**Rapport Projet AAR**

Architecture des Applications Réticulaires

KHAN Shazad  
BAH Sarifou  
AOUAD Anas

Création d’une application Web d’évaluation de produits culturels

SOMMAIRE

I. Le projet 2

II. Organisation du projet 2

A. Organisation du travail 2

B. Présentation de l’équipe 2

III. Analyse des besoins et spécifications du projet 3

A. Besoins 3

B. Fonctionnalités 3

C. Spécifications techniques : technologies retenues 4

D. Architecture de l’application 6

IV. Environnement de developpement 8

A. Environnement matériel 8

B. Environnement logiciel 8

C. Modèle de données de l’application 9

D. Initialisation du projet 10

E. Nos extensions 11

V. Modèle économique de culture-me 11

VI. Analyse du projet et problemes rencontrés 11

# Le projet

La projet consiste à réaliser une application web d’évaluations de produits culturels (albums de musique, films ou DVDs, jeux vidéo, livres) qui propose à ses utilisateurs de donner leur avis (par des notes et des commentaires) sur de tels produits, récents ou non.

D’après l’énoncé on peut déjà distinguer des « entités » : Utilisateur, Produits culturel, Catégorie de produit, Commentaire. Nous allons voir les différents outils et technologies utilisés pour la réalisation de l’application. Le détail de leur utilisation sera décrit dans le cœur du projet.

# Organisation du projet

## Organisation du travail

Le travail a été réalisé par trois étudiants. Tout d’abord, nous avons listé tous les termes spécifiques à un site de vente aux enchères, sous forme de glossaire. Cela nous a permis de mieux comprendre le principe d’un site d’évaluations de produits culturel. Ensuite, nous avons étudié de manière précise les différents outils et technologies proposés par JEE. Nous avons ainsi pu voir leurs caractéristiques et choisir en connaissance de cause. A partir de là, nous avons effectué une phase d’analyse nous permettant de cibler les besoins et fonctionnalités de l’application en fonction du cahier des charges. Puis nous nous sommes répartis les tâches en fonction des couches de l’application. Après une première prise en main des outils choisis, nous avons développé l’application. Enfin, nous avons effectué plusieurs tests sur toutes les fonctionnalités. Pour la gestion de versions de code source, nous avons publié nos sources sur un repository « Google Project Hosting » et utilisé SVN.

## Présentation de l’équipe

## Anas AOUAD

//FAIRE UNE PRESENTATION PERSONELLE DE ANAS

* Mamadou BAH Sarifou

//FAIRE UNE PRESENTATION PERSONELLE DE SARIF

* Shazad KHAN

Suite à un DUT Informatique au CNAM ainsi qu’une Licence ACSID et une expérience de trois ans en programmation de site web et d’administration de parc informatique chez CAHOUET, spécialisé dans la vente de matériels de régulation de pression pour les fluides (gaz et liquides), il a continué l’INSTA pour approfondir ses connaissances et a ensuite rejoint le Master 2 STL à l’UPMC tout en continuant sa mission chez CAHOUET.

# Analyse des besoins et spécifications du projet

## Besoins

* L’administrateur : L’administrateur aura pour rôle de gérer le peuplement de la base de donnée qui devrait être géré par des tâches « cron » mais qui a la possibilité via un menu de peuplé la base selon
* Le membre : Accède à la page d’accueil avec les derniers produits culturels et un accès à une fiche détaillé par produit, peut évaluer un produit et ajouter un commentaire. Il accède aux commentaires sur les produits culturels et à tous ses commentaires.
* Le visiteur : Accède à la page d’accueil et à la page détaillé des produits culturels mais ne peut que visiter le site, ne peut pas évaluer un produit. Par contre, il pourra laisser un commentaire « libre » qui sera géré par une extension de l’application que nous verrons plus loin.

