Rapport Projet JEE

ING2 GSI 3
Professeur encadrant: Haddache Mohamed

Léon Mérino Hugo
Xu Hénoch
Liang Yixin
Hagard Loric
Mirzica–Vigé Lucas

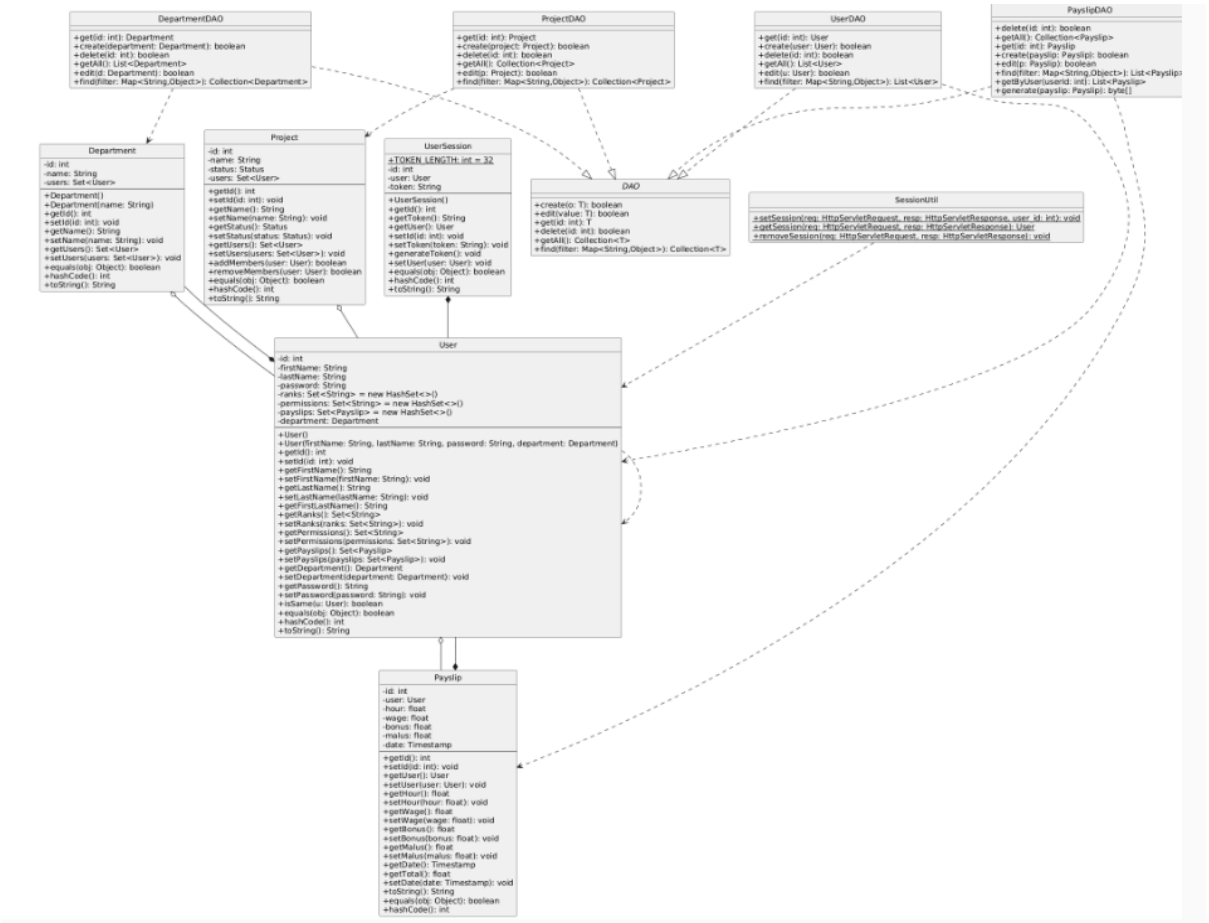
20/11/2025

Sommaire

Introduction

Ceci est un rapport du projet JEE avec l'un qui utilise Jakarta et Tomcat, et l'autre qui utilise SpringBoot et Angular.

Notre UML de base (incluant les DAO pour les requêtes):



Partie Jakarta et Tomcat

Structure

Nous avons conçu le projet à partir des MCDs qu'on a conçu, on a fait des modèles pour employé, département, projet, fiche de paie. Les tables intermédiaires sont gérées par Hibernate, comme la relation employé - département.

Nous avons utilisé pour le modèle MVC, des fichiers JSP, qui interagissent avec le code Java.

Tout d'abord dans le dossier main, il y a le dossier ressource contenant la configuration pour Hibernate et le dossier modèle qui contient les fichiers de définition des entités pour la base de données.

