB\_USER=ua5xfv4rmu6drlfe  DB\_PASSWORD=ZqNpH4dsu5UPlADJH6Fd  DB\_NAME=bjam2g6lhokq9u5ev092  PORT=5000 |

|  |
| --- |
| index.html  <!DOCTYPE html>  <html lang="es">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>CRUD Agentes</title>  <!-- Bootstrap CSS -->  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">  <!-- Estilos personalizados -->  <link rel="stylesheet" href="css/styles.css">  </head>  <body>  <div class="container-fluid">  <div class="row mt-4">  <!-- Columna izquierda: Formulario -->  <div class="col-md-4">  <div class="card shadow">  <div class="card-header bg-primary text-white">  <h5 class="mb-0">Formulario de Agentes</h5>  </div>  <div class="card-body">  <form id="agentesForm">  <input type="hidden" id="id" />  <div class="mb-3">  <label for="grado" class="form-label">Grado:</label>  <input type="text" id="grado" class="form-control" required />  </div>  <div class="mb-3">  <label for="nombre" class="form-label">Nombre y Apellido:</label>  <input type="text" id="nombre" class="form-control" required />  </div>  <div class="mb-3">  <label for="credencial" class="form-label">Credencial:</label>  <input type="text" id="credencial" class="form-control" required />  </div>  <div class="mb-3">  <label for="cuil" class="form-label">Cuil:</label>  <input type="text" id="cuil" class="form-control" required />  </div>  <div class="mb-3">  <label for="sector" class="form-label">Sector:</label>  <input type="text" id="sector" class="form-control" required />  </div>  <div class="d-grid gap-2">  <button type="submit" class="btn btn-success rounded-pill">Guardar</button>  <button type="button" onclick="limpiarFormulario()" class="btn btn-secondary rounded-pill">Cancelar</button>  </div>  </form>  </div>  </div>  </div>  <!-- Columna derecha: Búsqueda, Tabla y Paginación -->  <div class="col-md-8">  <div class="card shadow">  <div class="card-header bg-primary text-white">  <h5 class="mb-0">Lista de Agentes</h5>  </div>  <div class="card-body">  <!-- Campo de búsqueda -->  <div class="mb-3">  <label for="search" class="form-label">Buscar:</label>  <!--<input type="text" id="search" class="form-control" placeholder="Escribe para buscar..." oninput="renderizarTabla()" />-->  <input type="text" id="search" class="form-control" placeholder="Escribe para buscar..." oninput="cargarDatos()" />  </div>  <!-- Selector de registros por página -->  <div class="mb-3">  <label for="rowsPerPage" class="form-label">Registros por página:</label>  <select id="rowsPerPage" class="form-select" onchange="cambiarRegistrosPorPagina()">  <option value="5">5</option>  <option value="10">10</option>  <option value="20">20</option>  <option value="50">50</option>  </select>  </div>  <!-- Tabla de datos -->  <table class="table table-striped table-hover">  <thead>  <tr>  <th>ID</th>  <th>Grado</th>  <th>Nombre</th>  <th>Credencial</th>  <th>Cuil</th>  <th>Sector</th>  <th>Acciones</th>  </tr>  </thead>  <tbody id="agentesTableBody">  <!-- Datos cargados dinámicamente -->  </tbody>  </table>  <!-- Información de registros -->  <div class="info mb-3" id="info"></div>  <!-- Paginación -->  <div class="pagination d-flex justify-content-center" id="pagination">  <button class="btn btn-outline-primary rounded-pill me-2" onclick="irAPrimeraPagina()">« Primero</button>  <button class="btn btn-outline-primary rounded-pill me-2" onclick="irAPaginaAnterior()">‹ Anterior</button>  <span id="pageButtons"></span>  <button class="btn btn-outline-primary rounded-pill me-2" onclick="irAPaginaSiguiente()">Siguiente ›</button>  <button class="btn btn-outline-primary rounded-pill" onclick="irAUltimaPagina()">Último »</button>  </div>  </div>  </div>  </div>  </div>  </div>  <!-- Loader -->  <div id="loader" style="display: none; text-align: center; margin-top: 20px;">  <img src="assets/loading.gif" alt="Cargando...">  </div>  <!-- Bootstrap JS y Popper.js -->  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"></script>  <!-- Lógica JavaScript -->  <script src="js/app.js"></script>  </body>  </html> |

