



# Cookies on Demand

*Rapport d'architecture*

Borg Guillaume  
Dito Maxime  
Veniat Tom

# Table des matières

---

[Vison du produit](#)  
[Cas d'utilisation](#)  
    [Haut niveau](#)  
    [Make order](#)  
    [Manage Account](#)  
    [Manage Store](#)  
    [Manage Admin](#)  
[Diagramme de composants](#)  
[Diagramme de classe](#)  
    [Package recipe](#)  
    [Package order](#)  
    [Package user](#)  
    [Package manage](#)  
[Modèle relationnel](#)  
[Mapping objet-relationnel](#)  
[Déploiement](#)

## Vision du produit

Nous souhaitons offrir un système centralisé pour l'ensemble des utilisateurs : clients, franchisés et administrateurs de Cookies On Demand utiliseront le même système de type SaaS(Software as a Service). Ainsi, seule la gestion des statuts permettra d'obtenir les fonctionnalités désirées du côté front-end.

Il y a deux types de client les anonymes qui pourront simplement passer des commandes et les utilisateurs enregistré (avec ou sans Loyalty program) qui pourront enregistrer leur préférences ainsi que de formuler d'éventuelle requête. De même ces personnes sont sujets à la récupération d'information statistique.

Enfin, côté management de l'entreprise, le système permettra de récupérer les statistiques mais aussi de créer de nouvelle recette pour l'administrateur de The CookieFactory. Tandis que le franchisé lui pourra récupérer uniquement les statistiques le concernant ainsi que fixer ses horaires aux clients et le today's spécial parmi les recettes préétablie par l'administrateur. De même, il pourra définir le nombre de cookies qu'il pourra fournir par tranche de  $\frac{1}{4}$  d'heure permettant ainsi de ne pas être surcharger par les retraits de commandes clients.

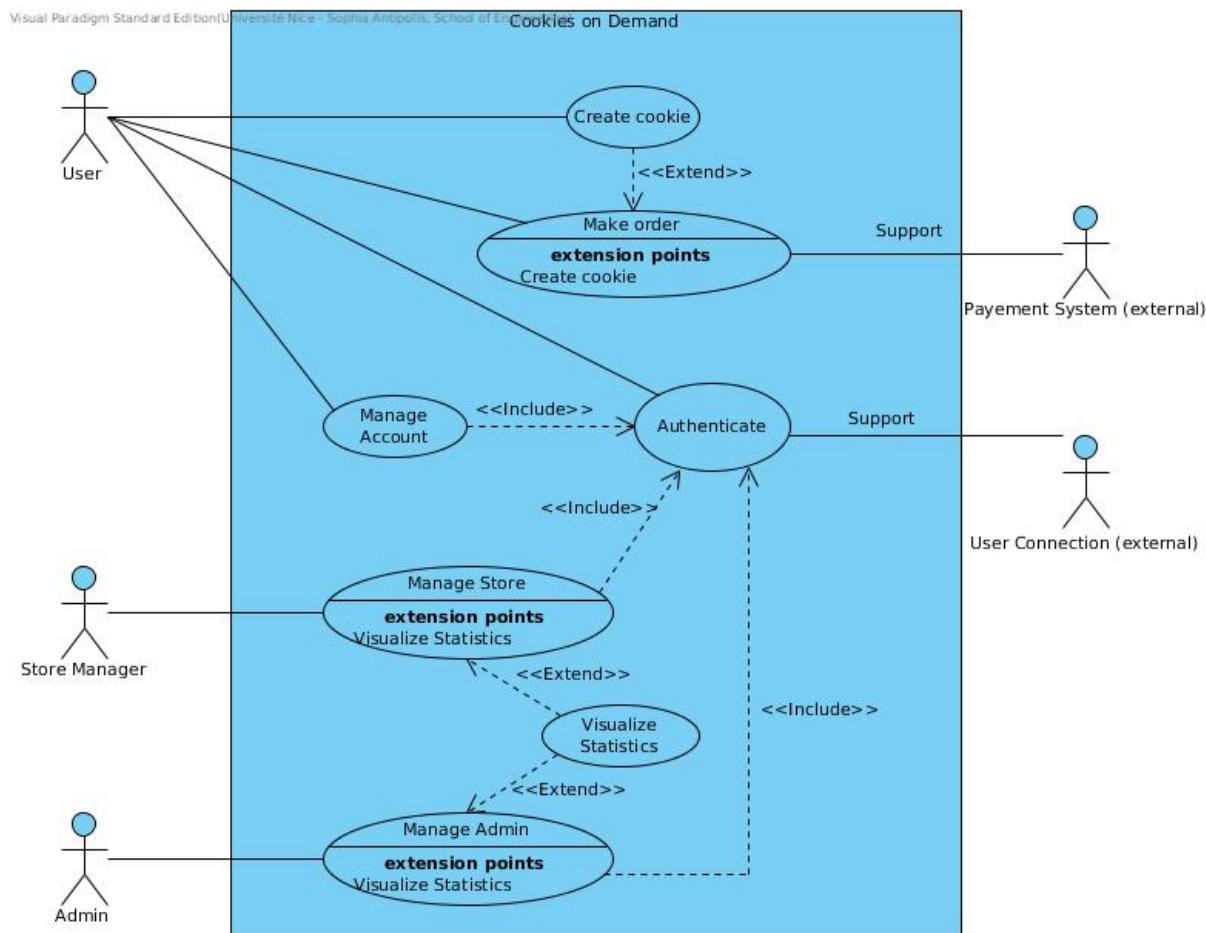
Pour ce qui concerne la gestion des comptes clients nous devrons utiliser l'API .NET qui de l'entreprise de même pour le paiement en lui-même.

L'accès aux services se fera par un simple navigateur internet tandis qu'en interne un programme java couplé à une librairie externe(pour le calcul des statistiques) et à une base de données sera le coeur du système.

# Cas d'utilisation

## 1. Haut niveau

Nous avons commencé par créer le diagramme use case de haut niveau afin de pouvoir identifier les principales fonctionnalités de Cookies on Demand.



Les 3 acteurs principaux sont les clients, les gestionnaires de magasins et les administrateurs.

