Université de Rennes

ESIR ECOLE SUPERIEURE D'INGENIEURS DE RENNES

Compte rendu de WEB

-Backend & Frontend-

ALLAIN Arthur
DE ZORDO Benjamin

I. Les outils de développement Web

Durant les séances de TP Web, nous avons pu mettre en relation les technologies vues en cours. Dans cette partie nous listerons ces technologies en faisant une brève description de chacune d'elles.

- NodeJS

NodeJS est un environnement d'exécution Java Script qui se lance du côté serveur. Son utilisation se fait en ligne de commande de manière simple et possède un mode d'exécution asynchrone (les différentes tâches sont exécutées de façon organisée sans blocage).

- NestJS

NestJS est un Framework utilisé par l'environnement de NodeJS : il permet de créer des applications Web en utilisant une architecture basée sur les composants.
Il nous a permis de créer un squelette de projet pour notre backend et de pouvoir créer différent module de façon simple (en exécutant une commande *nest g module users*).
La puissance de ce Framework réside dans le fait qu'il puisse mettre à jour automatiquement les fichiers de notre projet lorsqu'on crée un nouveau module.

- Angular

Initialement, un projet Web est composé de fichiers HTML permettant l'organisation d'informations, d'une couche d'apparence avec des fichiers CSS et d'une partie logique métier qui est écrit sous forme de fichier JavaScript.
Dans notre cas, nous avons choisi d'utiliser Angular qui est un Framework de développement. En réalité, il utilise le langage Type Script qui se base sur le langage Java Script en ajoutant une surcouche simplifiant l'implémentation.
Ses particularités sont :

- La décomposition de l'application Web en différents composants,
- Facilité la gestion des données et de l'interface (exemple : logique de binding de données).

- Type ORM

Comme dans notre conception, nous avons eu besoin de gérer une base de données, nous avons utilisé le mapper d'objet relationnel Type ORM. Il agit de pair avec le langage Type Script et nous permet de manipuler des objets de notre application sans avoir à utiliser de requêtes SQL.
Sa force réside dans sa simplicité d'implémentation mais aussi sur le fait qu'il ne dépend pas d'un seul langage de base de données (on peut l'utiliser pour du MySQL, du PostgreSQL et d'autres).

II. Le frontend

Le frontend est l'appellation donnée au côté client. Il va permettre de mettre en relation la logique métier du backend avec une interface client.

On aura donc pour chaque module :

- Un fichier Composant en Type Script : il est responsable de la gestion des interactions avec l'affichage. Il va contenir des fonctions à utiliser dans le module et les éléments de bases de données que l'on souhaite afficher.

- Un fichier HTML : Il contient l'architecture des informations sous forme de balises. On pourra également associer des fonctions implémentées dans les composants à des balises comme des boutons.
- Un fichier Spec en Type Script : C'est un fichier généré automatiquement pour la cohérence frontend / backend.
- Un fichier CSS : permet de traiter le côté design du module, mais est très utile pour rendre notre interface agréable à utiliser.

D'autres fichiers sont reconnaissables comme celui de routage (X-routing.module.ts). Celui-ci permet de gérer les différentes vues de l'application et de naviguer entre celles-ci. Il permet également de définir des routes et leur URLs associées, ce qui permet de créer des applications single page qui chargent rapidement, d'ajouter des fonctionnalités de redirections ou des restrictions d'accès (AuthGuard).

III. Le backend

Le backend est l'appellation donnée au côté serveur. C'est l'ensemble des éléments qui permettent à l'application de fonctionner mais qui n'est pas visible par l'utilisateur. C'est dans cette partie que l'on va organiser les données, les stocker et créer l'architecture globale de notre application, notamment la logique métier.

Nous pouvons tester la logique en effectuant des requêtes via un terminal avec l'outil *curl* ou bien utiliser directement l'url de notre backend pour tester les requêtes GET par exemple.

Nous retrouvons les mêmes éléments que sur le frontend avec deux types d'éléments en plus, les Controller et les Service :

- Service : ce sont des objets qui sont conçus pour partager du code et des données entre différents composants de l'application.
  Ce fichier regroupe un ensemble de méthodes retournant des promesses. Ces fonctions vont permettre d'interroger la base de données sur des éléments à obtenir, créer, modifier ou supprimer.
- Controller : ce sont des fichiers qui reprennent les méthodes implémenté dans la partie service et les utilise de façon à créer une interaction entre l'objectif de la méthode et l'interface utilisateur. Ces méthodes sont précédées de décorateurs qui permettent, entre autres, de gérer les requêtes automatiquement.

IV. Le design

Le design est une partie à ne pas négliger puisqu'elle permet à l'utilisateur de se repérer et de lui donner envie d'utiliser notre application.

Dans ce TP, nous avons utilisé des design préconstruit comme celle proposée par Bootstrap ou celles d'Angular (pour les tableaux d'utilisateurs par exemple).

En effet, il est facile de trouver de Template sur internet, néanmoins elles ne correspondent pas toujours à ce que nous souhaitons faire. Nous sommes alors obligés de rajouter une couche de CSS afin de changer certains détails et de passer sur l'inspecteur d'éléments du navigateur afin de voir -visuellement- à quoi correspondent les classes utilisées (puisque nous n'avons pas accès aux paramètres de cette feuille de styles).