Projet d'Introduction à l'Architecture
Carte d'infidélité

Mercredi 22 février 2017

Groupe 8:
AIELLO Axel
FILIOL DE RAIMOND-MICHEL Guillaume
GONNIN Thibaut

Table des matières

1. Vue fonctionnelle

1.1. Scénarios

En lisant l'appel d'offre, nous avons déterminé un ensemble de scénario pour représenter toutes les fonctionnalités de l'application.

Scénario N°1 : Abonnement Client

- Remplire le formulaire d'adhésion
  - En ligne
  - En magasin
- Récupérer la carte
- Générer un numéro de compte unique
- Se désabonner

Scénario N°2 : Abonnement Commerçant

- Remplire le formulaire d'adhésion en mairie
- Activer son compte "Commerçant"
- Générer un numéro de compte unique
- Se désabonner

Scénario N°3 : Gestion Client

- Activer un avantage
- Charger sa carte
- Gestion et alertes des magasins favoris
- Regarder les horaires des magasins
- Accéder au catalogue
  - Si le client est VUP
  - Si le client est pas VUP

Scénario N°4 : Gestion Commerçant

- Gérer leurs offres dans le catalogue
  - Si le client est VUP
  - Si le client est pas VUP
- Accéder aux statistiques et informations de la boutique et des boutiques partenaires
- Pouvoir inscrire/désinscrire un client

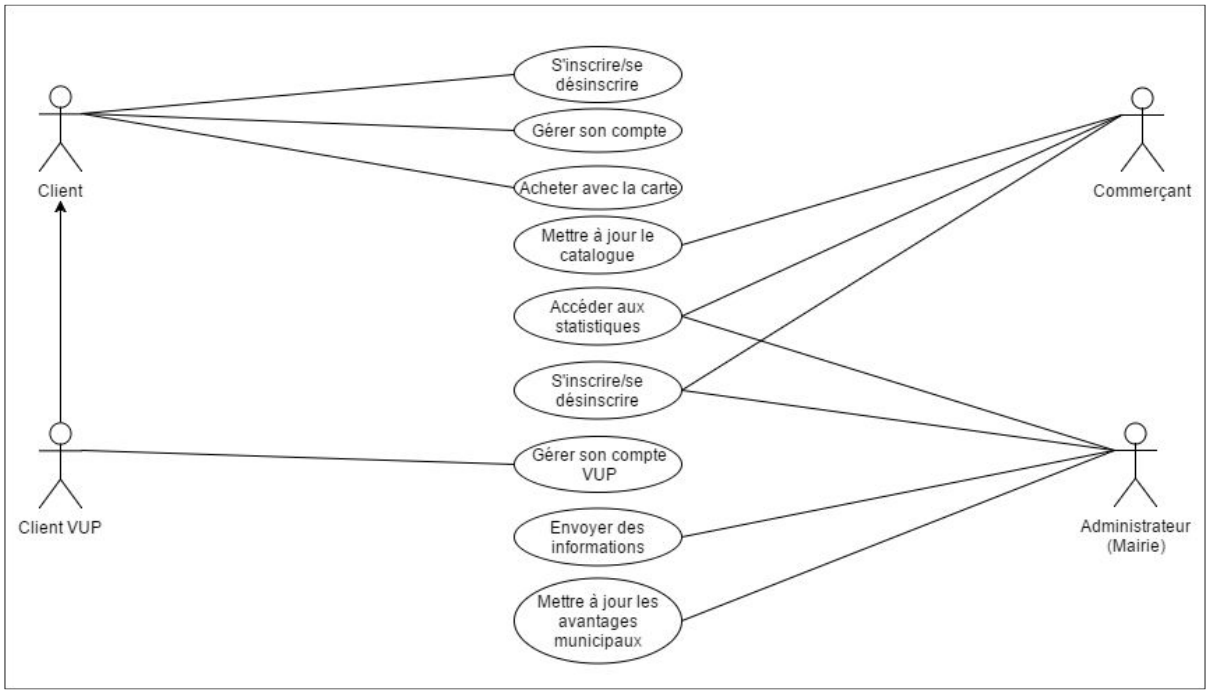
Scénario N°5 : Gestion Municipalité

- Pouvoir inscrire/désinscrire un commerçant
- Envoyer des offres promotionnelles
- Permettre l'envoi de questionnaire de satisfaction aux clients et commerçants
- Gérer les avantages municipaux dans le catalogue
- Relancer des clients
- Récolter et exploiter les informations et les statistiques

1.2. Diagrammes de cas d'utilisation

1.2.1. Diagramme de haut niveau

Le diagramme de cas d'utilisation de haut niveau ci-dessous nous permet de mettre en place les différent acteurs qui interagissent et de regrouper les différentes fonctionnalités que nous avons pu mettre en évidence dans les scénarios.



