**Atelier 5 : Programmation Asynchrone - Devoir**

# Objectif :

**L’objectif principal de ce travail pratique est de se familiariser avec les concepts de la programmation asynchrone.**

# Exercice 1

1. Créer une fonction asynchrone qui simule la récupération de données utilisateur après un délai, puis traiter ces données:

**const userData = {**

**name: 'John Doe',**

**email: 'johndoe@example.com',**

**avatar: 'avatar.png',**

**gender: 'M'**

**};**

**async function fetchUserData() {**

**return new Promise((resolve) => {**

**setTimeout(() => {**

**resolve(userData);**

**}, 2000);**

**});**

**}**

**fetchUserData().then(user => {**

**console.log("Utilisateur récupéré :", user);**

**});**

**2. Une fois la donnée est récupérée afficher les informations dans une page selon le thème profile:**

<!DOCTYPE html>

<html lang="fr">

<head>

<meta charset="UTF-8">

<title>Profil Utilisateur</title>

<style>

.profile {

border: 1px solid #ccc;

padding: 20px;

width: 300px;

font-family: Arial;

}

.profile img {

width: 100px;

height: 100px;

border-radius: 50%;

}

</style>

</head>

<body>

<h1>Profil</h1>

<div id="profile" class="profile"></div>

<script>

const userData = {

name: 'John Doe',

email: 'johndoe@example.com',

avatar: 'avatar.png',

gender: 'M'

};

async function fetchUserData() {

return new Promise((resolve) => {

setTimeout(() => {

resolve(userData);

}, 1500);

});

}

fetchUserData().then(user => {

document.getElementById("profile").innerHTML = `

<img src="${user.avatar}" alt="Avatar">

<h3>${user.name}</h3>

<p><strong>Email:</strong> ${user.email}</p>

<p><strong>Genre:</strong> ${user.gender}</p>

`;

});

</script>

</body>

</html>

**3**. Ajouter plus de données puis afficher dans un tableau:

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Tableau des utilisateurs</title>**

**</head>**

**<body>**

**<h2>Utilisateurs</h2>**

**<div id="userTable"></div>**

**<script>**

**const users = [**

**{**

**name: 'John Doe',**

**email: 'john@example.com',**

**login: 'johndoe',**

**password: 'pass123',**

**address: '123 rue Exemple'**

**},**

**{**

**name: 'Sara Bakkali',**

**email: 'bakkali.sara@ensi.ma',**

**login: 'sarabakkali',**

**password: 'ensi2025',**

**address: '2 Rue Melilla, Tangier 90060'**

**}**

**];**

**function displayUserTable() {**

**let table = `<table border="1" cellpadding="5">**

**<thead>**

**<tr><th>Nom</th><th>Email</th><th>Login</th><th>Mot de passe</th><th>Adresse</th></tr>**

**</thead>**

**<tbody>**

**${users.map(u => `**

**<tr>**

**<td>${u.name}</td>**

**<td>${u.email}</td>**

**<td>${u.login}</td>**

**<td>${u.password}</td>**

**<td>${u.address}</td>**

**</tr>`).join('')}**

**</tbody>**

**</table>`;**

**document.getElementById("userTable").innerHTML = table;**

**}**

**displayUserTable();**

**</script>**

**</body>**

**</html>**

4. Chainer les Promises pour simuler plusieurs actions:

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Utilisateur + Commandes</title>

</head>

<body>

<h2>Données de l'utilisateur</h2>

<div id="info"></div>

<script>

const user = {

name: "John Doe",

email: "john@example.com"

};

function fetchUser() {

return new Promise(resolve => {

setTimeout(() => resolve(user), 1000);

});

}

function fetchOrders() {

return new Promise(resolve => {

setTimeout(() => resolve([

{ id: 1, item: "Ordinateur" },

{ id: 2, item: "Clavier" }

]), 1500);

});

}

fetchUser()

.then(user => {

document.getElementById("info").innerHTML = `<p><strong>Nom:</strong> ${user.name}</p>`;

return fetchOrders();

})

.then(orders => {

const list = orders.map(o => `<li>${o.item}</li>`).join('');

document.getElementById("info").innerHTML += `<h3>Commandes:</h3><ul>${list}</ul>`;

});

</script>

</body>

</html>

# Exercice 2

**Créer un frontend JavaScript qui interagit avec une API Laravel pour télécharger et récupérer des fichiers.**

1. Route et contrôleur pour upload de fichiers (Laravel):

Déclaration des routes (dans routes/api.php)

use App\Http\Controllers\FileController;

Route::post('/upload', [FileController::class, 'upload']);

Route::get('/files', [FileController::class, 'listFiles']);

Contrôleur FileController (dans app/Http/Controllers/FileController.php)

