Rapport de Projet – CY-SIAO

CY Tech – ING1 GI – Année 2024-2025

Nom du projet : CY-SIAO

Équipe projet :

• BOUAYAD Sirine
• MAYEUR Charly
• EL FADALI Sarah
• HENNION Jules
• JACQUET Mathis

1. Introduction

Dans le cadre du module de projet informatique, notre équipe a développé une application de gestion pour un centre d'hébergement d'urgence. Le projet CY-SIAO répond aux besoins des dispositifs SIAO, qui coordonnent l'hébergement temporaire de personnes en situation précaire. L'application permet de gérer les lits, salles, personnes hébergées, et d'optimiser leur affectation via une interface graphique JavaFX connectée à une base de données MySQL.

2. Organisation de l'équipe

Notre groupe était composé de cinq membres. Voici la répartition des tâches :

• Sirine BOUAYAD : choix techniques, gestion du backlog, création de la base de données, gestion de l'interface graphique
• Charly MAYEURS : création du diagramme de classe et de la base de données et des cas d'utilisation, intégration JavaFX, résolution de problèmes de packages.
• Sarah EL FADALI : conception de la base de données, saisie des données, tests unitaires
• Jules HENNION : intégration de la base de données, contribution à la vue (JavaFX)
• Mathis JACQUET : modélisation, remplissage des objets, liaison BDD-Java

Nous avons utilisé Jira pour répartir les tâches et suivre l'avancement. GitHub a servi de dépôt principal avec des commits réguliers.

3. Analyse du besoin

L'objectif était de réaliser une application qui gère :
• Les personnes (nom, genre, date et lieu de naissance, etc.)
• Les salles et les lits (avec restrictions d'âge et de genre)
• Le planning d'occupation (dates de début/fin, affectation)
• L'affichage graphique de l'état d'occupation (code couleur)
• L'affectation automatique selon critères
Toutes les données devaient être stockées dans une base de données relationnelle.

4. Conception

Nous avons adopté une architecture MVC (Modèle-Vue-Contrôleur). Le diagramme de classes UML présente les entités principales : Person, Room, Bed, Assignment. Chaque entité est modélisée de façon à permettre une sérialisation simple et des jointures efficaces dans MySQL.
Les cas d'utilisation incluent :
• Ajouter / modifier / supprimer une personne
• Créer / supprimer des salles ou des lits

• Affecter automatiquement des personnes selon les critères
• Visualiser l'état des lits en temps réel

5. Implémentation

Le code a été développé en Java avec JavaFX pour l'interface. Nous avons structuré les packages en :
• controllers : pour la logique métier et les événements UI
• models : pour les classes métiers (Person, Room, etc.)
• database : pour la gestion JDBC avec MySQL
Le module d'affectation automatique propose des lits selon la disponibilité, les restrictions, et la durée souhaitée. Un système de couleurs indique visuellement l'occupation sur un calendrier intégré.

6. Problèmes rencontrés et solutions apportées

Nous avons rencontré plusieurs difficultés :
• Connexion instable à la base de données ➜ solution : pooling JDBC + gestion d'erreurs
• Affichage JavaFX lent avec beaucoup de données ➜ optimisation des listes et rafraîchissement partiel
• Affectation automatique trop rigide ➜ amélioration de l'algorithme avec tri par priorité (âge, genre)
• Conflits Git ➜ adoption d'un workflow avec branches claires et merges fréquents

7. Résultats

Toutes les fonctionnalités principales ont été implémentées :
Gestion complète des entités (CRUD)
Visualisation graphique et code couleur fonctionnel
Export/import de la base MySQL réalisé
JavaDoc générée pour l'ensemble du code
Application stable, sans crash ni boucle infinie

L'affectation automatique fonctionne bien dans la majorité des cas, et permet un réel gain de temps pour l'opérateur. Le design de l'interface est sobre, ergonomique, et personnalisable.

