

SR03

Architecture des applications internet

Projet 2 :

Rapport projet application web en Java EE

LAVIOLETTE Etienne
FRISCH Gabriel



Printemps 2016

Table des matières

1	Cahier des charges	2
2	Spécification	4
2.1	Tomcat	4
2.2	Eclipse	4
2.3	Bootstrap	5
2.4	DAO et Factory	5
2.5	Recherche	6
2.6	Pagination	7
2.7	Jquery	8
2.8	Mysql	8
2.9	Accès sections admin / stagiaires	9
2.10	Envoi de mails après chaque inscription	9
3	Organisation	10
3.1	Répartition du travail	10
3.2	Utilisation d'outils	11
3.2.1	Git et Gitlab	11
3.2.2	Trello	12
3.2.3	phpMyAdmin	13
4	Architecture	15
4.1	Base de données Mysql	15
4.2	Division MVC	16
4.3	Les différents design pattern utilisés	16

4.4	Schéma récapitulatif de notre architecture	17
5	Idées d'améliorations	19
5.1	Amélioration de la BDD	19
5.2	Amélioration des fonctionnalités proposées	19
5.3	Amélioration de notre code	20
6	Sources	23

Introduction

Après avoir réalisé un premier projet en PHP totalement axé sur l'ergonomie et l'expérience utilisateur nous nous sommes vu confiés le développement d'une application web totalement différente comme second projet de SR03 pour ce semestre de P16. En effet, nous avons en charge la réalisation d'une application web en JAVA EE. Celle-ci se devait d'être axée "fonctionnalités". Pour mettre en pratique nos connaissances acquises en cours Mr.Nalatizio nous a imposé la réalisation d'une application de gestion de questionnaires. Deux utilisateurs étant en interaction avec notre application : des stagiaires réalisant des questionnaires et des administrateurs gérant les questionnaires, les questions et les réponses ainsi que les utilisateurs de notre plateforme.

Tout au long de ce rapport nous allons vous présenter la démarche qui a été la notre lors de la réalisation de ce second projet web de SR03. Nous essaierons de vous présenter le cahier des charges originel que nous avons traduit en spécifications, l'architecture mise en place puis les outils nous ayant accompagné pendant la réalisation de cette application.

Chapitre 1

Cahier des charges

Lors de la séance de TD du 28 Avril, Monsieur Enrico Natalizio nous a présenté le second projet web de l'UV SR03.

A la fin de la présentation de ce projet nous a été remis un cahier des charges détaillant ses désirs et les objectifs qu'il nous fixait quand à la réalisation de l'application web.

Voici les principaux points de ce cahier des charges :

- Réaliser une application web permettant d'évaluer les compétences des stagiaires à l'aide de différents questionnaires basés sur des sujets différents.
- Chaque questionnaire devra se présenter sous la forme d'un QCM à choix multiple, une seule réponse n'étant bonne pour chaque question.
- Gérer les fonctionnalités suivantes : gestion des utilisations, des questionnaires, des résultats et évaluation des compétences
- Réaliser au minimum la gestion des administrateur et des stagiaires et la possibilité de gérer les questionnaires dans notre projet. Cette base est considérée comme le strict minimum pour la réalisation de ce projet.
- Aucune contrainte de design ou d'ergonomie n'est imposée.
- Utiliser JAVA EE.
- Utiliser un serveur d'application Tomcat ou JBoss
- Utiliser un système de gestion de base de données au choix
- S'engager à fournir le rapport décrivant notre démarche pour le lundi 23 Mai au plus tard.

- S’engager à présenter notre pendant pendant une soutenance orale le jeudi 26 Mai.

Après deux semaines de développement nous avons pu solliciter Monsieur Natalizio pour qu’il éclaire les points qui nous paraissaient obscures, grâce à cela nous avons pu déterminer des spécifications, répondant selon nous, le plus adéquatement au besoin formulé lors de la première séance.

Chapitre 2

Spécification

2.1 Tomcat

Dans le choix du serveur d'application il nous était laissé le choix entre Tomcat et JBoss. Chaque membre de notre groupe utilisant une machine Apple et n'ayant installé ni l'un ou l'autre des solutions proposées par le cahier des charges nous avons décidé de nous renseigner sur internet afin de faire le meilleur choix par rapport à notre environnement de développement et notre système d'exploitation. Très rapidement nous avons recueilli des témoignages comparant les possibilités de chacun des serveurs d'applications. Il s'est avéré que nous avions seulement besoin d'un container de Servlet (implémentant les servlets et les JSP) pour notre projet et non d'un serveur 'full-compilant" pour JAVA EE.

Enfin la complexité de JBoss et sa faible documentation nous ont découragé à opter pour cette solution, encore plus sous MACOSX.

2.2 Eclipse

Eclipse n'était pas imposé pour la réalisation de notre projet mais vivement conseillé et nous avons suivi cette recommandation. En effet Eclipse permet de simplifier grandement la mise en place d'une application JAVA EE grâce à son outil de création de projet dynamique. Une fois associé à tomcat notre projet est utilisable directement.

