Rapport TFE

Groupe 03

D'haeyere Corentin, Fuentes Gonzalez Lucas, Johnen Thomas,
Thomas Loïc, Vandermeersch Laurent

D'haeyere Corentin
Fuentes Gonzalez Lucas
Johnen Thomas
Thomas Loïc
Vandermeersch Laurent

Table des matières

D'haeyere Corentin
Fuentes Gonzalez Lucas
Johnen Thomas
Thomas Loïc
Vandermeersch Laurent

Introduction :

Pour ce projet, nous avons choisi Angular comme technologie pour notre front-end. Son déploiement se fait via l'utilisation de Docker, où l'application est compilée puis exécutée dans un conteneur Nginx. En complément, nous utilisons des bibliothèques telles que @angular/material, qui offre une série de composants prédéfinis pour faciliter le développement de l'interface utilisateur, ainsi qu'une bibliothèque pour rendre l'application PWA, conformément à nos besoins. Pour assurer la communication entre le back-end et le front-end, nous utilisons une API REST.

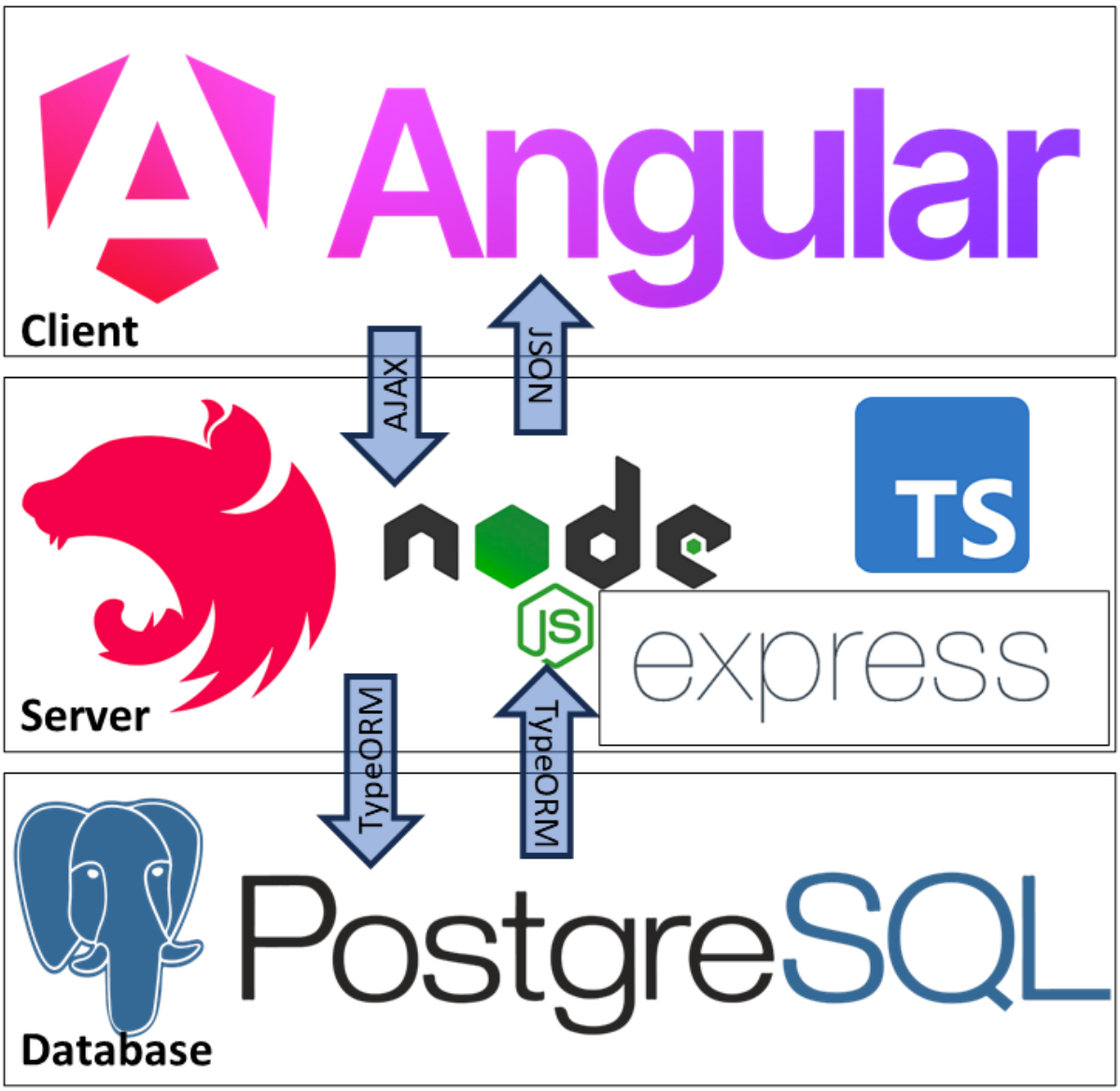
Pour notre back-end, nous avons opté pour Nest JS. Son déploiement a également été réalisé via Docker, où l'application est compilée puis exécutée avec Node.js. Nous utilisons un modèle de programmation orientée composants pour simplifier la création d'applications robustes et évolutives.

