Rapport

Sujet : Back-Office d'un point de revente des produits du Bateau de Thibault

Soutenu par: Leandre AKAKPO, Anil HADJELI, Wassim HOUIAT, Saif Merouar

ENTREPRISE: Bateau de Thibault

Product Owner: Bui-Xuan BINH-MINH et Arthur ESCRIOU

Période du 15 mars au 29 mars 2023

Sommaire

Introduction

Nous sommes en 4e année de formation pour obtenir le titre Architecte Technique en Informatique et Réseaux parcours Logiciel au CFA INSTA à Paris 2.

Cette année, dans le cadre du module du nom de Développement d'un back office, on a dû réaliser un projet consistant à développer une application web

Le Bateau de Thibault est une société SAS de vente de pêche artisanale de la flotte des bateaux des ports de la Manche. La société a été fondée en 2016 et son actuel président est Thibault-Imanol Gobin.

La société vend des produits de plusieurs catégories :

- Poissons
- Crustacés
- Fruits de mer

Ce rapport sera divisé en plusieurs parties :

1ère partie : Contexte où on décrira le contexte du projet, on présentera l'équipe en charge de ce projet et les outils utilisés pour conceptualiser l'application, pour communiquer et pour la réalisation du projet.

2e partie : Conception, ici on présentera la conception de ce projet et comment on a organisé la réalisation de ce projet.

3e partie : Réalisation, dans cette partie, on va montrer les différentes pages et fonctionnalités qu'on a pu développer, avec quel technologie, le processus de recherche de solution pour un problème qu'on a rencontré lors de la réalisation et comment on cherche à être à jour sur la méthode de développement.

4e partie : Bilan, enfin, on va conclure en présentant quelques exemples de fonctionnalités qu'on pourrait mettre en place dans cette application à l'avenir et en résumant brièvement le projet.

PARTIE 1 : Contexte

Contexte

Les gérants d'un point de revente de produits livrés par Le Bateaux de Thibault ont besoin, pour gérer leur e-commerce, d'une application web qui permettra de back-office pour son site internet https://www.lebateaudethibault.fr/.

Pour cela on doit rédiger avec le client, un cahier des charges, voici un résumé du cahier des charges :

- Authentification : Les gérants du point de revente pourront se connecter au back-office en renseignant leur identifiant et leur mot de passe.
- Tableau de bord : Dès lors qu'on se connecte au back-office, on arrive directement au tableau de bord où on peut visualiser, en temps réel, les statistiques que fait le point de revente.
- Page liste de produits : Sur cette page qui sera dans le menu/la navigation, la liste des produits qui sont actuellement en vente qui seront divisé en trois tableaux :
    - Poissons
    - Crustacés
    - Fruits de mer

  Tous ces tableaux auront les colonnes suivantes :

    - Nom
    - Prix
    - Prix en promotion (uniquement si le produit est en promotion).
    - Pourcentage de promotion.
    - Quantité en stock.
    - Nombre d'articles vendus.
    - Commentaires.
    - Champ pour la gestion d'articles.
    - Deux boutons "Ajouter" et "Enlever" lié au champ pour la gestion d'articles.
    - Champ pour la gestion du pourcentage de promotion.
    - Un bouton "envoyer données" pour la gestion du pourcentage de promotion

  Il y aura une fonctionnalité permettant la recherche en auto-complétion.

- Page produit en détails : Sur cette page on verra les informations suivantes pour un produit :
    - Nom
    - Prix
    - Prix en promotion (uniquement si le produit est en promotion).
    - Pourcentage de promotion.
    - Quantité en stock.
    - Nombre d'articles vendus.
    - Commentaires.
    - Champ pour modifier le stock du produit
    - Champ pour modifier le pourcentage de promotion
    - 1 bouton pour ajouter le stock saisie dans le champ
    - 1 bouton pour enlever le stock saisie dans le champ
    - 1 bouton pour enregistrer la donnée de pourcentage de promotion.

On pourra accéder à l'application web avec un ordinateur, une tablette ou un smartphone via internet sur les navigateurs internet comme Chrome, Firefox, Opera ou encore Safari.

Equipe

Dans ce projet, l'équipe est composé de 4 personnes :

- Wassim Houiat qui est le SCRUM Master et développeur full stack dans ce projet.
- Leandre Akakpo qui a eu le rôle de développeur full stack dans ce projet.
- Saif-Eddine Merouar qui a eu le rôle de développeur full stack dans ce projet.
- Anil Hadjeli qui a eu le rôle de développeur full stack dans ce projet.

Outils

On a pu s'aider de plusieurs technologies pour effectuer la conception ainsi que le développement de cette application. Voici une liste des technos ainsi que leur logo :

Technologies utilisé pour la conception des diagrammes et des maquettes :


On a utilisé l'environnement de développement Visual Studio Code pour développer l'application, Postman pour tester l'API côté Back End :

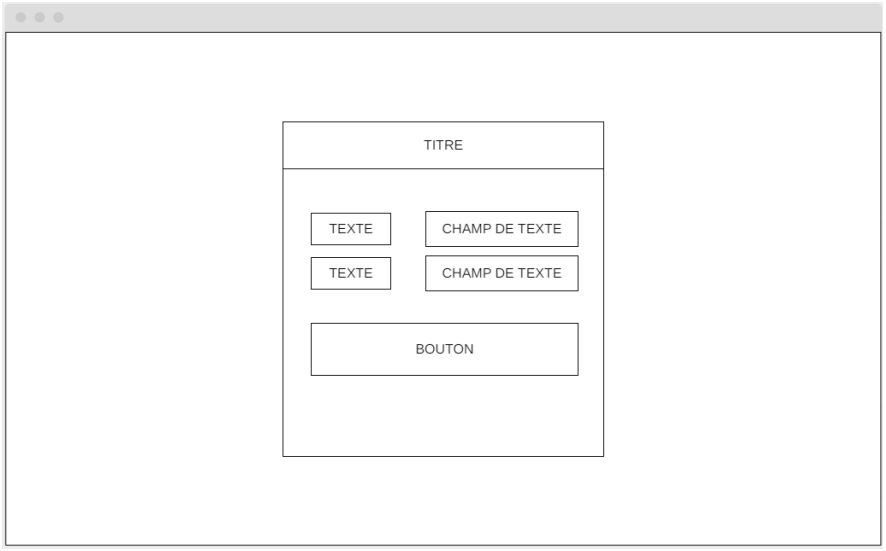
Pour versionner le code, on a utilisé la technologie GitHub :

Pour la gestion du projet, on a pu utiliser le logiciel GanttProject :

Maquettes

Authentification

Zoning



Wireframe

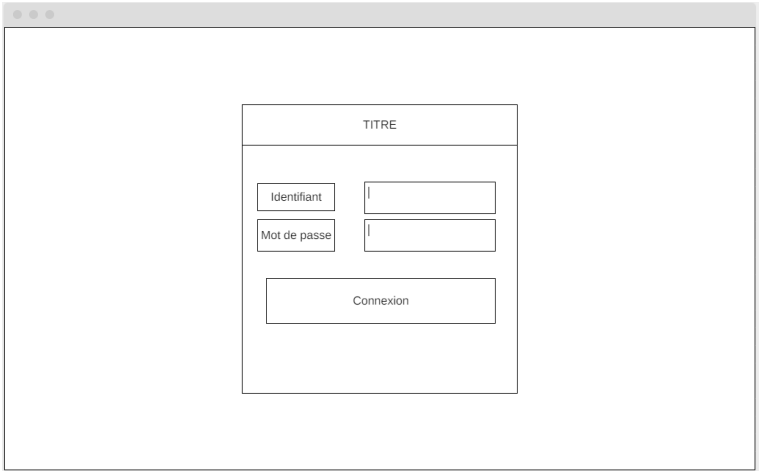
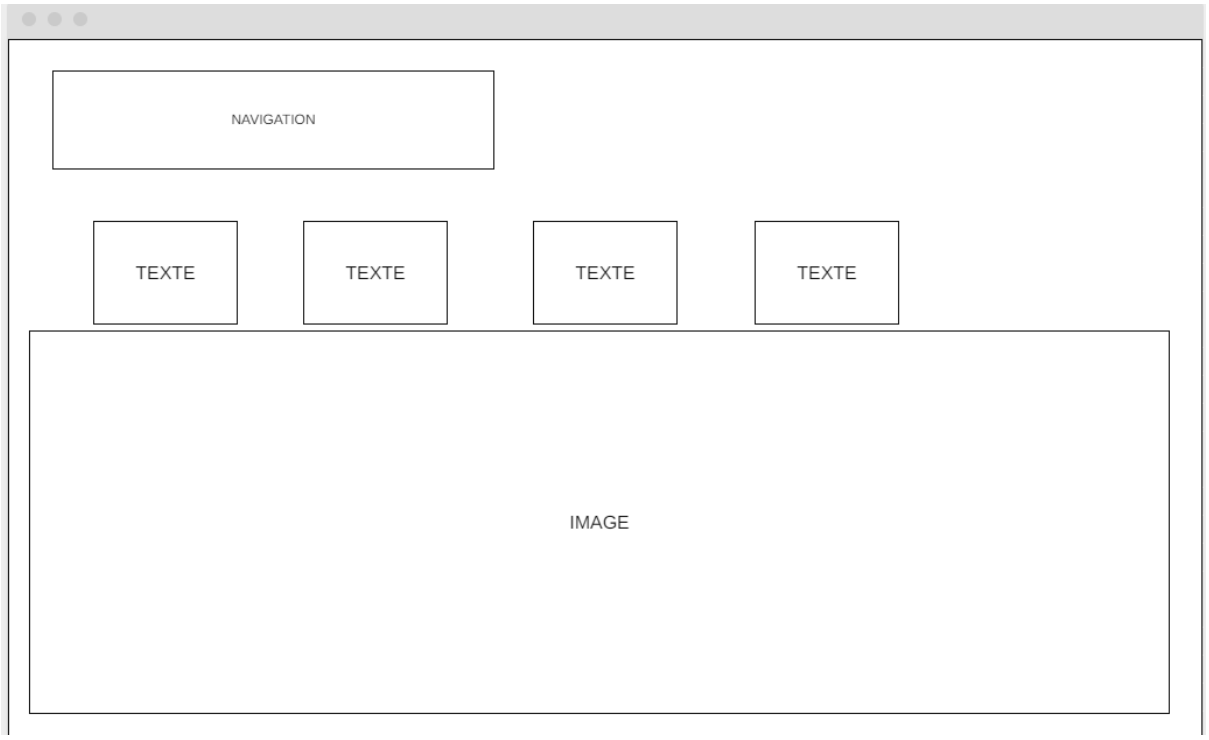
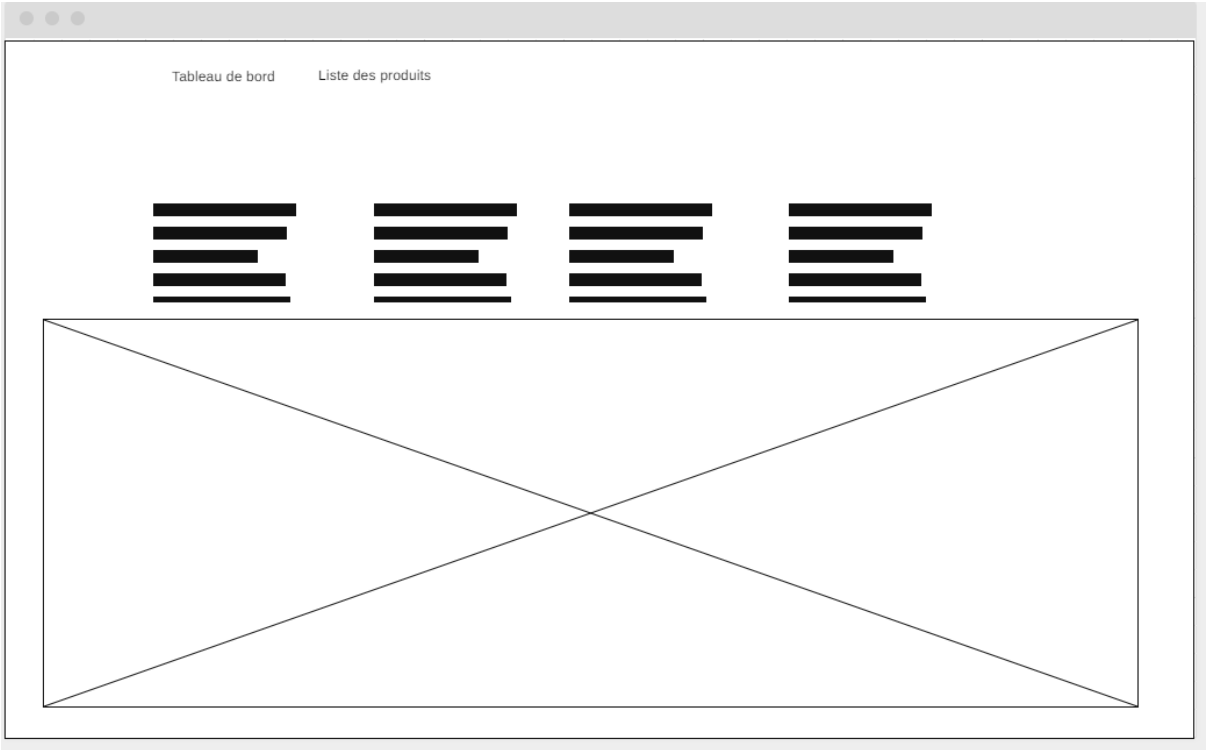


