Rapport du projet de Web Avancée :

Gestion d'un parc d'automate

Etablissement : Ecole d'ingénieurs Sup Galilée
Spécialité : Informatique
Année scolaire : Troisième année - équivalent M2
Lieu : Université Paris 13 - 93430 Villetaneuse

Equipe : Florent SERFAS et Ludovik TEKAM

Année universitaire : 2019 - 2020

Table des matières

1 Présentation du projet **2**

2 Outils utilisés **2**

3 Implémentation **3**
3.1 Modélisation avant implémentation 3
3.2 Implémentation du web service 6
3.3 Implémentation de l'application web 7

4 Difficultés rencontrées **9**

5 Améliorations possibles **9**

6 Réponse aux questions bonus **10**
6.1 Compte-rendu des techniciens 10
6.2 Solutions techniques pour le directeur marketing 10

1 Présentation du projet

Ce projet consiste à créer une application Web permettant de gérer un parc d'automates de ventes de boissons et de confiseries. Nous avions à modéliser les échanges entre les automates et l'application web et aussi à modéliser la structure contenant les informations de ces automates.

2 Outils utilisés

L'intégralité du développement s'est effectué sous **Eclipse IDE for Java EE Developers**. Nous avons choisi d'utiliser le framework **Spring Boot** afin de faciliter le développement de nos applications (web service et web app) et leurs mises en production éventuelle via des packages *jar* totalement autonome.

Pour des raison de commodité nous avons opté pour **MySQL** comme système de gestion de bases de données.

Afin de simuler l'envoie des rapports par les automates de vente nous avons utilisé **REST-Client** (modules complémentaires pour Firefox).

3 Implémentation

Après analyse, et modélisation des différentes problématiques, nous avons, dans un premier temps, développé un web service qui fournit un ensemble de services liés à la gestion de notre parc d'automate. Puis, nous avons développé une application web qui consomme notre web service et le web service qui fournit les informations météorologiques.

3.1 Modélisation avant implémentation

Avant de développer les différentes applications il nous a fallut définir les structures *XML* et *JSON* provenant des différents automates.

JSON des rapports journalier

```
{"serial_number": 1,
"report_date":"2019-11-23",
"status": "en service",

"automate_information":{
 "state": "ok",
 "temperature": 20,

 "payment_state":{
        "coins": "normal",
        "smart_card": "normal",
        "card": "normal"
 }
},
"errors":[
 {"type": "ax23",
 "description": "azerty"}
],

"articles":[
 {"name": "cafe",
 "quantity": 20,
 "type":"chaud"}
],
"income": 300}
```

JSON des fiches d'information

```
{
"type": "Cafe",

"address_installation":{
 "street_number": 20,
 "street_description": "Rue victor
                       hugo",
 "city":"Paris",
 "state": "France"
},

"emplacement": "pres de la
                sortie du metro 4",

"coordinated":{
 "longitude": 20,
 "latitude": 40},

"intervention_date":"2019-11-23",

"note": "Bonjour Je suis  l'automate
        xx et je fais du Cafe"
}
```

XML des rapports journaliers

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<daily_report>

  <serial_number>1</serial_number>
  <report_date>2019-11-23</report_date>
  <status>En service</status>

  <automate_information>
    <state>ok</state>
    <temperature>5</temperature>

    <payment_state>
      <coins>normal</coins>
      <smart_card>erreur</smart_card>
      <card>normal</card>
    </payment_state>


  </automate_information>

  <errors>
    <error>
      <type>ax23</type>
      <description>azerty</description>
    </error>
  </errors>

  <articles>
    <article>
      <name>Cafe</name>
      <quantity>20</quantity>
      <type>chaud</type>
    </article>

    <article>
      <name>Eau</name>
      <quantity>30</quantity>
      <type>froid</type>
    </article>

