

**Année universitaire 2022-2023**  
**Ingénieur Informatique 2e année**

**Rapport de devoir**  
**Programmation Web Avancée**



**Étudiants :**  
Abdoulaye Barry  
Chloé Hamilcaro

**Enseignant :**  
M. Benoit Villa

<b>I. OUTILS ET TECHNOLOGIES</b>	<b>3</b>
A. Généralités	3
B. Architecture globale	3
C. Base de données	3
D. XML/JSON	4
<b>II. CHOIX D'IMPLEMENTATION</b>	<b>4</b>
<b>III. RÉSULTATS</b>	<b>5</b>
A. Accueil	5
B. Gestion des fiches	5
C. Tableau de bord et prise en compte des informations météo	6
<b>IV. PROBLÈMES RENCONTRÉS</b>	<b>7</b>

## I. OUTILS ET TECHNOLOGIES

### A. Généralités

Nous avons effectué ce projet à l'aide de l'environnement de développement Eclipse, sur lequel nous avons utilisé l'ensemble de bibliothèques logicielles **JDK 11**. Nous avons utilisé l'outil **Apache Maven 3.8.7**, ainsi qu'un serveur **Tomcat 10.0.13** pour le développement de cette application web.

### B. Architecture globale

Notre application est développée en MVC. Les vues sont contenues dans des fichiers **JSP**, tandis que les services (contrôleurs) utilisent des **Servlet**.

### C. Base de données

Nous avons utilisé le système de gestion de bases de données **Oracle XE 21c**, ainsi que **Hibernate**, une implémentation de **JPA**, en tant qu'ORM pour créer et accéder aux données de la base. JPA nous a également permis de gérer facilement la persistance de données.

Pour interroger les données, nous avons utilisé des **requêtes simples**, comme par exemple `SELECT a FROM Automate a ORDER BY a.montantVentes DESC` mais également des **requêtes programmées** grâce à **CriteriaBuilder** afin de créer des requêtes plus lisibles et d'éviter les injections SQL.

Par exemple, nous avons déterminé qu'une fiche d'un automate ne serait reliée qu'à un seul automate. Par conséquent, il est inutile de laisser à l'utilisateur la possibilité de choisir, dans la liste d'automates à associer, un automate qui possède déjà une fiche : cela aurait été refusé par la base de données quoiqu'il en soit. Ainsi, nous avons le code suivant dans la classe `AutomateDao` :

```
public List<Automate> findAllHasNotFiche() {
    EntityManager em = factory.createEntityManager();
    List<Automate> exclusion = new ArrayList<Automate>();
    List<FicheAutomate> fiches =
Gestionnaire.getInstance().getFicheAutomateDao().findAll();
    // Liste des automates à exclure
    for (FicheAutomate f : fiches)
        exclusion.add(f.getAutomate());
    // Utilisation d'un CriteriaBuilder
    CriteriaBuilder builder = em.getCriteriaBuilder();
    CriteriaQuery <Automate> query = builder.createQuery(Automate.class);
    Root<Automate> p = query.from(Automate.class);
    query.select(p);
    query.where(builder.not(p.in(exclusion)));
    // Retour du résultat
    return em.createQuery(query).getResultList();
}
```

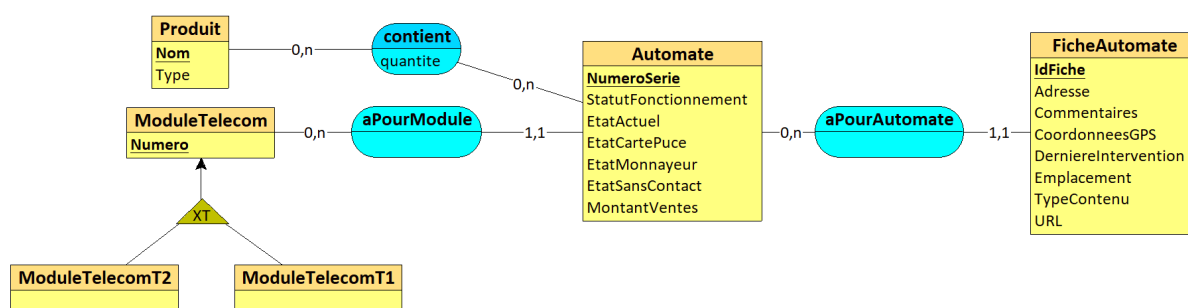
Ce code récupère tout d'abord la liste des automates ayant déjà une fiche : il s'agit de la liste des automates à exclure. Puis, une requête programmée grâce à **CriteriaBuilder** et **CriteriaQuery** récupère les automates qui ne se trouvent pas dans la liste d'exclusion.