Tableau de bord

Zoning

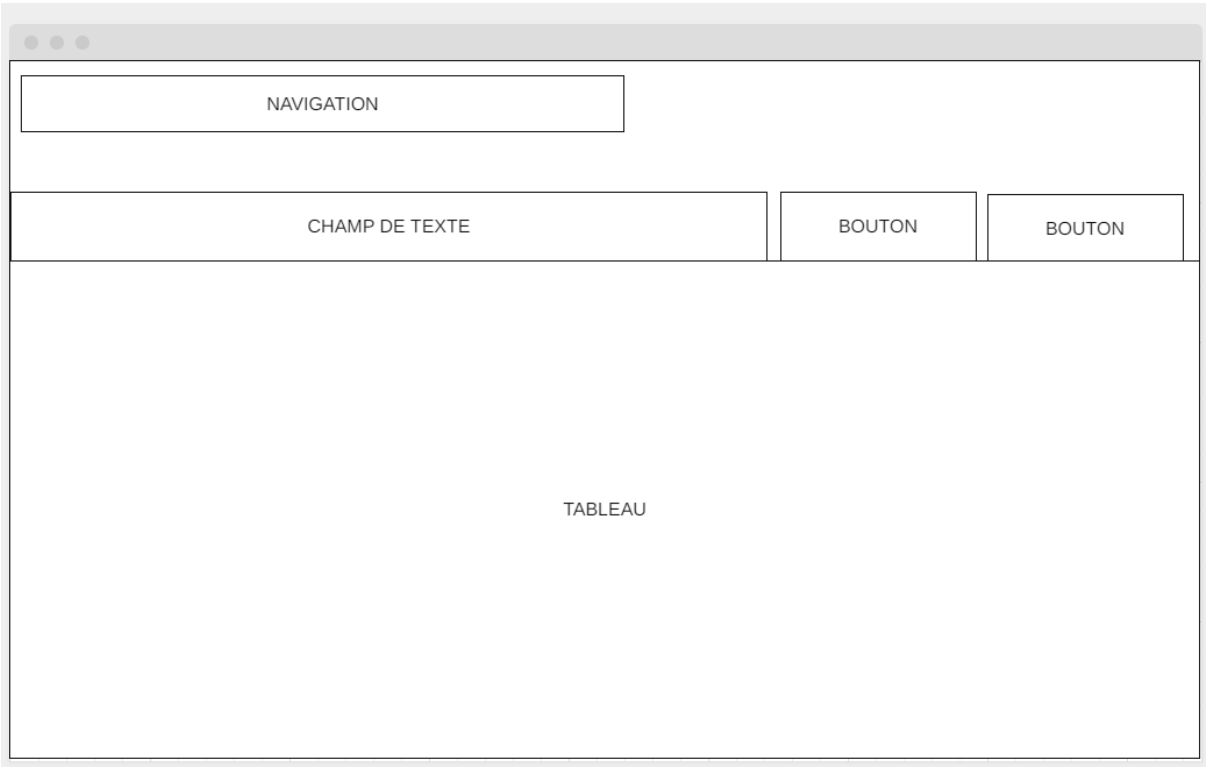


Wireframe

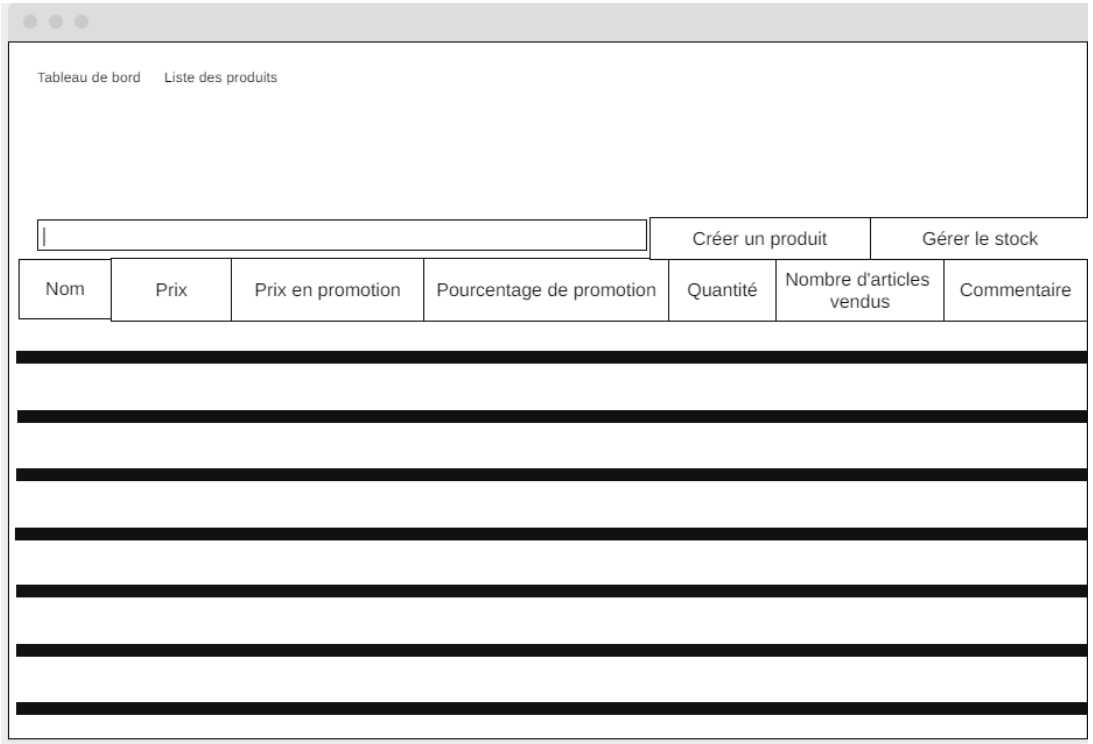


Liste des produits

Zoning

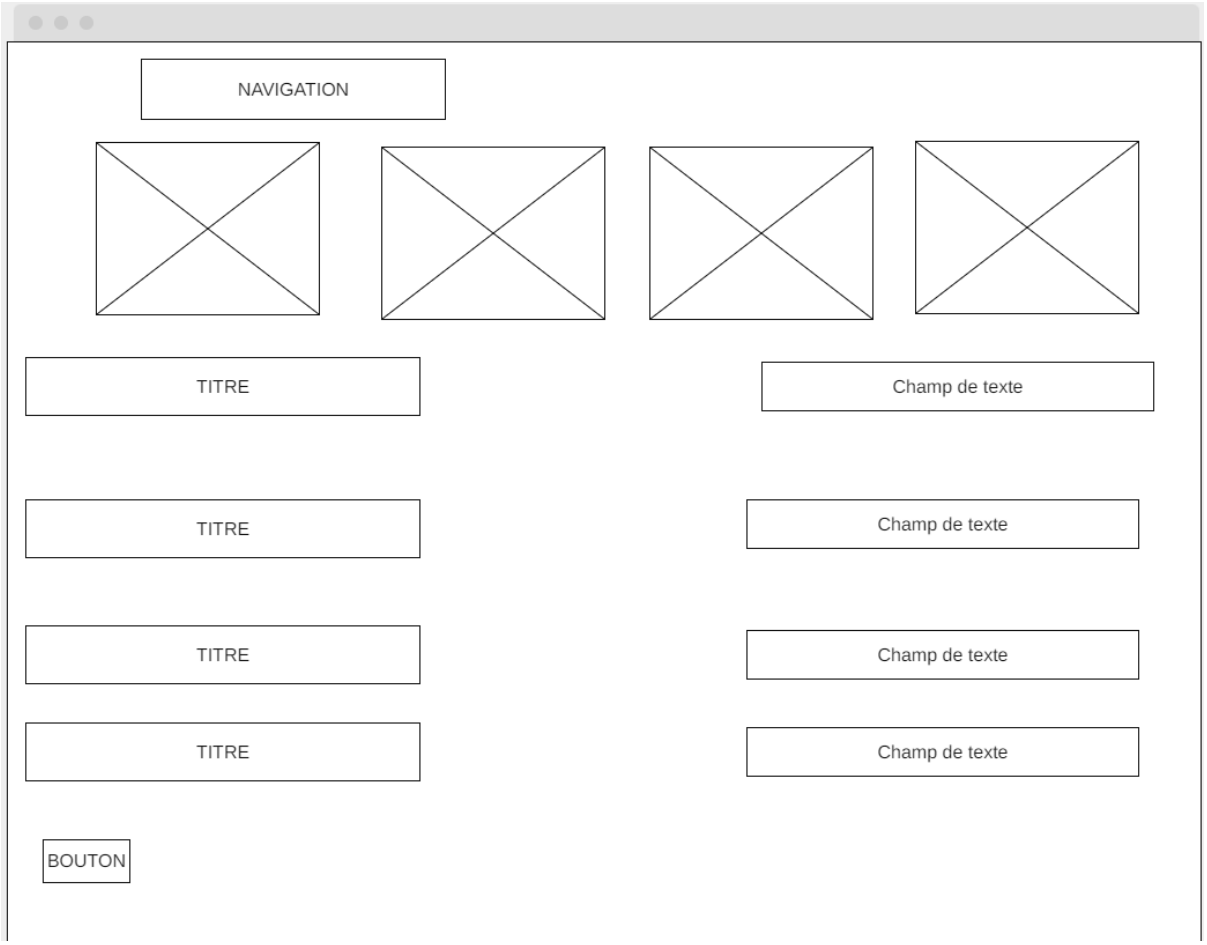


Wireframe

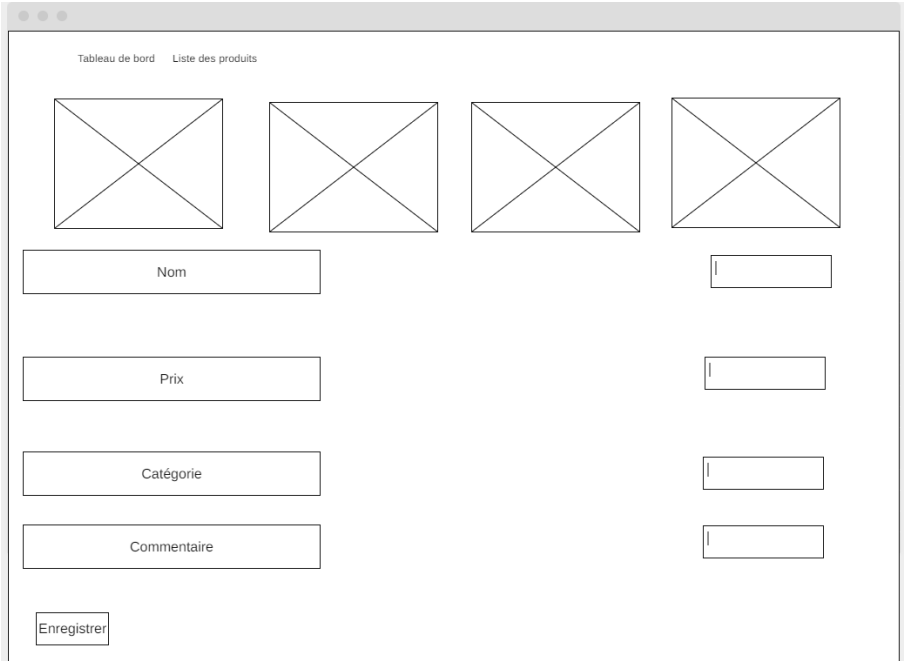


Produit en détails

Zoning

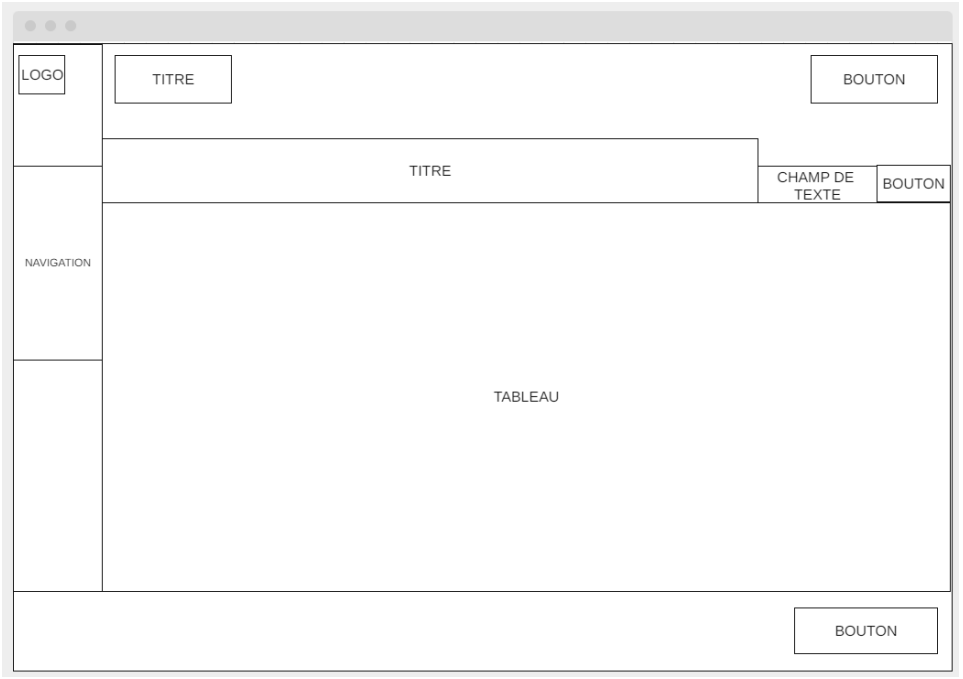


Wireframe

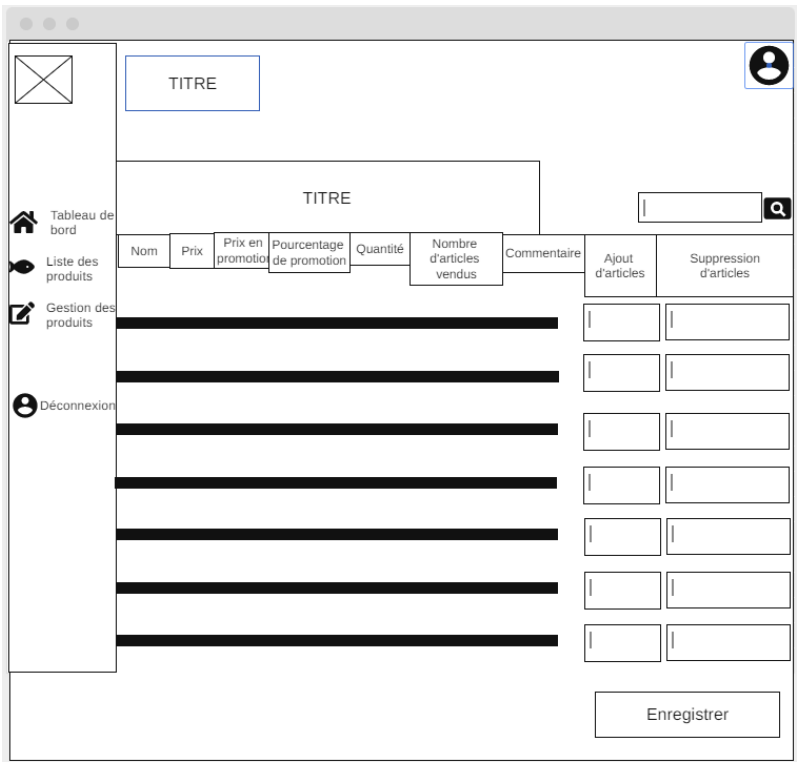


Gestion de stock et promotion

Zoning



Wireframe



Conception de l’application

Afin de conceptualiser l’application web, la première chose à faire c’est de faire 2 diagrammes UML de chaque groupe :

Diagramme de comportement :

- Un diagramme de cas d’utilisations pour représenter les comportements fonctionnels de l’application.
- Des diagrammes d’activités pour illustrer le flux d'événements décrit dans le diagramme de cas d’utilisations.