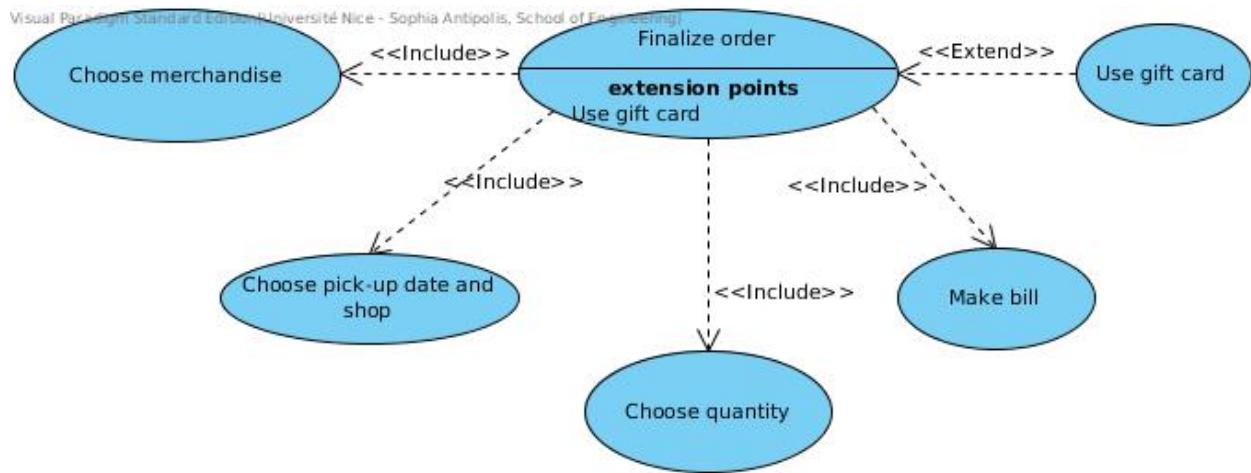
Les clients, possédant un compte ou non peuvent créer des cookies et/ou passer une commande. Les clients possédant un compte peuvent s'identifier afin de pouvoir gérer les différents paramètres de leur compte. ceux qui n'en possèdent pas peuvent en créer un.

Les store managers sont les comptes des gestionnaires de franchise. Chaque franchise a donc un compte lui permettant d'accéder à la gestion de leur magasin et aux statistiques.

Les Admins représentent la maison mère de The Cookie Factory, il s'agit d'un compte permettant de gérer globalement les données de la firme. Ce type d'acteur peut également consulter les statistiques.

Le diagramme de haut niveau fait également apparaître deux acteurs support : le système de paiement à distance et le système d'authentification des utilisateurs.

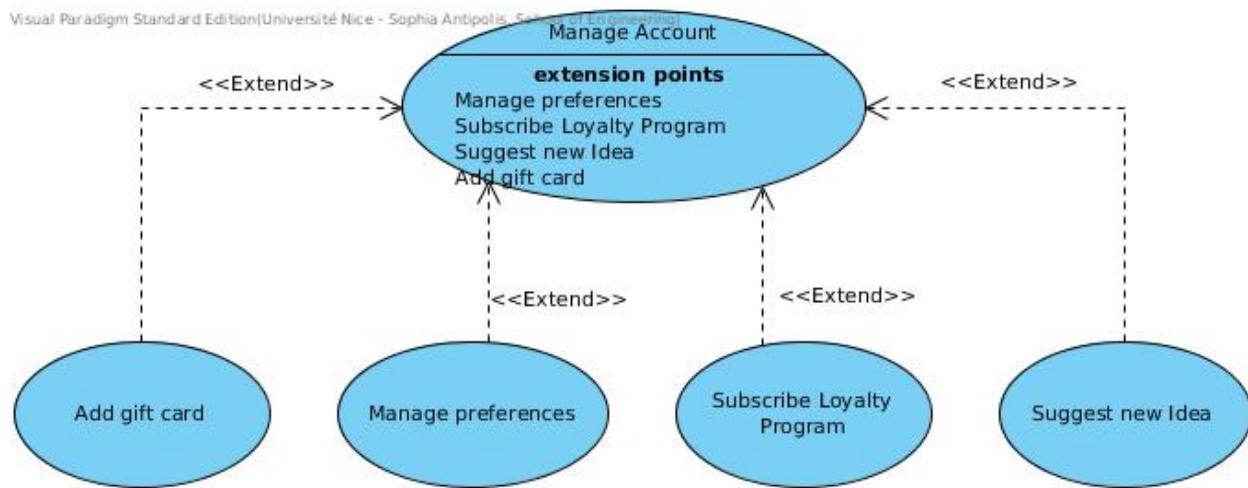
## 2. Make order



Pour réaliser une commande un client devra passer par plusieurs étapes :

- Le choix de recettes des cookies qu'il souhaite commander : cette étape est représentée par le cas d'utilisation "Choose merchandise". Ce choix peut comporter des cookies présents actuellement sur le menu et/ou des cookies qu'il aura créé lui-même (à partir de l'outil de création de cookies).
- Le choix de la quantité souhaitée pour chaque cookie.
- Le choix du magasin, de la date et de l'heure de retrait de la commande.
- Éventuellement l'utilisation de cartes cadeau.
- Le paiement (au moyen du système de paiement à distance).

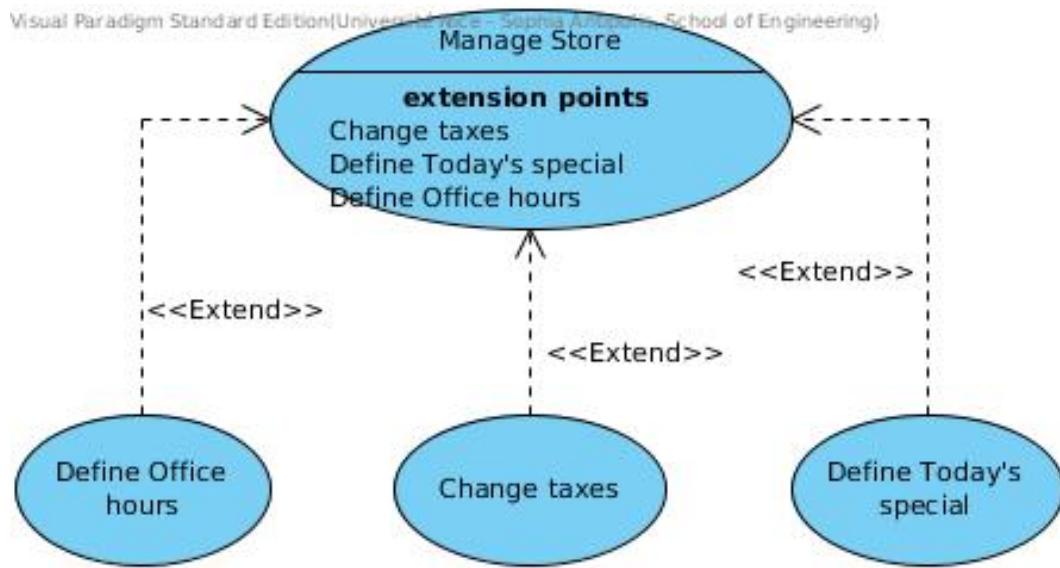
### 3. Manage Account



En tant que client enregistré, un utilisateur de Cookies on Demand a accès à diverses actions :

- a. Il peut ajouter un code de carte cadeau afin de s'en servir dans une commande future.
- b. Il peut gérer ses préférences (Recettes favorites, moyens de paiement utilisé, magasins de retraits).
- c. Il peut s'inscrire au Loyalty Program lui donnant accès à des réductions à partir d'un certain nombre de cookies achetés.
- d. Il peut poster une suggestion dans une boîte à idée, ces suggestions seront lues par les responsables de The Cookie Factory.