  </articles>

  <income>300</income>

</daily_report>
```

Puis nous avons défini les structures de nos données et les relations qui existeront entre elles une fois que la base de données sera générée.

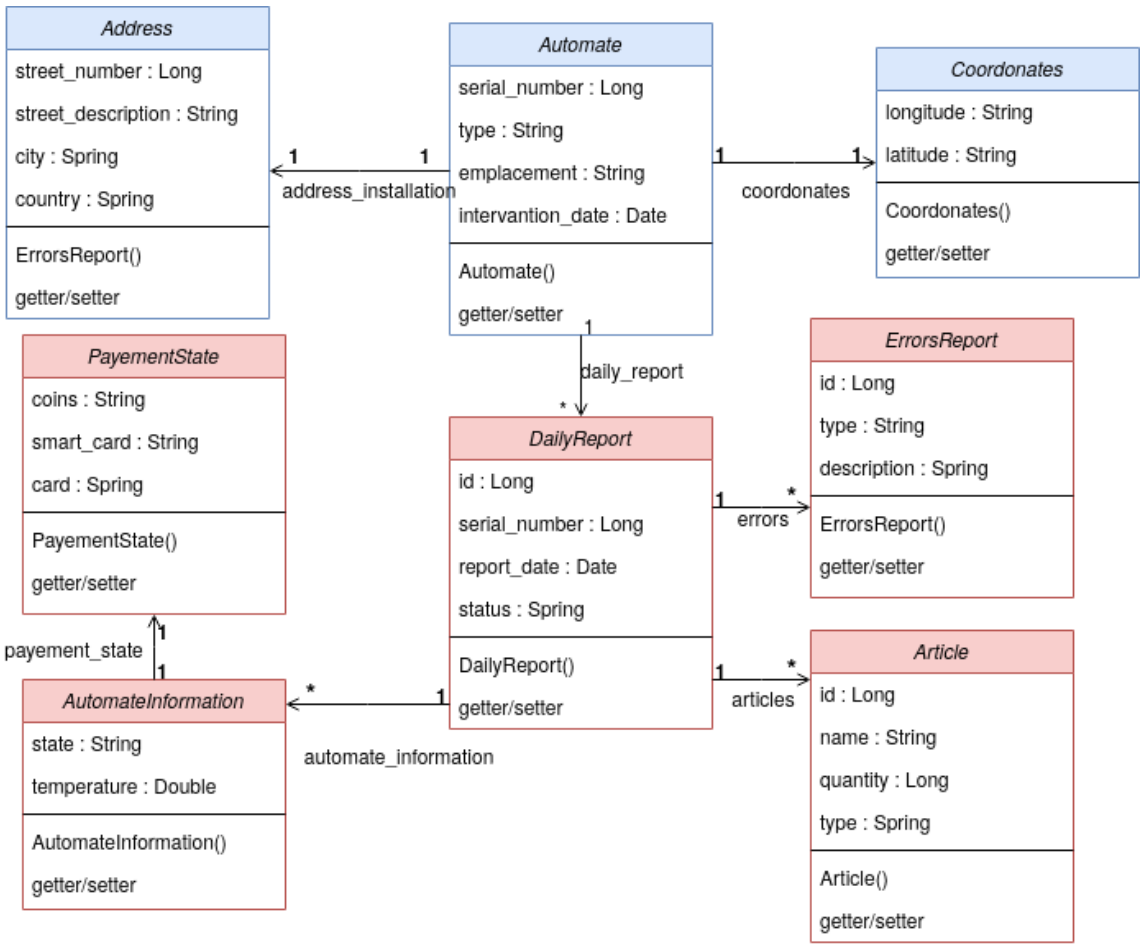


FIGURE 1 – Diagramme de classe Java

Et enfin, nous avons dû modéliser les différents flux d'échanges qui existeraient entre nos applications.

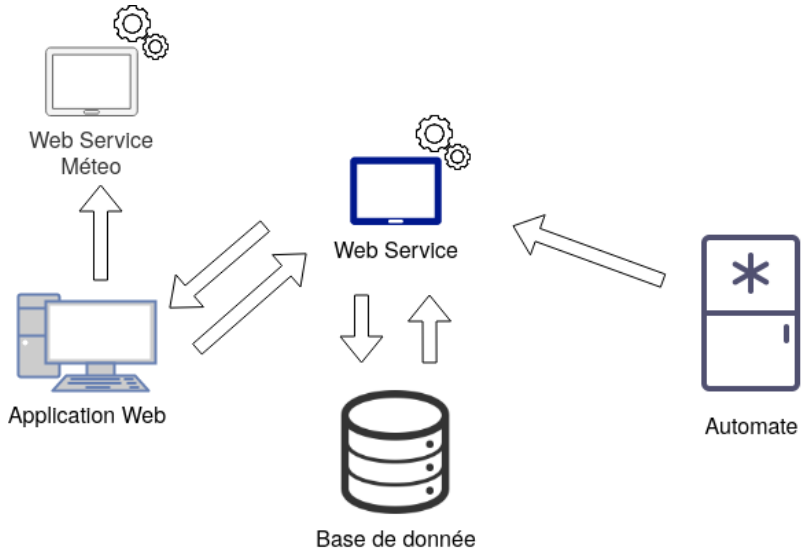


FIGURE 2 – Représentions des échanges de données entre les applications

3.2 Implémentation du web service

Pour la mise en place du web service nous avons utilisé le modèle MVC de spring qui comprend :

- Le package Model, qui permet de définir les classes avec des annotations spécifiques qui permettront à Hibernate de les traduire en tables dans notre base de données.
- Le package repositories, c'est ici que seront définies les méthodes qui permettront d'accéder à la base de données. Mais dans notre cas, nous nous sommes contentés d'implémenter une interface qui hérite des méthodes de la classe JpaRepository.
- Le package service, qui définit les services qui pourront être utiliser par le package controller
- Le package controller, c'est ici que les différents services seront associer à des URLs spécifique

Notre web service fournit donc les services suivant :

- Méthode Get
  - Liste des rapports
    (http ://localhost :8080/pwa/services/dailyReport/all)
  - rapport en fonction du numéro de séries de l'automate
    (http ://localhost :8080/pwa/services/dailyReport/serialNumber/1)
  - Rapport en fonction de son identifiant
    (http ://localhost :8080/pwa/services/dailyReport/1)
  - Liste des automates
    (http ://localhost :8080/pwa/services/automate/all)
  - Automates en fonction de son numéro de séries
    (http ://localhost :8080/pwa/services/automate/serialNumber/1)
  - Vérifier si un automate existe
    (http ://localhost :8080/pwa/services/automate/Check/serialNumber/1)

- Méthode Post
  - Ajouter un rapport au format JSON et XML
    (http ://localhost :8080/pwa/services/dailyReport/add)
  - Ajouter une liste de rapports au format JSON et XML
    (http ://localhost :8080/pwa/services/dailyReport/add/list)
  - Ajouter un automate au format JSON
    (http ://localhost :8080/pwa/services/automate/add)
  - Ajouter une liste d'automates au format JSON
    (http ://localhost :8080/pwa/services/automate/add/list)

- Méthode Delete