Diagramme de structure :

- Un diagramme de classe pour présenter les classes et les interfaces du système ainsi que leurs relations.
- Un diagramme de déploiement pour modéliser l’architecture physique du système.

Et enfin, conceptualiser l’architecture de l’application :

- Le Design Pattern (ou Patron de conception) pour organiser le code source.
- L’architecture N-tiers pour répartir les objets sur les différents serveurs d’applications.
- L’architecture Multi-couche pour séparer les différents niveaux de l’application.

Diagrammes de comportement

Diagramme de cas d’utilisations

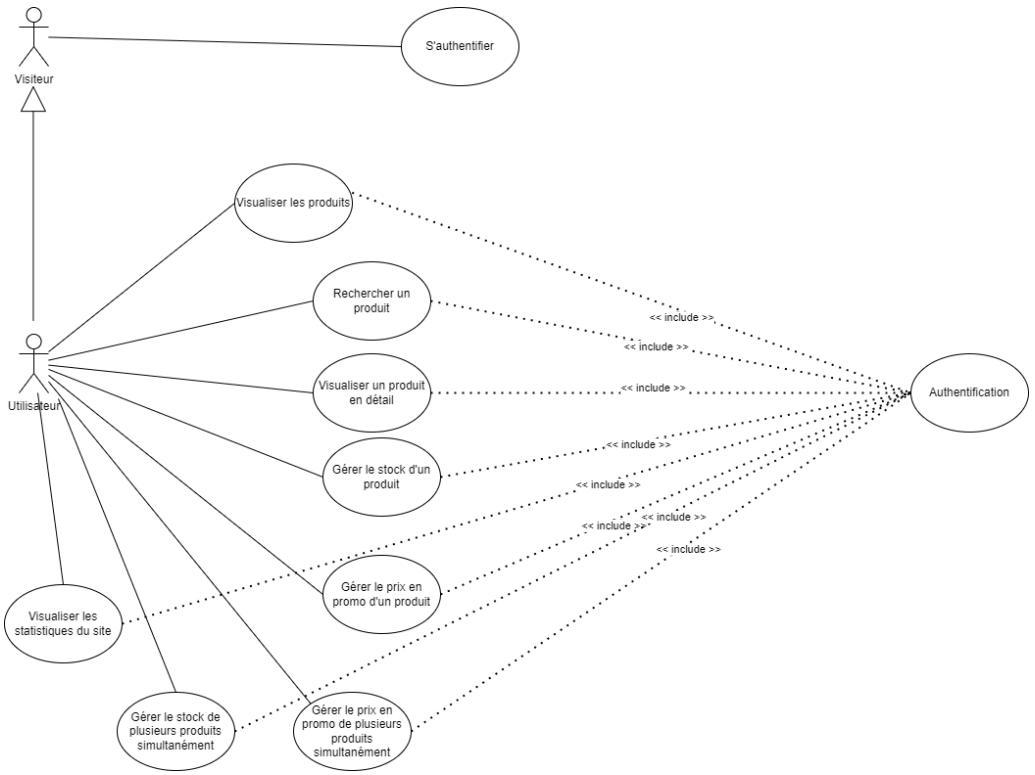


Diagramme d’activité - Connexion

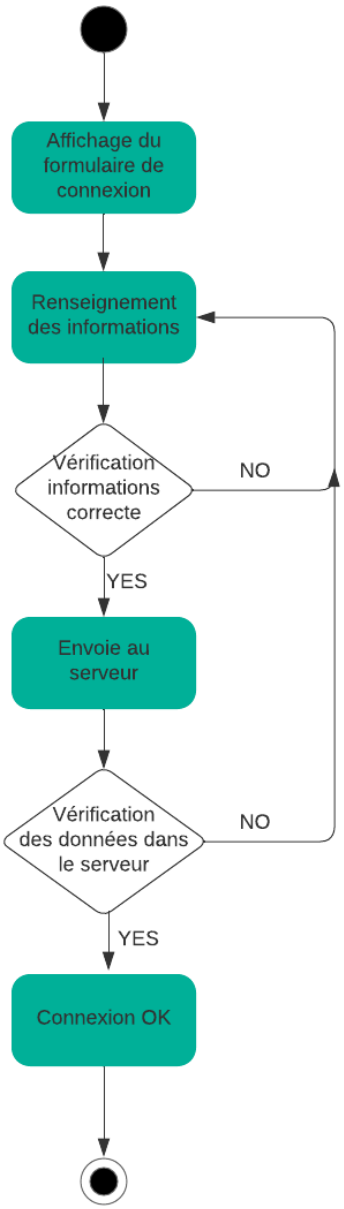
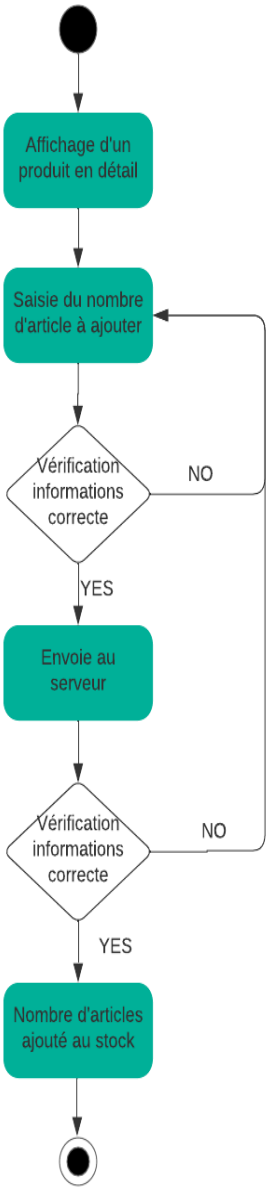


