TP de groupe JAVA SPRING BOOT

Alvin Carneiro
Randy Bourdon

14 janvier 2024

1 Répartition des tâches

Tâche	Qui
Spécification	Ensemble
Planning prévisionnel	Alvin
Diagramme Use Case	Randy
Maquette graphique	Randy
Tableau de toutes les routes	Alvin
Diagramme merise	Alvin
Diagramme UML (classe et séquence)	Randy
Mode d'emploi	Ensemble
Bilan	Ensemble
Code API	Alvin
Code application client	Randy

Table 1 – Tableau listant la répartion des tâches.

2 Spécification

L'application vise à améliorer la gestion des réservations de salles dans l'établissement, offrant une solution informatique pour simplifier ce processus complexe. Les utilisateurs cibles comprennent les Responsables de Gestion (RG), les Responsables Pédagogiques de Formation (RF), les Professeurs/Intervenants, les Étudiants, et le Responsable de Maintenance.

Afin de gérer les salles, elles seront caractérisées par un numéro, un nom, le nombre de places, et un type par exemple : ordinaire, munie d'ordinateurs, multimédia, langue, amphi,.. Les RG pourront ajouter, modifier, et supprimer des salles et les types de salle seront gérés de manière similaire.

Pour la planification des réservations, il sera possible de le faire pour la semaine courante ainsi que pour la semaine suivante, des réservations pourront être récurrentes, comme un cours hebdomadaire fixe. C'est les RG qui réaliseront le planning pour les semaines à venir, et seront capables de modifier la semaine courante avant l'expiration de la date.

Les cours ou formations sont caractérisés par un nom, des dates de début et de fin, un formateur, et une classe. Les classes sont définies par leur nom et le nombre d'étudiants, les RF informeront les RG des cours via un moyen externe à l'outil.

Concernant l'accès des utilisateurs, les RG auront un accès complet pour créer, modifier et supprimer des informations, ainsi que pour planifier. Les RF auront quant à eux un accès en consultation et pourront informer les RG des cours. Les professeurs/intervenants auront un accès en consultation avec la possibilité de faire des remarques sur les problèmes rencontrés. Les étudiants pourront consulter les informations pour la semaine courante. Le responsable de maintenance pourra consulter les problèmes signalés et les plages horaires non occupées.

Chaque professeur ou intervenants pourront signaler des problèmes dans les salles et le responsable de maintenance peut consulter la liste des problèmes, les modifier (initial, en cours, terminé), et voir les moments où les salles ne sont pas occupées.

Pour la gestion des utilisateurs, le responsable global pourra ajouter des utilisateurs (étudiants, RG, RF, formateur, resp maintenance), et il sera possible d'ajouter massivement des utilisateurs via l'importation d'un fichier (xls, xml).

Et enfin, pour l'automatisation, le décalage de la semaine suivante vers la semaine courante se fera automatiquement le lundi à 0 heure.

Les objectifs de l'application : - Faciliter la gestion des réservations de salles. - Permettre la planification des cours et des formations. - Améliorer la communication entre les différents acteurs. - Optimiser l'utilisation des salles et résoudre rapidement les problèmes signalés. - Offrir une interface conviviale pour différents niveaux d'utilisateurs.

Les principaux utilisateurs : - Responsables de Gestion (RG) - Responsables Pédagogiques de Formation (RF) - Professeurs/Intervenants - Étudiants - Responsable de Maintenance

Les fonctionnalités principales : - Gestion des Salles - Planification des Réservations - Gestion des Cours et Formations - Accès Utilisateur différencié - Gestion des Problèmes Signalés - Gestion des Utilisateurs - Automatisation du décalage hebdomadaire

L'application vise à rationaliser les processus liés à la réservation de salles, à améliorer l'efficacité opérationnelle et à fournir une plateforme centralisée pour la gestion transparente de ces activités au sein de l'établissement.