  - Supprimer un rapport en fonction de son identifiant
    (http ://localhost :8080/pwa/services/dailyReport/delete/1)
  - Supprimer tous les rapports
    (http ://localhost :8080/pwa/services/dailyReport/delete/all)

- Supprimer un automate en fonction de son numéro de serie
  (http ://localhost :8080/pwa/services/automate/delete/3)
- Supprimer tous les automates
  (http ://localhost :8080/pwa/services/automate/delete/all)

— Méthode Put

—

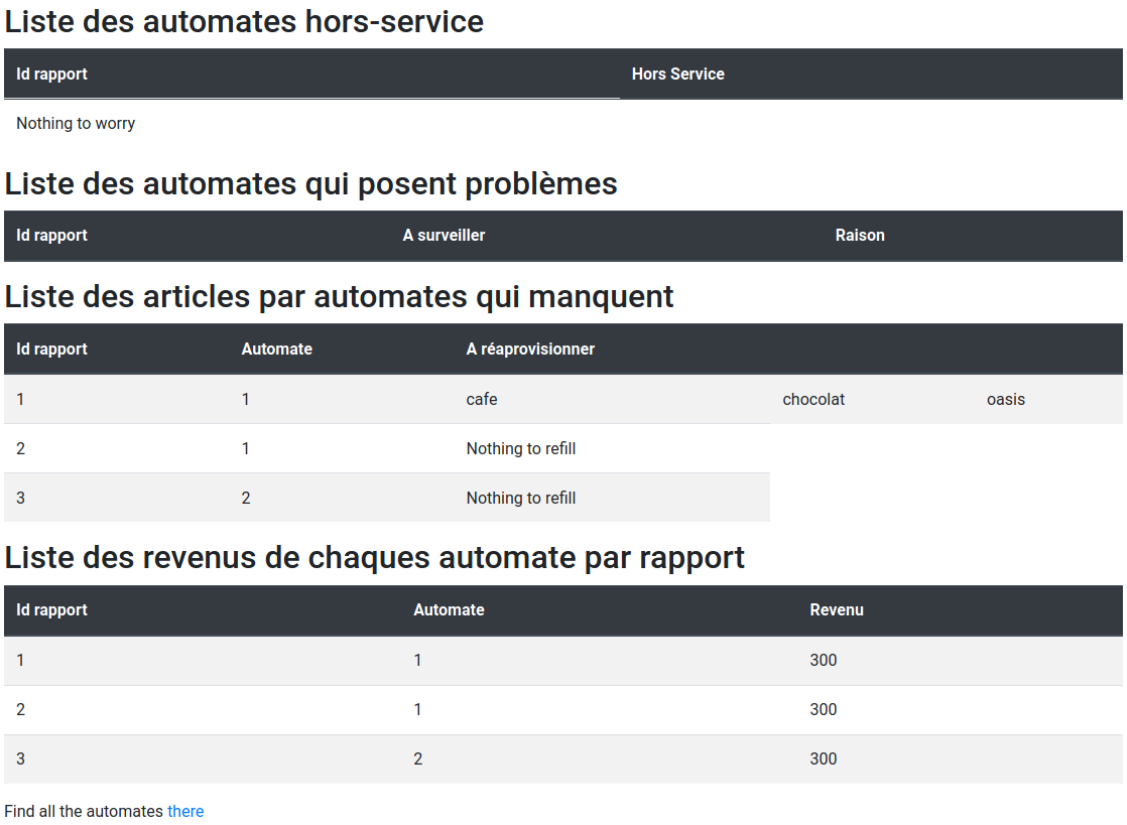
3.3 Implémentation de l'application web

Pour la mise en place de l'application Web nous avons utilisé le modèle Web de Spring Boot, qui contient notamment RestTemplate.

En effet le principe de notre application web est de consommer un web service qui est du type API Restful. Le principe comme expliqué précédemment est que l'application web communique avec le web service grâce a différentes requêtes HTTP telles que GET, POST ou/et DELETE.

Nous sommes partis du principe que comme nous devellopions le web service et l'application web, que nous connaissions dès le départ la structure des informations envoyées par l'API. C'est à ce moment qu'interviens restTemplate, en connaissant la structure des fichiers JSON recueillis nous pouvons créer les classes contenant les champs que nous souhaitons et qui sont identique aux champs du fichier JSON. En appellant alors les fonctions de la classe restTemplate, les champs de la classe donnée en paramêtre seront alors automatiquement remplis en fonction des données récupérées. Le principe est le même lorsque nous voulons faire une requête POST.

Voic quelques captures d'écran de l'application Web :



Liste des automates hors-service

Id rapport	Hors Service
Nothing to worry	

Liste des automates qui posent problèmes

Id rapport	A surveiller	Raison

Liste des articles par automates qui manquent

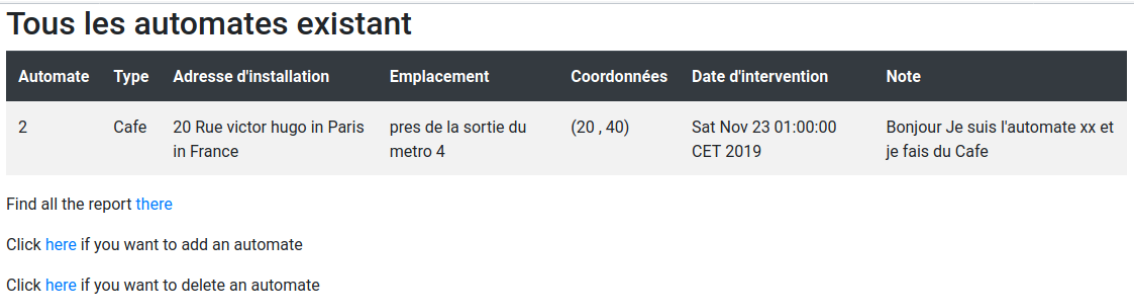
Id rapport	Automate	A réaprovisionner		
1	1	cafe	chocolat	oasis
2	1	Nothing to refill		
3	2	Nothing to refill		

Liste des revenus de chaques automate par rapport

Id rapport	Automate	Revenu
1	1	300
2	1	300
3	2	300

Find all the automates there

FIGURE 3 – Page de l'application ou sont représentés les différents tableaux de bords liés aux Rapports journaliers



Tous les automates existant

Automate	Type	Adresse d'installation	Emplacement	Coordonnées	Date d'intervention	Note