Diagramme d’activité - Gestion du stock d’un produit



Partie 2 : Maîtrise d'œuvre technique

Diagrammes de structure

Diagramme de classe

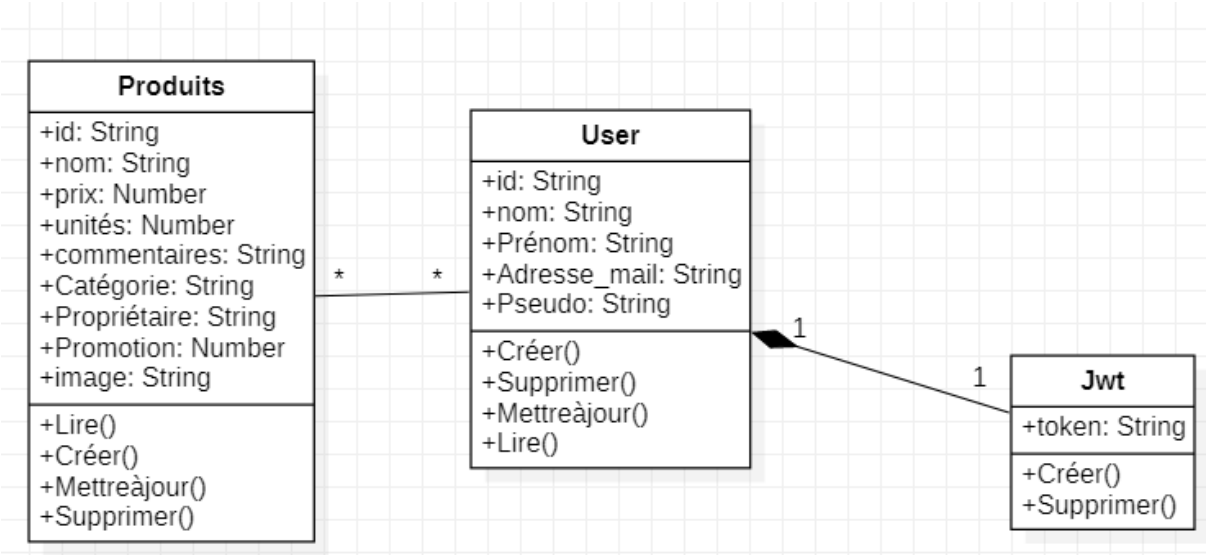
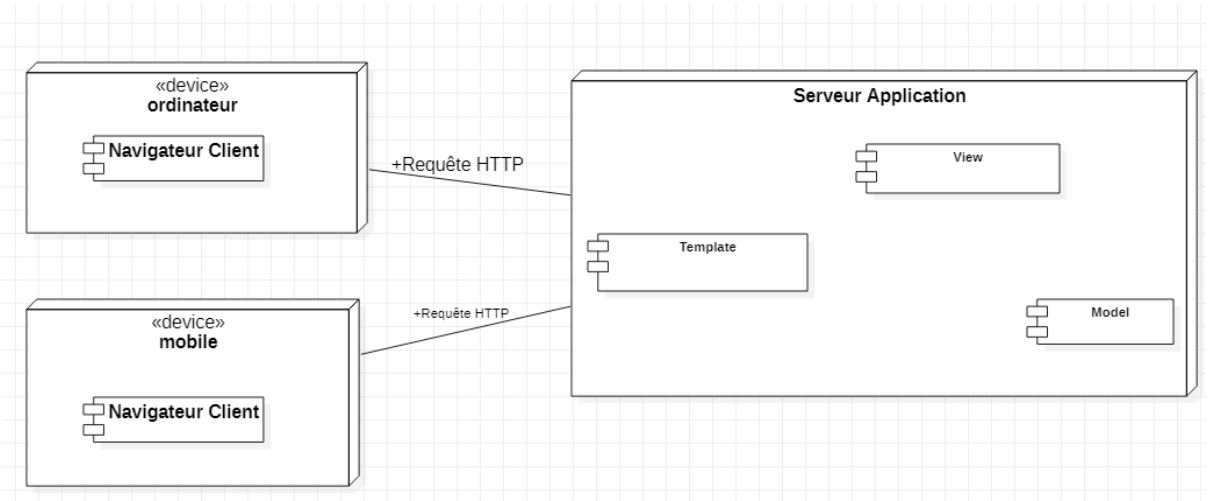
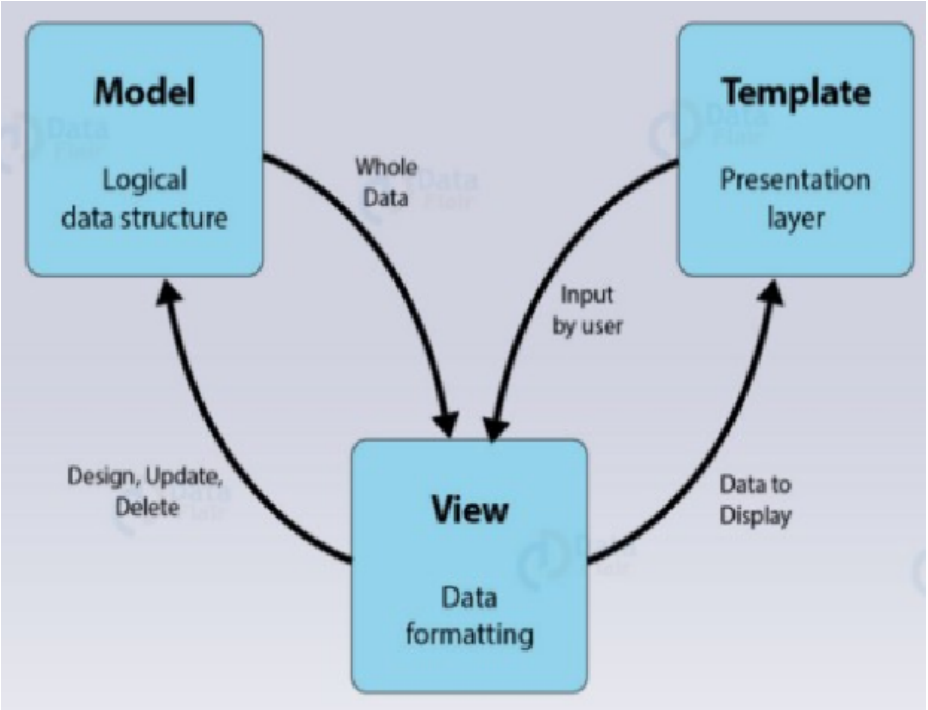


Diagramme de déploiement



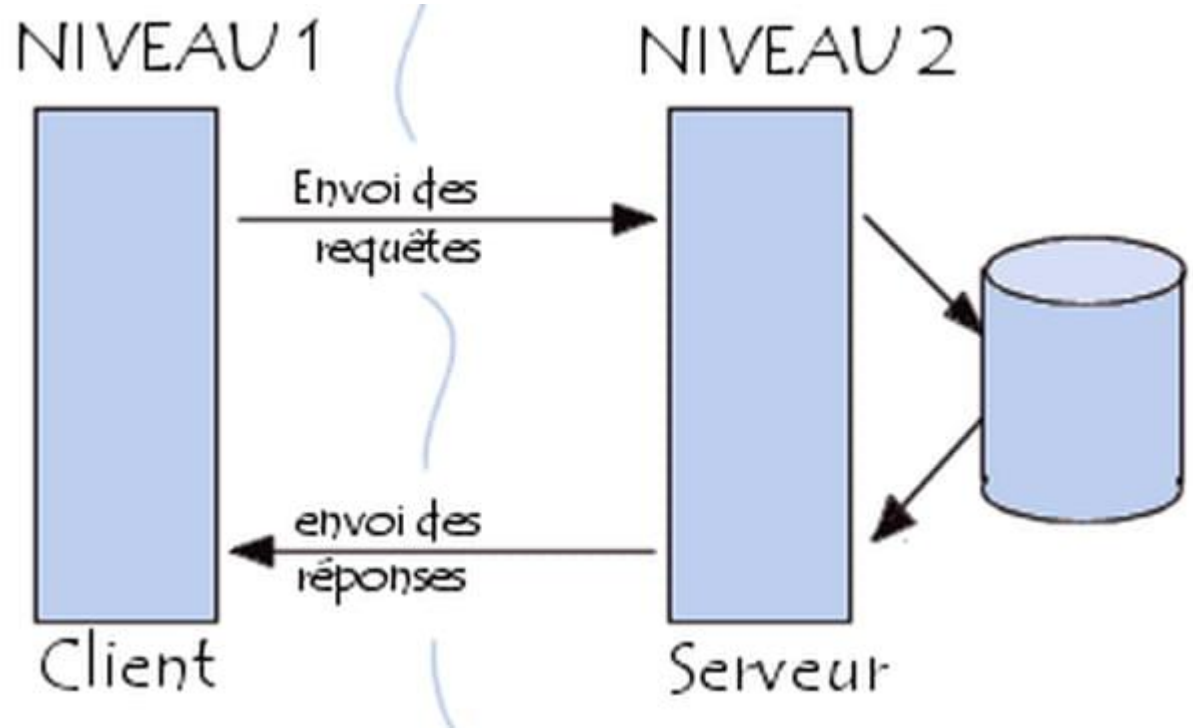
Architecture

Design Pattern MTV



L'architecture logicielle choisie est le Model - Template - View (MTV) car c'est l'architecture la plus adaptée pour la création de l'application web.

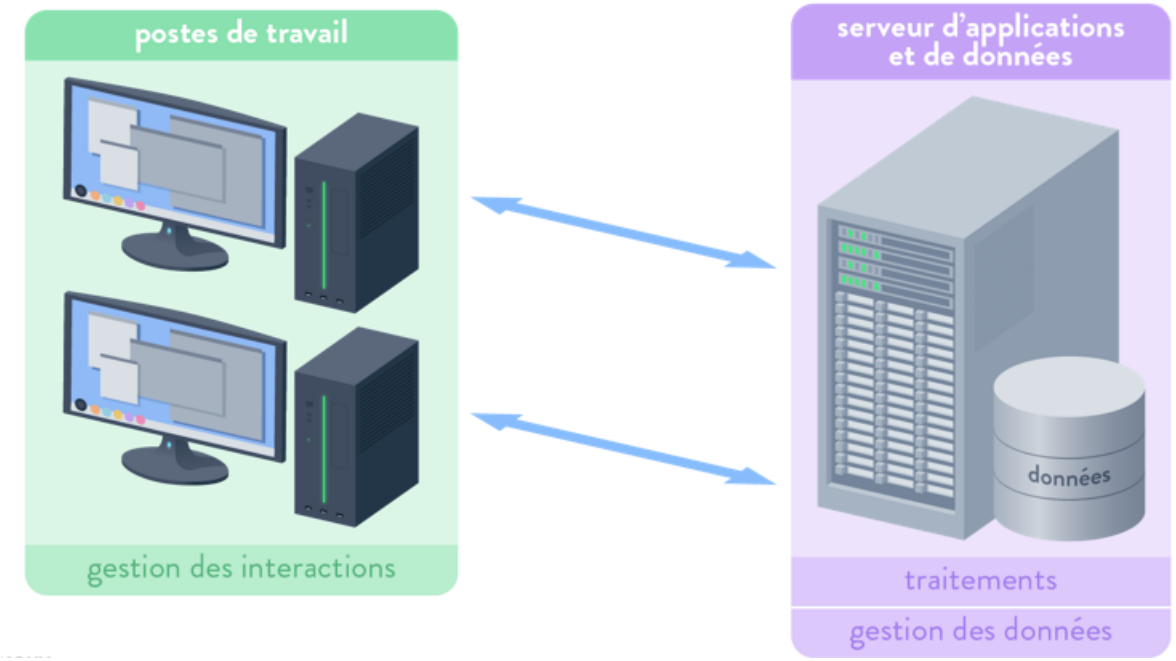
Architecture N-tiers



Voici une illustration de l'architecture 2 Tiers. Elle comporte 2 niveaux :

- La partie Client qui constitue l'Interface Homme Machine (IHM).
- La partie Métier, c'est la partie fonctionnelle qui réceptionne les requêtes HTTP, contrôle, stock et gère les données de l'application.

Architecture multi-couche



L'architecture multicouche est une illustration de notre application nous montre qu'il y a 2 couches.

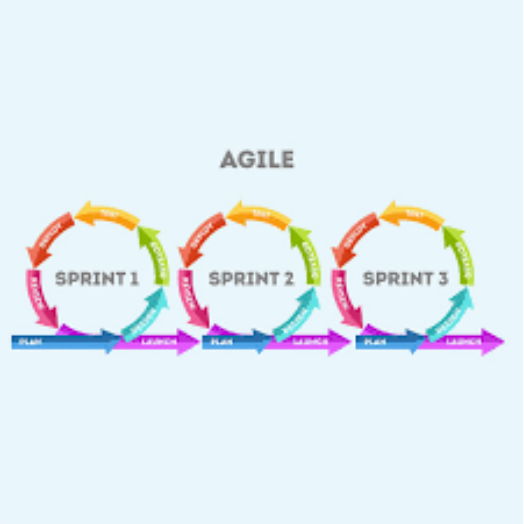
Gestion

La méthode utilisée lors de ce projet est la méthode Agile qu'on a adaptée à notre niveau.

La méthode Agile consiste, en résumé, à planifier étape par étape le projet, soit plus communément appelé l'itération et à organiser des points quotidiennement pour énoncer l'avancement de chacun.

Comparé à la méthode cycle en V qui consiste à conceptualiser et à documenter à 100% le projet avant de toucher au développement. Son inconvénient c'est qu'on ne peut pas modifier quelque chose et tout faire en amont du développement. Si le client souhaite une nouvelle fonctionnalité, on perdra du temps sur l'avancement.

