

Rapport de stage

Stagiaire : Evsen Arsène

Entreprise : Supermarché Istanbul – 1 Rue de la Mettrie à Lorient
56100

Ecole : Lycée LaPaix à Ploemeur – BTS SIO 2 SLAM

Date : 3 Janvier – 10 Février 2023

Objectif : Développer un logiciel de gestion de stock

Sommaire :

- 1 – Présentation de l'entreprise
- 2 – Audit de la présence en ligne de l'organisation
- 3- Missions réalisé en stage
- 4 – Présentation de l'existant : architecture logicielle et matérielle
- 5 – Problématiques auxquelles fait face l'entreprise
- 6 – Objectifs de la solution logicielle à développer
- 7 – Mon environnement de travail
- 8 – Outils et technologies utilisés
- 9 – Réalisation : la version 1 de la solution
- 10 – Réalisation : la version 2 de la solution
- 11 – Mise à disposition du service
- 12 – Sources utilisés
- 13 – Conclusion sur la finalité du stage
- 14 – Annexes

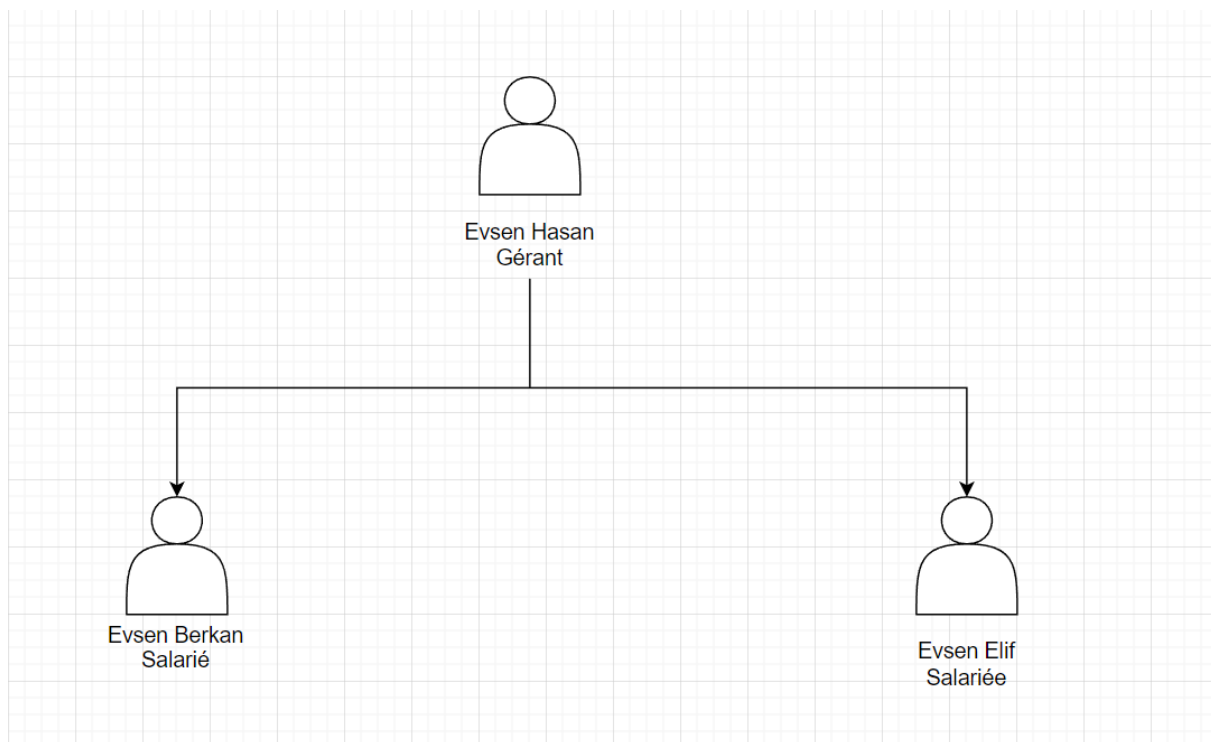
1- Présentation de l'entreprise



L'entreprise « Supermarché Istanbul » est un supermarché familial de 3 personnes, créé le 27 Décembre 2005. Au cours de ses 17 ans d'activités, l'entreprise s'est implémenté à deux endroits : d'abord à Cours Chazelles à Lorient, puis a déménagé pour un local plus grand à 1 Rue de la Mettrie, là où il gère son activité aujourd'hui.

Le supermarché est d'abord spécialisé dans les produits exotiques et notamment turque, par rapport aux demandes de sa clientèle. Cependant, au cours des dernières années, face à une demande de plus en plus varié, les produits vendus sont d'origines multiples.

L'entreprise est une Société à Responsabilité Limitée détenue par M. Hasan EVSEN, qui est aussi le gérant. Les deux autres salariés sont : Evsen Elif et Evsen Berkan.



2 – Audit de la présence en ligne de l'organisation

L'entreprise a une présence en ligne qui se limite à des référencement Google et des profils sur les réseaux sociaux Facebook et Instagram.

L'organisation n'a pas de site web pour l'instant, la création d'un site web est prévu lors du prochain déménagement prévu de l'entreprise pour un nouveau local commercial, cependant la date reste inconnu.



Supermarché Istanbul Lorient

125 J'aime • 136 followers

Publications

À propos


Photos

Vidéos


...

Intro

 Page · Épicerie

 1 rue de la Mettrie

 02 97 64 53 82


★ Pas encore évalué (0 avis) 

Photos

[Toutes les photos](#)



Supermarché Istanbul Lorient

19 février 2022 · 

...



 9


1 

 J'aime

 Commenter



Supermarché Istanbul Lorient

1 février 2022 · 

...



Profil Facebook du Supermarché Istanbul Lorient

3- Missions réalisé en stage

Au cours de ce stage, j'ai dû analyser le contexte et le fonctionnement de l'entreprise, écouter l'expérience et les tâches qu'effectuaient les employés et le gérant, pour ainsi trouver une solution informatique qui pourrait répondre aux besoins évoqués.

Ainsi, après avoir effectué cette analyse et entamé des discussions profonde à propos du métier de l'entreprise, nous avons conclu avec cette dernière, que l'entreprise avait besoin d'un logiciel de stock pour gérer le cycle de vie des stocks de l'entreprise et aussi de l'optimiser. En effet, le stock de l'entreprise et son optimisation était notamment une tâche manuelle répétitive car elle était effectuée manuellement par le gérant.

Cela provoquait une perte de temps et les erreurs qui en découlaient (choix d'achat, stock vide) amenait l'entreprise à perdre de l'argent. Le stock et son optimisation sont donc des enjeux financiers pour une entreprise telle que le supermarché istanbul.

La première partie de mon stage a donc été celle de développer un logiciel de stock qui avait pour objectif de respecter toute les **exigences fonctionnelles** évoqués avec l'entreprise. Cela m'a amené à développer la version 1 de la solution, une application tout-en-un en Java, et communiquant avec une base de donnée MySql.

Cependant, après avoir développé cette première version, il se trouve que malgré le fait qu'elle respectait les exigences fonctionnelles définies, elle avait du mal à respecter toute les **exigences techniques** (contraintes de l'entreprise et exigences de base d'une application professionnelle).

Ainsi, la solution a été repensé et son architecture aussi. Pour respecter les exigences fonctionnelles j'ai appris à utiliser des frameworks qui permettaient de respecter les principes SOLID (principes permettant de développer une « bonne » application). Cela m'a amené à développer un webservice et un site web permettant son utilisation.

4 – Présentation de l'existant : architecture logicielle et matérielle

L'entreprise n'utilise peu ou pas l'informatique dans son activité, hormis pour les besoins de base d'une entreprise.

Ainsi, le « Supermarché Istanbul », a un bureau informatique équipé d'un ordinateur fixe, d'imprimantes, d'un fax et d'une box internet.

L'organisation utilise des logiciels spécialisés, pour notamment consulter des mails et en rédiger (Gmail/OrangeMail), concevoir des factures et devis (EBP Devis et Facturation).

