

Documentation de la Phase 1 du Projet SDIAL

Architecture Logicielle

Exigences : Ressources

Sami Ouraghene

Selyan KABLIA

Nathan LESTRADE

Asma Mansour

04/11/2024

Sommaire

Architecture Logicielle	0
04/11/2024	0
Sommaire	1
Introduction	3
1. Procédure d'installation	4
1.1. Pré-requis	4
Remarques	4
1.2. Installation du code source	5
1.3. Configuration de la base de donnée	5
1.4. Lancement de l'application via l'IDE	6
1.5. Compilation et génération du binaire	6
2. Procédure de désinstallation	7
3. Interfaces des différents composants	7
4. Environnement Matériel Nécessaire	8
4.1 Processeur (CPU)	8
4.2 Mémoire Vive (RAM)	8
4.3 Stockage (Disque)	8
4.4 Autres Éléments Matériels	8
4.5 Environnements Matériels Prévus	9
5. Environnement Logiciel Nécessaire	9
5.1. Système d'Exploitation (OS)	9
5.2. Environnement d'Exécution Java (JRE)	9
5.3. Environnement de Développement Java (JDK)	9
5.4. Environnements de Compilation	9
5.5. Base de Données	10
6. Séquence des Appels de Méthodes pour les Fonctionnalités Principales	11
6.1 Fonctionnalité : Création d'une Réservation (Personne - Salle - Créneau)	11
1. Interface Utilisateur (IHM)	11
2. Gestionnaire d'Événements (ReservationsIHM)	11
3. Service de Réservation (ReservationService)	11
4. Contrôleur de Réservation (ReservationController)	11
5. Confirmation à l'IHM	12
Diagramme de séquence	12
6.2. Fonctionnalité : Suppression d'une Salle	12
Étapes de la Séquence	12
1. Initiation	12
2. Vérification des Réservations Actuelles	12
3. Suppression	12
4. Confirmation	13

Diagramme de Séquence	13
7. Exigences satisfaites / non-satisfaites	14
9. Vue de déploiement	16
12. Liste des anomalies	17
12.1 Exigences non satisfaites	17
12.2 Documentation incomplète :	17

Introduction

Le Projet SDIAL vise à développer un système de gestion des réservations, tels que des salles et des créneaux horaires, destiné à simplifier et optimiser l'organisation des événements. Ce système permettra aux utilisateurs de créer, modifier et supprimer des réservations, tout en assurant une gestion efficace des ressources disponibles.

Cette documentation présente la Phase 1 du projet, au cours de laquelle notre groupe a été assigné à la gestion des exigences fonctionnelles liées aux "ressources". Nous avons pour mission de garantir que toutes les fonctionnalités relatives à la gestion des salles et des créneaux horaires sont correctement implémentées et documentées.

Dans cette phase, nous détaillons les procédures d'installation, de configuration et d'utilisation de l'application. Cela inclut également les exigences matérielles et logicielles nécessaires au bon fonctionnement du système, ainsi qu'une description des interfaces des différents composants de l'application.

L'objectif principal de cette première phase est d'établir une base solide pour le développement futur, en veillant à ce que toutes les fonctionnalités de base soient opérationnelles et bien documentées.

1. Procédure d'installation

1.1. Pré-requis

Java Development Kit (JDK) :

- Assurez-vous que le JDK est installé sur votre machine. Vous pouvez télécharger la dernière version de [Oracle](#) ou utiliser une distribution open source comme [OpenJDK](#).
- Vérifiez que la variable d'environnement `JAVA_HOME` est correctement configurée.

Apache Maven :

- Installez Apache Maven pour gérer les dépendances et la construction du projet. Vous pouvez télécharger Maven depuis le site officiel [Apache Maven](#).
- Assurez-vous que la variable d'environnement `MAVEN_HOME` est configurée et que le dossier `bin` de Maven est inclus dans votre `PATH`.

IDE (Integrated Development Environment) :

- Pour ce projet, il est recommandé d'utiliser Eclipse. Téléchargez et installez Eclipse à partir du site officiel Eclipse IDE.