Méthodologie



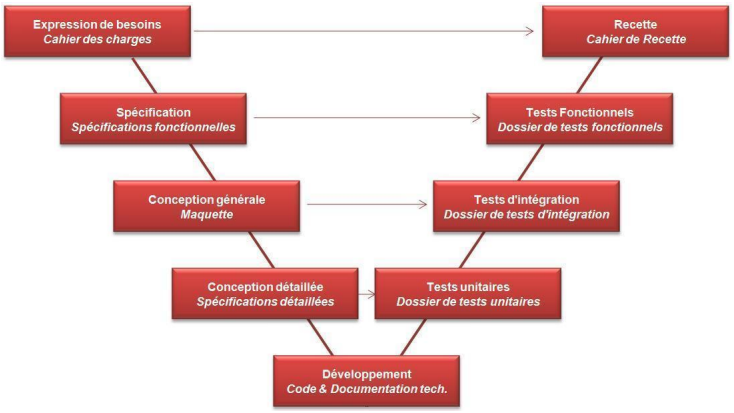
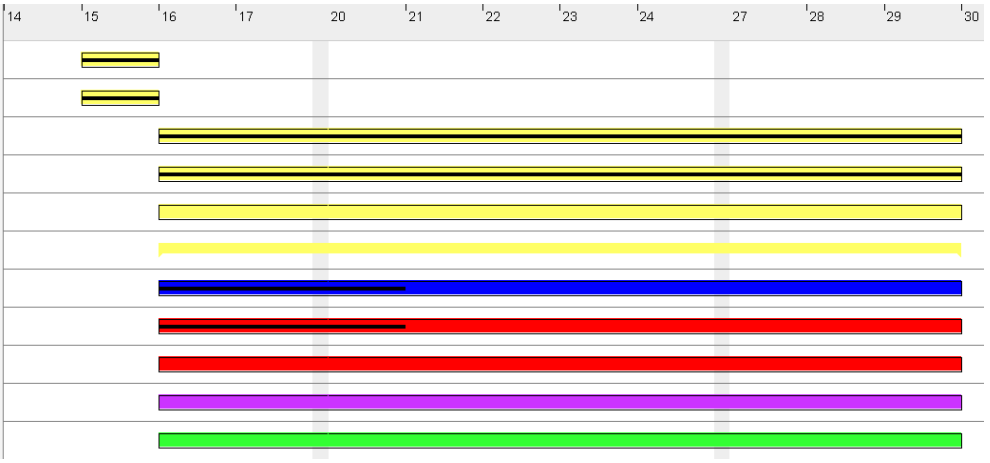


Diagramme de Gantt

Nom	Date de début	Date de fin
Rédaction du cahier des charges	15/03/2023	15/03/2023
Rédaction des spécifications fonctio...	15/03/2023	15/03/2023
Réalisation des maquettes	16/03/2023	29/03/2023
Conception des diagrammes fonctionnels...	16/03/2023	29/03/2023
Conception des diagrammes techni...	16/03/2023	29/03/2023
Développement	16/03/2023	29/03/2023
Authentification	16/03/2023	29/03/2023
Liste des produits	16/03/2023	29/03/2023
Produit en détail	16/03/2023	29/03/2023
Gestion des stock / promotions	16/03/2023	29/03/2023
Statistiques	16/03/2023	29/03/2023



On a établi un diagramme de gantt pour établir les tâches de chaque personne lors de ce projet et une estimation du temps pour l'accomplissement des tâches.

Partie 3 : Réalisation

Technologies et langages

Arborescence

Global back end

Settings.py

```
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    'corsheaders',
    "rest_framework",
    "rest_framework.authtoken",
    "rest_framework_simplejwt",
    "user",
    "product",
    "category",
    "owner",
    "discount",
    "image",
    "stock",
]
```

Global Front End

Router.ts

```
const routes: Array<RouteRecordRaw> = [
  {
    path: "/",
    name: "decider",
    component: DeciderView,
  },
  {
    path: "/dashboard",
    name: "dashboard",
    component: DashboardView,
  },
  {
    path: "/product-detail",
    name: "edit",
    component: ProductDetailsView,
  },
  {
    path: "/products",
    name: "products",
    component: ProductsView,
  },
  {
    path: "/login",
    name: "login",
    component: LoginView,
  },
  {
    path: "/about",
    name: "about",
    // route level code-splitting
    // this generates a separate chunk (about.[hash].js) for this route
    // which is lazy-loaded when the route is visited.
    component: () =>
      import(/* webpackChunkName: "about" */ "../views/AboutView.vue"),
  },
```

```
const router = createRouter({
  history: createWebHistory(process.env.BASE_URL),
  routes,
});

export default router;
```

constants.ts

```
const API_URL = "http://localhost:8000/";
const endPoints = {
  login: API_URL + "api/token/",
  user: API_URL + "user/",
  products: API_URL + "products/",
  stock: API_URL + "stock/",
};

export default endPoints;
```

Authentification - Backend

```
class UserView(APIView):
    permission_classes = (permissions.IsAuthenticated,)

    def get(self, request):
        return Response(UserSerializer(request.user).data)

You, last week | 1 author (You)
class UserCreateView(generics.CreateAPIView):
    queryset = UserModel.objects.all()
    permission_classes = (permissions.IsAuthenticated,)
    serializer_class = UserSerializer

You, last week | 1 author (You)
class UserConnectView(ObtainAuthToken):

    def post(self, request, *args, **kwargs):
        serializer = self.serializer_class(data=request.data,
                                           context={'request': request})
        serializer.is_valid(raise_exception=True)
        user = serializer.validated_data['user']
        token, created = Token.objects.get_or_create(user=user)
        user_serializer = UserSerializer(user)
        return Response({
            'token': token.key,
            'user': user_serializer.data,
        })      You, last week • user auth enhanced
```

```
UserModel = get_user_model()
```

```
class UserSerializer(ModelSerializer):
    You, last week | 1 author (You)
    class Meta:
        model = UserModel
        fields = (
            "username",
            "email",        You, last we
            "first_name",
            "last_name",
        )
```

```python
from rest_framework_simplejwt.views import (
    TokenObtainPairView,
    TokenRefreshView,
)

from .views import UserView, UserCreateView, UserConnectView

urlpatterns = [
    path('api/token/', TokenObtainPairView.as_view(), name='token_obtain_pair'),
    path('api/token/refresh/', TokenRefreshView.as_view(), name='token_refresh'),
    # path('api/token/verify/', TokenRefreshView.as_view(), name='token_verify'),
    path('user/', UserView.as_view(), name='user'),
    path('user/connect/', UserConnectView.as_view(), name='connection'),
    path('user/create/', UserCreateView.as_view(), name='registration'),
]
```

```python
urlpatterns = [
    path("admin/", admin.site.urls),
    path("", include("user.urls")),
    path("", include("product.urls")),
    path("", include("discount.urls")),
    path("", include("image.urls")),
    path("", include("category.urls")),
    path("", include("owner.urls")),
    path("", include("stock.urls")),
]
```

Authentification - Frontend

```
export default defineComponent({
  components: {
    TextField,
    FlatButton,
  },
  data() {
    return {
      username: "",
      password: "",
      error: "",
    };
  },
  methods: {
    ...mapActions(["login"]),
    setUsername(username: string) {
      this.username = username;
    },
    setPassword(password: string) {
      this.password = password;
    },
    validateUsername(username: string): string | null {
      if (username.length < 3) {
        return "Doit faire au moins 3 caractères";
      }
      return null;
    },
    validatePassword(password: string): string | null {
      if (password.length < 6) {
        return "Doit faire au moins 6 caractères";
      }
      return null;
    },
    async showErrorMessage(message: string) {
      this.error = message;
      await new Promise((resolve) => setTimeout(resolve, 3000));
      this.error = "";
    },
```

```
    async onLogin() {
      if (
        this.validateUsername(this.username) ||
        this.validatePassword(this.password)
      ) {
        console.log("Invalid login");
        return;
      }
      console.log("Logging in");
      this.login({ username: this.username, password: this.password }).then(
        (result) => {
          if (result.state) {
            this.$router.push({ name: "dashboard" });
          } else {
            this.showErrorMessage(result.message);
          }      Lbrzr, 4 days ago • trying toast …
        }
      );
    },
  },
});
```

Produits - Back - Model - Serializer

```
<template>
  <div class="login">
    <div class="login_content container">
      <h2 class="login_title">Connexion</h2>
      <TextField
        :onChange="setUsername"
        hint="Nom d'utilisateur"
        :required="true"
        :validator="validateUsername"
      />
      <TextField
        :onChange="setPassword"
        hint="Mot de passe"
        type="password"
        :required="true"
        :validator="validatePassword"
      />
      <FlatButton text="Se connecter" :onTap="onLogin"></FlatButton>
    </div>
  </div>
  <div
    class="error container"
    :class="{ toast: error.length > 0 }"
    v-if="error"
  >
    {{ error }}
  </div>
</template>
```

```python
class ProductModel(models.Model):
    name = models.CharField(max_length=80)
    comments = models.CharField(max_length=150, blank=True)
    unit = models.CharField(max_length=12)
    availability = models.BooleanField(default=False)
    price = models.FloatField()
    owner = models.ForeignKey(
        ProductOwnerModel, on_delete=models.DO_NOTHING, related_name='products')
    category = models.ForeignKey(
        CategoryModel, on_delete=models.DO_NOTHING, related_name='products')

    def discount(self):
        discount = self.discounts.order_by('start_date').last()
        if discount and discount.valid:
            return discount
        else:
            return None

    def discounted_price(self):
        discount = self.discount()
        if discount:
            return self.price - (self.price * discount.rate)
        else:
            return self.price

    def __str__(self):
        return f'{self.name} of {self.owner} in {self.category} at {self.price} per {self.unit}'
```

```python
class ProduitSerializer(serializers.ModelSerializer):
    # category = CategorySerializer()
    # owner = ProductOwnerSerializer()
    discount = DiscountSerializer(allow_null=True, required=False)

    images = ImageSerializer(many=True, required=False)

    category = serializers.PrimaryKeyRelatedField(
        queryset=CategoryModel.objects.all(), required=True)
    owner = serializers.PrimaryKeyRelatedField(
        queryset=ProductOwnerModel.objects.all(), required=True)
    # discount = serializers.PrimaryKeyRelatedField(queryset=DiscountModel.objects.all(), required=False)

    class Meta:
        model = ProductModel
        fields = (
            "id",
            "name",
            "price",
            "unit",
            "availability",
            "comments",
            "category",
            "owner",
            "discount",
            "images",
        )
        # set discount and images to optional
        extra_kwargs = {
            "discount": {"required": False},
            "images": {"required": False},
        }
```

```python
def create(self, validated_data):
    discount_data = validated_data.pop("discount", None)
    images_data = validated_data.pop("images", None)

    product = ProductModel.objects.create(**validated_data)

    if discount_data:
        DiscountModel.objects.create(product=product, **discount_data)
    if images_data:
        for image_data in images_data:
            ImageModel.objects.create(product=product, **image_data)

    return product

def to_representation(self, instance):
    data = super().to_representation(instance)
    data["category"] = CategorySerializer(instance.category).data
    data["owner"] = ProductOwnerSerializer(instance.owner).data
    data["images"] = ImageSerializer(instance.images.all(), many=True).data
    # discount = DiscountModel.objects.filter(product=instance).order_by("start_date").last()
    # if discount: data["discount"] = DiscountSerializer(discount).data
    return data
```

View

```python
class ProductListView(generics.ListAPIView):
    permission_classes = (permissions.IsAuthenticatedOrReadOnly,)
    queryset = ProductModel.objects.all()
    serializer_class = ProduitSerializer


class ProductCreateView(generics.CreateAPIView):
    permission_classes = (permissions.IsAuthenticated,)
    queryset = ProductModel.objects.all()
    serializer_class = ProduitSerializer


class ProductDetailView(generics.RetrieveUpdateDestroyAPIView):
    queryset = ProductModel.objects.all()
    serializer_class = ProduitSerializer
```

```python
urlpatterns = [
    path('products/', ProductListView.as_view()),
    path('product/create/', ProductCreateView.as_view()),
    path('product/<int:pk>/', ProductDetailView.as_view()),
]
```