## Fonctionnalités

Dès lors nous pouvons établir les différentes fonctionnalités qui nous permettent de cibler plus précisément la solution la plus adaptée et de répondre au mieux aux besoins et contraintes imposés pour la réalisation de ce projet :

* Importer les produits culturels depuis une API externe
* Gérer les produits culturels
* Ajouter une évaluation à un produit
* Ajouter un commentaire à un produit
* Afficher la liste des commentaires sur un produit
* Afficher la liste de ses commentaires
* Connexion utilisateur

## Spécifications techniques : technologies retenues

1. JavaEE :

Java Enterprise Edition, ou Java EE (anciennement J2EE), est une spécification plus particulièrement destinée aux applications d’entreprise. Ces applications sont considérées dans une approche [multi-niveaux](http://fr.wikipedia.org/wiki/Client-serveur#Architecture_.C3.A0_3_niveaux)[1](http://fr.wikipedia.org/wiki/Java_EE#cite_note-0). Dans ce but, toute implémentation de cette spécification contient un ensemble d’extensions au [Framework](http://fr.wikipedia.org/wiki/Framework) Java standard ([JSE](http://fr.wikipedia.org/wiki/J2SE), Java Standard Edition) afin de faciliter la création d’applications réparties.

1. Google AppEngine :

Google App Engine (ou GAE) est un serveur d’application pensé pour le Cloud. Visant historiquement les applications Python, il s’adresse aujourd’hui aussi aux applications Java. Le SDK d’App Engine permet de développer une application, la tester puis la déployer dans le Cloud. Il propose également une console d’administration minimaliste. Les applications hébergées sur Google App Engine sont hébergées dans les mêmes Data Centers et les mêmes serveurs physiques que les applications Google comme Google Search, Google Mail, Google Docs … Les applications déployées sur AppEngine profitent donc de l’infrastructure de Google réputée fiable, robuste et hautement scalable.

La gestion de la persistance est réalisé par l’ORM « DataNucleus », dont l’implémentation JDO que nous avons utilisé. La simplicité d’utilisation de JDO qui se base sur la manipulation de POJO (Plain Old Java Objects) a fait son succès.

La base de données de GAE appelée « DataStore » est la solution NoSQL qui repose sur le SGBD BigTable développé par Google. Le datastore utilise notamment « Map-Reduce » pour optimiser la lecture dans la base.

Une simple annotation « PersistenceCapable » rend une entité persistante dans le datastore.

1. HTML5, CSS3, JavaScript :

Nous avons utilisé la dernière version de HTML, HTML5, pour l’affichage de nos pages web. Il intègre une sémantique enrichie :

- de nouvelles balises décrivant la façon dont est segmentée la page (<header>, <section>, <article>, <footer>, etc, etc.). C’est donc la « fin » de la balise <div></div> à tout va

* - des «microdonnées» permettant de préciser la signification des données qui leurs sont associées (comme par exemple «name» ou «author» pour la description d’un livre sur un magasin en ligne).

Couplé à CSS3, nouvelle norme des feuilles de styles qui gère la présentation du contenu des pages, on met un pied dans le responsive design. Un site responsive est un site dont le contenu s’adapte à la résolution d’écran du client : écran d’ordinateur classique mais aussi tablette ou Smartphone. L’application culture-me a été développé en ce sens : les JSP génèrent des pages HTML.

Grâce à JavaScript, et notamment Ajax, nous avons la possibilité de

1. SVN :

Subversion est un système de gestion de versions des fichiers d’un projet. Il a plusieurs fonctions :

🡪 Garder un historique des différentes versions des fichiers d'un projet

🡪 Permettre le retour à une version antérieure quelconque

🡪 Garder un historique des modifications avec leur nature, leur date, leur auteur...

🡪 Permettre un accès souple à ces fichiers, en local ou via un réseau

🡪 Permettre à des utilisateurs distincts et souvent distants de travailler ensemble sur les mêmes fichiers

1. Maven :

Nous avons utilisé l’outil Maven pour la gestion et l’automatisation des importations de dépendances, à l’aide d’un fichier pom.xml qui décrit le projet, la liste des plugins utilisées mais aussi la version de Java utilisé pour la compilation.

1. API Restfull

L’application WEB Culture-me utilise l’API « RottenTomatoes » qui recense les informations sur les films au cinéma et en DVD. Après s’être enregistré sur le site, une « api-key » est associé au compte et nous permet de lancer des requêtes : récupération d’une liste de films (box-office, au cinéma, à venir en salle..) ou de dvd (meilleurs locations, nouvelles sorties, vente en cours…).

Restfull est un style d’architecture qui repose sur le protocole HTTP : on accède à une ressource par une uri unique pour procéder à diverses opérations, ici GET.

