Sommaire

I. Introduction

II. Présentation du projet

1. Présentation de l'équipe
2. Présentation du Bateau Thibault
3. Cahier des charges
4. MVP (Minimum Viable Product)

VI. Gestion de projet

1. Méthodologie de projet
2. Diagramme de Gantt

VII. Conception

1. Maquettes
2. Diagramme UML
3. Architecture MVVM

VIII. Réalisation

1. Technologies choisies
2. Outils utilisés
3. Développement Front
4. Développement Back
5. Sécurité

IX. Déploiement

X. Tests

1. Plan de tests
2. Tests unitaires
3. Tests d'intégration
4. Problèmes rencontrés

XI. Conclusion et perspectives d'avenir

Introduction

Nous allons vous présenter dans ce rapport notre projet back-office d'un point de revente de produits livrés par le bateau Thibault. Nous avons déjà travaillé pour cette entreprise sur une application mobile qui leur a beaucoup plu. Il nous ont donc présenté des contacts qui avaient besoin d'un back-office pour gérer les produits. C'étaient des gérants de points relais distribuant les produits du bateau Thibault. Nous avons discuté des besoins et d'un MVP (Minimum viable product). Il nous ont donc préparé un cahier des charges que nous vous présenterons un peu plus tard.
Dans ce rapport nous commencerons par vous présenter notre équipe et notre but. Ensuite nous allons vous parler de l'entreprise du bateau Thibault, le projet back-office avec son cahier des charges et le MVP à présenter. Puis nous allons vous montrer comment nous avons procédé pour la gestion de ce projet en vous montrant le planning, l'organisation. Nous allons aussi vous montrer les différents diagrammes qui ont été conçus avant le début du développement. La réalisation du projet et le déploiement arrivent après. Dans la partie réalisation nous vous présenterons les technologies choisies et les outils utilisés. Nous commencerons ensuite par vous montrer un peu de développement back car nous avons d'abord bien mis en place le serveur et les appels API avant de nous concentrer sur le front qui a pris plus de temps à coder. Nous parlerons aussi de sécurité pour finir cette partie. Nous finirons par montrer le déploiement du projet et comment celui-ci est parvenu aux clients.

Présentation du projet

Présentation de l'équipe

Notre équipe est constituée de 3 développeurs qui se complètent bien sur les rôles de chacun. Ensemble, l'équipe possède une expertise complémentaire pour développer une application web de haute qualité pour les gérants de points relais de distribution des produits du Bateau de Thibault. Ils sont capables de travailler en étroite collaboration pour assurer la qualité du code et respecter les délais impartis.
Yael Donat est un développeur fullstack avec une spécialisation côté front-end. Il possède une solide expérience dans la création d'interfaces utilisateur conviviales et responsives. Il est expert dans l'utilisation de frameworks tels que React et Vue, ainsi que dans l'intégration de designs de haute qualité.

Abderrahim Yourmeche est un développeur spécialisé dans la gestion de projet. Il possède une grande expérience dans la coordination d'équipes de développement, la planification de

projets, la gestion des ressources et la mise en œuvre de méthodologies agiles. Il est également expert dans la communication avec les clients et la résolution de problèmes.

Anojan Yogeswaran, est un développeur fullstack avec une spécialisation côté back-end. Il possède une expérience solide dans la création de systèmes d'API, la gestion de bases de données et l'optimisation des performances des serveurs. Il est expert dans l'utilisation de frameworks tels que Node.js et Django. Il code surtout en Rust.

Présentation du Bateau Thibault

“Le seul intermédiaire entre la mer et vous, c'est nous!”. Voilà la citation du bateau Thibault. Cette entreprise vend des poissons, coquillages et crustacés, en direct de leur bateau et des autres bateaux. Nous avions déjà travaillé sur une application mobile qui servait de marketplace pour qu’ils puissent vendre leurs produits. Il suffisait de remplir le panier en choisissant les produits et d’acheter les produits en choisissant le point relais correspondant. Depuis peu le bateau Thibault fait aussi la livraison à domicile sans frais pour Paris et l’Ile de France. Et avec frais pour le reste de la France. Tous leurs produits sont pêchés par leur soins, donc il est possible que des produits soient pêchés juste après notre commande (ce qui peut la retarder). Ils pêchent en utilisant leurs bateaux compris entre 6m50 et 14 m. Ils pratiquent la pêche artisanale en respectant la mer par leurs techniques de pêche. Les pêches les plus longues peuvent durer 8h/10h. Le bateau Thibault a plusieurs points relais qui vendent tous ces produits aussi et c’est eux nos clients pour ce projet.

Cahier des charges

1. **Vue de gestion de stocks**
   L'application doit permettre aux gérants de points relais de visualiser et de modifier les stocks de produits. Cette vue doit permettre de trier les produits par catégorie, d'afficher le nombre de produits disponibles et vendus, ainsi que de mettre à jour les quantités en stock. Les gérants doivent être en mesure d'ajouter de nouveaux produits et de supprimer des produits existants.