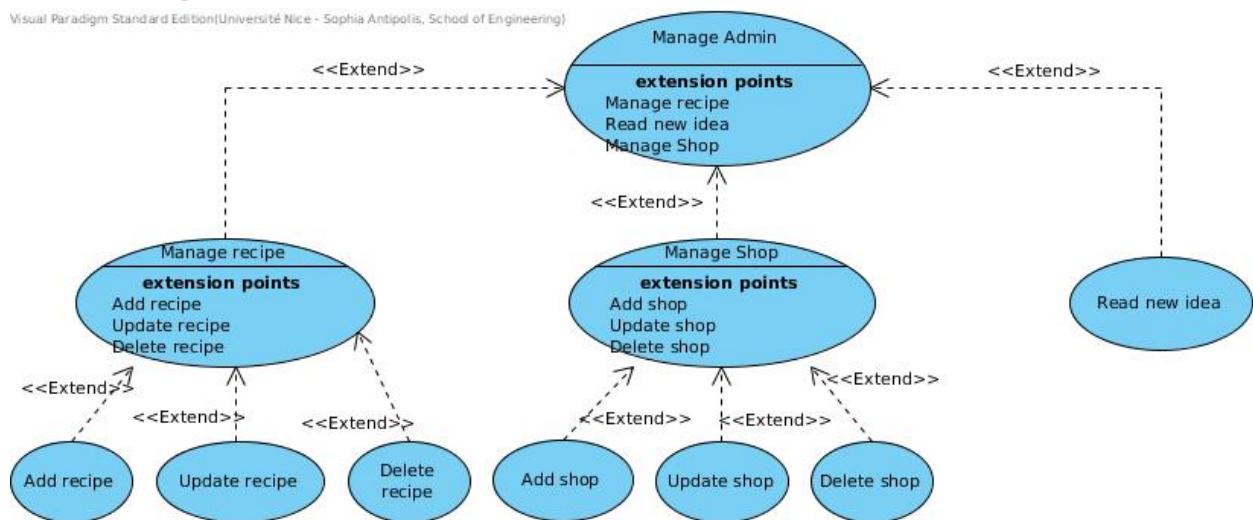
#### 4. Manage Store



Les propriétaires de magasins franchisés peuvent accéder à 3 types d'actions leur permettant de gérer leur magasin :

- Définir les horaires d'ouverture du magasin, afin de rendre impossible la planification d'un retrait de commande à des heures où le magasin est fermé.
- Modifier les taxes s'appliquant dans son magasin.
- Définir le Today's special.

#### 5. Manage Admin

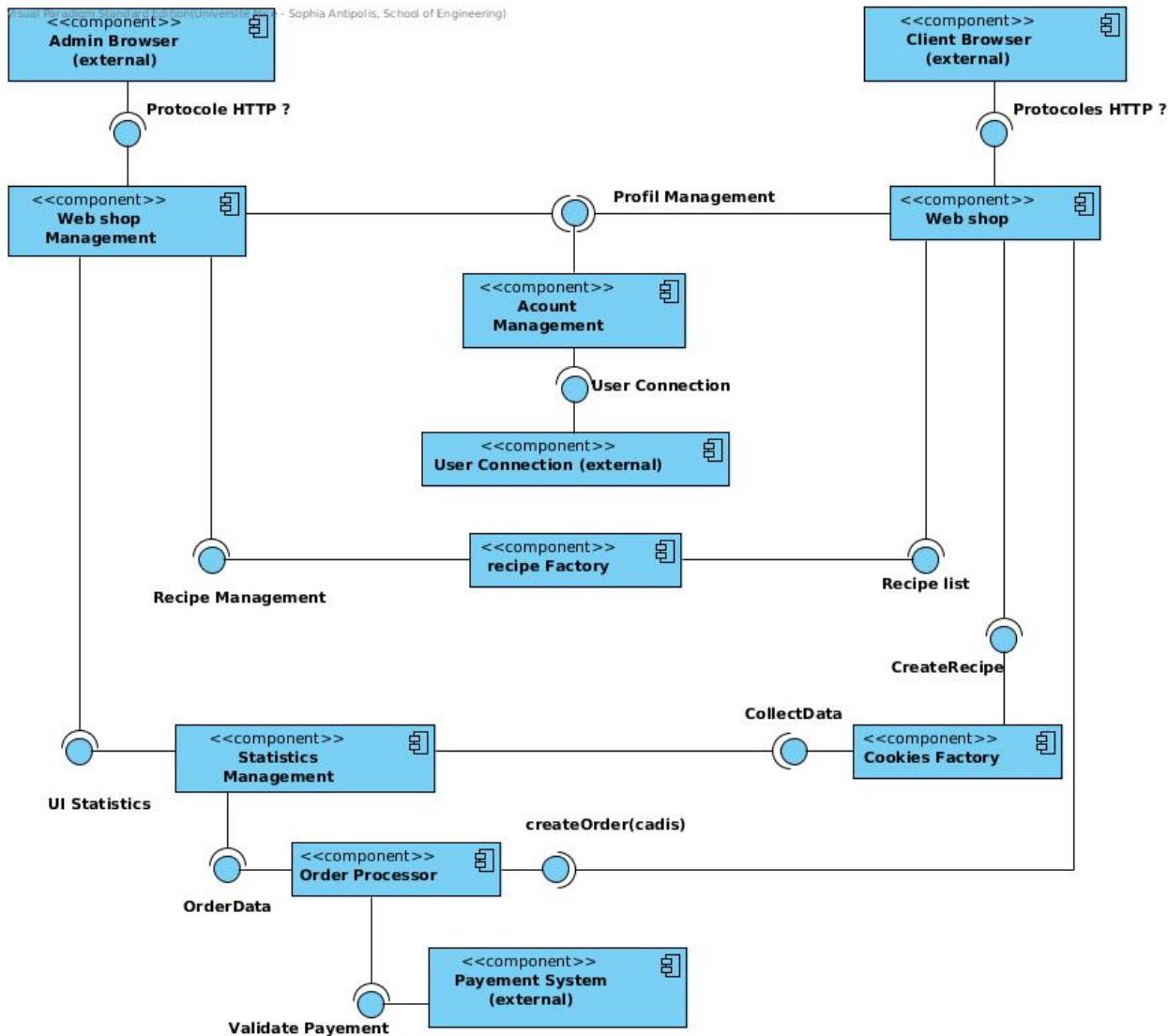


Un administrateur de Cookies on Demand est un représentant de la firme The Cookie Factory, il peut gérer :

- a. La liste des recettes (Ajouter, modifier ou supprimer une recette).
- b. La liste des magasin (Ajouter, modifier ou supprimer un magasin).

Un administrateur peut également avoir accès à la liste des suggestions des clients enregistrés.