Ces applications sont déployées sur un VPS (serveur privé virtuel) Oracle, où chaque application dispose de deux images distinctes dans Docker : une pour le développement (tag "dev") et une pour la production (tag "latest"), assurant ainsi une gestion différenciée des environnements.

En ce qui concerne la base de données, nous avons opté pour PostgreSQL. Nous utilisons deux bases de données distinctes : une pour le développement et une pour la production, toutes deux fournies par l'école. La liaison entre le back-end et la base de données est gérée par un ORM appelé TypeORM, simplifiant ainsi l'accès et la manipulation des données au sein de l'application.

D'haeyere Corentin
Fuentes Gonzalez Lucas
Johnen Thomas
Thomas Loïc
Vandermeersch Laurent

Diagramme d'architecture :



Rectangle 1 - Client (Angular) :

Angular : C'est le framework front-end développé par Google. Il facilite la création d'interfaces utilisateur dynamiques et réactives pour les applications web.

Communication avec le Serveur : Le client Angular communique avec le serveur à l'aide d'Ajax (Asynchronous JavaScript and XML), une technique qui permet des requêtes asynchrones côté client sans nécessiter un rechargement complet de la page.

Échange de Données : Les données échangées entre le client et le serveur sont au format JSON (JavaScript Object Notation), un format léger et facile à lire qui facilite le transfert de données structurées.

D'haeyere Corentin
Fuentes Gonzalez Lucas
Johnen Thomas
Thomas Loïc
Vandermeersch Laurent

Rectangle 2 - Serveur (Nest JS, Node JS, Express, TypeScript) :

Node.js : Une plateforme côté serveur basée sur le moteur JavaScript V8 de Chrome, permettant d'exécuter du code JavaScript côté serveur.

Express : Un framework web minimaliste pour Node.js qui simplifie la création d'applications web et la gestion des routes.

Nest JS : Un framework back-end progressif construit avec TypeScript et inspiré par Angular. Il offre une structure modulaire et facilite le développement d'applications évolutives.

TypeScript : Un sur-ensemble de JavaScript qui ajoute des fonctionnalités de typage statique. Cela améliore la maintenance du code et permet une meilleure gestion des erreurs potentielles.

Communication avec la Base de Données : Le serveur communique avec la base de données à l'aide de TypeORM, une bibliothèque ORM (Object-Relational Mapping) pour TypeScript et JavaScript.