2. **Suivi des ventes**
   L'application doit permettre aux gérants de suivre les ventes de produits et de visualiser les ventes en temps réel. La vue de suivi des ventes doit permettre de trier les produits par catégorie et de visualiser le nombre de produits vendus. Les gérants doivent être en mesure de mettre à jour les quantités vendues pour chaque produit. Il

doit aussi pouvoir mettre à jour les différentes promotions en pourcentage. Certains produits pourront aussi être mis en promotion automatiquement si la quantité disponible est trop élevé

3. **Dashboard**
   L'application doit fournir un dashboard pour les gérants de points relais. Ce dashboard doit inclure les KPIs de base tels que le chiffre d'affaires annuel, trimestriel, mensuel et hebdomadaire. Il doit également afficher la part des produits vendus en promotion par rapport à ceux vendus à prix fort et la marge réalisée par trimestre. Le dashboard doit être facilement accessible et offrir une vue synthétique de la performance du point relais.

4. **Sécurité**
   L'application doit être sécurisée et protéger les données. Les gérants de points relais doivent être en mesure de se connecter à l'application avec un identifiant et un mot de passe. L'application doit également être équipée de fonctionnalités de sécurité avancées pour prévenir les cyberattaques et les tentatives d'accès non autorisées.

5. **Accessibilité**
   L'application doit être accessible à tous, y compris les personnes en situation de handicap. Elle doit être conçue de manière à être facilement utilisable par les personnes ayant des difficultés visuelles ou auditives.

6. **Évolutivité**
   L'application doit être évolutive et capable de s'adapter aux besoins futurs des gérants de points relais. Elle doit être facilement extensible pour permettre l'ajout de nouvelles fonctionnalités ou l'intégration de nouveaux services. L'application doit également être conçue de manière à pouvoir être mise à jour régulièrement pour garantir une performance optimale.

Avec le temps restreint que nous avons pour faire tant de fonctionnalités. Nous avons décidé de proposer un MVP aux clients pour qu'ils puissent commencer à utiliser le produit.

Minimum Viable Product

Le Minimum Viable Product (MVP), ou Produit Minimum Viable en français, est une version simplifiée d'un produit qui contient les fonctionnalités de base nécessaires pour répondre aux besoins des utilisateurs et valider les hypothèses commerciales. L'objectif d'un MVP est de tester le produit sur le marché avec un investissement minimum et d'obtenir rapidement des commentaires des utilisateurs pour ajuster et améliorer le produit.

Voici ce que nous avons présenté à nos clients :

- Permettre aux administrateurs de se connecter à l'application en utilisant le protocole d'authentification JWT.
- Afficher une page d'accueil avec des images qui changent régulièrement.
- Afficher une liste des produits avec une fonction de filtrage par catégorie pour permettre aux gérants de rechercher des produits en fonction de leur catégorie.
- Mettre en place une barre de recherche avec une liste déroulante et une fonction d'autocomplétion pour faciliter la recherche des produits.
- Afficher une page de détail pour chaque produit avec une fonctionnalité d'ajout et de retrait de la quantité de produits.
- Permettre aux administrateurs de modifier le pourcentage de promotion des produits en vente pour améliorer les ventes.
- Mettre en place des fonctionnalités pour ajouter, supprimer et modifier les produits disponibles dans l'application.
- Mettre en place une fonctionnalité de mise en promotion automatique des produits si le stock est trop important.
- Mettre en place un système de gestion d'erreurs pour éviter les erreurs de saisie au gérant.

Ce MVP permettra aux utilisateurs de naviguer facilement à travers les différents produits, d'ajouter des produits dans leur panier et de profiter des promotions. Les administrateurs pourront ajouter, modifier et supprimer des produits facilement, et mettre en place des promotions pour améliorer les ventes. La fonctionnalité d'autocomplétion dans la barre de recherche facilitera la recherche de produits.

Gestion de projet

Avant de commencer un projet il est important de déterminer la méthodologie que nous allons adopter. Cela permet de structurer les tâches que nous devrons réaliser tout au long du projet. En plus de la méthodologie, il y a aussi le diagramme de Gantt qui permet de voir notre projet d’un point de vue chronologique. C’est ce que nous allons voir dans nos deux sous parties.

Méthodologie de projet

Il faut toujours essayer d’adopter la méthodologie qui convient le mieux à notre projet. Les facteurs qui sont à prendre en compte sont la durée, la complexité, les liens avec les clients et

bien d’autres. Les méthodologies agiles sont assez connues et les entreprises essaient d’utiliser cela de plus en plus. Scrum par exemple est une méthode agile qui favorise le retour client. Il y a 3 groupes différents qui s’allient pour arriver au projet final. Nous avons déjà le product owner, le scrum master et l’équipe des développeurs.
Le sprint est la pièce maîtresse de la méthode agile Scrum. D'où l'appellation couramment utilisée de sprint scrum. Entendez par sprints des itérations courtes de développement conçues pour créer un produit ou service de manière incrémentale.
Ces sprints permettent dans un long projet d’avoir constamment des feedbacks sur des fonctionnalités ce qui permet d’avancer dans le développement du produit avec un avis du client assez représentatif. C’est une méthode qui convient bien à des projets longs ou les clients sont assez présents. Sauf qu’il peut arriver très souvent que le client soit présent au début, mais avec le temps il nous accompagne de moins en moins dans la création de ce produit. Dans ce cas là il arrive que les entreprises retournent à des méthodologies plus classiques tel que le cycle en V

