

Application de Gestion de Maintenance

Rapport technique

Module : Programmation avancée

Enseignant : Cyril Marcq

Équipe : Fatine Kassabi, Othmane Elfarouqy, Hiba Abou-Diab

Université de Lille

Table des matières

1 Présentation du Projet

2 Technologies Utilisées

3 Structure du Projet

3.1	Package model - Les Classes Métier	
3.1.1	Batiment.java	
3.1.2	Technicien.java	
3.1.3	Intervention.java	
3.2	Package dao - Accès aux Données	
3.2.1	DatabaseConnection.java	
3.2.2	BatimentDAO.java	
3.2.3	TechnicienDAO.java	
3.2.4	InterventionDAO.java	
3.3	Package controller - Interface Utilisateur	
3.3.1	MainApp.java	
3.3.2	BatimentController.java	
3.3.3	TechnicienController.java	
3.3.4	InterventionController.java	

4 Fonctionnalités Principales

4.1	Gestion des Techniciens	
4.2	Gestion des Bâtiments	
4.3	Gestion des Interventions	

5 Base de Données

6 Architecture du Projet

7 Points Techniques Clés

7.1	Pattern DAO (Data Access Object)	
7.2	JavaFX	
7.3	Gestion des Erreurs	
7.4	PreparedStatement	

1 Présentation du Projet

Dans le cadre de notre formation en Master 2, nous avons développé une application de gestion de maintenance des infrastructures. Ce projet vise à répondre aux besoins de planification, de suivi et de traçabilité des interventions techniques. L'application permet de gérer trois entités principales : les techniciens, les bâtiments et les interventions. Chaque intervention associe un technicien qualifié à un bâtiment pour une date donnée, avec un suivi de son statut (planifiée, en cours, terminée) et une description détaillée des opérations effectuées. Sur le plan technique, le projet a été réalisé en Java avec une interface graphique JavaFX, une base de données relationnelle MySQL, et un déploiement conteneurisé via Docker. L'ensemble du code source est versionné sur GitHub, facilitant le travail collaboratif au sein de notre équipe. Ce rapport présente l'analyse, la conception, la réalisation technique et le déploiement de l'application, ainsi que les tests effectués pour valider le respect du cahier des charges.

Équipe : Fatine Kassabi, Othmane Elfarougy, Hiba Abou-Diab

2 Technologies Utilisées

- **Langage :** Java 17
- **Interface graphique :** JavaFX 21
- **Base de données :** MySQL 8.0.33
- **Gestionnaire de dépendances :** Maven
- **Conteneurisation :** Docker

3 Structure du Projet

3.1 Package model - Les Classes Métier

3.1.1 Batiment.java

Représente un bâtiment où les interventions ont lieu.

- **Attributs :** id, nom, localisation
- **Méthodes :** Getters/Setters pour accéder et modifier les informations

3.1.2 Technicien.java

Représente un technicien de maintenance.

- **Attributs :** id, nom, qualification, disponibilite
- **Méthodes :** Getters/Setters pour gérer les informations du technicien

3.1.3 Intervention.java

Représente une intervention de maintenance.

- **Attributs :** id, description, dateIntervention, statut, technicienId, batimentId
- **Méthodes :** Getters/Setters pour manipuler les données d'intervention

3.2 Package dao - Accès aux Données

3.2.1 DatabaseConnection.java

Rôle : Établir la connexion avec la base de données MySQL

Fonctions principales :

- `getConnection()` : Crée et retourne une connexion à la base de données

- `closeConnection()` : Ferme proprement la connexion à la base de données

Intérêt : Centraliser la gestion de connexion, éviter la duplication de code

3.2.2 BatimentDAO.java

Rôle : Gérer les opérations CRUD pour les bâtiments

Fonctions principales :

- `getAll()` : Récupère tous les bâtiments de la base de données
- `add(Batiment)` : Ajoute un nouveau bâtiment
- `update(Batiment)` : Modifie un bâtiment existant
- `delete(int id)` : Supprime un bâtiment par son ID
- `getId(int id)` : Récupère un bâtiment spécifique