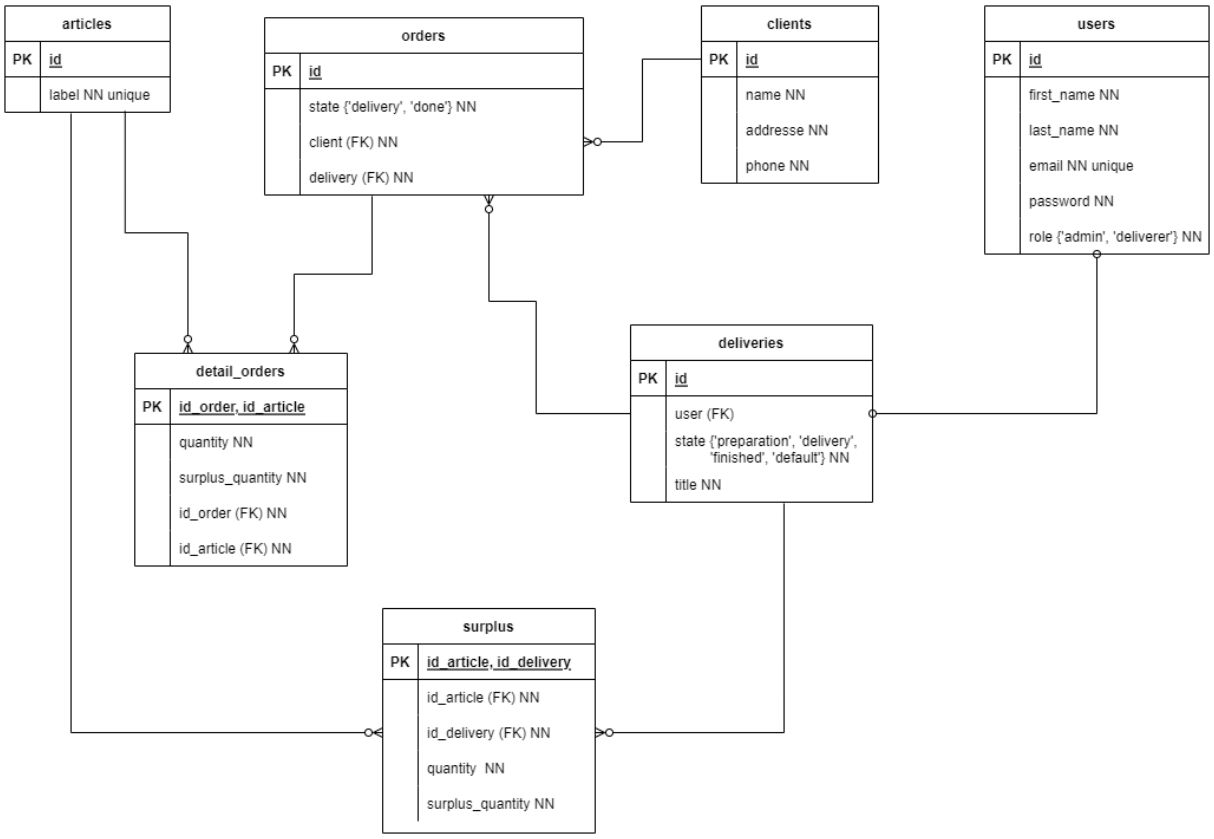
Rectangle 3 - Base de Données (PostgreSQL) :

PostgreSQL : Un système de gestion de base de données relationnelle open-source. Il est réputé pour sa fiabilité, son extensibilité et son respect des normes SQL. C'est le stockage persistant des données de l'application.

D'haeyere Corentin
Fuentes Gonzalez Lucas
Johnen Thomas
Thomas Loïc
Vandermeersch Laurent

2 Diagrammes au choix :

DSD :



Utilisateurs (users) :

Chaque utilisateur est identifié par un id unique.

Les informations de base sur un utilisateur sont stockées, telles que son prénom (first_name), nom de famille (last_name), adresse e-mail (email), mot de passe (password), et rôle dans le système (role) qui peut être soit *admin* soit *deliverer*.