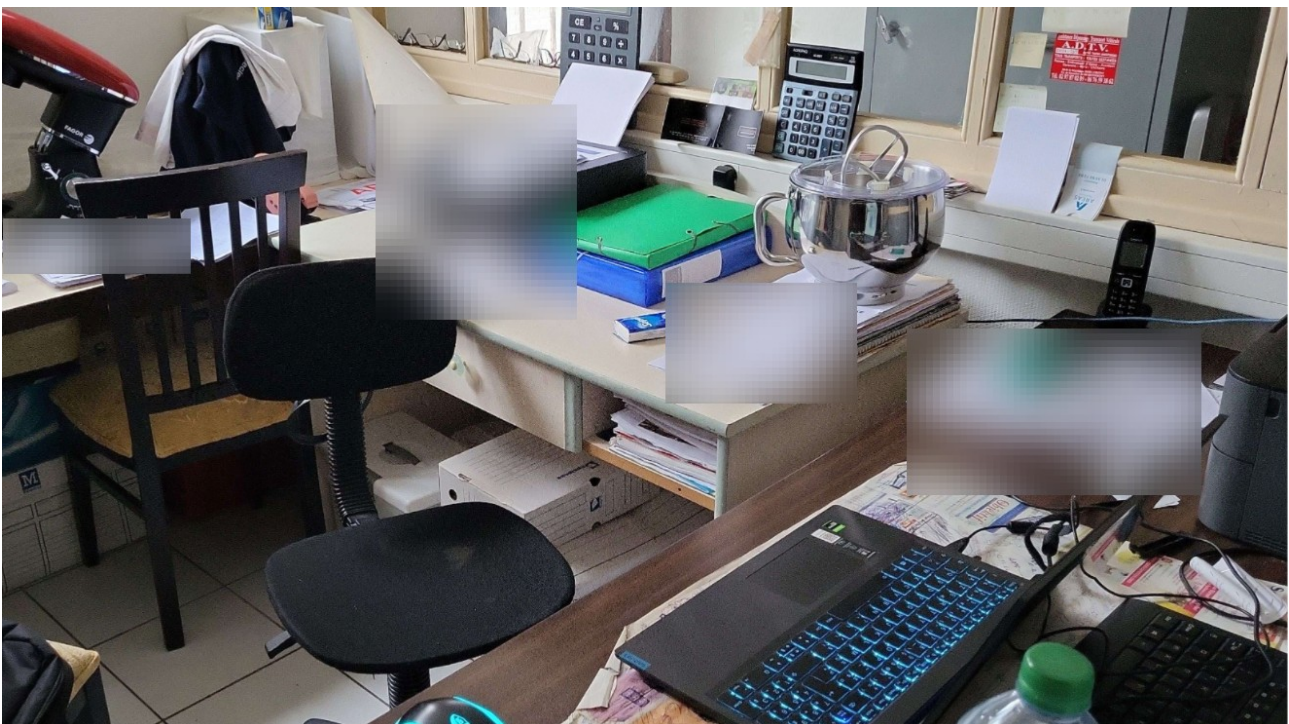


Photo du service informatique

5 – Problématiques auxquelles fait face l’entreprise :

Contexte :

Avec plusieurs dizaines de milliers d’articles, l’organisation fait face à un problème de gestion de son stock. En effet, elle perd du temps : au moment de l’inventaire, de l’argent avec des achats de produits non-optimaux et de l’espace dans sa zone de stockage. A cause d’une non-optimisation des achats de stocks de produit, il arrive que l’organisation achète des produits déjà présent en stock ou qui ne se vendent pas ou pas assez rapidement, ce qui résulte d’une perte d’argent lorsque le produit arrivera à date de péremption ou d’un long retour sur investissement, mais aussi qu’elle oublie ou ne prévoit pas correctement la demande, ce qui crée la perte d’un potentiel chiffre d’affaire. De plus, une mauvaise gestion de stock amène à une situation de saturation de place disponible dans le magasin souffrant déjà d’un manque d’espace. Aujourd’hui pour éviter ces problèmes, l’entreprise effectue régulièrement des inventaires des stocks manuellement ce qui gaspille énormément de temps.

6 – Objectifs de la solution logicielle à développer :

Phase 1. Analyse de besoins et des contraintes :

L’organisation “Supermarché Istanbul” a besoin d’une solution logiciel pour répondre à leur besoin de gestion de stock et d’optimisation d’achat.

Le logiciel doit permettre :

- De gérer les stocks du magasin et l’actualiser en fonction des entrées et sortie d’articles.
- Avoir un historique de vente des produits.
- Avoir un tableau de bord reflétant l’état des stocks, l’historique des ventes pour permettre à l’organisation de prendre de meilleur décision vis-à-vis des achats et des ventes des produits.
- Prévenir l’entreprise lorsqu’un stock d’un article est bas.

Contraintes :

- La solution logicielle doit être la plus accessible possible, elle doit fonctionner sur tout les types de plateformes et systèmes d’exploitations.
- La solution logicielle doit être facile d’utilisation pour être adapté à tout type de profils (jeunes et employés agés).
- La solution logicielle ne doit pas ralentir les processus d’achats et surtout de ventes (au moment du passage en caisse).
- L’application doit être sécurisé.

L'application doit être maintenable et extensible pour de futures améliorations.

7 – Mon environnement de travail :

Phase 2 . Planification :

Budget : l'application nécessitera du matériels informatiques exclusif pour fonctionner (lecteur de code barre, serveur), que l'entreprise financera.

Ressources : exemple de projets de gestions de stocks sur internet, documentations techniques en tout genre (udemy, youtube, openclassroom ect...)

Expérience : Immersion dans le magasin et contact avec les employés et la direction pour cerner les besoins, les contraintes, les attentes et avis en relation avec le personnels et le secteur de grande surface.

Le developpement est effectué dans le bureau de service informatique de l'organisation avec mon ordinateur personnel et les équipements acheté par l'entreprise : un serveur et un scanneur de code-barre.

8 – Outils et technologies utilisé :

Pour ce projet, j'ai utilisé les logiciels suivant :

- Windows
- les IDE : Eclipse, IntelliJ Community Edition et visual studio code
- Postman pour effectuer des requêtes API vers le webservice
- Wamp serveur
- MySql Server et MySql Workbench
- GitHub (site web)

Langage de programmations utilisés :

- Java
- Typescript
- Html
- CSS

Frameworks utilisé :

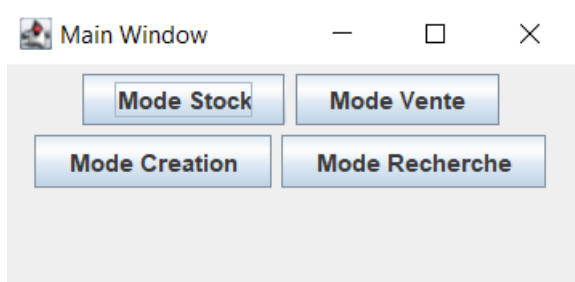
- Spring Framework (Junit, Spring data JPA, Spring MVC,
- Angular

9 – Réalisations : la version 1 de la solution :

La version 1 de la solution est une application en Java pur. La vue, les modèles et services font partie du même logiciel. Celle-ci est installée sur l'ordinateur de l'entreprise et grâce à des interfaces homme-machine, les actions sont effectuées en base de données.

Voici une présentation du logiciel :

Menu principal :



Le menu principal amène l'utilisateur à choisir entre les quatre principaux modes.

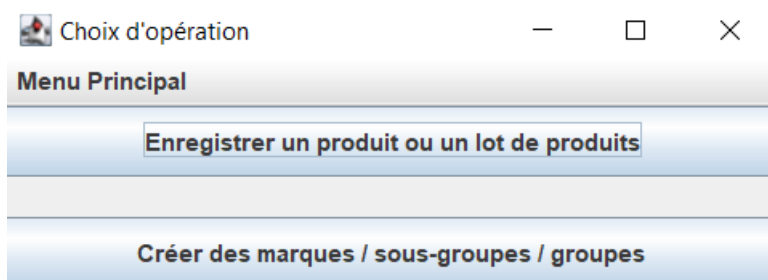
Le mode création sert à enregistrer les articles pour la première fois en base de données. Les articles à enregistrer peuvent avoir des caractéristiques communes (marque, groupe...), ces caractéristiques étant des tables en base de données, ce mode permet de créer des marques, des groupes... afin de les réutiliser plus tard.

Le mode stock permet d'ajouter des articles via leur code-barre mais aussi d'en retirer.

Le mode Recherche permet de retrouver des articles grâce à leur code-barre et ainsi d'en afficher leur contenu.

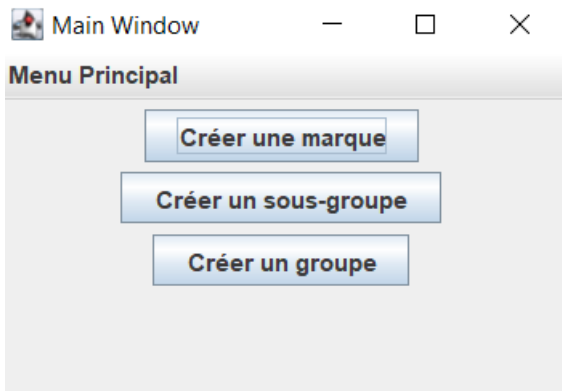
Le mode vente permet de vendre des articles grâce à leur code-barre et ainsi effectuer les traitements requis.

Mode Création :



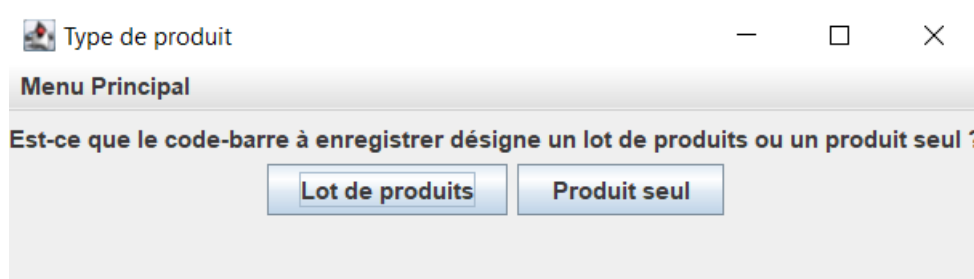
Le mode création demande tout d'abord quel type de création est à effectuer : de nouveaux articles ou des caractéristiques (marques, groupes...).

Mode création → Créer des marques / sous-groupes / groupes :



En base de donnée ces caractéristiques sont des tables, et de nouveau enregistrement équivaut à des clés étrangères utilisable par la création des articles ensuite.

Mode création → Enregistrer un produit ou un lot de produits :



Dans le contexte de l'application, il est important que comprendre la différence entre un article unitaire et un article en lot.