Ensuite, le dossier webapp qui contient les fichiers jsp séparés en dossiers distincts. Il y a le dossier auth pour l'authentification des employés, le dossier components contenant des jsp réutilisés par d'autres jsp. Les dossiers département, employee, payslip, project permettent l'affichage et la modification des entités. Le dossier payslip permet de modifier les feuilles de paie et le dossier report pour les rapports de l'utilisation de site. Le dossier ressources qui contient le css (le JS si on en met) et le dossier WEB-INF qui contient le fichier de configuration des jsp ainsi que les jsp en cas d'erreur (erreur 404 par exemple).

Puis le dossier Java qui contient le fichier HibernateSetup qui permet de remplir la base de données à chaque démarrage, le fichier HibernateUtil qui contient des fonctions facilitant les interactions à la base de données. Le fichier JspFilter qui permet d'empêcher l'utilisateur de voir les fichiers jsp directement, et le fichier SessionUtil qui contient des fonctions facilitant les interactions avec la session.

Enfin, le dossier beans qui contient les définitions des entités ainsi que les énumérations qu'elles utilisent dans le sous-dossier enums. Le dossier dao qui fait le lien entre les contrôleurs et la base de données. Et finalement le dossier contrôleur qui contient les contrôleurs utilisés pour récupérer les requêtes venant de jsp, eux aussi séparés en sous-dossier pour chaque entité.

Fonctionnement

Tout commence sur la page principale, qui redirige vers les autres pages. Cela est configuré avec webapp/WEB-INF/web.xml. Ensuite, chaque lien redirige vers un fichier JSP nommé index dans les dossiers correspondant, qui eux même redirige vers des pages plus spécifiques (comme la modification). Chacun de ces JSP est lié à son contrôleur, qui gère les appels et modifie les donnés en accord avec l'action, et la validité des données.

Partie Spring Boot et Angular

Structure

Spring Boot récupère des requêtes HTML, il renvoie les résultats en JSON, que Angular lit et affiche. Pour la partie conception, on a récupéré les mêmes classes que Jakarta, avec des contraintes en plus (pas de fichiers hbm.xml).

La structure des fichiers de ce projet est répartie en 2 gros dossiers: “src” qui est le backend avec Spring Boot, et le dossier “front” contenant Angular, avec la racine du dossier du projet contenant les fichiers de configurations des 2 côtés.

La structure de côté Spring Boot se décompose en 3 dossiers, et les fichiers de démarrage de Spring Boot qui sont préfixés par App. Le dossier contrôleur contient tous les contrôleurs qui écoutent les requêtes http, avec le sous-dossier dto pour les request-body. Le dossier data qui est séparé lui-même avec les 4 classes principales précédemment citées. Chaque dossier contient la classe entity, une classe repository pour communiquer avec la base de données, et un fichier service pour faire la liaison entre le repository et le reste du projet. Certains dossiers contiennent aussi les fichiers de déclarations des enums utilisés dans l’entité. Et finalement le dossier utils compte les fichiers permettant de faire la transformation en JSON, des réponses des contrôleurs.

Côté frontend, le dossier environnement qui contient les variables environnement, notamment l’URL de l’API et son port. Le dossier modèle contient des fichiers qui définissent les interfaces et les types utilisés. Le dossier service contient les différents services utilisés à travers les composants, notamment le service API qui permet de faire les requêtes au backend. Le dossier shared qui contient les composants qui n’ont pas de route associée, comme le composant header, footer et le dialogue. Finalement, le dossier app qui contient tous les composants visibles du site restant. Ils sont eux aussi séparés selon les différentes entités, avec l'addition de auth pour l’authentification des employés, main pour la page principale, et report pour les rapports de l’utilisation de site. Les dossiers des entités contiennent des sous-composants, notamment les formulaires qui apparaissent en pop-up.

Fonctionnement

La partie backend, gérée par Spring Boot, gère avec un contrôleur par entité, toutes les actions relatives à la base de données. Que ce soit récupérer, modifier, effacer, tout passe par eux. Ils renvoient leurs réponses en format JSON (JavaScript Object Notation) qui est ensuite interprété et affiché avec le frontend.

Chaque contrôleur communique avec des services, qui eux s'occupent de récupérer et transformer les données avant de retourner leurs résultats aux contrôleurs.

Avec le frontend, chaque page correspond aussi aux entités. Une fois la page voulue chargée, un service nommé API va faire les requêtes au backend, puis le résultat (ou l’erreur) est géré par le composant appelant.

La session est sauvegardée par un service nommé “auth”, qui lors de la connexion, va se rappeler de l’utilisateur connecté. La sauvegarde est éphémère, nous sommes

déconnecté dès que la page est rechargée, mais gardé cela en cookie ou en localStorage n'est qu'une question de quelques lignes si nécessaire.

Conclusion

L'application ainsi créée répond au cahier des charges en plus d'être simple d'utilisation.