(

Figure 1 : Diagramme UC de haut niveau

Ici, nous détaillerons les rôles de chacun des acteurs :

- Le **Client / Client VUP** peut s'inscrire / se désinscrire du programme (et recevoir la carte) ainsi que faire des achats avec sa carte si elle a été rechargée au préalable. Il peut aussi acquérir le statut de VUP (et donc le perdre) si les conditions sont remplies. Bien évidemment le Client doit être en mesure de gérer son compte (en accord avec son statut) , c'est à dire de pouvoir consulter le catalogue des cadeaux, activer des avantages.
- Le **commerçant** doit pouvoir gérer le catalogue des cadeaux qu'il propose aux clients, il doit avoir accès aux statistiques (chiffrage divers par rapport à son commerce et l'utilisation du programme auprès des autres partenaires).
- L'**administrateur** peut obtenir des statistiques sur les habitudes des consommateurs, il peut envoyer des offres promotionnelles (suite à la demande du service "Citoyen numérique" ou de l'association des commerçants), il peut exploiter la base d'utilisateur (par exemple pour en relancer certains suite à la perte de leurs statut VUP). Il peut lancer des sondages, alimenter le catalogue. Il est aussi en charge de créer les comptes commerçants.

1.2.2. Cas d'utilisation : S'inscrire / se désinscrire (Client)

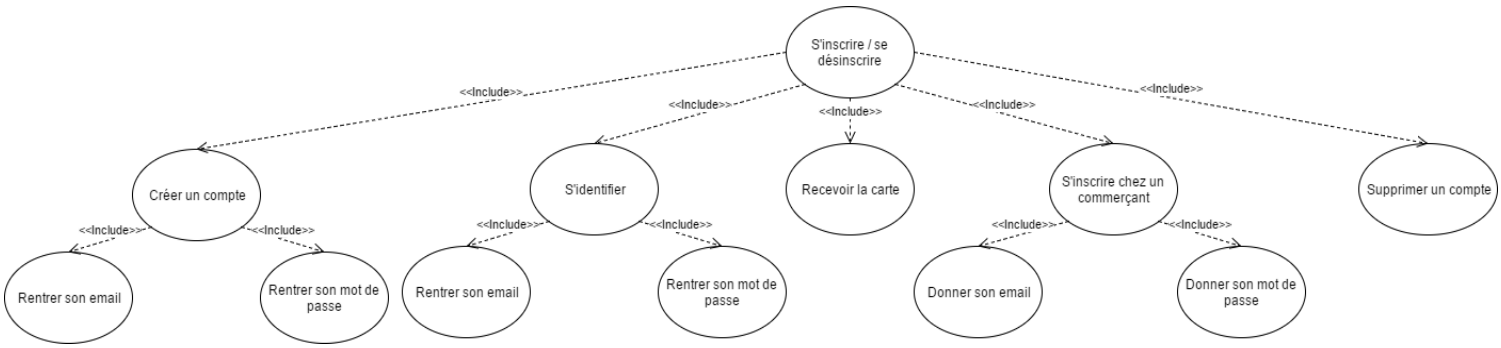


Figure 2 : Diagramme UC détaillé, S'inscrire / se désinscrire (Client)

Un client à deux possibilités pour s'inscrire ou se désinscrire, il peut soit le faire par internet, soit en allant voir un commerçant qui le fera pour lui.

Pour l'inscription :

Le client doit donner son email ainsi que son mot de passe pour créer son compte, ou alors il va chez un commerçant qui fera la manipulation pour lui. Il recevra ensuite la carte chez lui ou chez un commerçant.

Pour la désinscription :

Le client peut se désinscrire par internet après s'être identifié, ou alors il va chez un commerçant qui fera la manipulation pour lui.

1.2.3. Cas d'utilisation : Gérer son compte / compte VUP

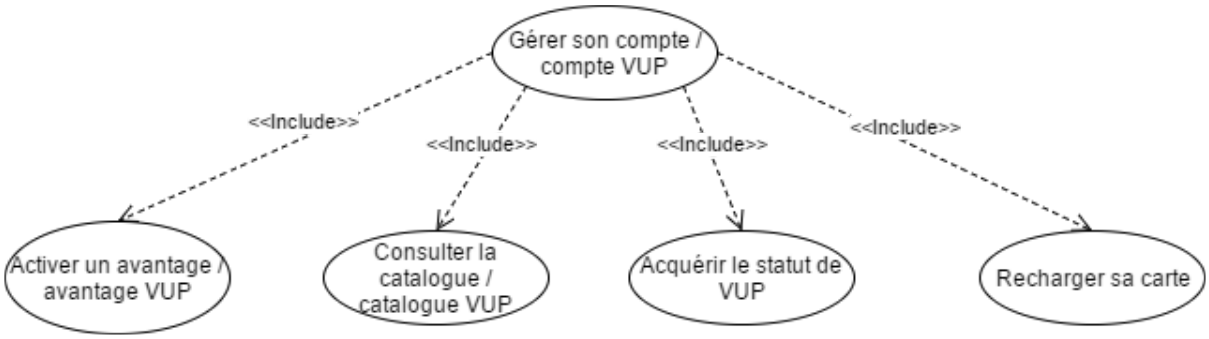


Figure 3 : Diagramme UC détaillé, Gérer son compte / compte VUP

Chaque client peut gérer son compte, ce qui comprends :

- Consulter le catalogue, et voir les articles proposés en fonction du statut (VUP / non VUP).
- Acquérir le statut de VUP de façon automatique(en fonction de la fréquence d'utilisation).
- Activer un avantage (selon son rang) et si les conditions sont remplies)
- Recharger sa carte avec un petit montant pour pouvoir payer chez les commerçants affiliés.

