Université Grenoble Alpes

Rapport Projet BDI

GroMed
MÉDICAMENTS POUR
ÉTABLISSEMENT DE SANTÉ

Ilyass BOUISSA

Iris FATELA LE ROY

Rayan GUERBAA

Emerik MEASSON

Février 2023

Sommaire

Sommaire 2

Introduction 3

Organisation 3

Initialisation 3

Réalisations 4

- Les fonctionnalités réalisés 4
- Les fonctionnalités non réalisés 4

Les choix 5

- Base de données 5
- IHM 5
- Patron et Composant 6

Les changements 7

Les difficultés et les limites de l'application 7

Conclusion / Bilan 8

Annexes 9

- UML JPA 9
- Modèle de tâche 9
- BPMN 10

Introduction

GroMed est une application de vente de médicaments destinée aux établissements de santé. Notre objectif est de proposer une application de commerce en ligne répondant au mieux aux besoins de la société GroMed Inc. (Groland Medical Inc).

Leurs besoins étaient d'avoir une application efficace et efficiente permettant de gérer des comptes, de rechercher des produits par différents critères et de passer des commandes à l'aide d'un panier. Nous devons donc proposer une solution facile et rapide à utiliser.

Organisation

Avant de commencer le projet, nous avons eu une phase importante de modélisation.
Cette partie, nous l'avons réfléchie et faite avant le début du projet, de façon à avoir un maximum de temps pour le développement.

Ensuite, nous avons eu une phase de 2 semaines de développement.
Nous avons fait les fonctionnalités dans l'ordre en fonction de leur priorité. En premier, nous avons fait l'authentification, puis la recherche avec l'ajout au panier et la validation du panier, enfin la gestion des commandes types.
Nous nous sommes réparti les tâches principalement en termes de back-end et de front-end. Au début du projet, nous avons mis l'accent sur le back afin d'avoir un mapping correct, puis nous avons fait en parallèle le développement du front et celui du back de tel sorte à ce que l'équipe sur le back implémente les services et les données nécessaires pour que l'équipe front puisse implémenter les fonctionnalités principales. A la fin, nous avons dû mettre l'accent sur le front pour arriver à terminer à temps.

Initialisation

Pour se connecter sur l'application, un compte existe déjà :

- Son identifiant : 5
- Son mot de passe : 1234

Nous avons deux dépôts github, l'un pour le back-end et l'autre pour le front-end :

- https://github.com/projetG3/projetFront
- https://github.com/projetG3/projetBack

Réalisations

Les fonctionnalités réalisés

Les fonctionnalités réalisées sont :

- L'authentification,
- La recherche de produit,
- L'ajout au panier d'un élément,
- La création d'une commande type,
- L'ajout d'une commande type,
- L'affichage des commandes type,
- La validation du panier.

La recherche peut se faire à partir d'un nom de médicaments, d'un générique, d'un libellé d'une présentation.
L'utilisateur peut modifier la quantité qu'il souhaite d'une présentation sans forcément ajouter cette présentation dans le panier.
Lors de la validation d'un panier, s'il n'y a pas suffisamment de stock, l'utilisateur a la possibilité d'annuler sa validation. Après la validation, l'utilisateur peut ajouter sa commande aux commandes-type, en ajoutant un nom.
Lorsqu'un utilisateur a un panier et qu'il se déconnecte, son panier est sauvegardé, ce qui veut dire que quand il revient son panier est toujours là avec les présentations qu'il avait ajouté.

Les fonctionnalités non réalisés

Les fonctionnalités que nous souhaitions réaliser, mais que nous n'avons pas eu le temps :

- La modification du panier,
- La création de compte,
- Le back-office de GroMed,
- La prise en compte des différentes alertes.
- L'affichage des détails des produits présents dans une commande type,

Les choix

Base de données

- **Conception / UML :** nous avons fait un UML pour nous aider à faire le mapping.