Eclipse permet aussi de ré-organiser certains fichiers JAVA EE et en crée d'autres par défaut. Ainsi nous considérons Eclipse comme plus qu'un simple outil ou un simplet IDE. Il nous permet une facilité de gestion d'un projet JAVA EE qui est appréciable. N'ayant jamais fait de projets JAVA EE nous avons apprécié travailler avec Eclipse et n'avons presque rencontré aucun bug du à Eclipse lors de nos développements. Même si d'autres IDE existent nous sommes pleinement satisfaits de notre expérience Eclipse.

2.3 Bootstrap

Même s'il ne nous était imposé aucune contrainte graphique sur ce projet JAVAEE, nous avons décidé que l'utilisation du framework CSS/JS que constitue bootstrap apportait plus d'avantages que sa mise en place ne représentait de contrainte. Chacun des membres de notre groupe était familier avec ce framework et les avantages procurés en terme de recherche/pagination (voir ci-dessous) n'était pas négligeable.

Nous avons estimé qu'au vu de la facilité d'implémentation d'un bootstrap dans un projet web il était obligatoire pour nous de l'utiliser. Une application web quelle qu'elle soit aujourd'hui ne peut s'envisager sans une des contraintes minimales (qu'apportent bootstrap) en terme d'interaction et d'ergonomie.

2.4 DAO et Factory

Utilisation du design pattern factory ainsi que les DAO pour ainsi normaliser les classes modèles s'occupant des requêtes la BDD. Ainsi tous nos factory sont standards, leur utilisation est simple et efficace et le code est propre et est souvent similaire. Pour ce faire nous avons implémentés une classe abstraite DAO avec ses méthodes abstraite standards (create, update, find, delete). DAOFactory s'occupe d'instancier les différents types de DAO que nous avons créés (parcours, reponse, utilisateurs ...).

Nous avons estimé que l'implémentation de cette fonctionnalité était simple et nous

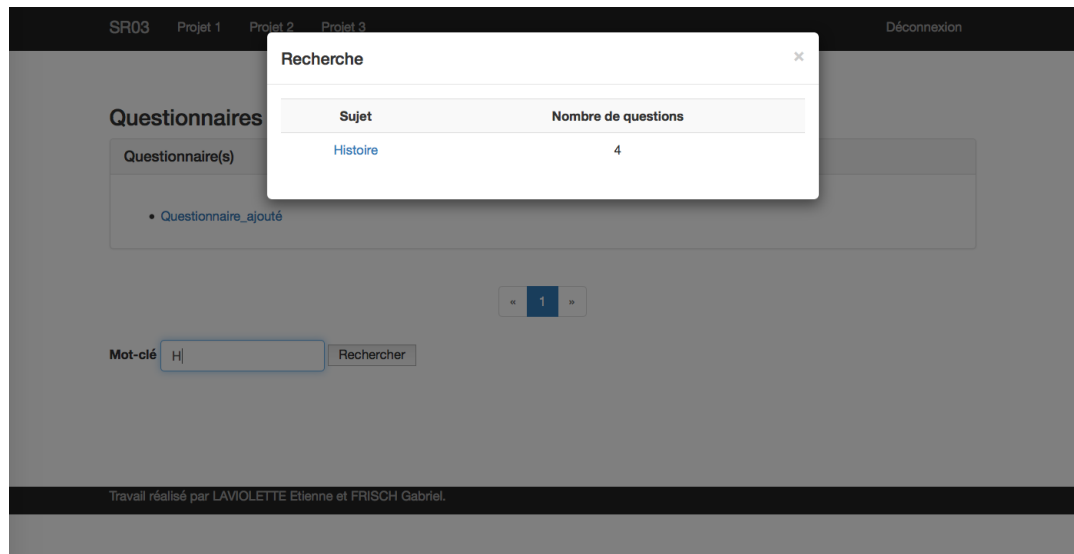
a apporté un confort inestimable tout au long du développement. Facilitant ainsi nos développement et notre interaction avec la BDD et les différentes tables.

2.5 Recherche

Pour effectuer la recherche sur les sujet de questionnaire du côté stagiaire nous avons décidé de proposer un formulaire sur la page d'accueil du stagiaire (page affiché une fois le stagiaire identifié). Une fois ce formulaire validée, une requête AJAX, vers une URL uniquement accessible depuis une requête AJAX, est lancée. La servlet va s'occuper, tout d'abord de modifier la chaîne de caractère entrée. On supprime les espaces que l'on remplace par des "%". On ajoute ces même « % » au début et à la fin de notre chaine. Cette chaine est ensuite envoyée à une méthode de notre DAO gérant les questionnaires pour effectuer une requête de TYPE LIKE sur la colonne 'sujet' de notre table 'questionnaire'. Nous avons décidé d'utiliser le moteur de recherche proposé par SQL. En effet pour nos besoins (petite BDD et application locale), cette recherche propose des garanties satisfaisante par rapport à sa facilité d'implémentation.

Nous récupérons ensuite tous les questionnaires pouvant correspondre à la chaîne entrée dans le formulaire. Grâce à Bootstrap nous les affichons dans un modal pop-up récapitulant les résultats de notre recherche (avec des liens cliquables) et le nombre de questions de chaque questionnaire proposé.

Si aucune question ne correspond on affiche un message spécifique.



