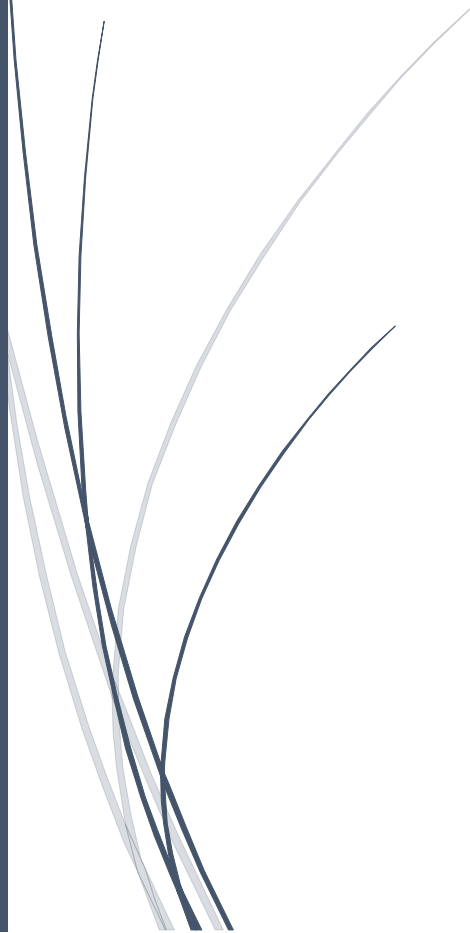
A dark blue vertical bar on the left side of the page. A blue arrow points to the right from this bar, containing the date.

29/05/2022

Rapport projet de gestion des réclamations

Architecture des applications
d'entreprise

Several thin, curved lines in dark blue and light grey originate from the bottom left and curve upwards and to the right.

MEYA Georges HUGOT Alexandre MONIMEAU
PAUL

POLYTECH MARSEILLE

Sommaire

Table des matières

1.	Description du projet.....	2
2.	Architecture du projet.....	2
3.	Persistance des données Erreur ! Signet non défini.	
4.	Conceptions des classes	4
5.	Gestion de l'ergonomie	5
6.	Conclusion	7

1. Description du projet

Notre projet correspond à un mini projet développé avec la technologie Springboot dont l'objectif est de mettre en place un système de gestion de réclamation. Celui-ci sera composé de plusieurs entités qui vont interagir entre elle afin de le faire fonctionner.

Le fonctionnement reste simple, une personne s'inscrit ou se connecte si elle possède un compte, si cette personne est un utilisateur lambda (sans aucun privilèges) elle pourra soumettre des réclamations qui vont s'enregistrer dans la base de données.

Dans le cas contraire, la personne est un utilisateur administrateur et peut modifier le rôle des différents utilisateurs ou bien voir toutes les réclamations (il a accès à plusieurs informations les concernant comme : la date à laquelle elle a été soumise, le titre, le message, et le compte de l'utilisateur ayant déposé la réclamation) et décider de leurs validations.

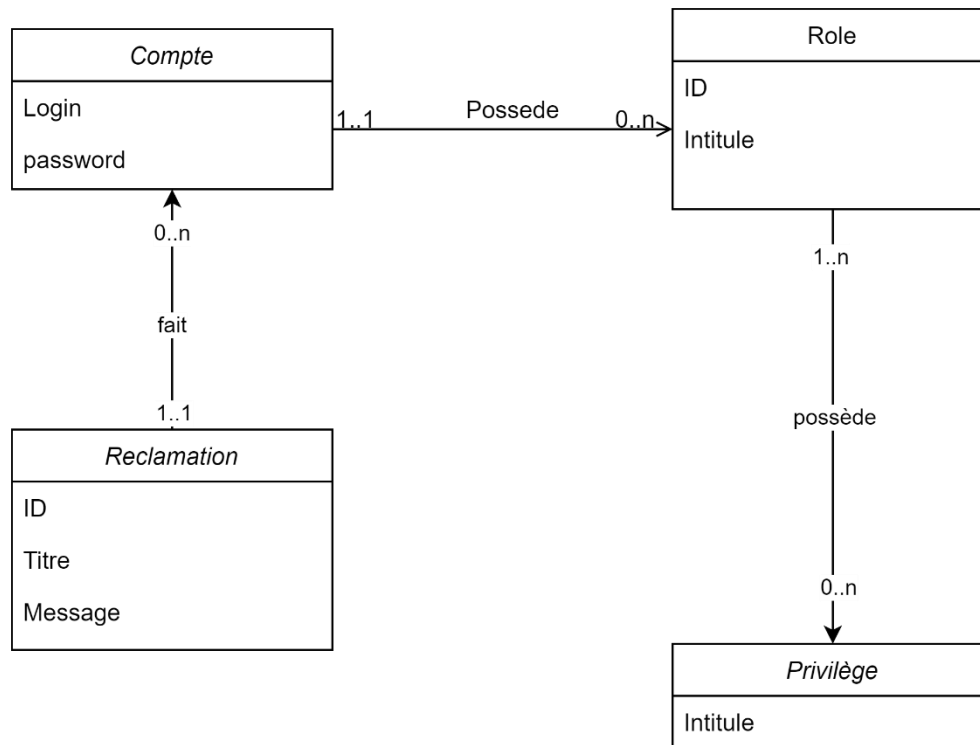
Nous allons détailler par la suite l'architecture de l'application (ses entités, l'organisation des dossiers...).