- **Mapping :** pour certaines données nous avons pu les mettre en lazy lorsque c'était du ManyToOne, pour le reste nous avons dû les laisser en eager. Dans un premier temps, nous avons fait le choix d'éviter au maximum les listes, avec un mapping très proche de la base de données. Cependant, nous avons changé le mapping pour qu'il soit plus simple et plus logique en termes de métier. Par exemple, au départ, une commande connaissait le compte qui l'avait créé. Mais il est logique aussi de connaître les commandes d'un compte. Nous avons donc modifié le mapping pour qu'un compte connaisse ses commandes facilement, et inversement. (voir l'UML en annexe)
- **Nettoyage de données :** nous avons supprimé les présentations n'ayant pas de prix, les présentations avec un format de prix particulier.
- **Performance :** nous avons réalisé les "select" en requêtes préparées, afin de gagner en performance, vu que nous perdons déjà du temps à cause du mapping en eager.

IHM

Concernant notre IHM, on s'est tout d'abord appuyés sur le modèle de tâches qu'on a réalisé pour le premier rendu d'IHM. Globalement, on a bien respecté la charte graphique malgré quelques changements apportés sur des petits détails par exemple :

- **La bulle de notification du panier en rouge :**
  Pour l'information que comporte cette couleur. En effet, le rouge est considéré comme une couleur d'alerte et est souvent associé à l'urgence ou à une situation importante. Par conséquent, il est souvent utilisé pour les notifications pour attirer l'attention de l'utilisateur sur une mise à jour importante ou une action qu'ils doivent effectuer.
  Elle s'inscrit dans l'heuristique de cohérences et standards. (normes)

- **Le sous-menu et l'arrière-fond grisâtre :**
  Pour la lisibilité du menu et ainsi dissocier le menu et le sous-menu. Elle s'inscrit dans l'heuristique ergonomie de visibilité de l'IHM, si tout le haut de la page est de la même couleur cela risquerait de déranger l'utilisateur sur l'utilisation des menus qu'il ne pourra pas différencier.

- **Aspect responsive :**
  Nous avons décidé de ne pas s'attarder sur l'aspect responsive du site demandé par le client faute de compétences techniques sur cette partie et également du temps de travail très restreint. De plus, on a analysé le besoin réel du client qui souhaitait une

application pour les professionnels et donc nécessitant une sécurité particulière. Or, le fait d'avoir une application responsive avait comme seul avantage de pouvoir faire tourner le site sur téléphone. Le plus sécurisé étant de commander sur l'ordinateur de travail.

- **L'affichage des présentations :**
  On a choisi d'utiliser une pagination pour afficher nos résultats de recherche avec 9 petites cartes par page affichant les informations importantes des présentations. Nous pouvons cliquer sur une des cartes pour afficher ses détails avec un menu accordéon pour les informations du médicament et les informations supplémentaires. Cela permet à l'utilisateur de ne pas avoir trop de choix proposé devant lui, nous aurions aimé ajouter des filtres de recherches pour faciliter ses choix dans les résultats, mais nous n'avons pas eu le temps de l'implémenter. Cette heuristique a pour but de faciliter la recherche d'information, sa perception, sa reconnaissance.

- **La gestion du panier :**
  Lorsque que l'on ajoute un produit, un pop-up de validation s'affiche pour notifier la réussite de l'ajout du produit. Cette notification permet d'informer l'utilisateur sur l'état cohérent du système. En plus de cela, nous avons mis un pop-up de confirmation lors de la validation du panier dans le cas où il y aurait des produits du panier avec une quantité nulle. Cela donne le choix à l'utilisateur d'annuler sa commande dans le cas où il ne souhaiterait pas avoir de délai dans sa livraison.

Patron et Composant

Dans notre projet, nous avons utilisé une structure de projet Spring Boot Angular, avec l'utilisation d'Hibernate pour pouvoir faire le Mapping JPA de nos données et l'utilisation de microservices pour pouvoir récupérer les données et les envoyer des API.

