

2024/2025

RAPPORT FINAL JAVA

BROUCAS Adélaïde, CISSE Flavie, RUIZ Julien,
FERREIRA DA SILVA Marwan, LANDURE Yvann

Le projet vise à déployer un réseau de poubelles intelligentes équipées de capteurs mesurant en temps réel le volume et la nature des déchets déposés, accessibles via un badge ou un code utilisateur pour tracer chaque dépôt et créditer automatiquement des points de fidélité (ou appliquer des pénalités en cas de tri incorrect)

ENSEIGNANT :

Renaud VERIN

CLASSE :

ING1 GM4



Table des matières

Introduction	2
I. Présentation de la structure de notre projet.....	3
a) Modifications apportées à notre Base de Données.....	3
b) Modifications apportées à notre UML.....	4
c) Organisation de nos packages avec le logiciel Eclipse	7
II. L'interface pour le ménage	8
III. L'interface pour le centre de tri	11
Conclusion.....	16

Introduction

Dans le cadre de notre première année de cycle ingénieur, nous avons dû mener à réaliser un projet de programmation en langage Java. Ce projet qui vise à développer une application de gestion du tri sélectif a nécessité de faire appel à des compétences variées issues des différentes thématiques du cours.

L'objectif de cette application est de mettre en place un système de fidélisation des ménages encourageant le tri des déchets grâce à des poubelles intelligentes. Ces dernières, équipées de capteurs, permettent de reconnaître le type et la quantité de déchets déposés. Les utilisateurs accumulent ainsi des points qu'ils peuvent convertir en réductions auprès des commerces partenaires. Le projet implique plusieurs acteurs : les ménages, les centres de tri, les commerces partenaires et les poubelles intelligentes, chacun ayant des interactions spécifiques.

Ce projet était découpé en trois grandes étapes, chacune donnant lieu à un rendu. Tout d'abord, nous avons modéliser notre application sous forme d'un diagramme de classe UML à l'aide du logiciel StarUML. Dans un second temps, nous avons dû implémenter une partie de code de notre projet avec des fonctions de tests en utilisant le logiciel Eclipse. Finalement, nous nous sommes intéressés au développement de l'interface Homme-Machine (IHM) en créant une interface graphique permettant aux utilisateurs d'interagir avec l'application.

Ce rapport porte sur le dernier rendu, qui consiste à réaliser l'interface Homme-Machine (IHM) correspondante à notre projet. Tout d'abord, nous reviendrons sur les modifications apportées depuis les rendus précédents. Cela concerne la base de données, l'UML et le code. Ensuite, nous présenterons les deux interfaces de notre projet : celle pour les ménages et celle pour les centres de tri. Enfin, nous ferons le point sur les principales difficultés rencontrées lors de la réalisation de ce projet ainsi que sur les améliorations possibles.

I. Présentation de la structure de notre projet

a) Modifications apportées à notre Base de Données

Lors du second rendu du projet, nous utilisions PHPMyAdmin pour la création et la gestion de notre base de données. Toutefois, pour ce dernier rendu, nous avons décidé de migrer notre base vers MySQL Workbench. Ce choix s'est imposé naturellement, car MySQL Workbench propose une interface plus performante pour la conception visuelle de schémas relationnels, tout en facilitant l'établissement et la visualisation des relations entre les différentes entités. Grâce à cet outil, nous avons pu générer un schéma clair et structuré, améliorant ainsi la cohérence entre notre modèle UML, notre base de données et notre code Java.

En analysant notre ancienne base, nous avons identifié plusieurs améliorations nécessaires, notamment dans la définition des relations entre les tables. Certaines liaisons étaient redondantes ou mal positionnées, ce qui risquait de créer des incohérences lors de la synchronisation avec nos objets métiers.

Nous avons modifié la relation entre les tables Déchets et Dépôts. La table Déchet était liée à la table Dépôts via une clé étrangère DepotId. Nous avons revu cette conception en supprimant cette clé étrangère. Désormais, c'est la table Depot qui contient une référence au DechetId car un dépôt est associé à un type de déchet précis.

De plus, la table Depot comportait un champ supplémentaire indiquant le type du déchet. Cette information est déjà donnée dans la table Dechet donc nous avons supprimé ce champ pour éviter toute redondance et assurer la cohérence des données.

Lors de la création de notre IHM, nous nous sommes rendus compte qu'un centre de tri doit pouvoir se connecter sur notre interface de façon sécurisée. Nous avons donc ajouté un champ entier nommé « code » dans la table CentreTri. Ce champ joue le rôle de mot de passe et permettra d'authentifier les centres de tri lors de la connexion à l'interface dédiée.