Serveur de base de données :

- Installez un serveur de base de données, comme MySQL ou MariaDB, pour exécuter la base de données de l'application. Vous pouvez utiliser XAMPP qui inclut phpMyAdmin pour la gestion de la base de données.
- Créez une base de données pour l'application avant de lancer le programme.

Connectivité Internet (si applicable) :

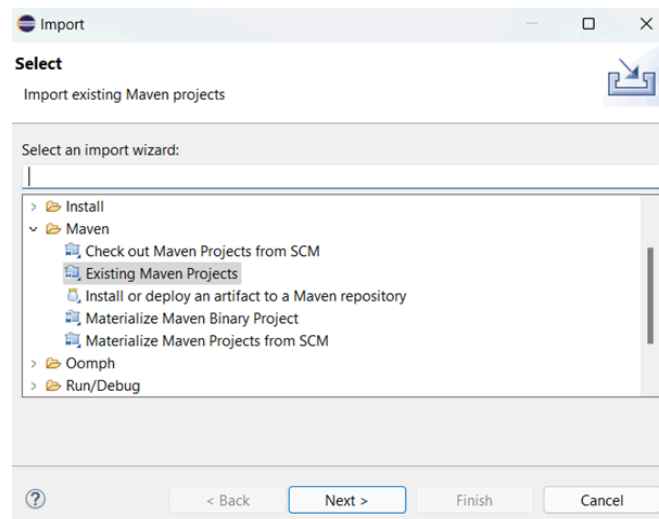
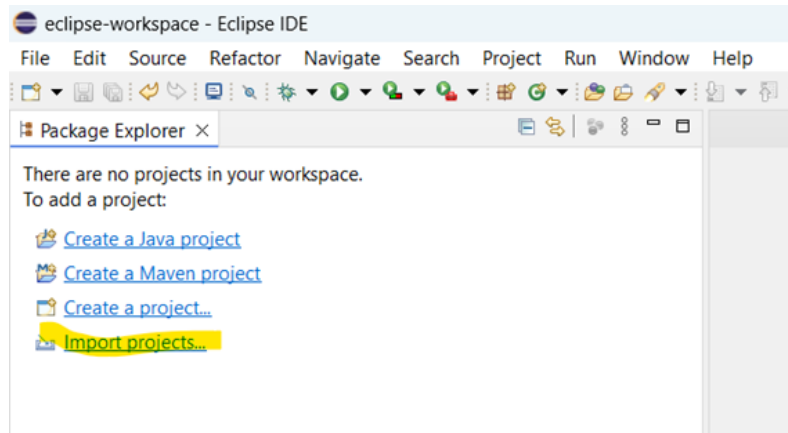
- Une connexion Internet peut être nécessaire pour télécharger des dépendances lors de l'installation de Maven et pour toute mise à jour des outils et bibliothèques.

Remarques

- Vérifiez les versions compatibles de Java et Maven pour éviter les problèmes de compatibilité.
- Assurez-vous que toutes les dépendances sont correctement téléchargées et que l'environnement est configuré avant de lancer l'application.

1.2. Installation du code source

Lancer votre IDE, ici nous utiliserons Eclipse. Ensuite vous devez importer un projet Maven.



Sélectionnez Existing Maven Projects et choisissez le dossier maven « projet-ressources », ensuite importez tous les fichiers qui vous seront proposés. Vous avez une visibilité du code source complet dans votre Package Explorer.

1.3. Configuration de la base de donnée

Avant de faire fonctionner l'application, vous devez configurer une base de donnée local ou hébergée sur un serveur. Dans le cas du développement, nous avons installé la base de donnée en local avec phpMyAdmin et XAMPP.

Il faudra changer donc le lien qui pointe vers votre base de donnée dans le package database du module « reservation-model ».

```
package database;

import java.sql.Connection;

public class Connexion {
    private static final String URL = "jdbc:mysql://localhost:3306/reservations"; // Remplacez par votre
    private static final String USER = "root"; // Remplacez par votre utilisateur
    private static final String PASSWORD = "password"; // Remplacez par votre mot de passe
}
```