## D. XML/JSON

L'accès aux données météo se fait par le biais de l'**API OpenWeatherMap**. Les données de cette api étant accessibles via Internet en **JSON**, nous avons dû extraire ces données grâce à la bibliothèque `org.json` afin de pouvoir les utiliser pour calculer le seuil de réapprovisionnement des produits en fonction de la position GPS des automates.

## II. CHOIX D'IMPLEMENTATION

Nous avons décidé de développer l'application en nous basant sur le modèle suivant :



Ainsi, nous avons déterminé que les coordonnées GPS, l'emplacement et l'adresse d'un automate étaient variables et ne dépendent pas directement de celui-ci : un automate peut être déplacé, ce changement sera signalé par l'utilisateur dans la fiche de l'automate, par conséquent ces attributs sont dans la fiche de l'automate. Par la même logique, le montant des ventes se trouve dans l'automate et non la fiche, car il dépend de l'interaction des consommateurs avec la machine et ne devrait pas être modifié directement par l'utilisateur de l'application web.

Les états (état actuel, état du monnayeur...) ainsi que le statut d'un automate sont des énumérations, que nous n'avons pas jugé utile de persister dans la base de données mais qui disposent néanmoins de classes de type énumération.

Pour simplifier l'architecture, nous avons décidé qu'un même module de télécommunication pouvait être affecté à plusieurs automates, car il s'agit en réalité plus d'une représentation d'un type de module que d'un module en particulier.

### III. RÉSULTATS

#### A. Accueil

La création et le peuplement de la base de données avec des données de test se fait à la première ouverture de l'application web, à la racine du projet (par exemple : `http://localhost:8082/projet-pwa-ac/`). Cette page est gérée par la Servlet `AccueilServlet`.

L'écran ci-contre s'affiche alors, avec la liste des fiches des automates contenues dans la base.

Trois choix s'offrent à l'utilisateur : ajouter une fiche d'automate, afficher le tableau de bord et afficher une fiche, identifiée par le numéro de série de l'automate.

**Gestion des fiches automate**

Numéro de série	Fiche
AUTO_CHAUD_1	<input type="button" value="Afficher"/>
AUTO_CHAUD_2	<input type="button" value="Afficher"/>
AUTOTEMPBASE	<input type="button" value="Afficher"/>

#### B. Gestion des fiches

Lorsqu'un utilisateur désire ajouter une fiche d'automate, il peut simplement cliquer sur le bouton "Ajouter une fiche". Le formulaire s'affiche alors à l'écran (voir la capture d'écran ci-dessous) et l'utilisateur pourra compléter les champs en entrant les informations telles que le type de boisson, la date de la dernière intervention et même ajouter un commentaire s'il le souhaite. L'utilisateur peut uniquement choisir un numéro de série parmi ceux des automates qui n'ont pas encore de fiche (comme expliqué précédemment).

##### Ajout d'un fiche automate

Numéro de série :

Type :

Adresse d'installation :

Emplacement :

Coordonnées GPS :

Date de dernière intervention :

URL :

Commentaires :

Pour afficher une fiche d'automate, l'utilisateur peut cliquer sur le bouton "Afficher" situé à droite de l'automate en question. Cela affichera un écran détaillant les informations sur l'automate (voir la capture ci-dessous), telles que son numéro de série, son type, son adresse d'installation, son emplacement, etc.

