Rapport de projet Jee

Phong Vu NGUYEN
Nicolas AMBLARD
Clément DESCAMPS
Cédric LE LOGEAIS
Yuelin WANG

Informatique-2ème année
Année 2016-2017

I. Description du squelette de l'application

Au cours de l'implémentation de ce projet, nous avons été amenés à séparer les tâches en 2 parties : La partie monitoring, responsable de l'affichage du contenu de la base de données pour l'utilisateur, et la partie insertion de pannes, capable de limiter la quantité minimale de données que l'utilisateur doit insérer pour générer une entrée et de lui permettre d'automatiser un remplissage de la base.
Ces deux tâches ayant des rôles bien distincts, nous avons choisi que l'utilisateur puisse se rendre directement sur la page qu'il souhaite et avons donc rédigé deux couples jsp/servlet afin de ne pas procéder avec une seule page et des gestions de cas complexes qui rendraient le code illisible. La partie monitoring quant à elle fait appel à un servlet tiers afin de pouvoir gérer les rafraîchissements partiels.

Notre partie métier est calquée sur la structure entourant la classe Book codée lors du TP, cependant, l'interface BookService ne faisant que faire suivre les demandes au BookDAO n'a pas eu droit à son adaptation. Nous avons donc une classe Panne, une interface PanneDAO et la classe PanneDAOImpl l'implémentant.

Pour ce qui est JavaScript et CSS, nous avons créé un dossier dans le WebContent contenant le JavaScript pour l'un et le CSS pour l'autre. Nous avons laissé les fichiers JSP à la racine du WebContent

La base de données a été créée avec une seule table : table_panne. Elle contient l'id de la panne, le nom de la machine, le type de la panne, le type de la machine, la date de la panne et si elle a été résolue ou non.

II. Choix techniques

Pour l'implémentation du rafraîchissement, nous avions des formulaires avec des posts au début, qui étaient utilisés après un temps que nous avions définis. En passant par ajax pour que le rafraîchissement soit partiel, nous avons retiré tous les formulaires et avons remplacé le rafraîchissement par une fonction qui est appelée à partir d'un clic en jQuery. On sauvegarde les paramètres du dernier bouton utilisé et ils sont réutilisés dans la fonction refresh().

Pour gérer les accès à la base de données, nous avons utilisé les primitives fournies dans le DBManager afin de nous ménager un accès à la base SQL tournant sur la même machine que le serveur.
Autour de ces primitives, nous avons alors utilisé les classes Connection, Statement, ResultSet du package java.sql pour émettre directement vers la base de données nos commandes en langage SQL tout en s'assurant d'utiliser executeUpdate et executeQuery en accord avec le type de requête que l'on envoie. À la fin, nous n'oublions pas de fermer la connexion à la base de données, sans le programme aura un problème après un grand nombre de rafraîchissement.

Dans le cadre de la partie optionnelle de créer un graphique avec les pannes du jour, nous avons décidé d'utiliser la librairie JavaScript Chart.js, qui nous a permis de réaliser des graphiques facilement.

Afin de pouvoir ajouter des pannes en rafale et en cascade, nous avons implémenté une sous-classe de Thread qui permet au servlet de renvoyer à l'utilisateur une nouvelle page sans attendre que le remplissage de la base de donnée soit fini pour lui rendre la main. Cette classe, en recevant comme argument de délai 0, procède à l'envoi en rafale, et sinon, fait un envoi en cascade.

Finalement, nous avons choisi de créer un servlet pour gérer tous les appels ajax de la page de monitoring, nous aurions pu gérer cela avec un servlet pour chaque type d'interaction mais, nous avons préféré centraliser les appels ajax.

III. Difficultés rencontrées

Lors de l'implémentation, nous avons rencontré des difficultés sur différents points. L'un de ces points était l'implémentation du niveau 42. Nous avons eu différentes idées, qui consistaient à créer un DB listener, cependant ce n'était pas pratique en mysql et nous n'avions pas le temps nécessaire pour le faire. Une autre idée était de lire les logs et de faire un refresh à chaque nouvelle entrée qui concernait la mise à jour de la table. Ensuite, nous ne savions pas trop comment faire que la page ait les informations obtenues en java même si nous avions réussi à les récupérer.

Autrement dit nous n'avons pas vraiment rencontré de difficulté et si nous en avons eu, nous avons pu les corriger.