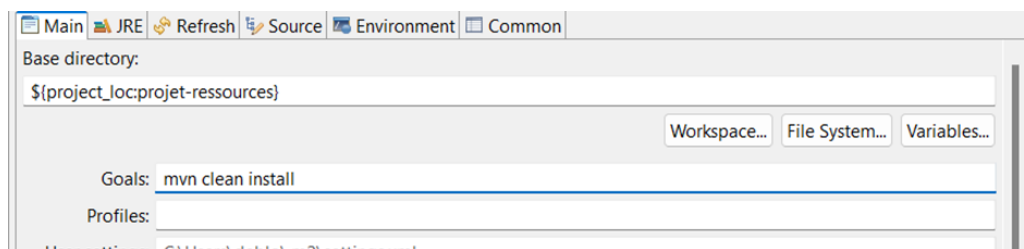
Une fois cela fait, sauvegardez le fichier.

1.4. Lancement de l'application via l'IDE

Une fois la configuration de la base de donnée faite, il faut build le projet afin « d'activer » toutes les dépendances des fichiers pom.xml.

Pour ce faire, faites un clique droit sur le dossier « projet-ressources », ensuite cliquez sur « Run as » puis sélectionnez Maven build.

Ensuite vous devrez entrer « mvn clean install » dans les Goals comme ceci :



Puis cliquez sur Run.

Ensuite, allez dans le package de l'IHM pour lancer l'application. Faites un clique droit sur « Main » qui représente le main, puis « Run as » et cliquez sur Java application.







L'application va se lancer. Si la configuration avec la base de donnée a été bien faite, l'application fonctionnera.

1.5. Compilation et génération du binaire

Pour obtenir le binaire il faut faire un clique droit sur projet-ressources puis « Run as » puis « Maven install ».

Le fichier binaire va se générer. Le chemin d'accès pour le retrouver est : projet-ressources/reservation-ihm/target/Application-jar-with-dependencies.jar

Cliquez deux fois dessus pour lancer l'application.

 archive-tmp	03/11/2024 23:21	Dossier de fichiers	
 classes	03/11/2024 23:21	Dossier de fichiers	
 generated-sources	03/11/2024 23:21	Dossier de fichiers	
 generated-test-sources	03/11/2024 23:21	Dossier de fichiers	
 maven-archiver	03/11/2024 23:21	Dossier de fichiers	
 maven-status	03/11/2024 23:21	Dossier de fichiers	
 test-classes	03/11/2024 23:21	Dossier de fichiers	
 Application	03/11/2024 23:21	Executable Jar File	17 Ko
 Application-jar-with-dependencies	03/11/2024 23:21	Executable Jar File	4 074 Ko

2. Procédure de désinstallation

Pour désinstaller l'application il suffit de supprimer le dossier contenant le fichier binaire ainsi que toutes les dépendances associées.

3. Interfaces des différents composants

Les interfaces **CreneauIHM**, **ReservationsIHM**, **SalleIHM** et **PersonnelIHM** jouent un rôle crucial dans l'interaction utilisateur au sein de l'application.

CreneauIHM permet de gérer les créneaux horaires. Les utilisateurs peuvent saisir l'ID, l'heure de début et de fin d'un créneau. Le bouton "Ajouter Créneau" enregistre le créneau via `CreneauService`, tandis que "Supprimer Créneau" permet de retirer un créneau en utilisant son ID. Des messages d'erreur s'affichent en cas de saisie invalide.

ReservationsIHM `ReservationsIHM` gère les réservations. Les utilisateurs peuvent entrer les identifiants de la réservation, de la personne, de la salle et du créneau. "Ajouter Réservation" crée une nouvelle réservation avec `ReservationService`, "Supprimer Réservation" retire une réservation par son ID, et "Changer Créneau" modifie le créneau associé à une réservation.

SalleIHM `SalleIHM` est dédiée à la gestion des salles. Les utilisateurs saisissent l'ID et le nom de la salle. "Ajouter Salle" envoie les données au service `SalleService` pour créer une nouvelle salle, tandis que "Supprimer Salle" permet de retirer une salle en utilisant son ID. Des messages de confirmation ou d'erreur apparaissent selon la saisie.

PersonnelHM PersonnelHM gère les personnes. Les utilisateurs peuvent entrer l'ID, le nom, le prénom et le type d'une personne. Les boutons "Ajouter Personne" et "Supprimer Personne" permettent d'ajouter ou de retirer une personne par son ID, avec des messages informatifs en cas d'erreur de saisie.

