Institut Paul Lambin

BAC 2 Informatique de gestion

AAM

Synthèse AAM

Auteurs :
Christopher Sacré

Professeur :
L. Leleux

20 mai 2017

Table des matières

1 Schémas vus au cours

1.1 Semaine 2

1.1.1 Diagramme

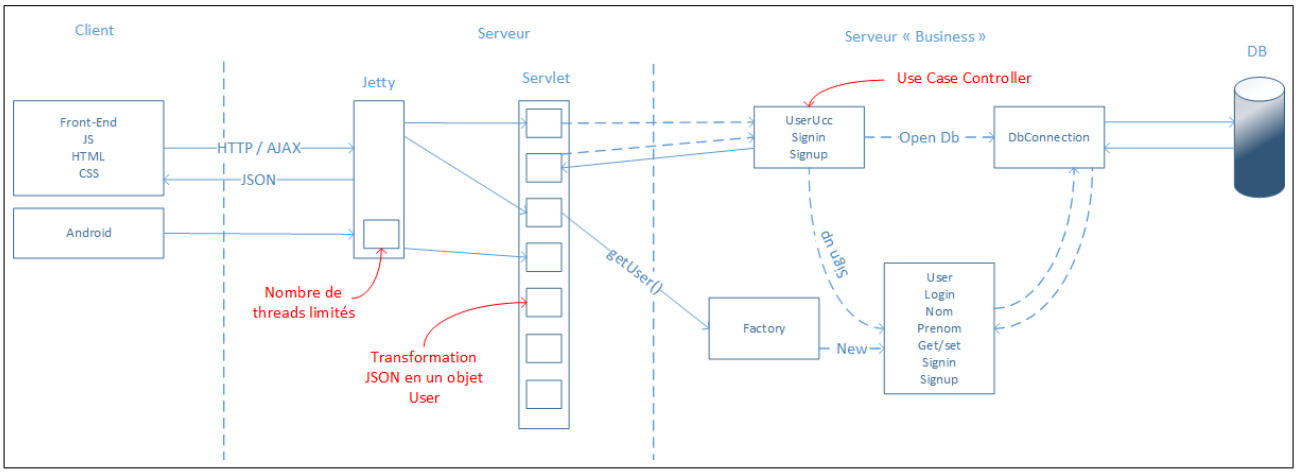

FIGURE 1 – Introduction du cours

1.1.2 Informations supplémentaires :

Si vous souhaitez des informations supplémentaires, plusieurs logiciels ont été mentionné durant ce cours : Harmony, Visual Studio Code ainsi que Electron. (Il s'agit d'un cours nous introduisant les concepts de notre application ainsi que les bases de son architecture).