|  |
| --- |
| package.json  {  "name": "partediario",  "version": "1.0.0",  "main": "server.js",  "scripts": {  "start": "node server.js"  },  "keywords": [],  "author": "",  "license": "ISC",  "dependencies": {  "body-parser": "^1.20.3",  "cors": "^2.8.5",  "dotenv": "^16.4.7",  "express": "^4.21.2",  "mysql2": "^3.12.0"  },  "description": ""  } |

|  |
| --- |
| server.js  require('dotenv').config(); // Carga las variables de entorno desde .env  const express = require('express');  const mysql = require('mysql2/promise');  const cors = require('cors');  const app = express();  app.use(cors());  app.use(express.json());  // Configuración de la conexión a MySQL  const db = mysql.createPool({  host: process.env.DB\_HOST,  port: process.env.DB\_PORT,  user: process.env.DB\_USER,  password: process.env.DB\_PASSWORD,  database: process.env.DB\_NAME,  waitForConnections: true, // Añade esta opción para manejar conexiones simultáneas  connectionLimit: 10,  queueLimit: 0  });  db.getConnection()  .then(() => {  console.log('Conexión exitosa a la base de datos');  })  .catch(err => {  console.error('Error al conectar a la base de datos:', err);  process.exit(1); // Detener el servidor si no se puede conectar  });  // Validación de datos  function validateAgentData(data) {  const { grado, nombre, credencial, cuil, sector } = data;  if (!grado || !nombre || !credencial || !cuil || !sector) {  return 'Faltan campos obligatorios';  }  if (!/^\d+$/.test(credencial)) {  return 'La credencial debe ser numérica';  }  // Validar que el CUIL sea un número de 11 dígitos  if (!/^\d{11}$/.test(cuil)) {  return 'El Cuil debe ser un número de 11 dígitos';  }  return null;  }  // Rutas CRUD  app.get('/api/agentes', async (req, res) => {  try {  const searchTerm = req.query.search || '';  const [results] = await db.execute(`  SELECT \* FROM agentes  WHERE grado LIKE ? OR nombre LIKE ? OR credencial LIKE ? OR cuil LIKE ? OR sector LIKE ?  `, [`%${searchTerm}%`, `%${searchTerm}%`, `%${searchTerm}%`, `%${searchTerm}%`, `%${searchTerm}%`]);  res.json(results);  } catch (err) {  res.status(500).json({ error: err.message });  }  });  app.get('/api/agentes/:id', async (req, res) => {  try {  const { id } = req.params;  const [results] = await db.execute('SELECT \* FROM agentes WHERE id = ?', [id]);  if (results.length === 0) {  return res.status(404).json({ error: 'Registro no encontrado' });  }  res.json(results[0]);  } catch (err) {  res.status(500).json({ error: err.message });  }  });  app.post('/api/agentes', async (req, res) => {  const error = validateAgentData(req.body);  if (error) {  return res.status(400).json({ error });  }  const { grado, nombre, credencial, cuil, sector } = req.body;  try {  const [result] = await db.execute('INSERT INTO agentes (grado, nombre, credencial, cuil, sector) VALUES (?, ?, ?, ?, ?)', [grado, nombre, credencial, cuil, sector]);  res.json({ id: result.insertId, ...req.body });  } catch (err) {  res.status(500).json({ error: err.message });  }  });  app.put('/api/agentes/:id', async (req, res) => {  const { id } = req.params;  const error = validateAgentData(req.body);  if (error) {  return res.status(400).json({ error });  }  const { grado, nombre, credencial, cuil, sector } = req.body;  try {  await db.execute('UPDATE agentes SET grado = ?, nombre = ?, credencial = ?, cuil = ?, sector = ? WHERE id = ?', [grado, nombre, credencial, cuil, sector, id]);  res.json({ message: 'Registro actualizado correctamente' });  } catch (err) {  res.status(500).json({ error: err.message });  }  });  app.delete('/api/agentes/:id', async (req, res) => {  const { id } = req.params;  try {  await db.execute('DELETE FROM agentes WHERE id = ?', [id]);  res.json({ message: 'Registro eliminado correctamente' });  } catch (err) {  res.status(500).json({ error: err.message });  }  });  // Iniciar el servidor  const PORT = process.env.PORT || 5000;  app.listen(PORT, () => {  console.log(`Servidor corriendo en http://localhost:${PORT}`);  }); |