4. Environnement Matériel Nécessaire

Cette section décrit les ressources matérielles nécessaires pour installer et exécuter le système dans de bonnes conditions. Ces spécifications couvrent à la fois les besoins **minimaux** pour un fonctionnement de base et les besoins **recommandés** pour des performances optimales.

4.1 Processeur (CPU)

- **Exigence minimale** : Processeur double cœur de 2 GHz
- **Exigence recommandée** : Processeur quadricœur de 2,5 GHz ou plus

Explication : Un processeur double cœur suffira pour exécuter les opérations basiques, mais un processeur quadricœur est conseillé pour des performances améliorées, en particulier si des opérations multiples sont attendues.

4.2. Mémoire Vive (RAM)

- **Exigence minimale** : 4 Go de RAM
- **Exigence recommandée** : 8 Go de RAM ou plus

Explication : Avec 4 Go, le système pourra s'exécuter de manière basique. Cependant, 8 Go ou plus permettent une utilisation fluide, surtout si plusieurs sessions ou utilisateurs sont connectés.

4.3. Stockage (Disque)

- **Exigence minimale** : 500 Mo d'espace libre pour l'installation de l'application
- **Exigence recommandée** : 2 Go d'espace libre pour une meilleure performance (permet aussi le stockage de logs et autres données temporaires)

Explication : 500 Mo sont nécessaires pour les fichiers de base, mais prévoir 2 Go permet de conserver les fichiers temporaires et de gérer l'espace requis pour d'éventuelles mises à jour.

4.4. Autres Éléments Matériels

- **Accès au réseau** : Connexion réseau stable pour la communication avec la base de données.
- **Périphériques** : Aucun matériel spécifique requis pour le fonctionnement, à l'exception d'un clavier et d'une souris standard pour la navigation.

4.5. Environnements Matériels Prévus

Le système est conçu pour fonctionner sur les plateformes suivantes :

- **Machines physiques** : Toute machine personnelle ou serveur répondant aux exigences ci-dessus.
- **Machines virtuelles** : Le système peut fonctionner dans un environnement virtualisé, tel que VirtualBox, VMware, ou un environnement cloud (AWS, Azure) avec les spécifications minimales ci-dessus.

5. Environnement Logiciel Nécessaire

5.1. Système d'Exploitation (OS)

- **Exigence** : Windows 10 ou plus récent (Windows 11 recommandé)

Explication : Le système est conçu pour fonctionner uniquement sous Windows, conformément aux exigences du cahier des charges.

5.2. Environnement d'Exécution Java (JRE)

- **Exigence** : Java Runtime Environment (JRE) 21 ou plus récent

Explication : La version Java 21 est requise pour assurer la compatibilité avec le code de l'application. Il est possible d'installer le JRE 21 seul si aucune modification de code n'est nécessaire.

5.3. Environnement de Développement Java (JDK)

- **Exigence** : Java Development Kit (JDK) 21 ou plus récent

Explication : Si des modifications ou une compilation du code source sont nécessaires, le JDK 21 doit être installé, car il fournit les outils de développement Java et la version minimale compatible pour le projet.

5.4. Environnements de Compilation

Pour construire la version binaire, l'un des environnements de développement suivants doit être installé :

- **Eclipse IDE** (version 2023-06 ou plus récent)
- **Apache Maven** (version 3.9 ou plus récent)
- **Gradle** (version 8.0 ou plus récent)

Explication : Le système est compatible avec Eclipse, Maven, ou Gradle pour la génération de la version binaire. Le choix de l'environnement dépend de vos préférences ou de la configuration existante de votre système.

5.5. Base de Données

- **Exigence** : Base de données relationnelle compatible avec JDBC, telle que MySQL, installée et configurée séparément.

Explication : Nous utilisons MySQL en local avec XAMPP et phpMyAdmin mais chacun utilise l'application qu'il souhaite.

5.6. Autres Dépendances et Outils Recommandés

- **Connexion réseau** : Une connexion réseau stable est recommandée pour accéder à la base de données et pour les opérations nécessitant des mises à jour ou une synchronisation de données.
- **Logiciel de gestion de version** : Git (facultatif, mais recommandé pour le suivi et la gestion du code source).