2.6 Pagination

La pagination est gérée à de nombreux endroits dans notre code. Côté stagiaire : pagination des questionnaires effectués, des questionnaires non effectués, du ranking pour chaque questionnaire Côté admin : pagination des résultats pour un stagiaire données (bouton résultat pour un stagiaire

Notre pagination a été à chaque fois gérée de la même manière. Nous avons décidé d'afficher 10 éléments pour chaque page à chaque endroit où la pagination est présente. Cette valeur est une constante qui est paramétrable. L'idée est de faire une requêtes pour calculer le nombre total d'éléments à afficher. Grâce à ce nombre on en déduit le nombre de pages. Une autre requête nous permet de récupérer les 10 premiers éléments que nous affichons.

Bootstrap nous permet facilement de mettre en place une pagination efficace et user friendly. Lorsque nous cliquons sur les boutons de la pagination une requête AJAX est envoyé en passant le numéro de la page que l'on souhaite. La servlet correspondante récupérer cette valeur et va calculer les limit basse et haute (LIMIT ET OFFSET utilisés dans SQL) pour récupérer les éléments voulus (afin d'éviter les duplications ou les oublis).

Un JSP contient la structure du tableau et est appelé depuis la servlet AJAX pour

construire le tableau correspondant à la page.

Enfin dans notre page dans laquelle nous avons la pagination nous finissons notre appel AJAX par la mise à jour (dans la méthode success de \$.ajax) de la div correspondant à notre tableau.

Ranking

Classement			
Position	Nom	Score	Durée
1	Etienne	1	00:00:08
2	test	1	00:00:08
3	Etienne	1	00:00:08
4	Gabriel	1	00:00:05
5	test	1	00:00:06
6	Etienne	1	00:00:07
7	You !	1	00:00:14
8	Gabriel	1	00:00:14
9	Etienne	0	00:03:14
10	Etienne	0	00:05:06

« 1 2 3 4 »

2.7 JQuery

Par rapport à nos dires précédents sur la recherche et la pagination, puisque nous avons utilisé des requêtes ajax, nous avons estimé que la mise à jour de notre dom et sa manipulations seraient beaucoup plus aisés avec jQuery. C'est tout naturellement que nous l'avons utilisé lors de ce projet. De plus l'aisance que chacun des membres du groupe avait avec cette bibliothèques a permis de rapidement nous décider quant à son utilisation.

2.8 Mysql

Le choix du système de gestion de base de données s'est fait tout naturellement. En effet chaque membre de notre groupe a suivi l'UV NF17 et nous nous sommes accordé sur le fait que tous les outils disponibles rapidement et facilement étaient ceux qui nous pousser à opter pour la solution libre et gratuite : Mysql. En effet la présence de phpMyAdmin que nous détaillons plus loin dans ce rapport ainsi l'API JDBC partie intégrante de JAVA EE étaient des avantages déterminants dans le

choix de la solution.

2.9 Accès sections admin / stagiaires

Afin de limiter l'accès aux différentes fonctionnalités, nous avons séparé les urls de l'application en 2 grandes catégories : celles commençant par /admin et celles commençant par /stagiaire. En séparant ainsi nos urls, nous pouvons implémenter des filtres java qui vont être appelés avant chaque servlet et ainsi sécuriser l'accès.

2.10 Envoi de mails après chaque inscription

Nous avons implémenté la fonctionnalité d'envoi de mails après l'inscription d'un nouvel utilisateur. Nous envoyons notamment le mot de passe dans le mail. Nous avons pour cela utilisé le serveur SMTP fourni par l'UTC (smtps.utc.fr) avec sécurité de connexion en SSL. L'adresse de l'expéditeur est celle de l'administrateur qui effectue l'inscription.

Chapitre 3

Organisation

3.1 Répartition du travail

Avant de commencer toute considération de conception à proprement parlée pour ce projet nous avons installé Eclipse et Tomcat sur chacune de nos machines. La configuration de l'IDE et la création d'un projet dynamique nous ont occupés lors de la première heure de la première séance de TD.

La phase de conception fut la première tâche que nous avons réalisé. Ensemble, lors de la première séance de TD consacrée à la réalisation du projet nous avons pensé : aux outils que nous allions utiliser, aux beans à mettre en place et à une ébauche de la BDD que nous allions construire pour ce projet.

Ensuite nous nous sommes réparti le travail.

Le projet tel qu'il nous était présenté dans le cahier des charges présentait plusieurs volets distincts :

- Un volet réalisation des beans
- Un volet réalisation de la BDD et mise en place du design pattern factory avec les DAO correspondantes à chacun de nos tables
- Un volet : gestion de l'identification (admin ou stagiaire)
- Deux volets de développements distincts : volet admin volet stagiaire

Les trois premiers volets ont été discuté à deux. Puis Gabriel s'est occupé de les implémenter. Ce fut la première partie de notre développement. Ensuite toute

la partie admin a été confiée à Gabriel tandis que tout ce qui concernait la partie stagiaire a été réalisée par Etienne.

Nous avons enfin testé ensemble toutes les fonctionnalités implémentées puis nous avons rédigé ce rapport en concertation lors de la dernière séance de TD consacrée au projet.

3.2 Utilisation d'outils

3.2.1 Git et Gitlab

Pour chacun des membres de notre groupe l'utilisation de GIT est devenue une habitude. En effet que ce soit pour un travail personnel ou en groupe nous utilisons régulièrement cet outil de versionning et c'est tout naturellement que nous avons décidé de l'utiliser pour ce projet web JAVA EE.

