T1e - Application web back-end

Mise en œuvre du framework Fastify

L'enseignant souhaite pouvoir accéder à distance aux notes de ses élèves via une application web.

Il prévoit d'utiliser le framework **Fastify** pour développer une **API REST** qui transmettra des données au format **JSON**.

Création d'un nouveau projet

- Dans **VS Code**, utilisez le menu : **File** > **Open Folder...** ou Ctrl+K Ctrl+O
- Naviguez jusqu'à votre **dossier de travail global** (celui qui contient déjà les projets précédents).
- Créez-y un nouveau dossier nommé `t1-web`
- Sélectionnez-le pour l'ouvrir dans **VS Code**.

Installation du générateur

- Dans **VS Code**, ouvrez la vue **Terminal** : Ctrl+ù
- Exécutez la commande :
  ```
  pnpm add -g fastify-cli
  ```
- Dans la vue **Explorer** (à gauche), observez que le dossier `node_modules` n'a pas été créé.

Le package **fastify-cli** a été stocké ailleurs.

Cela vient de l'option `-g` qui a indiqué à **PNPM** de faire une installation globale (partagée par tous les projets) et non une installation locale au projet courant.

- Dans la vue **Terminal**, exécutez :
  ```
  explorer $env:LOCALAPPDATA\pnpm
  ```

Cela vous amène dans le dossier de travail de **PNPM** qui se trouve dans votre profil **Windows**.

- Dans le sous-dossier `bin`, se trouvent des scripts exécutables parmi lesquels il y a `fastify.CMD` que vous allez bientôt utiliser
- Il y a aussi un sous-dossier `global` dans lequel sont stockés les packages installés globalement. En cherchant bien, on pourrait y trouver `fastify-cli` que vous venez d'installer.

Licence Creative Commons BY-SA 4.0 - AMBLARD Emmanuel 2026

Ce n'est pas utile de savoir exactement où il se trouve, car c'est le script `fastify.CMD` qui permet d'exécuter l'utilitaire contenu dans ce package.

En effet, **fastify-cli** n'est pas une bibliothèque à utiliser en tant que dépendances, mais un utilitaire à exécuter en ligne de commande. C'est ce qu'indique l'acronyme **CLI** (**Command Line Interface**).

CLI = utilitaire à exécuter en ligne de commande.

Génération d'un projet Fastify

Fastify est un framework permettant de créer un serveur web.

fastify-cli est l'utilitaire qui permet de générer le code d'un projet **Fastify**.

- Dans la vue **Terminal**, exécutez la commande :
  `fastify generate . --esm`
  NB. Le caractère `.` désigne le dossier courant comme cible du traitement.
  L'option `--esm` demande d'utiliser le standard **ESM** et non **CommonJS**.
- Observez le résultat produit dans la vue **Explorer**

Plusieurs dossiers et fichiers ont été créés. Ils contiennent le code d'une petite application web back-end.

Mais, le dossier `node_modules` n'a pas été créé. L'utilitaire `fastify-cli` a seulement créé des dossiers et des fichiers. Il n'a pas téléchargé, ni installé les dépendances. On ne peut donc pas encore exécuter l'application.

Contenu du fichier package.json

- Ouvrez le fichier `package.json`

Dans la section **dependencies**, vous voyez que plusieurs dépendances sont nécessaires pour exécuter l'application. Il s'agit de différents composants du framework **Fastify**.

La section **scripts** contient 3 items. Celui qui nous sera le plus utile est `dev`, car il permet de démarrer le serveur en mode développement.

Au tout début, la propriété `"type": "module"` indique qu'on utilise le standard **ESM**.

Téléchargement des dépendances

- Dans la vue **Terminal**, exécutez la commande :
  `pnpm install`

PNPM télécharge et installe les dépendances qui sont listées dans le fichier `package.json`.

Le dossier `node_modules` est apparu et il contient les différents composants du framework **Fastify**.