6. Séquence des Appels de Méthodes pour les Fonctionnalités Principales

6.1 Fonctionnalité : Création d'une Réservation (Personne - Salle - Créneau)

Cette fonctionnalité permet d'ajouter une nouvelle réservation en spécifiant une personne, une salle, et un créneau horaire.

Séquence des appels de méthodes

1. Interface Utilisateur (IHM)

- L'utilisateur saisit les informations de la réservation (ID de la réservation, ID de la personne, ID de la salle, et ID du créneau) dans l'interface graphique.
- Lors de la soumission, un clic sur le bouton **Ajouter Réservation** déclenche un événement.

2. Gestionnaire d'Événements (ReservationsIHM)

- La méthode **addActionListener** de **ajouterButton** est appelée, et elle récupère les informations fournies par l'utilisateur.
- Les champs de texte sont validés (conversion en entiers pour éviter les erreurs de type).
- Une fois les valeurs validées, la méthode **ajouterReservation** du service **ReservationService** est appelée avec les paramètres suivants : **id**, **idPersonne**, **idSalle**, et **idCreneau**.

3. Service de Réservation (ReservationService)

- La méthode **ajouterReservation** est définie dans l'interface **ReservationService** et implémentée dans la classe **ReservationController**.
- **ajouterReservation** reçoit les informations et crée une nouvelle réservation dans la base de données via la couche d'accès aux données.

4. Contrôleur de Réservation (ReservationController)

- Dans **ReservationController**, la méthode **ajouterReservation** exécute la requête SQL d'insertion :

```
INSERT INTO reservation (id, id_personne, id_salle, id_creneau) VALUES (?, ?, ?, ?)
```

- Les paramètres **id**, **idPersonne**, **idSalle**, et **idCreneau** sont définis dans la requête à l'aide de **PreparedStatement**.
- La requête est ensuite exécutée, et si l'insertion est réussie, un message de confirmation s'affiche dans la console.

5. Confirmation à l'IHM

- Après l'insertion, le contrôle revient à `ReservationsIHM`.
- Une fenêtre de dialogue (`JOptionPane`) notifie l'utilisateur que la réservation a été ajoutée avec succès.

Diagramme de séquence

Ce diagramme peut illustrer visuellement l'interaction entre les classes `ReservationsIHM`, `ReservationService`, et `ReservationController` :

```
Utilisateur -> ReservationsIHM -> ReservationService -> ReservationController -> Base de données
```

6.2. Fonctionnalité : Suppression d'une Salle

Objectif : Cette fonctionnalité permet à un utilisateur de supprimer une salle du système.

Étapes de la Séquence

1. Initiation

- L'utilisateur sélectionne une salle à supprimer en entrant l'ID de la salle dans le champ de saisie prévu à cet effet dans l'interface utilisateur (IHM).
- L'utilisateur déclenche la suppression en cliquant sur le bouton `Supprimer Salle`.

2. Vérification des Réservations Actuelles

- Avant de procéder à la suppression, le système doit vérifier si la salle a des réservations en cours.
- Cela peut être réalisé en implémentant une méthode dans le `SalleController` pour obtenir les réservations associées à la salle. Par exemple, on pourrait ajouter une méthode `getReservationsBySalleId(int id)` qui interroge la base de données pour vérifier les réservations.
- Si des réservations sont trouvées, un message d'erreur est affiché à l'utilisateur pour l'informer que la salle ne peut pas être supprimée tant qu'elle a des réservations actives.

3. Suppression

- Si aucune réservation n'est associée à la salle, le système procède à la suppression de celle-ci.
- La méthode `supprimerSalle(int id)` de `SalleController` est appelée avec l'ID de la salle à supprimer.
- Dans cette méthode, une requête SQL est exécutée pour supprimer la salle :

```
DELETE FROM salle WHERE id = ?
```