# Diagramme de composants



Profil Management prototype (AccountManagement -> WebShop): C'est la partie des composants traitant de l'ensemble des fonctionnalités additionnelles d'un client enregistré contrairement au client anonyme. Il pourra ainsi gérer des préférences ainsi que rentrer des code cadeaux et suggérer des idées de recettes auprès de la société.

```

void addNewFavouriteRecipe(String recipeName);
void deleteFavouriteRecipe(String recipeName);
void suggestIdea(String request);
boolean addCredits(int code);

```

Profil Management prototype (AccountManagement -> WebShopManagement): c'est la partie gestion pour chaque magasin franchisé. Il pourra définir les heures ouverture/fermeture, définir le today's spécial du jour et lire l'ensemble des commandes de la journée/semaine/mois.

```
void setCloseDate(Date d);  
void setOpenDate(Date d);  
void chooseTodaySpecial(String name);  
Order getOrderByDay(Date d);  
Order getOrderByWeek(Date d);  
Order getOrderByMonth(Date d);
```

User Connection : ce composant s'occupe de tout ce qui concerne la gestion de compte utilisateur de manière global peu importe le type d'utilisateur. L'utilisateur peut donc se connecter et se déconnecter ainsi que s'inscrire.

```
boolean signIn(String email, String password);  
boolean logIn(String email, String password);  
boolean logOut();
```

RecipeList prototype : Cette interface permet juste au client de récupérer la liste des recettes préexistantes créée par la société c'est un simple affichage des recettes.

```
void getRecipe();
```

CreateRecipe prototype : Cette interface permet l'élaboration d'une recette personnalisée.

```
void createRecipe(String name, Dough d, Flavour f,  
                  List<Topping> t, Mix m, Cooking c);
```

CollectData prototype : Cette interface permet de récupérer les recettes client afin par la suite de pouvoir être visualisé par la société.

```
void sendRecipe(Recipe recipe);
```

CreateOrder prototype : Cette interface concerne l'ensemble des fonctionnalités disponible pour l'exécution d'une commande. Ajout de recette, le choix de la quantité pour chaque recette et l'impression de bon de commande. Le paiement en lui même repose sur le service

externe en .NET dont nous récupéreront les méthodes via son interface pour le paiement de la commande (Cf interface ci-dessous).

```
void addRecipe(String recipeName);  
void deleteRecipe(String recipeName);  
void chooseQuantity(String recipeName, int quantity);  
void makeBill();
```

Validate Payment prototype : Permet le paiement de la commande.

```
boolean payment(int numCard, int crypto, int prices);
```

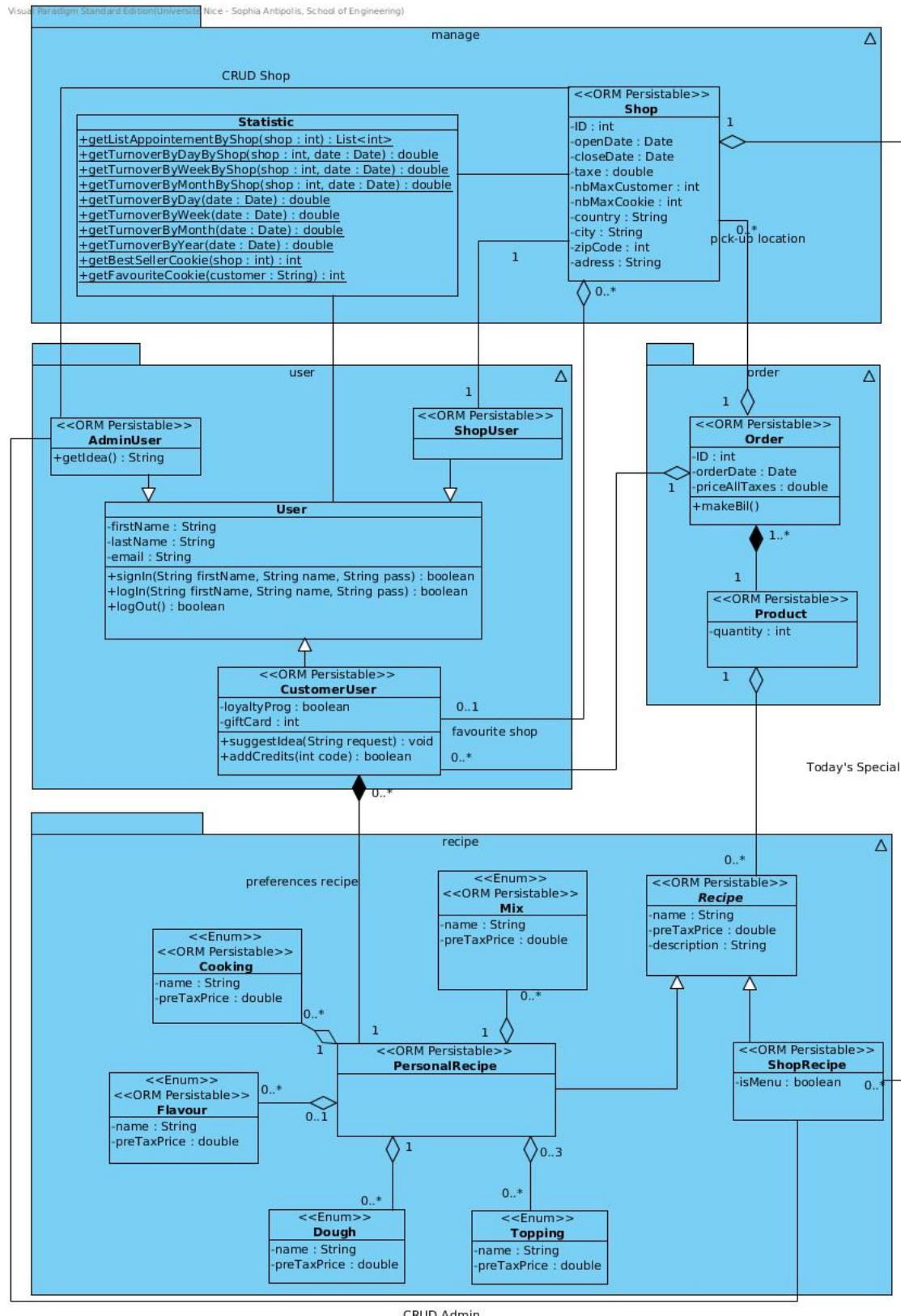
OrderData prototype : Chaque commande est enregistrée en temps que statistique pour le calcul futur de chiffre d'affaire mais aussi la réception par le shop pour pouvoir commencer la conception des cookies.

```
void sendBill(Date d, double price, List<Recipe> c);
```

UI Statistics prototype: Interface permettant la récupération pour les administrateurs et les magasins de récupérer l'ensemble des statistiques (global et par magasin pour les AdminUsers tandis uniquement les statistiques du magasin pour les ShopUsers).

