Jérôme Gambiez - Tom Frances - Riad Mahi - Jérémie Gourdon

Livrable GroMed

from Groland

03/02/2023

Sommaire

1. Organisation

Notre groupe de 4 personnes a décidé de se répartir en deux équipes, une pour le front-end et une pour le back-end, en fonction de nos préférences et de nos compétences. Il était évident que ce projet serait fortement axé sur la thématique base de données, et nous avons donc décidé que tous les binômes toucheraient à minima le back-end.

Pour préparer notre projet nous avons choisi de mettre en place un chat discord avec divers éléments de productivité comme un bot github pour avoir une notification sur les commit et pull request.
Pour toute la partie gestion de projet, nous avons mis en place un Notion pour l'historisation des fichiers, le compte-rendu et les rendus intermédiaires. Notre objectif était de cadrer au maximum le développement du projet, notamment en réalisant des pulls requests nécessitant des review de la part d'au moins un autre membre de l'équipe.
Au niveau de l'organisation nous avons choisi de décomposer les projets front / back en plusieurs branches :

- une pour la production, qui contient toujours une version fonctionnelle de l'application et qui est déployée
- une pour le développement (staging) et les tests de préproduction internes
- des branches pour chaque fonctionnalité ou corrections d'anomalies, que nous préfixions en fonction du but de la branche (fix, feat)

2. Notre application GroMed

2.1 La réalisation d'un client web

Pour cette partie client, nous avons choisi d'utiliser le framework Angular avec une librairie de composants Prime NG *(similaire à Angular Material)* qui est open-source. Pour la réalisation de l'application en elle-même, nous nous sommes fiés le plus possible à la maquette réalisée en amont, ce qui nous a permis d'accroître fortement notre productivité en partageant les composants au sein du binôme front. Nous avions alors à ce moment tous une vision précise du but du composant et son implémentation.
L'objectif et le but du client est de pouvoir communiquer avec le back-end. Pour cela nous nous sommes appuyés sur l'api "open api generator" qui permet de générer des services en fonction des endpoints de l'API. Nous avons choisi

d'utiliser cette API pour pouvoir générer rapidement et facilement les services et pour clarifier notre gestion de GET/POST fournie par le back-end.

L'application angular permet d'avoir une découpe des composants optimale pour la maintenance, et pour la gestion d'une page. On a choisi d'avoir un répertoire parent pages qui inclut les différentes pages de notre application et un répertoire composant qui contient par pages / objectifs des composants angular.

2.2 Serveur

Nous avons développé le back-end de notre application en Java avec le framework Spring Boot. Le code a été divisé en 2 modules java :

- restAPI, qui contient toutes les spécifications de notre API REST, à savoir les endpoints, les réponses possibles, et les objets retournés (DTO)
- server, qui contient l'implémentation de notre serveur

Nous avons séparé notre API en 3 couches distinctes, chacune ayant une responsabilité bien définie.
La couche Controller, qui implémente les endpoints de notre API et qui sert d'intermédiaire entre les clients et le serveur. Chaque controller implémente l'interface endpoint définie dans le module restAPI.
La couche Service, appelée par les controllers, contient toute la logique métier du serveur.
La couche Composant, appelée par les services, est chargée de la gestion des données via les repository.
Cette organisation nous permet de mieux séparer notre code et les différentes responsabilités (données, métier, communication), le rendant plus maintenable et plus facilement testable.

Nous avons choisi d'utiliser Swagger UI pour décrire notre API REST, qui apporte de nombreux avantages. Cela nous permet d'avoir une description détaillée de notre API, des différents endpoints existants, des types d'objets retournés ou des paramètres attendus. Swagger UI nous permet également de tester manuellement notre API, ce qui se révèle parfois plus rapide et pratique que de devoir passer par des tests unitaires ou fonctionnels.

Nous avons ajouté des index dans notre base de données sur les champs utilisés par nos critères de recherche (nom de médicament, nom de fabricant...) afin d'accélérer la recherche sur ces valeurs.

