Environnement du travail

Moncef BEN RAJEB

August 19, 2013

1 Introduction

Certes, le sucées ou l'échec d'un projet informatique dépend du choix des technologies utilisés. Ce choix dérive essentiellement des objectifs à attendre et des contraintes d'accompagnement qui doivent être prisent en considération.

Dans cette partie je vais présenter les différentes technologies envisageables par les architectes de Stample et relatives à la développement du Backend la partie sur la quel j'ai travaillé et le FrontEnd. Vous trouverez plus de détailles sur l'environnement du travail sur ma page[1] web.

2 Working tools

2.1 Environnement de Développement

- PlayFramework[2] 2.1.0
- JDK 7
- Maven3
- OS X 10.8.4
- SBT[3] 0.12.4
- ElasticSearch[4] 19.4.0
- MongoDB[5] 2.2.x

[1] http://ideas2d.com/moncef/home.php
[2] http://www.playframework.com/
[3] http://www.scala-sbt.org/
[4] http://www.elasticsearch.org/
[5] http://www.mongodb.org/

2.2 Langages de Programmation

- BackEnd : SCALA , JAVA, JSON
- FrontEnd : HTML5, JavaScript, Jquery, CSS3, JSON, Ajax.

2.3 Plugins

- BackEnd : SecureSocial, Salat,
- FrontEnd :Backbonejs

2.4 Outils de développement

- IDE : ItelliJ IDEA 12.1.14 Community Edition;
- Base de donnée : Mongodb NoSQL data base;
- outils de compilation automatique : sbt.

3 Aspects technique de Stample

3.1 Architecture

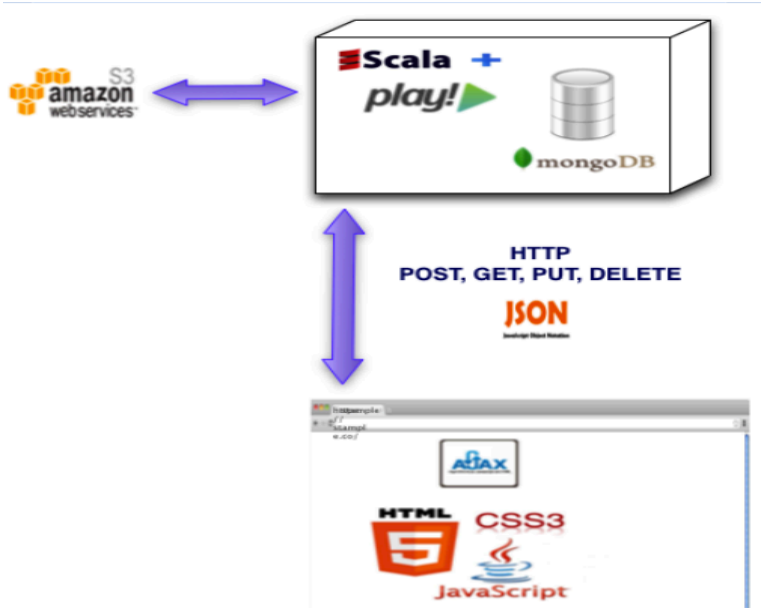

Figure 1: Architecture de la plateforme

3.2 Description de l'architecture globale

Les requêtes HTTP sont faites en JSON. Le JSON est désérialisé dans des objets métiers (Scala case class) pour être manipulés au sein de l'application.

3.2.1 Communication entre la base et l'API

On utilise le driver Casbah MongoDB en Scala et une librairie "Salat" qui permet une bonne intégration de Casbah dans l'application. Salat permet de sérialiser les objets métiers (case class) en BSON [6], cette libairie propose un DAO (Data Access Object) générique un model companion qui expose les méthodes basique des requêtes Mongo (find(), findById(), search()...)

3.2.2 Communication entre L'API et ElasticSearch

L'indexation dans la base se fait manuellement après l'insertion, la mise à jour d'un document MongoDB dans la base de donnée. L'échec d'indexation est non bloquant car il est ratrappé par un Job.

ElasticSearch prend en entrée du JSON cela implique qu'il fontionne particuliérement bien avec Mongo. De plus, utilise salat aussi pour produire le JSON d'un objet métier (case class). Donc nous avons le même document dans MongoDB et ElasticSearch. Cela permet de rendre un document similaire à Backbone pour la recherche. On utlise un client JAVA pour ElasticSearch.

3.2.3 SBT

Le build est réalisé par SBT, il faudrait mettre en place un Jenkins qui fera tourner les tests unitaires et embarqués à chaque push pour réparer les regressions et que nous en soyons alertés.

3.2.4 Les collections MongoDB

Stample structure les données sous forme de diffirents collections. Ils Sont tous identifier par un ObjectId, elle regroupe tout les informations nécessaire sur les utilisateurs (Compte utilisateurs, Stamples ...).

3.3 Configuration de départ

3.3.1 Description de la configuration

Le projet contient de quatre répertoires séparer : fixturedatabase contenant les fichiers JSON et leurs sérialisation binaire encodé BSON, restore inclu le build de l'application et les plugins, stample-search contenant les classes java et la configuration nécessaire pour

[6] http://bsonspec.org/

le lancement du projet et stample-web contient les diffirents packages du Front/back End.

Pour l'interraction avec le projet le permier pas est l'installation des outils cités auparavant. ensuite, j'ai configuré ma machine avec les paths nécessaires pour lancer sbt, play, mongo et elasticSearch. Puis, j'ai lancé le mvn dans le repartoire stample-search. Enfin il est indisponsable de lancer play ou sbt et de générer les fichier idea, eclipse ou autre dans stample-web avec l'IDE envisagé.

3.3.2 Gestion du travaille en groupe

Les régles du travaille en groupe sur le projet :

1. NE PAS travailler directement sur la branche master. Ce n'est que pour la production.
2. La branche de développement principale est "preProd". A partir de cette branche on commence une nouvelle branche.
3. Lors du développement sur une branche, merger régulièrement preProd en elle, pour s'assurer d'être à jour avec le travaille des autres développeurs.
4. Lorsque la branche est stable et a été testé par le chef de projet, il va tester, merger dans preProd, puis on peut supprimer cette branche.
5. Utiliser toujours camelCase pour nommer les branches et des noms intelligents!
6. Ajouter des explications intelligents lors de chaque commit.

4 Conclusion

Dans cette partie j'ai détaillé les outils qu'on utilises. Ensuite, j'ai présenté l'architécture globale de la plateforme et les diffirentes interractions. La prochaine section se focalise sur la solution développer pour le login sur Stample.