3 Diagramme UseCase

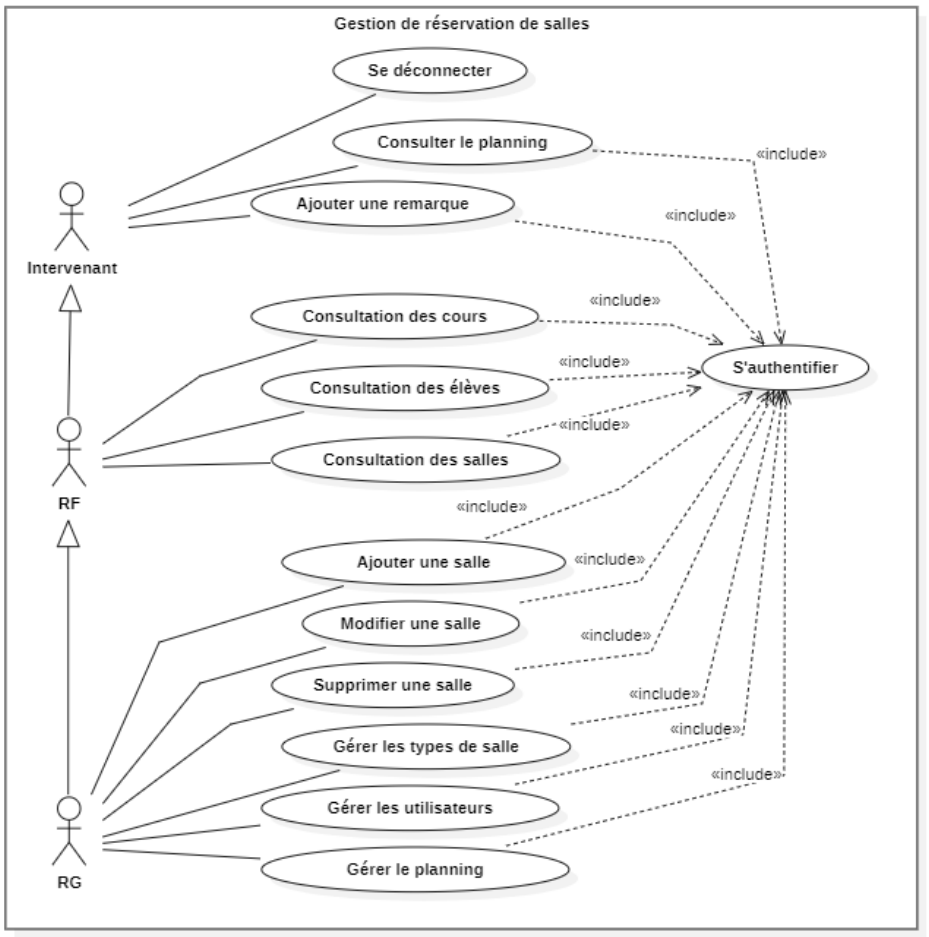

Figure 1 – UseCase.

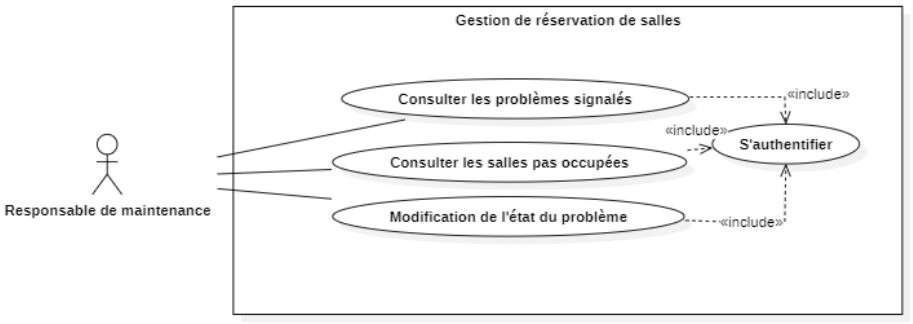

FIGURE 2 – UseCase.

4 Maquette graphique

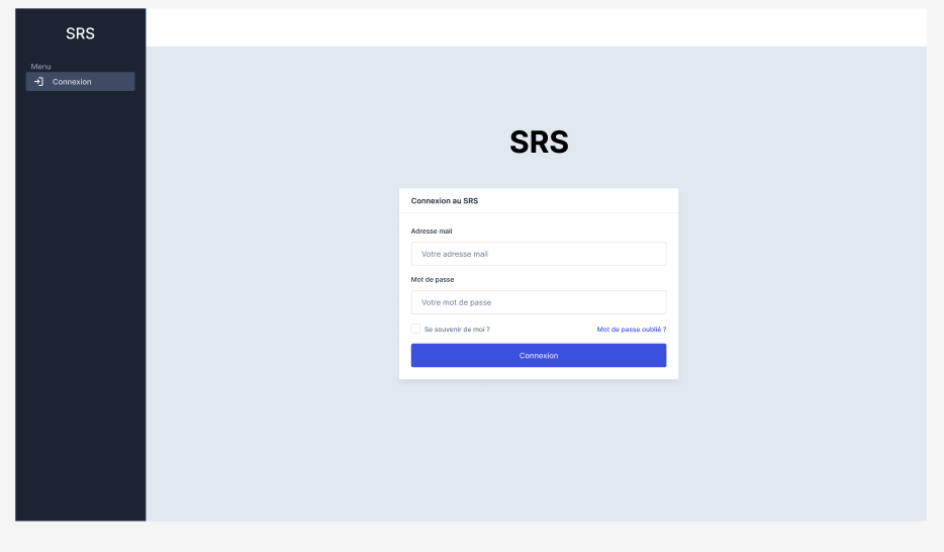

FIGURE 3 – Maquette graphique de connexion.