Il existe une relation entre la table deliveries et users : chaque tournée est associée à un utilisateur via la clé étrangère user.

Tournées (deliveries) :

Chaque tournée est identifiée par un id unique.

La table enregistre des détails tels que l'état de la tournée (state), les états possibles sont : *preparation*, *delivery, finished*, *default*. Elle enregistre également son titre (title).

La relation avec la table users permet de lier une tournée à un utilisateur précis.

D'haeyere Corentin
Fuentes Gonzalez Lucas
Johnen Thomas
Thomas Loïc
Vandermeersch Laurent

Clients (clients) :

Chaque client est identifié par un id unique.

Les informations du client incluent son nom (name), adresse (address), et numéro de téléphone (phone).

Il existe une relation avec la table orders via la clé étrangère client, liant chaque commande à un client spécifique.

Articles (articles) :

Chaque article est identifié par un id unique.

La table enregistre le libellé de chaque article (label).

Commandes (orders) :

Chaque commande est identifiée par un id unique.

Les détails de la commande comprennent son état (state), les états possibles sont : *delivery*, *done*. Ils comprennent aussi le client associé via la clé étrangère client, et la tournée associée via la clé étrangère delivery.

Les commandes sont liées aux articles via la table detail_orders.

Détails des Commandes (detail_orders) :

Cette table agit comme une table de liaison entre les commandes (orders) et les articles (articles).

Chaque enregistrement représente une quantité commandée d'un article spécifique, avec des détails tels que la quantité principale (quantity) et une quantité supplémentaire (surplus_quantity).

Les clés étrangères id_order et id_article référencent respectivement les commandes et les articles associés.

Surplus pour une tournée (surplus) :

Cette table agit comme une table de liaison entre les tournées (deliveries) et les articles (articles).

Chaque enregistrement permet de savoir quelle quantité supplémentaire il y a pour une tournée.

Les clés étrangères id_delivery et id_article référencent respectivement les tournées et les articles associés.

D'haeyere Corentin
Fuentes Gonzalez Lucas
Johnen Thomas
Thomas Loïc
Vandermeersch Laurent

Relations clés :

Les utilisateurs sont liés aux tournées par l'intermédiaire de la clé étrangère user.