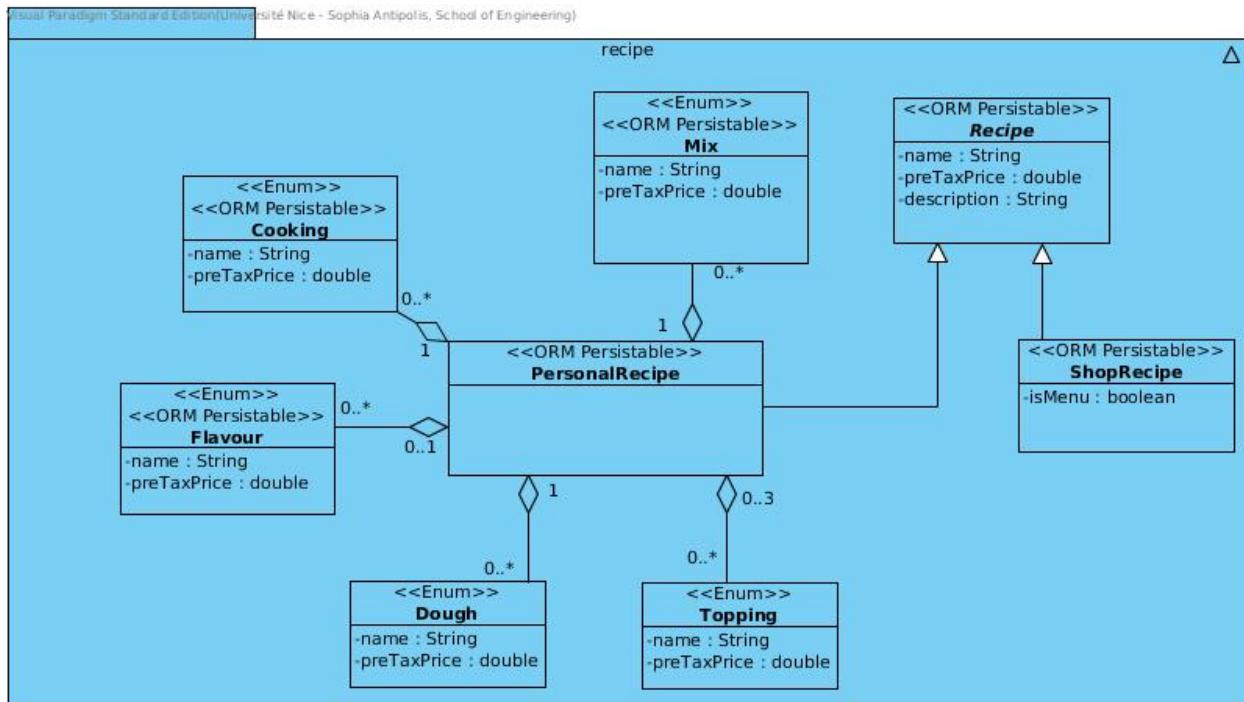
```
void getListAppointementByShop(int ID);  
double getTurnoverByDayByShop(int ID, Date d);  
double getTurnoverByWeekByShop(int ID, Date d);  
double getTurnoverByMonthByShop(int ID, Date d);  
  
double getTurnoverByDay(Date d);  
double getTurnoverByWeek(Date d);  
double getTurnoverByMonth(Date d);  
double getTurnoverByYear(Date d);
```

# Diagramme de classe



Nous avons découpé nos classes en quatre packages :

### 1. Package recipe



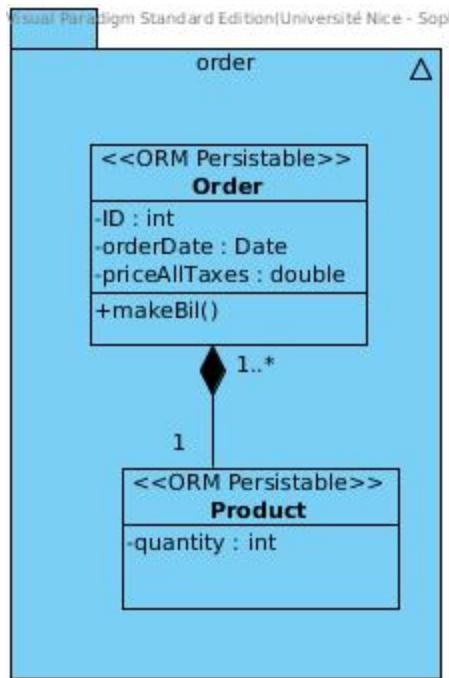
Le package recipe regroupe toutes les classes concernant les recettes de cookies. La classe Recipe représente une recette, qu'elle fasse partie du menu ou qu'elle ait été créée par un client. Chaque recette possède un nom, une description et un prix hors-taxes.

Un objet de type PersonalRecipe est une recette créée par un client, sa composition respecte obligatoirement les contraintes fournies par The Cookie Factory.

Bien qu'une PersonalRecipe soit composée de Cooking, Dough, Flavour, Mix et Topping, la relation entre ces classes et PersonalRecipe est une agrégation car nous avons choisi de représenter ces composants par un type énuméré (qu'on ne détruira donc pas lorsque l'on détruira une PersonalRecipe) afin d'éviter d'avoir à créer de nombreux objets identiques.

Un objet de type PersonalRecipe est une recette de TCF (ne respectant donc pas forcement les contraintes de composition). Nous avons fait le choix de représenter le menu (les cookies actuellement disponibles à l'achat) par un booléen associé à chaque recette de TCF plutôt que par une classe Menu qui regrouperait les cookies du moment. Ce choix a été fait dans le but de simplifier le modèle et donc d'accélérer le développement et la sortie de la première version de CoD.

## 2. Package order

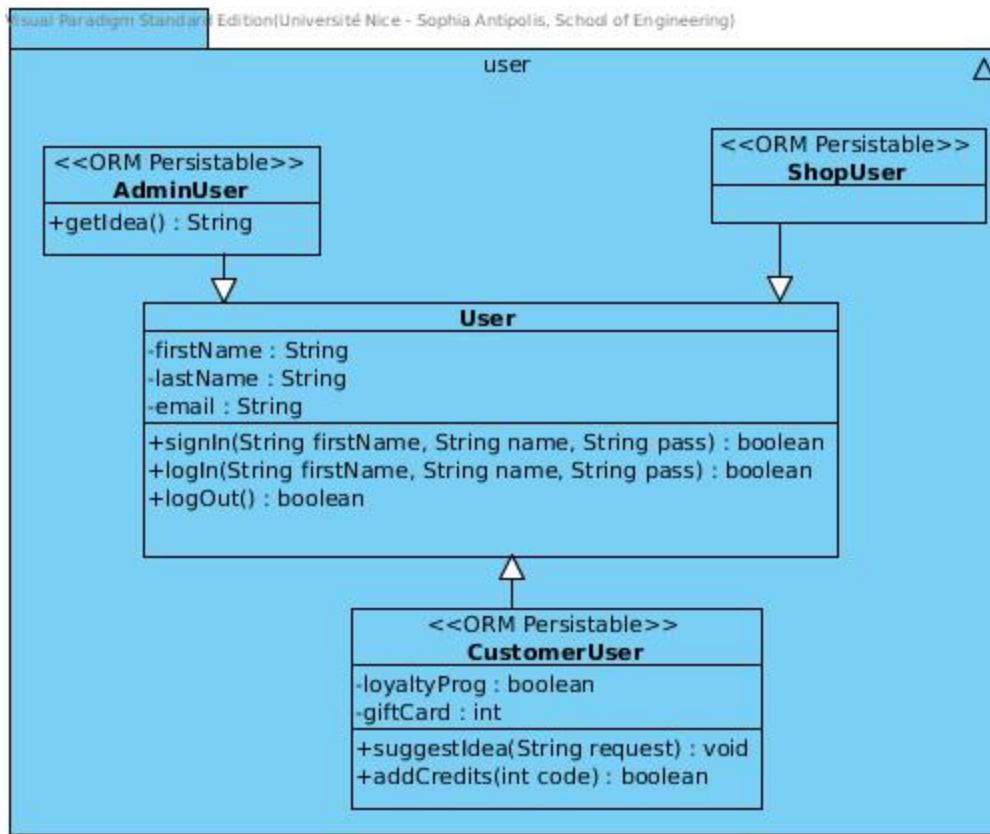


Le package order contient les classes nécessaires à la réalisation et à l'identification d'une commande.