2. Architecture du projet

Le projet suit une architecture MVC, où le modèle est constitué des classes suivantes :

- Compte
- Rôle
- Reclamation
- Privilège

Voici le diagramme de classe associé à cela :



Chaque classe du modèle est associée à une classe contrôleur, une classe service et une classe répertoire toutes regroupées dans différents packages.

Les classes contrôleur auront pour rôle de router les différents liens vers les différentes vues du projet et de gérer les interactions de l'utilisateur avec le site web en fonction du modèle qu'il manipule. Par exemple si nous nous trouvons sur la page de création d'une réclamation c'est la classe ReclamationController qui gèrera les événements liés à cette page.

Les classes Service servent à faire des opérations sur la base de données en utilisant les objets des classes modèles. Ces classes sont utilisées dans les classes contrôleurs afin de réaliser des requêtes vers les bases de données.

Les classes répertoires servent à faire le lien entre notre application et la base de données. C'est dans ces classes que nous pourrions spécifier des requêtes qui seront utilisées dans les classes service.

Les vues sont-elles générées dans des fichiers .jsp

3. Conceptions des classes Controller

- LoginController

Le login controller s'occupe de proposer les pages de connexion et d'enregistrement d'un utilisateur grâce aux mapping « /login » et « /addCompte » .

Il crée aussi un cookie de session permettant d'être authentifié sur toutes les autres pages de l'application.

En fonction de si l'utilisateur possède des privilèges d'administrateur ou de simple utilisateur, il sera redirigé vers le dashboard correspondant à ses privilèges.

A savoir que le premier utilisateur créé de l'application sera un ADMIN.

- PrivilegeController

Il sert simplement à affecter un privilège à un rôle.

- ReclamationController

Plus complexe, ce dernier permet l'affichage des tableaux de bord en fonction des privilèges de l'utilisateur connecté. Et de gérer les réclamations.

Pour un utilisateur normal, on arrive sur la liste de nos réclamations ainsi que leur état (validé / refusé / en cours de traitement). On peut vouloir ajouter une réclamation, alors on est redirigé vers « /reclamationsUser » qui nous propose un formulaire pour remplir une nouvelle réclamation.

Pour ce qui est du dashboard de l'admin, on a le choix entre rajouter un privilège à un utilisateur et voir les réclamations.

Pour ce qui est des réclamations, on y arrive avec le lien « /reclamationsAdmin »

Ce qui nous affiche la liste des réclamations avec un bouton pour la valider et un pour la refuser.

- RoleController

« /addCompteRole » et la fonction principale de ce controller, elle nous permet d'affecter un rôle à un compte à l'aide de son login et mot de passe.

4. Persistance des données

La persistance des données est assurée grâce à une Base de données SQL MySQL. Les tables présentes dans la base de données sont automatiquement créées grâce au code contenu dans les classes modèle que nous avons.

a. Inscription

Pour l'inscription, la fonction `creerCompte` de `LoginService` permet d'utiliser la méthode `Save` de la classe `LoginRepository`. Cette méthode permet de mettre à jour ou insérer un nouvel utilisateur dans la base de données.

b. Ajout d'un rôle

Pour l'ajout d'un rôle, la fonction `creerRole` de `RoleService` permet d'utiliser la méthode `Save` de la classe `RoleRepository`. Cette méthode permet de mettre à jour ou insérer un nouveau rôle dans la base de données.

c. Nouvelle réclamation

Pour l'ajout d'une réclamation, la fonction `creerReclamation` de `ReclamationService` permet d'utiliser la méthode `Save` de la classe `ReclamationRepository`. Cette méthode permet de mettre à jour ou insérer une nouvelle réclamation dans la base de données.

Cookie de connexion :

Afin de pouvoir sauvegarder quel utilisateur est connecté il est nécessaire de créer des cookies de connexions.