Le cycle en V est la méthodologie que nous avons choisi d’utiliser pour ce projet. Et nous allons vous expliquer dans les lignes qui suivent pourquoi nous avons trouvé que cela convient très bien.
Le cycle en V est une méthodologie qui s’inspire grandement de la méthode en cascade parue en 1970. Elle respecte le même principe qui est de permettre de représenter des processus de développement de manière linéaire et en phases successives. Cette méthode existe depuis 1980 mais à partir de 2000 les changements technologiques on fait que les méthodologies agiles sont devenues plus utilisées. Reste que beaucoup d'entreprises passent souvent d’une méthodologie à un cycle en V par manque de présence des clients par exemple. Cette méthodologie est fondamentalement plus simple à respecter que les méthodologies agiles.
La lettre V fait référence à la vision schématique de ce cycle, qui prend la forme d’un V : une phase descendante suivie d’une phase ascendante. Le cycle en V associe à chaque phase de réalisation une phase de validation, comme l’illustre le schéma ci-dessous :

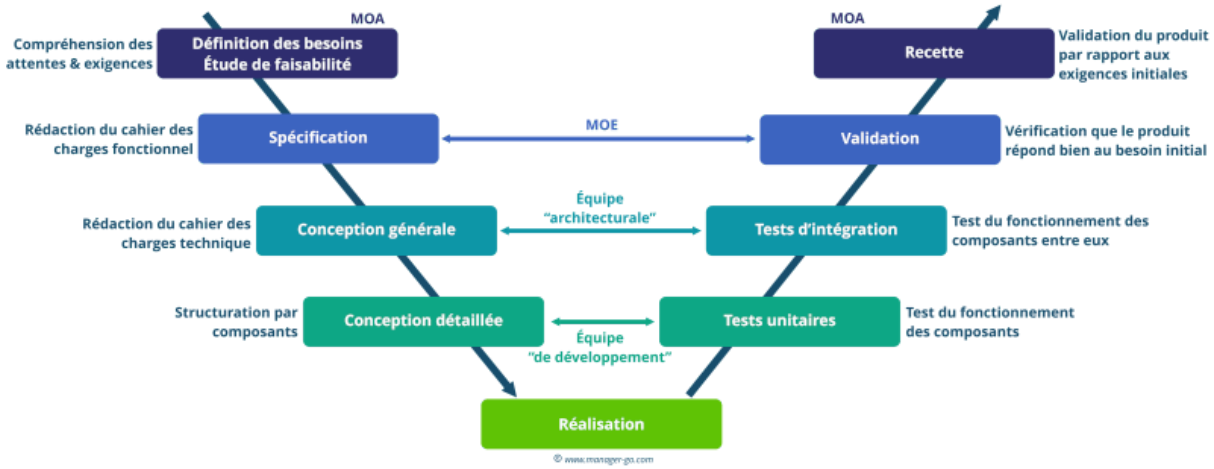


Schéma montrant les différentes étapes du cycle en V

Pour un projet qui se passe dans un délai assez court avec une petite équipe où nous ne voulons pas forcément organiser des réunions à tout va avec les clients. Le cycle en V convient plutôt bien. Nous avons juste à suivre les différentes étapes en planifiant bien le temps que nous prendrons pour chacune d'elles. C'est pour cela que nous avons décidé de faire un diagramme de Gantt prévisionnel pour essayer de rester dans les temps.

Diagramme de Gantt

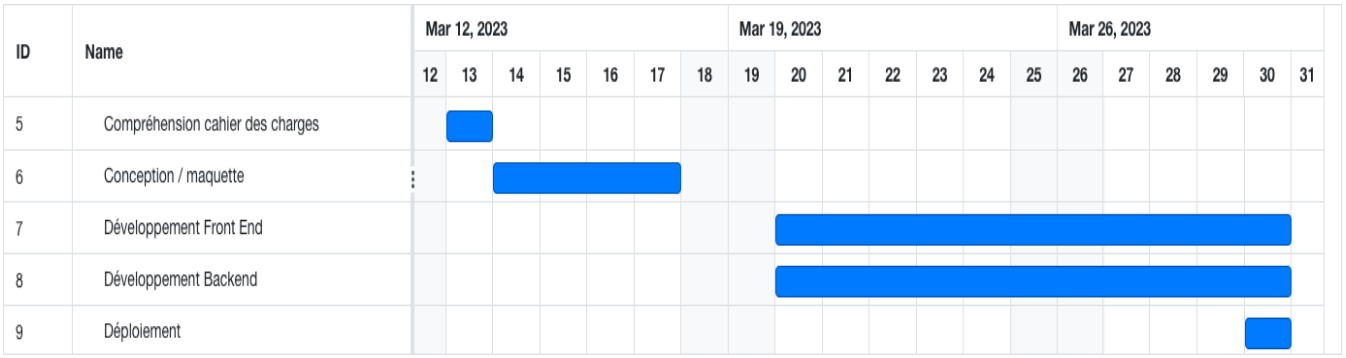


Diagramme de Gantt prévisionnel

Ce diagramme nous permet de planifier le travail à faire et les délais à respecter. On voit qu'on s'est bien concentré sur la compréhension du besoin et la conception du projet avant d'attaquer tranquillement la partie développement.
C'est important pour pouvoir bien avoir les maquettes et les idées de ce que nous allons faire en front. Et l'étude du cahier des charges permet de savoir les fonctionnalités dont nous aurons besoin de développer en back. Elle permet aussi de faire des maquettes qui conviennent aux besoins de l'administrateur. Arrive après le développement front et back qui se feront en parallèle. On a décidé celà car nous avons préféré laisser les développeurs dans leur spécialité. Pour qu'ils puissent assez vite connecter le back et le front pour tester les fonctionnalités. Cela ressemble un peu à de l'extrem programming avec du pair programming et des itérations très rapide.

