[IM2AG] PROJET E-COMMERCE

CONCEPTION SYSTEM

Documentation

Déménagements & échanges entre particuliers.

Groupe 3

Rédaction	**Derradji Amine**
	Décembre 2017

Table des matières

CHAPITRE 1 : PRESENTATION DU PROJET

Ce dossier de conception système a pour but de donner une vision globale sur ce que l'on a accompli ces deux derniers mois allant de L'architecture jusqu'au déploiement en passant par l'implémentation et en abordant les contraintes rencontrées au cours de l'élaboration.

1. Présentation & contexte du projet

On est partis d'un problème où nos utilisateurs postent des annonces sur les réseaux sociaux, les plateformes de commerces comme leboncoin… Mais souvent à bas prix et tout en se focalisant sur le déménagement.

De ce fait, nous nous plaçons dans un contexte de développement d'une plateforme web e-commerce pour répondre à ce problème. Pour notre part, nous nous sommes particulièrement intéressés aux ventes et échanges de particuliers lors d'un déménagement d'un individu. Un utilisateur peut être à la fois vendeur et acheteur, une session utilisateur est pourvu à cet effet et lui permet de superviser ses ventes et achats. Il peut également faire partie du public et juste consulter les offres poster sur le site.

Les valeurs ajoutées étant qu'un individu peut vendre ses produits selon une fourchette de prix. Il peut également mettre un produit en Don, mais également acheter d'autres produits à petit prix.

Dans ce projet, nous utilisons les dernières technologies de pointe, à savoir Angular 4, Bootstrap 4, J2EE (Servlets & EJB), Hibernate (Persistance et Transactions) et Mysql pour les bases de données. De quoi nous permettre d'offrir la meilleure expérience utilisateur en termes de rapidité, fluidité et sécurité.

CHAPITRE 2 : STRUCTURE DU PROJET

Cette partie du document a pour but de donner une vision globale de l'architecture de notre application. Le détail de l'implémentation se trouve dans le code source.

1. L'Architecture générale

Notre application repose sur une application à 3-tiers : Client, Serveur d'application et Serveur de Base de données.

- Le premier tiers fait objet d'interface entre le client web et le tiers métier (Serveur d'application), développer en Angular 4.
- Le tiers métier représente l'ensemble des EJBs (Entity, Session), ainsi que les servlets (exposer les services). Ce dernier implémente les fonctionnalités de l'application web.
- Le dernier métier de l'application sert à assurer la persistance des données. Ce tiers est matérialisé par une base de données MySQL.

Le schéma suivant illustre notre conception :