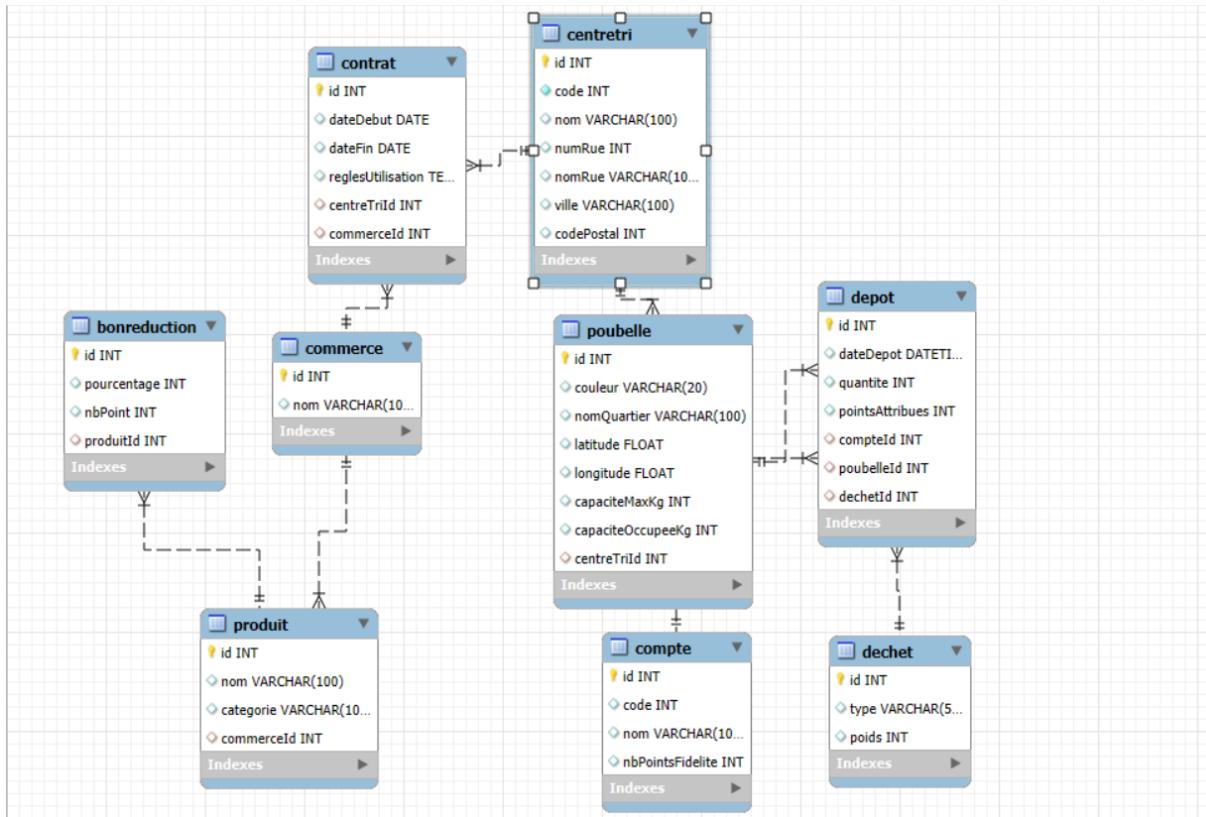


Figure 1 : Schéma relationnel de la base de données réalisé sous MySQL Workbench.

b) Modifications apportées à notre UML

Dans la version initiale, les classes PoubelleVerte, PoubelleBleue, PoubelleNoire et PoubelleJaune possédaient un attribut couleur de type Couleur. Dans la version finale, cet attribut a été supprimé afin d'éviter une redondance inutile, la couleur pouvant être déterminée autrement, sans que les sous-classes aient besoin de stocker directement cette information.

Par ailleurs, dans la version initiale, il existait une dépendance implicite entre les classes Contrat et CentreTri. Dans la version finale, aucun attribut CentreTri n'apparaît dans la classe Contrat, la relation étant désormais uniquement gérée par une association UML, ce qui respecte mieux les normes de modélisation.

De plus, dans la version initiale, les classes PoubelleVerte, PoubelleBleue et PoubelleNoire ne disposaient pas de l'ensemble de leurs getters et setters. Nous avons donc ajouté dans la

version finale les méthodes d'accès nécessaires pour chaque capacité spécifique (Verre, Papier, Autres, permettant un accès correct et sécurisé aux attributs.

Enfin, certaines classes comme Contrat, Depot, Produit et Rejet ne possédaient pas d'attribut id dans la version initiale. Dans la version finale, des identifiants ont été ajoutés à ces classes afin de garantir une meilleure traçabilité.