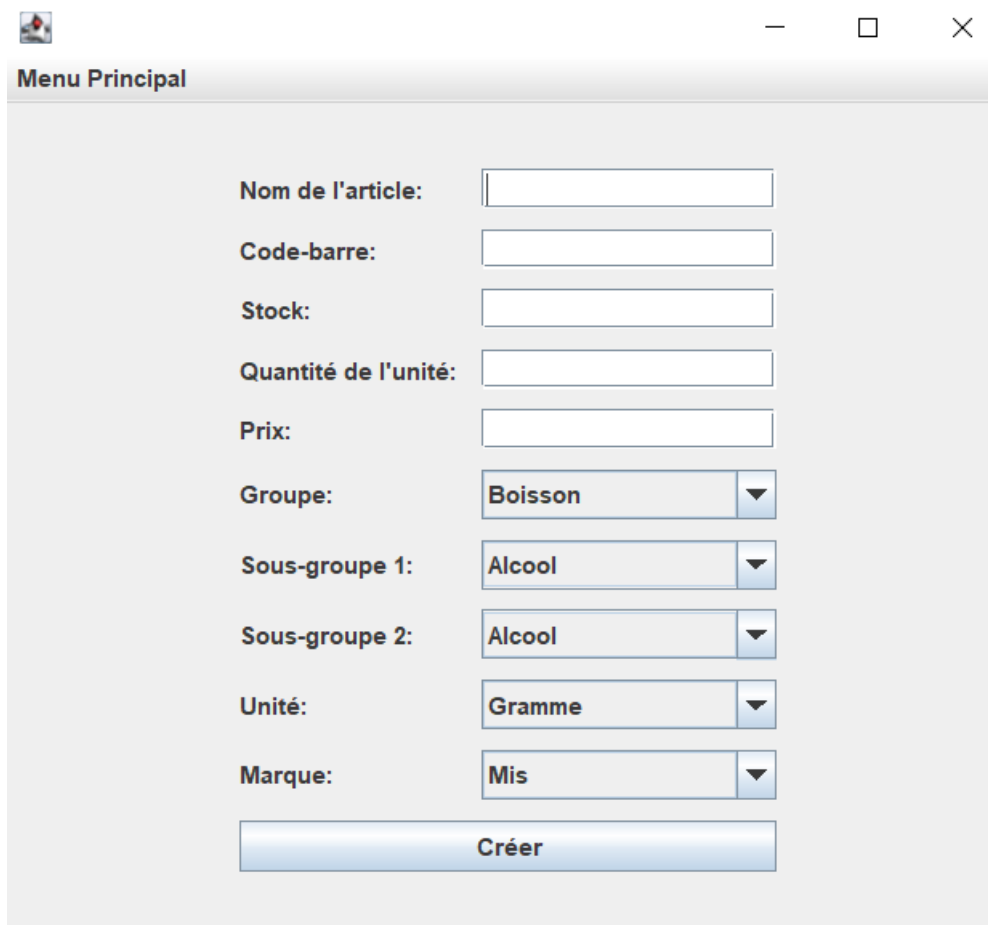
Leurs différenciation en base de donnée est très importante car elle permet d'éviter des bugs de stocks et de prix, mais aussi gagner du temps lors de rentré/sortie de stock.

Tout les articles, peut importe leur type, ont des code-barres différents. Par exemple, lors de l'arrivé de nouveau stock, des palettes peuvent être remplis de carton, qui à leur tour sont remplis d'articles.

Ici le carton à son propre code-barre, qui est différent du code-barre des articles à l'intérieur du carton. Ainsi, l'utilisateur n'a pas besoin de scanner les articles à l'intérieur du carton un à un pour les mettre en stock. L'utilisateur a simplement besoin de scanner le code-barre du carton.

Ainsi, les lots d'articles sont caractérisé par une quantité, un prix de groupe et un référence vers l'article unitaire. En terme de base de donnée, cette référence est effectué grâce au code-barre d'un article qui sert de clé étrangère pour la table « ArticleLot ».

Mode création → Produit seul :



The screenshot shows a window titled "Menu Principal" with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains a form for creating a product. The form has the following fields and controls:

- Nom de l'article:
- Code-barre:
- Stock:
- Quantité de l'unité:
- Prix:
- Groupe:
- Sous-groupe 1:
- Sous-groupe 2:
- Unité:
- Marque:
-

Cette interface permet de créer en base de donnée un article unitaire. Les listes déroulantes sont automatiquement actualisé en fonction de leur table respective.

Par exemple, si dans le mode création, une nouvelle marque est créé, alors celle-ci sera enregistrer dans la table « Marque », cette nouvelle marque apparaître automatiquement dans la liste déroulante et pourra donc être choisie pour un article. Cela permet d'éviter des erreurs d'intégrités.

Mode création → Lot de produits :

The screenshot shows a window titled 'Menu Principal' with a standard Windows title bar (minimize, maximize, close buttons). Inside the window, there are four input fields with labels to their left: 'Code-barre du lot:', 'Quantité du lot:', 'Code-barre de l'article de base:', and 'Prix du lot:'. Below these fields is a large blue button with the text 'Créer'.

Cette interface permet d'effectuer un enregistrement d'un article lot en base de donnée. Le champ « Code-barre de l'article de base » est fait pour y renseigner le code-barre d'un article unitaire.

Mode recherche :

The screenshot shows a window titled 'Recherche d'article' with a standard Windows title bar. Inside, there is a sub-header 'Menu Principal'. Below it, the label 'Recherche d'article :' is followed by a text input field. A blue button labeled 'Rechercher' is positioned below the input field.

Cette interface attend comme saisie le code-barre d'un article.

Après qu'un code-barre est été saisie, il effectue plusieurs vérification :

- Est ce que l'article est un article de lot ?
- Est ce que l'article est un article unitaire ?
- Sinon, il n'existe pas → prévenir l'utilisateur avec une alerte.

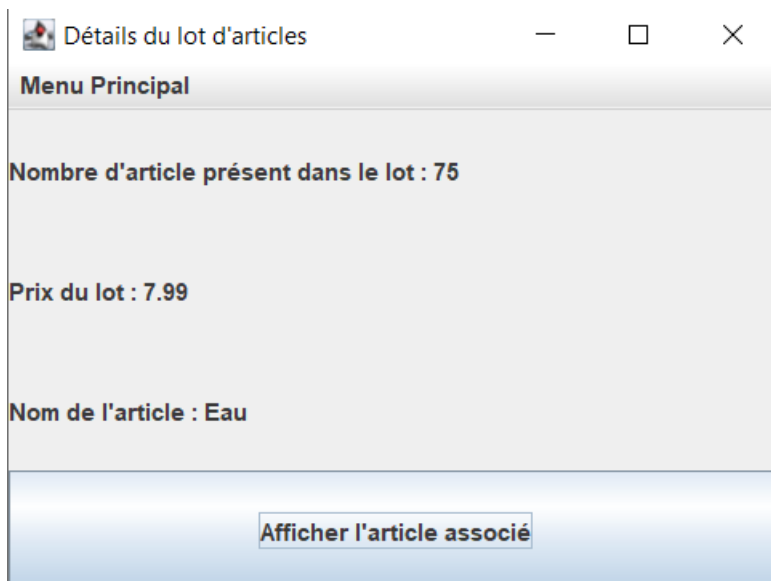
Mode recherche → cas d'un article unitaire :



Si le code-barre est celui d'un article, alors le logiciel va créer un objet « Article » à partir des résultats de la requête Sql.

Cette interface permet d'afficher le contenu d'un article donnée.

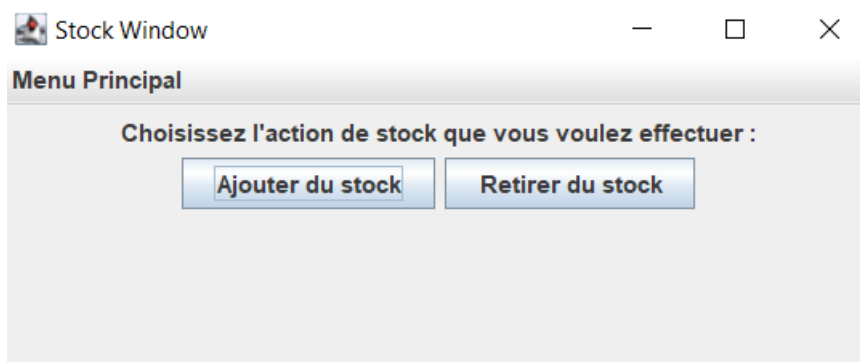
Mode recherche → cas d'un article en lot :



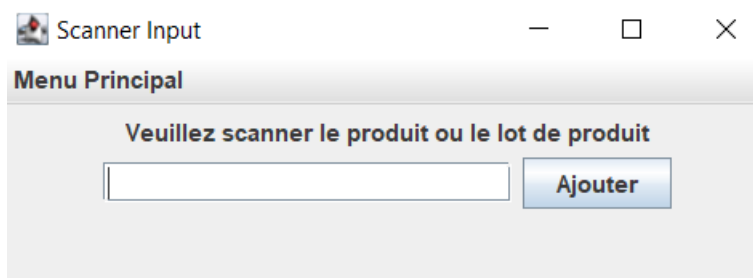
Tout comme l'interface précédente, celle-ci affiche le contenu d'un objet « ArticleLot ».

De plus, elle permet d'afficher l'article associé, en appelant l'interface précédente.

