Projet Gestion des Sous-Traitants

Introduction

L'objectif principal du Projet Gestion des Sous-Traitants sur une application web de type RH est que l'utilisateur connecté ait accès rapidement aux données du collaborateur externe (Sous-Traitant) se trouvant dans son périmètre de « Business Units », en sortie de mission d'ici 15 jours, et puisse prolonger directement sa mission.

Caractéristiques du projet :

- ❑ Nouvelle fenêtre avec un tableau recensant les collaborateurs sous-traitants en sortie d'ici 15 jours
- ❑ Panneau « Externes » affiché dans la page « Planning » de l'application web
- ❑ Web service intitulé getCollabSortieQuinzeJours qui retourne une liste de map d'objets en format JSON

Aspects spécifiques :

Langages : HTML / CSS / JADE côté Front-End et Java J2EE / SQL en lien avec une base de données relationnelle MySQL côté Back-End

Logiciels de développement : Eclipse IDE + serveur Tomcat v9.0, Visual Studio Code

Outil de communication des tâches à faire, en cours et terminées : Microsoft Teams

Autres ressources logicielles : Turquoise SVN (Versionning), Wampserver, MySQL Workbench

Ressources utilisées : le guide du développeur pour l'installation de l'environnement du projet sur l'ordinateur, des lignes de code dans le projet, des recherches de solutions techniques sur Internet et l'aide de mes collègues développeurs sur le projet.

Résolution technique :

Le module framework Spring MVC est présent dans le projet de l'application web, respectant le modèle de conception MVC (Model-View-Controlleur). Ce module permet le développement d'applications web en Java.

Côté Back-end, j'ai créé un web service intitulé getCollabSortieQuinzeJours qui retourne une liste de map d'objets en format JSON. Son but est de collecter les données dont j'ai besoin pour remplir le tableau côté Front-End en fonction des noms de colonne et du périmètre de « Business Units » de l'utilisateur connecté à l'application web. J'ai implémenté le web service dans les couches Service, Controller et DAO de la classe Collaborateur. Dans la couche DAO, pour manipuler les données de la base de données MySQL, j'ai utilisé l'API Criteria d'Hibernate pour que le code ne représente pas une faille de sécurité par l'intermédiaire de possibles injections SQL dans l'URL du navigateur.

Ensuite, j'ai créé une instance de Criteria, Session avec sa méthode createCriteria() ayant en paramètre le nom de la classe mappée, comme dans l'exemple ci-dessous :

"Criteria criteria= session.CreateCriteria(Collaborateur.class);"

L'instance d'objet créée va requêter sur la classe mappée Collaborateur en se basant sur des critères de recherche.

L'API Criteria comprend 5 classes :

- Criteria
- Criterion
- Restrictions
- Projections
- Order.

J'ai appelé certaines de ces classes pour obtenir les restrictions que je souhaitais dans le tableau.

J'ai d'abord fait des jointures dans l'instance de Criteria en utilisant la méthode setFetchMode pour pouvoir accéder aux objets de la classe Tarif depuis la classe Collaborateur en passant par la classe Signature et j'ai nommé avec des alias les jointures de ces classes.

Pour le mappage d'objet de la classe Collaborateur, j'ai sélectionné les objets dans une liste de projection ajoutée à l'instance de Criteria via la méthode setProjection :

« crit.setProjection(Projections.projectionList()
.add(Property.forName("c.id"), "id")); »

"c.id" correspond à l'objet Id de la classe Collaborateur et "id" correspond au nom de la propriété.

Ensuite, j'ai mis en place des contraintes en utilisant certaines méthodes de la classe Restrictions. Cette classe définit plusieurs méthodes « static » héritées de la classe Criterion qui permettent d'implémenter aussi bien des restrictions que via la clause « where » d'une requête SQL.

Voici quelques-unes de ces méthodes utilisées :

- « eq(String propertyName, Object value) » : la valeur de l'objet est égale à la propriété nommée
- « gt(String propertyName, Object value) » : la valeur de l'objet est strictement supérieure à la propriété nommée
- « le(String propertyName, Object value) » : la valeur de l'objet est inférieure ou égale à la propriété nommée
- « in(String propertyName, Object[] values) » : le périmètre de la propriété nommée est les valeurs du tableau d'objet.

J'ai utilisé les méthodes « le » et « gt » pour n'afficher dans le tableau que les collaborateurs en sortie entre demain et + 15 jours, la méthode « eq » pour n'afficher que les externes et la méthode « in » pour n'afficher que les collaborateurs dans le périmètre de « Business Units » de l'utilisateur connecté à l'application web. J'ai dû récupérer l'Id de l'utilisateur connecté dans une variable de type Collaborateur via l'interface HttpServletRequest dans la couche Controller de la classe Collaborateur. J'ai dû aussi faire passer la variable en paramètre du web service pour pouvoir associer l'Id de l'utilisateur connecté aux Id des « BU » dans l'implémentation DAO de la classe Collaborateur.

