SI4

Rapport d'architecture Cookie On Demand The Cookie Factory

SERVEURS D'ENTREPRISE

MARONGIU ANAIS, STROBBE ETIENNE, PETILLON SEBASTIEN

Table des matières

Introduction

“Il est inconcevable pour nous de ne pas pouvoir satisfaire
100% de nos clients, pour 100% de leurs envies”

-- John Mc Cook, fondateur de The Cookie Factory

The Cookie Factory, leader aux USA dans la fabrication de cookies artisanaux, a décidé d’ouvrir un nouveau service de Cookies on Demand (CoD). Entre le e-Shop et le drive-in, ce service permettra aux clients de commander des cookies déjà existants ou des cookies qu’ils pourront composer en choisissant parmi une sélection d’ingrédients. Ils pourront ensuite spécifier la date et l’heure à laquelle ils viendront les chercher pour les déguster tout chaud.

Pour réaliser ce service et couvrir les besoins de l’entreprise, nous proposons un produit dont l’architecture est décrite dans ce rapport.

Environnement et contraintes

Déjà leader aux USA, The Cookie Factory se veut encore plus innovant avec leur nouveau service qui leur permettra de toujours proposer à ses clients leurs recettes de cookies préférées en temps et en heure.
Ce service doit donc garantir une extrême fraicheur des cookies tout en évitant de mauvaises surprises comme une rupture de stock d'ingrédients ou de l'attente dans le magasin.

En plus de pouvoir réaliser l'ensemble des fonctionnalités proposées par ce service, plusieurs contraintes sont nécessaires.
Baignant dans un contexte .Net l'entreprise pense à migrer vers des technologies Java. Ce projet doit leur permettre de valider l'adéquation de Java à leurs problématiques.
Par ailleurs, le système devra utiliser les deux services déjà existants : le système de paiement à distance et celui d'authentification des utilisateurs. Ces deux systèmes devront donc être interfacés avec la solution Java apportée.
Voici un schéma montrant différents éléments qui vont entrer en compte dans l'architecture.