**<?php**

**namespace App\Http\Controllers;**

**use Illuminate\Http\Request;**

**use Illuminate\Support\Facades\Storage;**

**class FileController extends Controller**

**{**

**public function upload(Request $request)**

**{**

**$request->validate([**

**'file' => 'required|file|max:10240' // max 10 Mo**

**]);**

**$path = $request->file('file')->store('uploads');**

**return response()->json([**

**'success' => true,**

**'path' => $path**

**], 201);**

**}**

**public function listFiles()**

**{**

**$files = Storage::files('uploads');**

**$filesWithUrl = array\_map(function($filename) {**

**return [**

**'path' => $filename,**

**'url' => Storage::url($filename)**

**];**

**}, $files);**

**return response()->json([**

**'success' => true,**

**'files' => $filesWithUrl**

**]);**

**}**

**}**

2. Frontend JavaScript avec fetch et Promises :

<!DOCTYPE html>

<html lang="fr">

<head>

<meta charset="UTF-8">

<title>Upload de fichier</title>

</head>

<body>

<h2>Uploader un fichier</h2>

<input type="file" id="fileInput">

<button onclick="uploadFile()">Envoyer</button>

<h2>Fichiers disponibles</h2>

<button onclick="getFiles()">Afficher les fichiers</button>

<ul id="fileList"></ul>

<script>

function uploadFile() {

const file = document.getElementById('fileInput').files[0];

const formData = new FormData();

formData.append('file', file);

fetch('http://localhost:8000/api/upload', {

method: 'POST',

body: formData

})

.then(response => response.json())

.then(data => {

alert("Fichier envoyé avec succès : " + data.path);

})

.catch(error => {

alert("Erreur lors de l'envoi : " + error);

});

}

function getFiles() {

fetch('http://localhost:8000/api/files')

.then(response => response.json())

.then(data => {

const list = data.files.map(file => `

<li>

<a href="${file.url}" target="\_blank">${file.path}</a>

</li>`).join('');

document.getElementById("fileList").innerHTML = list;

})

.catch(error => {

alert("Erreur de récupération : " + error);

});

}

</script>

</body>

</html>

1. Affichage des fichiers :

**<!DOCTYPE html>**

**<html lang="fr">**

**<head>**

**<meta charset="UTF-8">**

**<title>Liste des fichiers</title>**

**</head>**

**<body>**

**<h2>Fichiers disponibles</h2>**

**<button onclick="getFiles()">Charger les fichiers</button>**

**<ul id="fileList"></ul>**

**<script>**

**function getFiles() {**

**fetch('http://localhost:8000/api/files')**

**.then(response => response.json())**

**.then(data => {**

**if (!data.success) throw new Error("Réponse API invalide");**

**const listHTML = data.files.map(file => `**

**<li>**

**<a href="${file.url}" target="\_blank">${file.path}</a>**

**</li>**

**`).join('');**

**document.getElementById("fileList").innerHTML = listHTML;**

**})**

**.catch(error => {**

**console.error("Erreur:", error);**

**document.getElementById("fileList").innerHTML = `<li>Erreur lors du chargement</li>`;**

**});**

**}**

**</script>**

**</body>**

**</html>**

# Exercice 3

**Gestion des salles avec API Laravel et JavaScript**.

1. **Migration des salles :**

**Créer la migration**

**php artisan make:migration create\_rooms\_table**

**Modifier la migration générée**

**<?php**

**use Illuminate\Database\Migrations\Migration;**

**use Illuminate\Database\Schema\Blueprint;**

**use Illuminate\Support\Facades\Schema;**

**class CreateRoomsTable extends Migration**

**{**

**public function up()**

**{**

**Schema::create('rooms', function (Blueprint $table) {**

**$table->id(); // id (clé primaire auto-incrémentée)**

**$table->string('name'); // nom de la salle**

**$table->integer('capacity'); // capacité de la salle**

**$table->timestamps(); // colonnes created\_at & updated\_at**

**});**

**}**

**public function down()**

**{**

**Schema::dropIfExists('rooms');**

**}**

**}**

**Lancer la migration**

**php artisan migrate**

1. CRUD dans RoomController :

**Créer le contrôleur**

**<?php**

**namespace App\Http\Controllers;**

**use App\Models\Room;**

**use Illuminate\Http\Request;**

**class RoomController extends Controller**

**{**

**public function index()**

**{**

**return response()->json(Room::all());**

**}**

**public function store(Request $request)**

**{**

**$request->validate([**

**'name' => 'required|string',**

**'capacity' => 'required|integer'**

**]);**

**$room = Room::create([**

**'name' => $request->name,**

**'capacity' => $request->capacity**

**]);**

**return response()->json($room, 201);**

**}**

**public function update(Request $request, $id)**

**{**

**$room = Room::findOrFail($id);**

**$request->validate([**

**'name' => 'sometimes|required|string',**

**'capacity' => 'sometimes|required|integer'**

**]);**

**$room->update($request->only('name', 'capacity'));**

**return response()->json($room);**

**}**

**public function destroy($id)**

**{**

**$room = Room::findOrFail($id);**

**$room->delete();**

**return response()->json(['message' => 'Salle supprimée avec succès']);**

**}**

**}**

**Ajouter les routes API (routes/api.php)**

**use App\Http\Controllers\RoomController;**

**Route::get('/rooms', [RoomController::class, 'index']);**

**Route::post('/rooms', [RoomController::class, 'store']);**

**Route::put('/rooms/{id}', [RoomController::class, 'update']);**

**Route::delete('/rooms/{id}', [RoomController::class, 'destroy']);**

**3. Créer des fonctions pour chaque opération CRUD:**

<!DOCTYPE html>

<html lang="fr">

<head>

<meta charset="UTF-8">

<title>Gestion des Salles</title>

</head>

<body>

<h2>Ajouter une salle</h2>

<input type="text" id="name" placeholder="Nom de la salle">

<input type="number" id="capacity" placeholder="Capacité">

<button onclick="createRoom()">Ajouter</button>

<h2>Liste des salles</h2>

<button onclick="listRooms()">Actualiser</button>

<ul id="roomList"></ul>

<script>

function listRooms() {

fetch('http://localhost:8000/api/rooms')