J'ai ajouté chacune de ces contraintes à l'instance d'objet Criteria ainsi que la méthode addOrder : « addOrder(Order.asc("dateSortie")) » pour classer les dates de sortie par ordre ascendant.

Côté Front-end, dans le fichier JavaScript, via une URL HTTP, j'ai appelé les objets mappés issus du web service getCollabSortieQuinzeJours et je les ai ajoutés dans une ArrayList (liste dynamique), puis dans un tableau d'objet.

J'ai construit le tableau en JQuery, visible par l'utilisateur de l'application, associé au fichier CSS pour sa mise en forme. J'ai ajouté les titres des colonnes et je les ai associés aux objets correspondants. La cinquième colonne du tableau correspond à la prolongation : j'ai associé le titre de la colonne à un bouton « Prolonger » avec une icône sur chaque ligne.

Ce bouton « Prolonger » ouvre une pop-up intitulée « Prolonger / Ecourter ». Cette pop-up était déjà existante dans le projet : j'ai repris le code HTML et JQuery de création du modal en ne gardant que les lignes de code dont j'avais besoin pour traiter le cas de la prolongation et envoyer les données « Nouvelle date de disponibilité » et « Motif de la modification » vers le web service correspondant. Dans le processus « success » de validation de la prolongation, en plus de l'envoi de ces deux données, un spinner s'active, la pop-up se ferme, une pop-up de message positif s'affiche pendant une seconde puis se ferme et, enfin, la fenêtre du tableau se recharge. Dans le cas d'un échec du processus de prolongation, un message « Vous n'avez pas les droits pour effectuer cette action » s'affiche, le spinner s'arrête et la pop-up de prolongation reste ouverte.

Image 1 : Visuel de la pop-up "Prolonger / Ecourter"

Dans l'arborescence « planning-client » du projet, et plus précisément dans la page « Planning », j'ai inséré un panneau intitulé « Externes » comprenant le nombre de collaborateurs sous-traitants en sortie d'ici 15 jours et le bouton « Prolonger » d'accès à la fenêtre du tableau de gestion des sous-traitants. Pour ce faire, j'ai développé, à part du projet, la mise en forme du panneau dans un fichier HTML couplé avec un fichier CSS. Une fois ceci réalisé, j'ai fait convertir le code HTML en JADE sur un site internet. Concernant le bouton « Prolonger », j'ai développé la méthode « Window.open() » en JQuery, dans la fonction « onClick » pour gérer l'ouverture de la fenêtre du tableau de gestion des sous-traitants.

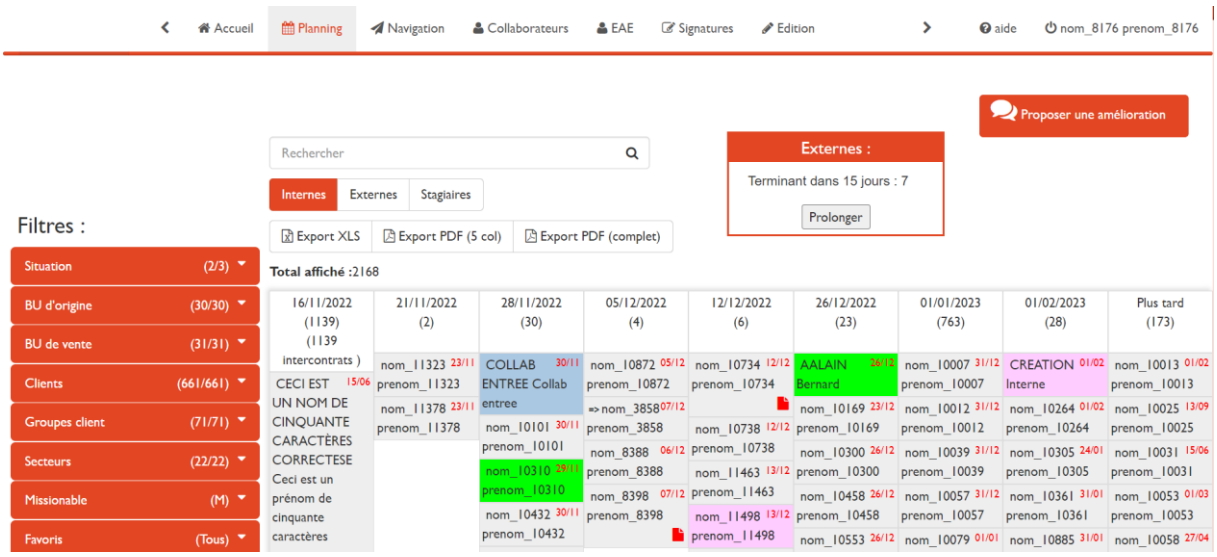

Image 2 : Visuel de la page « Planning » avec le panneau de gestion des collaborateurs externes