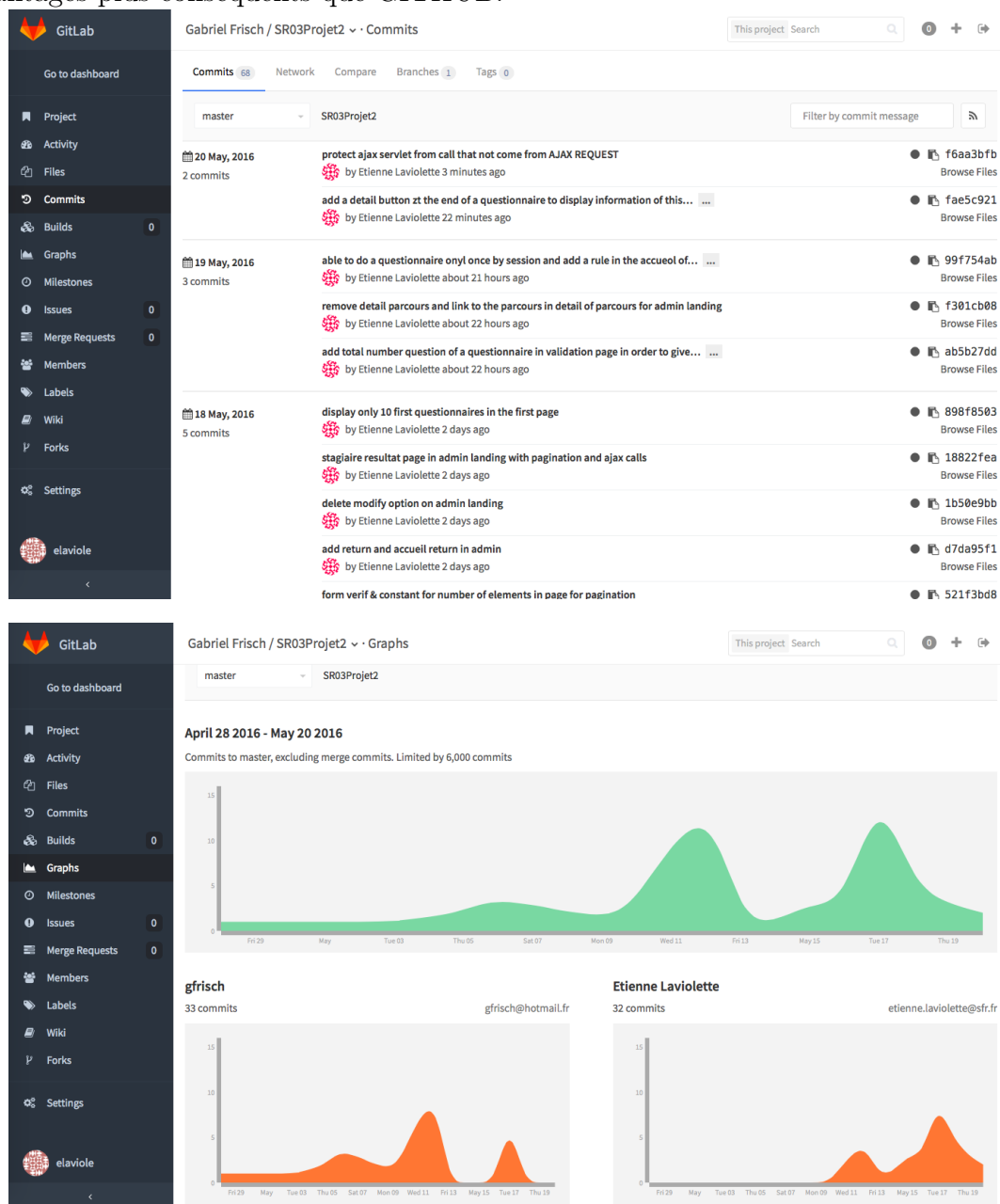
Nous avons ainsi pu travailler de manière indépendante sur ce projet collaboratif. GIT nous permettait de toujours posséder une version du projet qui fonctionne et d'aggrémenter celle-ci, petit à petit, de nouvelles fonctionnalités. Chacun des membres s'engageait à « pousser » des nouvelles fonctionnalités seulement après qu'il ait réalisé une batterie de test ne remettant pas en cause le fonctionnement global du projet et son intégrité. Le second développeur récupérait régulièrement les ajouts de l'autre et faisait ses propres tests pour valider les développements réalisés.

Le suivi des modifications apporté par GIT permettait de revenir, si un problème était détecté, à une version pleinement fonctionnelle. Nous pouvions régulièrement effectuer des corrections par petites touches.

Nous avons adopté la convention de réaliser des petits 'commit', chacun d'entre eux étant accompagné d'un message explicite en anglais.

Concernant le serveur de stockage de notre code, nous avons préféré GITLAB à GITHUB. GITLAB étant proposé par l'UTC, nous avons saisi cette opportunité. Aussi gitlab propose de créer autant de repositories privés que l'on souhaite alors que github (privé) nous imposait de mettre l'ensemble de notre code dans un repository

public. Nous avons donc privilégié la solution offerte par l'UTC et présentant des avantages plus conséquents que GITHUB.