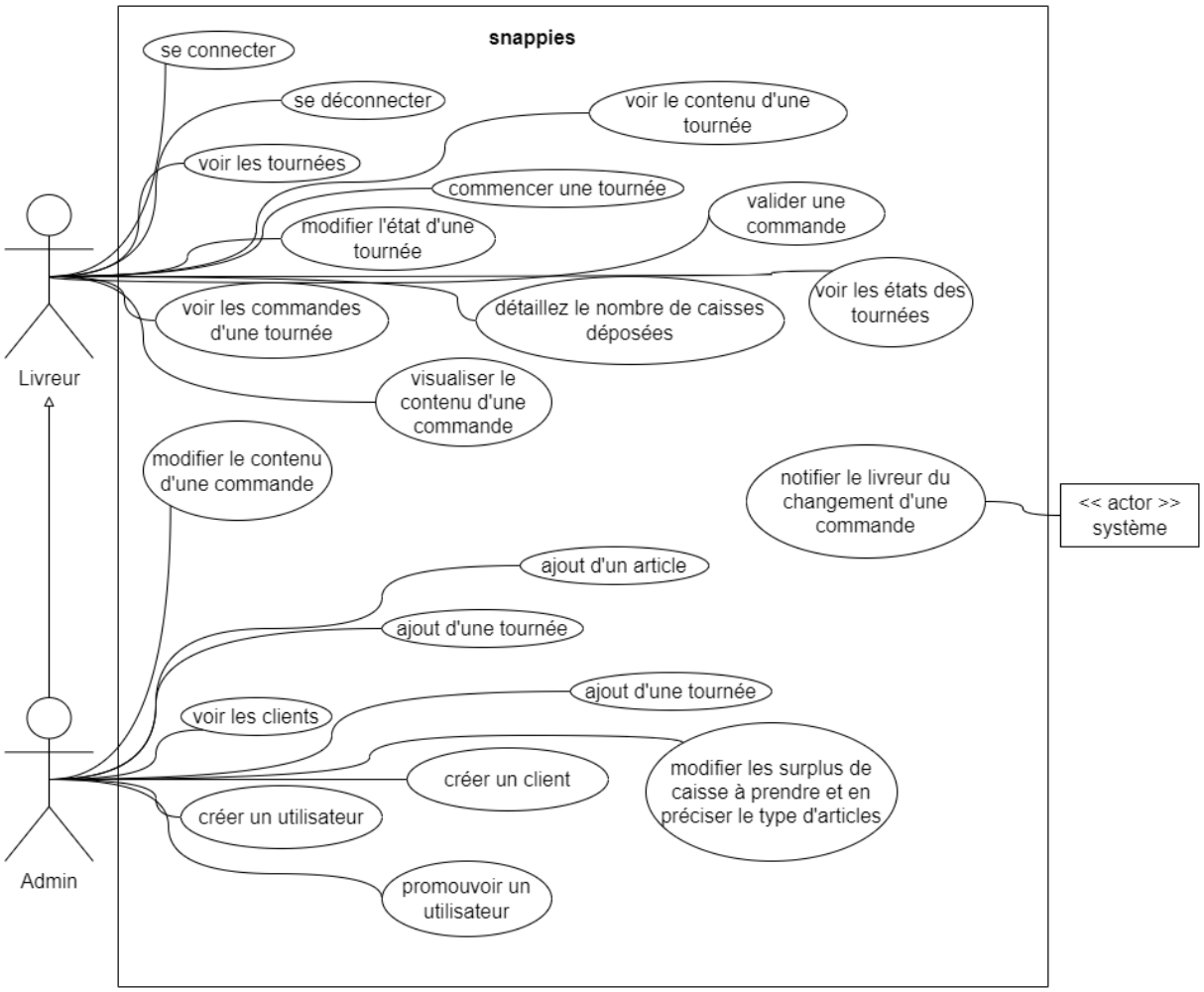
Les commandes sont liées aux clients et aux tournées par les clés étrangères client et delivery.

Les détails des commandes relient les commandes aux articles via les clés étrangères id_order et id_article.

Les surplus relient les tournées aux articles via les clés étrangères id_delivery et id_article.

Ces relations permettent une représentation complète des interconnexions entre les entités de votre système, assurant l'intégrité des données et facilitant les requêtes complexes.

Diagramme des cas d'utilisations :



Le diagramme des cas d'utilisation ci-dessus permet une représentation visuelle des fonctionnalités ou interactions que les utilisateurs peuvent avoir.

Les trois acteurs de ce diagramme sont les admins et les livreurs. Les admins ont accès à leurs fonctionnalités ainsi qu'à celles des livreurs. Nous avons également le système qui se chargera de notifier le livreur qu'une des commandes dont il est en charge a été modifiée.

État d'avancement et améliorations possibles :

Nous avons réussi à implémenter une grande majorité des fonctionnalités que le client nous avait demandées.

Nous pourrions améliorer la façon dont le livreur est informé des modifications apportées à une commande en lui envoyant une notification sur son téléphone. Jusqu'à présent, nous nous contentons de mettre en évidence la commande modifiée.

Nous pourrions également améliorer la manière dont nous stockons les informations afin d'en garder un historique et de potentiellement en tirer des statistiques.

Nous ne permettons pas la modification des utilisateurs, des clients et des tournées. Cela pourrait être une possibilité d'ajout dans le futur.