1.2.4. Cas d'utilisation : Acheter avec la carte

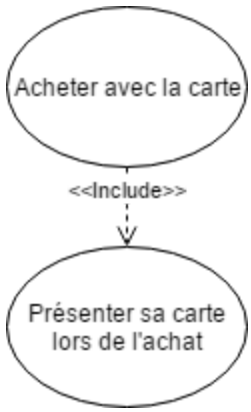


Figure 4 : Diagramme UC détaillé, Acheter avec la carte

Le client peut payer un petit montant avec sa carte dans n'importe quel commerce affilié, du moment qu'il à recharger sa carte au préalable

1.2.5. Cas d'utilisation : Mettre à jour le catalogue

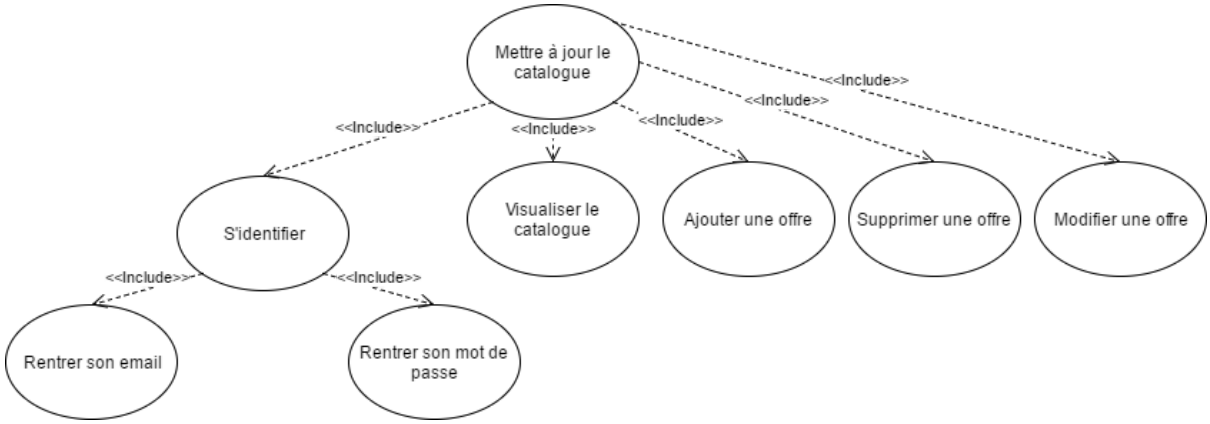


Figure 5 : Diagramme UC détaillé, Mettre à jour le catalogue

Un commerçant doit être capable de mettre à jour le catalogue d'offre qu'il propose.

Pour faire, il doit bien entendu s'identifier par le biais de son adresse mail et de son mot de passe.

Ensuite il peut visualiser le catalogue d'offres qu'il à pour le moment, et peut donc ajouter, supprimer ou modifier une offre, avec pour chaque offre les prérequis pour en bénéficier.

1.2.6. Cas d'utilisation : Accéder aux statistiques

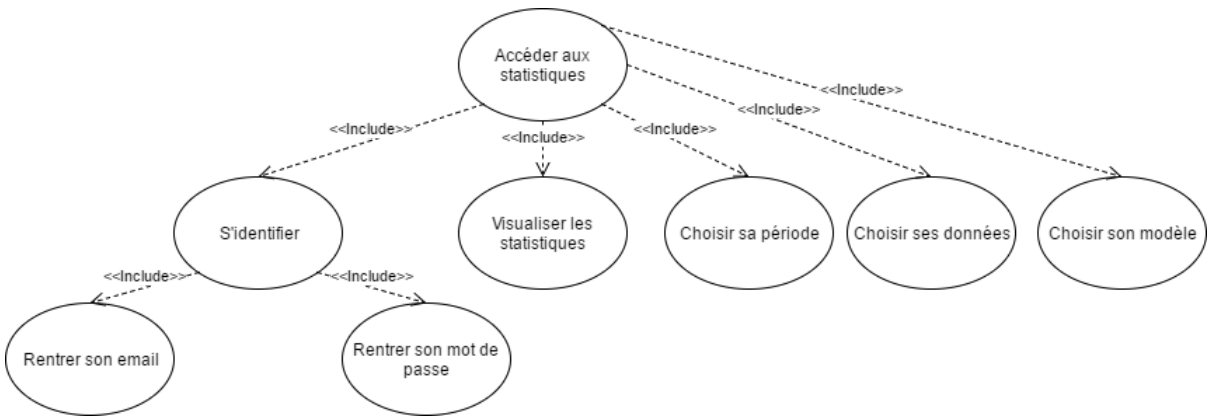


Figure 6 : Diagramme UC détaillé, Accéder aux statistiques

Un commerçant / administrateur doit être capable de voir des statistiques pour répondre rapidement à des questions telles que "Est-ce que je gagne quelque chose à participer à ce programme ?".