3.2.2 Trello

Ayant tous les deux déjà mené des projets web d'envergure lors de nos différents stages ou expériences professionnelles personnelles nous avons décidé d'utiliser l'outil de gestion de projet Trello.

Cet outil se présentant sous la forme d'un tableau permet de discrétiser les tâches à réaliser pour notre projet. Nous nous sommes ensuite réparti les tâches à réaliser

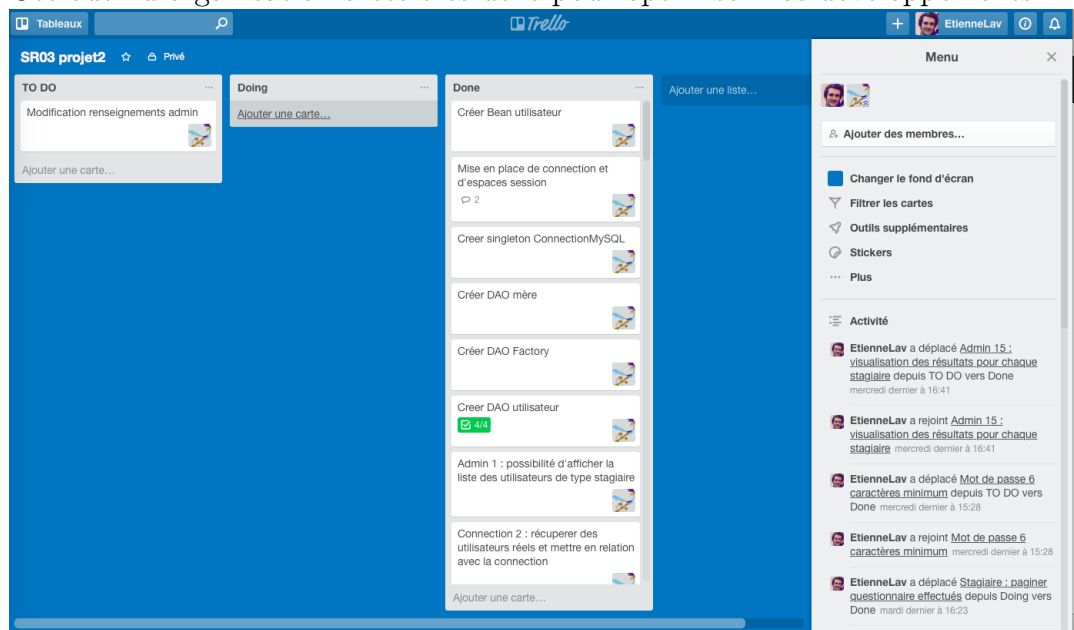
27 mai 2016

agrémentées de deadlines pour assurer un avancement linéaire du développement de notre projet.

Chaque tâche correspond sous Trello à une carte que nous plaçons dans la colonne du tableau correspondant à son état d'avancement.

Nous avons ainsi un rapide aperçu des tâches déjà réalisées, des tâches à faire, des tâches en train d'être réalisées et d'avoir des informations sur : qui le fait (?) et pour quand c'est à faire (?)

Cet outil d'organisation a été très utile pour optimiser nos développements.



3.2.3 phpMyAdmin

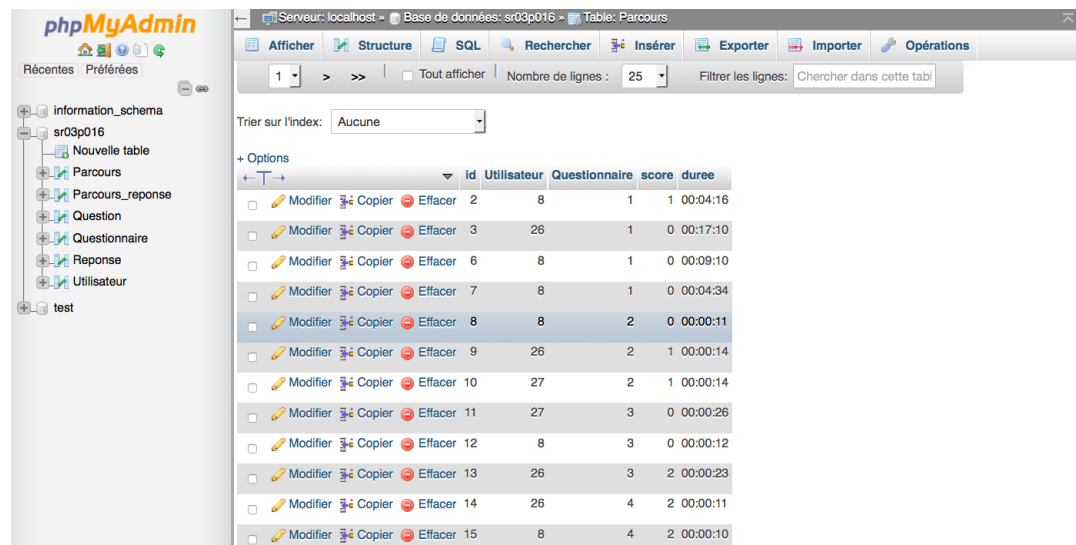
Même si la base de données que nous avons à gérer était plutôt simple, nous avons décidé d'utiliser l'outil phpMyAdmin pour rapidement avoir une vision global de nos tables SQL. En effet, cet outil est directement installé et accessible à l'url :

tuxa.utc.fr/pma

En nous connectant avec nos identifiants SR03 nous pouvions gérer intégralement notre BDD en mode graphique.