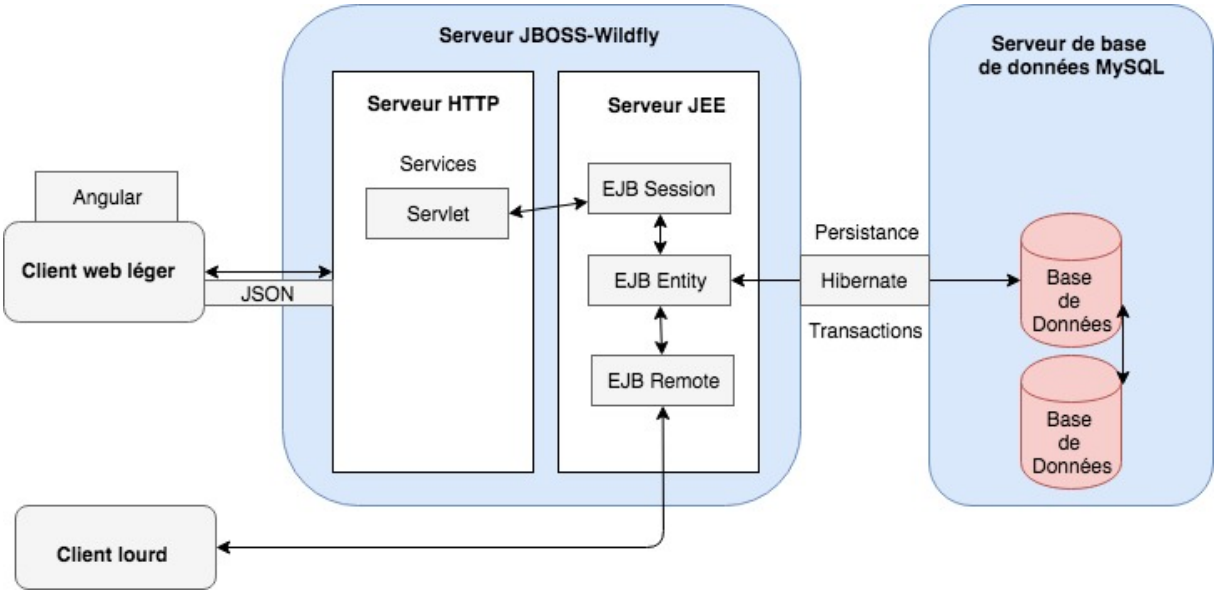

Figure 1 : Schémas de l'architecture physique de l'application

2. Conception de la base de données

Pour ce projet nous avons choisis d'utiliser le serveur de base de données MySql. Ce serveur est connecté avec le serveur d'application JBOSS-Wildfly. Ci-dessus se trouve le modèle UML correspondant à notre architecture de base de données.

