Rapport Projet WEB Serveur

Saphir GOBBI, Semih DOYNUK

Sommaire

- Introduction
- Les choix architecturaux
- Les URL
- Conclusion

Introduction

Notre projet est d'implémenter une application web de gestion de propriété en se basant sur l'architecture Modèle Vue Controleur.

Nous verrons dans une premier temps les choix architecturaux puis les URL et enfin nous conclurons.

Les choix architecturaux

1. Quelles sont les écrans qui semblent intéressantes à afficher aux utilisateurs ?

Une fois connecté via les pages d'identification, une page d'accueil nous affiche les

choix qui s'offre à nous : User pour les afficher, supprimer ou en ajouter, de même pour Syndicat et Immeuble. Ce dernier peut voir son contenu modifié par une redirection, ce qui est également le cas pour Appartement et Personne. Et enfin une option de déconnexion.

2. Quelles vont être les principales classes de votre application ?

Dans le respect de notre architecture MVC nous avons en guise de view les classes GUI, en guise de model qui communiquent avec les classes DAO (celles qui gèrent de la base de données) nous avons les classes Core. Les classes Entity qui définissent les objets métiers de l'application, de plus certaines classes sont liés entres elles par compositions tels que ImmeubleEntity et AppartementEntity ainsi que AppartementEntity et PersonneEntity.

les classes SignInGUI, LoginGUI et AccueilGUI sont des classes de représentation ne possédant pas de classes Entity DAO ou Core.

3. Quelles vont être les tables de votre base de données ?

Les tables de nôtres base de données sont :

- User
- Syndicat
- Immeuble
- Appartement
- Personne

Certaines tables sont liés par leurs suppression comme Immeuble et Appartement, si un Immeuble est supprimé tout les appartements qui sont liés seront supprimé a leurs tours. De même pour Appartement et Personne.

4. Quelles solutions technologiques envisagez vous ?

On a donc utilisé Java Spark en guise de Controleur, qui fait des appels aux classes Core du Modèle qui inclus les classes Entity et DAO pour communiquer avec la base de donnée. Pour cette dernière nous avons employé java.sql. Notre Vue utilise un système de template en push via FreeMarker et utilise un langage restreint pour les représentations.

Les liens URL

Tout d'abord lorsqu'on accède au site si l'utilisateur n'est pas connecté il est redirigé vers la page de connexion.

Le lien /logout change le statut de connexion et redirige vers la page /login.

Dès lors de la connexion, l'utilisateur est redirigé vers la page d'acceuil avec comme url /:id/home où :id est remplacé par l'identifiant de l'utilisateur, ce dernier est créée par la base de donnée comme vu précédemment.

Nous utilisons les mêmes types de lien pour les pages :

- /:id/user
- /:id/syndicat
- /:id/personne
- /:id/appartement
- /:id/immeuble

Le lien /:id/user permet de récupérer tout les utilisateurs en ajouter et supprimer.

/:id/syndicat Permet d'afficher et modifier tout les syndicats, de même pour /:id/immeuble.

Pour acceder aux appartements on peut directement aller dans /:id/appartement pour avoir une liste de tout les appartements, ou être redirigé sur /:id/immeuble/:idadresse/appartement pour en ajouter ou supprimer.

On peut accéder aux personnes avec /:id/personne, on peut trouver les habitants un appartement avec /id/immeuble/:idadresse/:idappartement/personne

---

Conclusion

En conclusion, l'application web de gestion de propriété implémentée suit une architecture MVC classique. Elle offre aux utilisateurs un ensemble de fonctionnalités pour gérer les propriétés, y compris les utilisateurs, les syndicats, les immeubles, les appartements et les personnes.

L'application utilise Java Spark comme framework côté serveur, une base de données relationnelle et FreeMarker pour le templating.

L'utilisation de l'architecture MVC a permis de structurer le code de manière claire et maintenable, tandis que le choix des technologies permet une bonne évolutivité de l'application.

Points forts de l'application :

- Respect des principes de l'architecture MVC
- Fonctionnalités complètes pour la gestion des propriétés
- Technologies modernes et évolutives
- Code clair et maintenable

Perspectives d'amélioration :

- Implémentation d'une interface utilisateur plus moderne et responsive
- Ajout de fonctionnalités supplémentaires, telles que la gestion des contrats de location ou des charges
- Mise en place d'un système de sécurité plus robuste