Mode stock :



Mode stock → Ajouter du stock ou en retirer :



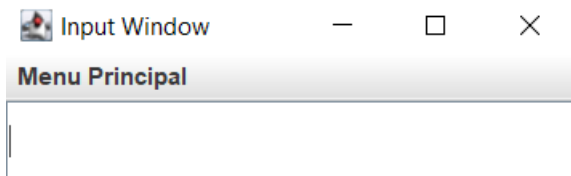
Que ce soit pour l'interface « AjouterStock » ou « Retirer Stock », les deux attendent un code-barre qu'elles vont pouvoir traiter en ajoutant ou soustrayant le stock.

Deux cas sont possible : le code-barre concerne un article unitaire ou un article en lot.

Dans le premier cas, le stock de l'article unitaire sera soustrait ou additionner de 1.

Dans le second cas, le programme va aller rechercher l'article unitaire de base référé à l'article groupe, il va ensuite prendre la « quantité » associé à l'article en lot, et va additionner ou soustraire cette somme au stock de l'article unitaire de base.

Mode vente :



L'interface du mode vente est très simple, mais cela est dû au faite que c'est l'interface **qui présente le plus de contrainte à respecter**.

En effet, dans le contexte d'une vente, un employé est amené à scanner les articles, son rôle est de ne pas perdre de temps. Ainsi il ne doit pas se soucier de l'interface et doit seulement se concentrer sur son métier et le scannage.

L'interface doit donc être capable de récupérer le code-barre dès qu'elle est renseigné, et d'y effectué les traitements nécessaires, sans action de la part de l'utilisateur.

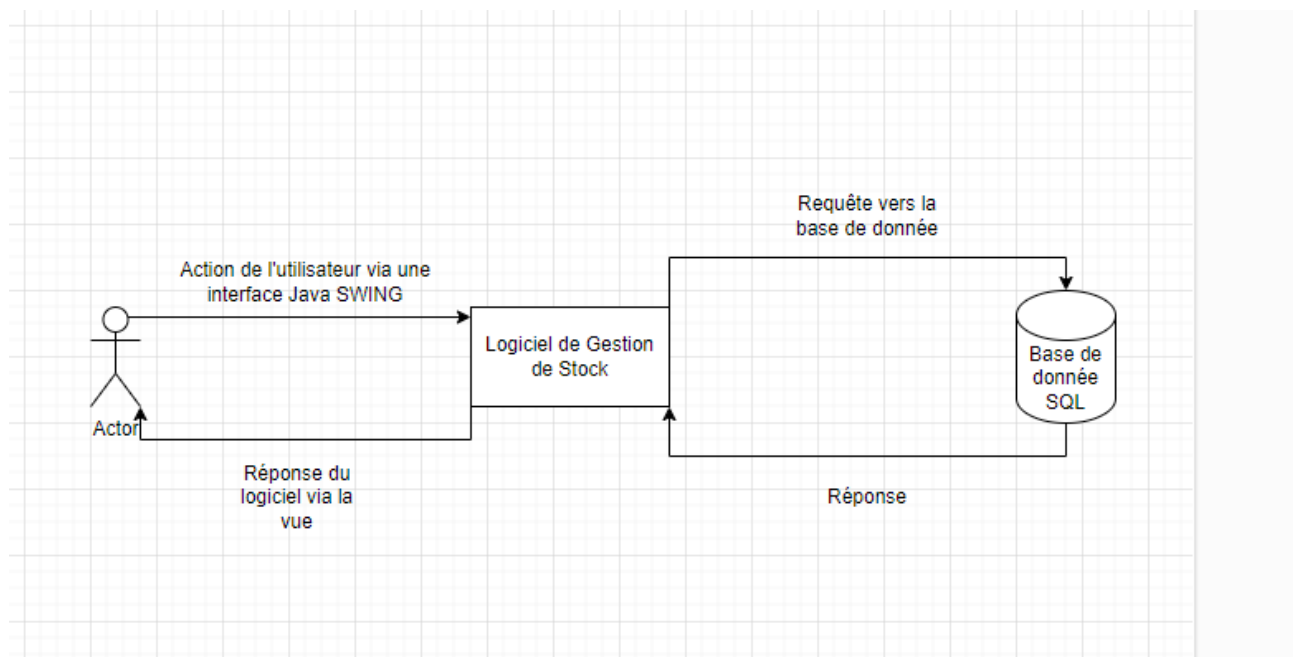
Pour répondre à ces contraintes voici les fonctionnalités qu'utilise l'interface :

- le mode « autoFocus » permet d'éviter que l'utilisateur ait besoin de cliquer

sur le champs de saisie pour y renseigner un code-barre. Dès l'ouverture du mode, le focus est automatiquement fait et le champ est préparé à recevoir n'importe quelle entrée du scanneur.

- pour détecter si l'utilisateur a fini de renseigner le code-barre et effectuer le traitement de vente, l'interface déclenche un compteur au moment où une saisie est commencé. Si l'interface ne reçoit plus d'autre saisie au bout de 1,5 secondes, alors cela veut dire que le code-barre est renseigné. Le champs de saisie est ensuite automatiquement effacé pour laisser la place à une autre.

Architecture applicative de la solution :



Entités de l'application et de la base de donnée :

Dans le package modèle va se trouver toutes les classes responsables de **l'état des données**. Voici les entités qui seront utilisé par le logiciel de gestion de stock :

- la classe Article
- la classe ArticleLot
- la classe Categorie
- la classe Groupe
- la classe Sousgroupe
- la classe Marque
- la classe Unite
- la classe Vente

Voici les liens entre les entités ci-dessus :

Il existe deux type d'article dans le magasin : les articles unitaires et les articles en lot. Chaque article du magasin est identifié par son code-barre qui est son identifiant, dans le contexte d'une base de donnée, ce sera donc sa clé primaire.

Un article en lot est précisément, plusieurs articles unitaires, vendu ensemble. Un article en lot est donc lié à des articles unitaires, dans le contexte de la base de donnée, la table ArticleLot est relié à la table Article grâce à sa clé primaire. Le « codebarre » d'un article unitaire devient la clé étrangère « ref code-barre »de la table ArticleLot.

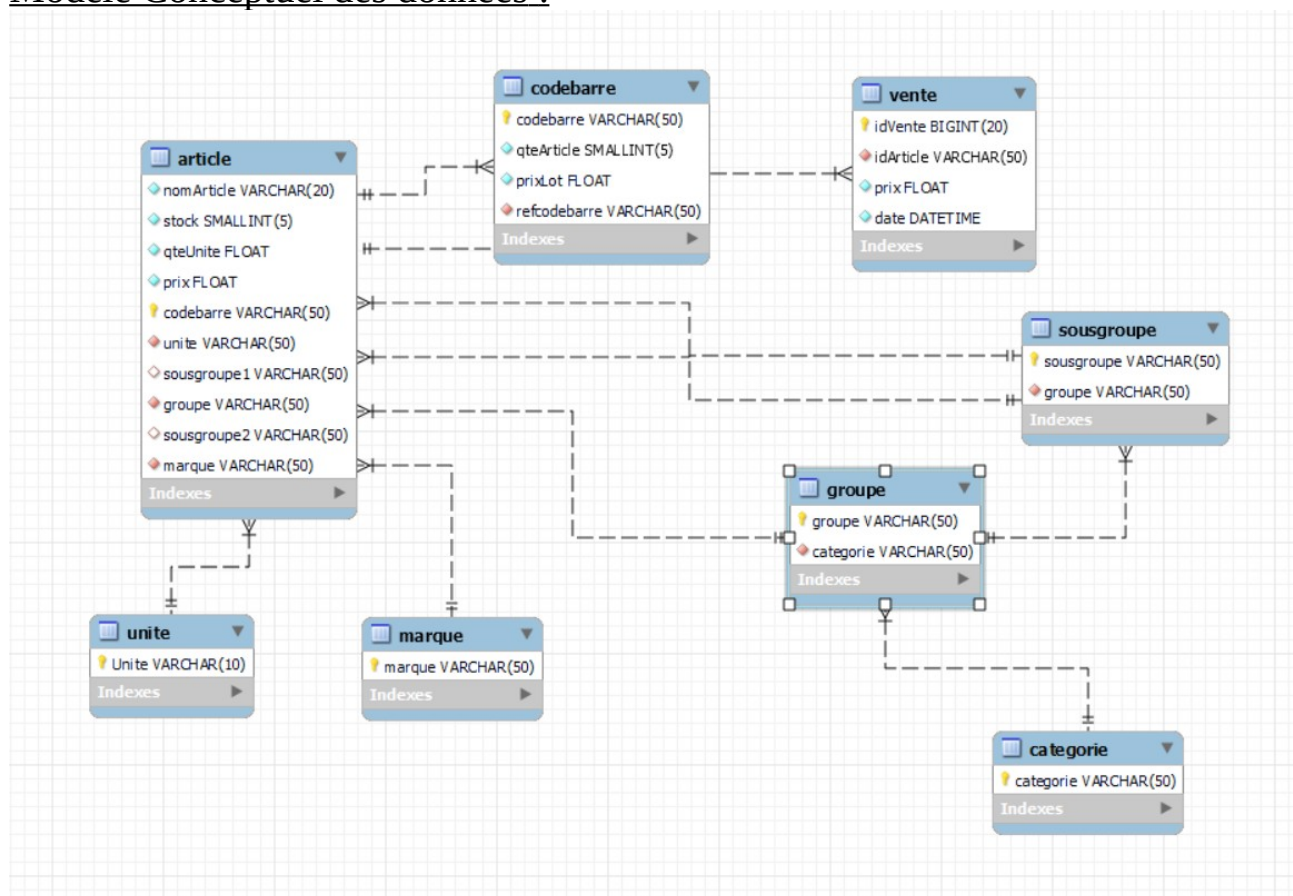
Tout les articles ont des caractéristiques propres : nom, code-barre, prix, quantité d'unité de mesure, son stock.

