Documentation de l'application web

Table des matières

I. Informations importantes

Le code se découpe en deux parties.

D'une première part, nous avons le dossier *back*, qui contient tout ce qui concerne le côté serveur. Celui ci se compose de la manière suivante :

⇒ **bin** qui contient la compilation des différentes classes java

⇒ **config** qui doit contenir le fichier application.properties. C'est ici que l'on configure la clé secrète pour le JWT (voir le ReadMe)

⇒ **libs** qui contient les librairies utilisées dans le projet

⇒ **database** qui contient toute la gestion à la base de données mySql. Dans mon cas, j'ai utilisé Xampp.

⇒ **src** qui contient tout ce qui suit :

⇒ **controller** qui contient toutes les classes appelant la classe jumelle se trouvant dans **dao**, et qui fait la liaison entre l'appel d'API et les requêtes SQL.

⇒ **dao** qui contient les classes manipulant les **models** et les requête sql afin de créer une interaction entre serveur et base de données.

⇒ **models** qui contient les records java reprenant la structure des tables de la base de données.

⇒ **pdf** qui est le répertoire où sont enregistrées les différents pdf déposés à travers la plateforme.

⇒ **utils** qui contient la méthode de hashage du mot de passe.

⇒ **webserver** qui gère les communications client-serveur

⇒ **App.java**, l'application principal permettant de lancer le serveur et où se trouve les différents endpoints appelant les méthodes adéquate au sein des **controllers**.

D'une seconde part, nous avons le dossier *front* qui lui, contient tout ce qui est en rapport avec le client. Son architecture reprend le modèle MVC et ce compose comme suit :

⇒ **css** qui contient les scripts gérant le css des pages webs.

⇒ **libs** qui contient les librairies utilisées.

⇒ **pages** qui contient le code html des différentes pages.

⇒ **services** qui contient les scripts js « service » de chacune des pages. C'est à dire qu'ici on retrouvera les méthodes communiquant avec les endpoints dans **App.java**. En résumé, on y retrouve ce qui lie le client et le serveur.

⇒ **pdftest** qui contient juste des pdf pour mes test.

⇒ js qui contient tout ce qui suit :

⇒ **views** qui gère l'affichage des pages webs. C'est ici qu'est appelé les méthodes crées dans **services.**

⇒ **controllers** qui appelle les **views**.

⇒ les **scripts .js** de chacune des pages.

Enfin nous avons **hashage.py** qui me permet de rapidement hasher les mots de passe pour compléter la base de donnée grâce au **script.sql.**

Pour faire fonctionner tout ceci j'utilisais VSCode, plus précisément Live Server sur index.html et je lançais App.java.

II . Authentification

Voici les différents codes permettant d'accéder à la plateforme :

Username	Password
admin	admin
machin	bidule
geredestrucs	responsible
geredestrucsdevis	responsibledeux
geredestrucsdevistel	responsibletrois
gerelesstocks	carton
dirigedestrucs	classeur
faitdestrucs	

Chacun de ces utilisateurs possèdent des rôles et des autorisations différentes. Bien évidemment admin les possèdent tous.

III. Connect.html

III.1 Client

⇒ handleLogin (view) :

Gère la connexion de l'utilisateur en appelant la méthode `login` du service `connectServices`. * Récupère le nom d'utilisateur et le mot de passe à partir des champs de formulaire, puis tente de se connecter. Si la connexion est réussie, le token JWT est stocké dans le `localStorage` et l'utilisateur est redirigé vers la page d'accueil. Sinon, un message d'erreur est affiché.

⇒ **login (service) :**

Tente de se connecter en envoyant une requête POST au serveur avec les informations de connexion. La réponse du serveur est ensuite renvoyée en tant qu'objet JSON.

III.2 Serveur

⇒ **/api/login (App.java) :**

Crée le endpoint précédemment appelé par le connect-services.

⇒ **login (UtilisateurController) :**

Gère la connexion d'un utilisateur en traitant la requête HTTP. Cette méthode extrait les informations de connexion (nom d'utilisateur et mot de passe) du corps de la requête, les désérialise en un objet `LoginRequest`, puis vérifie les informations de connexion contre la base de données. Si les informations sont valides, elle génère un JWT (JSON Web Token) et le renvoie avec un message de succès. Si les informations sont incorrectes ou manquantes, elle renvoie un message d'échec. En cas d'erreur pendant le traitement, elle renvoie un message d'erreur.

⇒ **findByUsernameAndPassword(UtilisateurDao) :**

Recherche un utilisateur dans la base de données en fonction du nom d'utilisateur et du mot de passe fournis. Cette méthode exécute une requête SQL pour trouver un utilisateur dont le nom d'utilisateur et le mot de passe correspondent aux valeurs fournies. Elle retourne un objet `Utilisateur` si une correspondance est trouvée, sinon elle retourne `null`.

IV. home.html

Étant donné le très grand nombre d'API contenu dans ce projet, nous ne traiterons que les plus importantes afin donner une idée globale de comment fonctionne 'implémentation d'une fonctionnalité (explication se trouvant dans le ReadMe)