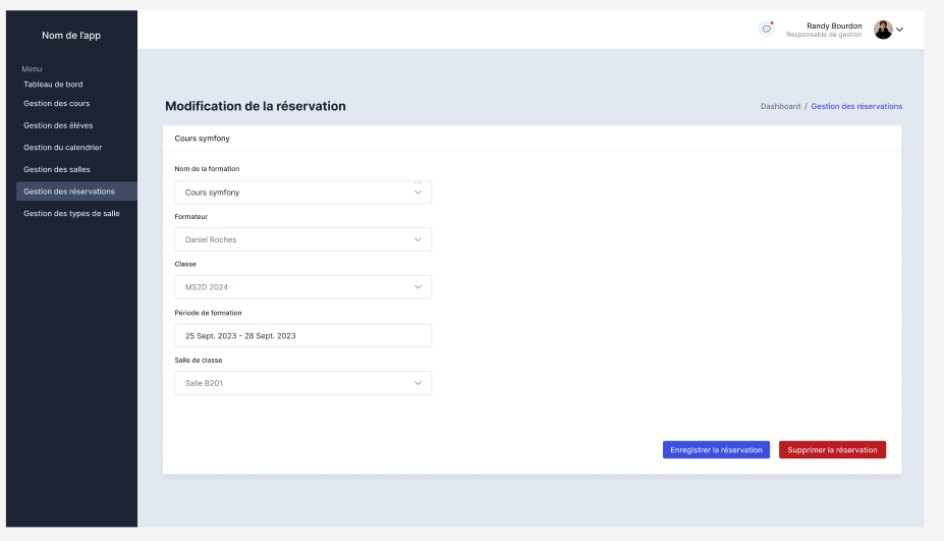

FIGURE 4 – Maquette graphique modification d'une réservation.

5 Tableau de l'ensemble des route de l'api

Chaque route commence par http ://localhost :8080/api/v1/...
(Voir Table 2)

Méthode	Endpoint	Usage	Rôle
GET	/salles/	Consulter toutes les salles	RG, RF
GET	/salles/consulter/id	Consulter les information d'une salle	RG, RF
POST	/salles/creer	Créer une salle	RG
PUT	/salles/modifier/id	Modifier une salle	RG
DELETE	/salles/supprimer/id	Supprimer une salle	RG
GET	/type-salle/	Consulter tous les types de salle	RG, RF
GET	/type-salle/consulter/id	Consulter le type d'une salle	RG
POST	/type-salle/creer	Créer un nouveau type de salle	RG
PUT	/type-salle/modifier/id	Modifier un type de salle	RG
DELETE	/type-salle/supprimer/id	Supprimer un type de salle	RG
GET	/reservations/	Consulter toutes les reservations	RG, RF
GET	/reservations/consulter/id	Consulter une réservation	RG, RF, Etudiant, Formateur
POST	/reservations/creer	Créer une nouvelle réservation	RG
PUT	/reservations/modifier/id	Modifier une réservation	RG
DELETE	/reservations/supprimer/id	Supprimer une réservation	RG
GET	/cours/	Consulter toutes les formations	RG, RF
GET	/cours/consulter/id	Consulter une formation	RG, RF
POST	/cours/creer	Créer une nouvelle formation	RG
PUT	/cours/modifier/id	Modifier une formation	RG
DELETE	/cours/supprimer/id	Supprimer une formation	RG
GET	/classes	Consulter toutes les classes	RG, RF
GET	/classes/consulter/id	Consulter une classe	RG, RF, Etudiant
POST	/classes/creer	Créer une nouvelle classe	RG
PUT	/classes/modifier/id	Modifier une classe	RG
DELETE	/classe/supprimer/id	Supprimer une classe	RG
GET	/utilisateurs/	Consulter tous les utilisateurs	RG, RF
GET	/utilisateur/consulter/id	Consulter un utilisateur	RG, RF
POST	auth/register	Créer un nouvel utilisateur	All
POST	auth/authenticate	Connexion	All
PUT	/utilisateur/modifier/id	Modifier un utilisateur	RG
DELETE	/utilisateur/supprimer/id	Supprimer un utilisateur	RG

Table 2 – Tableau listant les différentes routes/endpoint de l'api.

