RAPPORT PROJET 4A IA2R

Création d’un service web pour la gestion de stocks d’une entreprise

Rédigé par :

ANNA Christianna

NANA Benjamin

Date du document

Sommaire

INTRODUCTION

Pour ce projet, nous avons voulu réaliser un site capable de gérer les produits et les commandes d'une entreprise. Pour pouvoir gérer les produits, il faut posséder un compte. Il est possible d'avoir un compte en tant que client ou en tant qu'employé de l'entreprise (ou administrateur).

Les sites web gérant des commandes sont aujourd'hui de plus en plus utilisés grâce à la fondation de startups ou de petites entreprises personnelles. Ce genre de business a souvent besoin d'un site web. Ainsi, nous avons choisi de travailler sur ce type de service web pour avoir une idée du travail à accomplir.

L'utilisateur peut avoir accès à ses commandes par l'intermédiaire de l'interface graphique. Ce rapport présente les démarches que nous avons dû entreprendre pour réaliser le projet.

I - Objectif du projet et analyse du besoin

1. Enoncé du besoin

Définir le besoin, présenter les différents diagrammes (diagramme de classe, diagramme de cas d'utilisation etc…).

L'objectif de ce projet est de mettre en place un service web permettant à une entreprise commerciale (une entreprise qui fait de l'achat-revente) à gérer ses stocks de marchandises, à gérer les achats auprès de ses fournisseurs ainsi que les ventes auprès des clients. Ainsi, grâce au service web, une entreprise donnée doit avoir la possibilité :

- D'avoir accès à la liste de ses marchandises et les informations relatives à celles-ci (Nom, prix, description, quantité présente en stock), de modifier ces informations à sa guise.
- D'avoir accès à un récapitulatif des achats et ventes effectuées, les marchandises concernées, les prix respectifs ainsi que les quantités.
- De suivre l'évolution de son fond (monétaire)
- De consulter les informations relatives au profil (Nom, adresse)

Ce service doit aussi servir à des clients, et doit leur permettre :

- D'avoir accès à une liste de produits et faire un choix en fonction des prix ou de leurs préférences
- D'effectuer des commandes
- D'avoir accès à une liste des commandes qu'ils ont effectuées

En plus d'un service web, il question de créer une application graphique afin d'exploiter le service web.

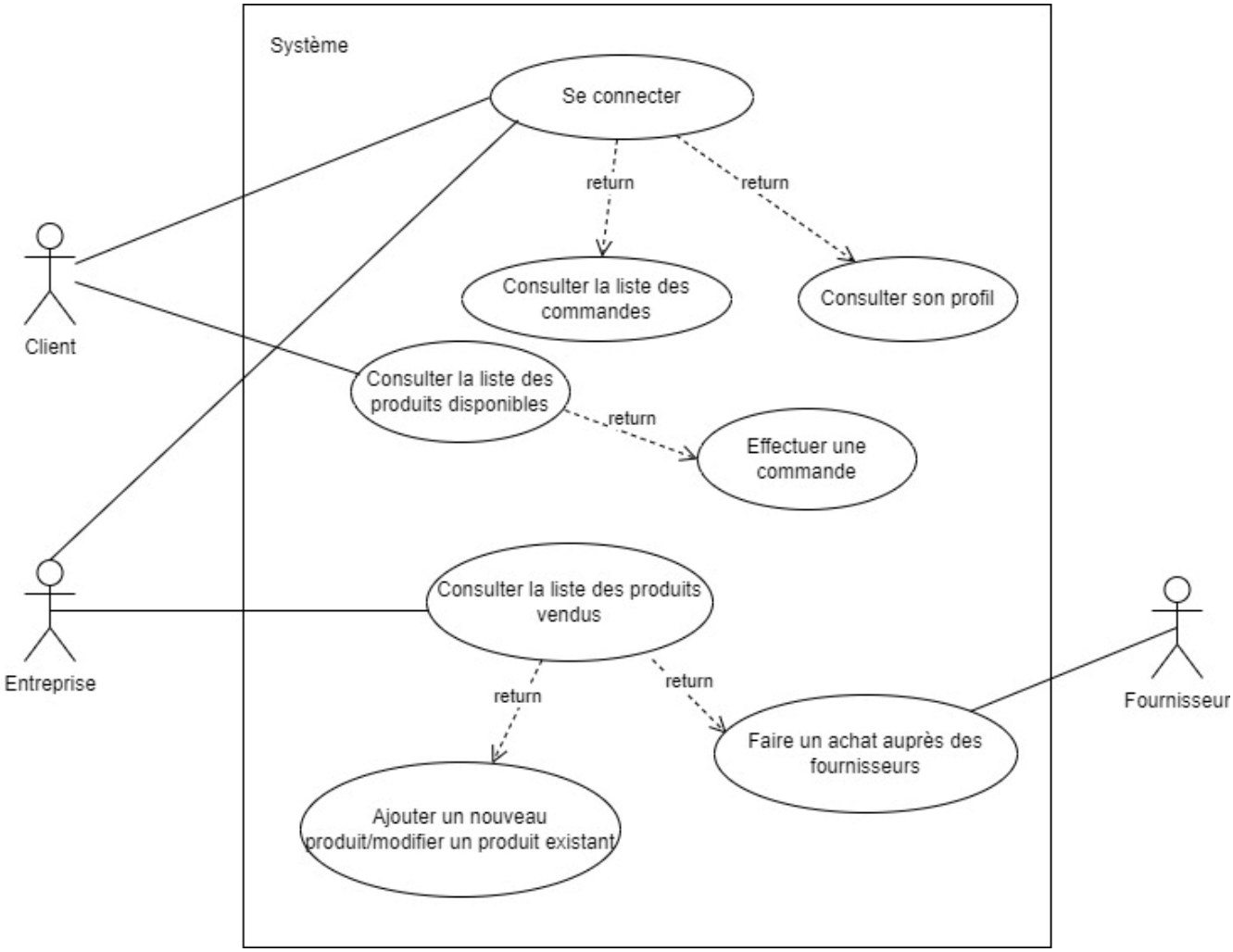
Pour pouvoir proposer une solution convenable à ce besoin, une analyse profonde est nécessaire. Cette analyse sera effectuée au moyen de différents diagrammes.