Conception

Les maquettes

Dans cette partie nous allons voir les différents éléments de conception dont nous avons eu besoin. Il y a avant tout les maquettes qui nous ont permis de voir visuellement à quoi ressemblerait l'appli. Ça aide beaucoup plus à avoir une idée de ce que nous allons développer en front. Nous avons des wireframes pour la page de connexion, pour la page home et aussi des produits.

Les diagrammes UML

Les diagrammes UML, ou Unified Modeling Language, sont des outils de modélisation largement utilisés dans le développement logiciel pour représenter les différentes parties d'un système et les relations entre elles. Il existe plusieurs types de diagrammes UML, chacun ayant sa propre fonction et représentation graphique.

Les diagrammes de cas d'utilisation sont utilisés pour décrire les différentes actions qu'un utilisateur peut effectuer dans un système, tandis que les diagrammes de classes représentent les différentes classes et leurs attributs dans un système. Les diagrammes de séquence montrent l'interaction entre les différents objets dans un système, tandis que les diagrammes d'activité décrivent les processus et les flux de travail.

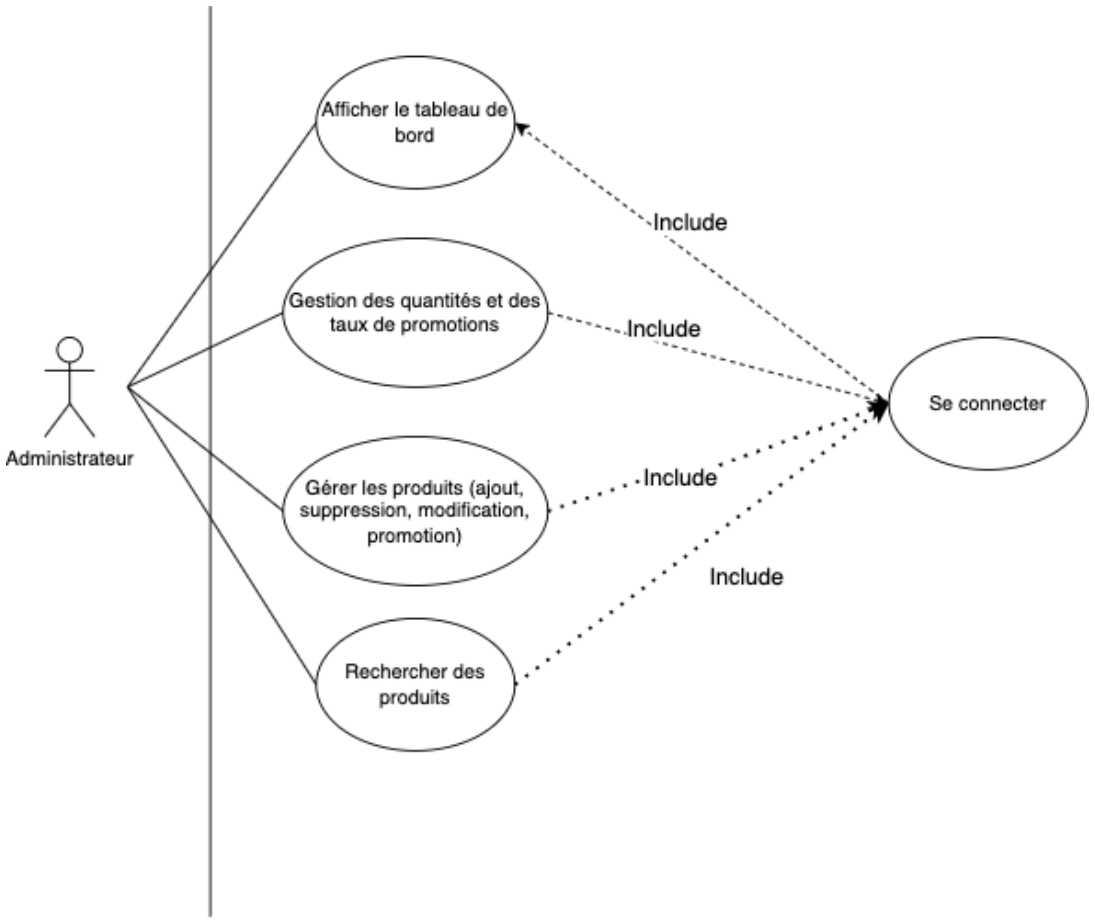


Diagramme de cas d'utilisation

Nous voyons dans la page précédente un diagramme de cas d'utilisation qui présente ce que peut faire notre seul acteur (l'administrateur). Il doit d'abord se connecter à travers une page de login avec ses identifiants. Une fois connecté, il pourra afficher le tableau de bord. Il pourra aussi rechercher des produits à travers une barre de recherche à autocomplétion. Il pourra gérer les différents produits en faisant plusieurs actions dessus. Comme par exemple, modifier des produits, ajouter un nouveau produit ou même supprimer un produit. Dans la modification on pourra modifier plusieurs champs du produit. L'administrateur pourra aussi choisir quel produit il veut mettre en promotion, ainsi que le taux de promotion. Il pourra aussi mettre à jour les quantités de chaque produit.