2.3 Les fonctionnalités réalisées

- Afficher les présentations d'un médicament avec une pagination (24 par pages)
- Faire une recherche avec la possibilité d'utiliser les filtres suivant :
    - Principes actifs
    - Fournisseur
    - Disponibilité du médicament
    - Médicament générique ou original
- Le tri par prix croissant ou décroissant

Nous donnons la possibilité à l'utilisateur de chercher par la dénomination d'un médicament. Si l'utilisateur souhaite ajouter des filtres supplémentaires, il pourra choisir parmi la liste ci-dessus. Une fois les résultats affichés, il pourra choisir de trier ou non par prix croissant ou décroissant.

- Afficher la page détail d'une présentation

Pour cette page, nous avons fait le choix de d'afficher les informations importantes du médicaments, ainsi que la composition. On laisse évidemment le choix de la quantité à l'utilisateur. Sur cette page on retrouve une checkbox pour accepter les indications de prévention si le médicament en contient. Tant que cette case n'est pas cochée (si elle est présente), l'utilisateur ne peut pas ajouter le produit au panier.

- Le panier

Notre panier se déroule en 3 étapes, la première est un récapitulatif des présentations ajoutées au panier avec les quantités associées.

La deuxième étape est la partie ou l'on notifie l'utilisateur s'il manque des stocks. Si c'est le cas, nous laissons le choix à l'utilisateur d'accepter un délai de livraison de 7 jours ou de supprimer le produit du panier.

La troisième étape, est la validation du panier ou l'on affiche à l'utilisateur que sa commande est validée.

- Le restockage des médicaments plus en stock
- L'authentification
- Afficher la liste des commandes passé
- Les commandes types

Nous pouvons sauvegarder une commande type au moment du passage d'une commande en renseignant un nom de commande. Nous affichons la liste des commandes types, ainsi que l'ajout au panier.

2.4 Les fonctionnalités non réalisées

- Éditer un profil
- L'édition de la facture
- Annuler une commande sous 30 minutes
- Afficher les statistique sur les clients
- Désactiver les références

2.5 Gestion des stocks et accès concurrent

La mise à jour des stocks à la suite d'une commande de médicaments par des utilisateurs se fait de manière concurrente au sein d'une transaction. Nous avons ajouté manuellement une pause de 5 secondes juste avant la mise à jour des stocks afin de mettre en évidence la gestion de l'accès concurrent.

Prenons l'exemple de Paul et Julie, deux utilisateurs de Gromed qui souhaitent commander 100 boîtes du même médicament. Par malchance, ils effectuent leur commande simultanément, et il ne reste que 100 boîtes de ce médicament en stock.

Dans notre processus de validation d'un panier, Paul et Julie n'auront pas d'alertes d'indisponibilité du médicament, car à ce moment les stocks sont suffisants pour leurs commandes respectives, et les articles dans un panier ne sont pas exclusifs. Au moment où ils passent leur commande, la validation de la commande de Julie (par exemple) démarre en premier, et va ainsi verrouiller dans la base de données le tuple de cette présentation. Juste après, la validation de la commande de Paul démarre, mais puisque la présentation est verrouillée en BD, sa validation va se mettre en attente. La mise à jour des stocks liée à la commande de Julie va s'effectuer (passage des stocks à 0) et la transaction va se terminer, ce qui va libérer le verrou. La transaction de Paul va pouvoir reprendre, et mettre à nouveau les stocks à jour (à -100).

Nous avions pour objectif de pouvoir notifier l'utilisateur dans un cas où aucune alerte d'indisponibilité n'était affichée avant la validation de sa commande mais qu'un accès concurrent changeait cela, mais nous avons manqué de temps pour implémenter cette alerte.