2. Diagramme de cas d'utilisation

Il s'agit d'un outil de modélisation qui permet de représenter les interactions entre des acteurs et un système afin d'atteindre un objectif spécifique. Pour se faire, nous devons commencer par identifier les différents acteurs.

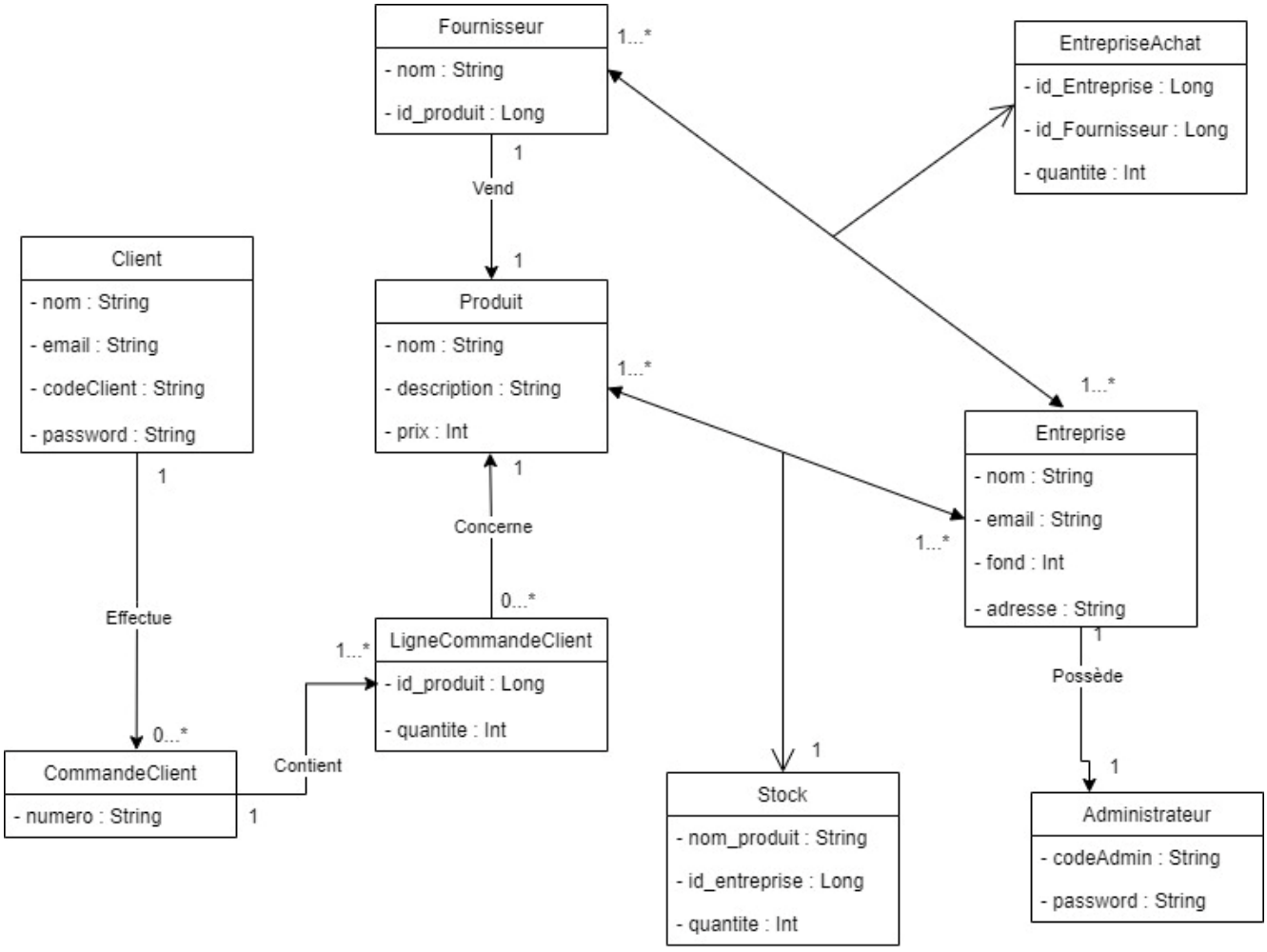
- Une entreprise
- Un client

Passons maintenant au tracé du diagramme



3. Diagramme de classe

Cet outil de modélisation permet de décrire la structure du système en termes de classes et d'objets, et comment ils interagissent entre eux.



II - Technologies utilisées

Pour mettre en place ladite solution, nous nous sommes servis de plusieurs technologies, partagées entre IDE, langages de programmation et architectures. Ainsi, pour mettre en place le service web, nous avons utilisé

- Comme technologie le framework Spring boot
- L'IDE IntelliJ
- Le logiciel XAMP
- Le logiciel Postman qui nous a été d'une grande utilité notamment pour tester des requêtes HTTP
- Comme langage de programmation, le langage JAVA, avec des bibliothèques telles que Lombok qui nous faisaient gagner en temps (Lombok génère automatiquement les constructeurs, getters et setters)

Pour ce qui est de l'application graphique :

- Comme technologie, le framework Angular
- L'IDE Visual Studio Code
- Comme langage, le TypeScript, HTML, CSS, JavaScript,

III - Réalisation

Expliquer comment se structure chaque partie (backend, frontend), expliquer le contenu ? présenter les résultats (captures d'écran) ?

La solution proposée est organisée en deux grandes parties : une partie back-end (le service web) une partie front-end (l'application graphique). Ces deux parties savent fonctionner indépendamment l'une de l'autre.

1. Réalisation de la partie back-end