1.2 Semaine 3

1.2.1 Diagramme

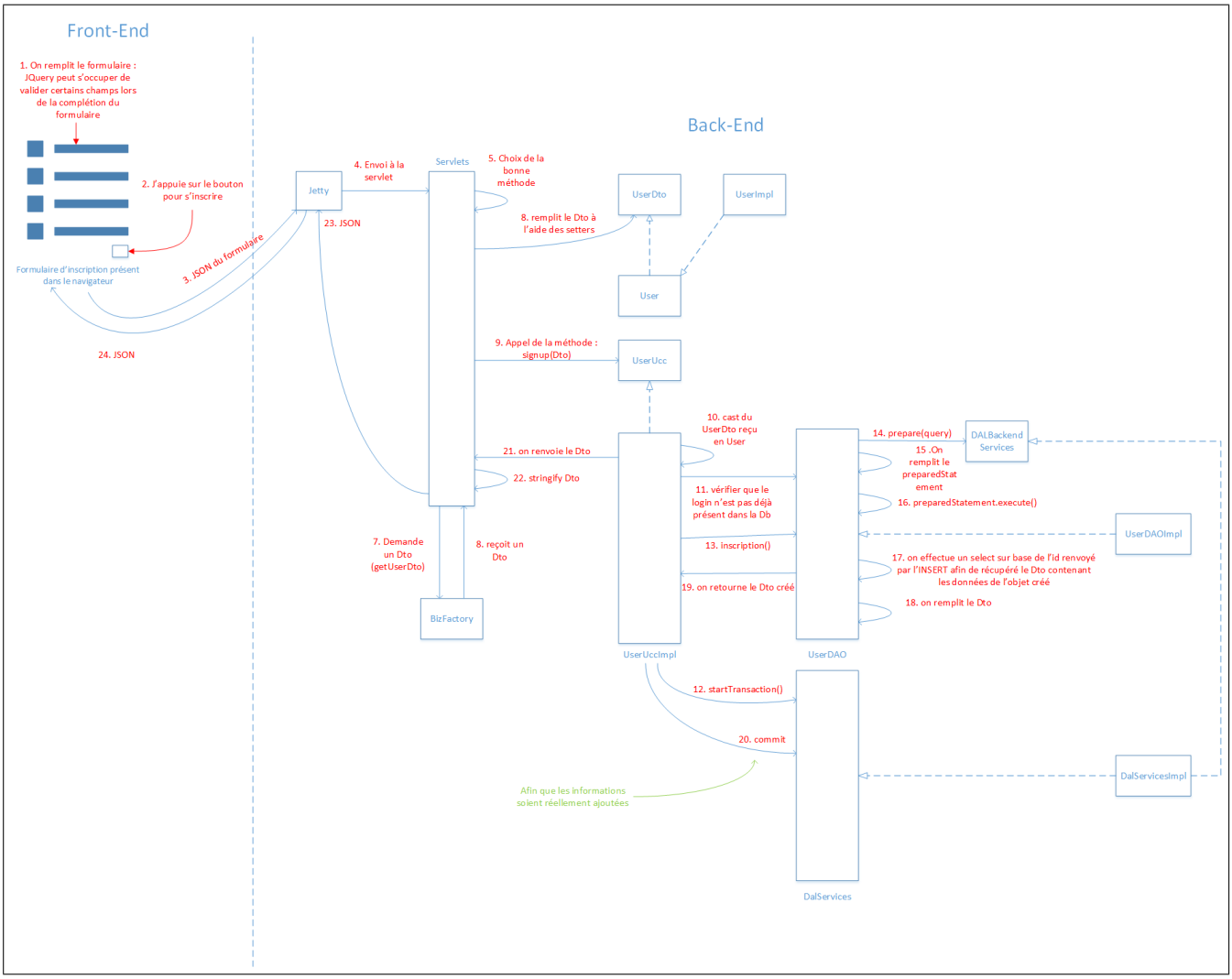

FIGURE 2 – Exemple d'utilisation de notre architecture (UC : s'inscrire)