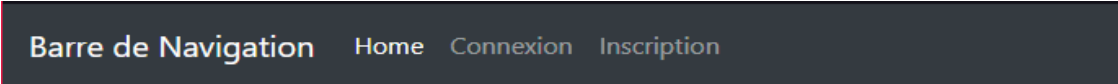
Ces cookies seront utiles pour pouvoir faire des requêtes à la base de données en utilisant le login de l'utilisateur connecté ou savoir quel est le rôle de l'utilisateur.

5. Gestion de l'ergonomie

La gestion de l'ergonomie a été la dernière étape de notre projet. Une fois le projet fonctionnel, il a été nécessaire de modifier une grande partie de notre code notamment supprimer ou modifier des fonctions obsolètes dont le code laissait à désirer.

La mise en place des pages web a été un point important dans la gestion de l'ergonomie. Nous avons ainsi intégré Bootstrap à notre projet afin de pouvoir faciliter la mise en page. Les pages sont faciles à comprendre et à utiliser tout en apportant les informations nécessaires. On va présenter dans ce documents trois exemples :

1. L'utilisation d'une navbar qui s'adapte en fonction de notre position sur le site et de la taille de l'écran,



Barre de Navigation Home Connexion Inscription

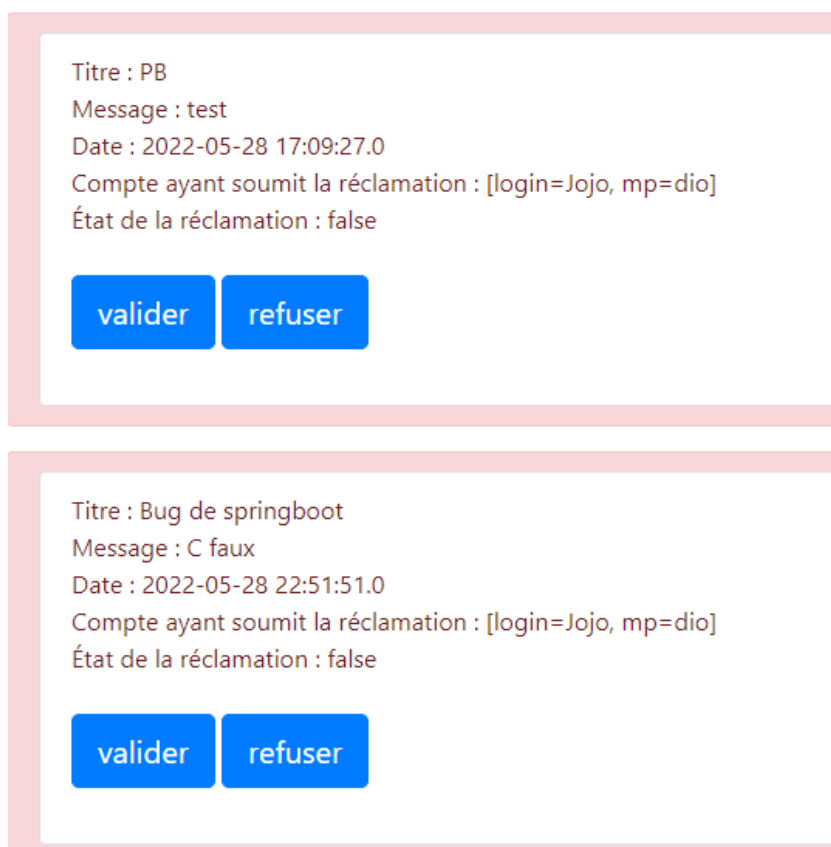
Ceci est la barre de navigation présente dès l'arrivée sur le site avec 3 possibilités, aller à l'accueil (Home), se connecter et s'inscrire.



Ceci est la barre de navigation une fois connecté avec une compte user et avec l'écran réduit.

2. Dans le Dashboard de l'admin, l'affichage des réclamations sous forme de liste afin de bien recenser toutes les informations les concernant tout en les séparant (pas d'incompréhension ou de confusion sur celles-ci). Les boutons valider et refuser permettent de modifier l'état de la réclamation choisit.

Liste des réclamations



3. Dans le Dashboard user, l'affichage des différentes réclamations faites par cet utilisateur :

Liste de vos réclamations

#	Titre réclamation	Date	Status
11	Bug de springboot	Sun May 29 22:25:19 CEST 2022	En traitement

Nouvelle réclamation

6. Conclusion

Le Framework SpringBoot est une technologie permettant de bénéficier complètement de l'utilité du MVC. Couplé au service JSP, cela nous a permis de produire des applications web plus complexes.

Bien que nous aurions pu nous rediriger vers des Frameworks plus complexes, utiles pour de l'affichage comme Thymleaf ou Vadim, l'utilisation de SpringBoot et des services JSP nous ont amplement suffi.

Ce projet nous a permis d'entrevoir les possibilités qu'offre SpringBoot et notamment en termes de backend, et de gestion de la donnée.