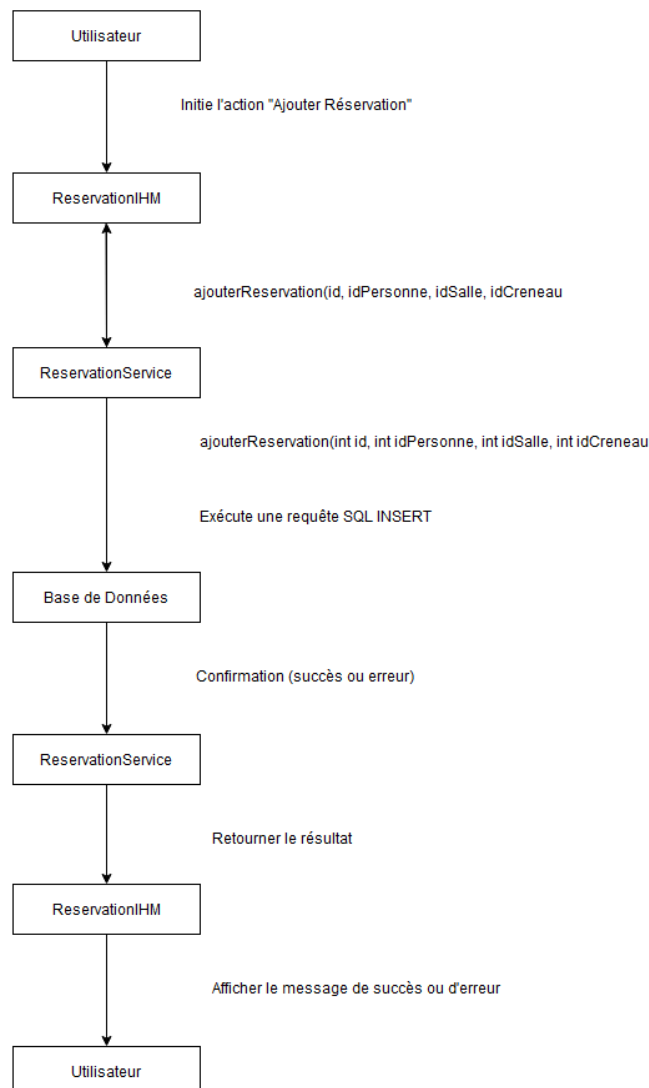
- Si la salle est supprimée avec succès, un message de confirmation est affiché dans l'IHM. Sinon, un message indique qu'aucune salle n'a été trouvée avec l'ID spécifié.

4. Confirmation

- Une fois la salle supprimée ou si l'utilisateur reçoit un message d'erreur, une confirmation ou un message d'erreur est renvoyé à l'utilisateur via une boîte de dialogue.
- La méthode `JOptionPane.showMessageDialog` est utilisée pour afficher ces messages à l'utilisateur, que ce soit pour confirmer la suppression ou pour signaler une erreur.

Diagramme de Séquence

Voici un diagramme simplifié des interactions entre les composants :



7. Exigences satisfaites / non-satisfaites

Numéro	Exigence	Statut
Req-ses-01	L'utilisateur peut créer une salle.	Satisfait
Req-ses-02	L'utilisateur peut supprimer une salle.	Satisfait
Req-ses-03	L'utilisateur peut créer une personne.	Satisfait
Req-ses-04	L'utilisateur peut supprimer une personne.	Satisfait
Req-ses-05	L'utilisateur peut créer un créneau (temporel).	Satisfait
Req-ses-06	L'utilisateur peut supprimer un créneau (temporel).	Satisfait
Req-ses-07	L'utilisateur peut créer une réservation (personne – salle - créneau).	Satisfait
Req-ses-08	L'utilisateur peut supprimer une réservation (personne – salle – créneau)	Satisfait
Req-ses-09	L'utilisateur peut changer une réservation de créneau temporel.	Satisfait
Req-ses-10	L'utilisateur peut créer des réservations sur un ensemble de créneaux.	Non Satisfait
Req-doc-01	La documentation doit décrire la procédure d'installation.	Satisfait
Req-doc-02	La documentation doit décrire la procédure de désinstallation.	Satisfait
Req-doc-03	La documentation doit décrire les interfaces de chaque composant.	Satisfait
Req-doc-04	La documentation doit décrire l'environnement matériel nécessaire à l'installation.	Satisfait
Req-doc-05	La documentation doit décrire l'environnement logiciel nécessaire à l'installation.	Satisfait
Req-doc-06	La documentation doit décrire la séquence des appels de méthodes pour 2 fonctionnalités.	Satisfait
Req-doc-07	La documentation doit comporter une vue d'exécution conforme à ce qui a été fait en cours.	Non Satisfait
Req-doc-08	La documentation doit comporter une vue de développement conforme à ce qui a été fait en cours.	Non Satisfait
Req-doc-09	La documentation doit comporter une vue de déploiement conforme à ce qui a été fait en cours.	Satisfait
Req-doc-10	La documentation doit lister les exigences satisfaites.	Satisfait