Pour faire, il doit bien évidemment s'identifier (email + mot de passe), ensuite il doit choisir plusieurs paramètres :

- La période sur laquelle il veut avoir des statistiques,
- Les données sur lesquelles il veut avoir des statistiques (fréquentation, bénéfices, pertes engendrées par les offres du catalogue, ...)
- Le modèle d'affichage (histogramme, diagramme circulaire, …)

Et ensuite il pourra donc visualiser ses statistiques.

1.2.7. Cas d'utilisation : S'inscrire / se désinscrire (Commerçant)

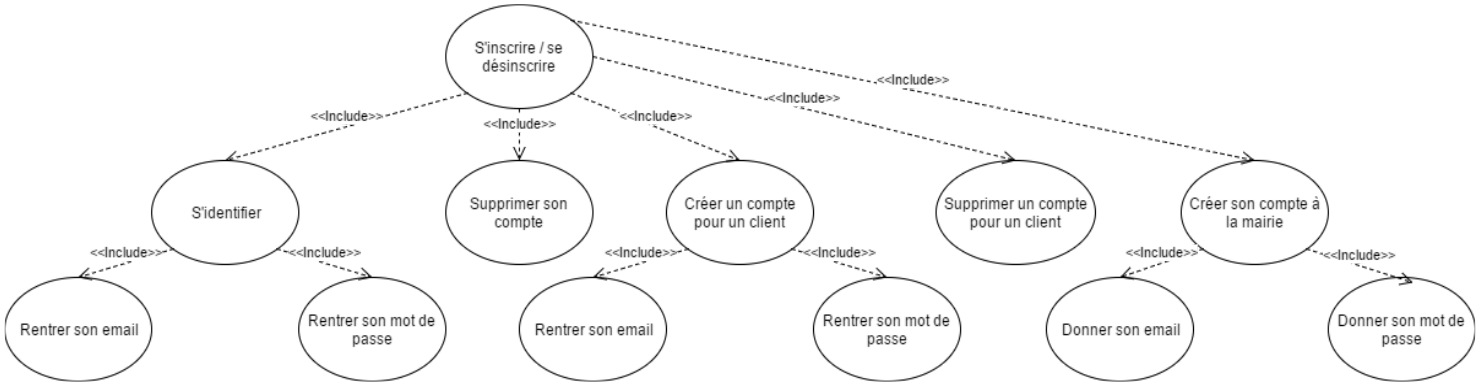


Figure 7 : Diagramme UC détaillé, S'inscrire / se désinscrire (Commerçant)

Pour un commerçant l'inscription / désinscription passe obligatoirement par la mairie, en effet seul un administrateur peut ajouter / enlever un compte commerçant.

Un commerçant doit être capable de créer / supprimer un compte client après s'être identifié.

1.2.8. Cas d'utilisation : Envoyer des informations

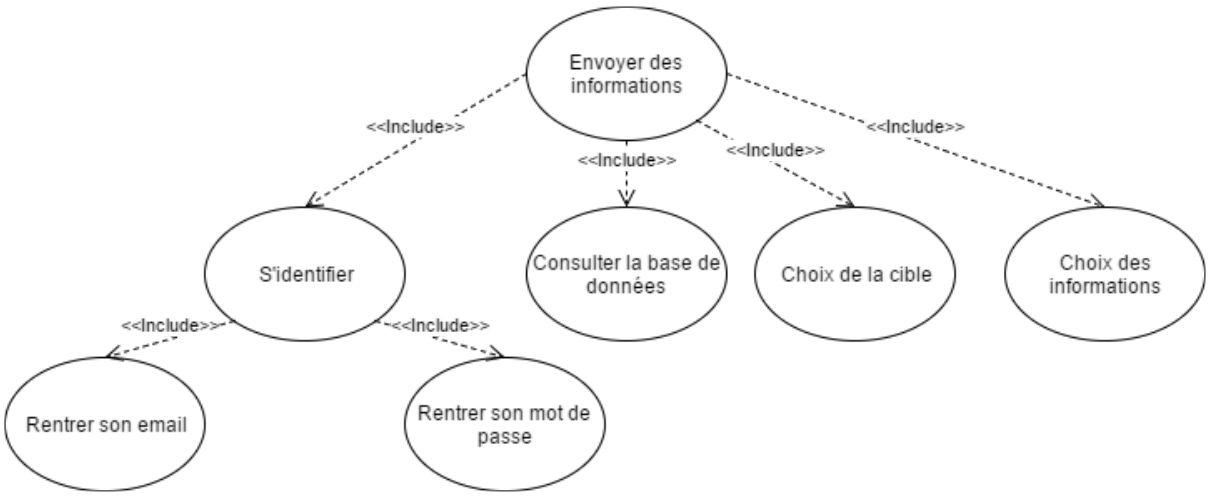


Figure 8 : Diagramme UC détaillé, Envoyer des informations

Un administrateur doit pouvoir consulter la base de données pour avoir les moyens de contacter les utilisateurs.

De plus, il doit être capable après identification de pouvoir envoyer des informations à ces utilisateurs.