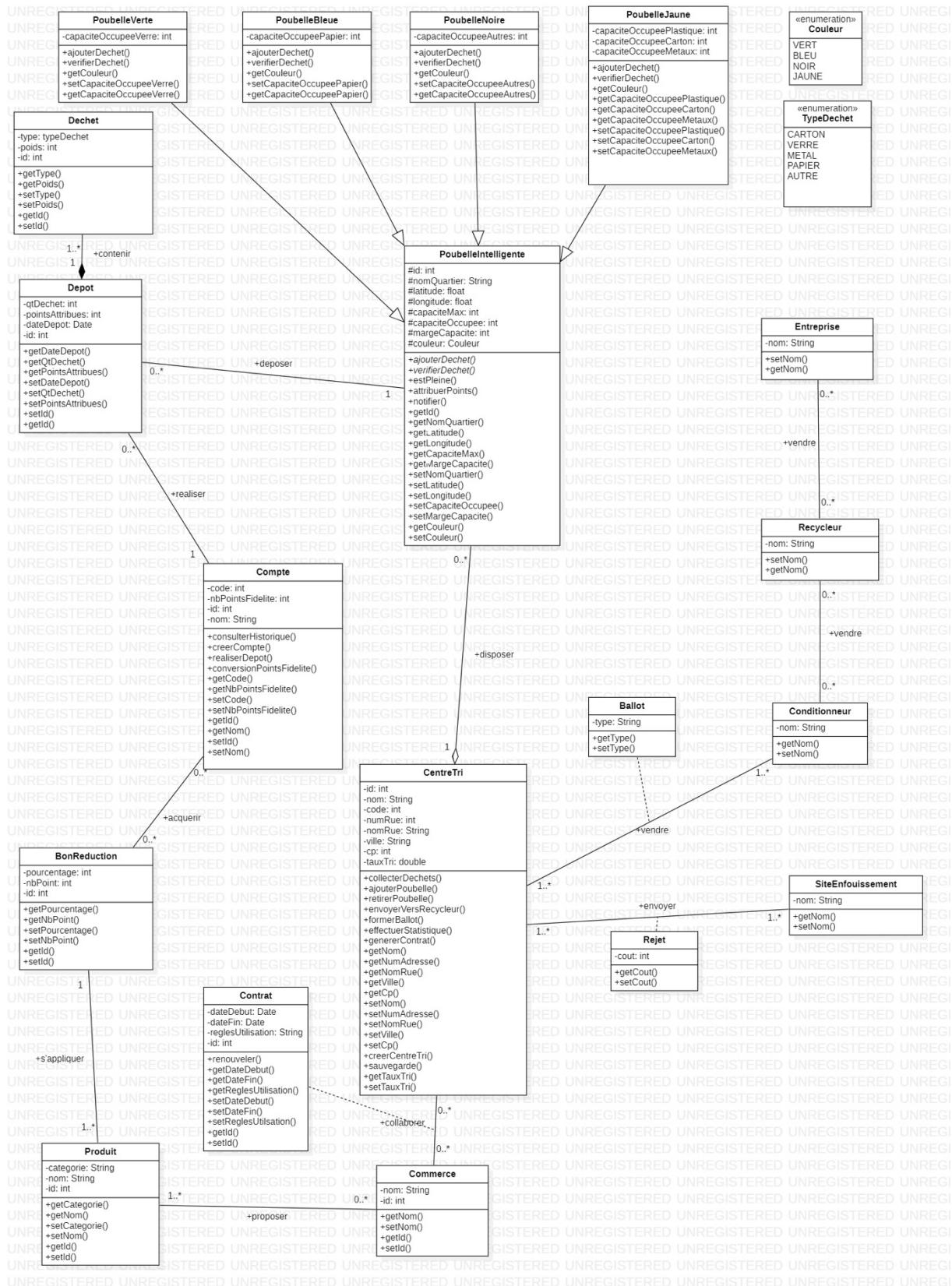


Figure 2 : Schéma UML final de l'application de tri sélectif

c) Organisation de nos packages avec le logiciel Eclipse

Notre projet s'organise sous forme de plusieurs packages regroupés dans le dossier src. Nous avons créé 5 packages :

1. com.cytech.connexionDAO
2. com.cytech.fxml
3. com.cytech.main
4. com.cytech.model
5. com.cytech.testUnitaires

Chaque package gère un aspect spécifique de l'application.

Le package com.cytech.connexionDAO regroupe les classes gérant la connexion et les interactions avec la base de données sur MySQLWorkbench.

Le package com.cytech.model rassemble les classes métiers représentant les acteurs et objets de notre système.

Le package com.cytech.testUnitaires contient les classes de tests unitaires, permettant de vérifier individuellement le bon fonctionnement des différentes parties du projet. On y retrouve les classes tests des classes métier et des classes DAO.

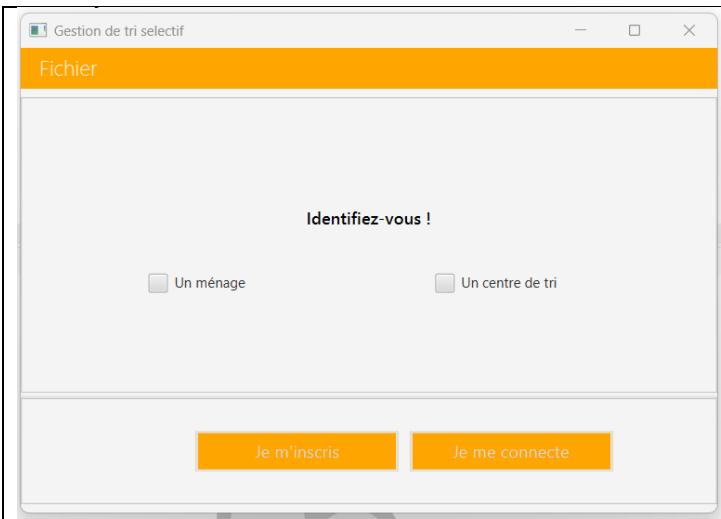
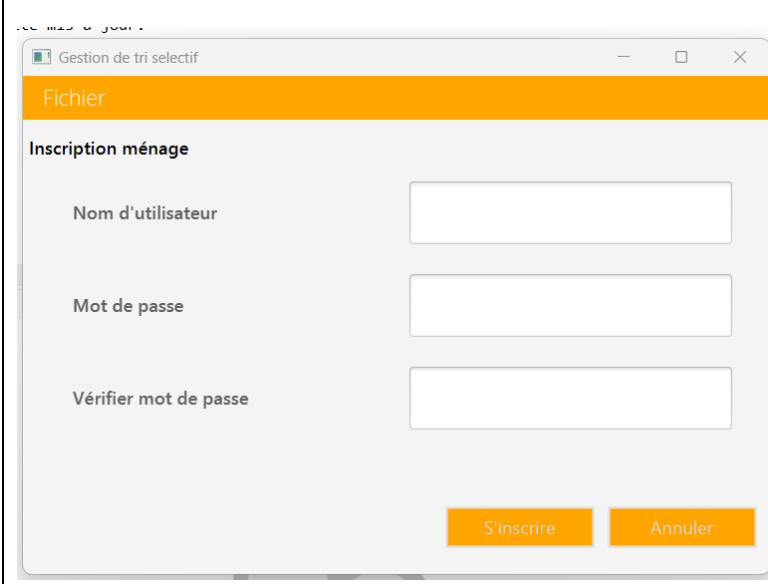
Le package com.cytech.main contient la classe TestMain qui permet de lancer les classes tests ainsi que la classe MainApp qui permet d'exécuter l'IHM et de gérer la navigation entre les scènes.