Intérêt : Séparer la logique d'accès aux données de l'interface utilisateur

3.2.3 TechnicienDAO.java

Rôle : Gérer les opérations CRUD pour les techniciens

Fonctions principales :

- `getAll()` : Récupère tous les techniciens
- `add(Technicien)` : Ajoute un nouveau technicien
- `update(Technicien)` : Modifie les informations d'un technicien
- `delete(int id)` : Supprime un technicien
- `getId(int id)` : Récupère un technicien spécifique

Intérêt : Faciliter la gestion des données des techniciens avec des requêtes SQL préparées

3.2.4 InterventionDAO.java

Rôle : Gérer les opérations CRUD pour les interventions

Fonctions principales :

- `getAll()` : Récupère toutes les interventions
- `add(Intervention)` : Enregistre une nouvelle intervention
- `update(Intervention)` : Modifie une intervention existante
- `delete(int id)` : Supprime une intervention

Intérêt : Gérer les relations entre techniciens, bâtiments et interventions

3.3 Package controller - Interface Utilisateur

Les figures suivantes présentent les trois interfaces principales de l'application développée avec JavaFX.

[illegible]

FIGURE 1 – Interface de gestion des bâtiments

[illegible]

FIGURE 2 – Interface de gestion des techniciens

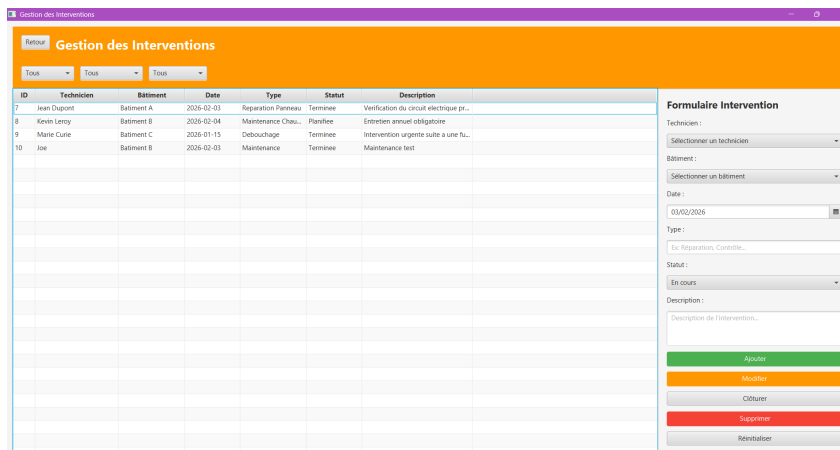


FIGURE 3 – Interface de gestion des interventions

3.3.1 MainApp.java

Rôle : Point d'entrée de l'application et menu principal

Fonctions principales :

- `start(Stage)` : Crée le menu principal avec trois boutons
- `ouvrirTechniciens()` : Ouvre l'interface de gestion des techniciens
- `ouvrirBatiments()` : Ouvre l'interface de gestion des bâtiments
- `ouvrirInterventions()` : Ouvre l'interface de gestion des interventions
- `main(String[])` : Lance l'application JavaFX

Intérêt : Navigation centralisée entre les différents modules de l'application

3.3.2 BatimentController.java

Rôle : Gérer l'interface de gestion des bâtiments

Fonctions principales :

- `createScene(Stage)` : Construit l'interface complète avec tableau et formulaire
- `creerHeader(Stage)` : Crée l'en-tête avec titre et bouton retour
- `creerTableau()` : Crée le tableau affichant la liste des bâtiments
- `creerFormulaire()` : Crée le formulaire de saisie/modification
- `charger()` : Charge les données depuis la base de données
- `ajouter()` : Ajoute un nouveau bâtiment
- `modifier()` : Modifie le bâtiment sélectionné
- `supprimer()` : Supprime le bâtiment sélectionné
- `vider()` : Réinitialise le formulaire
- `alert(String)` : Affiche des messages à l'utilisateur

Intérêt : Interface CRUD complète pour gérer les bâtiments avec validation des données