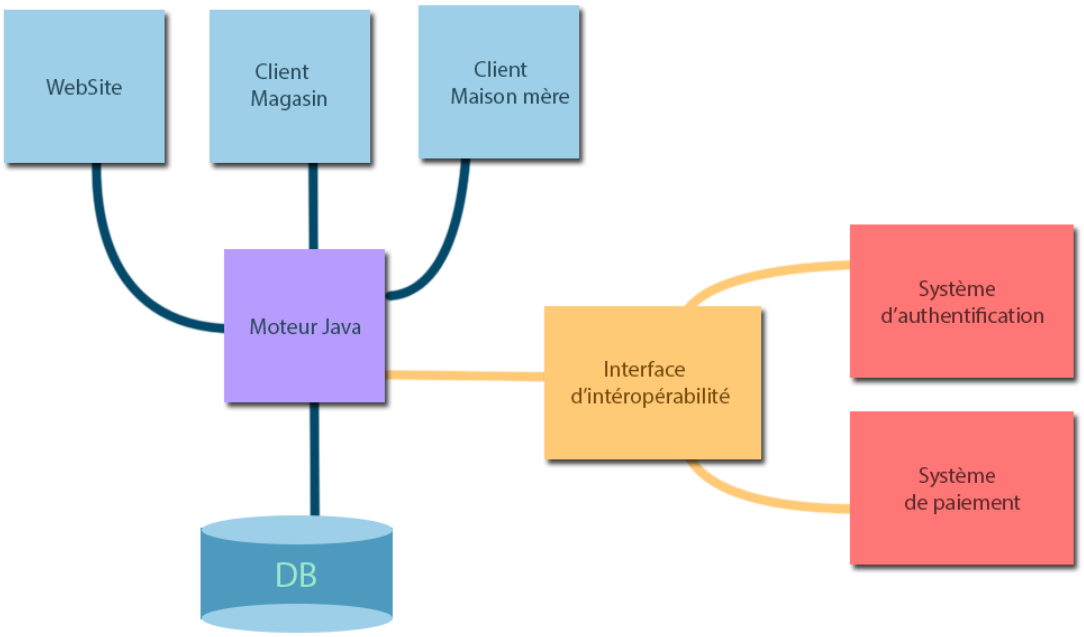

Figure 1 : Schéma représentant des éléments de l'architecture

Le site web et les deux clients devront permettre d'utiliser les fonctionnalités du système. Le moteur exécutera la partie métier de l'application en se servant de la base de données. Enfin, l'interface d'interopérabilité permettra au moteur Java d'interagir avec les systèmes .NET préexistants.

Scénarios

Pour bien comprendre le contexte d'utilisation et pour assurer toutes les fonctionnalités de ce nouveau service, voici quelques exemples de scenarios qui devront être satisfaits.

Scenario 1 : Commande (non inscrit)

Un utilisateur non inscrit effectue une commande d'un cookie « Chocolalala » et d'un cookie personnalisé. Il compose son cookie personnalisé d'une pâte au chocolat, d'un arôme à la vanille et d'une garniture mélangée de chocolat blanc et de chocolat au lait. Il choisit une cuisson moelleuse et valide la commande. Il choisit la boutique proche de chez lui, sélectionne l'horaire parmi les créneaux qui lui sont proposés et effectue le paiement.

Scenario 2 : Inscription, préférences et adhésion

Un utilisateur non inscrit et satisfait de sa première commande, décide de s'inscrire. Il renseigne ses informations d'inscription et décide également de renseigner sa boutique préférée ainsi que ses informations de paiement. Il décide ensuite d'adhérer au programme de fidélité pour profiter des avantages.

Scenario 3 : Commande (inscrit)

Un utilisateur inscrit commande un cookie qu'il a précédemment créé et enregistré. Il se connecte et accède à ses créations. Il commande son cookie personnalisé et profite du fait qu'il a déjà enregistré sa boutique préférée et ses informations de paiement dans ses préférences , pour simplement choisir l'heure qui lui convient parmi les créneaux qui lui sont proposés.

Scenario 4 : Cadeau

Un utilisateur inscrit offre un cadeau à un ami. Il se connecte et accède à la fonctionnalité d'offre de cadeaux. Il renseigne les informations de son ami ainsi que le montant à offrir et valide la transaction.

Scenario 5 : Enregistrement de création et proposition d'ingrédient

Un utilisateur inscrit veut enregistrer une création de cookie. Il se connecte et accède à la fonctionnalité de personnalisation de cookie. Il compose son cookie d'une pâte nature et d'une garniture mélangée de chocolat blanc. Cependant il aurait bien voulu ajouter des pépites de caramel en garniture. Il propose alors d'ajouter un ingrédient et renseigne le caramel. Il choisit une cuisson croustillante et enregistre son cookie pour le commander la prochaine fois.

Scenario 6 : Paramètre de la boutique

Un responsable de magasin veut configurer les paramètres pour sa boutique. Il se connecte en tant que responsable de magasin puis définit les horaires, spécifie les taxes et choisit le prochain cookie du jour (Today's special).

Scenario 7 : Traitement d'une commande

Un employé veut traiter une commande. Il consulte la commande à traiter puis lance la préparation des cookies pour cette commande. Lorsque le client arrive, les cookies sortent justement du four ! Il peut donc servir le client puis valider la commande.

Scenario 8 : Statistiques

Un responsable de magasin veut consulter les statistiques. Il se connecte en tant que responsable de magasin et regarde le chiffre de vente de la journée, de la semaine, du mois et de l'année. Il regarde aussi quelles sont les recettes préférées de ses clients.

Conception

Cas d'utilisation

Dans ce diagramme de cas d'utilisation de haut niveau apparaissent les principales fonctionnalités proposées par le nouveau service de Cookie on Demand.