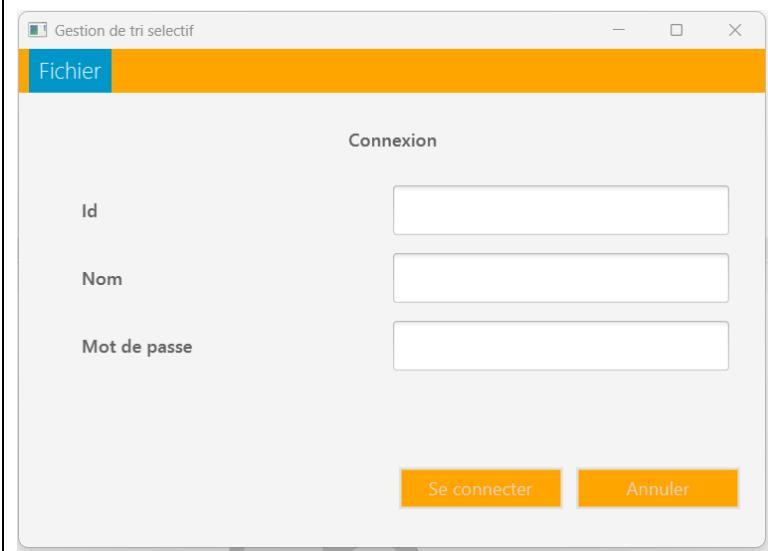
Finalement, le package com.cytech.fxml contient l'ensemble des fichiers FXML décrivant la structure visuelle de notre IHM ainsi que leurs contrôleurs Java associés. Les fichiers FXML vont venir décrire la structure et le graphisme de chaque scène de notre IHM ainsi que les propriétés graphiques de chaque composant de ces scènes. Les contrôleurs de ces fichiers vont permettre de manipuler les différents composants définis dans le fichier FXML, d'y accéder et de modifier leur contenu en fonction des instructions de l'utilisateur.

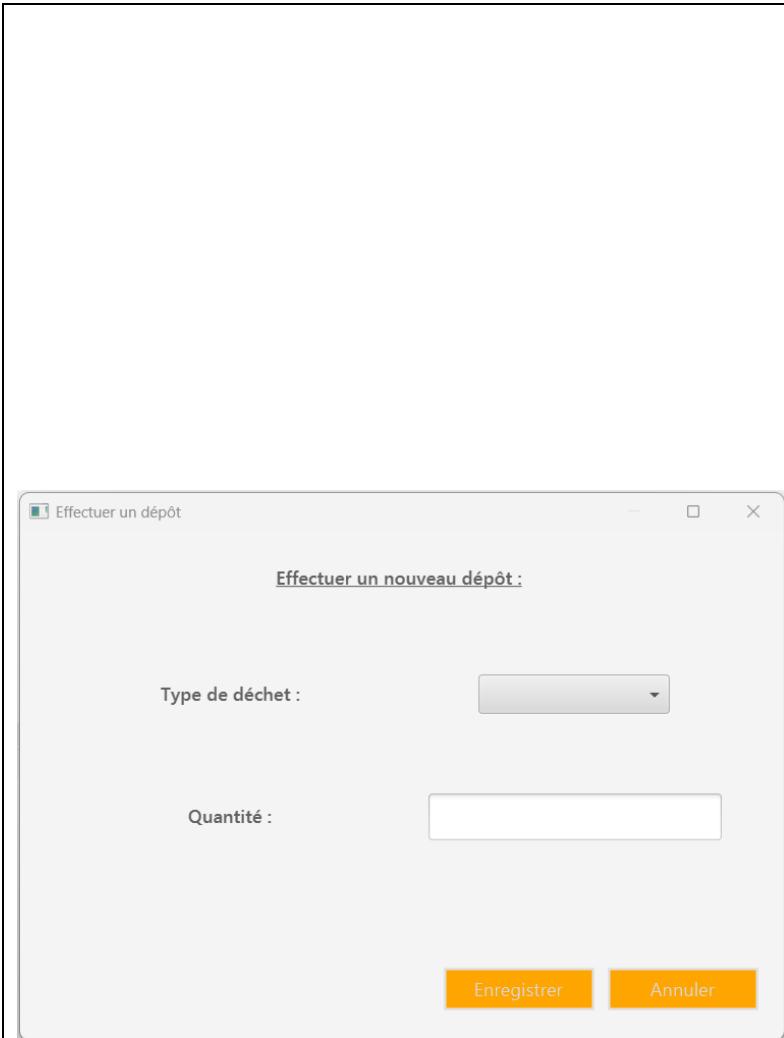
Si nous avions disposé de davantage de temps, nous aurions pu poursuivre l'optimisation de notre code en simplifiant certaines méthodes, en supprimant les redondances inutiles et en améliorant encore la lisibilité générale du projet. En effet, certaines méthodes sont assez longues ou complexes, ce qui peut rendre leur compréhension plus difficile. Nous aurions pu