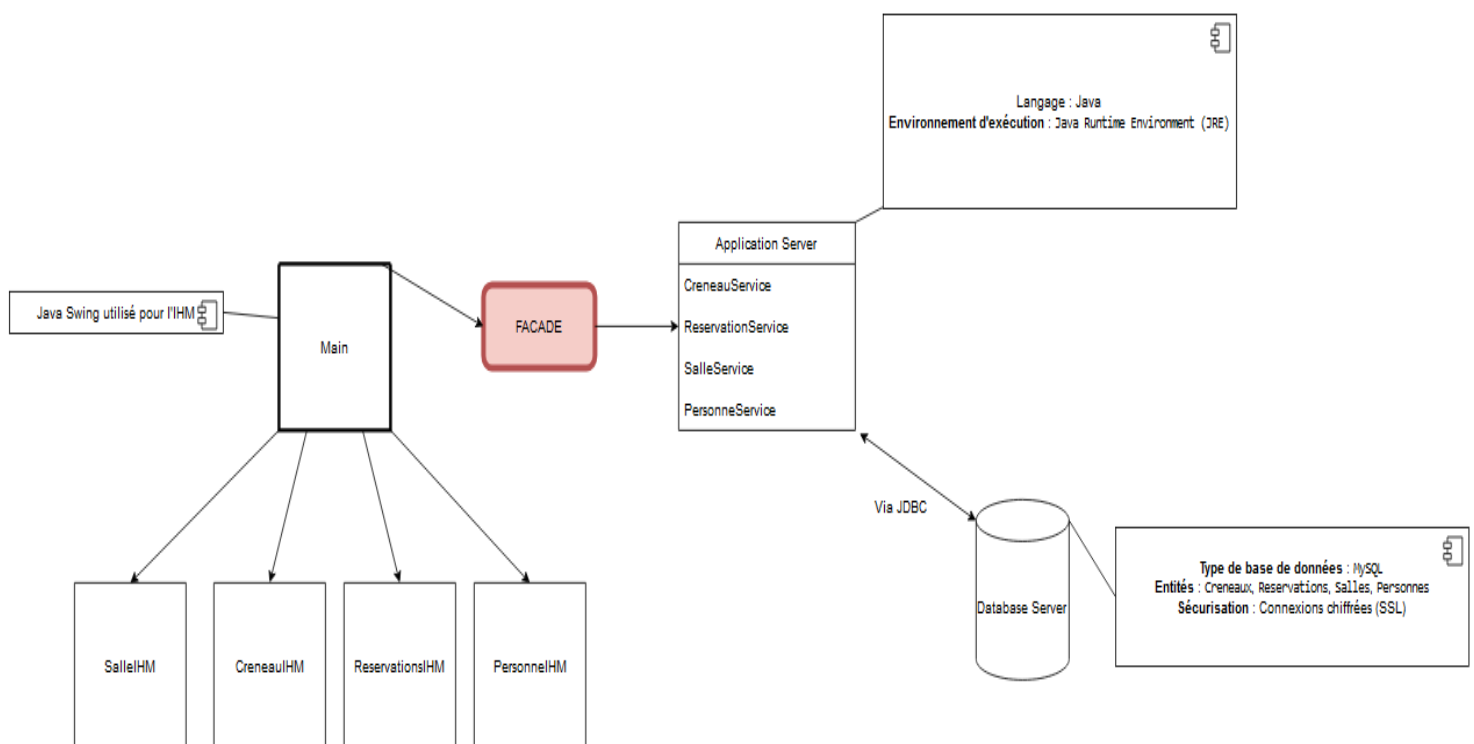
Req-doc-11	La documentation doit lister les exigences non satisfaites.	Satisfait
Req-doc-12	La documentation doit lister les anomalies.	Satisfait
Req-doc-13	La documentation doit décrire la procédure de compilation.	Satisfait
Req-liv-01	La livraison doit contenir tous les éléments nécessaires à la génération de la version binaire qui ne peuvent pas être trouvés ailleurs.	Satisfait
Req-liv-02	La livraison doit contenir toute la documentation, y compris la présentation réalisée en fin de phase.	Satisfait
Req-arc-01	Le système est basé sur un style client-serveur, dont la base de données est fournie par ailleurs.	Satisfait
Req-arc-02	Le programme principal instancie deux objets : l'un implémentant l'IHM et l'autre exposant les fonctionnalités.	Satisfait
Req-arc-03	Les communications entre IHM et fonctions passent par une ou des interfaces Java.	Satisfait
Req-arc-04	Les opérations exposées par l'interface sont de 4 types : lecture, création, modification, suppression d'un objet.	Satisfait
Req-arc-05	Les classes implémentant l'IHM, les fonctionnalités (modèle métier) et l'application (main) sont packagées dans trois jar distincts. Un quatrième peut au besoin être créé pour les interfaces	Satisfait
Req-arc-06	L'IHM peut être mise à jour (automatiquement ou à la demande de l'utilisateur)	Non Satisfait
Req-tec-01	Le système doit fonctionner sous Windows.	Satisfait
Req-tec-02	Le système doit être développé en Java 21.	Satisfait
Req-tec-03	La version binaire doit pouvoir être construite avec au moins un des trois environnements suivants : Eclipse, Maven, Gradle	Satisfait