Pour cela, nous avons créé des modèles de données similaires côté Back et Front pour pouvoir utiliser les API, des Repository et Services pour implémenter nos méthodes de récupération des données avec pour certaines des requêtes préparées, et des Controllers permettant de faire la passerelle entre le Front et le Back. Nous avons séparé le code métier du reste de façon à séparer les responsabilités, il se trouve dans les services et utilise les modèles pour accéder à la base donnée.

Le fait d'avoir un projet Spring Boot Angular induit le fait d'utiliser un patron MVP par l'utilisation d'Observables dans nos services et composants pour afficher les différents états du panier ou des produits récupérés grâce aux API.

Notre code a été poussé sur Sonar, cela nous a permis de le nettoyer et de corriger des erreurs. Nous avons fait différents types de tests, sur les services, les tests nous ont permis de tester notre mapping. Les tests sur les controllers ont été fait avec des mocks, pour ne pas attaquer la base de données avec les tests.

Les changements

En réalisant l'application, nous nous sommes rendu compte que nous avons besoin d'un attribut supplémentaire dans la classe EstConstitueeDe, afin de savoir si l'ensemble des présentations ont été livrées.
Nous avons également créé des classes FormePharmaceutique, StatusAdministratif, TypeProcedure et EtatCommercialisation avant, c'était des attributs dans la classe Médicament, afin de simplifier cette dernière.

Les difficultés et les limites de l'application

Le mapping nous a causé des difficultés, nous avons passé de nombreux jours dessus, ce qui nous a fait perdre du temps pour réaliser les autres fonctionnalités, que nous souhaitions faire.
Nous chargeons nos données en eager, donc nous chargeons toutes les jointures à chaque fois, ce qui entraîne un chargement des données un peu lent. Il faut donc faire attention au nombre de requêtes exécutées. De plus, de nombreuses informations sont stockées et chargées alors qu'elles ne sont pas utilisées puisqu'elles n'ont pas été implémentées dans le Front.
Notre recherche se fait avec les noms de la présentation, du médicament, du générique et du principe actif. Nous avons décidé d'ajouter, par rapport aux scénarios, le nom des présentations dans la recherche puisque nous vendons des présentations, et que nous pensons qu'ils peuvent être présents sur les ordonnances. Pour les utilisateurs, cela peut être un gain de temps. Nous n'avons pas eu le temps de rajouter un filtre, afin de n'afficher que les présentations qui sont en stock.
Certains médicaments de la base de données n'ont pas de présentations, nous aurions pu poursuivre le nettoyage de la base de données de telle sorte que cela n'apparaisse pas. Nous

n’avons pas voulu passer beaucoup de temps sur le nettoyage pour en garder sur le développement de l’application.