3. Les difficultés rencontrées

La recherche filtrée de présentations de médicaments a été une fonctionnalité délicate à implémenter côté serveur.
Cette recherche peut contenir jusqu'à 5 critères de recherche : le nom du médicament, sa disponibilité, si le médicament doit être original, générique, ou les deux, la composition du médicament et les fabricants.
Nous avons vite écarté la possibilité de faire une requête à notre base de données pour chaque combinaison de filtre possible, car le nombre de requêtes nécessaires grandit exponentiellement avec le nombre de filtre. Nous avons donc cherché un moyen de construire une requête générique, adaptée à n'importe quelle combinaison de filtre.
Pour cela, nous avons utilisé l'API Criteria de JPA, qui permet de créer des requêtes en mode objet de manière programmatique, et qui s'adapte à n'importe quelle combinaison de filtre. Cette solution rend le code facilement extensible à l'ajout d'un autre filtre.
Elle présente néanmoins une limite au niveau de la pagination. En effet, cette API ne propose pas de méthodes prenant en compte efficacement la pagination, et il nous a fallu la gérer à la main.
Pour cela, nous avons exécuté 2 requêtes : une première qui récupère les présentations de médicaments correspondants aux critères de recherche et de pagination, et une seconde qui compte le nombre total de présentations correspondant aux critères de recherche uniquement. Avec ces 2 résultats, nous sommes capables de créer correctement la page des présentations.

Un autre point critique du serveur a été un test particulier de controller, celui permettant de récupérer une page de présentation de médicaments. En effet, la Page que nous renvoyons dans ce controller est une interface, et ne dispose donc pas d'implémentation concrète. Ainsi dans notre test de ce controller, lorsque l'on essaie de regarder le contenu de la page renvoyée par l'API, une erreur nous indique que la page n'est pas créée, car elle n'est pas désérialisée correctement.
Afin de corriger ce problème et de pouvoir tester ce controller, nous avons donc implémenté un deserializer json de page.

4. Charte graphique

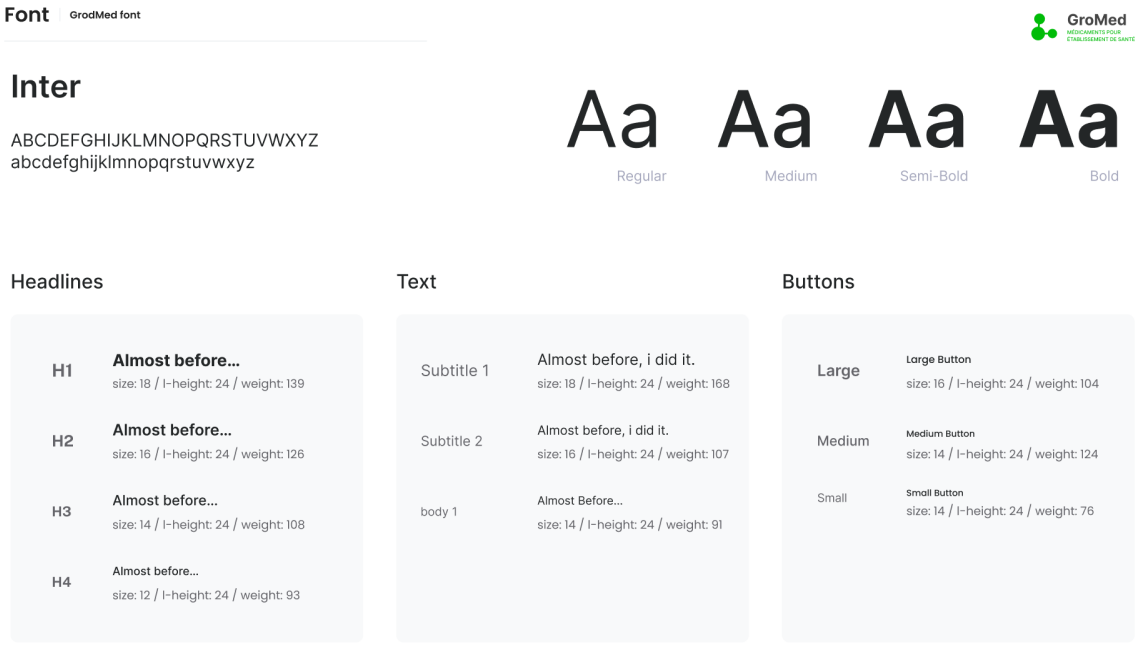
4.1 La marque

GroMed
MÉDICAMENTS POUR
ÉTABLISSEMENT DE SANTÉ

Reprise de l'image de marque fournie en png, pour la transformer en svg.