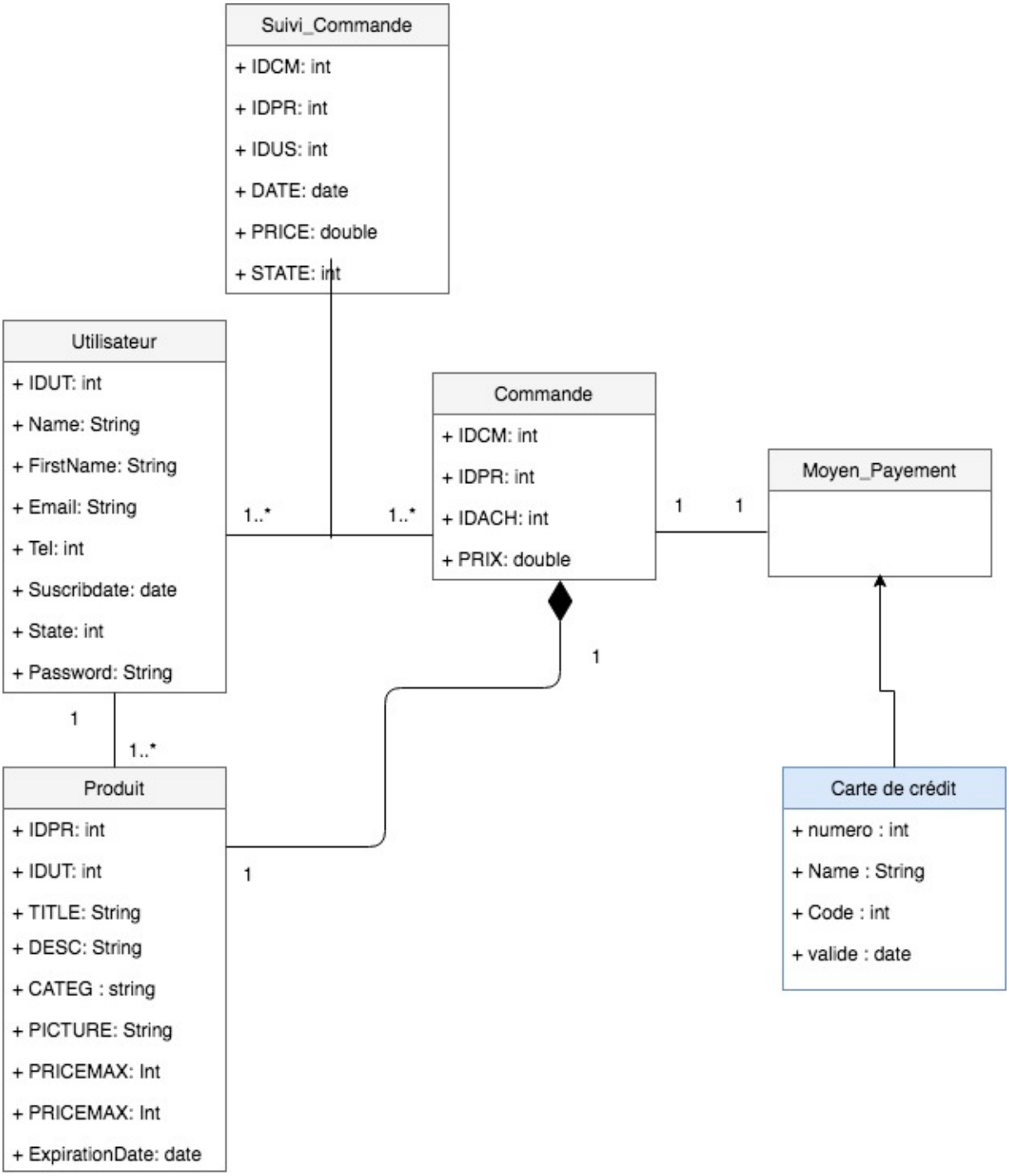

Figure 2 : Schémas de conception de la base de données

Cette étape était la plus importante du projet, car si on s'était trompés sur cette architecture, le projet n'aurait pas abouti. Cette architecture a évolué avec le développement du projet et nous avons apportés les modifications nécessaires.

CHAPITRE 3 : IMPLEMENTATION & STRUCTURE DU PROJET

Nous avons essayé de séparer autant que possible les différentes couches métiers. Nous distinguons 3 couches : les entités, les sessions et services.

1. Nos entités et EJBs :

a. Entités :

- **Utilisateur :** Cet objet permet d'enregistrer les informations relatives à un client (Son identifiant, email, mot de passe et son état).
- **Produit :** Cet objet permet d'enregistrer les informations relatives à un produit mis en vente (Sa référence, sa description complète, son utilisateur, sa date d'expiration, la ville, sa catégorie et la fourchette de prix).
- **Commande :** Cet objet permet d'enregistrer les informations relatives à une commande sur un produit. Une commande regroupe toutes les offres faites sur un même produit par plusieurs utilisateurs.
- **SuiviCommande :** Cet objet permet d'enregistrer les informations relatives à une offre cliente (la commande concernée, client concerné, le produit, l'offre soumise, l'état de l'offre, la date de la commande).
- **Card :** Cet objet permet d'enregistrer les informations relatives à une carte bancaire (Numéro de carte bancaire, cryptogramme, Nom et prénom propriétaire, date de validité).

Les entités se trouvent dans **/src/model/**

b. Sessions : ci-dessus on présente

UtilisateurDao :

Méthodes pourvues :

- **GetUserInfo :** permettant d'avoir les informations sur un utilisateur.
- **SubscribeRequest :** permettant d'inscrire un nouvel utilisateur.
- **UpdateUser :** permettant de mettre à jour les infos utilisateur.
- **BlockUser :** permettant de bloquer temporairement un utilisateur ainsi que ses transactions de ventes ou d'achats.

ProduitDao :

Méthodes pourvues :

- **Create :** permettant de créer un nouveau produit.
- **Update :** permettant de mettre à jour les informations d'un produit.
- **UpdateDon :** permettant de faire passer un produit en un Don.
- **Remove :** permettant de supprimer un produit.
- **GetProductInfo :** permettant d'obtenir les informations liées à un produit.
- **GetProducts :** permettant d'obtenir la liste de tous les produits qui n'ont pas encore expirer.