C'est pour des questions de simplicité d'utilisation (éviter une connexion SSH + requêtes en ligne de commandes) que nous avons fait le choix de cet outil.

27 mai 2016



phpMyAdmin

Récentes | Préférences

information_schema
sr03p016
Nouvelle table
Parcours
Parcours_reponse
Question
Questionnaire
Reponse
Utilisateur
test

Serveur: localhost - Base de données: sr03p016 - Table: Parcours

Afficher Structure SQL Rechercher Insérer Exporter Importer Opérations

1 > >> Tout afficher Nombre de lignes : 25 Filtrer les lignes: Chercher dans cette table

Trier sur l'index: Aucune

+ Options

		id	Utilisateur	Questionnaire	score	duree
<input type="checkbox"/>	Modifier Copier Effacer	2	8	1	1	00:04:16
<input type="checkbox"/>	Modifier Copier Effacer	3	26	1	0	00:17:10
<input type="checkbox"/>	Modifier Copier Effacer	6	8	1	0	00:09:10
<input type="checkbox"/>	Modifier Copier Effacer	7	8	1	0	00:04:34
<input type="checkbox"/>	Modifier Copier Effacer	8	8	2	0	00:00:11
<input type="checkbox"/>	Modifier Copier Effacer	9	26	2	1	00:00:14
<input type="checkbox"/>	Modifier Copier Effacer	10	27	2	1	00:00:14
<input type="checkbox"/>	Modifier Copier Effacer	11	27	3	0	00:00:26
<input type="checkbox"/>	Modifier Copier Effacer	12	8	3	0	00:00:12
<input type="checkbox"/>	Modifier Copier Effacer	13	26	3	2	00:00:23
<input type="checkbox"/>	Modifier Copier Effacer	14	26	4	2	00:00:11
<input type="checkbox"/>	Modifier Copier Effacer	15	8	4	2	00:00:10

Chapitre 4

Architecture

Dans ce chapitre, nous allons détailler l'architecture de notre projet ainsi que l'architecture de la base de données que nous que nous avons implémenté.

4.1 Base de données Mysql

Avant de construire notre base données, nous avons commencé par réaliser un schéma UML résumant les liens entre les différentes entités.

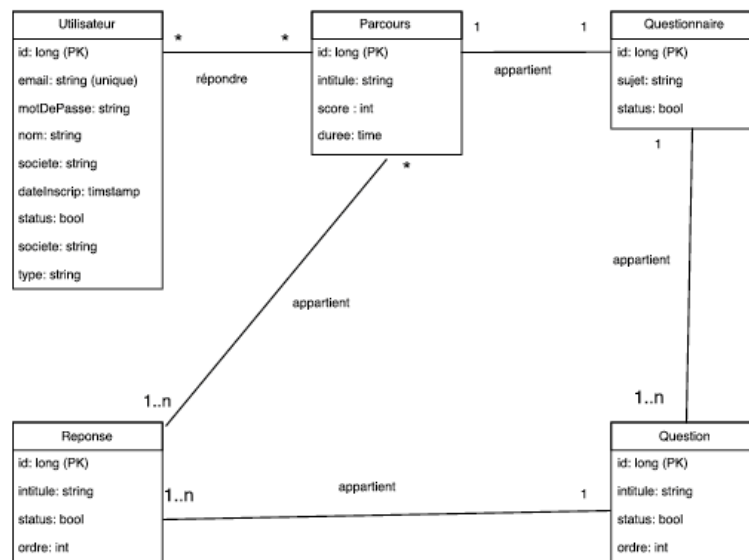


FIGURE 4.1 – Schéma UML décrivant les relations entre les entités

Après analyse nous créons les tables suivantes :

- Questionnaire (long id PK,String sujet,Boolean status)
- Question (long id PK ; String intitule, Boolean status, int ordre, int questionnaire FK)
- Reponse (long id PK ; String intitule, Boolean status, Boolean correct, int ordre, int question FK)
- ReponseParcours (long id PK, int parcours FK, int reponse FK)
- Utilisateur (long id PK, String email UNIQUE, String motDePasse, String nom, String societe, String telephone, Timestamp dateInscription, Boolean status, String type)
- Parcours (long id PK, int utilisateur FK, int questionnaire FK,int score,Time duree)

Pour la création de nos tables, nous avons utilisé PhpMyAdmin.

4.2 Division MVC

Pour notre projet, nous avons utilisé le modèle d'architecture Modèle Vue Controlleur (MVC). Ce modèle d'architecture consiste à séparer la vue (couche de présentation à l'utilisateur), l'accès aux données et la couche métier. Le but de cette architecture est d'obtenir une dépendance minimale entre les différentes couches de l'application.

Dans le cas d'une application JEE, le modèle correspond aux beans et aux DAO, la vue correspond aux fichiers JSP et les contrôleurs correspondent aux Servlets.

4.3 Les différents design pattern utilisés

Les design patterns ou modèles de conception sont des organisations pratiques de classes objets.