Comme précédemment cette partie est un projet spring boot. Nous avons d'abord généré la structure minimale du projet grâce au starter spring boot Spring Initializr, que nous avons complété au fil du temps.

Un service web est un service informatique qui utilise des protocoles de la couche application d'internet, principalement http, pour fournir des fonctionnalités accessibles via un réseau. Ces fonctionnalités sont accessibles grâce à une méthode http (les plus utilisées sont GET, POST, PUT, DELETE) et l'URL d'un endpoint (Exemple : http://).

Ainsi, il était question pour nous mettre en place une base de données, de définir les méthodes http et les endpoints par lesquels seront accessibles les informations. Pour cela, le projet a été organisé en quatre couches principales qui sont la couche model, la couche controller, repository, service, chacune ayant un rôle spécifique à jouer

a) La couche model

Dans cette couche on crée toutes les classes JAVA qui interviendront dans le fonctionnement du système (voir diagramme de classe ci-dessus). Cette couche permet aussi d'organiser la base de données. En effet, après avoir crée la base de données à l'aide de XAMPP, on peut dans cette couche définir les tables. Ceci est possible grâce aux annotations telles que @Entity et @Table qui permettent de de définir des classes Java comme des tables et leurs attributs comme des colonnes de table de base de données. Ces annotations sont introduites par JPA (Java Persistence API) et servent d'interface entre la base de données et l'application.

b) La couche controller

C'est cette couche qui permet de définir les méthodes http ainsi que les URL des endpoints par lesquels seront accessibles les informations. Il existe autant de controllers que de model (un controller par model).