Dashboard

```vue
<script lang="ts">
import { defineComponent } from "vue";
import { mapActions } from "vuex";
import AppBarView from "@/components/AppBar.vue";
import StatsView from "@/components/Stats.vue";

export default defineComponent({
  name: "DashboardView",
  components: {
    AppBarView,
    StatsView,
  },
  async created(): Promise<void> {
    console.log("Redeciding...");
    if (!(await this.isAuthenticated())) {
      console.log("Waiting for authentication");
      this.$router.push({ name: "login" });
    }
    console.log("Authenticated");
  },
  methods: {
    ...mapActions(["isAuthenticated"]),
  },
});
</script>
```

```vue
<template>
  <div class="home">
    <AppBarView title="Dashboard" />
    <div class="content">
      <div class="subtitle">Bienvenue sur la page d'accueil</div>
      <StatsView />
    </div>
  </div>
</template>      You, 5 days ago • big involves, must be tested
```

```
methods: {
  ...mapActions(["fetchStock"]),
  setSumByType() {
    console.log("sumByType: ", this.sumByType);
    switch (this.sumByType) {
      case "Jour" as SumByType:
        this.sumBy = this.sumByDay;
        break;
      case "Mois" as SumByType:
        this.sumBy = this.sumByMonth;
        break;
      case "Année" as SumByType:
        this.sumBy = this.sumByYear;
        break;
    }
    console.log("sumBy: ", this.sumBy);
  },
```

```
type SumByType = "Jour" | "Mois" | "Année";

export default defineComponent({
  name: "StatsView",
  components: {
    VueApexCharts,
  },
  mounted() {
    this.fetchStock().then((data) => {
      console.log("stock fetched");
      this.stock = data;
      this.sumByDay = this.getSumByDay();
      this.sumByMonth = this.getSumByMonth();
      this.sumByYear = this.getSumByYear();
      this.setSumByType();
    });
  },
```

```
getSumByDay() {
  let out: number[] = [];
  let expense: number[] = [];
  let label: string[] = [];
  let mainData = this.stock;
  mainData = mainData.sort(
    (a: StockMove, b: StockMove) => a.date.getTime() - b.date.getTime()
  );
  // console.log(mainData.length)
  for (let index = 0; index < mainData.length; index++) {
    // console.log(index)
    let date = mainData[index].date;
    // console.log(date)
    let localeDate = date.toLocaleDateString();
    if (label.includes(localeDate)) {
      // console.log('include')
      //if amount is sell
      if (mainData[index].type == "OUT") {
        out[label.indexOf(localeDate)] += mainData[index].price;
        expense[label.indexOf(localeDate)] += 0;
      } else {
        expense[label.indexOf(localeDate)] += mainData[index].price;
        out[label.indexOf(localeDate)] += 0;
      }
    } else {
      // console.log('else include')
      label.push(localeDate);
      if (mainData[index].type == "OUT") {
        out.push(mainData[index].price);
        expense.push(0);
      } else {
        expense.push(mainData[index].price);
        out.push(0);
      }
```

```
getSumByDay() {
  let out: number[] = [];
  let expense: number[] = [];
  let label: string[] = [];
  let mainData = this.stock;
  mainData = mainData.sort(
    (a: StockMove, b: StockMove) => a.date.getTime() - b.date.getTime()
  );
  // console.log(mainData.length)
  for (let index = 0; index < mainData.length; index++) {
    // console.log(index)
    let date = mainData[index].date;
    // console.log(date)
    let localeDate = date.toLocaleDateString();
    if (label.includes(localeDate)) {
      // console.log('include')
      //if amount is sell
      if (mainData[index].type == "OUT") {
        out[label.indexOf(localeDate)] += mainData[index].price;
        expense[label.indexOf(localeDate)] += 0;
      } else {
        expense[label.indexOf(localeDate)] += mainData[index].price;
        out[label.indexOf(localeDate)] += 0;
      }
    } else {
      // console.log('else include')
      label.push(localeDate);
      if (mainData[index].type == "OUT") {
        out.push(mainData[index].price);
        expense.push(0);
      } else {
        expense.push(mainData[index].price);
        out.push(0);
      }
```

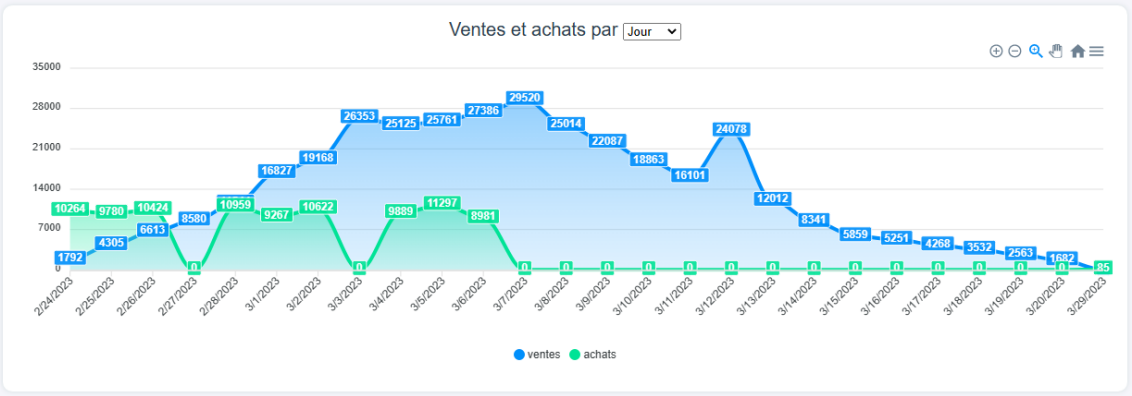
```
<main>
  <h5 class="text-3xl m-8">Stats de ventes :</h5>
  <div class="m-8 cards">
    <div class="flex stat_container">
      <div class="title text-xl mr-5">Total des ventes</div>
      <div class="text-xl">{{ sumOut }} €</div>
    </div>
    <div class="flex stat_container">
      <div class="title text-xl mr-5">Total des achats</div>
      <div class="text-xl">{{ sumIn }} €</div>
    </div>
    <div class="flex stat_container">
      <div class="title text-xl mr-5">Chiffre d'affaire total</div>
      <div class="text-xl">{{ sumIn + sumOut }} €</div>
    </div>
  </div>
  <div class="m-8 stat_container">
    <div class="text-2xl pt-5">
      Ventes et achats par
      <select v-model="sumByType" @change="(event) => setSumByType()">
        <option v-for="type in sumByTypes" :key="type" :value="type">
          {{ type }}
        </option>
      </select>
    </div>
    <VueApexCharts
      type="area"
      height="350"
      ref="chart"
      :options="sumBy.chartOptions"
      :series="sumBy.series"
    ></VueApexCharts>
  </div>
```

Dashboard Produit

Stats de ventes :

C:\Dev\Projects\Dac\pharma_map\assets\data\data.json

Total des ventes	Total des achats	Chiffre d'affaire total
352625 €	91568 €	444193 €



Liste des produits

```python
class Command(BaseCommand):
    help = 'Inject predefined list of products.'

    distant_server_product_end_point = 'http://51.255.166.155:1352/tig/products/'

    def handle(self, *args, **options):
        self.stdout.write(self.style.SUCCESS('Injecting products...'))
        self.inject_products()
        self.stdout.write(self.style.SUCCESS(
            'Products injected successfully!'))

    def fetch_products(self) -> list:
        response = requests.get(self.distant_server_product_end_point)
        return response.json()

    def format_product(self, product: dict) -> dict:
        return {
            'name': product.get('name'),
            'comments': product.get('comments'),
            'unit': product.get('unit'),
            'availability': product.get('availability'),
            'price': product.get('price'),
            'owner': 1,  # product.get('owner'),
            'category': product.get('category') + 1,
            'discount': {
                "rate": product.get('discount'),
            } if product.get('discount') else None,
        }
```

```python
def inject_products(self):
    products = self.fetch_products()
    for product in products:
        product = self.format_product(product)
        product_serializer = ProduitSerializer(data=product)
        if product_serializer.is_valid():
            product_serializer.save()
            self.stdout.write(self.style.SUCCESS(
                f'Product injected: {product.get("name")}'))
        else:
            self.stdout.write(self.style.ERROR(
                f'Error while injecting product: {product}'))
            self.stdout.write(self.style.ERROR(
                f'Error: {product_serializer.errors}'))
```

```python
class Command(BaseCommand):
    help = 'Erase product list.'

    def handle(self, *args, **options):
        self.stdout.write(self.style.SUCCESS('Crushing products...'))
        self.crush_products()
        self.stdout.write(self.style.SUCCESS(
            'Products crushed successfully!'))

    def crush_products(self):
        ProductModel.objects.all().delete()
```

```
methods: {
  ...mapActions(["fetchProducts"]),
  filter(term: string) {
    this.filteredProducts = this.products.filter((product: Product) => {
      return product.name.toLowerCase().includes(term.toLowerCase());
    });
  },
  onSelect(product: Product) {
    this.selected = product;
  },
  async goToStock() {
    return this.$router.push({ name: "stock" });
  },
},
mounted() {
  this.fetchProducts().then((products) => {
    this.products = [];
    products.forEach((val: Product) => {
      this.products.push(Object.assign({}, val));
    });
    this.filteredProducts = products;
    // this.keys = Object.keys(products[0]);
    this.keys = [
      "id",
      "name",
      "price",
      "availability",
      "comments",
      "category",
      "owner",
      "discount",
    ];
  });
},
```

```
export default defineComponent({
  name: "ProductsView",
  components: {
    FlatButton,
    TextField,
    TableView,
    AppBarView,
  },
  data() {
    return {
      // keys: Object.keys(store.state.fishes[0]),
      keys: [] as string[],
      products: [] as Product[],
      filteredProducts: [] as Product[],
      selected: {} as Product,
      expandFields: ["comments", "name"],
    };
  },
```

Gestion des stocks & promo

```python
class StockMoveItemAdminModel(admin.ModelAdmin):
    list_display = ('id', 'product', 'amount', 'stock_move',)
    list_filter = ('stock_move',)
    search_fields = ('product', 'amount', 'stock_move',)
    list_per_page = 25


class StockMoveAdminModel(admin.ModelAdmin):
    list_display = ('id', 'date', 'price', 'type',)
    list_filter = ('date', 'type',)
    search_fields = ('date', 'type',)
    list_per_page = 25


admin.site.register(StockMoveItemModel, StockMoveItemAdminModel)
admin.site.register(StockMoveModel, StockMoveAdminModel)
```

```python
class StockMoveModel(models.Model):
    date = models.DateTimeField(default=datetime.now())
    type = models.CharField(choices=[(x, x)
                            for x in _stockMoveTypes], max_length=3,)
    price = models.DecimalField(
        max_digits=10, decimal_places=2, null=True, blank=True,)

    def products(self):
        return self.items.all()

    def calculate_update_price(self):
        total = 0
        for item in self.items.all():
            # if there is a valid discount on product
            if item.product.discount() and item.product.discount().valid:
                total += item.product.discounted_price() * item.amount
            else:
                total += item.product.price * item.amount
        self.price = total
        self.save()

    def __str__(self):
        return f"{self.date} - {self.type}"


class StockMoveItemModel(models.Model):
    product = models.ForeignKey(
        ProductModel, on_delete=models.CASCADE, related_name="stock_move_items",
    )
    amount = models.PositiveIntegerField()
    stock_move = models.ForeignKey(
        StockMoveModel,
        on_delete=models.CASCADE,
        related_name="items",
    )
```

```python
def __str__(self):
    return f"{self.product.name} - {self.amount}"
```

```python
class StockMoveSerializer(serializers.ModelSerializer):
    products = StockMoveItemSerializer(many=True)

    class Meta:
        model = StockMoveModel
        fields = (
            "id",
            "date",
            "type",
            "price",
            "products",
        )

    def create(self, validated_data):
        items_data = validated_data.pop("products")
        stock_move = StockMoveModel.objects.create(**validated_data)
        for item_data in items_data:
            StockMoveItemModel.objects.create(
                stock_move=stock_move, **item_data)
        if 'price' not in validated_data:
            stock_move.calculate_update_price()
        return stock_move

    items_field_name = "products"  # "items"

    def to_representation(self, instance):
        data = super().to_representation(instance)
        data[self.items_field_name] = StockMoveItemSerializer(
            instance.items.all(), many=True).data
        return data
```

```python
class StockMoveItemSerializer(serializers.ModelSerializer):
    stock_move = serializers.PrimaryKeyRelatedField(
        queryset=StockMoveModel.objects.all(),
        required=False,
    )

    class Meta:
        model = StockMoveItemModel
        fields = "__all__"
        # set stock_move to optional
        extra_kwargs = {
            "stock_move": {"required": False},
        }

    def to_representation(self, instance):
        data = super().to_representation(instance)
        data.pop("stock_move")
        return data
```

```python
class StockMoveListView(generics.ListAPIView):
    permission_classes = (permissions.IsAuthenticatedOrReadOnly,)
    serializer_class = StockMoveSerializer
    queryset = StockMoveModel.objects.all()


class StockMoveCreateView(generics.CreateAPIView):
    permission_classes = (permissions.IsAuthenticated,)
    serializer_class = StockMoveSerializer
    queryset = StockMoveModel.objects.all()


class StockMoveDetailView(generics.RetrieveUpdateDestroyAPIView):
    permission_classes = (permissions.IsAuthenticated,)
    serializer_class = StockMoveSerializer
    queryset = StockMoveModel.objects.all()


class StockMoveItemListView(generics.ListAPIView):
    permission_classes = (permissions.IsAuthenticatedOrReadOnly,)
    serializer_class = StockMoveItemSerializer
    queryset = StockMoveItemModel.objects.all()


class StockMoveItemCreateView(generics.CreateAPIView):
    permission_classes = (permissions.IsAuthenticated,)
    serializer_class = StockMoveItemSerializer
    queryset = StockMoveItemModel.objects.all()


class StockMoveItemDetailView(generics.RetrieveUpdateDestroyAPIView):
    permission_classes = (permissions.IsAuthenticated,)
    serializer_class = StockMoveItemSerializer
    queryset = StockMoveItemModel.objects.all()
```

```python
urlpatterns = [
    path("stock/", StockMoveListView.as_view()),
    path("stock/move/create/", StockMoveCreateView.as_view()),
    path("stock/move/<int:pk>/", StockMoveDetailView.as_view()),
    path("stock/move/items", StockMoveItemListView.as_view()),
]
```