6 Diagramme Merise

6.1 MCD

(Voir Figure 3)

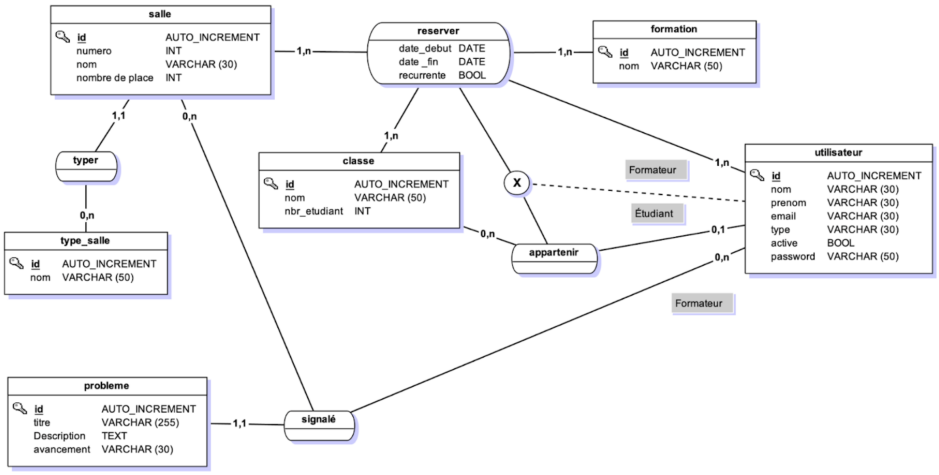

FIGURE 5 – MCD généré par JMerise.

6.2 MLD

(Voir Figure 4)

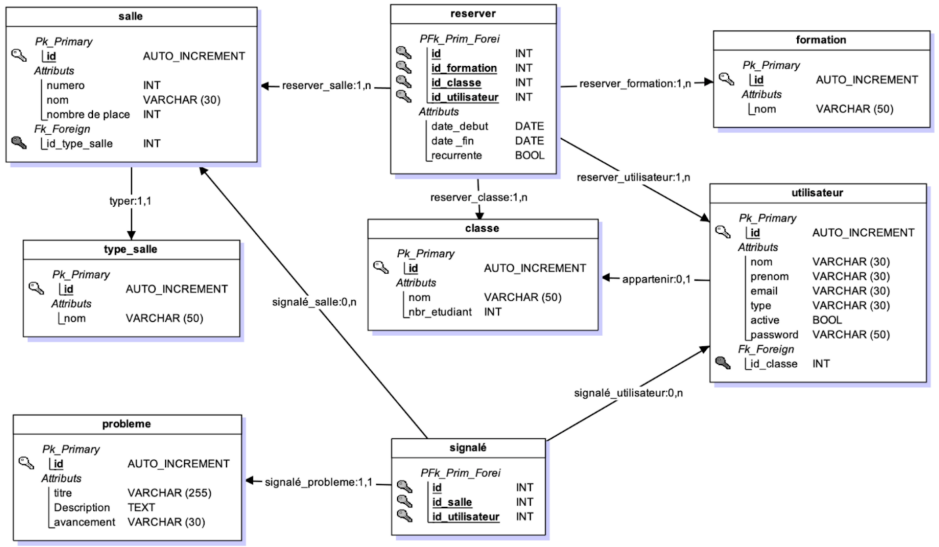

FIGURE 6 – MLD généré par JMerise.

7 Diagramme UML de classe et de séquence

7.1 Diagramme de classe

(Voir Figure 5)

