# Express JS

## Introduction

Express.Js est un framework minimaliste pour Node.js. Il permet la création **simple** et **rapide** d'application web, ainsi que des API.


## Projet web

Dans le but de comprendre le fonctionnement d'Express.js, développez un site de quelques pages sur le thème de votre choix.

## Job 1

Créer un dossier nommé "**runtrack_express**", ainsi que deux sous-dossiers "**projet_web**" et "**api**".

Ouvrez votre éditeur de texte préféré et dans le sous-dossier "**projet_web**" initialisez un nouveau projet Node.js et installez Express. Si tout s'est bien passé, vous devriez avoir un fichier nommé "**package.json**" avec une entrée indiquant la présence d'Express JS.

```
{
  "name": "Projet_web",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.19.2"
  }
}
```

## Job 2

Créer un fichier "**index.js**" qui sera le point d'entrée de votre site web. Importer le module nécessaire au bon fonctionnement d'Express et créer une instance de celui-ci.

## Job 3

Créer un **serveur web** sur le port **80**. Ouvrez un navigateur et taper l'URL suivant : HTTP://localhost:80

Vous obtenez une erreur, comme celle-ci ? `Cannot GET /` C'est normal, Express JS va chercher des routes pour afficher les pages de votre site et n'en trouve pas.

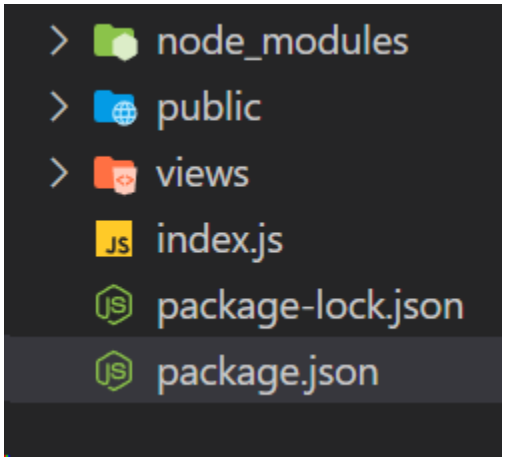
## Job 4

Créez deux routes :

- ➔ "**/**" Qui correspondra à la page d'accueil de votre site
- ➔ "**/about**" Qui présentera votre projet

À l'appel de chaque URL, une description de la page doit être renvoyée.

## Job 5




Créez deux dossiers : "**views**" contenant vos pages HTML et "**public**" contenant vos fichiers de styles et images. Créez deux pages HTML, "**index.html**" et "**about.html**". Ajoutez une structure de base à vos deux pages avec au minimum un titre sur chaque page.

Reprendre votre code pour que vos routes "**/**" et "**/about**" ne renvoient plus un message, mais l'une des pages HTML créées.

## Job 6

Maintenant que vos routes affichent vos pages HTML, ajoutez des images ainsi que du style sur la page index et about.

**Le but de ce job n'est pas de faire de vos pages une œuvre d'art, mais de comprendre le fonctionnement du framework.**

## Job 7

Afin de guider au mieux les utilisateurs, créez une **page 404** qui sera appelée lorsque l'utilisateur tente d'accéder à une URL qui n'existe pas.

## API

Passons maintenant à la création d'une API avec Express.js. Vous aurez besoin de récupérer la base de données **"LaPlateforme"** créée lors de votre **runtrack**.

## Job 8

Initialisez un nouveau projet Node et Express dans le dossier "**api**" et créez un "**index.js**". Importez Express et mettez en place le serveur web..

## Job 9

Votre API doit être composé des routes suivantes :

- ➔ "**/etudiants**" en méthode **GET** : cette route récupère l'ensemble des étudiants de la collection "**student**"
- ➔ "**/etudiant/:id**" en méthode **GET** : cette route récupère un étudiant spécifique en fonction de son **ID**.

Tester vos routes en vous rendant sur chacune des URL dans votre navigateur et vérifier que les données récupérées sont affichées correctement sous forme de JSON.

# Job 10

Créez les routes suivantes :

- ➔ "**/etudiants**" en méthode **POST** : cette route ajoute un étudiant à la collection
- ➔ "**/etudiant/:id**" en méthode **DELETE** : cette route supprime un étudiant de la collection

Votre navigateur préféré n'est pas capable de gérer l'ensemble des requêtes HTTP. Il est limité aux méthodes GET. Renseignez-vous sur les outils tels que **Postman** pour pouvoir tester vos routes.

# Pour aller plus loin sur votre projet web...

Ajoutez à votre site web, une page affichant l'ensemble des données contenu dans la base de données "**LaPlateforme**" créée au jour 4 de votre runtrack.

# Pour aller plus loin sur votre API...

Ajouter l'ensemble des routes du **CRUD** sur la collection "**student**".

# Compétences visées

- ➔ Express JS
- ➔ Nosql
- ➔ API

# Rendu

Le projet est à rendre sur
**https://github.com/prenom-nom/runtrack-nodeJS**.

# Base de connaissances

- ➔ Documentation officielle
- ➔ La structure d'une route
- ➔ Les méthodes de requête HTTP
- ➔ Guide des codes de statut HTTP
- ➔ Postman