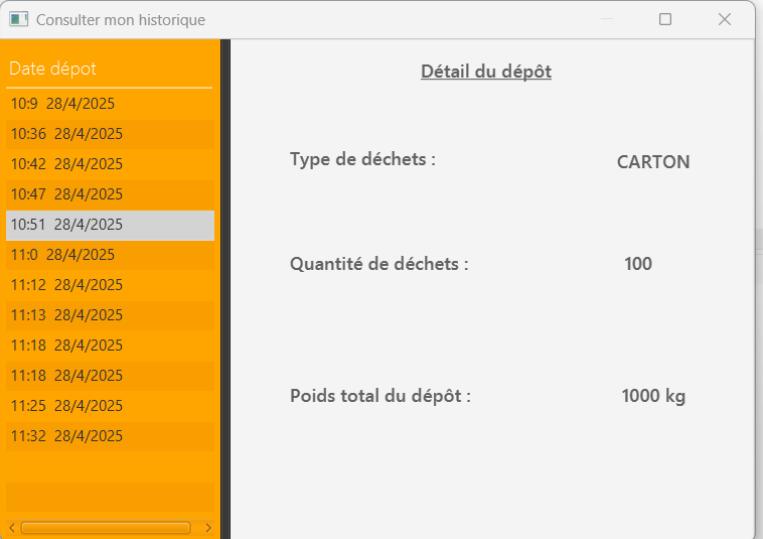
améliorer cet aspect en simplifiant certaines fonctions, en les découplant en sous-méthodes plus courtes et en ajoutant davantage de commentaires pour mieux expliquer les choix réalisés.

II. L'interface pour le ménage

	<p>Tout d'abord, le ménage doit pouvoir s'inscrire ou se connecter à l'interface qui lui est propre. Lors de l'exécution de notre IHM, l'utilisateur arrive sur la page d'accueil, qui va lui permettre de s'identifier en tant que ménage. Puis il va devoir décider s'il souhaite s'inscrire ou se connecter.</p>
	<p>S'il décide de s'inscrire, le ménage sera redirigé vers une page d'inscription qui va lui permettre de renseigner différents champs. Si ces différents champs correspondent aux critères que l'on a fixés (champs non nuls et mot de passe composé uniquement de chiffres), alors l'utilisateur se voit attribuer un identifiant. Pour que le ménage puisse s'inscrire correctement, nous lui demandons de confirmer le mot de passe. Si les deux mots de passe rentrés ne correspondent pas alors un message d'erreur s'affiche, invitant ainsi l'utilisateur à corriger son erreur.</p>

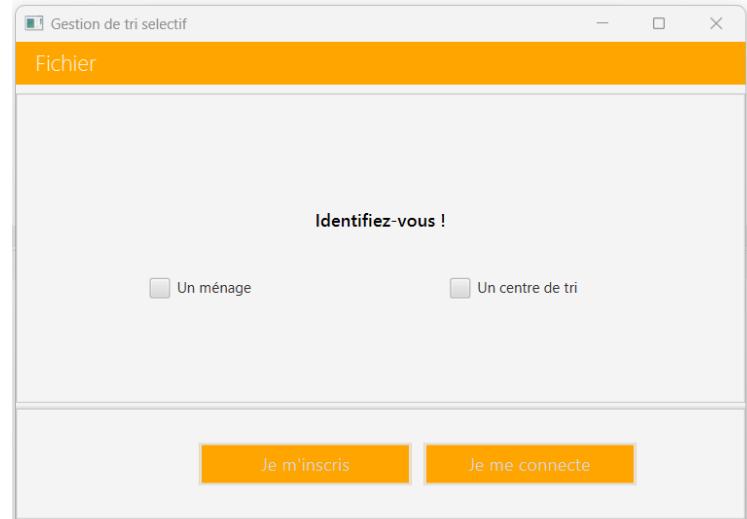
	<p>S'il décide de se connecter, le ménage sera redirigé vers une page de connexion. Par souci de simplicité, cette page de connexion est identique pour le ménage et le centre de tri. Cependant, afin de pouvoir rediriger l'utilisateur vers l'interface qui lui correspond, nous avons créé un booléen <code>isConnexionCompte</code> dans la classe « <code>PageConnexionController</code> » qui va prendre la valeur <code>True</code> si c'est un ménage qui se connecte et <code>False</code> s'il s'agit d'un centre de tri. Pour se connecter, un ménage aura besoin de son id, de son nom et du mot de passe qu'il a lui-même défini.</p>
	<p>Une fois qu'il a pu accéder à l'interface qui lui est attribuée, le ménage peut visualiser le nombre de points de fidélité dont il dispose. Cette information se trouve sur la gauche de notre interface graphique. Il peut également convertir ses points de fidélité en bons d'achats. Pour cela, il doit disposer d'au moins 200 points. Si cette condition est remplie, alors le ménage pourra convertir 200 points en un bon d'achat de 10%.</p>