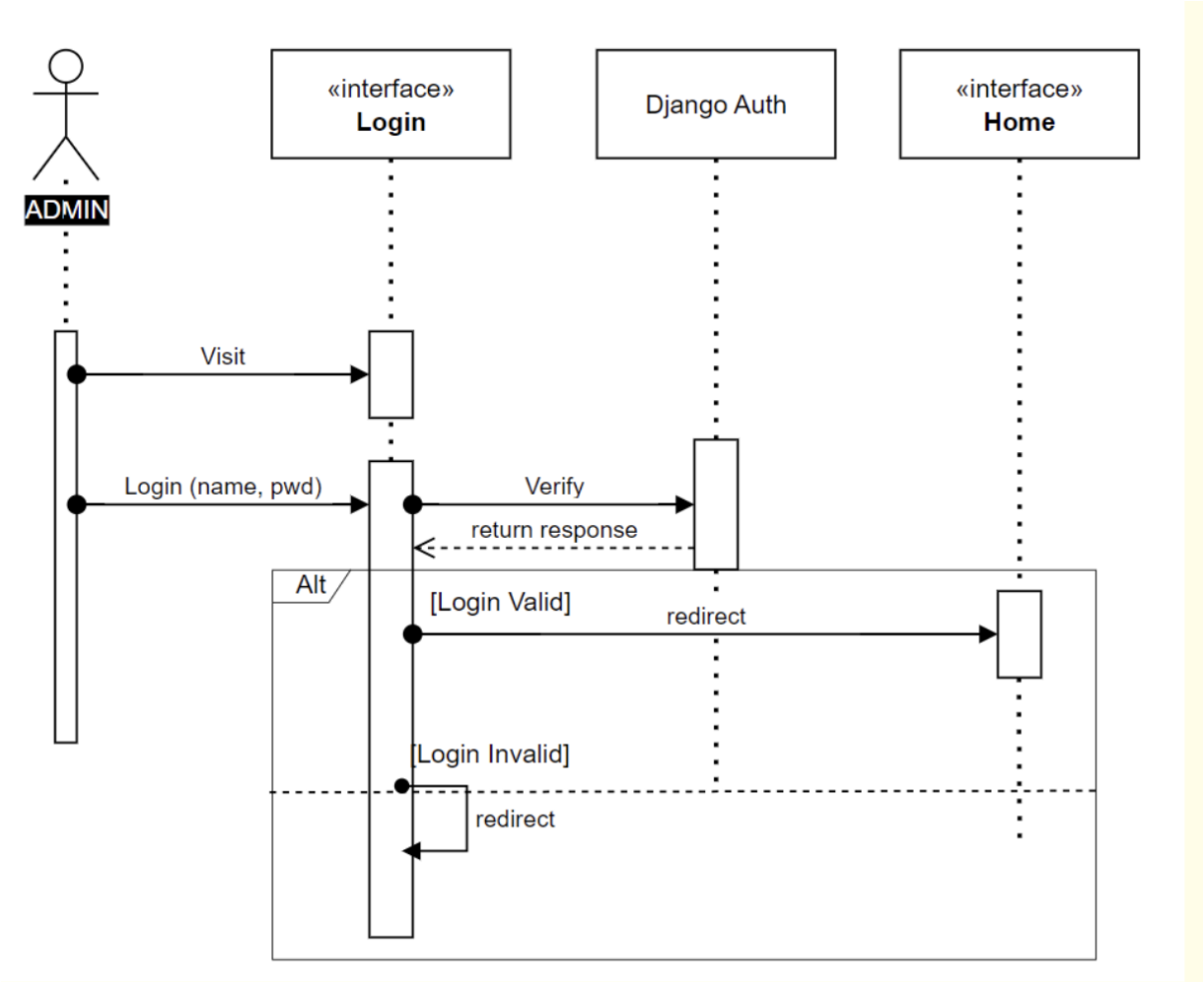


Diagramme de séquence

Ce diagramme de séquence nous montre comment se passe l'authentification. Notre admin arrive sur la page de login il rentre ses identifiants. L'authentification côté Django vérifie si les identifiants sont bons et return une réponse. Si le login est valide ca nous redirige vers le home sinon ca nous redirect vers le login.

Architecture MVVM

L'architecture MVVM, ou Model-View-ViewModel, est un modèle de conception d'interface utilisateur qui permet de séparer clairement les responsabilités du code et d'optimiser la maintenance et la réutilisation du code. Cette architecture est souvent utilisée avec des frameworks tels que Vue et Django pour le développement de projets web.