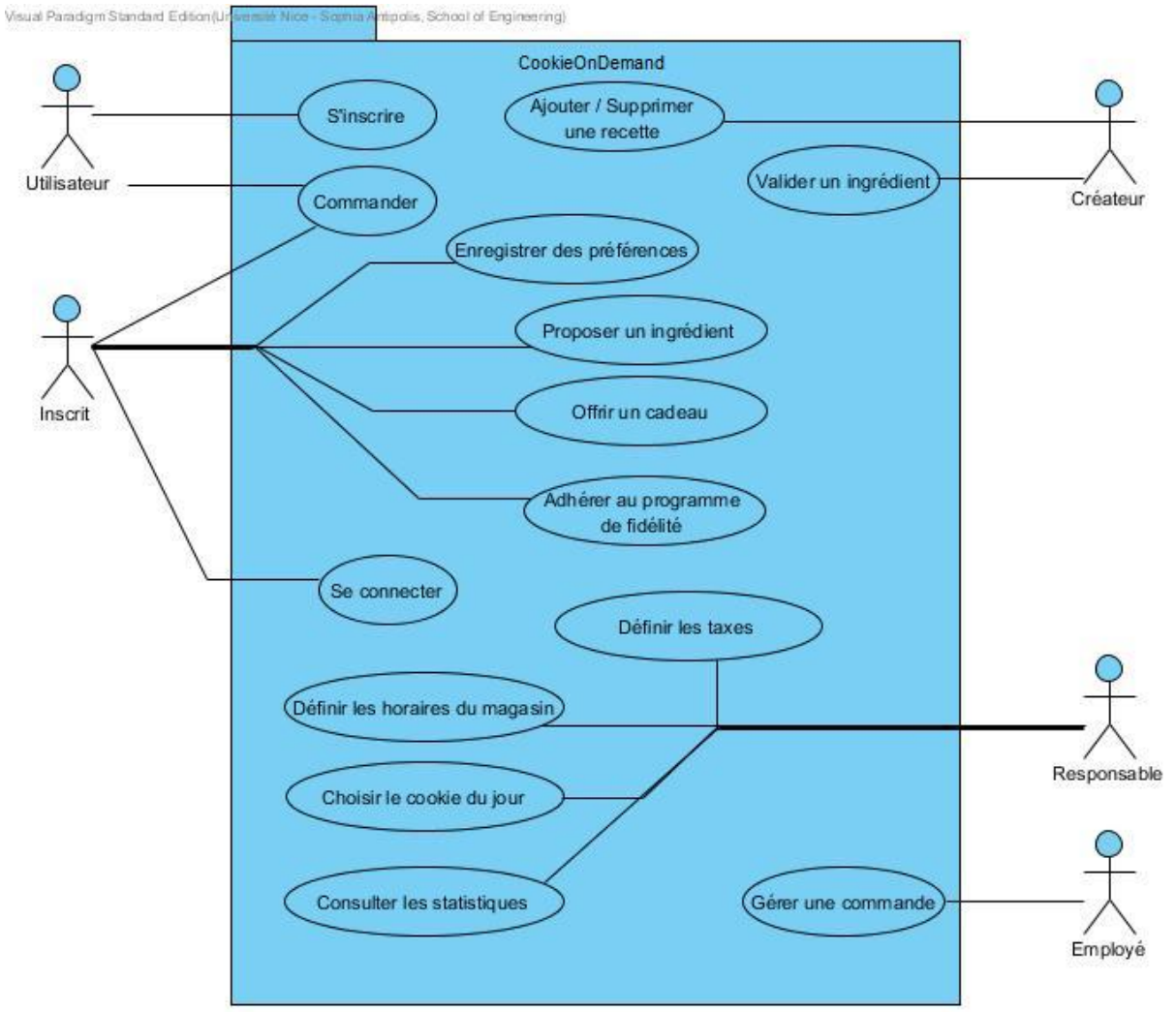

Figure 2 : Diagramme de cas d'utilisation du CoD

Nous avons identifié cinq utilisateurs. Trois d'entre eux sont en lien direct avec l'entreprise : le créateur (au niveau de la maison mère) peut gérer les recettes, le responsable gère le magasin et l'employé gère les commandes.