	<p>Depuis son interface, le ménage peut réaliser un dépôt. Pour cela, il devra renseigner l'identifiant de la poubelle dans laquelle il souhaite faire son dépôt ainsi que l'identifiant et le nom du centre de tri auquel la poubelle est rattachée. Ces informations seraient indiquées sur une étiquette en métal visible sur chaque poubelle intelligente, et donc facilement lisible par le ménage. Puis, le ménage peut effectuer son dépôt en cliquant sur le bouton « Réaliser mon dépôt ». En cliquant sur ce bouton, le ménage voit apparaître un pop-up qui lui permet de renseigner la quantité de déchets qui constituent son dépôt. Pour indiquer le type des déchets qu'il souhaite ajouter à son dépôt, le ménage devra sélectionner le bon type dans un menu déroulant. Les éléments de ce menu déroulant sont ceux de la classe d'énumération nommée « TypeDechet ». Lorsque l'utilisateur va cliquer sur le bouton « Enregistrer », toutes les informations qu'il a renseignées concernant son dépôt seront enregistrées dans l'historique de ses dépôts.</p>
---	--

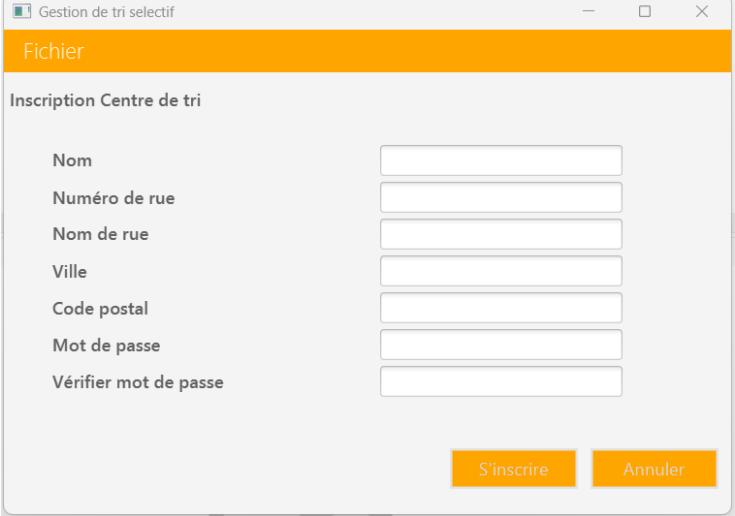
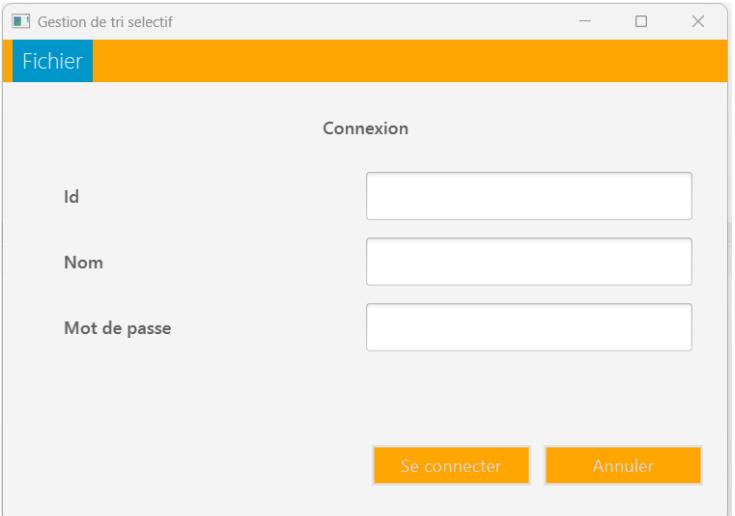


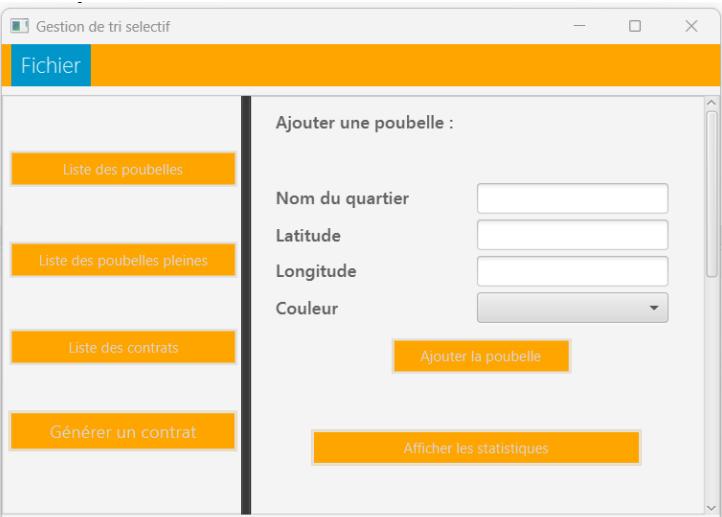
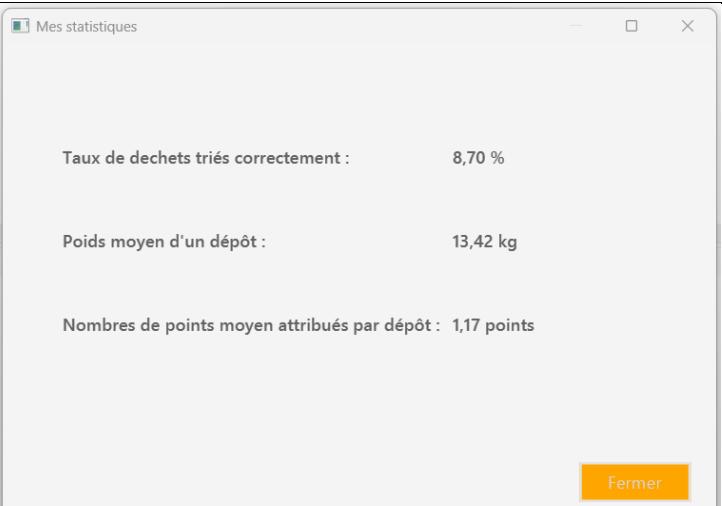
Pour consulter l'historique de ses dépôts, le ménage peut cliquer sur le bouton « Consulter l'historique de mes dépôts ». En cliquant sur ce bouton, le ménage va déclencher l'apparition d'une fenêtre pop-up. Sur la gauche de cette fenêtre se trouvent les différents dépôts réalisés par le ménage. Lorsqu'il sélectionne un dépôt dans la liste, il peut afficher la quantité et le type des déchets qui le composent ainsi que le poids total du dépôt.