Le modèle MVVM se compose de trois parties principales : le modèle (Model), la vue (View) et le ViewModel. Le modèle représente les données de l'application, la vue est la partie

visible de l'interface utilisateur et le ViewModel agit comme un intermédiaire entre les deux en gérant les données et la logique métier.

L'un des avantages de l'architecture MVVM est qu'elle permet une meilleure gestion de la complexité du code. En séparant les responsabilités, il devient plus facile de maintenir et de modifier le code sans affecter les autres parties de l'application. De plus, cela permet de faciliter la réutilisation du code et de rendre l'application plus évolutive. Ce qui était demandé dans le cahier des charges

En choisissant d'utiliser Vue et Django (on en parlera un peu plus tard dans la partie réalisation), l'architecture MVVM est un choix judicieux car ces frameworks offrent une prise en charge native de l'architecture MVVM. Vue fournit un système de liaison de données bidirectionnel qui permet de relier facilement les vues et les modèles, tandis que Django fournit une structure de modèle MVC (Model-View-Controller) qui est facilement adaptable pour l'architecture MVVM. En combinant ces deux outils, il est possible de créer des applications web efficaces et évolutives en utilisant l'architecture MVVM.

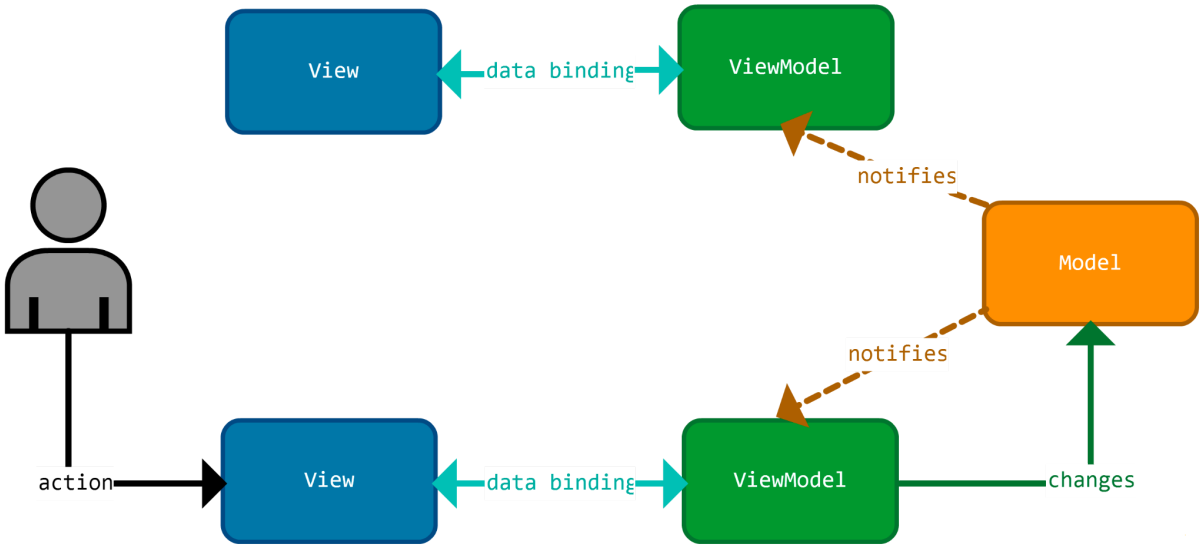


Schéma de l’architecture MVVM

Réalisation

Technologies choisis

Django et Vue sont deux frameworks largement utilisés pour le développement web. Ils sont tous deux open-source, ce qui signifie que leur code source est disponible publiquement et peut être modifié et amélioré par la communauté de développeurs.

Django est un framework web en Python qui offre une structure de modèle MVVM pour la création d'applications web. Il facilite la création de sites web complexes en proposant des fonctionnalités telles que la gestion des bases de données, l'authentification utilisateur, la gestion des formulaires, la sécurité, etc. Il est également hautement évolutif et personnalisable, ce qui permet aux développeurs de créer des applications web sur mesure pour les besoins de leurs clients.

Vue est un framework JavaScript (ici on se force à utiliser TypeScript) pour la création d'interfaces utilisateur réactives. Il permet de créer des interfaces utilisateur interactives et dynamiques en utilisant une syntaxe simple et intuitive. Vue est également hautement personnalisable et offre une grande flexibilité pour la création d'applications web.

En choisissant d'utiliser Django et Vue, on bénéficie de plusieurs avantages. Tout d'abord, la combinaison de ces deux frameworks permet de créer des applications web hautement évolutives et personnalisables. Django offre une structure solide pour la gestion de la base de données, de la sécurité et de l'authentification, tandis que Vue permet de créer des interfaces utilisateur interactives et dynamiques.

En outre, Django et Vue sont tous deux largement utilisés et bénéficient donc d'une grande communauté de développeurs. Cela signifie qu'il est facile de trouver des tutoriels, de la documentation et de l'aide en cas de besoin. De plus, les deux frameworks sont régulièrement mis à jour, ce qui garantit la pérennité des applications web créées avec eux.