SuiviCommandeDao :

Méthodes pourvues :

- **Create :** permettant de créer une nouvelle proposition d'offre d'un client.
- **Update :** permettant de mettre à jour l'état d'une offre. (En attente, refuser, accord vendeur, accord acheteur)
- **Remove :** permettant de supprimer une offre cliente.
- **GetOffersByProducts :** permettant d'avoir les offres de chaque produit.
- **GetOffersByUser :** permettant d'avoir les offres par utilisateur.

CommandeDao :

Méthodes pourvues :

- **Create :** permettant de créer une nouvelle commande sur un produit qui regroupera toutes offres cliente.
- **Update :** permettant de mettre à jour les dernières offres retenues par le vendeur.
- **Remove :** permettant de supprimer une commande ainsi que ses offres.

Connection :

Méthodes pourvues :

- **IsAuthgood :** permettant de vérifier si un utilisateur est bien inscrit et que les informations fournis sont identifiable.

Toutes nos EJB sont Stateless (une instance est créée à chaque requête) avec une gestion de persistance ainsi que des transactions.

Les EJBs session se trouvent dans **/src/EJBLOCAL**

2. Services exposés :

Nous exposons via des servlets sous forme d'API-REST toutes nos fonctionnalités implémentées dans nos EJBs, qui à leurs tours seront utilisées par la partie front-end.

Les services se trouvent dans **/src/Service**

3. Sécurité :

a. En ce moment :

- Une clé de sécurité est échangée entre l'application cliente (Angular) et notre back-end.

b. A l'avenir :

Suite à des contraintes de temps et de l'effectif de l'équipe, nous n'avions pas pu atteindre nos objectifs de départ mais nous aurions souhaité ajouter ce qui suit :

- à l'avenir, chaque utilisateur aura sa propre clé et ne pourra envoyer des requêtes qu'avec cette dernière.
- Nous souhaitons également crypter chaque donnée utilisateur sensible (ses coordonnées, son mot de passe, sa carte bancaire …).
- Nous voulons également rajouter la double authentification pour confirmer l'identité du vendeur/acheteur et avoir des transactions fiables.

4. Front-End :

Le premier tiers, qui tient lieu d'interface entre le client web et le tiers métier (Serveur d'application) a été entièrement développé en Angular4. La plateforme web a été développée en composants, chaque partie du site représente un composant.

La structure du code Angular fait en sorte que chaque composant soit responsable, et sépare d'une façon unique la vue et le modèle ce qui permet d'alléger la lisibilité d'une classe TypeScript et d'une vue HTML.

- **Dashboard :** Contient le menu qui redirige vers les différents composants d'une session utilisateur.
- **Header :** Contient l'entête de la plateforme web.
- **Footer :** Contient le pied de la plateforme web.
- **Login-form :** Contient un formulaire de connexion.
- **My-purchases :** Contient la gestion des achats d'un utilisateur
- **New-ad :** Contient le formulaire d'ajout d'un nouveau produit.
- **New-product :** Contient l'espace de gestion des produits qui ont était mis en vente.
- **Offers :** Gestion des offres sur un produit mis en vente.
- Profile : Gestion du profil d'un utilisateur.
- **Search-product :** Contient toutes les propositions de ventes avec possibilité de soumission.
- **Shopping :** Contient le panier utilisateur et sa gestion.
- **Subscribe :** Contient le formulaire d'inscription d'un nouvel utilisateur.

5. Contraintes respectées :

- Un utilisateur ne peut pas soumettre d'offres sur ses propres ventes.
- Il n'y a pas de quantité dans notre panier.
- Un utilisateur peut soumettre une nouvelle offre sur un même produit, L'offre se met à jour si elle n'a pas encore été accepter.