3.3.3 TechnicienController.java

Rôle : Gérer l'interface de gestion des techniciens

Fonctions principales :

- `createScene(Stage)` : Construit l'interface utilisateur
- `createHeader(Stage)` : Crée l'en-tête de la page
- `createTableView()` : Crée le tableau avec colonnes (ID, Nom, Qualification, Disponibilité)
- `createFormulaire()` : Crée le formulaire avec champs de saisie et boutons d'action
- `chargerDonnees()` : Actualise l'affichage avec les données de la base

- `ajouter()` : Enregistre un nouveau technicien
- `modifier()` : Met à jour les informations d'un technicien
- `supprimer()` : Supprime un technicien avec confirmation
- `reinitialiser()` : Vide les champs du formulaire
- `showAlert(String, String, AlertType)` : Affiche des messages d'information/erreur

Intérêt : Interface complète avec gestion de la disponibilité des techniciens

3.3.4 InterventionController.java

Rôle : Gérer l'interface de planification des interventions

Fonctions principales :

- `createScene(Stage)` : Construit l'interface de gestion des interventions
- `createHeader()` : Crée l'en-tête avec filtres et bouton retour
- `createTableView()` : Affiche les interventions avec leurs détails
- `createFormulaire()` : Formulaire avec sélection de technicien, bâtiment, date et statut
- `chargerTechniciens()` : Charge la liste des techniciens dans le ComboBox
- `chargerBatiments()` : Charge la liste des bâtiments dans le ComboBox
- `chargerDonnees()` : Actualise la liste des interventions
- `ajouter()` : Planifie une nouvelle intervention
- `modifier()` : Modifie une intervention existante
- `supprimer()` : Supprime une intervention

Intérêt : Coordination entre techniciens et bâtiments, suivi du statut des interventions

4 Fonctionnalités Principales

4.1 Gestion des Techniciens

- Ajouter un technicien avec nom, qualification et disponibilité
- Modifier les informations d'un technicien
- Supprimer un technicien
- Consulter la liste complète des techniciens

4.2 Gestion des Bâtiments

- Enregistrer un nouveau bâtiment avec nom et localisation
- Modifier les informations d'un bâtiment
- Supprimer un bâtiment
- Afficher tous les bâtiments

4.3 Gestion des Interventions

- Planifier une intervention en associant technicien et bâtiment
- Définir la date et le statut (Planifiée, En cours, Terminée)
- Modifier ou annuler une intervention
- Filtrer les interventions par technicien ou bâtiment
- Consulter l'historique des interventions

5 Base de Données

Nom : maintenance_db

Tables :

- **techniciens** : Stocke les informations des techniciens
- **batiments** : Contient les bâtiments et leurs localisations
- **interventions** : Enregistre toutes les interventions avec relations vers techniciens et bâtiments

6 Architecture du Projet

```

maintenance-app/
  src/main/java/com/maintenance/
    MainApp.java           (Point d'entr e)
  model/                   (Classes m tier)
    Batiment.java
    Technicien.java
    Intervention.java
  dao/                     (Acc s aux donn es)
    DatabaseConnection.java
    BatimentDAO.java
    TechnicienDAO.java
    InterventionDAO.java
  controller/              (Interface utilisateur)
    BatimentController.java
    TechnicienController.java
    InterventionController.java
  database/                (Scripts SQL)
  docker-compose.yml       (Configuration Docker)
  pom.xml                  (Configuration Maven)
  README.md

```

7 Points Techniques Clés

7.1 Pattern DAO (Data Access Object)

- Séparation claire entre logique métier et accès aux données
- Facilite la maintenance et les tests
- Permet de changer de base de données facilement

7.2 JavaFX

- Interface graphique moderne et responsive
- Composants TableView pour affichage de données
- Formulaires avec validation des champs
- Système de navigation entre les différentes vues

7.3 Gestion des Erreurs

- Try-catch pour les opérations de base de données
- Messages d'alerte pour informer l'utilisateur
- Validation des champs avant insertion en base

7.4 PreparedStatement

- Sécurité contre les injections SQL
- Performances optimisées pour les requêtes répétitives