De plus, ils ont des caractéristiques qui peuvent être communes : unité de mesure, catégorie, groupe/sous-groupe et la marque.

Dans le contexte de la base de donnée, les caractéristiques communes seront des tables à part qui seront relié à la table Article par leur clé primaire.

Enfin, pour la création d'un historique de vente des articles, la table Vente stockera les détails d'une transaction avec l'heure, la date ainsi que le prix et l'identifiant de l'article vendu.

Modèle Conceptuel des données :



Conclusion sur la première version :

Cette première version respecte la plupart des exigences fonctionnelles établies. Cependant, elle présente beaucoup d'inconvénients.

Tout d'abord, elle mélange les vues, les modèles et services. C'est un problème d'abord de sécurité, car les services responsables de l'accès à la base de donnée sont accessible à n'importe quel employés.

Ensuite, pour qu'elle puisse être utilisé, elle doit être déployé sur tout les périphériques des utilisateurs. Les employés utilisant tous des périphériques et des systèmes d'exploitations différents (ordinateur Windows, Iphone, Android), la création de d'une version adapté à chacun devient inadapté pour une entreprise.

Pour finir, elle présente un gros problème d'évolutivité car pour la modifier il faut, redéployer l'application, la recompiler et modifier le code source. La modification du code source peut amener des problèmes de régressions. La réutilisabilité du code est aussi difficile, car elle a été codé en n'utilisant peu de design pattern.

Un des principaux apprentissage de ce stage a donc été de voir que les exigences techniques sont tout aussi, voir plus important que le côté fonctionnel. Pour créer des applications respectants les exigences de base, je vais utiliser des frameworks, qui vont s'occuper de la partie technique pour que le developpeur puisse s'occuper de la partie fonctionnelle.

10 – Réalisation : la version 2 de la solution

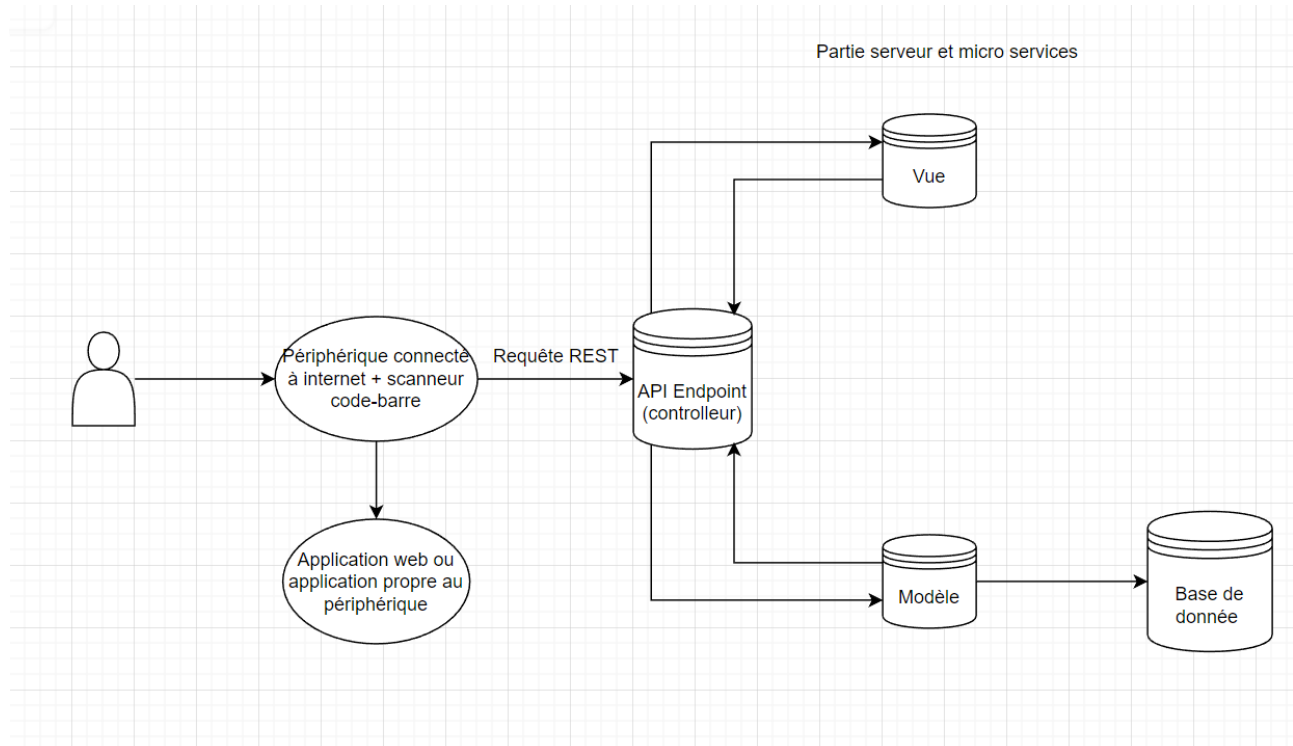
La seconde version de l'application est un webservice et un site web pouvant y accéder.

Dans un contexte d'accessibilité, un web service est plus adapté et ce, pour plusieurs raisons :

- le webservice a une API qui agit en fonction des fichiers JSON reçus.
- un site web est accessible peut importe le périphérique et son système d'exploitation, résout le problème.

- l'application n'a pas besoin d'être déployé partout.
- la solution est séparé entre la vue d'un côté et les contrôleurs et la logique métier de l'autre, et enfin la base de donnée.

Architecture applicative de la solution :



Les principes SOLID :

Le principe SOLID est un ensemble de principes de conception de logiciels qui ont été développés pour aider les développeurs à créer des systèmes logiciels robustes, flexibles et faciles à maintenir. Le principe SOLID est un acronyme qui représente cinq principes de conception :

- S - Le principe de responsabilité unique (Single Responsibility Principle) : chaque module, classe ou fonction devrait n'avoir qu'une seule responsabilité.
- O - Le principe ouvert/fermé (Open/Closed Principle) : les entités logicielles (classes, modules, etc.) devraient être ouvertes pour l'extension, mais fermées pour la modification.
- L - Le principe de substitution de Liskov (Liskov Substitution Principle) : les objets d'une classe dérivée devraient être capables de remplacer les objets de la classe de base sans que cela ne provoque d'erreurs ou d'effets indésirables.
- I - Le principe d'interface de ségrégation (Interface Segregation Principle) : les

clients ne devraient pas être forcés de dépendre des interfaces qu'ils n'utilisent pas.

- D - Le principe d'inversion de dépendance (Dependency Inversion Principle) : les modules de haut niveau ne devraient pas dépendre des modules de bas niveau. Les deux devraient dépendre des abstractions.

En suivant les principes SOLID, les développeurs peuvent concevoir des systèmes logiciels plus flexibles, plus facilement extensibles, plus faciles à maintenir et à tester, et plus résistants aux changements.

Les design patterns :

Un design pattern, ou modèle de conception en français, est une solution générique et réutilisable à un problème commun en conception de logiciel.

Les design patterns sont des approches éprouvées pour résoudre des problèmes de conception de logiciels courants tels que la gestion de l'état, la gestion de la concurrence, la création d'objets, la gestion des événements, etc. Ils sont généralement décrits en termes de classes, d'objets, de relations entre classes et d'interactions entre objets.

Il existe plusieurs types de design patterns, notamment les patterns de création, les patterns de structure et les patterns de comportement. Les patterns de création sont utilisés pour instancier des objets, les patterns de structure pour organiser les objets en structures plus grandes et les patterns de comportement pour gérer les interactions entre objets.

Les design patterns sont utilisés pour créer des logiciels modulaires, extensibles et maintenables, en réduisant la complexité et en améliorant la lisibilité et la compréhensibilité du code. Ils sont largement utilisés dans la conception orientée objet, mais peuvent également être appliqués à d'autres paradigmes de programmation.

Les design patterns peuvent être considérés comme des mises en œuvres spécifiques des principes SOLID.

Architecture logicielle :

L'accès aux services de l'application doivent être accessible par tous et selon le principe de Single Responsibility, certaines parties du code doivent avoir leur propre responsabilité :

Ainsi, l'architecture de l'application est organisée en plusieurs couches :

- La couche **Modèle** qui représente la structure des données
- La couche de(s) **Contrôleur(s)** qui est/sont responsable(s) de recueillir et traiter les requêtes des utilisateurs, grâce à la couche service.
- La couche **Service**, responsable des règles métiers, c'est en somme la logique de l'application.
- La couche **DAO**, responsable de la partie accès aux données de la base de donnée.

Pour répondre aux exigences techniques d'une application accessible rapidement, par tous et avec n'importe quel support, l'architecture propose une API pour assurer des services rapides à tous.

Le principe est que, grâce au **protocole HTTP**, tous les utilisateurs peuvent faire des **requêtes** pour effectuer certaines actions, en s'adressant aux **endpoints** du logiciel.

Les requêtes HTTP seront accompagnés de fichiers texte de type **JSON**, pour transférer de manière automatiquement exploitable les données nécessaires pour effectuer les actions que propose la couche service.

Les Frameworks :

Les frameworks permettent de gérer les exigences techniques, pour que les développeurs puissent se concentrer sur les exigences fonctionnelles.