2	Cafe	20 Rue victor hugo in Paris in France	pres de la sortie du metro 4	(20 , 40)	Sat Nov 23 01:00:00 CET 2019	Bonjour Je suis l'automate xx et je fais du Cafe

Find all the report there

Click here if you want to add an automate

Click here if you want to delete an automate

FIGURE 4 – Page affichant les différents automates présent dans la base de données

L'application web contient d'autres pages, mais ce sont seulement des simples forms utilisés pour ajouter un automate ou en supprimer un.

L'addresse web à laquelle vous devez vous rendre pour pouvoir le voir est la suivante : http ://localhost :8081/
A partir de cette page vous pourrez naviguer dans l'application web sans soucis.

4 Difficultés rencontrées

Lors de la réalisation de ce projets nous avons rencontrés quelques difficultés que nous avons toutes résolues.

Les premières difficultées furent la prise en main de Spring boot, en effet la logique est un tout petit peu différente que celle vue en cours mais nous savions que une fois prise en main, nous n'aurions plus de soucis et que nous avancerions plus vite.

Par exemple, le main de l'application web se doit d'être en terme de hierarchie de package plus haut que toutes les autres classes et controlleurs, sinon ceux-ci ne seront pas pris en compte car ils sont en dehors du contexte du main en question. (Ce n'est pas vraiment un main, c'est une fonction qui initialise et lance tout ce qui a été préparé avec Spring).

Un chose qui s'est avérée compliquée fut aussi l'utilisation de Rest Template. Il y a différentes fonctions pour arriver au même résultat mais chacunes doit s'utiliser dans des cas vraiment précis. Notre cas à nous s'est avéré un cas particulier. En effet, lors de nos requêtes GET nous récupérons une liste de classes que nous avons appelé DailyReport ou Automate. Or afin de récuperer une liste et de pouvoir faire des actions dessus avant de les envoyer aux JSP (comme par exemple trier la liste) il y a des choses à préparer parce que cela ne se fait pas automatiquement. Le plus gros problème fut de prendre dès le début les bonnes manières avec Spring qui sont une condition nécessaire pour que le programme fonctionne.

5 Améliorations possibles

Nous avons comme expliqué précedemment réussi implémenter l'ajout pour les fiches des automates et des rapports journaliers (en XML ou en JSON) et la suppréssion des fiches automates. Et de nombreuses autres features comme l'affichage des rapports journalier en ordre décroissants des ventes.