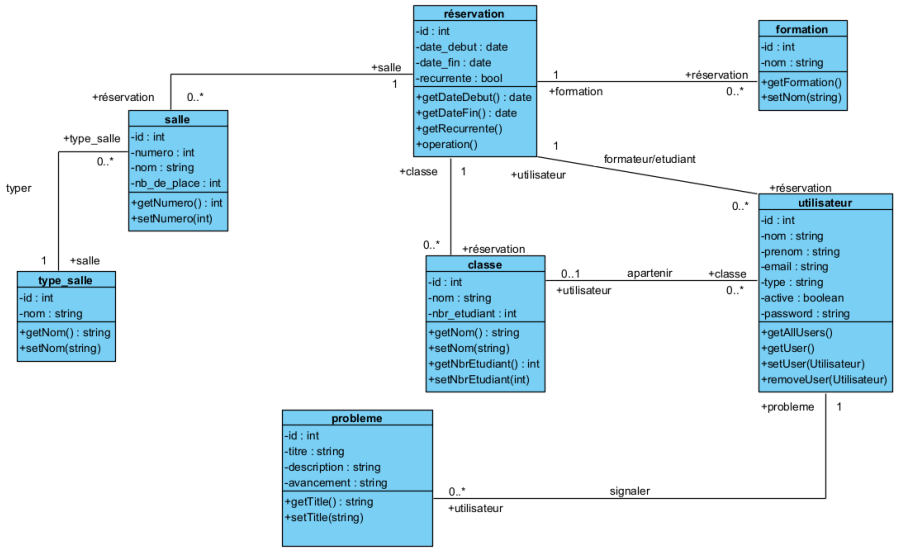

FIGURE 7 – Diagramme de classe.

7.2 Diagramme de séquence

(Voir Figure 6)

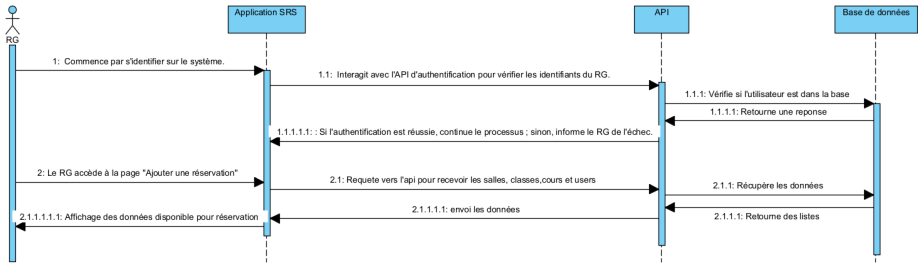

FIGURE 8 – Diagramme de séquence.

8 Diagramme de Gantt

(Voir Figure 7)

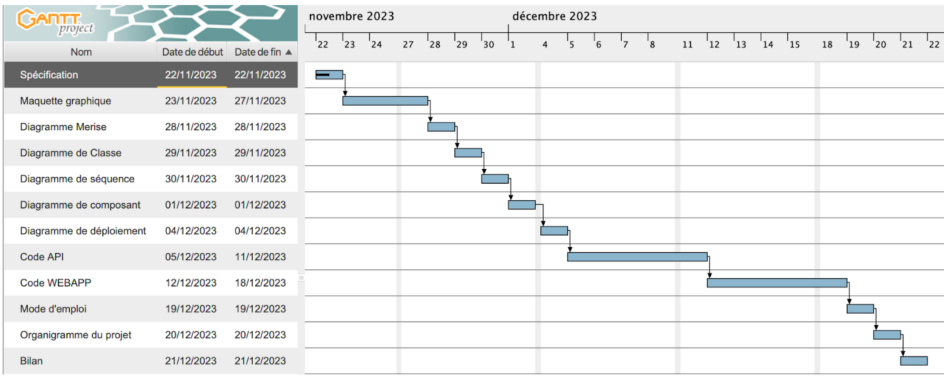

FIGURE 9 – Diagramme de Gantt.

9 Mode d'emploi et d'initialisation de l'application

9.1 Initialiser l'API

1. Implémenter la base MySQL sous un outils tel que PHPmyAdmin
2. Ouvrir l'API dans l'IDE NetBeans
3. S'assurer que NetBeans utilise JDK 17
4. Run l'api en faisant clique droit sur le fichier ApiApplication.java (dans com.srs.api) puis Run File (et non pas via le bouton run de l'IDE NetBeans cela ne fonctionnera pas)
5. L'API est normalement lancé sur le port 8080 (il doit donc être libre)

9.2 Initialiser l'application client

1. Avoir NodeJS sur son PC (dernière version)
2. Ouvrir l'invite de commandes et accéder à la racine du projet
3. Installer les dépendances avec npm install
4. Exécuter le projet avec la commande npm run dev
5. Accéder au site local via http ://localhost :5173/

10 Bilan

L'api prend en charge toutes les routes prévues. Niveau difficulté, ça a été principalement d'implémenter le JWT car j'ai dû tout reprendre de A à Z et c'était très différent de ce qu'on a vu en formation.

L'application contient une page de login, un calendrier récapitulatif, une page de gestion des réservation de salle et une page de gestion des formations. La communication entre l'application et l'API était parfois compliqué notamment quand il a fallut configurer les CORS Policy nécessaire pour que javascript puisse faire des requêtes XMLHttp.