Dans le contexte de cette application, j'ai utilisé Spring Framework, un framework regroupant d'autres frameworks, et se basant principalement sur Java,

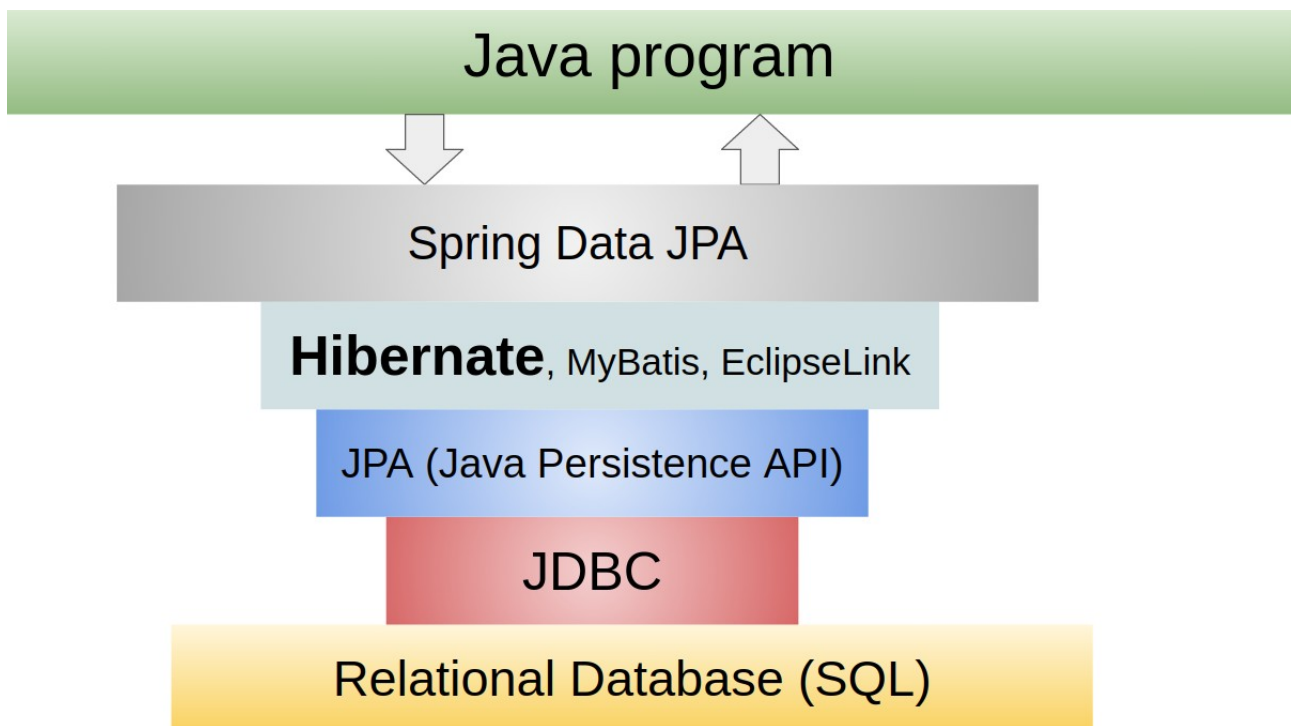
pour developper une API REST.

Cette API aura pour rôle de recevoir les instructions sous forme de fichier JSON et de renvoyer des données avec ce même format.

Le framework Spring MVC aura pour rôle de recevoir les fichier JSON, de les convertir sous forme d'objet, et inversement.

Spring Data JPA est utilisé dans plusieurs points essentiels d'une application web :

- Il embarque un serveur par défaut, « Tomcat » qui permet de rapidement d'exécuter le .Jar de notre application et donc faciliter sa mise en production.
- Dans une architecture MVC, il permet grâce à des annotations, de configurer la partie « Controller » pour en faire des « Endpoints » qui seront accessible via des requêtes HTTP (donc il gère leur réception/envois et la conversion des messages JSON/Objet). Ainsi, il nous permet de créer facilement des APIs et applications RESTful.
- Il gère la partie de communication avec la base de donnée, sans que le developpeur n'ai à écrire une seule ligne de code SQL, en utilisant « JPA », une interface de programmation qui permet d'organiser des données relationnelles.



Le schéma ci-dessus nous montre comment va fonctionner l'accès au donnée d'une base de donnée relationnelles (MySQL) dans notre application.

Tout d'abord, pour qu'une application Java standard, puisse communiquer avec une base de donnée extérieure et relationnelles, il lui faut l'API (interface de programmation d'application) JDBC « Java Database Connectivity ».

JDBC est une interface commune à des sources de données. De ce fait, il peut permettre la communication entre une application et une base de donnée grâce au driver propre de la base de donnée. Dans notre application, puisque la base de donnée est une base de donnée MySQL, JDBC va pouvoir communiquer avec celle-ci grâce au driver « MySQL Connector ».

Cependant, sa configuration et son exploitation amène à coder de façon redondant et nuit à la clareté du code. Deplus, JDBC est dépendant à la base de donnée et son type à cause du driver. Ce qui apporte des problèmes d'évolutivité.

Mais l'inconvénient essentiel vient du faite que, une application orientée objet et une base de donnée relationnelle ne sont pas naturellement compatible car la relation entre les données est différentes. Ainsi, il est nécessaire d'effectuer un mapping entre les données reçu d'une base de donnée et les objets du programmes, et vice-versa.

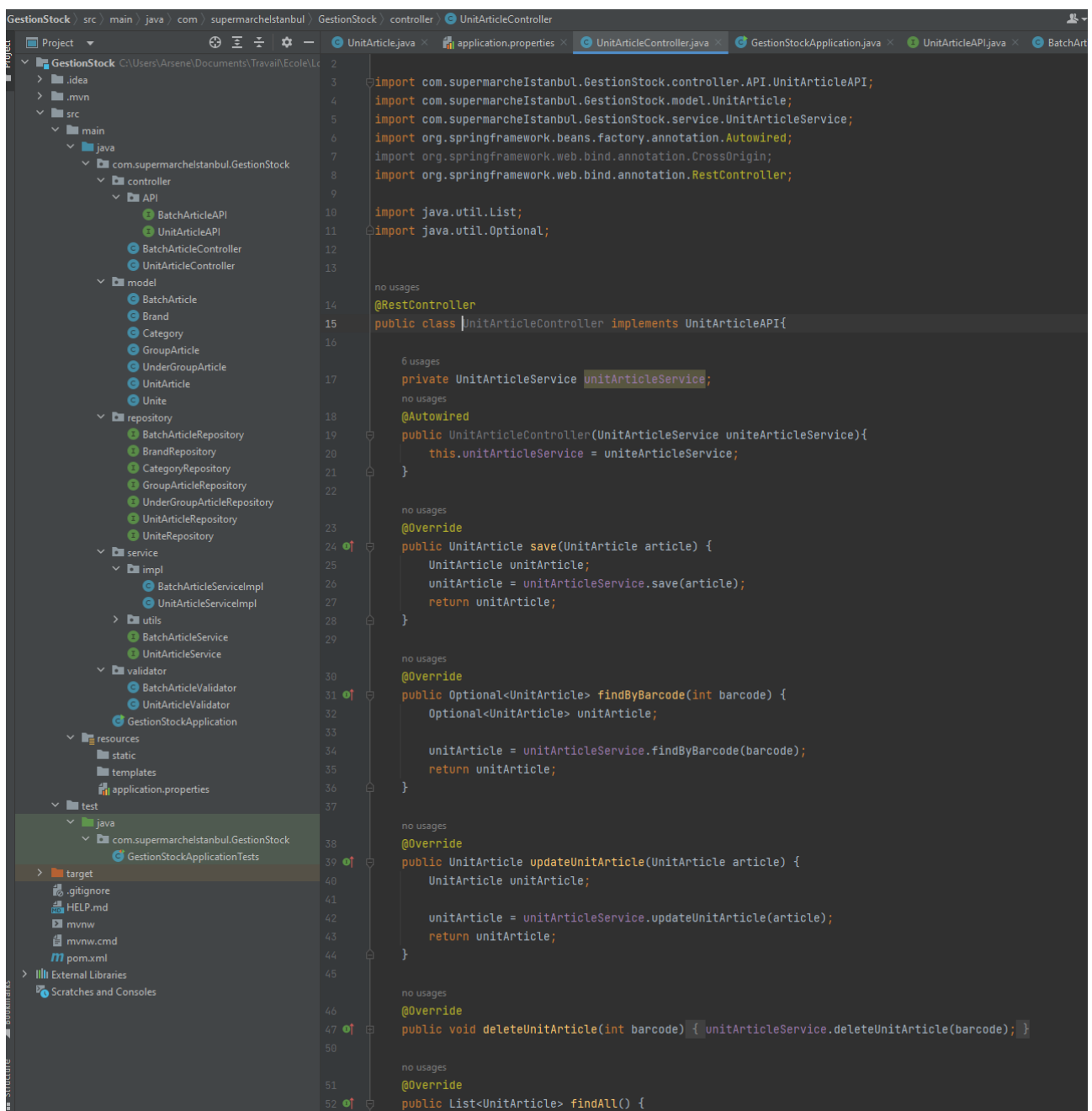
Ce mapping apporte beaucoup de code et, peut devenir très redondant dans les classes.

C'est pour cela que les frameworks ORM « Objet Relationnal Mapping » ont

été créée, ces frameworks vont gérer pour le développeur, la partie mapping. Dans notre application, je vais utiliser l'ORM « Hibernate », mais il en existe d'autre tel que « MyBatis » ou « EclipseLink »...

