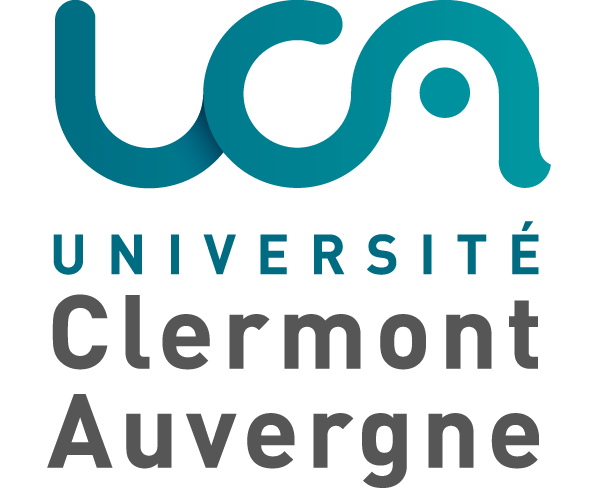
Projet présenté par :

BENLOULOU Sarra NEKKAA Yousra



Université Clermont Auvergne

Institut d’Informatique

Département de l’informatique

|  |
| --- |
| **Gestionnaire De Liste** |

Encadré par :

Yannick Loiseau

*Année Universitaire : 2018-2019*

Table des Matières :

* Cycle de vie de notre application
* Fonctionnalités implémentées
* Fonctionnalités non implémentées
* Fonctionnalités à améliorer
* Difficultés
* Notions apprises

**Cycle de vie de notre application :**

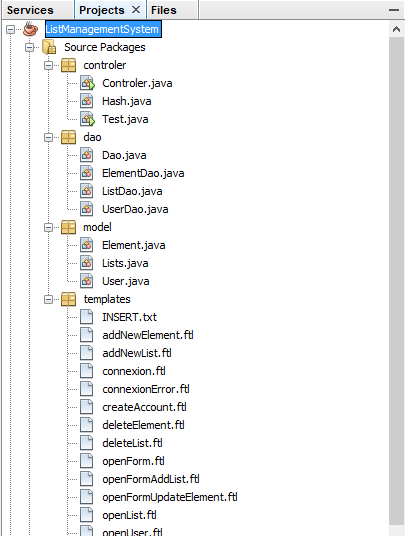
*Analyse* : la première tâche était d’analyser les objectifs attendus et les contraintes que nous pouvons avoir pour réaliser ce projet.

La contrainte la plus importante était le manque de connaissance sur les technologies imposées : spark, freeMarker, sql2o , et aussi le modèle MVC.

Le fait de se documenter sur chaque technologie nous a pris beaucoup de temps tout en l’essayant sur des petits tests

*Décomposition* : à l’aide d’un UML nous avons discuté la manière dont notre projet doit être décomposé pour faciliter son implémentation. nous avons mis en place plusieurs UML, puis nous avons choisi la version qui à notre point de vue nous semblait la meilleure car elle respecte les règles de POO et répond à nos besoins.

* Notre UML contient : des sous Classes DaoList , DaoUser et DaoElement qui héritent de la surClasse Dao qui a comme objectif de se connecter à la base de donnée. Ce choix nous a aidé à mieux visualiser notre programme et le comprendre.
* le contrôleur: rassemble toutes les méthodes http de tous les objets (Element, User, List) nous pouvons également faire un contrôleur pour chaque objet, mais comme notre application n’est pas très grande, un seul suffit.

**Conception et *Realisation :***

* nous avons utilisé NetBeans comme IDE.

nous avons commencé par la partie modèle, création des classes: Lists , User, Element.

Puis , dans le contrôleur à chaque fois qu’une méthode est implémentée nous lui configurons sa vue avec le Système de templates.

le Système de templates que freeMarker utilise est en *push*, nous envoyons les données à partir du contrôleur et nous ne faisons à aucun moment un appel d’une méthode ou une modification à partir de la vue.