Pour une fonctionnalité donnée, dans chaque controller, on définit une expression qui comporte

- Une instance du Service correspondant (voir couche service)
- Une annotation qui précise le verbe http ainsi que l'URL
- Une fonction qui appelle une autre fonction (créée dans le service) et qui s'occupera d'accéder/modifier les données, selon le verbe http

c) La couche service

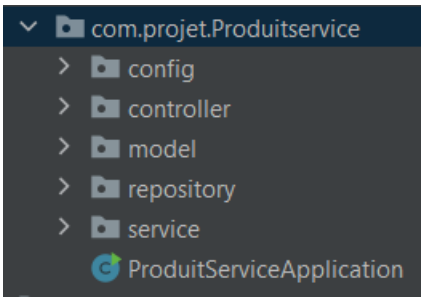
Cette couche permet d'implémenter le traitement métiers spécifiques à l'application. C'est dans celle-ci qu'on effectue les différentes opérations sur les objets java. Tout comme avec les controllers, il y'a un service par model et chaque service contient des fonctions qui sont des ensembles d'opérations sur l'objet java correspondant. Un service contient aussi une instance du repository correspondant.

d) La couche repository

Cette couche permet d'effectuer des opérations sur la base de données.

Tout comme pour les autres couches, il y'a un repository par model. Chaque repository effectue des opérations sur la table de base de données du model correspondant.

En dehors de ces quatre couches, une couche de configuration a été mise en place. Celle-ci contient une classe WebClientConfig qui sera appelé dans un service donné, pour que celui-ci puisse faire appel à un autre service en passant par l'url correspondant.



Pour exploiter le service web mis en place, il a fallu créer une application graphique (front-end)

2. Réalisation de la partie front-end

Le front-end de notre projet est un projet Angular organisé en plusieurs composants, un composant pour un élément ou alors un ensemble d'éléments graphique.

Le projet contient un dossier racine appelé app, et c'est dans celui-ci que nous plaçons tous les éléments que nous créons. Ce dossier app est généré de façon automatique et contient :

- Un fichier html pour structurer l'élément graphique
- Un fichier css pour appliquer du style au composant graphique
- Un fichier TypeScript
- Un fichier spec.ts

Chaque nouveau composant a cette structure là. A la création d'un composant, Angular génère automatiquement un sélecteur qui va permettre d'appeler ce composant dans un autre plus grand (composant parent).

En plus de ces fichiers, le dossier app contient un module (app.module.ts) dans lequel il est nécessaire de spécifier tous les composants utilisés, et qui permet de définir une système de routes. Ce système de route n'a rien à voir avec celui du back-end et c'est par celui-ci que se fera désormais l'accès aux fonctionnalité.

Pour avoir accès aux services offerts par le service web, nous avons créé un service nommé gestion-de-stock représenté par le fichier gestion-de-stock.service.ts. Il représente un objet qui a un attribut de type HttpClient et contient des fonctions qui retournent les réponses http envoyées par le service web. Ces fonctions précisent l'url de l'endpoint ainsi que la méthode http et sont appelées au sein du projet afin d'afficher les informations obtenues du service web.

CONCLUSION

Pour conclure, nous avons intégré les fonctionnalités qui nous semblaient nécessaires dans une entreprise gérant des commandes et produits. Nous sommes passés par spring boot pour la réalisation de la partie back, elle est composée de quatre couches à part la couche config. Le front est quant à lui réalisé sur angular, parfois nous nous sommes servis d'angular material.

Nous avons également rencontré quelques difficultés, par exemple, la mise à jour des pages du coté front end. Lorsqu'on modifie le code du front, les modifications ne s'affichaient pas toujours. En ce qui concerne la partie back, la communication entre les services étaient difficiles à réaliser.

ce projet nous a permis d'acquérir de nouvelles connaissances et d'apprendre à nous servir d'outils de programmation tels que angular, intelliJ, postman, …