POLYTECH DIJON

TRAVAUX DIRIGÉS

NodeJS – TypeScript

Requêter une API

OBJECTIFS

- Manipuler une API à partir d'un frontend afin de transmettre et recevoir des données.
- Manipuler l'API Fetch
- Manipuler le langage TypeScript en frontend

PREPARATION

PREREQUIS

- NodeJS doit être installé.

PROJET

- Créez un dossier pour le projet du TD.
- Ouvrez ce dossier dans VS Code.

BACKEND

- Créez un sous-dossier nommé `api` et qui contiendra les fichiers de l'archive suivante :

  https://www.lamarmotte.info/wp-content/uploads/2026/06/api.zip

- Ouvrez un terminal dans VS Code et assurez-vous d'être dans le dossier `api` précédemment créé.
- Installez les paquets NPM requis.
- Transpilez le code et exécuter le fichier `index.js` présent dans le dossier `api/dist`.

FRONTEND

- Dans le terminal, placez vous à la racine de votre projet.
- Créez un nouveau projet ViteJS (Vanilla + TypeScript) nommé `front`.
- Supprimez tous les éléments présents à l'intérieur du dossier src du projet ViteJS créé, sauf le fichier `main.ts`.
- Editez le fichier `main.ts` et supprimez en le contenu pour repartir d'une page vierge.
- Ouvrez le lien fourni par ViteJS dans le terminal et vérifiez que vous obtenez une page blanche sans erreur dans la console.

charles.meunier@ube.fr

TODO LIST

➢ Réalisez un frontend permettant de gérer une TodoList.

> **Note**
> Le backend téléchargé précédemment gère une base de données des tâches à réaliser manipulable à travers une API documentée en Annexe 1.

> **Tips**
> L'API Fetch est documentée en Annexe 2.

ANNEXES

ANNEXE 1

Documentation de l'API fournie par le backend.

Annexe 1.1

Obtenir la liste des tâches enregistrées dans la base de données.

Client request	
URL	`/tasks`
Method	`GET`
Server response	
Content-Type	`application/json`

Body	`[`
	`  {`
	`  id : number,`
	`  content : string,`
	`  done : number`
	`  },`
	`  …`
	`]`

Response code	`200 – OK` `500 – Internal serveur error`

Annexe 1.2

Enregistrer une nouvelle tâche dans la base de données.

Client request	
URL	`/tasks`
Method	`POST`
Content-Type	`application/json`

Body	`{`
	`  content : string`
	`}`

Server response	

Content-Type	application/json

Body	{
	id : number, // Inserted row id
	}

Response Code	201 – OK : task created 500 – Internal server error

Annexe 1.3

Modifier une tâche existante dans la base de données.

Client request	
URL	/tasks/{taskId}
Method	PUT
Content-Type	application/json

Body	{
	content : string,
	done : number,
	}

Server response	
Response Code	204 – OK : task updated 500 – Internal server error

Annexe 1.4

Supprimer une tâche existante de la base de données.

Client request	
URL	/tasks/{taskId}
Method	DELETE
Server response	
Response Code	204 – OK : task deleted 500 – Internal server error

ANNEXE 2

Documentation de l'API Fetch qui permet de réaliser des requêtes HTTP vers un serveur.

```
// Effectue une requête auprès du serveur
const response = await fetch(
      "url_to_request",
      {
            method: "GET (default) / POST / DELETE / PUT",
            headers (optional): { "Content-Type": "application/json" },
            body (optional): "json_data"
      }
);
// Récupère les données transmises par le serveur (si nécessaire)
if(response.ok)
{
      const data = await response.json();
}
```