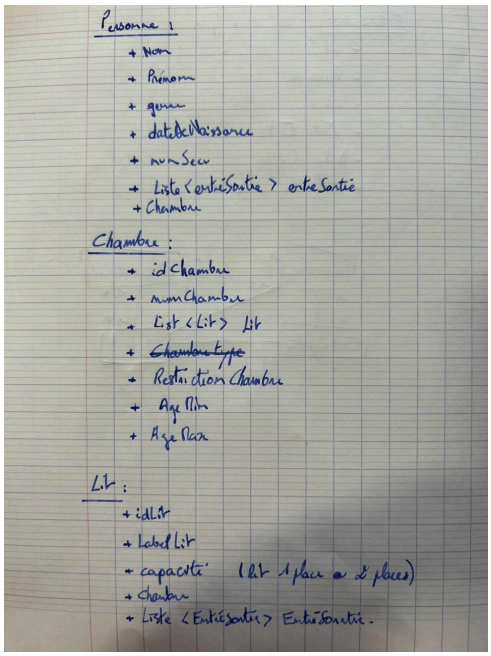
8. Conclusion et perspectives

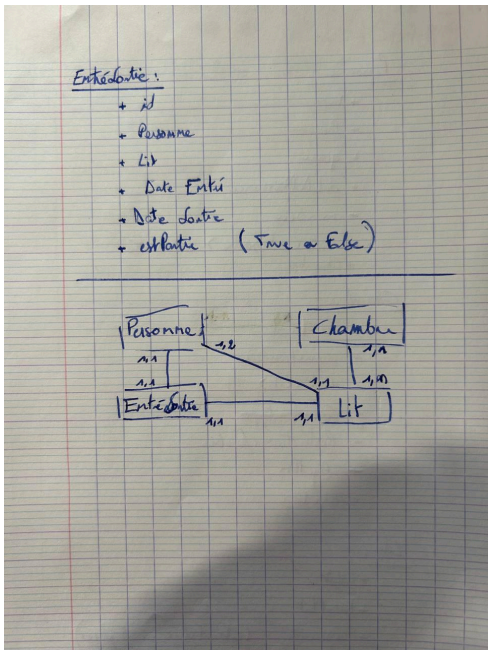
Ce projet nous a permis d'acquérir des compétences solides en gestion de projet, en conception orientée objet, et en développement Java avec interface graphique. Nous envisageons plusieurs pistes d'amélioration :
• Ajouter une gestion des utilisateurs et des rôles

• Intégrer une API pour importer des données externes
• Optimiser l'affichage temps réel avec des WebSockets ou un polling
• Déployer une version web avec une API REST

9. Annexes

• Diagramme de classes UML (à insérer ici)





• Script SQL de la base de données

```sql
CREATE TABLE IF NOT EXISTS person (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    last_name VARCHAR(100),
    first_name VARCHAR(100),
    gender VARCHAR(10),
    birth_date DATE,
    birth_city VARCHAR(100),
    social_security_number VARCHAR(20) UNIQUE
);

CREATE TABLE IF NOT EXISTS address (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    person_id INTEGER,
    address_text TEXT,
    FOREIGN KEY (person_id) REFERENCES person(id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS room (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name VARCHAR(100),
    gender_restriction VARCHAR(10),
    min_age INTEGER,
    max_age INTEGER
);

CREATE TABLE IF NOT EXISTS bed (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    label VARCHAR(100),
    capacity INTEGER CHECK (capacity >= 1 AND capacity <= 2),
    room_id INTEGER,
```

```sql
CREATE TABLE IF NOT EXISTS occupation (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    person_id INTEGER,
    bed_id INTEGER,
    start_date DATE,
    end_date DATE,
    has_left BOOLEAN,
    FOREIGN KEY (person_id) REFERENCES person(id) ON DELETE RESTRICT,
    FOREIGN KEY (bed_id) REFERENCES bed(id) ON DELETE RESTRICT
);
```

• Capture d’écran de l’interface