Back End marketing conversationnel
Technologies web côté serveur

Paul Orhon

LP – MiAR – Université de Nantes

23 novembre 2017

Table des matières

Table des figures

1 Projet

Dans le cadre du module de technologies web côté serveur, il a été demander de créer une partie du back end d'un marketing conversationnel.

Pour cela il va falloir développer différent services pour pouvoirs mettre en place ce système.

2 Contexte du projet

2.1 Objectifs

L'objectif est donc de développer une partie du Back End d'un marketing conversationnel, en exposant les service avec une api REST.

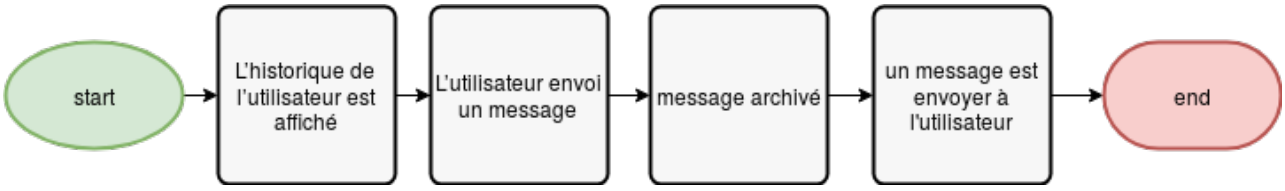

FIGURE 1 – Workflow classique d'un utilisateur

La partie à développer est l'envoi des messages et les sauvegardes des données. Ces sauvegardes sont utilisé pour conserver les messages, qui seront daté, afin d'afficher l'historique à l'utilisateur et pouvoir effectuer des statistiques anonyme sur toute les conversation.

Le projet doit être séparer en sous module, cela permettra de manipuler, modifier et diffuser plus facilement.

2.2 Contraintes

1. Le projet est à réalisé seul.
2. Le programme sera développé sur la plateforme Java.
   — Il sera programmé en Java pour des raisons de connaissance.
3. Le rapport est à rendre le 23 novembre 2017 à 17h50.
4. Le projet est à rendre le 5 décembre 2017 à 12h20.
5. Le système doit sauvegarder les messages pour réaliser des statistiques et pour l'historique des utilisateurs.
6. Le système doit être testé et testable sur le poste du développeur.
7. Le couplage doit être lâche.

3 Les vues

3.1 Vue globale du projet

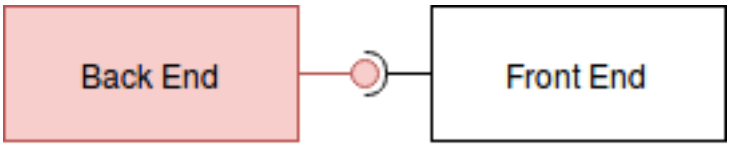

FIGURE 2 – Diagramme de la séparation du Back / Front End

Comme dit précédemment, seule le back end et son interface sera développer (partie en rouge de la figure 2).

3.2 Vue logique du Back End

Le back end sera diviser en plusieurs modules permettent ainsi de diviser les tâches, de réaliser un couplage lâche et de faciliter le maintien du système.

Cette représentation repose sur le *domain-driven design*[1]. Le but du DDD est de former le code autour de notion métiers.

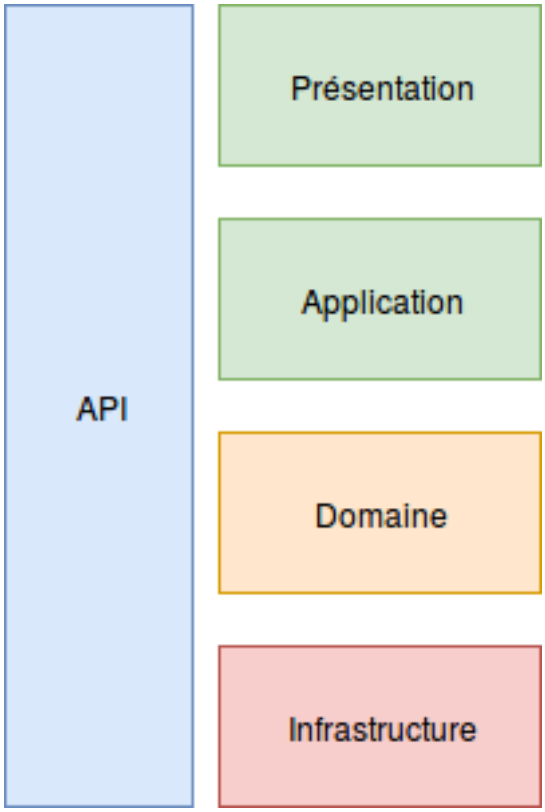

FIGURE 3 – Diagramme de la vue logique du back end

Ici chaque couleur correspond à un module.

1. *domain-driven design*(DDD) : en français *conception pilotée par le domaine* est une approche de la conception de logiciel. Plus d'information : `https://en.wikipedia.org/wiki/Domain-driven_design`

Application contient la **présentation** du fait de l'utilisation du framework *Spring* qui permet une mise en place simple d'une api REST et donc de l'interface du back end. Sont rôle est de récupérer les requête arrivant à l'interface, de les envoyer au bon service et de répondre.