* Avant d’utiliser une méthode http on regarde bien si elle est vraiment adaptée à son contexte et cela pour éviter ce genre de problème :

Apparition de ID dans la barre de navigation à cause d'une mauvaise utilisation de la méthode get



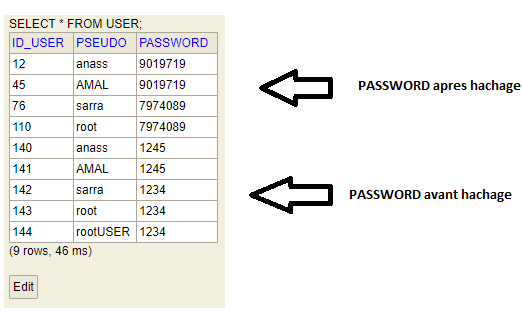
* préférer la méthode POST à la méthode PUT pour l'ajout des données.
* respecter le principe CQS : une Query retourne une valeur, sans qu’il ait d’effet de bord, une Commande a un effet de bord et ne retourne rien
* respecter le fait que l’URL soit opaque, pas d'apparition des extensions de fichier dans l’URL (.ftl) , pas de nom ou pseudo d’un utilisateur ….etc
* Les méthodes implémentées dans le contrôleur ne garde pas état d’un Client

*Test* : nous faisons des tests de fonctionnements à chaque fois qu’une nouvelle fonction est ajoutée ou une modification est faite dans l’application, par contre nous n’avons pas implémenté des tests unitaires au saint de notre application (comme cela était fait durant le TP de génie logiciel)

ça pourrait être une amélioration si nous souhaitons un niveau élevé de performance.

**Fonctionalite implementés :**

nous avons mis en place un système MVC( model vue controleur ) pour implémenter notre application web,

* l’utilisateur peut se connecter en entrant son Pseudo et mot de passe
* le contrôleur appel une fonction de vérification sur le pseudo et le mot de passe hashé (les mots de passe sont haché puis stocké )
* si la vérification s’est bien passée , une variable globale d'authentification passe à *true*
* l’utilisateur est dirigé vers une page où se trouve toutes ses listes
* l’utilisateur peut ouvrir, ajouter, modifier ou supprimer une liste
* à l'ouverture d’une liste il peut voir les éléments de sa liste
* chaque élément peut être supprimé ou modifié
* à la déconnection la variable globale d'authentification passe à *false*
* un utilisateur peut créer un compte, le pseudo et le mot de passe entrés seront stockés dans la base de donnée à l’aide des objets DaoUser et User

*Id jamais en clair*

**Fonctionnalité non implémentés :**

la fonctionnalité de partager une liste entre plusieurs utilisateurs n’est pas implémentée dans notre application, par contre elle est modélisée dans le MLD.

**Fonctionnalités à améliorer :**

notre système d'authentification est basé sur une variable globale qui passe à *true* si l’utilisateur tape le bon pseudo et mot de passe, et à *false* quand il se déconnecte. Nous pouvons améliorer ceci en ajoutant des cookies ou des sessions

**Difficulté :**

Les plus grandes difficultés sont apparues au début de notre implémentation. Elles ont figuré dans la compréhension du modèle MVC, la gestions des technologies et la base de données simultanément, ainsi que la correction des erreurs.

**Notions apprises :**

En faisant ce projet nous avons appris comment créer une application web, chose que nous n’avons jamais vue auparavant , et voir à quoi ressemble le modèle MVC en pratique, même si nous l’avons bien compris en théorie, la pratique reste plus intéressante . De plus nous avons appris à utiliser les technologies proposées : spark, freeMarker et sql2o. Nous avons aussi amélioré notre niveau en POO.

*Le Temps* : Nous avons également appris à ne pas accorder beaucoup de temps à l'amélioration d’une fonctionnalité alors qu'il en reste d’autres plus importante à implémenter.

*Partage des tâches* : Mieux partager les tâches implique moins de perte de temps et moins de bugs au moment du rassemblement, chaque membre du groupe doit savoir ce qu’il faut faire et comment le faire.

*UML* : Passer plus de temps à discuter de l’UML, pour éviter de faire de gros changement dans le code plus tard et s’assurer que l’UML choisi est bien compris par les membres du groupe, et surtout éviter d’apporter des modifications à l’UML sans les communiquer au reste du groupe.

NB : Nous vous prions de nous excuser pour le fait que notre application ne comporte pas un moyen de compiler comme gradle ou maven.

Après l’exécution de l’application avec netbeans, nous ouvrons la barre de navigation et nous tapons http://localhost:4567/