Choix technologiques

Plan

I- Choix technologiques

Le projet réalisé dans le cadre du cours d'application n-tiers consistait à réaliser une application de type CRUD pour administrer des demandes de mobilité.

1- Backend

Le backend devait donc pouvoir gérer la base de données de ces demandes de mobilités, la logique métier et une interface pour le frontend. La technologie qui a été choisit pour ce faire est Spring boot.

Objectivement, la raison première de ce choix était ma volonté d'apprendre. En se plaçant dans une situation d'entreprise, Spring boot apporte de nombreux avantages.

Cette framework permet de s'affranchir du SQL. Elle gère automatiquement les interactions avec la base de données. Le développeur manipule des classes qui représentent la base de données. La documentation et la communauté de Spring boot sont très importantes, ce qui est très utile en cas de blocage.

Par comparaison à Spring, Spring boot simplifie davantage en automatisant. Il optimise la gestion des dépendances et s'auto-configure, afin que l'on puisse se concentrer sur la logique métier. Le déploiement est également plus simple, notamment grâce au Tomcat embarqué que comportent les applications Spring boot.

2- Frontend

De nombreuses frameworks existent et permettent là aussi d'automatiser le développement. Les frameworks de front actuellement les plus utilisés sont : React, Vue.js, Angular.

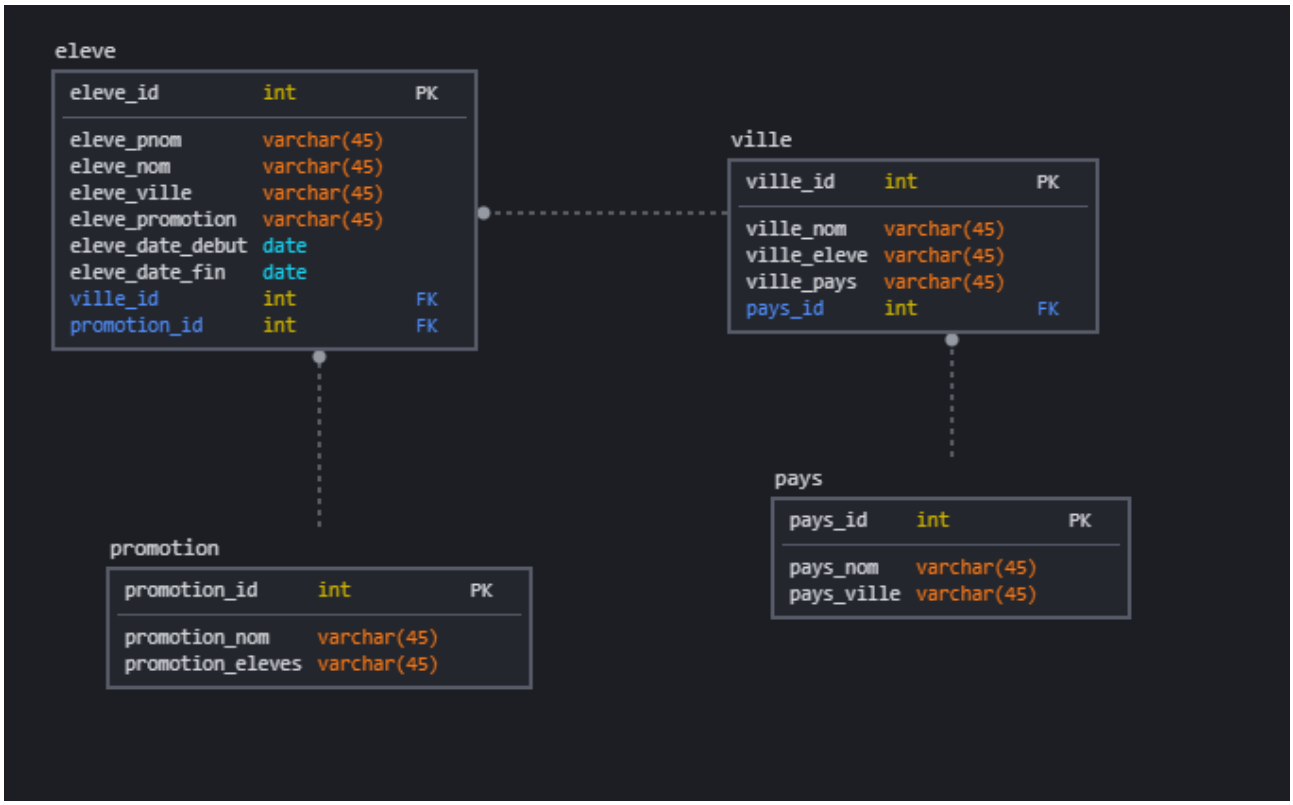
Angular convient généralement pour des applications de tailles importantes (par comparaison à React par exemple). Dans notre mise en situation, nous pourrions imaginer que l'application à vocation à évoluer avec l'arrivée de nouvelles fonctionnalités.