Les deux autres sont des clients. Le simple utilisateur (non inscrit) pourra uniquement commander, mais s'il décide de s'inscrire il devient alors un « utilisateur inscrit » et pourra utiliser le reste des fonctionnalités clients du système.

Ici, nous avons détaillé le cas d'utilisation « commander ». Comme décrit dans les scenarios, l'utilisateur peur composer son cookie ou en choisir un déjà existant.

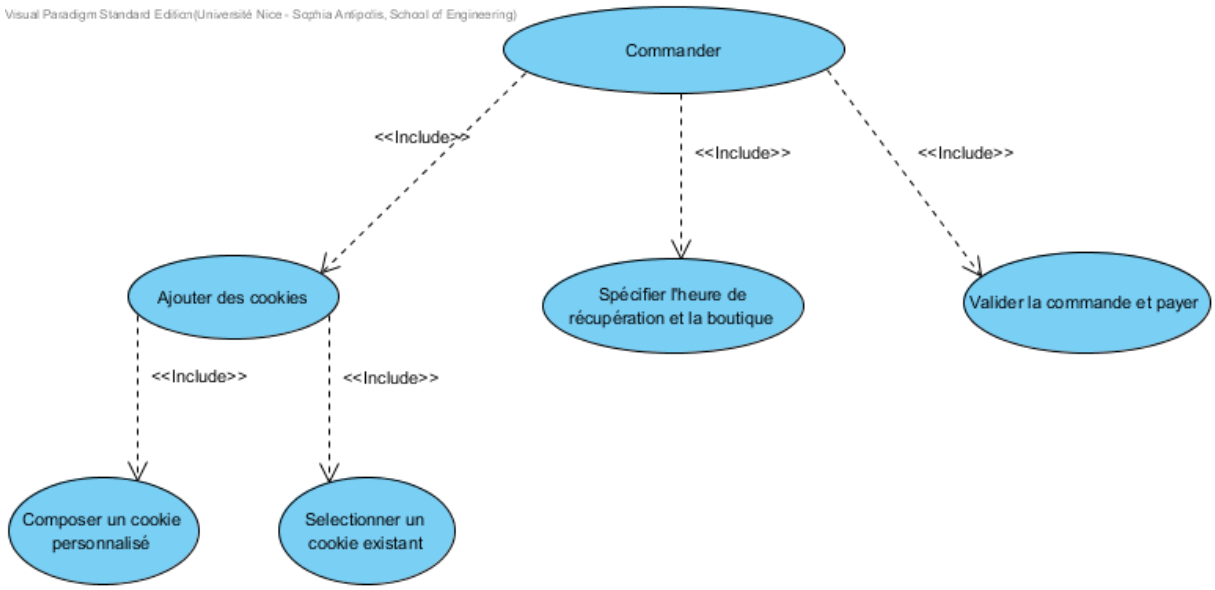

Figure 3 : Diagramme de cas d'utilisation détaillé de commander

Diagramme de classes

Voici le diagramme de classes du modèle qui représente les données.