Dans notre projet, nous avons implémenté le design pattern Factory pour la création de nos DAO. La Factory est une classe qui n'a pour rôle que de construire des

objets. Cette classe utilise des classes abstraites pour masquer l'origine des objets, dans notre cas, la classe abstraite DAO.

Nous avons aussi utilisé le design pattern Singleton. Un singleton restreint les instantiations d'une classe. Puisque le singleton limite le nombre d'instance en mémoire, il permet d'améliorer les performances et évite de gaspiller de la mémoire. Dans notre projet, nous avons utilisé le singleton pour les classes DAOFactory et ConnectionMySQL.

4.4 Schéma récapitulatif de notre architecture

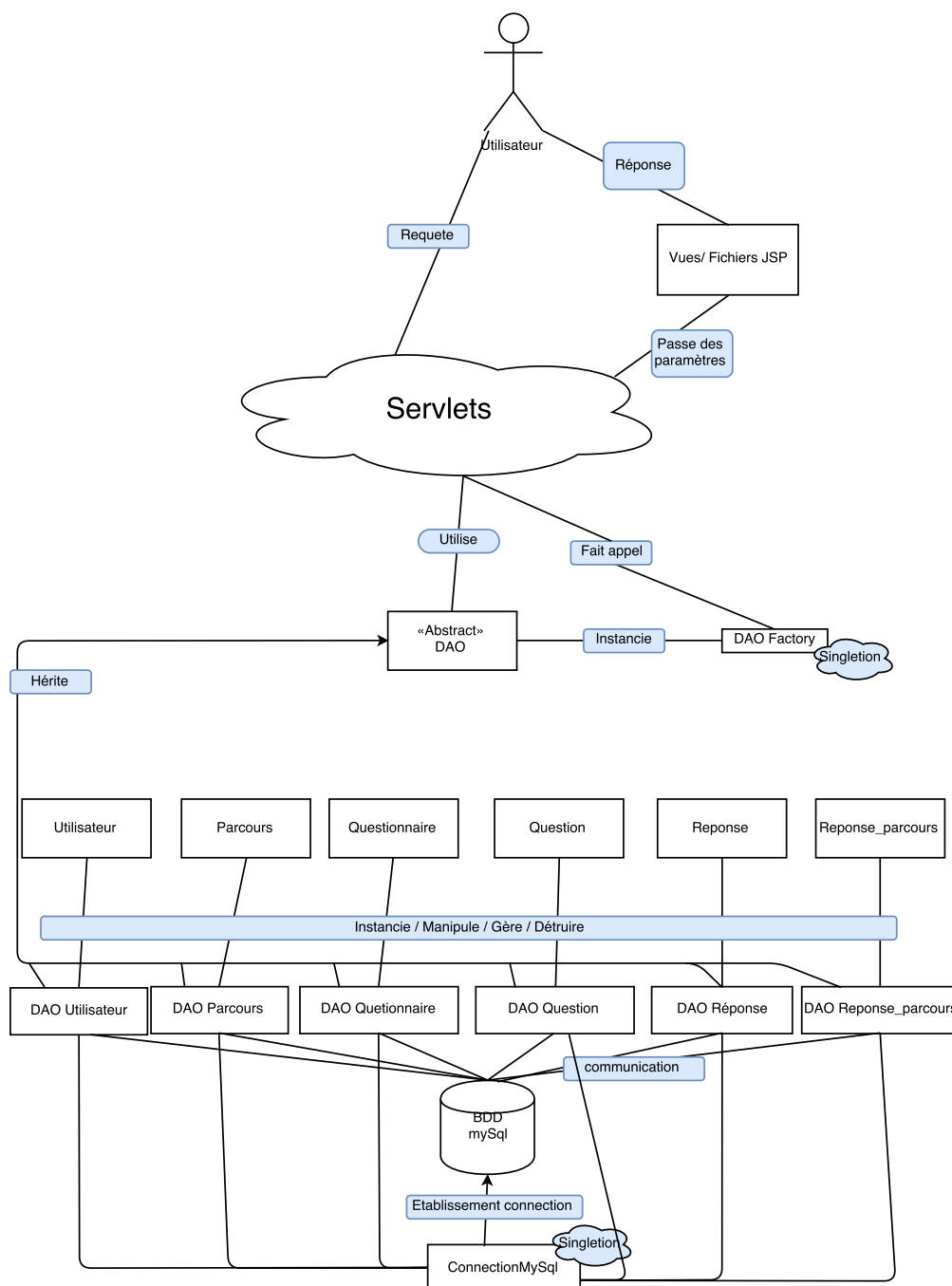


FIGURE 4.2 – Schéma résumant l'architecture mise en place pour le projet

Chapitre 5

Idées d'améliorations

5.1 Amélioration de la BDD

- Faire des triggers sur la base de données :
exemple : vérifier régulièrement qu'une question créée sans réponse ajoutée passe à status = false
exemple : vérifier régulièrement qu'une question créée sans réponse correcte ajoutée passe à status = false
Même genre de gestion avec les questionnaires
- Scinder notre BDD utilisateurs en 2 BDD : admin et stagiaires et ainsi enregistrer des éléments plus pertinents sur les deux groupes d'utilisateurs
- Les mots de passes sont stockés en clair dans notre BDD sans être hashés ou cryptés. L'ajout simple et peu coûteux d'un hashage de type md5 avant le stockage en base et l'opération inverse pour la vérification des mots de passe devrait être envisagé sérieusement si l'application que nous avons codé passait en production.