CHAPITRE 4 : DEPLOIEMENT DU PROJET

1. Notre configuration :

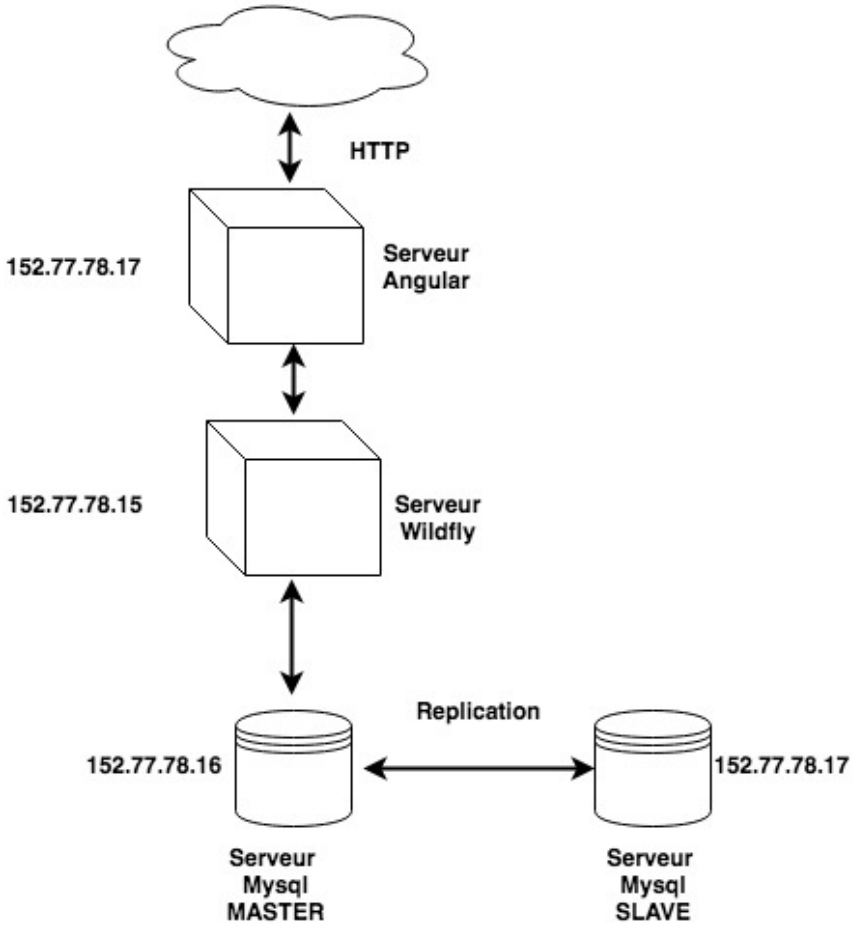

Figure 3 : Schémas de l'architecture de déploiement de l'application

2. Choix de la base de données :

Le choix s'est vite porté sur une base de données relationnelle MySql, nous justifions cela par le fait que ce soit OpenSource, facile à utiliser par tous les membres de l'équipe et pour son mécanisme de réplication des données.

a. Mode de réplication :

Etant donné que nos utilisateurs sont plus amener 90% de leurs temps à consulter des offres, nous avons choisis de répliquer les données utilisateur sur une deuxième base de données selon un système Master-Slave et de l'utiliser comme back-up en cas d'un crash-system. La charge utilisateur devrait être supportable par une seule instance.

3. Choix du serveur d'application :

Parmi les serveurs d'applications gratuit, nous avions le choix entre Glassfish, jboss et la version Wildfly de Jboss. Le choix s'est vite porté sur Wildfly pour sa simplicité d'utilisation, ses exemples prédéfinis ainsi que sa grande communauté.

CHAPITRE 5 : CONTRAINTES & DIFFICULTES

Au cours de ce projet nous avons rencontré des contraintes et difficultés, parmi eux, nous retrouvons :

- Trop de temps passé sur la prise en main des technologies et outils.
- Trop de temps passé sur la configuration des EJBs, la connexion hibernate, et la communication Serveur-Angular.
- Manque de documentations.
- L'équipe avec le moins d'effectifs.
- Le projet est sur une durée de deux mois, à savoir du 6 Octobre au 18 décembre ou un jour de formation et un jour pour travailler sont réservés.