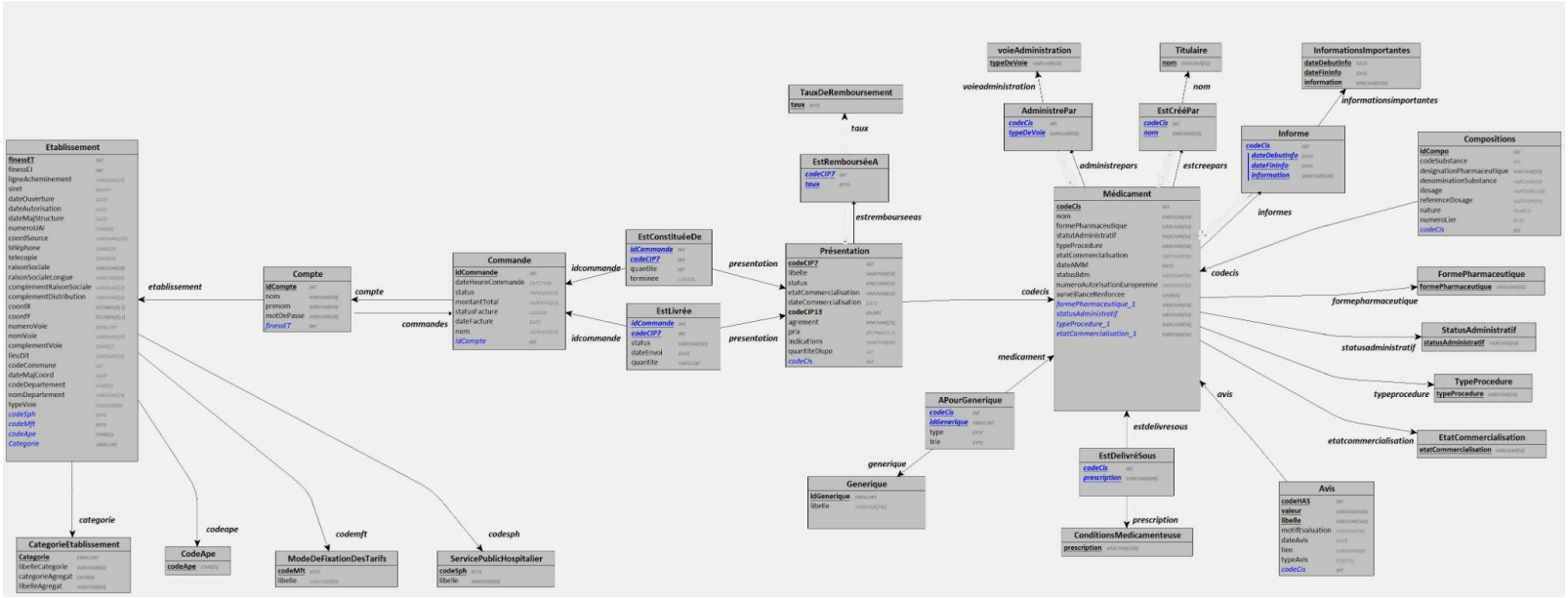
Conclusion / Bilan

Pour terminer, nous pouvons faire une critique sur l’ensemble de notre projet réalisé.
Nous avons réussi à terminer plus ou moins l’incrément 1 avec les fonctionnalités principales fonctionnelles. Il manque tout de même certains détails dans ces fonctionnalités permettant de valider complètement cet incrément, que nous n’avons pas eu le temps d’implémenter.

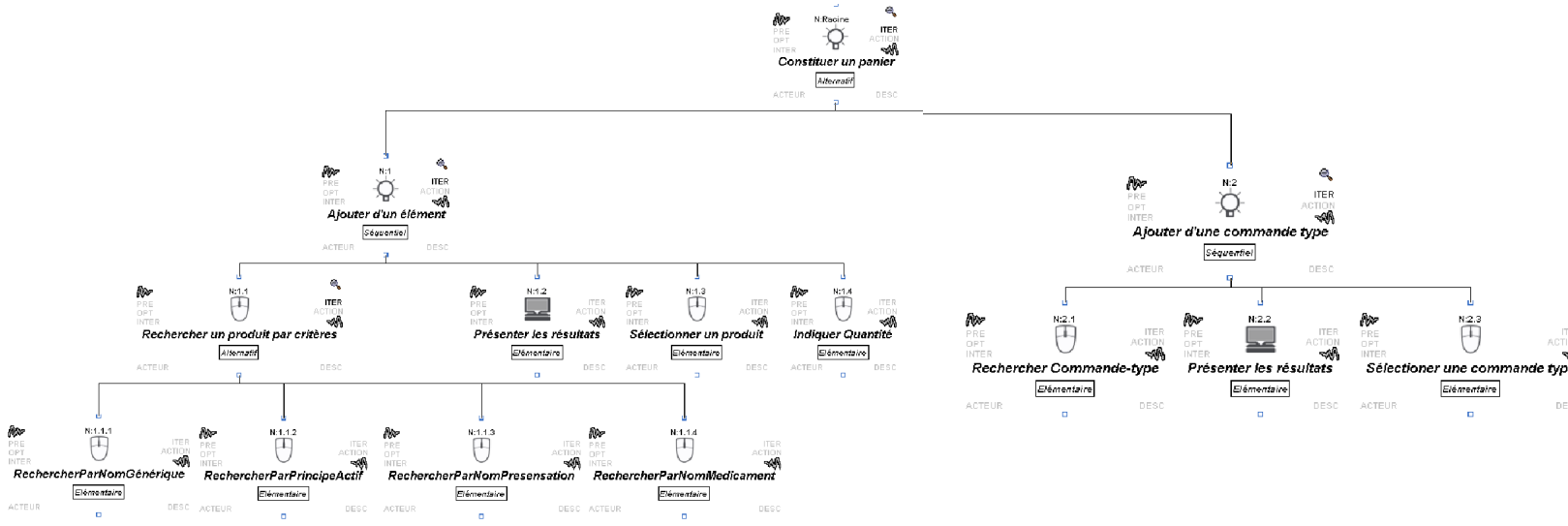
Pour la partie développement, au départ nous nous étions principalement concentré sur le back-end en raison de différentes erreurs et difficultés rencontrées, ce qui a entraîné un délaissement sur front-end et engendrant du retard sur l’implémentation de nos fonctionnalités. Cependant, nous nous sommes rendu compte de la tournure de notre organisation sur les priorités de développement. Nous nous sommes donc mieux réparti les tâches pour implémenter et faire fonctionner les fonctionnalités principales de l’incrément 1 pour proposer une MVP fonctionnelle et utilisable.