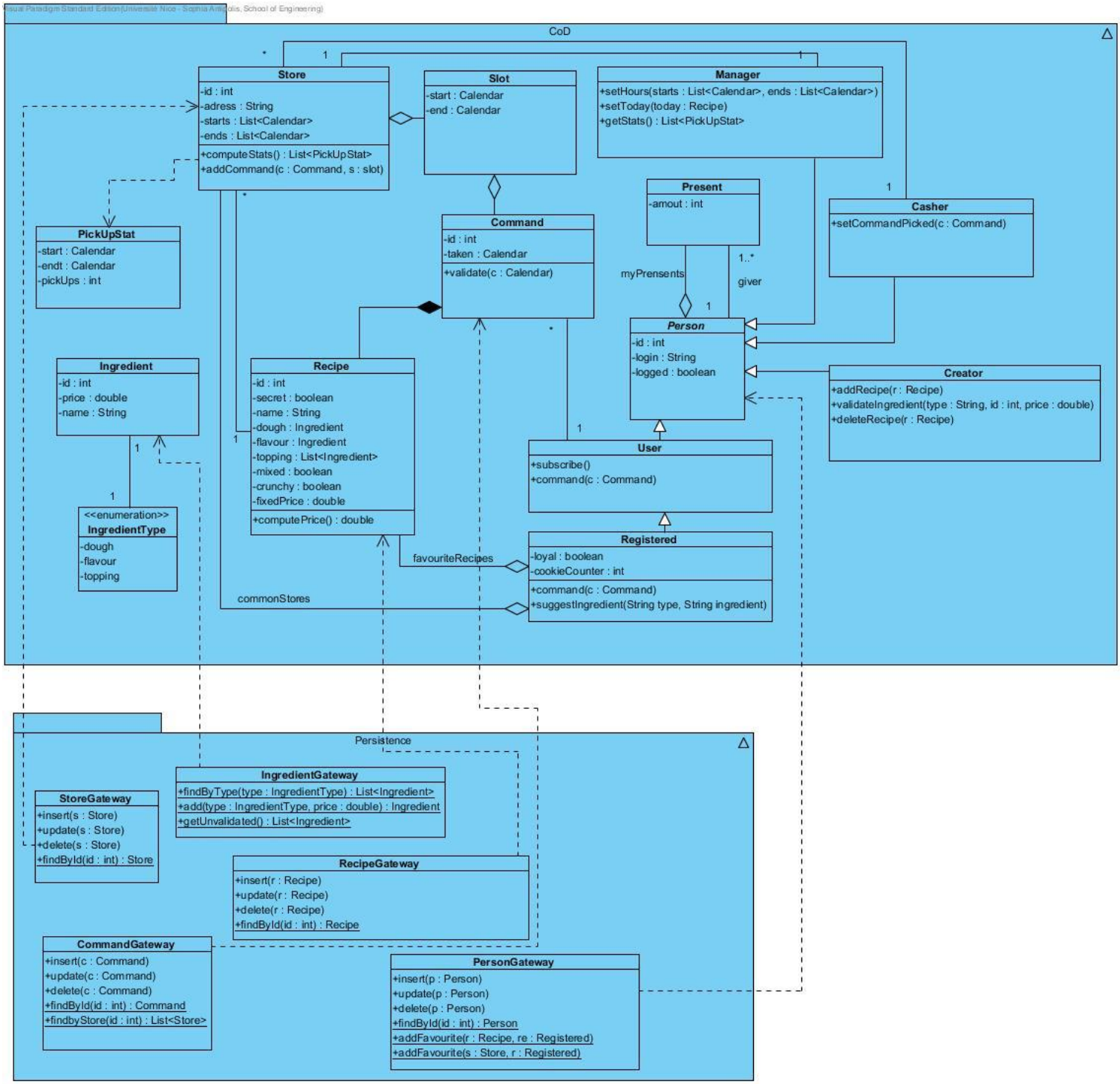

Figure 4 : Diagramme de classes du CoD

Choix de conception

Pour modéliser notre projet, nous avons effectué de nombreux choix que nous allons expliquer ici.

Premièrement, nous avons identifié les entités phares du projet : les magasins, les différents utilisateurs, les commandes et les recettes. Puis nous avons décidé de séparer nos classes en deux packages.
Le premier concerne les différentes entités nécessaires pour représenter les données de notre modèle.
Le second regroupe les passerelles dédiées (« gateway »). Cette méthode, qui nous a été présentée en cours, permet le lien avec les données qui seront enregistrées dans la base. Cela implique de rendre des méthodes statiques mais l'organisation sera plus simple.

