Voici un exemple d'application JavaScript qui utilise **React** et consomme l'API de [JSONPlaceholder](https://jsonplaceholder.typicode.com/). Cette application est une Single Page Application (SPA) qui répond à votre demande. Assurez-vous d'avoir Node.js et npm installés pour exécuter cette application.

**Structure du projet**

1. **public/index.html**: Point d'entrée de l'application.
2. **src/index.js**: Point d'entrée JavaScript.
3. **src/App.js**: Contient la logique principale.
4. **src/components/PostList.js**: Affiche la liste des posts.
5. **src/components/PostDetails.js**: Affiche les détails d’un post et permet de modifier ou supprimer un post.

Voici un exemple d'application JavaScript qui utilise **React** et consomme l'API de [JSONPlaceholder](https://jsonplaceholder.typicode.com/). Cette application est une Single Page Application (SPA) qui répond à votre demande. Assurez-vous d'avoir Node.js et npm installés pour exécuter cette application.

### Structure du projet

1. **public/index.html**: Point d'entrée de l'application.
2. **src/index.js**: Point d'entrée JavaScript.
3. **src/App.js**: Contient la logique principale.
4. **src/components/PostList.js**: Affiche la liste des posts.
5. **src/components/PostDetails.js**: Affiche les détails d’un post et permet de modifier ou supprimer un post.

#### Étape 3 : Modifier les fichiers

##### src/index.js

javascript

Copier le code

import React from "react";

import ReactDOM from "react-dom";

import App from "./App";

import "./index.css";

ReactDOM.render(<App />, document.getElementById("root"));

##### src/App.js

javascript

Copier le code

import React, { useState } from "react";

import PostList from "./components/PostList";

import PostDetails from "./components/PostDetails";

const App = () => {

const [selectedPost, setSelectedPost] = useState(null);

const handlePostClick = (post) => {

setSelectedPost(post);

};

const handleBackToList = () => {

setSelectedPost(null);

};

return (

<div className="app-container">

<h1>JSONPlaceholder Posts</h1>

{selectedPost ? (

<PostDetails post={selectedPost} onBack={handleBackToList} />

) : (

<PostList onPostClick={handlePostClick} />

)}

</div>

);

};

export default App;

##### src/components/PostList.js

javascript

Copier le code

import React, { useEffect, useState } from "react";

import axios from "axios";

const PostList = ({ onPostClick }) => {

const [posts, setPosts] = useState([]);

const [loading, setLoading] = useState(true);

useEffect(() => {

axios

.get("https://jsonplaceholder.typicode.com/posts")

.then((response) => {

setPosts(response.data);

setLoading(false);

})

.catch((error) => {

console.error("Error fetching posts:", error);

});

}, []);

if (loading) {

return <p>Loading posts...</p>;

}

return (

<div>

<ul>

{posts.map((post) => (

<li key={post.id}>

<button onClick={() => onPostClick(post)}>{post.title}</button>

</li>

))}

</ul>

</div>

);

};

export default PostList;

##### src/components/PostDetails.js

javascript

Copier le code

import React, { useState } from "react";

import axios from "axios";

const PostDetails = ({ post, onBack }) => {

const [isEditing, setIsEditing] = useState(false);

const [formData, setFormData] = useState({

title: post.title,

body: post.body,

});

const handleDelete = () => {

axios

.delete(`https://jsonplaceholder.typicode.com/posts/${post.id}`)

.then(() => {

alert("Post deleted!");

onBack();

})

.catch((error) => {

console.error("Error deleting post:", error);

});

};

const handleEditToggle = () => {

setIsEditing(!isEditing);

};

const handleInputChange = (e) => {

const { name, value } = e.target;

setFormData({ ...formData, [name]: value });

};

const handleUpdate = (e) => {

e.preventDefault();

axios

.put(`https://jsonplaceholder.typicode.com/posts/${post.id}`, formData)

.then((response) => {

alert("Post updated!");

setIsEditing(false);

})

.catch((error) => {

console.error("Error updating post:", error);

});

};

return (

<div>

<button onClick={onBack}>Back to List</button>

<h2>Post Details</h2>

{isEditing ? (

<form onSubmit={handleUpdate}>

<div>

<label>Title:</label>

<input

type="text"

name="title"

value={formData.title}

onChange={handleInputChange}

/>

</div>

<div>

<label>Body:</label>

<textarea

name="body"

value={formData.body}

onChange={handleInputChange}

/>

</div>

<button type="submit">Save Changes</button>

</form>

) : (

<>

<p><strong>Title:</strong> {post.title}</p>

<p><strong>Body:</strong> {post.body}</p>

</>

)}

<button onClick={handleDelete}>Delete</button>

{!isEditing && <button onClick={handleEditToggle}>Edit</button>}

</div>

);

};

export default PostDetails;

#### Étape 4 : Exécuter le projet

Lancez l’application avec :

bash

Copier le code

npm start

### Résultat

* **Liste des posts** : Affiche tous les posts.
* **Détails d’un post** : Cliquez sur un post pour afficher ses détails.
* **Supprimer un post** : Cliquez sur "Delete" pour supprimer le post.
* **Modifier un post** : Cliquez sur "Edit", modifiez le contenu, puis cliquez sur "Save Changes".