Domaine correspond au *services* de l'application. Contient aussi les interface, pressant dans l'**API** et implémenté dans l'**infrastructure**, utile à ces services.

Infrastructure correspond à l'implémentation des différentes ***interfaces*** et ***classe abstraites*** qui sont les factory. Il gère le stockage persistant des données.

API contient les ***classe*** communes aux modules. Cela permet notamment au **domaine** d'utiliser une classe et de laisser l'implémentation à l'**Infrastructure**.

4 Implémentation

4.1 Les Services

Le back end va donc proposer différant services.

4.1.1 L'envoi de messages

Un service proposera au client d'envoyer un message ver le serveur. Ce message sera transmis au personnes concerner. Il sera aussi sauvegarder. Cette sauvegarde est utile pour l'historique de conversation de l'utilisateur (cf. 4.1.3) et pour réaliser des statistique sur l'ensemble des demande.

4.1.2 L'envoi de réponse

Suite à l'envoi d'un message par un utilisateur, si il le peut, le serveur envoi un message sinon la personne concerner répond à la demande.
Pour le moment le serveur répondra avec un message générique, pour cause d'un manque de temps et de moyen.

4.1.3 Récupérer l'historique

L'or de sa connections l'utilisateur peu récupérer son historique de conversation afin de reprend le fil de sa conversation.
Si l'utilisateur na pas d'historique une liste vide luis est retourner.

4.2 Java

Comme dit précédemment le programme sera développer pour la plateforme java et développer en java et plus précisément dans la version 8. Cela permet de faciliter le déploiement du programme vue que java est multi-plateforme.

4.3 Librairie

4.3.1 Slf4j

Slf4j[1] est une api de logging. Elle permet de loger les informations souhaiter à l'endroit souhaiter. Par exemple dans un fichier. Cela va permettre de pouvoir avoir les log même après l'arrêt du programme car les message ne sont plus directement dans le terminale.

4.3.2 Spring Boot

Nous allons aussi utiliser *Spring Boot*[2] en version 1.5.8 qui est un framework permettant de développer une application web avec une gestion simplifié des différents services mis en place et de leur présentation sous forme de service REST.

4.3.3 Spring Boot Test

Spring Boot Test[3] est utiliser pour réaliser les tests de l'api REST. Nous l'utiliserons dans la même version que Spring Boot (cf. 4.3.2).

4.3.4 Junit

Pour les test plus générale, nous allons utiliser *junit*[4] en version 4.12, qui est un framework de test unitaire.

4.3.5 Gson

Gson[5] est une librairie qui permet de convertir des objet Java en JSON, et inversement. Il nous sera utile pour comprendre les information reçu et de pouvoir en renvoyer.

4.4 Outils

Pour ce projet nous utiliserons différant outils qui vont simplifier le développement.

4.4.1 IntelliJ IDEA

IntelliJ IDEA[6] est l'IDE java qui est utiliser pour ce projet. Il va permettre de faciliter le développement grâce à ces différente options.

4.4.2 Maven

Maven[7] est un outil de gestion et d'automatisation de production des projets Java. Il va être utiles pour la séparation du programme en module, la gestion des dépendance et les build.

1. `http://www.slf4j.org`
2. `http://projects.spring.io/spring-boot`
3. `https://spring.io/guides/gs/spring-boot/#_add_unit_tests`
4. `http://junit.org/junit4/`
5. `https://github.com/google/gson`
6. `https://www.jetbrains.com/idea/`
7. `http://maven.apache.org/`

4.4.3 MongoDB

MongoDB[1] est un système de gestion de base de données orientée documents, il est simple d'utilisation. Il va être utiliser pour sauvegarder les messages des utilisateur afin de leur retourner leur historique (cf. 4.1.3).

4.4.4 GitHub

GitHub[2] est un service de gestion de développement. Il va être utiliser pour gérée les versions du projet. Cela va aussi permettre le partage des sources sous licence MIT affin de récupérer le retour des utilisateurs et de leur amélioration.

4.4.5 Travis-ci

Travis-ci[3] est un outil d'intégration continue, qui permettre d'effectuer les build et les tests sous différent environnement UNIX. Il permet aussi de déployer les build.
Il est aussi possible d'utiliser *AppVeyor* pour un environnement Windows.

4.4.6 Codecov

Codecov[4] est un outil permettant de regrouper, fusionner, archiver et comparer des rapports de couverture de code. Cela permet, notamment, de trouver les code mort.

5 Conclusion

Ce projet a donc pour but de développer le back end et l'interface d'un marketing conversationnel. Pour des raison de temps seul l'échange de messages et l'historique sera développer. Le projet sera réaliser en java. Le développement ce fera autour des tests afin de valider le fonctionnement des différée méthodes. Les différente tâche doive être séparer pour avoir un couplage lâche. Différent outil serons utiliser pour faciliter le développement.

FIGURE 4 – Logo

1. https://www.mongodb.com/what-is-mongodb
2. https://github.com/
3. https://travis-ci.org/
4. https://codecov.io/gh