# **Bài thực hành: Xây dựng một web frontend CRUD thông tin sinh viên**

## Mục tiêu:

* Xây dựng trang web gồm 4 thao tác CRUD dữ liệu sinh viên
* Phân trang kết quả

## Đầu vào:

* Dữ liệu response trả về dạng **json**, có key là **data**, value là **một mảng sinh viên**
* Dữ liệu mỗi sinh viên gồm các trường: **\_id, studentId, name, dob, address**
* Gọi các đầu API được cung cấp bởi backend (script.js)

const apiUrl = 'http://localhost:3001';

## Các bước thực hiện:

### Tạo file index.html với cấu trúc cơ bản:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Student CRUD</title>

<link rel="stylesheet" href="./style.css" />

</head>

<body>

<script src="./script.js"></script>

</body>

</html>

### Tạo form nhập thông tin sinh viên (index.html)

<!-- Modal Popup for Student Form -->

<div id="studentModal" class="modal">

<div class="modal-content">

<span class="close" id="closeModal">&times;</span>

<form id="studentForm">

<input type="hidden" id="dbEntryId" />

<label for="studentId">Student ID:</label>

<input type="number" id="studentId" />

<label for="name">Name:</label>

<input type="text" id="name" required />

<label for="dob">Date of Birth:</label>

<input type="date" id="dob" required />

<label for="address">Address:</label>

<input type="text" id="address" required />

<button type="submit" id="addSubmit">Add Student</button>

<button type="submit" id="editSubmit">Edit Student</button>

</form>

</div>

</div>

* Form sẽ được hiển thị dưới dạng một cửa sổ pop-up (modal) khi người dùng nhấn nút thêm (add) hoặc thay đổi (edit) thông tin sinh viên.

### Tạo bảng để hiển thị danh sách sinh viên (index.html)

<div class="container">

<h1>Student Information</h1>

<button id="addStudentBtn">Add Student</button>

<table id="studentTable">

<thead>

<tr>

<th style="display: none">Database Entry Id</th>

<th>Student ID</th>

<th>Name</th>

<th>Date of Birth</th>

<th>Address</th>

<th>Actions</th>

</tr>

</thead>

<tbody id="studentList">

<!-- Rows will be dynamically added here -->

</tbody>

</table>

<div id="paginationControls" class="pagination-controls"></div>

</div>

* Mỗi sinh viên được hiển thị trên một hàng của bảng. Danh sách sinh viên sẽ được khởi tạo và cập nhật bằng javascript.
* Thêm thẻ div với id “paginationControls" để chuyển trang. Các nút bấm tương ứng với số trang sẽ được thêm dần bằng code javascript.

* Sử dụng fetch để gọi API, tham khảo: [Fetch document](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch)

#### Lấy danh sách sinh viên và thêm vào bảng

// Fetch and display students in a table

async function fetchStudents() {

const response = await fetch(apiUrl);

students = (await response.json()).data;

displayStudents()

setupPagination()

}

function displayStudents() {

const studentTableBody = document.getElementById('studentList');

studentTableBody.innerHTML = ''; // Clear existing data

const start = (currentPage - 1) \* pageSize

const end = start + pageSize

const paginatedStudents = students.slice(start, end)

paginatedStudents.forEach(student => {

const tr = document.createElement('tr');

tr.innerHTML = `

<td style="display: none">${student.\_id}</td>

<td>${student.studentId}</td>

<td>${student.name}</td>

<td>${student.dob.slice(0, 10)}</td>

<td>${student.address}</td>

<td>

<button class="edit" onclick="loadStudent('${student.\_id}')">Edit</button>

<button class="delete" onclick="deleteStudent('${student.\_id}')">Delete</button>

</td>

`;

studentTableBody.appendChild(tr);

});

}

* Ngoài các thông tin của sinh viên, mỗi dòng còn có nút “Edit" và “Delete" tương ứng để thay đổi thông tin sinh viên và xoá sinh viên. Các nút bắt sự kiện click (onclick) để gọi hàm chức năng tương ứng.