Cette application est évolutive et peut être enrichie avec des styles CSS ou des bibliothèques comme Material-UI ou Tailwind CSS.

Créer une application en JavaScript pur (vanilla JavaScript) pour consommer l'API [JSONPlaceholder](https://jsonplaceholder.typicode.com/) implique de manipuler directement le DOM, de gérer les événements et de faire des requêtes HTTP via fetch. Voici une version qui répond à vos besoins sans framework.

### Structure du Projet

lua

Copier le code

/project-folder

|-- index.html

|-- style.css

|-- app.js

### 1. index.html

C'est le point d'entrée de l'application.

html

Copier le code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>JSONPlaceholder Posts</title>

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

</head>

<body>

<div id="app">

<h1>JSONPlaceholder Posts</h1>

<div id="content"></div>

</div>

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

</body>

</html>

### 2. style.css

Ajout d'un style basique.

css

Copier le code

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #f4f4f4;

}

#app {

max-width: 800px;

margin: auto;

padding: 20px;

background: white;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

button {

background-color: #007BFF;

color: white;

border: none;

padding: 10px;

margin: 5px;

cursor: pointer;

border-radius: 5px;

}

button:hover {

background-color: #0056b3;

}

button.delete {

background-color: #dc3545;

}

button.delete:hover {

background-color: #a71d2a;

}

form {

margin-top: 20px;

}

form input, form textarea {

width: 100%;

padding: 10px;

margin: 10px 0;

border: 1px solid #ccc;

border-radius: 5px;

}

### 3. app.js

Voici le fichier JavaScript qui gère toute la logique.

javascript

Copier le code

// Point d'entrée de l'application

const app = document.getElementById("content");

// Charger la liste des posts

async function loadPosts() {

app.innerHTML = "<p>Loading posts...</p>";

try {

const response = await fetch("https://jsonplaceholder.typicode.com/posts");

const posts = await response.json();

const postList = document.createElement("ul");

posts.forEach((post) => {

const postItem = document.createElement("li");

const postButton = document.createElement("button");

postButton.textContent = post.title;

postButton.addEventListener("click", () => loadPostDetails(post));

postItem.appendChild(postButton);

postList.appendChild(postItem);

});

app.innerHTML = ""; // Efface le chargement

app.appendChild(postList);

} catch (error) {

console.error("Error loading posts:", error);

app.innerHTML = "<p>Failed to load posts.</p>";

}

}

// Charger les détails d'un post

async function loadPostDetails(post) {

app.innerHTML = "<p>Loading post details...</p>";

try {

const response = await fetch(`https://jsonplaceholder.typicode.com/posts/${post.id}`);

const postDetails = await response.json();

app.innerHTML = `

<button onclick="loadPosts()">Back to Posts</button>

<h2>${postDetails.title}</h2>

<p>${postDetails.body}</p>

<button class="delete" onclick="deletePost(${postDetails.id})">Delete</button>

<button onclick="showEditForm(${postDetails.id}, '${postDetails.title}', '${postDetails.body}')">Edit</button>

`;

} catch (error) {

console.error("Error loading post details:", error);

app.innerHTML = "<p>Failed to load post details.</p>";

}

}

// Supprimer un post

async function deletePost(postId) {

if (!confirm("Are you sure you want to delete this post?")) return;

try {

await fetch(`https://jsonplaceholder.typicode.com/posts/${postId}`, {

method: "DELETE",

});

alert("Post deleted!");

loadPosts();

} catch (error) {

console.error("Error deleting post:", error);

alert("Failed to delete post.");

}

}

// Afficher le formulaire de modification

function showEditForm(postId, title, body) {

app.innerHTML = `

<button onclick="loadPosts()">Back to Posts</button>

<h2>Edit Post</h2>

<form onsubmit="updatePost(event, ${postId})">

<label>Title</label>

<input type="text" id="edit-title" value="${title}" required>

<label>Body</label>

<textarea id="edit-body" required>${body}</textarea>

<button type="submit">Save Changes</button>

</form>

`;

}

// Mettre à jour un post

async function updatePost(event, postId) {

event.preventDefault();

const title = document.getElementById("edit-title").value;

const body = document.getElementById("edit-body").value;

try {

await fetch(`https://jsonplaceholder.typicode.com/posts/${postId}`, {

method: "PUT",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({ title, body }),

});

alert("Post updated!");

loadPosts();

} catch (error) {

console.error("Error updating post:", error);

alert("Failed to update post.");

}

}

// Charger la liste des posts au démarrage

loadPosts();

### Fonctionnalités

1. **Affichage des posts** :
   * Une liste des titres des posts s'affiche au chargement.
2. **Détails d'un post** :
   * En cliquant sur un titre, vous accédez aux détails du post.
3. **Suppression d'un post** :
   * Cliquez sur "Delete" pour supprimer un post.
4. **Modification d'un post** :
   * Cliquez sur "Edit" pour afficher un formulaire permettant de modifier le titre et le corps du post.

### Pour tester

1. Créez les trois fichiers (index.html, style.css, app.js) dans un même dossier.
2. Ouvrez index.html dans votre navigateur.
3. Testez les fonctionnalités. 🎉