Historiquement, chaque ORM avait son propre langage, ce qui a amené la création de JPA. Tout comme JDBC est une interface commune aux différents pilotes, JPA l'a aussi pour les framework ORM. De ce fait, aujourd'hui chaque ORM implémente l'interface JPA pour fournir les mêmes méthodes. On n'utilisera donc pas Hibernate directement, mais via les méthodes de l'interface JPA.

Webservice :

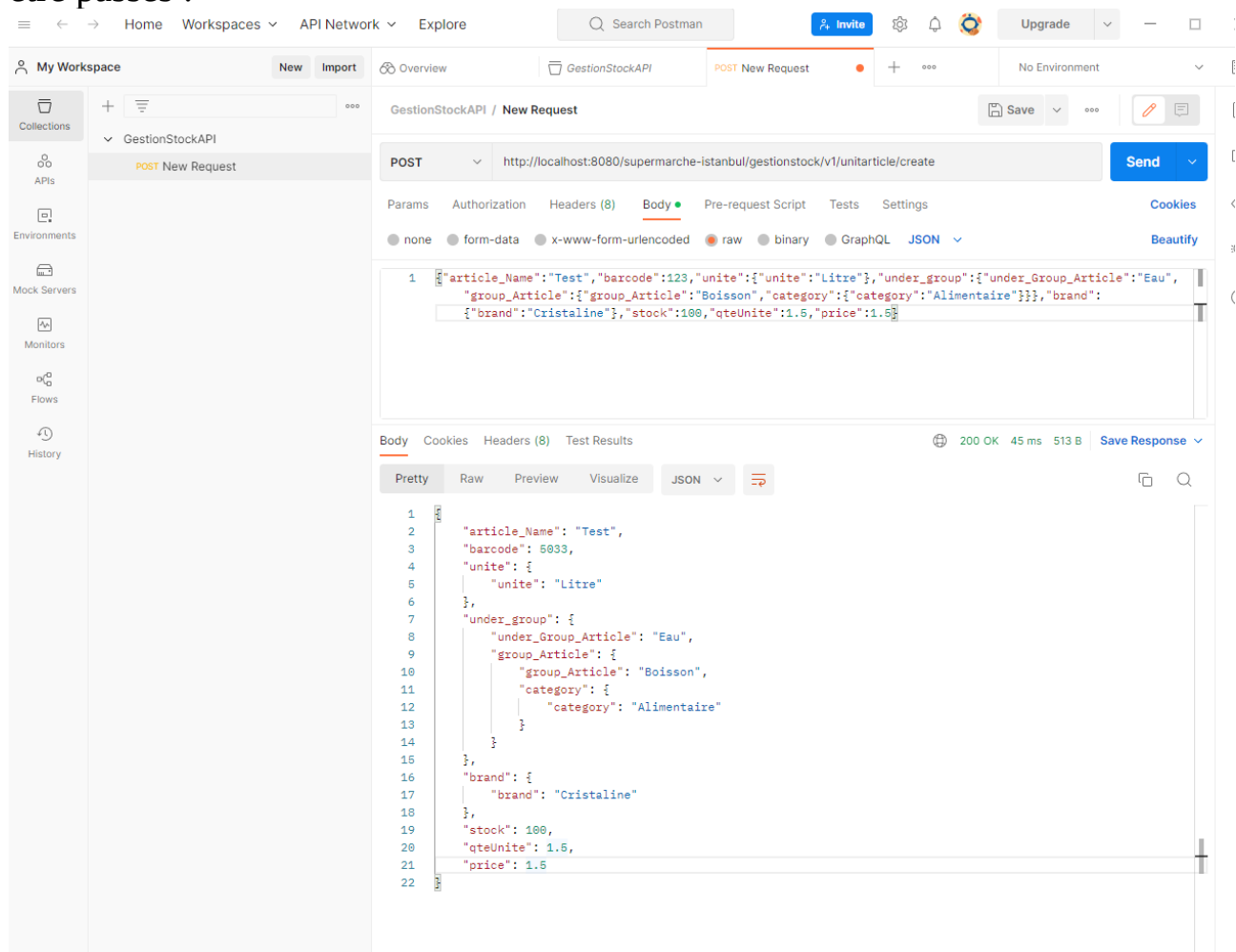


```
2
3 import com.supermarcheIstanbul.GestionStock.controller.API.UnitArticleAPI;
4 import com.supermarcheIstanbul.GestionStock.model.UnitArticle;
5 import com.supermarcheIstanbul.GestionStock.service.UnitArticleService;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.web.bind.annotation.CrossOrigin;
8 import org.springframework.web.bind.annotation.RestController;
9
10 import java.util.List;
11 import java.util.Optional;
12
13
14 @RestController
15 public class UnitArticleController implements UnitArticleAPI{
16
17     6 usages
18     private UnitArticleService unitArticleService;
19     no usages
20     @Autowired
21     public UnitArticleController(UnitArticleService unitArticleService){
22         this.unitArticleService = unitArticleService;
23     }
24
25     no usages
26     @Override
27     public UnitArticle save(UnitArticle article) {
28         UnitArticle unitArticle;
29         unitArticle = unitArticleService.save(article);
30         return unitArticle;
31     }
32
33     no usages
34     @Override
35     public Optional<UnitArticle> findByBarcode(int barcode) {
36         Optional<UnitArticle> unitArticle;
37
38         unitArticle = unitArticleService.findByBarcode(barcode);
39         return unitArticle;
40     }
41
42     no usages
43     @Override
44     public UnitArticle updateUnitArticle(UnitArticle article) {
45         UnitArticle unitArticle;
46
47         unitArticle = unitArticleService.updateUnitArticle(article);
48         return unitArticle;
49     }
50
51     no usages
52     @Override
53     public void deleteUnitArticle(int barcode){ unitArticleService.deleteUnitArticle(barcode); }
54
55     no usages
56     @Override
57     public List<UnitArticle> findAll() {
58         List<UnitArticle> unitArticles;
59         unitArticles = unitArticleService.findAll();
60         return unitArticles;
61     }
62 }
```

Ce webservice a pour rôle de proposer une API et les actions CRUD (Create/Read/Update/Delete) pour les entités : ArticleUnitaire et ArticleStock.

Grâce aux deux contrôleurs « BatchArticleController » et « UnitArticleController », les requêtes d'API sont réceptionnées et traitées grâce aux services dédiés qui vont, à leur tour appeler les « validateurs » d'abord pour effectuer des vérifications. Pour finir, les services appellent les méthodes des « repository » qui implémentent les méthodes relatives aux CRUD et qui s'occupent de gérer la communication avec la base de données.

Grâce au logiciel Postman, les tests des fonctionnalités CRUD de l'API ont pu être passés :



Application web :

L'application web, qui est en cours de developpement, à pour rôle d'être l'interface multi-plateforme et accessible par tous. Le site sera le point d'entrée des utilisateurs pour effectuer des requêtes API vers le webservice. Elle a été developpé grâce au framework « Angular ».

Pour l'instant, elle permet de créer un article, et de liste les articles :

[Article List](#) [Create Article](#)

GestionStockUI

Créer un article

| | | | |
|--------------------|----------------------|----------------------|---------------------------------------|
| Article Name: | <input type="text"/> | Barcode: | <input type="text"/> |
| Unit: | <input type="text"/> | Under Group Article: | <input type="text"/> |
| Group Article: | <input type="text"/> | Category: | <input type="text"/> |
| Brand: | <input type="text"/> | Stock: | <input type="text"/> |
| Quantity per Unit: | <input type="text"/> | Price: | <input type="text"/> |
| | | | <input type="button" value="Submit"/> |

[Article List](#) [Create Article](#)

GestionStockUI

Article List

| Barcode | Price | qteUnite | Stock | article_Name | unite | under_group | brand |
|---------|-------|----------|-------|--------------|-------|-------------|------------|
| 5024 | 5 | 1.5 | 100 | Eau Crista | Litre | Eau | Cristaline |
| 5025 | 5 | 1.5 | 100 | Eau Crista | Litre | Eau | Cristaline |
| 5026 | 5 | 1.5 | 100 | Eau Crista | Litre | Eau | Cristaline |
| 5027 | 5 | 1.5 | 100 | Eau Crista | Litre | Eau | Cristaline |
| 5028 | 5 | 1.5 | 100 | Eau Crista | Litre | Eau | Cristaline |
| 5029 | 5 | 1.5 | 100 | Eau Crista | Litre | Eau | Cristaline |
| 5030 | 5 | 1.5 | 100 | Eau Crista | Litre | Eau | Cristaline |
| 5031 | 5 | 1.5 | 100 | Eau Crista | Litre | Eau | Cristaline |
| 5032 | 5 | 1.5 | 100 | Eau Crista | Litre | Eau | Cristaline |
| 5033 | 1.5 | 1.5 | 100 | Test | Litre | Eau | Cristaline |
| 5034 | 1 | 1 | 1 | Test | Litre | Eau | Cristaline |
| 5035 | 1.2 | 1.5 | 1000 | Test2 | Litre | Eau | Cristaline |

11 – Mise à disposition du service

Un document « Mode d'emploi » est disponible pour l'entreprise, concernant le logiciel fonctionnel V1.

Ce document reprend les interfaces une par une et indique, ce que l'utilisateur peu faire, et comment interpréter les informations des différentes interfaces.

