Projet - Web Avancé

Site de gestion association sportive

Élèves :

Aïssa Pansan

El Varougou Haïballa

Enseignant :

Benoit Villa

16 avril 2023

Send you our best regards,

Sommaire

1 Introduction générale

Nous avons l'honneur de vous présenter le rapport de notre projet de Web Avancé. Ce projet s'inscrit dans le cadre de notre formation en d'informaticien ingénieur, qui comprend l'utilisation du langage Java EE. Ce projet a été l'occasion pour nous de mettre en pratique les concepts appris en classe.

Le but de ce projet est de développer une application Web pour une association sportive. Dans cette application, on doit pouvoir réaliser les actions suivantes : ‘

— Voir la liste des adhérents

— Ajouter des adhérents

— Modifier/Supprimer des adhérents

— Constituer des groupes et assigner des adhérents dans ces groupes.

— Ajouter, modifier et supprimer des groupes

Une API **REST** doit aussi être développé afin de réaliser plusieurs opérations sur le site. Enfin, pour tester le site, on doit utiliser **Selenium** afin de réaliser des actions automatiques.

Enfin, ce rapport présente les détails de notre travail, expliquant les décisions prises et les difficultés rencontrées durant son élaboration. Il a pour objectif de synthétiser notre démarche de résolution en détaillant succinctement les étapes clé du projet.

2 Description de l'application

Le projet est structuré selon **Maven**. Il comprend donc 3 dossiers primordiaux localisés dans **Projet/src/main** :

— **Java :** Contient toutes les **Servlet** nécessaire aux fonctionnements du projet et contient le dossier $\mathbf{Class}_{D}efinitionpourladfinitiond'unAdhrentetd'unGroupe.LaServletForm.$

— **resources** : Le dossier qui contient **persitence.xml**, le fichier incontournable pour mettre en relation la JPA avec **Hibernate**, notre driver pour utiliser une base de donnée **MariaDB**

— **webapp** : Le dossier qui contient plusieurs fichiers **HTML** et **JPA** afin d'afficher le rendu front-end. Il est aussi accompagné d'un dossier **style** pour gérer la décoration du site en **CSS**

3 Fonctionnalité implémentés

Pour ce projet, nous avons pu mettre en place les fonctionnalités **d'ajout d'utilisateurs** et **de suppression d'utilisateurs**. Enfin, l'API REST a été développé, ainsi qu'un test en **Selenium**.

4 Nos difficultés

4.1 Fonctionnalités manquantes

Une fonctionnalité que nous n'avons pas pu mener à terme est la gestion des groupes et la relation entre utilisateurs et groupes. Notre objectif initial était de permettre la création de groupes, l'affectation d'utilisateurs à ces groupes, et la persistance de ces informations dans la base de données. Cependant, faute de temps et à cause des problèmes techniques antérieurs, cette partie a été laissée inachevée.

4.2 Difficultés

Au cours de ce projet, plusieurs contraintes ont ralenti le développement et ont conduit à l'absence ou l'incomplétude de certaines fonctionnalités prévues :

Problèmes de configuration et de prise en main d'Eclipse

Dès le début, nous avons rencontré des difficultés liées à la configuration du projet dans l'IDE Eclipse. L'installation et la compatibilité de différents plugins, la gestion des dépendances via Maven, ainsi que les conflits entre JUnit 4 et 5 ont provoqué des retards. La correction des fichiers pom.xml et la validation de la compilation ont nécessité de nombreuses itérations avant de parvenir à un environnement de développement stable.

M**anque de temps et emploi du temps chargé**

Parallèlement aux difficultés techniques, nous avons fait face à un emploi du temps très chargé. D'autres cours et projets ont limité notre disponibilité pour travailler sur ce projet, entraînant des retards supplémentaires et repoussant la finalisation de certaines parties du code.

Difficultés avec les tests Selenium

L'écriture de tests fonctionnels via Selenium pour valider l'interface s'est avérée plus complexe que prévu. Outre la configuration du WebDriver (GeckoDriver pour Firefox) et

l'ajout des dépendances requises, nous avons rencontré quelques soucis de synchronisation et de sélection de bons éléments sur la page. Il a fallu notamment gérer les temps de chargement, veiller à l'existence des éléments HTML, et ajuster finement les localisateurs (By.id, By.linkText, etc.). Ces points techniques ont nécessité un travail de débogage important pour fiabiliser la suite de tests.

Difficultés dans l'affichage et la gestion des utilisateurs

Concernant l'affichage des utilisateurs, nous avons initialement tenté de gérer une ArrayList locale pour stocker et manipuler les données, plutôt que de lire et mettre à jour directement la base de données. Cette approche a engendré de la complexité : il fallait maintenir la cohérence entre la liste locale et les données réelles. Nous avons ensuite réalisé qu'il aurait été plus efficace de faire des requêtes directes sur la base, assurant ainsi la fiabilité de l'affichage.

Problèmes liés à l'API REST avec RESTEasy

Déploiement sur Tomcat : Nous avons implémenté une API REST via RESTEasy, mais Tomcat refusait le déploiement à cause d'une erreur dans le fichier persistence.xml. Cette erreur rendait impossible la mise en place correcte des entités et l'accès à la base de données au sein du conteneur. Résoudre ce problème a pris du temps, car il fallait ajuster la configuration de persistance et veiller à la cohérence entre le persistence unit name, le driver JDBC et les dépendances Maven. Insertion des données : Même après la correction de persistence.xml, nous avons constaté des difficultés lors de l'insertion de données dans la base : certaines requêtes ne s'exécutaient pas comme prévu, en raison d'incompatibilités entre la configuration du EntityManager et la structure de notre base MariaDB. Il a fallu procéder à plusieurs ajustements et tests pour parvenir à une insertion réussie et cohérente des entités.