9. Vue de déploiement

La vue de déploiement illustre l'architecture physique de l'application, mettant en avant les relations entre les composants principaux : le serveur d'application, le serveur de base de données, et les clients. Chaque service est déployé sur des nœuds distincts, facilitant la communication via des protocoles sécurisés.

Le serveur d'application, développé en Java, héberge les services métiers, tandis que la base de données MySQL gère les données. Les connexions entre les composants sont sécurisées par SSL, garantissant la confidentialité des échanges.

Ce diagramme offre une compréhension claire de l'architecture de déploiement de notre projet.



12. Liste des anomalies

Aucune anomalie détectée en exécutant l'application sur l'environnement logiciel et matériel requis.

Voici les exigences non satisfaites :

12.1 Exigences non satisfaites

- **Req-ses-10** : L'utilisateur ne peut pas créer des réservations sur un ensemble de créneaux, ce qui limite la flexibilité dans la gestion des réservations.

12.2 Documentation incomplète :

- **Req-doc-07** : La documentation ne comporte pas de vue d'exécution conforme à ce qui a été fait en cours, ce qui peut entraîner une compréhension incomplète du fonctionnement du système par les utilisateurs.
- **Req-doc-08** : La documentation ne présente pas de vue de développement conforme à ce qui a été fait en cours, laissant ainsi un manque de clarté sur l'architecture du projet.
- **Req-doc-12** : La documentation ne liste pas les anomalies, ce qui pourrait entraîner des malentendus ou des attentes non satisfaites chez les utilisateurs.

Conclusion

La Phase 1 du Projet SDIAL a permis d'établir les bases solides pour le système de gestion des réservations, avec un accent particulier sur la gestion des ressources. Nous avons réussi à définir et à documenter les procédures d'installation, de configuration et d'utilisation de l'application, tout en garantissant que les exigences fonctionnelles essentielles soient satisfaites.

L'intégration des interfaces utilisateur pour la gestion des salles, créneaux, et réservations facilite l'interaction et améliore l'expérience utilisateur. De plus, les exigences matérielles et logicielles spécifiées garantiront un fonctionnement optimal du système.

Nous avons identifié certaines exigences encore non satisfaites et des domaines nécessitant une documentation plus complète, ce qui sera pris en compte lors des phases suivantes du projet. Cette première étape jette les bases pour un développement continu, et nous sommes confiants que le système évoluera pour répondre aux besoins croissants de gestion des réservations.

Nous remercions tous les membres de l'équipe pour leurs contributions et leur engagement envers le succès du projet.