Il sélectionne la cible et le type d'informations qu'il veut envoyer, il peut par exemple relancer les clients qui ont récemment perdus leurs statut de VUP (information prise dans la base de données), envoyer des offres promotionnelles à la demande du service "Citoyen Numérique" ou de l'association des commerçant, ou encore envoyer des sondages de satisfaction aux utilisateurs.

1.2.9. Cas d'utilisation : Mettre à jour les avantages municipaux

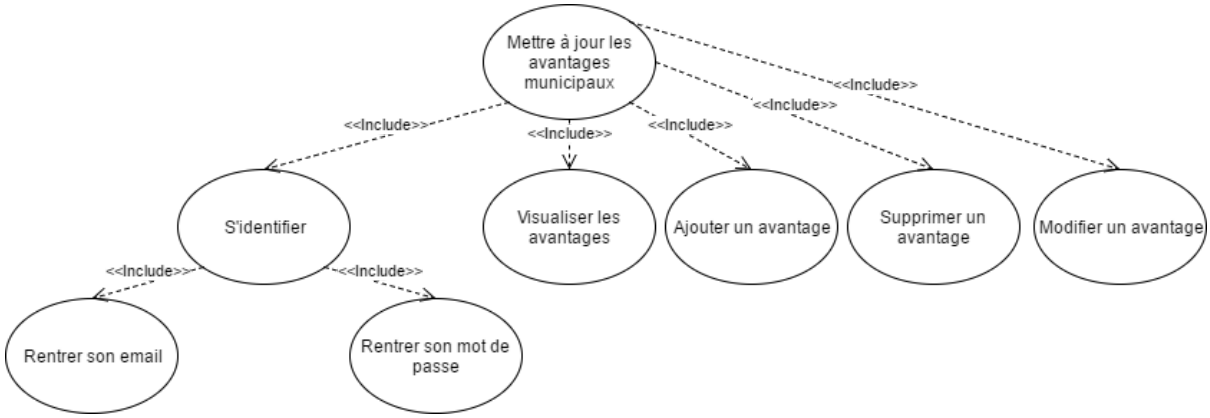


Figure 9 : Diagramme UC détaillé, Mettre à jour les avantages municipaux

Un administrateur, après authentification, doit pouvoir visualiser le catalogue de la municipalité, ainsi que de créer, modifier et supprimer les offres présentes dedans (en définissant les pré requis pour débloquer ces offres).

1.3. Diagrammes de composants

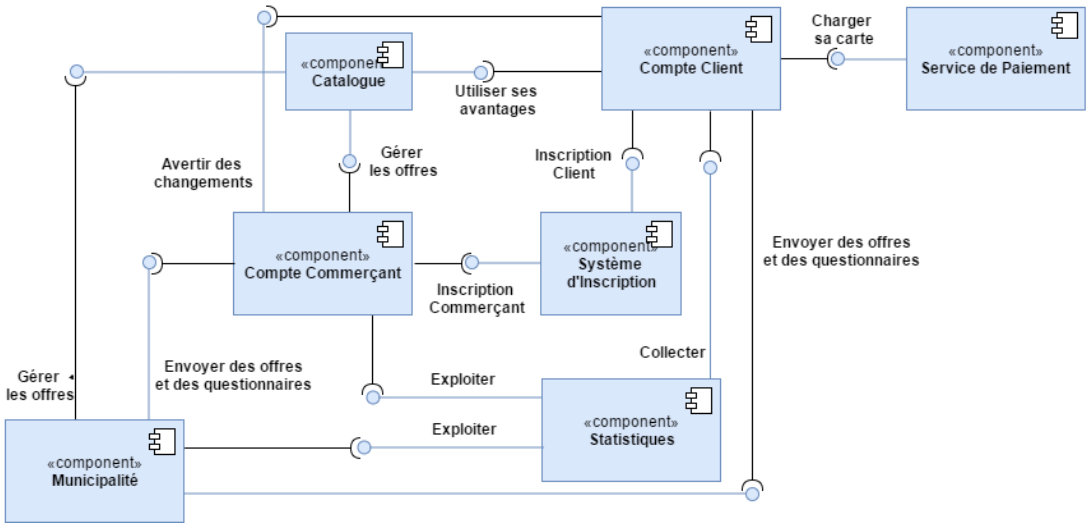


Figure 10 : Diagramme de composants

Charger sa carte :

+ charger(montant : float) : void

montant : montant que l'usager a payé

Inscription Client :

+ inscrire(void) : void

Associer à un compte, une valeur unique pour inscrire le client

+ désinscrire(void) : void

Désinscrire le client

Inscription Commerçant :

+ inscrire(c : CompteCommerçant) : void

Associer à un compte, une valeur unique pour inscrire le commerçant c

+ désinscrire(c : CompteCommerçant) : boolean

Désinscrire le commerçant c

CompteCommerçant : objet qui représente le compte d'un commerçant avec toutes ses informations personnelles

Collecter :

+ collecterDonnées(c : CompteClient) : Map<Element, donnée>

Récupérer les statistiques sur le client c

CompteClient : objet qui représente le compte d'un client avec toutes ses informations personnelles
Element : énum des éléments récupérables pour les statistiques

Exploiter :