Un exemple d’uri : *http://api.rottentomatoes.com/api/public/v1.0/lists/movies/box\_office.json*

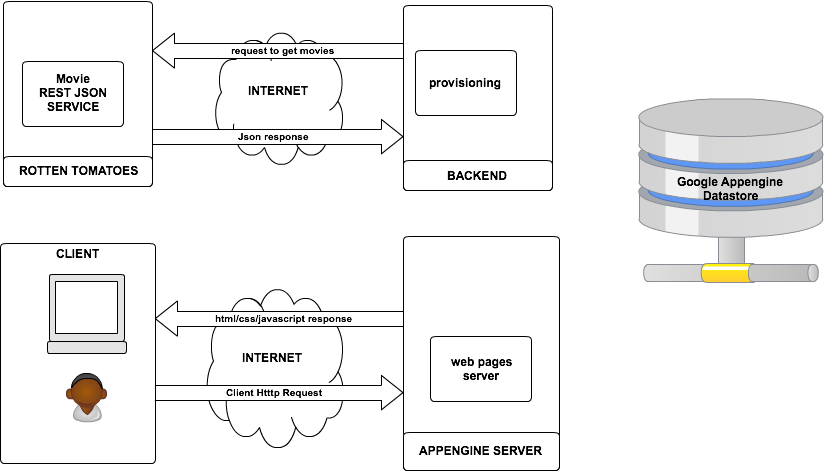
Comme on le voit dans l’uri, la réponse retournée est au format JSON. Ce fichier est formaté de manière claire et concise, il faut alors utiliser un parser de JSON pour traduire ses données pour qu’ils soient exploitables par l’application. Pour cela nous avons utilisé Gson et des annotations « @SerializedName » sur nos POJO (classes du package modèle).

## Architecture de l’application

L’architecture JEE utilise le pattern reconnu MVC qui sépare l’application en 3 couches distinctes :

* *Le Modèle* qui contient la logique de l’application, il peut accéder à une base de données pour la persistance.
* *La Vue* qui est la partie chargée de la présentation (IHM)
* *Le Contrôleur* qui gère la synchronisation entre la vue et le modèle, réagit à l’action de l’utilisateur met à jour le modèle et informe la vue des modifications.

Ci dessous le schéma de l’application développée :



Cron tasks

CRUD operations

Write data

# Développement de l’application

## Environnement matériel

Voici les différentes machines utilisées pour le projet par ses acteurs :

Shazad KHAN :

* Système d’exploitation : Mac OSX Mavericks 10.9
* Processeur : Intel Core 2 Duo 1,86 GHz
* Mémoire vive : 4 Go de RAM

Sarifou BAH :

* Système d’exploitation : Mac OSX Mavericks 10.9
* Processeur : Intel Core 2 Duo 2,4 GHz
* Mémoire vive : 8 Go de RAM

Anas AOUAD :

* Système d’exploitation : VM Linux Ubuntu 12.4
* Processeur : Intel Core i7 2,2 GHz
* Mémoire vive : 4 Go

## Environnement logiciel

1. Eclipse :

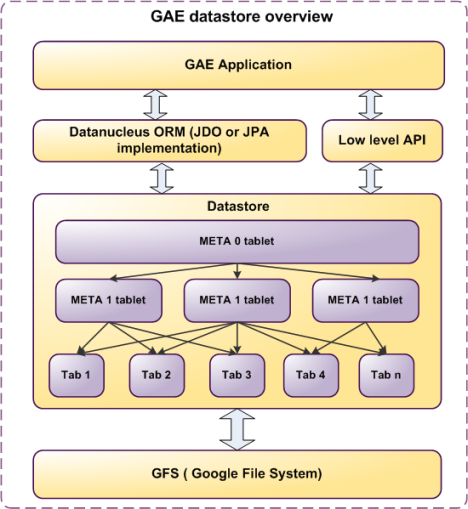
Nous avons tout les trois utilisé Eclipse comme environnement de développement car il libre, universel et adapté à notre application. Grâce au plugin AppEngine pour Eclipse, nous avons pu commencer le développement de l’application avec l’architecture de AppEngine. Eclipse permet aussi d’utiliser SVN simplement avec l’aide du plugin SVNKit.