Components

```
export type OnTapFunction = () => Promise<any>;

export default defineComponent({
  name: "FlatButton",
  components: {
    LottieAnimation,
  },
  props: {
    onTap: { type: Function as PropType<OnTapFunction>, required: true },
    text: { type: String, required: true },
    elevate: { type: Boolean, default: true, required: false },
  },
  data() {
    return {
      loading: false,
      animationPath: "lotties/loading-dots.json",
    };
  },
  methods: {
    react(): void {
      if (this.loading) {
        console.log("Already reacting");
      } else {
        console.log("reacting");
        this.loading = true;
        this.onTap().then(() => {
          this.loading = false;
        });
      }
    },
  },
});
```

```vue
<template>
  <span
    class="flat_button"
    :class="{
      reacting: loading,
      elevated: elevate,
      text: true,
    }"
    @click="react"
  >
    <div v-if="loading" class="loading">
      <LottieAnimation
        :path="animationPath"
        :autoplay="true"
        :loop="true"
        :height="20"
      />
    </div>
    <span v-else>
      {{ text }}
    </span>
  </span>
</template>
```

```vue
  <div class="app_bar">
    <h2 class="app_bar_title">{{ title }}</h2>
  </div>
</template>

<script lang="ts">
import { defineComponent } from "vue";

export default defineComponent({
  name: "AppBarView",
  props: {
    title: {
      type: String,
      required: true,
    },
  },
  components: {},
  methods: {},
});
</script>
```

```ts
export type OnChangeFunction = (value: string) => void;
export type ValidatorFunction = (value: string) => null | string;

export default defineComponent({
  name: "TextField",
  components: {},
  props: {
    onChange: { type: Function as PropType<OnChangeFunction>, required: true },
    validator: {
      type: Function as PropType<ValidatorFunction>,
      required: false,
    },
    hint: { type: String, required: false },
    type: { type: String, default: "text", required: false },
    required: { type: Boolean, default: false, required: false },
    id: { type: String, required: false },
  },
  data() {
    return {
      input: "",
    };
  },
  methods: {
    reactOnChange() {
      console.log("this.input: ", this.input);
      this.onChange(this.input);
    },
    validate(value: string): null | string {
      if (this.validator && this.input.length > 0) {
        console.log("validating");
        return this.validator(value);
      } else {
        return null;
      }
    },
```

```vue
<template>
  <input
    class="text_field"
    :id="id ? id : hint"
    :type="type"
    :class="{ invalid: validator && !isValid() }"
    :placeholder="hint"
    v-bind:required="required"
    v-on:input="reactOnChange"
    v-model="input"
  />
  <div class="validation_text" v-if="validator && !isValid()">
    {{ validate(input) }}
  </div>
</template>
```

```
  validate(value: string): null | string {
    if (this.validator && this.input.length > 0) {
      console.log("validating");
      return this.validator(value);
    } else {
      return null;
    }
  },
  isValid(): boolean {
    return this.validate(this.input) === null;
  },
},
```

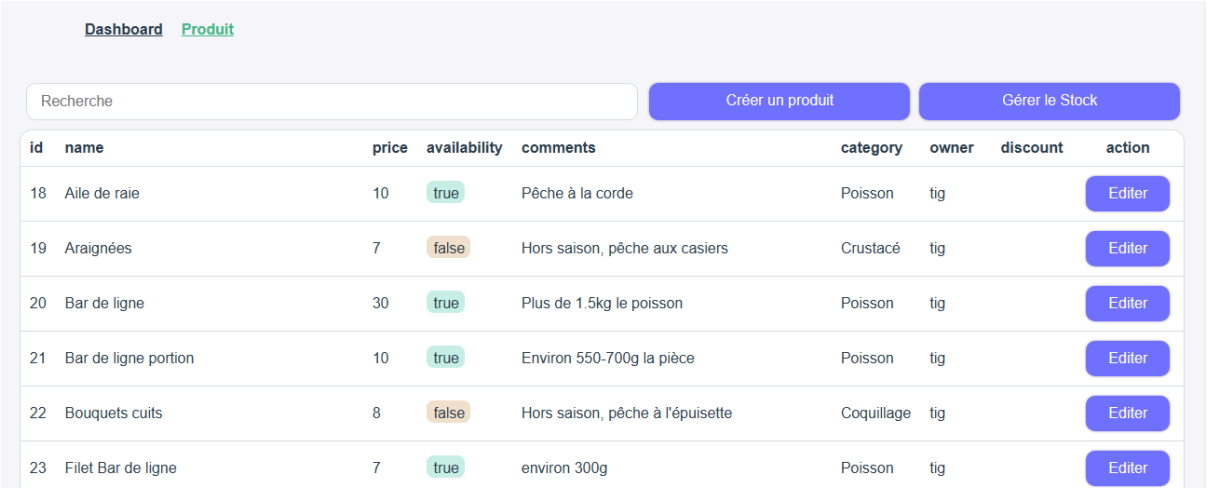
```ts
export default defineComponent({
  name: "ProductTile",
  components: {
    FlatButton,
  },
  props: {
    keys: { type: Object as PropType<string[]>, requiered: true },
    product: { type: Object as PropType<Product>, requiered: true },
    mayExpand: {
      type: Function as PropType<(key: string) => boolean>,
      requiered: false,
      default: () => false,
    },
  },
  methods: {
    // : string | number | boolean
    get(key: string) {
      let field = this.product![key as keyof Product];
      return field;
    },
    async goToEdit() {
      console.log("go to edit");
      return this.$router.push({
        name: "edit",
        params: { id: this.product?.id },
      });
    },
  },
});
```

```vue
<template>
  <td v-for="key in keys" v-bind:key="key" :class="{ expand: mayExpand(key) }">
    <span
      v-if="typeof get(key) == 'boolean'"
      class="badge"
      :class="{ good: get(key), bad: !get(key) }"
    >
      {{ get(key) }}
    </span>
    <span v-else>
      {{ get(key) }}
    </span>
  </td>
  <td class="action_td">
    <FlatButton :onTap="goToEdit" text="Editer" />
  </td>
</template>
```

```
export default defineComponent({
  name: "TableView",
  components: {
    ProductTile,
  },
  props: {
    products: {
      type: Object as PropType<Product[]>,
      required: true,
    },
    keys: {
      type: Array as PropType<string[]>,
      required: true,
    },
    onSelect: {
      type: Function as PropType<(product: Product) => void>,
      required: false,
    },
    selected: {
      type: Object as PropType<Product>,
      required: false,
    },
    expandFields: {
      type: Array as PropType<string[]>,
      required: false,
      default: [] as string[],
    },
  },
```

```
<template>
  <table id="v-for-object">
    <thead>
      <tr>
        <th
          scope="col"
          v-for="key in keys"
          v-bind:key="key"
          :class="{ action_td: key == 'action' }"
        >
          {{ key }}
        </th>
        <th class="action_td">action</th>
      </tr>
    </thead>
    <tbody>
      <tr
        v-for="product in products"
        v-bind:key="product.id"
        :class="{ selected: product.id == selected?.id }"
        v-on:click="($event) => (onSelect ? onSelect(product) : null)"
      >
        <ProductTile
          v-bind:keys="keys"
          v-bind:product="product"
          v-bind:mayExpand="(key: any) => expandFields.includes(key)"
        ></ProductTile>
      </tr>
    </tbody>
  </table>
  <div class="end_table">{{ products.length }} poissons.</div>
</template>
```

```
<template>
  <div class="home">
    <AppBarView title="Produits" />
    <div class="bar">
      <TextField v-bind:onChange="filter" hint="Recherche"></TextField>
      <FlatButton text="Nouveau" :onTap="goToStock"></FlatButton>
      <FlatButton text="Stock" :onTap="goToStock"></FlatButton>
    </div>
    <!-- FilterBar></FilterBar -->
    <TableView
      v-bind:products="filteredProducts"
      v-bind:keys="keys"
      v-bind:onSelect="onSelect"
      v-bind:selected="selected"
      v-bind:expandFields="expandFields"
    ></TableView>
  </div>
</template>
```