+ exploiterDonnées(l : List<Element>) : Map<Element, donnée>

Récupérer les statistiques demandées de tous les clients

+ récupérerStatVente() : : Map<Element, donnée>

Récupérer les statistiques demandées pour le commerçant

Envoyer des offres et des questionnaires :

+ envoyerOffre(u : List<Utilisateur>, promos : List<Promo>) : void

Envoie les offres promotionnelles promos aux utilisateurs u

+ envoyerQuestionnaire(u : List<Utilisateur>, q : List<Questionnaire>) : void

Envoie les questionnaires de satisfaction q aux utilisateurs u

Promo : objet qui représente une offre promotionnelle
Questionnaire : objet qui représente un questionnaire de satisfaction
Utilisateur : objet abstrait représentant un utilisateur, CompteClient et CompteCommerçant en héritent

Gérer les offres :

+ ajouterOffre(l : List<Offre>) : void

Ajouter les offres l dans le catalogue

+ supprimerOffre(l : List<int>) : void

Supprimer les offres identifier par un int du catalogue

+ modifierOffre(l : Map<int, Offre>) : void

Modifie les offres identifier par un int du catalogue

Offre : objet qui représente une offre

Avertir des changements :

+ avertirClient(void) : void.

Averti les clients d'un changement d'horaire pour un commerçant en favoris

Utiliser ses avantages :

+ activerOffre(o : Offre) : void

Active une offre o pour le client

2. Vue développement

2.1. Diagramme de classe

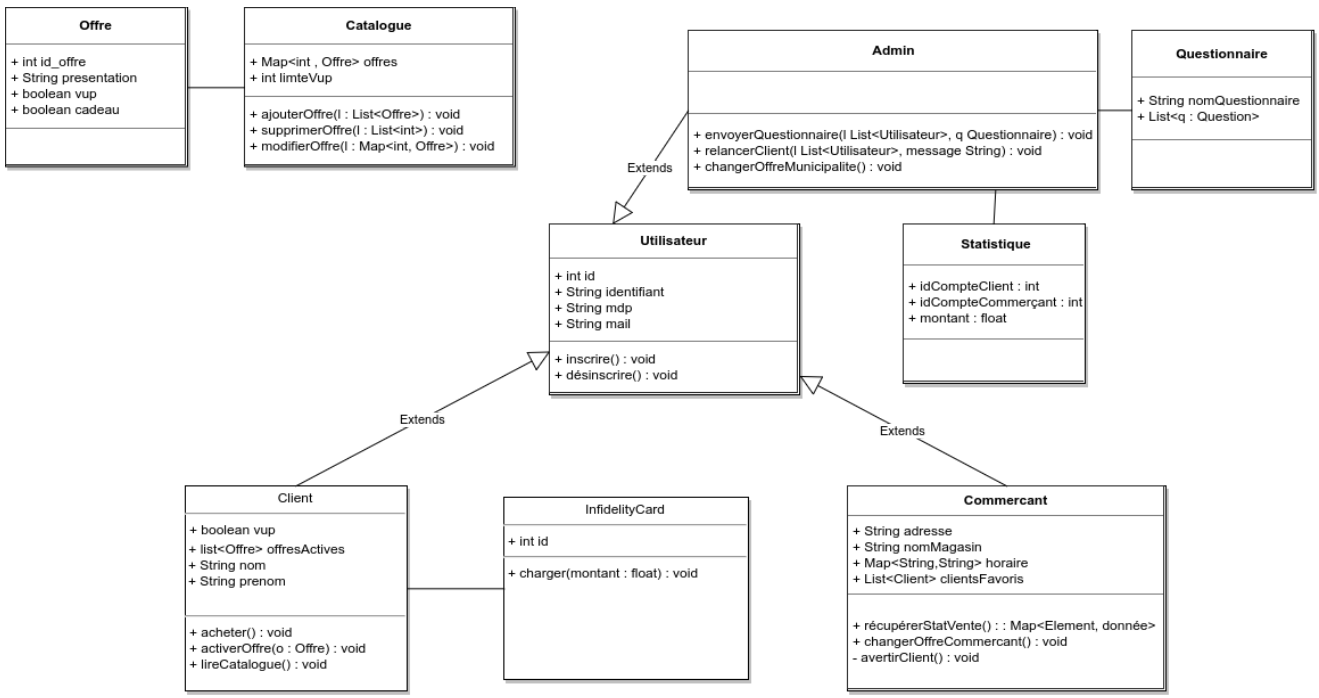


Figure 11 : Diagramme de classe

2.2. Modèle relationnel de stockage

On cherche ensuite comment l'application peut stocker les données utiles par la création d'un modèle relationnel de stockage.