Annexes

UML JPA

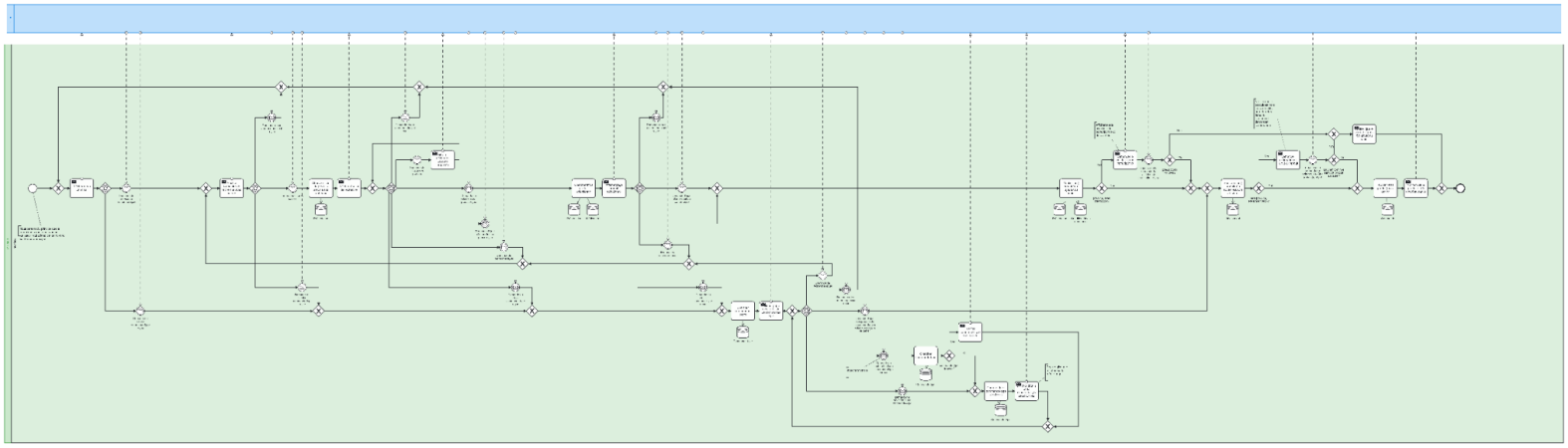


Modèle de tâche



BPMN

Ajouter un élément au panier



Valider le panier