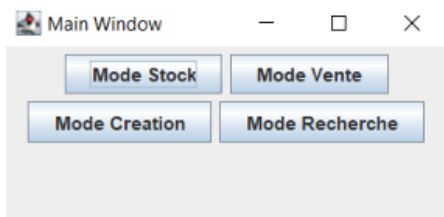
Extrait du mode d'emploi :

Mode d'emploi

L'application se compose en quatre groupes de fonctionnalités :

- le mode Stock
- le mode Creation
- le mode Vente
- le mode Recherche

Menu principal :



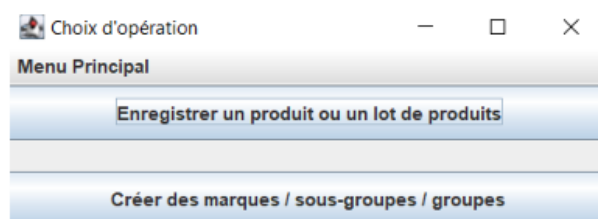
Le mode création sert à enregistrer les articles pour la première fois en base de donnée.

Le mode stock permet d'ajouter des articles via leur code-barre mais aussi d'en retirer.

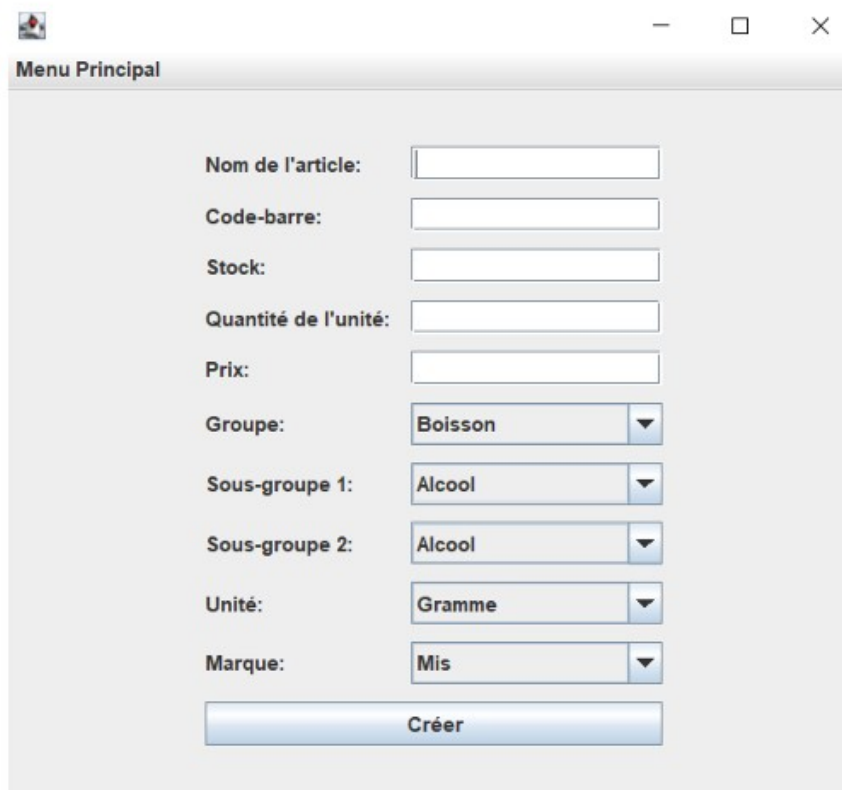
Le mode Recherche permet de retrouver des articles grâce à leur code-barre et ainsi d'en afficher leur contenu.

Le mode vente permet de vendre des articles grâce à leur code-barre et ainsi effectuer les traitements requis.

Mode Création :



Le mode création demande tout d'abord quel type de création est à effectuer : de nouveaux articles ou des caractéristiques (marques, groupes...).



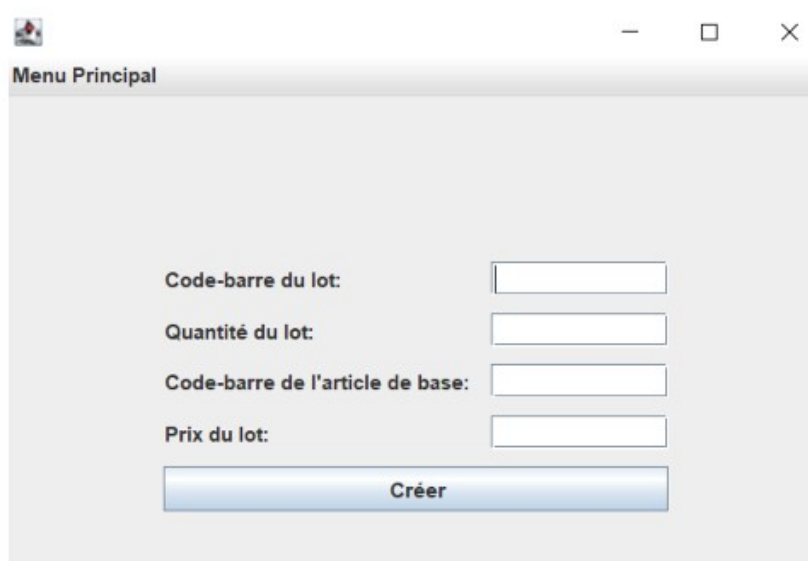
The screenshot shows a window titled "Menu Principal" with a standard Windows title bar (minimize, maximize, close buttons). The window contains a form for creating a unitary article. The form has the following fields and controls:

- Nom de l'article:
- Code-barre:
- Stock:
- Quantité de l'unité:
- Prix:
- Groupe:
- Sous-groupe 1:
- Sous-groupe 2:
- Unité:
- Marque:
-

Cette interface permet de créer en base de donnée un article unitaire. Les listes déroulantes sont automatiquement actualisé en fonction de leur table respective.

Veuillez remplir les caractéristiques de l'article à renseigner.

Mode création → Lot de produits:



The screenshot shows a window titled "Menu Principal" with a standard Windows title bar (minimize, maximize, close buttons). The window contains a form for creating a lot of products. The form has the following fields and controls:

- Code-barre du lot:
- Quantité du lot:
- Code-barre de l'article de base:
- Prix du lot:
-

12 – Sources utilisés :

Youtube :

- Cours théorique du professeur « Mohamed Youssfi » sur les exigences fonctionnelles/techniques, le rôle des frameworks, les designs patterns/ principes SOLID, le fonctionnement d'Angular. →

<https://www.youtube.com/@mohamedYoussfi>

- Exemple d'application de gestion de stock en Java avec « Ali Bouali »

→ <https://www.youtube.com/@BoualiAli>

- Cours JPA de « cours-en-ligne » →

<https://www.youtube.com/@coursenlignejava>

Openclassroom :

- Cours sur les API : <https://openclassrooms.com/fr/courses/6573181-adoptez-les-api-rest-pour-vos-projets-web>

- Spring framework : <https://openclassrooms.com/fr/courses/6900101-creez-une-application-java-avec-spring-boot/7082061-tirez-un-maximum-de-ce-cours>

13 – Conclusion sur la finalité du stage

Pour conclure, ce stage m'a permis d'avoir une expérience professionnelle dans le développement d'application pour des clients, de l'analyse à la conception et à la livraison d'un prototype.

J'ai appris que traduire les demandes des clients en solutions logicielles incluait beaucoup de paramètres à prendre en compte : d'un côté les contraintes du client, de l'autre les contraintes de l'informatique et la programmation en général.

Ainsi, j'ai appris à quel point les exigences techniques étaient importantes pour une solution, et comment utiliser l'expérience des autres grâce aux frameworks.

Les frameworks m'ont permis, contrairement à ce que j'aurais pu croire, de travailler mes fondamentaux en langage orienté objet afin d'en tirer parti, là où avant je ne programmais qu'en utilisant les classes et les associations, je comprend beaucoup mieux l'avantage des autres piliers de ce paradigme, comme le polymorphisme.

Enfin, la découverte des webservices et leurs avantages ont suscité grandement mon intérêt.

Je remercie l'entreprise « Supermarché Istanbul » de m'avoir offert cette expérience professionnalisante et de leur confiance accordée.

Evsen Arsène.

Annexe :

Vocabulaire :

Dans le cadre de ce rapport de stage, plusieurs termes techniques ont été utilisés. Cette annexe a pour objectif de définir quelques-uns de ces termes clés.

API L'acronyme API signifie "Application Programming Interface" ou interface de programmation d'application. Une API permet à des applications de communiquer entre elles de manière standardisée, en fournissant des fonctionnalités et des services à d'autres applications.

Framework Un framework est une plateforme logicielle qui fournit un ensemble de fonctionnalités prêtes à l'emploi pour faciliter le développement d'applications. Il s'agit d'un ensemble de bibliothèques, d'outils, de conventions de codage et de pratiques recommandées qui permettent aux développeurs de créer rapidement et efficacement des applications.

Principes SOLID Les principes SOLID sont un ensemble de principes de conception orientée objet qui visent à faciliter la compréhension, la maintenance et l'extension des logiciels. Les principes SOLID sont :

- **Single Responsibility Principle (SRP)** : chaque classe doit avoir une seule responsabilité.
- **Open/Closed Principle (OCP)** : les classes doivent être ouvertes à l'extension mais fermées à la modification.
- **Liskov Substitution Principle (LSP)** : les objets d'une classe dérivée doivent être utilisables comme des objets de la classe de base.
- **Interface Segregation Principle (ISP)** : les interfaces doivent être spécifiques à un seul usage.
- **Dependency Inversion Principle (DIP)** : les dépendances entre les classes doivent être basées sur des abstractions plutôt que sur des implémentations.

Ressources : tout le code des bases de données et solutions sont disponible sur le drive Google partagé avec l'école et l'académie, ou bien sur mon lien

GitHub : @ArseneEvsen.