Nous avons cependant pensé à quelques amélioration possible pour notre projet que voici :

— Spécifier les contraintes sur les colonnes dans les classes Java afin qu'elles soient rajoutées lors de la création de la base de données

— Implémenter une sécurité côte web service à l'aide de Spring Boot Security

— Utiliser la feature de Spring boot afin de parer les failles XSS coté Web App

— Implémenter la possibilité de modifier un automate du côté Web Service puis du côté Web App

— Vérifier que les champs rempli par les utilisateurs de l'application web respectent certaines contraintes, par exemple si le type de l'article n'est pas exactement "chaud", "froid" ou "Nourriture" les résultats affichés seront alors incorrects.

— Améliorer grandement l'interface, pour la navigation sur les différentes pages, pour l'éstethetique et la practicité du l'application web.

— Ajouter un champ sur la structure de l'automate afin de préciser si il communique en XML ou en JSON.

— Afficher les rapports journaliers du plus récent au plus vieux dans les tableaux de bords.

6 Réponse aux questions bonus

6.1 Compte-rendu des techniciens

Afin d'implémenter cette nouvelle fonctionnalité à nos applications. Il faudrait :

— Ajouter une nouvelle classe Java (CompteRendu) et définir la relation entre cette table et la table des automates (à priori, un automate peut avoir zéro ou N compte rendu)

— Mettre en place les services qui permettront l'ajout, la consultation, la modification et la suppression des comptes rendus

— Développe les interfaces qui permettront d'exploiter ces nouveaux services

6.2 Solutions techniques pour le directeur marketing

Afin d'affiner le suivit des ventes de chacunes des machines nous avons pensé à différentes solution.

La première serait de recuperer tous les automates et tous les rapports journaliers dans un controleur. De remplir soit une nouvelle classe soit s'en sortir avec les listes en recupérant sur le dernier rapport journalier (celui de date d'emission le plus grand) la liste des article et en l'associant a l'automate correspondant grâce a son numéro de série.

La seconde serait que le web service envoie sous format json, les éléments de la table associations de dailyReport et automate ne contenant que les articles des différents automates. Puis avec l'application web nous récupérons ces éléments et les affichons.