Pour le formulaire d'enregistrement d'une commande, nous avons opté pour une présentation simple et efficace en raison du délai de livraison relativement court. Par conséquent, le client doit saisir à plusieurs reprises les informations d'identification pour ajouter un élément à la commande. Si nous avions disposé de plus de temps, nous aurions probablement mis en place un formulaire en deux étapes :

- Les informations générales de la commande pour le client, avec la possibilité de cocher les éléments qui doivent en faire partie.
- Un deuxième formulaire où l'utilisateur spécifie les quantités pour chaque élément choisi précédemment.

Cette approche en deux étapes aurait permis une expérience utilisateur plus fluide et une saisie plus efficace des détails de la commande.

D'haeyere Corentin
Fuentes Gonzalez Lucas
Johnen Thomas
Thomas Loïc
Vandermeersch Laurent

Répartition des tâches et organisation du groupe :

Déploiement du front, back ainsi que la DB : Corentin

Développement de back : Corentin / Lucas

Développement du front : Corentin / Laurent / Loïc / Lucas / Thomas

Développement de la DB : Loïc

Écriture du rapport : Corentin / Laurent / Loïc / Lucas / Thomas

Choses utiles et intéressantes :

Notre application est basée sur une librairie visuelle qui nous permet de changer très rapidement et facilement les couleurs de l'application. Il nous suffit d'ajouter un fichier css pour changer les couleurs partout dans l'application.

Difficultés importantes rencontrées :

Problème de rechargement continu de la page pendant 30 secondes lors de la mise en place du PWA.

Déploiement sur azure impossible. Nous avons été obligés de passer à une autre solution.

Lien vers le(s) dépôts de code :

Organisation : https://github.com/pfe-2023-group-03

Front-end:

- Code : https://github.com/pfe-2023-group-03/snappies-front
- Dev : http://141.145.193.169:8080/
- Prod : http://141.145.193.169/

Back-end :

- Code : https://github.com/pfe-2023-group-03/snappies-back
- Dev : http://141.145.193.169:3001/
- Prod : http://141.145.193.169:3000/

Conclusion:

En conclusion, ce Projet de Fin d'Études a été une opportunité passionnante et enrichissante pour notre équipe. Nous avons choisi des technologies modernes et puissantes pour le développement de l'application, avec Angular pour le front-end, Nest JS pour le back-end, et PostgreSQL comme base de données.

Le déploiement via Docker a facilité la gestion des environnements de développement et de production, assurant une cohérence et une stabilité accrues. L'utilisation d'API REST a facilité la communication entre le client et le serveur, garantissant une expérience utilisateur fluide.

Bien que nous ayons réussi à implémenter la majorité des fonctionnalités demandées par le client, des pistes d'amélioration subsistent. Notamment, l'ajout d'une fonctionnalité de notification en temps réel pour les livreurs, un historique plus complet des données, et la possibilité de modifier les utilisateurs, clients, et tournées.

Nous sommes fiers du résultat obtenu et du chemin parcouru tout au long de ce projet. Les liens vers nos dépôts de code sont fournis pour ceux qui souhaitent explorer davantage notre travail. Ce PFE représente une étape significative dans notre parcours académique et professionnel, marquant la fin d'une étape et le début de nouvelles opportunités passionnantes.

Documentation et apprentissage :

Nous trouvons essentiel de souligner l'importance de la documentation et de l'apprentissage que nous avons fait durant ce projet. Donc, voici quelques ressources qui nous ont été utiles :

- Angular documentation : https://angular.io/
- Angular Material: https://material.angular.io/
- ChatGPT : https://chat.openai.com/
- Pipeline : https://ochoquet.be/syllabusAdminInfra/Syllabus_AdminInfra.html
- Progressive Web App : https://www.positronx.io/build-progressive-web-app-pwa-with-angular/
- Nest JS documentation : https://docs.nestjs.com/
- TypeORM Documentation : https://typeorm.io/#/