1. Google AppEngine Server 1.9.5 :

Plateforme d’hébergement d’applications web basé sur les serveurs de Google, supportant le langage Java. Le serveur intègre directement la base de données. Il peut être déployé localement et sur un site distant dans le domaine « appspot.com ». L’intégration de ce serveur dans eclipse se fait grâce à l’ajout du plugin adéquat.

## Modèle de données de l’application

Nous l’avons vu dans l’architecture de l’application, Google App Engine utilise le DataStore pour persister les données. Voici un schéma général du datastore :



Le datastore a un concept d’entité qui nous permet de structurer les informations. Les entités sont des regroupements de plusieurs paires clé – valeur. Dans notre application, nous avons par exemple besoin de récupérer les informations des acteurs d’un film. Pour cela nous ajoutons la propriété *« defaultFetchGroup= true »*.

Voici une partie de nos entités présentes dans le datastore :

movieKey  
title  
genre  
year  
rating  
runtime  
posters  
comments

**Movie**

commentKey  
username  
commentDate  
content  
mark

**Comment**

**User**

userKey  
username  
password  
lastName  
firstName  
mail

## Initialisation du projet

Pour démarrer le projet, nous avons commencé par crer un projet de type « Maven » avec l’archétype *« appengine-skeleton-archetype »* à partir de Eclipse. L’invocation de *« mvn clean install »* génère un projet de type App Engine vide, avec les fichiers et répertoires requis.

Nous pouvons donc commencer à coder avec une base valide et créer notre projet. Cet étape a été notre premier commit SVN.

## Nos extensions

Disqus :

Service Web qui nous permet de laisser la possibilité à un utilisateur de laisser un commentaire sur notre site sans y être préalablement inscrit. Il ne notera pas l’application mais pourra laisser un avis qui sera ensuite récupéré par l’administrateur. On peut filtrer les commentaires, les faire valider par un modérateur (ici l’admin de culture-me).

Mail :

L’application enverra un mail lorsqu’un utilisateur s’inscrira et s’il a oublié son mot de passe.

# Modèle économique de culture-me

Le modèle économique choisi pour l’application est assez simple. Il se base sur deux critères : la vente des avis d’utilisateurs récoltés par l’application et l’affichage de pubs vers les sites de cinémas. Avec la géolocalisation d’un visiteur ou d’un membre de culture-me, grâce à HTML5, l’utilisateur sera redirigé vers le site internet

# Analyse du projet et problemes rencontrés

Ce projet nous a permis de nous familiariser avec les outils de Google App Engine, qui sont riches en fonctionnalités et intègre nativement une multitude d’outils et plugins. La base de données utilisée est basé sur du NoSQL et deux des membres du groupes n’en avait auparavant jamais manipulé. Ce fut une expérience riche qui nous a aussi fait travaillé dans une équipe de trois personnes, où il a fallu gérer le versionning du code source. Nous avons aussi mis un pied dans le développement de Mais il faut aussi avouer que nous avons rencontré quelques problèmes pendant son déroulement.

* Gestionnaire de version :

Au départ de la création du projet, nous avons utilisé GIT. Il était directement utilisable par Eclipse et c’était le repository par défaut dans le Project Hosting de Google Code. Il s’est avéré après quelques jours d’utilisation que les commit et les update ne se faisaient pas correctement et que l’on a perdu la totalité du projet à un instant T. Nous avons décidé de passé à SVN et son utilisation a été beaucoup plus intuitive et le tout était surtout fonctionnel.

* API RestFull :

Au niveau du choix de l’API à utiliser, nous avons voulu utiliser l’API Metacritic qui recensait tous les produits culturels que nous voulions gérer : jeux vidéos, films, musiques. Le site de cet API a crashé et nous nous sommes dirigés vers l’api « RottenTomatoes » qui était complète mais centrée sur les films.

* Requête HTTP sur Google AppEngine

Les requêtes HTTP que nous avons codés pour récupéré le json depuis l’API Restfull utilisait la librairie Apache et ses classes HttpRequest. Tout fonctionnait très bien en local, mais nous n’avons pas eu la bonne idée de tester le site en production : en effet, l’utilisation de HttpRequest de Apache a été bani sur AppEngine. Toute tentative de connexion résulte d’une fermeture de la connexion. A la place, nous devions utiliser les méthodes FetchUrl de AppEngine. Ce qui nous a obligé à modifier la classe de récupération du JSON au tout dernier moment.