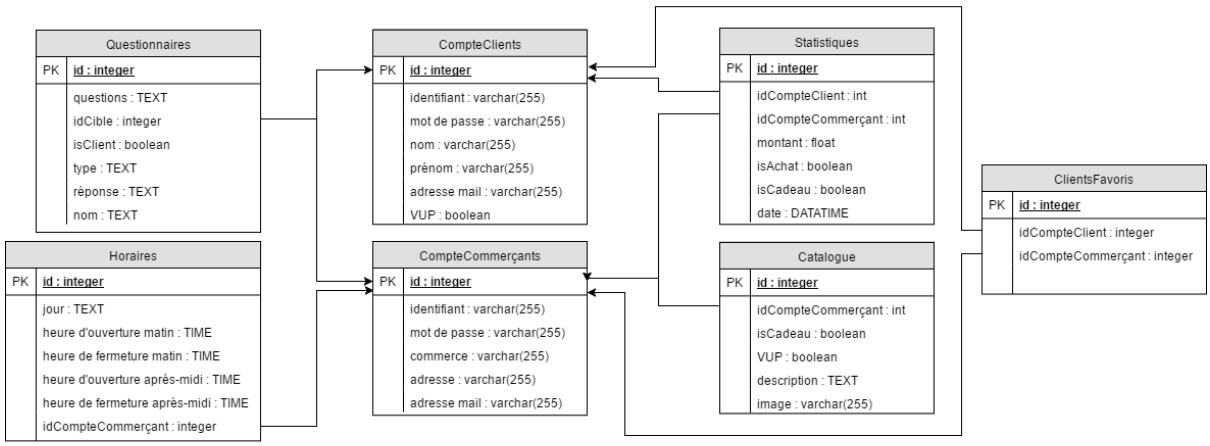


Figure 12 : Modèle relationnel de stockage

Voici quelques exemples de requête SQL adapté au modèle précédent et ayant du sens vis à vis des fonctionnalités à développer dans le système :

Ajouter un client :
INSERT INTO `CompteClient` (`id`, `identifiant`, `mot de passe`, `nom`, `prénom`, `adresse mail`, `VUP`) VALUES (NULL, '', '', '', '', '');

Ajouter un commerçant :
INSERT INTO `CompteCommerçant` (`id`, `identifiant`, `mot de passe`, `commerce`, `adresse`) VALUES (NULL, '', '', '', '');

Changer les horaires de son commerce :
UPDATE `Horaires` SET `jour`='', `heure d'ouverture matin`='', `heure de fermeture matin`='', `heure d'ouverture après-midi`='', `heure de fermeture après-midi`='' WHERE `Horaires`.`id`='';

Connaître le montant de l'argent dépensé chez un commerçant :
SELECT montant FROM `Statistiques` WHERE idCompteCommerçant="" AND isAchat="TRUE";

Connaître le montant de l'argent perdu par un commerçant :
SELECT montant FROM `Statistiques` WHERE idCompteCommerçant="" AND isAchat="FALSE";

Modifier une de ses offres dans le catalogue :
UPDATE `catalogue` SET `isCadeau` = '', `VUP` = '', `description` = '', `image` = '' WHERE `catalogue`.`id` = '';

Les précisions suivantes sont à apporter à notre modèle :

- Les **id** s'auto-incrémentent et commencent à 1.
- La mairie correspond au **CompteCommerçant** n°0
- **isCadeau** est un booléen, si true alors l'offre est gratuit, si false alors c'est une réduction
- **isAchat** est un booléen, si true alors l'article a été acheté, si false l'article est une offre du catalogue.
- **type** correspond aux différents types possibles de questionnaire : QCM, QUESTION_OUVERTE, etc…
- **isClient** est un booléen, si true alors c'est un **CompteClient**, si false alors c'est un **CompteCommerçant**.
- **réponse** a une valeur par défaut **NULL**
- **heure** à une valeur par défaut
- **jour** correspond aux différents types possibles de journée : LUNDI, MARDI, MERCREDI, JEUDI, VENDREDI, SAMEDI, DIMANCHE et FÉRIÉ.

2.3. Explicitation du mapping objet-relationnel

Nous sommes partis du diagramme de classe d'objet métier et du diagramme de composants pour déterminer les données à persister.

Ainsi rapidement on s'aperçoit de l'importance de garder les données des comptes pour les clients et les commerçants comme l'identifiant, le mot de passe, et l'adresse email de connexion, mais aussi les autres données. Pour cela on a créé une table **CompteClient** et **CompteCommerçant** regroupant les données respectivement pour le client et le commerçant. De plus, on remarque que niveau donnée la mairie se rapproche d'un commerce, elle peut donc être incluse dans la table **CompteCommerçant**.

Certaines données plus volumineuses, nécessitent une table à part.

C'est le cas pour la table **Horaires** qui contient les différents horaires (ouverture, fermeture, matin et après-midi) pour chaque commerce et chaque type de journée (férié, lundi, mardi, …). Ces horaires sont reliés au commerçant par son id de la table **CompteCommerçant**.

C'est le cas aussi pour la table **ClientsFavoris** qui relie l'id d'un client dans **CompteClient** et l'id d'un commerçant dans **CompteCommerçant,** on peut ainsi avertir les différents client qui ont mis un commerce en favoris d'un changement d'horaire.

De plus certaines données doivent être transmises d'une personne à une autre.

En effet, pour les statistiques, ils sont récupérés à partir des clients pour les commerçants et la mairie. Pour ce faire ils sont stockés dans la table **Statistiques** qui contient les historiques d'achats soit : le montant, la date, le type d'achat (représentés par deux booléens qui permettent d'avoir un produit acheté, soldé ou gratuit) ainsi que l'id de l'acheteur dans **CompteClient** et l'id du vendeur dans **CompteCommerçant**.