1.3 Semaine 4

1.3.1 Diagramme

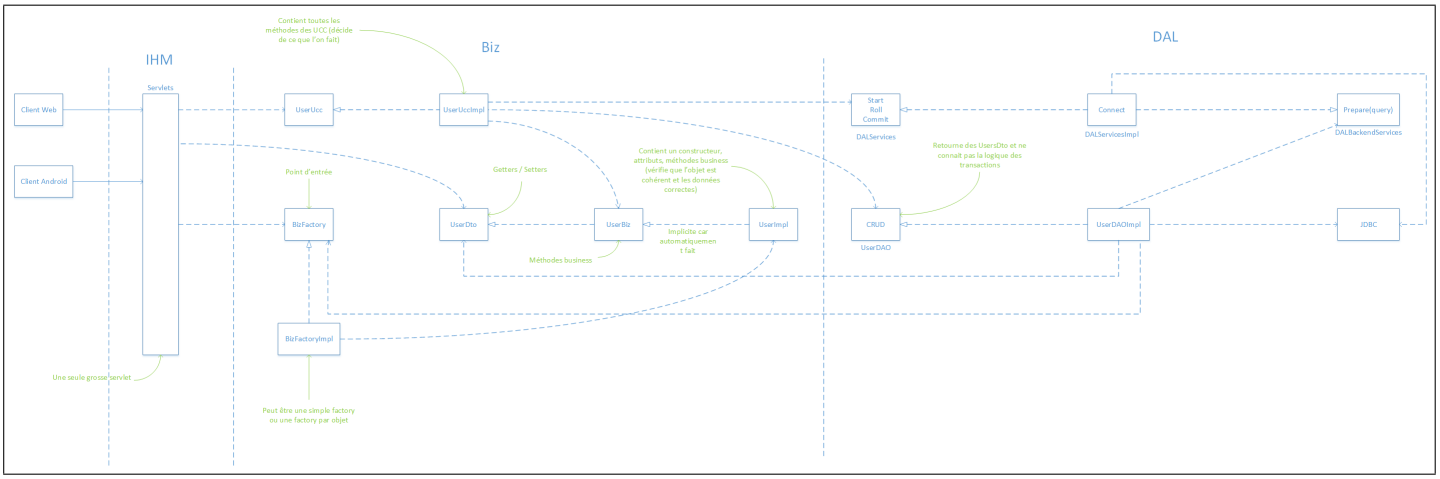

FIGURE 3 – Architecture Monothread complète

1.3.2 Informations suppémentaires :

Factoring : Il faut savoir que l'on mettra de nombreuses classes, ... public mais on tentera d'utiliser uniquement les points d'entrées (surtout au sein de la couche IHM), dans le même style, on utilisera de nombreuses interfaces afin de diminuer le nombre de dépendances concrètes et faciliter le changement entre l'environnement de prod et celui de dev. Par ailleurs, la servlet ne doit pas connaitre les méthodes bussiness (Car dans le cas contraire elle pourrait les modifier).

Traitement des INSERT : lorsque l'on insère, on tente de retourner l'entièreté de l'objet crée et non juste l'id, cela permet de vérifier que tout c'est bien passé.

Traitement des SELECT : point suivant est plutôt controversé et dépend de votre version des choses il s'agit de l'utilisation de transactions au niveau des select : on peut ne pas utiliser de transaction (perte de cohérence mais gain de performances) ou au contraire utiliser des transactions (perte de performances mais gain de cohérence).

Sécurité : Il est important que le back-end soit totalement indépendant du front-end (les données reçues ne sont en aucun cas sure).

Remarque : un framework nous pose des rails, cela nous permet uniquement de nous orienter , dès lors il faut parfois s'éloigner des rails.

1.4 Semaine 5

Main : Le main permet de démarrer Jetty, c'est lui qui s'occupe de créer les servlets Jetty. Il crée les Factory à l'ai de l'injection de dépendance. Dés que tout cela est créé, il se met en pause (Le main se lance au démarrage de l'application).