III. L'interface pour le centre de tri

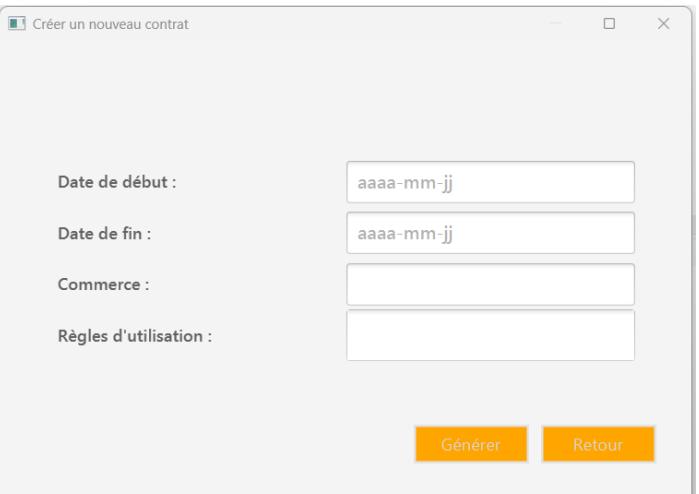
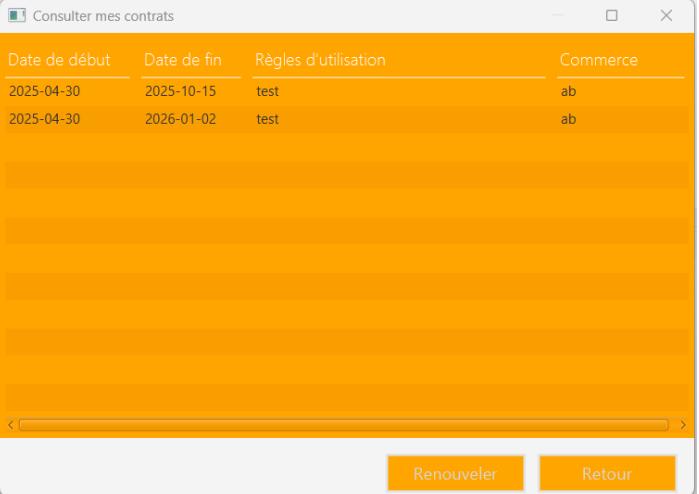
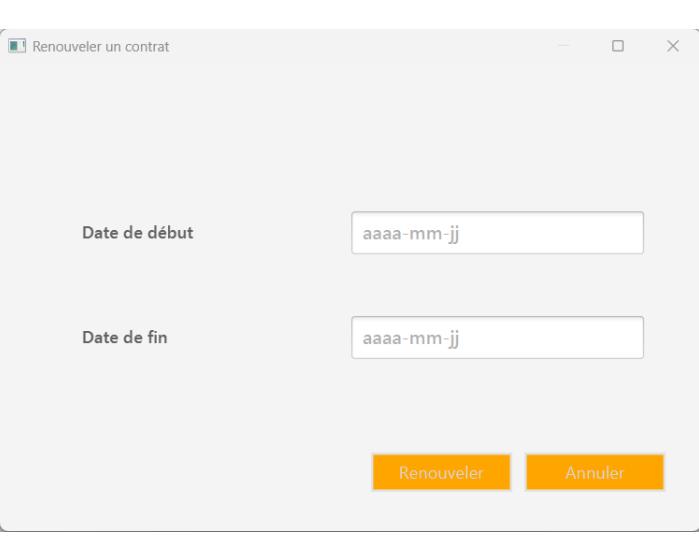


Tout d'abord, le centre de tri doit pouvoir s'inscrire ou se connecter à l'interface adaptée. Lors de l'exécution de notre IHM, l'utilisateur arrive sur la page d'accueil, qui va lui permettre de s'identifier en tant que centre de tri. Puis il va devoir décider s'il souhaite s'inscrire ou se connecter.

	<p>S'il décide de s'inscrire, le centre de tri sera redirigé vers une page d'inscription qui va lui permettre de renseigner différents champs. Si ces différents champs correspondent aux critères fixés (champs non nuls, mot de passe composé uniquement de chiffres...), alors l'utilisateur se voit attribuer un identifiant. Pour que le centre de tri puisse s'inscrire correctement, nous lui demandons de confirmer le mot de passe. Si les deux mots de passe rentrés ne correspondent pas alors un message d'erreur s'affiche, invitant ainsi l'utilisateur à corriger son erreur.</p>
	<p>S'il décide de se connecter, le centre de tri sera redirigé vers une page de connexion. Cette page de connexion est identique pour le ménage et le centre de tri. Cependant, afin de pouvoir rediriger l'utilisateur vers l'interface qui lui correspond, nous avons créé un booléen <code>isConnexionCompte</code> dans la classe « <code>PageConnexionController</code> » qui va prendre la valeur <code>True</code> si c'est un ménage qui se connecte et <code>False</code> s'il s'agit d'un centre de tri. Pour se connecter, un centre de tri aura besoin de son id, de son nom et du mot de passe qu'il a lui-même défini.</p>