.then(response => response.json())

.then(data => {

const list = data.map(room => `

<li>

${room.name} (${room.capacity} places)

<button onclick="deleteRoom(${room.id})">Supprimer</button>

<button onclick="updateRoom(${room.id}, '${room.name}', ${room.capacity})">Modifier</button>

</li>`).join('');

document.getElementById("roomList").innerHTML = list;

});

}

function createRoom() {

const name = document.getElementById("name").value;

const capacity = document.getElementById("capacity").value;

fetch('http://localhost:8000/api/rooms', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ name, capacity })

})

.then(response => response.json())

.then(data => {

alert("Salle ajoutée !");

listRooms();

});

}

function deleteRoom(id) {

fetch(`http://localhost:8000/api/rooms/${id}`, {

method: 'DELETE'

})

.then(response => response.json())

.then(data => {

alert(data.message);

listRooms();

});

}

function updateRoom(id, currentName, currentCapacity) {

const newName = prompt("Nouveau nom :", currentName);

const newCapacity = prompt("Nouvelle capacité :", currentCapacity);

fetch(`http://localhost:8000/api/rooms/${id}`, {

method: 'PUT',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ name: newName, capacity: parseInt(newCapacity) })

})

.then(response => response.json())

.then(data => {

alert("Salle mise à jour !");

listRooms();

});

}

listRooms();

</script>

</body>

</html>

4-Créer une interface utilisateur simple:

**<!DOCTYPE html>**

**<html lang="fr">**

**<head>**

**<meta charset="UTF-8">**

**<title>Interface de Gestion des Salles</title>**

**<style>**

**body { font-family: Arial, sans-serif; padding: 20px; }**

**input, button { margin: 5px; }**

**table { border-collapse: collapse; margin-top: 20px; }**

**th, td { border: 1px solid #aaa; padding: 10px; text-align: left; }**

**</style>**

**</head>**

**<body>**

**<h1>Gestion des Salles</h1>**

**<h3>➕ Ajouter une nouvelle salle</h3>**

**<input type="text" id="roomName" placeholder="Nom">**

**<input type="number" id="roomCapacity" placeholder="Capacité">**

**<button onclick="createRoom()">Ajouter</button>**

**<h3>📋 Liste des salles</h3>**

**<table id="roomTable">**

**<thead>**

**<tr>**

**<th>Nom</th>**

**<th>Capacité</th>**

**<th>Actions</th>**

**</tr>**

**</thead>**

**<tbody></tbody>**

**</table>**

**<script>**

**function listRooms() {**

**fetch("http://localhost:8000/api/rooms")**

**.then(res => res.json())**

**.then(data => {**

**const tbody = document.querySelector("#roomTable tbody");**

**tbody.innerHTML = "";**

**data.forEach(room => {**

**const row = `**

**<tr>**

**<td>${room.name}</td>**

**<td>${room.capacity}</td>**

**<td>**

**<button onclick="editRoom(${room.id}, '${room.name}', ${room.capacity})">✏️ Modifier</button>**

**<button onclick="deleteRoom(${room.id})">🗑️ Supprimer</button>**

**</td>**

**</tr>**

**`;**

**tbody.innerHTML += row;**

**});**

**});**

**}**

**function createRoom() {**

**const name = document.getElementById("roomName").value;**

**const capacity = document.getElementById("roomCapacity").value;**

**fetch("http://localhost:8000/api/rooms", {**

**method: "POST",**

**headers: { "Content-Type": "application/json" },**

**body: JSON.stringify({ name, capacity })**

**})**

**.then(res => res.json())**

**.then(() => {**

**document.getElementById("roomName").value = "";**

**document.getElementById("roomCapacity").value = "";**

**listRooms();**

**});**

**}**

**function editRoom(id, oldName, oldCapacity) {**

**const name = prompt("Nom :", oldName);**

**const capacity = prompt("Capacité :", oldCapacity);**

**fetch(`http://localhost:8000/api/rooms/${id}`, {**

**method: "PUT",**

**headers: { "Content-Type": "application/json" },**

**body: JSON.stringify({ name, capacity })**

**})**

**.then(() => listRooms());**

**}**

**function deleteRoom(id) {**

**fetch(`http://localhost:8000/api/rooms/${id}`, {**

**method: "DELETE"**

**})**

**.then(() => listRooms());**

**}**

**listRooms();**

**</script>**

**</body>**

**</html>**