Nous avons fortement découpé les utilisateurs de manière à répartir correctement les comportements autorisés. Nous représentons donc les utilisateurs avec de l'héritage. Les différentes classes d'utilisateur (Casher, Creator, Manager, User et Registered) héritent toutes de Person puis ont leurs méthodes spécifiques.
L'utilisateur, s'il est inscrit, a des préférences modélisées par les agrégations. S'il est un employé (Casher ou Manager), il se voit associé un Store dans lequel il effectue ses actions.

Naturellement, les commandes (classe Command) sont composées de recettes de cookie (classe Recipe). C'est un lien de composition fort puisque qu'une commande de cookie n'a de sens si elle ne contient pas de cookies.

Les recettes de cookies ont des ingrédients (classe Ingredient). Pour représenter les ingrédients, nous avons choisi une énumération plutôt qu'une classe par ingrédient car les ingrédients n'ont pas de comportement.

Les magasins (classe Store) gèrent les commandes par créneaux (classe Slot). Nous avons donc choisi de représenter les créneaux comme étant composés de commandes et non l'inverse. En effet du point de vue du magasin qui gère les commandes, il est intéressant de ranger ces dernières dans des créneaux et d'avoir un accès direct sur l'ordre de traitement. Le Store a donc pour chaque Slot un certain nombre de commandes, toutes reliées à un User et composées d'un certain nombre de Recipe.

Par ailleurs, les apparitions de la classe Calendar dans le diagramme seront certainement remplacées par une classe plus pertinente, mais elle est ici dans le but de montrer qu'il s'agit d'une entité temporelle.

Enfin, les préférences de l'utilisateur inscrit seront générées par le système et sont modélisées par les agrégations de Store et de Recipe dans le Registered.

Diagramme entité relation

Voici le diagramme d'entité relation qui décrit la manière dont seront stockées les données.