Licence Creative Commons BY-SA 4.0 - AMBLARD Emmanuel 2026

Exécution de l'application web

- Dans la vue **Terminal**, exécutez la commande :
  `pnpm dev`

Cela démarre le serveur en mode développement.

Un message indique que le serveur écoute sur l'**URL** **http://127.0.0.1:3000**

- Dans la vue **Terminal**, faites **Ctrl**+**clic** sur cette **URL** pour ouvrir la page dans le navigateur.

Le résultat affiché doit ressembler à ceci :

```
{"root": true}
```

NB. Si vous avez l'extension **JSON Viewer**, le contenu exact qui a été reçu par le navigateur, peut être affiché en cliquant sur le lien **Données brutes**, en haut de la page.
À côté de lui, le lien **JSON** permet d'avoir une présentation formatée et colorée.

Modification du code

- Dans la vue **Explorer** de **VS Code**, le dossier `routes` contient le code qui est exécuté lorsque le serveur reçoit une requête **HTTP**.
  Dans ce dossier, ouvrez le fichier `root.js`

On voit qu'il contient une fonction qui est exécutée lorsque le serveur reçoit une requête **GET** pour la route `'/'`

```
fastify.get('/', async function (request, reply) {
  return { root: true }
})
```

La fonction retourne l'objet `{ root: true }`. Cet objet est récupéré par **Fastify** qui le transforme en code **JSON** avant de l'envoyer au navigateur.

- Modifiez l'instruction **return** pour qu'elle devienne :
  `return { message: 'Bonjour !' }`
- Enregistrez la modification et observez ce qui se passe dans la vue **Terminal**.

Vous voyez s'afficher les 2 lignes qui indiquent que l'application vient de redémarrer.
Fastify a détecté que le code avait été modifié et il a déclenché le redémarrage du serveur web afin de prendre en compte la modification.

Si votre navigateur est toujours ouvert sur la page **http://127.0.0.1:3000**
il suffit de l'actualiser pour voir s'afficher le nouveau résultat.

```
{"message": "Bonjour !"}
```

Licence Creative Commons BY-SA 4.0 - AMBLARD Emmanuel 2026

Affichage de la liste des élèves

Récupération du code du projet précédent

- Dans le projet `t1-moyennes`, récupérez les fichiers :
  `groupe1.js`
  `jsconfig.json`
  `moyenne.js`
  Placez-en une copie à la racine du projet `t1-web`
- Ouvrez le fichier `groupe1.js`
- Supprimez :
  - Les 2 instructions `import` pour `figlet` et **chalk**
  - Les instructions qui font de l'affichage à la fin du fichier.

  Ainsi, le fichier contient juste ce qu'il faut pour déclarer le tableau `eleves`.
- Au début de l'instruction qui déclare la variable `eleves`, ajoutez le mot `export`
  Ainsi, le fichier est transformé en module **ESM** et le tableau `eleves` pourra être utilisé par le reste de l'application.

Définition d'une nouvelle route

- Dans le dossier `routes`, dupliquez le fichier `root.js` afin de créer un nouveau fichier nommé `moyennes.js`
- Ouvrez ce nouveau fichier.
- Dans la définition de la route, remplacez `'/'` par `'/moyennes'`
- Remplacez : `return { message: 'Bonjour !' }`
  par : `return eleves`
  Utilisez l'autocomplétion pour générer l'instruction `import` pour `eleves`
- Dans le navigateur web faites afficher le résultat de l'**URL** :
  `http://[::1]:3000/moyennes`

Choisissez l'onglet **JSON** et cliquez sur le lien **Tout réduire**.
L'affichage doit ressembler à :

```
{ "prenom": "Aline", "notes": (3)[…] }
{ "prenom": "Brice", "notes": (3)[…] }
{ "prenom": "Cathy", "notes": (3)[…] }
{ "prenom": "Diego", "notes": (3)[…] }
```

Licence Creative Commons BY-SA 4.0 - AMBLARD Emmanuel 2026