1.5 Semaine 6

1.5.1 Diagramme

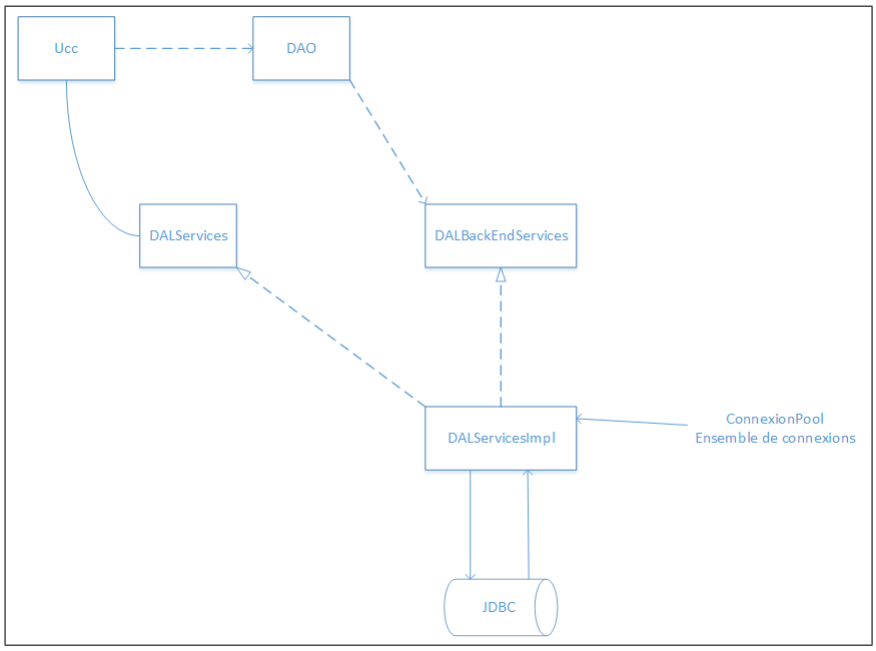

FIGURE 4 – Connexion Pool

1.5.2 Informations

Introduction : gros problème que l'on a tenté de résoudre est le fait que l'on ne pouvait ouvrir qu'une seule transaction à la fois (Il est totalement interdit d'ouvrir plusieurs transactions sur la même connexion en même temps, En cas de transactions imbriquées, si la première transaction à un soucis le rollback ne se propagera pas, mais s'arrêtera au premier commit).

Solution : Jetty à pour cela, un ensemble de Thread (Il n'en a pas un nombre infinis). On va donc remplacer notre précédente unique connexion par un ensemble de connexions (Le nombre de connexions à avoir est de notre ressort, et c'est donc à nous de faie ce choix).

Aide à la solution : On peut afin de nous aider à implémenter cela utiliser ThreadLocal (On aurait dès lors une connexion par thread, utilisation d 'une Map¡Thread, ConnexionDb¿ (pour cela on peut utiliser la méthode Thread.getId() qui renvoie l'id du threadCourant), ou plus facile encore ou pourrait utiliser la classe ThreadLocal, notre Map deviendrait alors un ThreadLocal¡Connexion¿, la clé est directement l'id du thread local (il s'agit tout simplement d'une map pour laquelle on a spécifié la syntaxe)).

Problème : le problème avec ces deux solutions vient du fait que si il y a une connexion par thread, on va donc multiplier le nombre de connexions (La taille du threadPool est un problème en soit (il faut pouvoir la fixer et cela dépend la plupart du temps fort de la couche Bussiness (Si il n'y a pas assez de Connexions, cela bloquera le thread tant qu'il n'y en aurait pas une de disponible).

Solution Finale : Afin de palier à tout cela, on va utiliser DBCP2 (Une librairie Java, DataBaseConnexionPool).

1.6 Semaine 7

1.6.1 Informations

Affichage des erreurs : Afin d'afficher des erreurs on peut utiliser System.err.println (La différence avec System.out.println est qu'il pourrait y avoir un décallqge au niveau de l'affichage).

Classe Logger : La classe Logger permet de définir une priorité au niveau des messages (Il plusieurs niveaux de priorité : SEVERE (valeur la plus élevée), WARNING, INFO, CONFIG, FINE, FINER, FINEST (valeur la moins élevée). L'avantage de cela est que l'on pourrait Logger sur un autre serveur , permettant ainsi d'empêcher la surcharge d'un appareil (Il est intéréssant de Logger : le temps de réponse, le type d'appareil utilisé , le type de route utilisé, ou tout ce qui vous semble intéressant d'être connus).

Profiler - SQL : permet de connaitre le temps d'exécution d'une réponse.

Sécurité : On pourrait rajouter une couche afin d'empêcher les attaques DDOS (Cette couche permettrait de filtrer les requêtes et de les dispatcher à des serveurs plus petit qui traiteraient alors les requêtes). Un autre point important pour la sécurité est qu'il ne faut jamais faire confiance à l'utilisateur, Il faut par ailleurs faire la distinction entre Admin et Utilisateur et utiliser JWT (cf cours de JavaScript).