#### Thiết lập phân trang

// Setup pagination controls

function setupPagination() {

const paginationControls = document.getElementById('paginationControls');

paginationControls.innerHTML = ''; // Clear existing pagination buttons

const totalPages = Math.ceil(students.length / pageSize);

for (let i = 1; i <= totalPages; i++) {

const button = document.createElement('button');

button.textContent = i;

button.classList.add('pagination-btn');

if (i === currentPage) {

button.classList.add('active'); // Highlight the current page

}

button.addEventListener('click', () => {

currentPage = i;

displayStudents(); // Display the students for the selected page

});

paginationControls.appendChild(button);

}

}

#### Thêm/sửa thông tin sinh viên

// Add or update student

document.getElementById('studentForm').addEventListener('submit', async (event) => {

event.preventDefault();

console.log(document.getElementById('update'))

const isUpdate = event.submitter.id != "addSubmit"

const method = isUpdate ? 'PUT' : 'POST'

let url = apiUrl

const studentData = {

dbEntryId: document.getElementById('dbEntryId').value,

studentId: document.getElementById('studentId').value,

name: document.getElementById('name').value,

dob: document.getElementById('dob').value,

address: document.getElementById('address').value

};

if (method === 'PUT') {

url += `/${studentData.dbEntryId}`

}

try {

const response = await fetch(url, {

method: method,

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(studentData)

});

if (response.ok) {

hidePopup(); // Hide the popup after saving

await fetchStudents();

} else {

const errorResponse = await response.json();

alert("Error: ", errorResponse)

}

} catch (error) {

console.error('Error adding/updating student: ', error)

}

});

* Đoạn code trên thực hiện các nhiệm vụ:
  + Bắt sự kiện form nhập thông tin sinh viên được submit
  + Xác định yêu cầu là thêm mới sinh viên hay sửa thông tin sinh viên dựa trên nguồn phát của sự kiện là nút “Add" hay “Edit"
  + Gửi request POST/PATCH tương ứng về api
  + Gọi lại hàm lấy danh sách sinh viên để cập nhật vào bảng.

#### Lấy thông tin sinh viên cần chỉnh sửa

// Load student data into the form for editing

async function loadStudent(id) {

const response = await fetch(`${apiUrl}/${id}`);

const student = (await response.json()).**data**;

document.getElementById('dbEntryId').value = student.\_**id**;

document.getElementById('studentId').value = student.**studentId**;

document.getElementById('name').value = student.**name**;

document.getElementById('dob').value = student.**dob**.slice(0, 10);

document.getElementById('address').value = student.**address**;

showPopup(true); // Show the form in popup when editing

}

#### Hiện/ẩn form nhập thông tin sinh viên

// Show the "Create Student" popup form

function showPopup(isUpdate) {

const popup = document.getElementById('studentModal');

popup.style.display = 'block';

if (isUpdate) {

popup.querySelector("#editSubmit").style.display = 'block'

} else {

popup.querySelector("#addSubmit").style.display = 'block'

}

}

// Hide the "Create Student" popup form

function hidePopup() {

const popup = document.getElementById('studentModal');

popup.style.display = 'none';

popup.querySelector("#addSubmit").style.display = 'none'

popup.querySelector("#editSubmit").style.display = 'none'

document.getElementById('studentForm').reset();

}

// Event listener for the "Add Student" button to show the popup

document.getElementById('addStudentBtn').addEventListener('click', () => { showPopup(false) });

// Event listener to close the modal

document.getElementById('closeModal').addEventListener('click', hidePopup)

#### Xoá sinh viên

// Delete student

async function deleteStudent(id) {

await fetch(`${apiUrl}/${id}`, { method: 'DELETE' });

await fetchStudents();

}

#### Lấy danh sách sinh viên lần đầu khi khởi tạo website

// Initial fetch of students

fetchStudents();

### Cài đặt hiển thị cho cửa sổ pop-up (modal) và form nhập thông tin sinh viên (style.css)

/\* Modal Styling \*/

.modal {

display: none;

position: fixed;

z-index: 1;

left: 0;

top: 0;

width: 100%;

height: 100%;

background-color: rgba(0, 0, 0, 0.5);

}

form#studentForm button[type="submit"] {

display: none;

}

### Căn chỉnh, làm đẹp bằng css:

* Class container: căn giữa với width và margin

.container {

width: 95%;

margin: auto;

}

* Gộp các viền của bảng lại bằng border-collapse

table {

border-collapse: collapse;

}

* Hiện viền cho bảng và các ô trong bảng:

table, th, td {

border: 1px solid #ddd;

}

* Làm đẹp cho các nút bấm:
  + Đổi con trỏ chuột khi di chuột lên nút bấm:

button {

cursor: pointer;

}

* Đổi màu nút bấm khi di chuột lên nút bấm:

button:hover {

background-color: #218838;

}