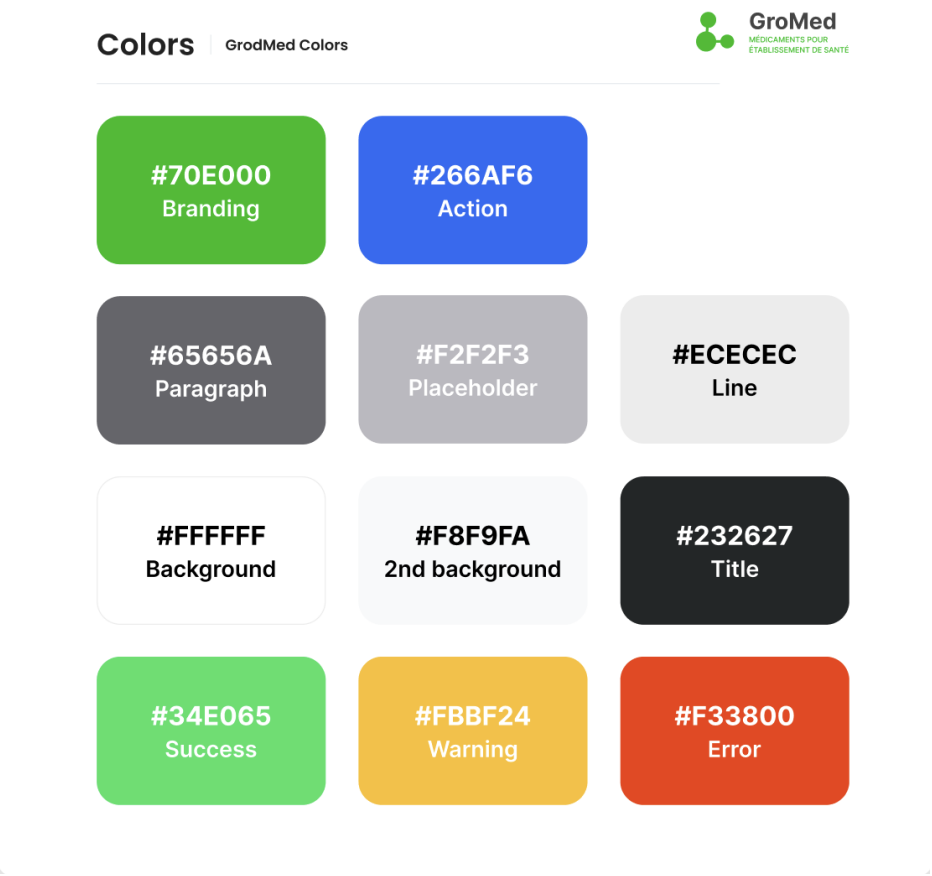
4.2 La typographie

Il y a plusieurs raisons pour lesquelles nous avons choisi la police de caractères Inter. Tout d'abord, Inter est une police open-source, ce qui signifie qu'elle est gratuite à utiliser pour tout le monde, contrairement à Arial qui est une police propriétaire. De plus, Inter a été conçue pour être utilisée sur des écrans numériques, donc elle est optimisée pour une utilisation sur des ordinateurs, des tablettes et des téléphones. Enfin, Inter est un peu plus moderne que Arial, avec des formes de caractères plus épurées et une palette de poids plus étendue qui peut être utilisée pour différents types de mise en page.



4.3 La palette de couleur

Nous avons également retravaillé la palette de couleur pour avoir un meilleur confort visuel, que nous justifions. Dans ce domaine très précis qu'est le médical nous avons choisi de garder des couleurs simples, voire même des couleurs pâles pour les différents tags pour rester dans la thématique sérieuse du sujet.



4.4 La couleur de la marque

Le vert était pour nous une couleur de marque et non une couleur pour nos actions utilisateurs. Nous avons également modifié la teinte du vert pour une couleur qui offre un meilleur contraste avec des fonds d'écran blancs et noirs.

4.5 La couleur d'action

La couleur bleu sur les actions permet de créer une relation de confiance entre l'utilisateur et la plateforme GroMed. En effet, cette couleur n'est pas utilisée n'importe où, nous l'avons utilisé uniquement sur des actions bien précises telles que l'ajout au panier, ou pour notifier à l'utilisateur une information importante comme le nombre d'éléments dans son panier.