## Fiche automate

### FICHE DE L'AUTOMATE

Numéro de série	AUTO_CHAUD_1
Type	BOISSON_CHAUDE
Adresse d'installation	1 Rue de Chloe
Emplacement	rez-de-chaussée
Coordonnées GPS	57.722250 -8.139337
Date de dernière intervention	2023-02-05 00:00:00.0
Commentaires	Fantastique
<input type="button" value="Supprimer la fiche"/>	
<input type="button" value="Modifier la fiche"/>	

L'utilisateur peut alors opter pour modifier ou supprimer la fiche en cliquant sur les boutons "Modifier" et "Supprimer", respectivement. En cas de modification, l'utilisateur sera dirigé vers un écran où il pourra changer le type de boisson, l'adresse ou ajouter un commentaire etc. (voir la capture ci-dessous). Les coordonnées GPS doivent obligatoirement suivre un format spécifique pour pouvoir être enregistrées (un nombre décimal positif ou négatif, un espace, puis un nombre décimal positif ou négatif). Toutefois, il est à noter que le numéro de série de l'automate ne peut pas être modifié : il s'agit en effet de l'élément principal d'une fiche. Une fois les modifications apportées, elles peuvent être validées et enregistrées dans la base en cliquant sur le bouton "Valider".

### Modification d'un fiche automate

Numéro de série :	<input type="text" value="AUTO_CHAUD_1"/>	Type :	<input type="text" value="Encas"/>
Adresse d'installation :	<input type="text" value="1 Rue de Barry"/>		
Emplacement :	<input type="text" value="1ère étage"/>		
Coordonnées GPS :	<input type="text" value="10.722250 -3.139337"/>		
Date de dernière intervention :	<input type="text" value="05/02/2023 14:41"/>		
URL :	<input type="text" value="https://www.fakedomain.cor"/>		
	<input type="text" value="Formidable"/>		
Commentaires :	<input type="text"/>		
	<input type="button" value="Valider"/>		

## C. Tableau de bord et prise en compte des informations météo

L'accès à ce service se fait par exemple par l'url <http://localhost:8082/projet-pwa-ac/dashboard>

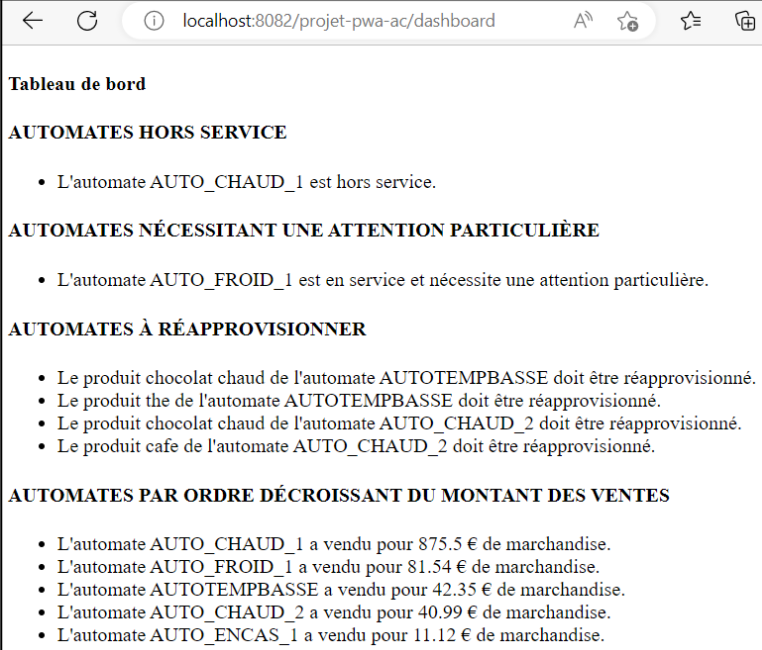
Si l'utilisateur choisit d'afficher le tableau de bord, il tombe sur la liste des automates hors services, ceux nécessitant une attention particulière, ceux à réapprovisionner et sur l'affichage des automates par ordre décroissant du montant des ventes.

La liste des automates à réapprovisionner dépend de la température aux coordonnées GPS de l'automate. Celle-ci est, comme expliqué plus tôt, récupérée d'un fichier JSON de l'api OpenWeatherMap. Or, ces coordonnées sont présentes sur la fiche de l'automate, et tous les automates n'en possèdent pas une. Quand nous rencontrons cette situation, nous avons fait le choix de ne pas déterminer sa température et de ne pas afficher l'automate dans la liste des automates à réapprovisionner, quand bien même celui-ci le nécessite : nous partons du principe que, si l'automate n'a pas de fiche, c'est que son état ne concerne pas l'utilisateur.