Les données importantes d'une commande sont l'identifiant du client qui l'a passée, sa date et son lieu de retrait, les cookies commandés et son prix total (calculé à partir du prix HT des cookies commandés et des taxes en vigueur dans la région du magasin de retrait).

Chaque commande est composée d'au moins un objet de type Product, qui est l'association d'une recette et d'une quantité. Il y aura autant d'instances de Product associées à une commande que de recettes différentes dans cette commande.

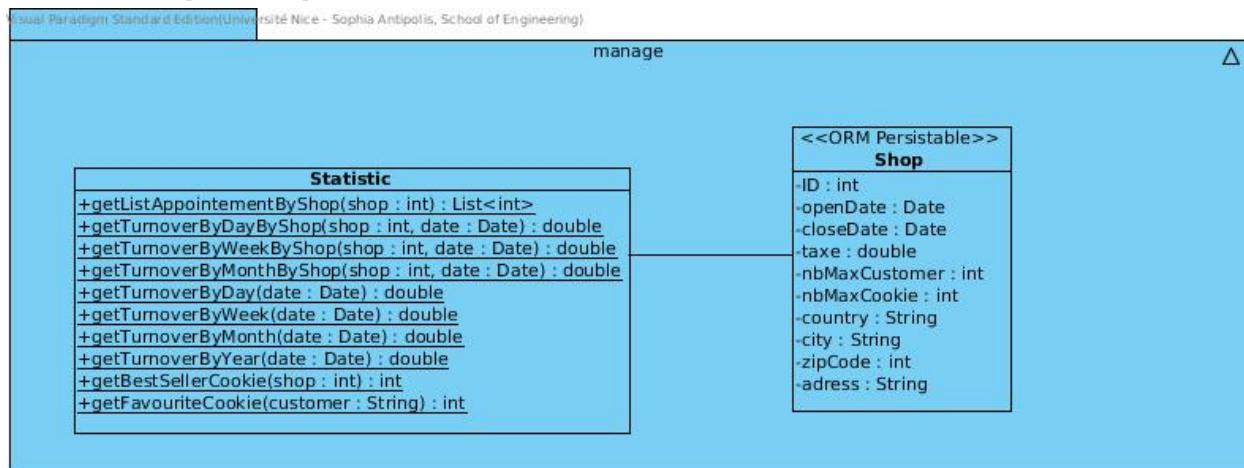
### 3. Package user



Le package user est composé des classes correspondantes aux différents types d'utilisateurs de Cookies on Demand, ainsi que d'une classe User qui généralise les 3 types d'utilisateurs pouvant interagir avec le système :

- Les CustomerUser qui sont les clients enregistrés. Ils peuvent ajouter des cartes cadeaux, déposer une idée dans la boîte à idée, passer une commande et gérer leurs préférences (magasin de retrait favoris, liste de recettes préférées, ...).
- Les ShopUser, qui sont les gestionnaires de magasins. Ils peuvent gérer les horaires d'ouverture de leur magasin et définir le Today's special.
- Les AdminUser, qui sont les représentants de la maison mère de TCF. Ils peuvent modifier la liste des recettes ainsi que modifier la carte (la liste des recettes actuellement disponibles). Ils peuvent également lire les idées déposées par les CustomerUser. Une idée n'étant pour le moment rien d'autre qu'une chaîne de caractères, nous avons décidé de ne pas créer de classe Idea pour la première version de CoD (toujours dans un but de simplification du modèle).

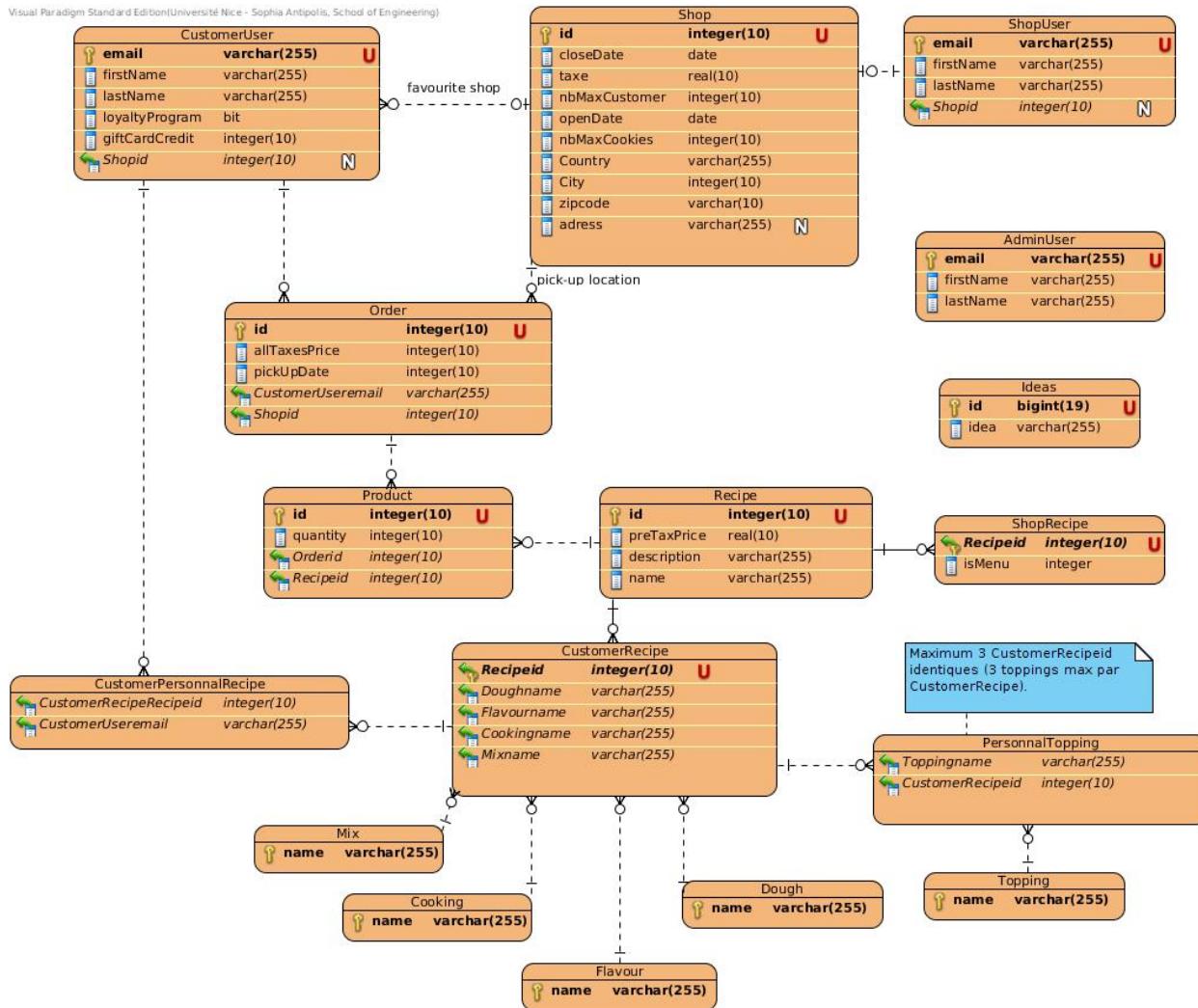
#### 4. Package manage



Le package manage représente les objets de gestion et de visualisation de l'entreprise en interne.