Ensuite, nous avons la table **Catalogue**, regroupant les différentes offres et avantages. Chaque offre est composé de l'id du commerçant dans **CompteCommerçant,** ainsi que, si c'est une offre VUP ou non, par un booléen, si c'est une réduction ou un cadeau par un booléen, et une courte description avec une image.

Enfin, nous avons la table **Questionnaires** décomposé par leurs questions mais identifiable par leur nom, qui est présent dans la table plusieurs fois (une fois par cible, qui peut être un client ou un commerçant) et de différents types (QCM, question ouverte, etc..). Les réponses à ces questions sont initialisées à NULL tant qu'elles ne sont pas remplis permettant ainsi de savoir si le client y a répondu.

3. Vue déploiement

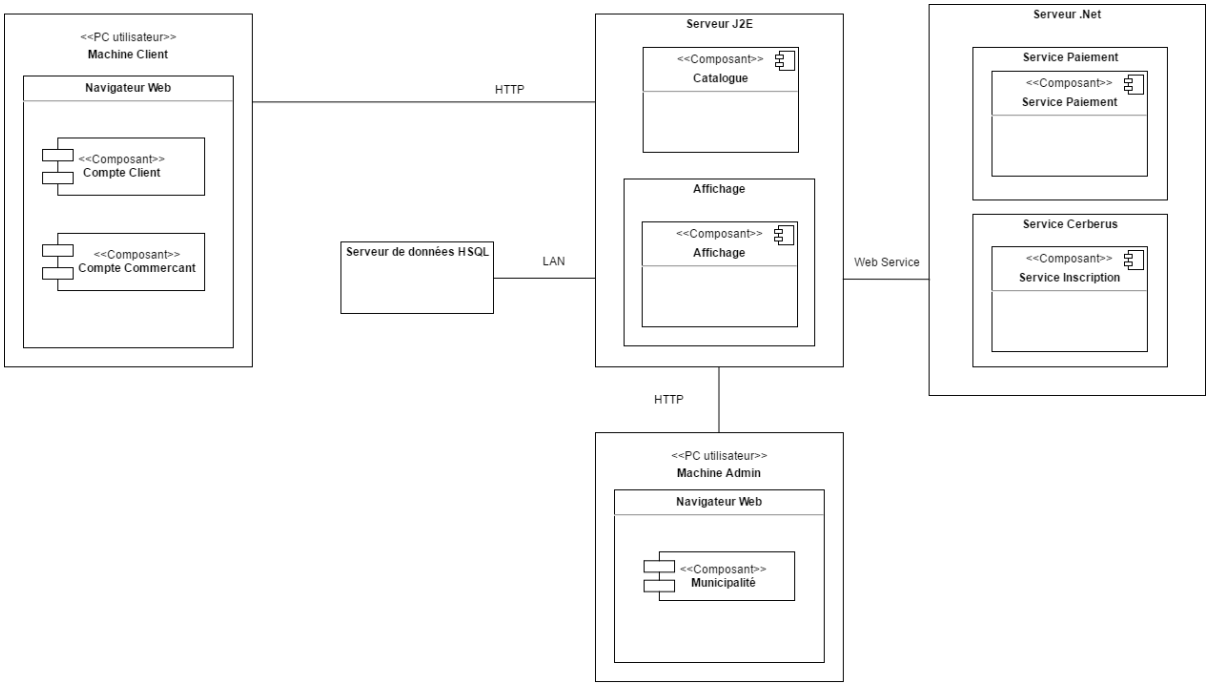


Figure 13 : Diagramme de déploiement

Nos données seront stockées sur le serveur de données HSQL, qui communique avec notre serveur J2E en LAN.

L'appel d'offre nous spécifie déjà que les services partenaires sont simulés avec des service .NET, ainsi que le fait que les services J2E et .NET seront localisés sur deux serveurs physiques et qu'ils communiquerons par service web.

Les utilisateurs utiliserons leurs navigateur web ou un système de ligne de commande.

Dans notre cas nous avons donc 5 entitées physiques.

FEEDBACK

TD 23 / Groupe 08

- Cas d'utilisations:
  - attention à l'orth dans les scénarios
  - Quel lien entre scénario et cas d'utilisation?
  - Identification des acteurs ok.
  - Beaucoup d'include, pas d'extension ou d'héritage entre acteurs. Pourquoi ?
- Composants:
  - Les composants sont orienté par acteurs et pas par fonctionnalités
  - Trop gros en l'état
  - Pourquoi des méthodes privés dans une interface ?
  - beaucoup de void, gros travail par effet de bord.
- Objets métiers:
  - objets métiers avec des méthodes => pas des OMs
  - graphe de classe non connexe. Pourquoi des îlots isolés?
  - pas de justification
- Modèle relationnel:
  - Pas de discussions sur les clés (qu'apportent des id auto-incrémenté ?)
- Mapping Objet- Relationnel
  - faiblement décrit, et pas justifié
- Déploiement:
  - pas de description des serveurs physiques

C'est moyen partout. Il faut prendre du recul et arriver a identifier ce qui a de la valeur dans les choix que vous faites.