De plus, Angular se base sur Typescript, qui est un langage très puissant en terme de possibilités.

Enfin, Angular « encourage » également à avoir un code bien structuré. Ce qui est préférable pour un futur ingénieur débutant.

II- Architecture du projet

1-Architecture de la base de données



Une demande de mobilité est représentée par un élève, qui est associé à une ville et une promotion. Une ville est elle-même associée à un pays.

Nb : Il est ici supposé que chaque élève ne faisait qu'une seule demande de mobilité et que l'application ne nécessitait aucun système d'authentification pour le moment.

2-Descriptif des packages backend

Package com.tse.ntiers

Ce package est le package de l'application.

Classe NtiersApplication
Il s'agit du point d'entrée de l'application. La fonction main lance l'application Spring.

Package com.tse.ntiers.business

Contient tout les classes de l'application qui structurent les données.

Classes Eleve, Pays, Promotion, Ville
Classes entités représentant les tables de la base de données (identifiées via les annotations Spring boot).

Classe Mobility

Classe représentant une demande de mobilité (qui elle n'est pas une table). Cette classe contient tous les champs d'une demande.

Package com.tse.ntiers.Controller

Contient les contrôleurs via lequel le front peut interagir.

Classe MobilityController
Cette classe contient tous les contrôleurs nécessaires à la gestion des mobilités (ajout, modification, suppression, obtention).

Package com.tse.ntiers.dao

Les classes de ce packages sont des interfaces avec la classe CrudRepository de Spring boot. Les méthodes de cette classe permettent d'interagir avec la base de données.

Package com.tse.ntiers.service

Contient les interfaces de tous les services de l'application.

Package com.tse.ntiers.serviceImpl

Contient les implémentations des services précédemment évoqués.

Classes EleveService, PaysServices, PromotionService,VilleService
Permets d'ajouter, supprimer, ou récupérer un Eleve, un Pays, une Promotion ou une Ville dans la base de données.

Classe MobilityService
Service gérant les demandes de mobilités, ainsi que leur suppression et modification. Les services précédemment évoqués sont tous contenu dans ce service qui orchestre leur fonctionnement.

3-Descriptif des modules frontend

Module app

Module contenant l'application.

Module protected

Contient toutes les « pages » qui dans la perspective d'une évolution futur, seraient protégées par authentification (avec notamment une redirection au niveau du routage et une authentification par jeton au niveau de l'API backend).

Module admin

Sous module de protected. Représente une « page » dédiée à l'administration. Elle va gérer des événements remontant de ses différents composants et les afficher ou non selon le contexte.

Composant liste-mobility
Représente une liste de toutes les mobilités avec des options de suppression ou de modification.

Composant modification
Représente le formulaire de modification d'une demande existante.

Module recherche

Sous module de admin. Avec son composant homologue, le module recherche représente une page permettant de faire des recherches multicritères.

Module demande

Sous module de protected. Représente une « page » dédiée à la demande de mobilité. Comporte un seul composant homologue.

Module home

Sous module de protected. Représente la « page » home. Page avec la carte des mobilités (non fonctionnelle).

Module shared

Module contenant tous les composants et services commun à plusieurs modules.

Service mobility
Objet représentant une mobilité (tous les champs du formulaire).

Service mobility-api
Service gérant tous les appels d'API (interface avec le backend).

Composant navbar
Composant représentant la barre de navigation de l'application. Dans le futur celle-ci pourrait être utilisé dans plusieurs composants différents et non dans tous comme actuellement.

III-Perspectives d’évolution

Fonctionnalités
Conformément au besoin exprimé, la première évolution à venir serait de rendre la carte des mobilités fonctionnelle.

Nous pourrions également mettre un système d’authentification en place. En backend cela demanderait la modification de la base de données actuelle, pour pouvoir stocker les identifiants chiffrés des utilisateurs de l’application. Il faudrait ajouter un service d’authentification en backend et un contrôleur à cette fin. En frontend, le module « public » pourrait alors être créé, contenant entre la page d’authentification vers laquelle serait redirigé les utilisateurs (avec AuthGuard).

Sécurité
Un système de jeton pourrait aussi être assez rapidement ajouté, afin d’empêcher les accès non autorisés à l’API.