Outils utilisés

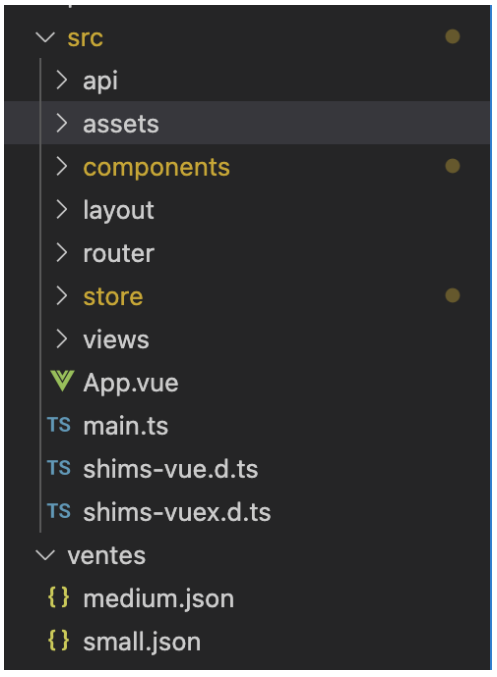
Beaucoup d'outils nous ont aidé pour ce projet. Certains pour le développement, d'autres pour la gestion de projet ou encore les maquettes et le rapport.

- VS Code : Visual Studio Code est l'IDE que nous avons tous les 3 utilisé pour ce projet. Les différentes extensions aident beaucoup pour le confort du codeur. Et nous avons tous l'habitude d'utiliser cette IDE depuis longtemps

- Postman ou HTTPIE : Postman et HTTPIE nous ont servi à tester les requêtes HTTP. C'est un outil qui aide beaucoup à voir les failles qu'on a dans notre code back et à le perfectionner. Anojan avait un mac donc il a utilisé HTTPIE par préférence et les autres ont préféré utiliser Postman avec lequel ils étaient habitués à tester des appels
- Figma : Pour faire les maquettes, le zoning et les wireframes.
- Canva : Pour faire le powerpoint et les logos
- Diagrams.net: Pour faire les différents diagrammes, ça nous a plutôt bien aidé dans la phase de conception.
- Git: Versioning, mettre en commun notre travail.
- Discord : L'endroit où l'on se retrouvait avec les collègues pour travailler et résoudre nos problèmes ensemble. On codait et quand quelqu'un avait un problème on essayait de se mobiliser pour aider. C'est une ressource qui a été importante pour ce projet. Aussi pour se partager des documents
- Free Online Gantt : Nous a servi à faire le diagramme de Gantt prévisionnel.

Développement Front

Architecture du projet



On voit ci dessus l'architecture du projet côté front. Le framework choisi est Vue. Il y a plusieurs dossiers dans le dossier src. Le dossier api contient des fonctions pour faire des fetch vers le back. Les assets contiennent les images et ressources que nous utilisons. Components contient les différents composants que nous utilisons pour les views, tels que les cards, les dialogbox, la navbar, l'autocomplete. Le router contient les différentes

routes pour accéder à chaque page, et aux détails des différents produits. Le dossier store contient des fonctions de Vuex dont on développera un peu plus tard l'utilité. Mais en bref ce sont des fonctions qui permettent de stocker certains éléments comme un tableau de produit ou les tokens dans le navigateur dans un store. Pour finir nous avons les views qui sont ce qui affichent les pages, leurs contenues.

Vuex : Bibliothèque de gestion de store

Vuex est une bibliothèque d'état pour les applications Vue.js. Elle fournit un moyen centralisé de gérer l'état de l'application et de faire communiquer les composants entre eux de manière prévisible et maintenable.

En utilisant Vuex, on peut stocker les données de l'application dans un magasin centralisé (store) qui peut être accédé et modifier de manière cohérente par tous les composants de l'application. Cela permet de simplifier la gestion de l'état, de réduire la complexité et d'améliorer la prévisibilité.

Le store de Vuex est composé de quatre éléments principaux :

State : représente l'état de l'application, qui peut être lu depuis n'importe quel composant.

Mutations : permettent de modifier l'état de manière synchrone et sont généralement invoquées depuis les composants.

Actions : permettent d'exécuter des opérations asynchrones et peuvent appeler des mutations pour modifier l'état.