Un Shop est un magasin physique de TCF, il s'agit d'un potentiel lieu de retrait pour les clients et possède donc des horaires d'ouverture, de fermeture, une adresse, une production de cookies maximum (par heure) et un nombre maximum de clients pouvant passer retirer une commande (par quart d'heure) et la taxe s'appliquant dans ce magasin.

# Modèle relationnel



Afin de pouvoir gérer la persistance des données, nous avons décidé d'utiliser le modèle relationnel ci-dessus pour la base de données de notre solution.

Pour gérer les associations 1..N, nous avons stocké la clé primaire dans l'entité ayant une seul association possible, pour la gestion du point de retrait préféré d'un client ou de la composition d'une recette créée par un client (hors Topping) par exemple.

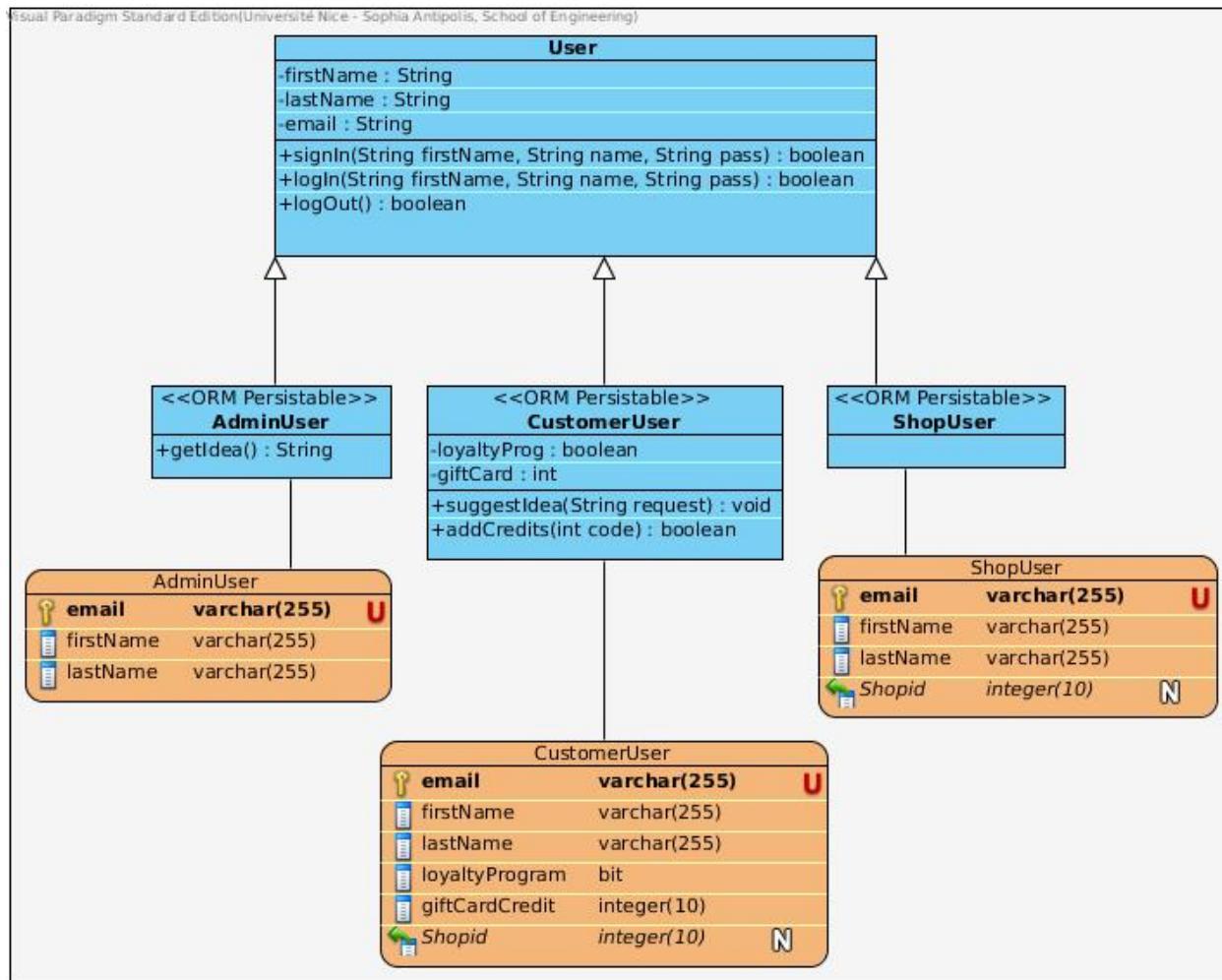
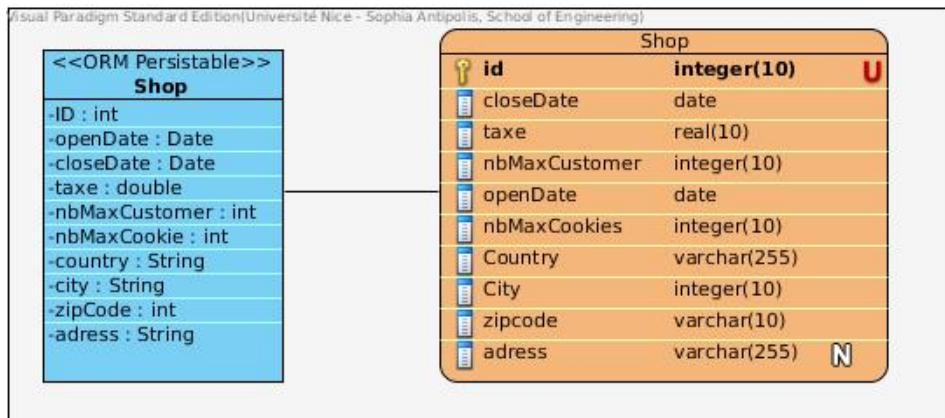
Nous avons également créé une table pour la gestion de chaque association de type N..M, ce qui est le cas pour les Toppings d'une recette client.