	<p>Une fois qu'il accède à l'interface qui lui est dédiée, le centre de tri peut choisir d'ajouter une poubelle intelligente. Pour cela, il va devoir renseigner différents champs. Pour indiquer la couleur de la poubelle qu'il souhaite ajouter, le centre de tri doit sélectionner un élément dans un menu déroulant, contenant les différentes couleurs possibles définies dans la classe d'énumération « Couleur ». Une fois qu'il a renseigné tous les champs demandés, le centre de tri peut cliquer sur le bouton « Ajouter la poubelle ». S'il n'y a pas d'erreur alors il recevra un message de confirmation lui indiquant que la poubelle a bien été ajoutée à la liste des poubelles. Dans le cas contraire, il recevra un message d'erreur l'invitant à modifier sa saisie.</p>
	<p>Le centre de tri peut aussi visualiser les statistiques associées aux dépôts qui ont été réalisés dans les poubelles intelligentes qu'il possède. Ils peuvent ainsi lire le taux de déchets triés correctement, le poids moyen d'un dépôt, ainsi que le nombre de points attribués par dépôt en moyenne.</p>

	<p>A gauche de l'interface graphique, se trouvent quatre boutons qui, lorsque l'on clique dessus, une fenêtre pop-up apparaît. Ces fenêtres affichent des tableaux TableView contenant des éléments observables. Par exemple, le pop-up intitulé PopUpListePoubelle va contenir un tableau TableView composés de PoubelleObservable et chaque colonne de ce tableau contiendra les différentes valeurs des différents attributs de la classe « PoubelleObservable ». En sélectionnant une poubelle dans la liste des poubelles, il peut choisir de la retirer ou de la vider. Dans les deux cas, il recevra un message de confirmation si l'action choisie a bien pu être réalisée.</p>
	<p>Le centre de tri peut également visualiser toutes ses poubelles pleines. Pour cela, il va devoir cliquer sur le bouton « Liste des poubelles pleines » sur la partie gauche de son interface. Une fenêtre pop-up avec les différentes poubelles pleines et leurs caractéristiques apparaît alors. En sélectionnant une poubelle dans la liste des poubelles pleines, il peut choisir de la retirer ou de la vider.</p>

	<p>Le centre de tri peut également décider de générer un contrat. Pour cela, il va devoir remplir différents champs avant de cliquer sur le bouton « Générer ». Afin d'éviter des erreurs inutiles, nous avons choisi de préciser le format sous lequel il fallait renseigner la date de début du contrat ainsi que la date de fin. Si le contrat a bien pu être généré un message de confirmation apparaît et le contrat apparaît alors dans la liste de contrat.</p>
	<p>Le centre de tri peut également visualiser tous ses contrats. Pour cela, il va devoir cliquer sur le bouton « Liste des contrats » sur la partie gauche de son interface. Une fenêtre pop-up avec les différents contrats et leurs caractéristiques apparaît alors.</p>
	<p>Pour le pop-up intitulé PopUpListeContrat, il est possible de renouveler un contrat en le sélectionnant dans la liste puis en cliquant sur le bouton « Renouveler ». Un nouveau pop-up apparaît alors. Le centre de tri doit renseigner plusieurs champs avant de pouvoir enregistrer la modification. Cependant, le contrat après le renouvellement apparaîtra dans la liste des contrats comme étant un contrat différent du premier. Nous avons fait le</p>

	choix de garder une trace de l'ancien contrat afin de faciliter la compréhension de l'historique de chaque contrat.
--	---

Conclusion

Nous avons ainsi mené à bien la dernière phase de ce projet de tri sélectif en Java. Malgré une deadline serrée, l'implication constante et la motivation de tous les membres du groupe ont permis de respecter les délais et de produire un travail de qualité. Ce projet s'est révélé être une expérience particulièrement enrichissante et nous avons dû nous remettre perpétuellement en question pour s'assurer de bien suivre les consignes malgré les divergences d'interprétation et d'appréhension du sujet.

Tout au long du projet, nous avons collaboré efficacement pour concevoir une solution réaliste et réalisable de gestion des déchets à l'échelle d'une ville. L'ensemble des trois rendus a été développé de manière progressive et structurée, ce qui nous a permis d'aboutir à une interface Homme-Machine (IHM) fonctionnelle, intuitive et ergonomique, répondant aux besoins opérationnels d'un tel système.

Au-delà de la simple application des connaissances théoriques acquises durant notre formation, ce projet nous a permis de consolider notre maîtrise du langage Java, de manipuler une base de données de manière concrète et de comprendre les enjeux liés à la conception d'une IHM efficace. Il nous a également poussés à développer de nouvelles compétences essentielles : travail d'équipe, gestion de projet, résolution de problèmes complexes et rigueur dans la programmation.

En définitive, ce projet nous a donné une vision plus claire de ce que représente le développement d'une application complète en conditions quasi-professionnelles, tout en nous sensibilisant aux enjeux environnementaux liés au tri des déchets. Il constitue une étape importante dans notre parcours d'apprentissage et nous a préparés de manière concrète aux défis que nous rencontrerons dans notre futur professionnel.