Partie 4 : Bilan

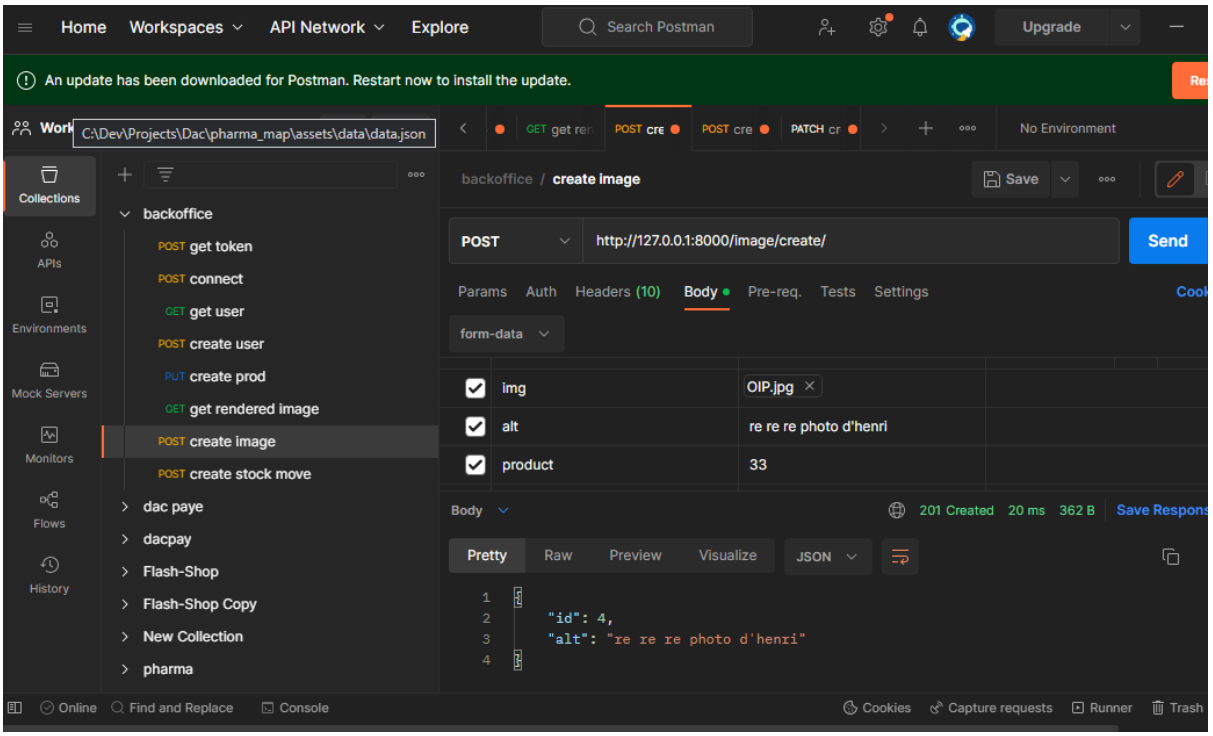
Avenir du projet

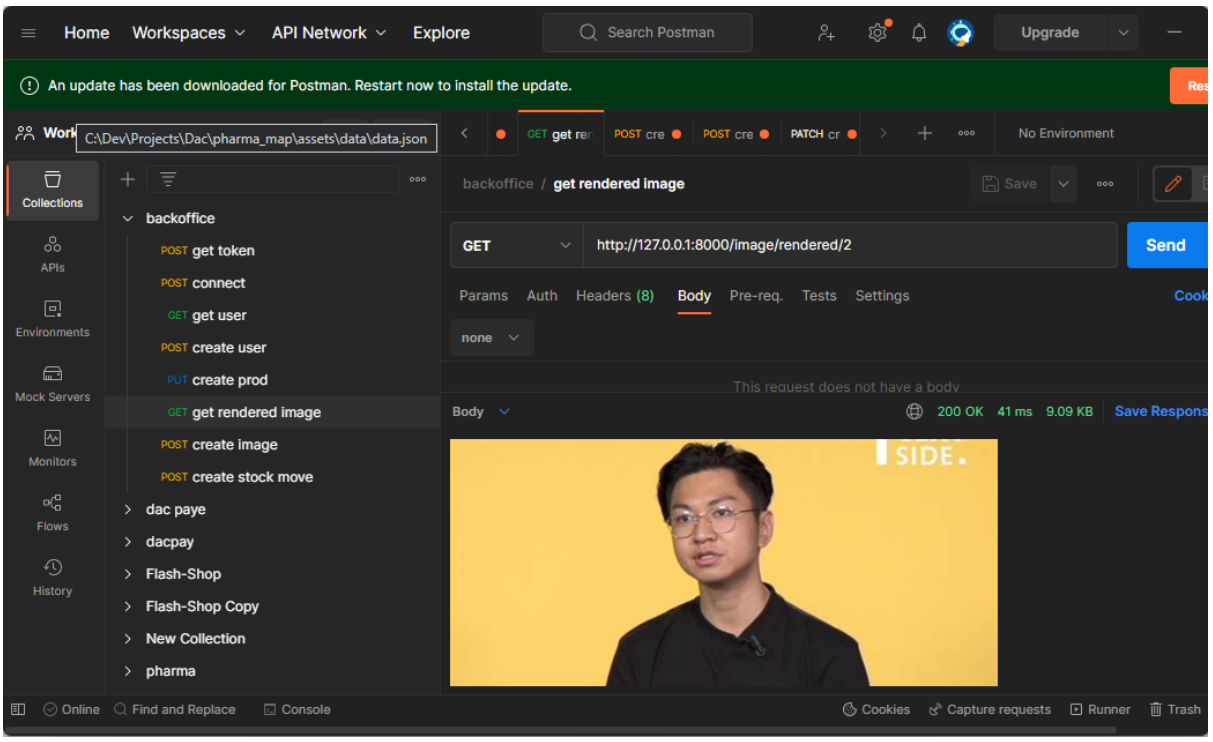
Conclusion

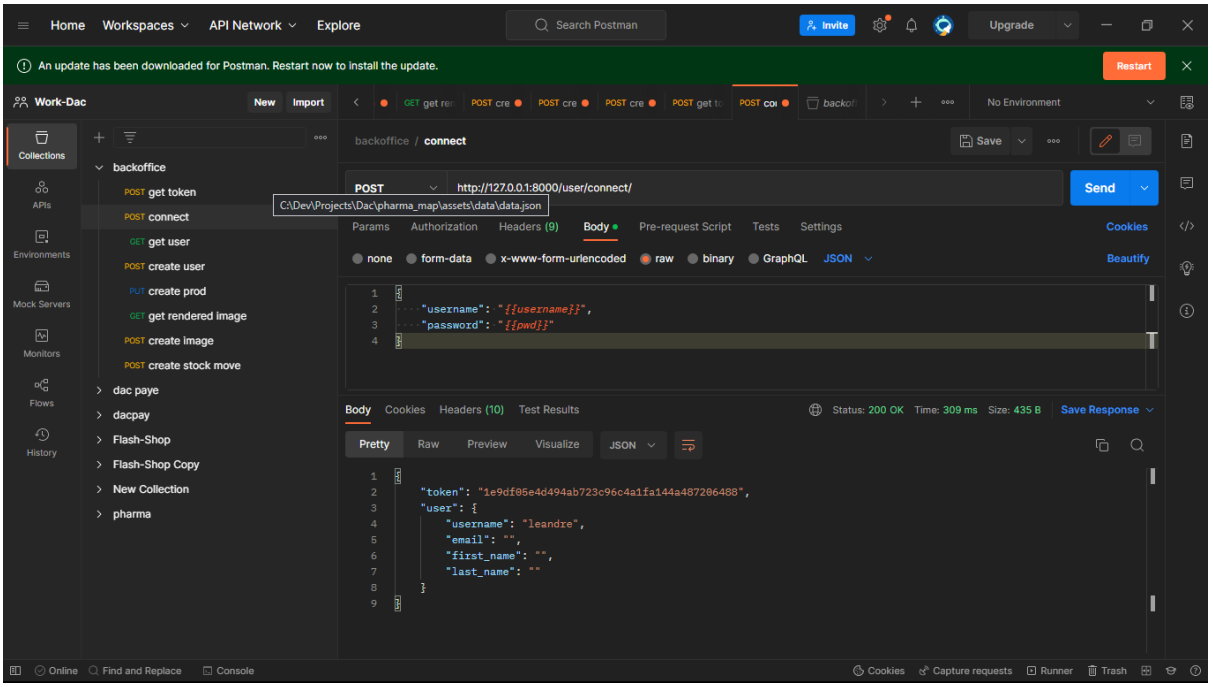
Ce projet nous a fait acquérir beaucoup de compétences dans la conception et le développement d'une application web en Django et Vue.

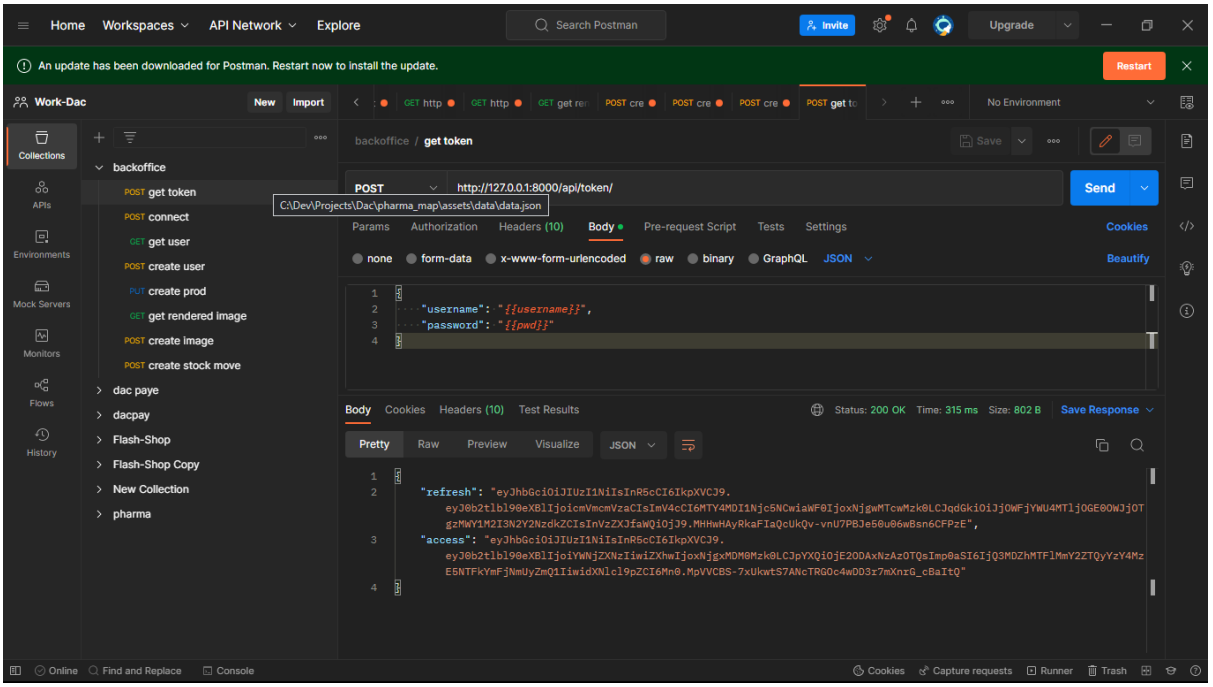
Le fait d'être de travailler en équipe selon une méthode définie et le fait de chercher les solutions aux problèmes rencontrés nous a fait monter en compétences en tant qu'Architecte logicielle.

Nous vous remercions d'avoir pris le temps de lire ce rapport.






Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
An update has been downloaded for Postman. Restart now to install the update.
Restart
Work-Dac
New
Import
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
backoffice
POST get token
POST connect
GET get user
POST create user
PUT create prod
GET get rendered image
POST create image
POST create stock move
dac paye
dacpay
Flash-Shop
Flash-Shop Copy
New Collection
pharma
C:\Dev\Projects\Dac\pharma_map\assets\data\data.json
backoffice / connect
Save
POST
http://127.0.0.1:8000/user/connect/
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"username": "{{username}}",
"password": "{{pwd}}"
}
Body
Cookies
Headers (10)
Test Results
Status: 200 OK
Time: 309 ms
Size: 435 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"token": "1e9df05e4d494ab723c96c4a1fa144a487206488",
"user": {
"username": "leandre",
"email": "",
"first_name": "",
"last_name": ""
}
}
Online
Find and Replace
Console
Cookies
Capture requests
Runner
Trash


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
An update has been downloaded for Postman. Restart now to install the update.
Restart
Work-Dac
New
Import
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
backoffice
POST get token
POST connect
GET get user
POST create user
PUT create prod
GET get rendered image
POST create image
POST create stock move
dac paye
dacpay
Flash-Shop
Flash-Shop Copy
New Collection
pharma
C:\Dev\Projects\Dac\pharma_map\assets\data\data.json
backoffice / get token
Save
POST
http://127.0.0.1:8000/api/token/
Send
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"username": "{{username}}",
"password": "{{pwd}}"
}
Body
Cookies
Headers (10)
Test Results
Status: 200 OK
Time: 315 ms
Size: 802 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"refresh": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJ0b2tlbl90eXBlIjoicmVmcmVzaCIsImV4cCI6MTY4MDI1Njc5NCwiaWF0IjoxNjgwMTcwMzk0LCJqdGkiOiJjOWFjYWU4MTljOGE0OWJjOTgzMWY1M2I3N2Y2NzdkZCIsInVzZXJfaWQiOjJ9.MHHwHAyRkaFIaQcUkQv-vnU7PBJe50u06wBsn6CFPzE",
"access": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNjgxMDM0Mzk0LCJpYXQiOjE2ODAxNzAzOTQsImp0aSI6IjQ3MDZhMTFlMmY2ZTQyYzY4MzE5NTFkYmFjNmUyZmQ1IiwidXNlcl9pZCI6Mn0.MpVVCBS-7xUkwtS7ANcTRGOc4wDD3z7mXnrG_cBaItQ"
}
Online
Find and Replace
Console
Cookies
Capture requests
Runner
Trash

```
actions: {
  // login user and store access and refresh tokens in local storage
  login(
    context,
    credentials: { username: string; password: string }
  ): Promise<Result> {
    return fetch(endPoints.login, {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(credentials),
    })
      .then((response) => {
        if (response.status === 200) {
          return response.json();
        } else {
          throw new Error("invalid credentials");
        }
      })
      .then((data) => {
        console.log("data: ", data);
        console.log("login successful");
        localStorage.setItem("access", data.access);
        localStorage.setItem("refresh", data.refresh);
        context.dispatch("fetchUser");
        return {
          code: 200,
          message: "login successful",
          state: true,
        };
```

```
export default createStore({
  state: {
    products: [] as Product[],
    user: {} as User,
    stock: [] as StockMove[],
  },
  getters: {
    products(state) {
      return state.products;
    },
    user(state) {
      return state.user;
    },
    stock(state) {
      return state.stock;
    },
  },
```

```
mutations: {
  setProducts(state, products) {
    state.products = products;
  },
  setUser(state, user) {
    state.user = user;
  },
},
```

```
port type User = {
id: number;
username: string;
email?: string;
first_name?: string;
last_name?: string;


port type ProductOwner = {
id: number;
name: string;


port type Category = {
id: number;
name: string;
description?: string;


port type Discount = {
id: number;
name?: string;
description?: string;
rate: number;
startDate: Date;
endDate?: Date;
```

```
export type Product = {
  id: number;
  name: string;
  category: string;
  price: number;
  unit: string;
  availability: boolean;
  sale: boolean;
  discount: number;
  comments: string;
  owner: string;
};

export type StockMoveItem = {
  id: number;
  product: number;
  amount: number;
};
```

```
  constructor(
    id: number,
    date: Date,
    products: StockMoveItem[],
    type: "IN" | "OUT",
    price: number
  ) {
    this.id = id;
    this.date = date;
    this.products = products;
    this.type = type;
    this.price = price;
  }

  static fromJson(json: any): StockMove {
    return new StockMove(
      json.id,
      new Date(json.date),
      json.products as StockMoveItem[],
      json.type,
      Number(json.price)
    );
  }
}

export type Result = {
  code: number;
  message: string;
  state: boolean;
};
```