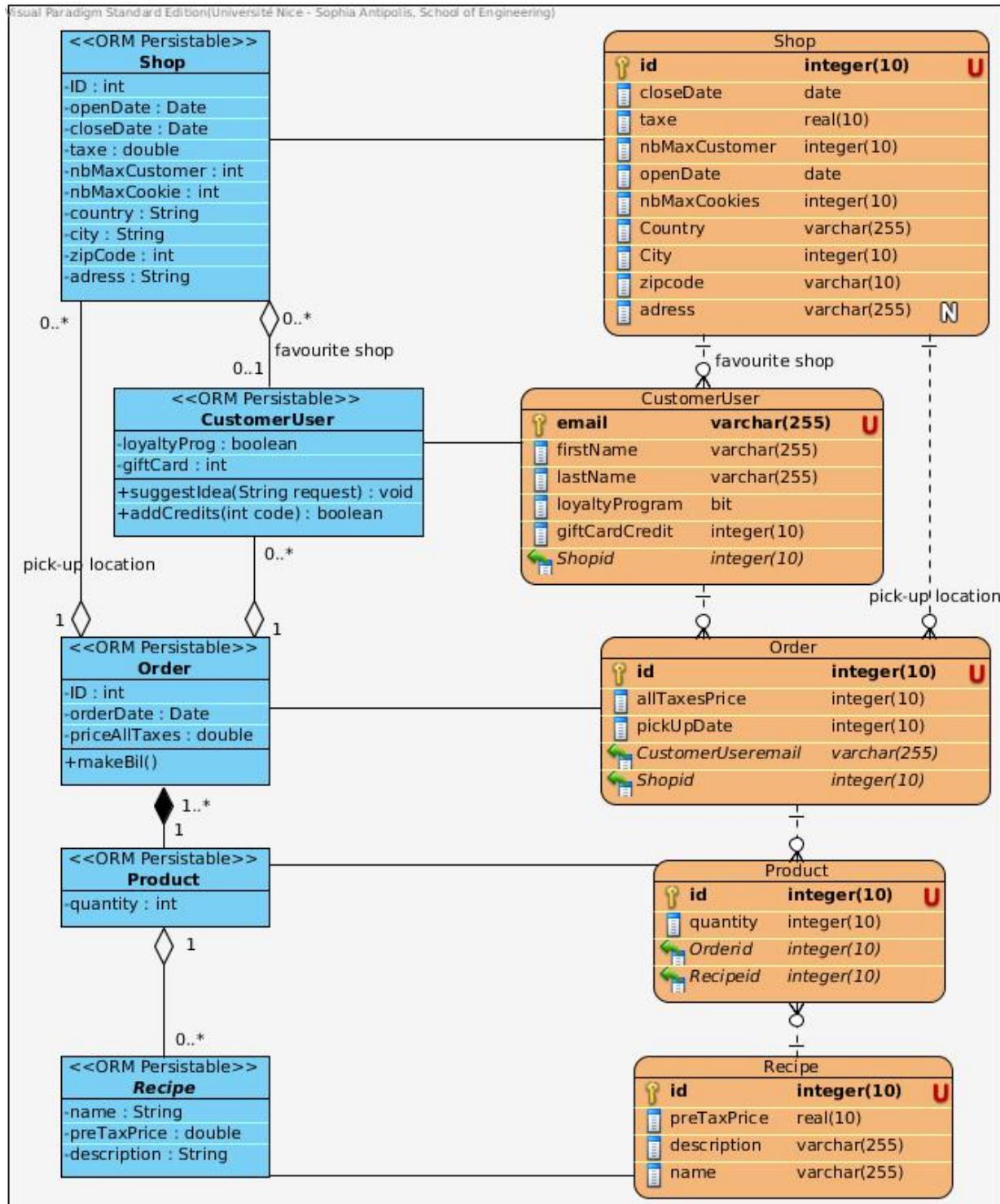
Le stockage des suggestions des clients se fait dans l'entité Idea, qui ne possède pour le moment rien d'autre que l'id de l'idée et le texte associé.

Les recettes sont stockées telles que représentées dans le diagramme de classe, tous les attributs communs seront stockés dans une relation, et deux autres relations stockent les attributs spécifiques aux recettes de TCF et aux recettes créées par les clients.

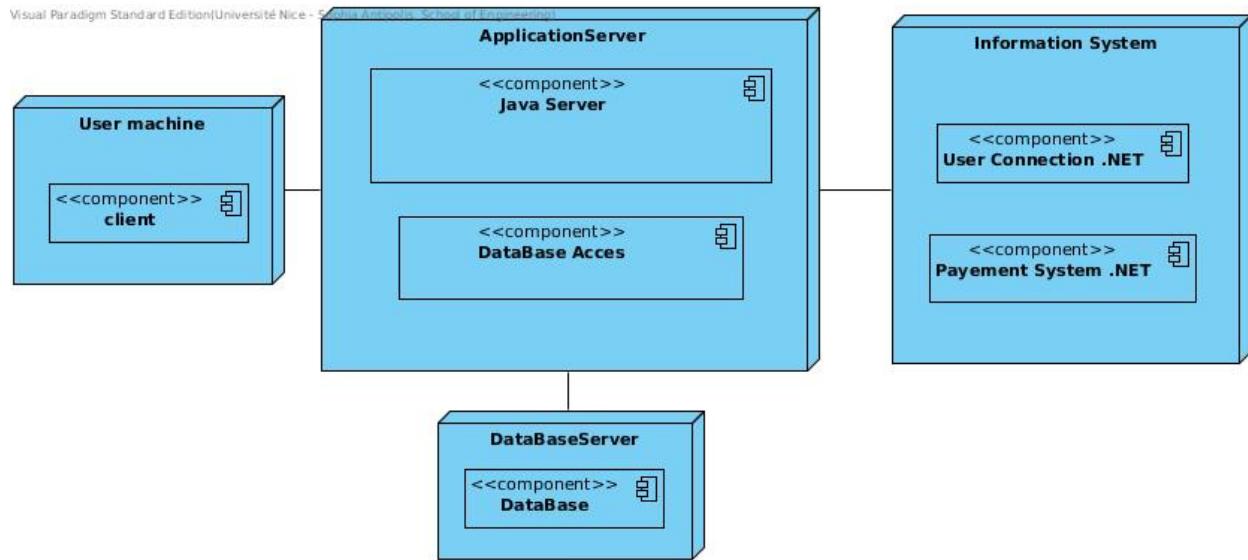
Les utilisateurs sont stockés différemment, nous avons décidé de ne pas créer d'entité générale “user”, mais de stocker tous les attributs de chaque utilisateur dans une des 3 relations contenant les utilisateurs.

# Mapping objet-relationnel





# Déploiement



Voici notre architecture machine pour le déploiement de CoD:

- Une machine principale sur laquelle sera déployée l'application JAVA CoD, elle devra bien entendu disposer d'une installation correcte, stable et récente de la JVM (Java Virtual Machine)
- La base de données sera hébergée sur un serveur dédié. Ainsi, si la machine contenant CoD ne fonctionne plus, la base de données restera accessible et utilisable et ne sera donc pas affectée par l'arrêt de la machine principale.
- Le client sera installé sur la machine des utilisateurs (client spécifique ou navigateur web)
- Enfin les système de paiement et de connections sont déjà déployés en interne chez Cookies Factory, nous ignorons à ce jour l'état et la puissance de chez machine, cependant nous savons que ces systèmes sont développés dans un environnement .NET.

Le but de ce déploiement sera de fournir facilement un environnement évolutif et maintenable.