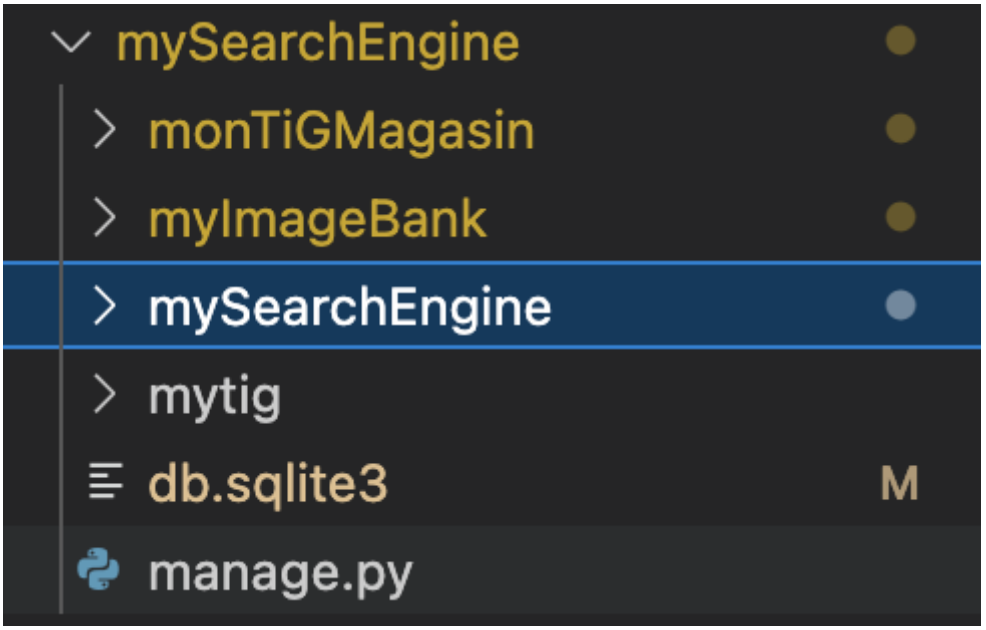
Getters : permettent de récupérer des données du state de manière calculée, c'est-à-dire que les valeurs renvoyées par les getters peuvent être dérivées du state.

```
export default createStore({
  state: {
    user: {
      data: {
        username: null,
        password: null,
      },
      token: {
        accessToken: localStorage.getItem('accessToken'),
        refreshToken: localStorage.getItem('refreshToken'),
      },
    },
    products: {
      data: [],
      loading: false,
    },
    currentProduct: {
      data: {},
      loading: false,
    },
    errorMessage: null,
  },
  getters: {
    products: state => {
      return state.products
    },
    product: state => {
      return state.currentProduct
    },
  },
  mutations: {
    saveProducts: (state, product) => {
      state.currentProduct = product
    },
    logout: state => {
      state.user.token.accessToken = null
      state.user.token.refreshToken = null
      localStorage.removeItem('accessToken')
      localStorage.removeItem('refreshToken')
    },
  },
  actions: {
    getProducts({ commit }, { url = null } = {}) {
      commit('setProductsLoading', true)
      url = url || 'http://127.0.0.1:8000/infoproducts/'
      const { data, error } = useFetch(url)
      commit('setProducts', data)
      //commit('setProductsLoading', false)
      return { data, error }
    },

    getProduct({ commit }, id) {
      commit('setCurrentProductLoading', true)
      const { data, error } = useFetch(
        `http://127.0.0.1:8000/infoproduct/${id}/`
      )

      commit('setCurrentProduct', data)
      //commit('setCurrentProductLoading', false)
      return { data, error }
    },

    saveProduct({ commit }, { id, product }) {
```

Au dessus il y a des screenshots du code de notre store. On voit bien les 4 éléments et une partie de leur contenu (Voir le code complet pour plus de détails)

Développement Back

Architecture du projet



Ci-dessus nous avons l'architecture du back end. On voit notre base de données db sqlite auquel on peut accéder avec la commande sqlite3. Dans mySearchEngine nous avons le fichier settings.py où on config plusieurs éléments comme les tokens et leurs durée de vie. Le dossier monTigMagasin contient le views.py, l'urls.py qui permettent de faire les appels API. Il y a aussi le model.py et le serializer.py pour structurer l'élément dans la base de données. Dans le views.py nous avons plusieurs méthodes get, put, post ou delete qui ont besoin qu'on soit authentifié. Pour l'authentification nous avons utilisé le JWT.

Le JWT, ou JSON Web Token, est un standard ouvert pour la sécurisation des échanges d'informations sur le web. Nous avons choisi d'utiliser cette technologie pour sécuriser les échanges entre notre produit back office et son administrateur.

Grâce à l'utilisation de JWT, nous sommes en mesure de fournir une authentification solide et une protection des données pour nos administrateurs. Les tokens JWT sont générés après l'authentification de l'utilisateur et contiennent des informations spécifiques, telles que l'identité de l'utilisateur et les autorisations d'accès, qui sont vérifiées à chaque requête sur les produits par exemple.

Comment ça marche ? En se connectant, l'utilisateur génère un access token et un refresh token. Les deux ont des durées de vie distinctes. Le refresh token vit

plus longtemps que l’access token. Donc chaque requête utilise cet access token en header et si l’access token est expiré alors on fait une requête en utilisant le refresh token pour reset l’access token et on relance la requête qui avait échoué.

Conclusion

Pour conclure, ce rapport a été très enrichissant pour les 3 personnes de notre équipe. Certains avaient déjà fait du Vue mais personne n’avait touché à Django encore donc ça nous a permis d’apprendre de nouveaux framework en codant un projet qui est plutôt abouti. Nous avons encore certaines améliorations à faire côté front. Mais pour ce qui est du MVP, on trouve que tout est assez bien respecté et est livrable. Nous nous sommes penché sur les fonctionnalités car c’est plus important que l’aspect graphique pour un back-office mais nous allons bien sûr améliorer le style du produit avant la livraison finale. Il nous reste encore certaines fonctionnalités à ajouter tel que le tableau de bord mais nous sommes dans les temps de ce qui est demandé en MVP voir en avance. Le jonglage entre le back et le front était aussi assez intéressant. Nous espérons que les gérants aimeront le back-office. Nous vous remercions d’avoir pris le temps de lire ce rapport.