Pour tester cette fonctionnalité, nous avons placé l'automate AUTOTEMPBASSE en Russie, à un endroit où les températures sont négatives en cette saison. Cet automate contient des produits en quantité supérieure à 30 unités mais inférieure à 30, ce qui fait qu'il entre dans la catégorie des automates à réapprovisionner.

Nous avons également estimé nécessaire d'indiquer quels sont les produits à réapprovisionner, ils apparaissent donc également sur le tableau de bord.

La liste des automates par ordre décroissant du montant des ventes ainsi que les autres listes font également apparaître les automates qui n'ont pas de fiche.



<b>Tableau de bord</b>	
<b>AUTOMATES HORS SERVICE</b>	
<ul style="list-style-type: none"><li>• L'automate AUTO_CHAUD_1 est hors service.</li></ul>	
<b>AUTOMATES NÉCESSITANT UNE ATTENTION PARTICULIÈRE</b>	
<ul style="list-style-type: none"><li>• L'automate AUTO_FROID_1 est en service et nécessite une attention particulière.</li></ul>	
<b>AUTOMATES À RÉAPPROVISIONNER</b>	
<ul style="list-style-type: none"><li>• Le produit chocolat chaud de l'automate AUTOTEMPBASSE doit être réapprovisionné.</li><li>• Le produit the de l'automate AUTOTEMPBASSE doit être réapprovisionné.</li><li>• Le produit chocolat chaud de l'automate AUTO_CHAUD_2 doit être réapprovisionné.</li><li>• Le produit cafe de l'automate AUTO_CHAUD_2 doit être réapprovisionné.</li></ul>	
<b>AUTOMATES PAR ORDRE DÉCROISSANT DU MONTANT DES VENTES</b>	
<ul style="list-style-type: none"><li>• L'automate AUTO_CHAUD_1 a vendu pour 875.5 € de marchandise.</li><li>• L'automate AUTO_FROID_1 a vendu pour 81.54 € de marchandise.</li><li>• L'automate AUTOTEMPBASSE a vendu pour 42.35 € de marchandise.</li><li>• L'automate AUTO_CHAUD_2 a vendu pour 40.99 € de marchandise.</li><li>• L'automate AUTO_ENCAS_1 a vendu pour 11.12 € de marchandise.</li></ul>	

## IV. PROBLÈMES RENCONTRÉS

Le sujet demandait d'utiliser un webservice pour l'accès aux données, malheureusement nous avons rencontré des problèmes de conflits entre les dépendances que nous n'avons pas su résoudre. Pour éviter de perdre trop de temps dans le développement des fonctionnalités attendues, nous n'avons pas utilisé de webservice et nous nous sommes contentés d'utiliser des Servlet.

Nous avons également passé beaucoup de temps à réfléchir aux données et aux classes qui allaient les représenter : au fur et à mesure que nous avançons dans le projet, nous avons dû revoir notre architecture de classes, ce qui a rallongé notre temps de développement. Cependant, même si cela nous a pris du temps, nous avons réussi à aboutir à une architecture qui nous convient et convient aux choix d'implémentation que nous avons faits.

En ce qui concerne l'envoi de rapports XML/JSON, nous avons conçu des classes (ModuleTelecomT1, ModuleTelecomT2 et une Servlet) pour envoyer des rapports sous forme textuelle à une URL configurable. Le module de rapport envoie un message à l'URL spécifiée et reçoit un code HTTP 200 pour confirmer que le message a été correctement reçu. Dans le cas contraire, le module réessayera l'envoi 5 minutes plus tard jusqu'à ce qu'un code HTTP 200 soit reçu.

Cependant, nous avons rencontré des difficultés lors de la réception et du stockage des données de rapport dans une base de données afin de permettre aux équipes de gestion et de maintenance de les visualiser via une application Web. Nous avons essayé d'utiliser le modèle JPA (Java Persistence API) pour stocker les données dans une base de données, mais on n'a pas réussi à trouver une solution pour intégrer les données reçues dans la base de données ni même tester leur envoi grâce à RESTClient.