5.2 Amélioration des fonctionnalités proposées

- Améliorer notre recherche : aujourd'hui on prend une chaîne de caractère, on rajoute "%" au début et à la fin et à la place de chaque espace et on exploite la recherche SQL. Or une indexation avec Elasticsearch ou Solr permettrait

des recherches plus poussées sur des colonnes spécifiques. En effet la recherche SQL n'est pas performante dès que la base atteint une taille importante. Nous aurions pu ajouter une regex pour parser notre chaîne de caractère et ainsi rendre notre recherche plus efficace

- Rajouter des fonctionnalités côté admin ou stagiaire. Rajouter la date à laquelle on effectue un questionnaire pour un stagiaire. Proposer une évolution de ses scores entre deux dates sur un même questionnaire.
- Créer un nouveau filtre pour pouvoir exploiter les résultats aux questionnaires ; Quelqu'un qui ne serait ni admin ni stagiaire mais un gérant qui aurait beaucoup d'informations statistiques sur les questionnaires sans pouvoir les modifier. Avoir par exemple temps moyen , nombre de bonnes réponses, moyennes...
- Proposer d'enregistrer un questionnaire en cours et de reprendre le parcours de ce questionnaire ultérieurement. Ainsi on aurait une nouvelles catégorie : les questionnaires en cours. On reprendrait exactement à la bonne question tout en gardant les résultats précédents enregistrés. Pour cela on pourrait stocker dans un cookie à la déconnexion d'une session d'un stagiaire les différents questionnaires en cours. Et ainsi lors de la future connexion d'un même utilisateur on récupérerait toutes ses données.

5.3 Amélioration de notre code

- Quelques appels Ajax réalisent des fonctions presque similaires : appel Ajax côté stagiaire pour paginer et afficher la pagination des questionnaires déjà parcourus est similaire à l'appel Ajax côté admin pour afficher les résultats d'un stagiaire. Nous aurions pu éviter la duplication de ce code en créant un package de servlet Ajax et en mettant notre JSP ni du côté admin ni du côté stagiaire et ainsi ne pas se préoccuper du filtre qui empêcher l'accès à la partie stagiaire si on est admin et vice-versa.
- Les méthodes de nos factory sont parfois similaires (à une contrainte d'ORDER BY près). En effet nous aurions pu créer des méthodes parentérales et

presque similaires pour ainsi éviter la duplication de code.

- L'utilité de nos managers est avéré niveau découplage des « types » de code et des fonctionnalités qu'elles représentent ; cependant la pertinence du point de vu MVC est à méditer. En effet nos manager prennent une requête en paramètre et renvoie une requête en sortie. Se découpler totalement des requêtes serait idéal.
- On a passé la plupart de nos paramètres en GET, une méthode peut sure et sur laquelle on peut attaquer facilement notre application en lui donnant des informations erronées et bien choisies.

Conclusion

Au début de ce projet de SR03, aucun des deux membres de notre groupe n'était à l'aise avec JAVA EE et Eclipse. Au contraire même, nous n'avions jamais utilisé ce langage et cet IDE. Nous avons tous les deux déjà réalisé plusieurs projets web mais toujours en PHP ou JS, ce fut donc une découverte enrichissante. Le couple Eclipse/JAVA EE est très efficace et permet rapidement de monter une application web complète, dans notre cas, en interaction avec une base de données. Eclipse est une véritable valeur ajoutée et permet de rendre plus aisée l'architecture projet imposée par JAVA EE. Chacun des membres de notre groupe ayant déjà travaillé avec le framework Symfony en PHP, nous avons retrouvé beaucoup de similitudes qui nous ont aidées. Ce projet fut donc une occasion parfaite pour étendre nos connaissances en programmation web tout en réutilisant au maximum nos backgrounds personnels.

Chapitre 6

Sources

Tout au long du développement de notre projet nous avons utilisé activement les différentes sources disponibles sur internet et plus particulièrement celles-ci :

- *[http : //getbootstrap.com/css/](http://getbootstrap.com/css/)*
- *[http : //www.w3schools.com/html/default.asp](http://www.w3schools.com/html/default.asp)*
- *[http : //www.w3schools.com/bootstrap/bootstrap_pagination.asp](http://www.w3schools.com/bootstrap/bootstrap_pagination.asp)*
- *[http : //botmonster.com/jquery – bootpag/.Vzr8y1cQqQt](http://botmonster.com/jquery-bootpag/.Vzr8y1cQqQt)*
- *[https : //www.npmjs.com/package/bootpag](https://www.npmjs.com/package/bootpag)*
- *[https : //www.futurehosting.com/blog/jboss – vs – tomcat – choosing – a – java – application – server/](https://www.futurehosting.com/blog/jboss-vs-tomcat-choosing-a-java-application-server/)*
- *[https : //openclassrooms.com/courses/creez – votre – application – web – avec – java – ee](https://openclassrooms.com/courses/creez-votre-application-web-avec-java-ee)*
- *[http : //java.developpez.com/cours/?page = java – ee – cat](http://java.developpez.com/cours/?page=java-ee-cat)*