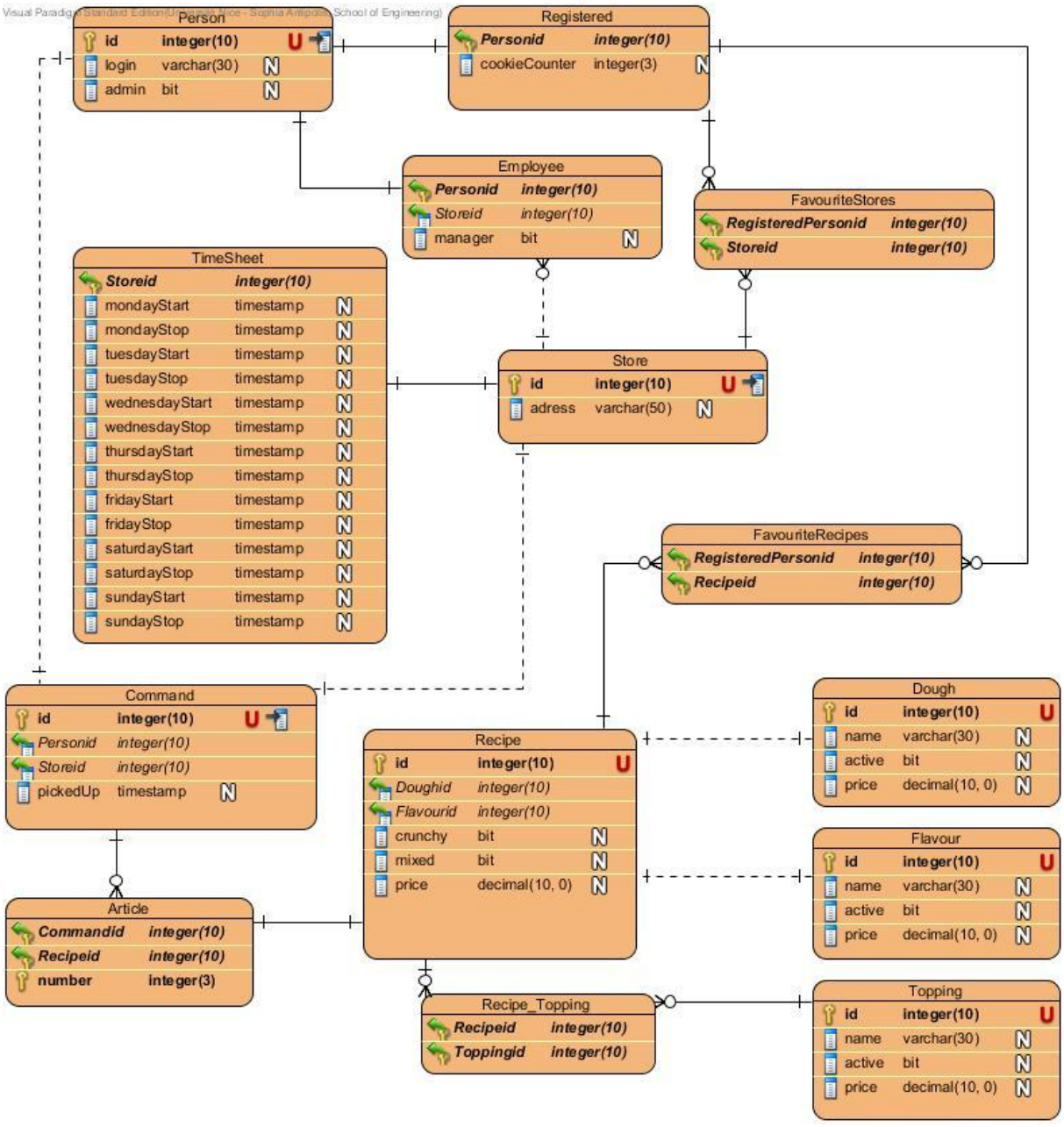

Figure 5 : Diagramme d'entité relation du CoD

Pour garantir l'unicité de nos attributs primaires, nous avons décidé d'ajouter un attribut id à toutes nos tables. Cet identifiant n'est pas destiné à être visible par les utilisateurs finaux, il ne rentre en compte que de manière interne.

Pour modéliser nos 5 acteurs, nous les avons réparties en 3 tables qui nous semblaient pertinentes.

Les utilisateurs inscrits et les utilisateurs fidélisés sont regroupés dans la table Registered. Comme ils peuvent alors avoir des préférences (recettes, magasins), ils sont reliés aux tables Store et Recipe par des tables d'association. Ce sont des relations N-N.

Les employés et responsables de magasin sont regroupés dans la table Employee et sont affectés à une boutique.

Nous avons préféré séparer les utilisateurs en différentes tables pour ne pas avoir une seule et gigantesque table de tous les utilisateurs du système. Bien que cela implique plus de jointures, les liens entre les autres tables sont plus pertinents en fonction du type de l'utilisateur.

Pour la table Store, nous avons décidé de la liée à une table d'horaires de la boutique. Les horaires étant définies par semaine, nous avons choisi de stocker pour chaque jour de la semaine, l'heure d'ouverture et de fermeture. Cette représentation peut-être un peu lourde à première vue, al'avantage d'être simple et de ne pas s'encombrer de dates.

Dans la table Recipe, nous stockons toutes les recettes (les recettes préexistantes et les recettes personnalisées créées par les utilisateurs). Ces recettes sont composées d'ingrédients qui sont répartis dans des tables pour chaque type d'ingrédients. Il y a une particularité pour la table Topping qui est reliée avec une table d'association car il peut y avoir plusieurs garnitures dans un même cookie.

Enfin, les commandes sont naturellement liées à une personne et une boutique et elles ont un certain nombre d'articles (recettes de cookie et quantité) ainsi que la date de récupération prévue.

Finalement, ce modèle d'entité relation privilégie la modularité.

Architecture

Diagramme de composants

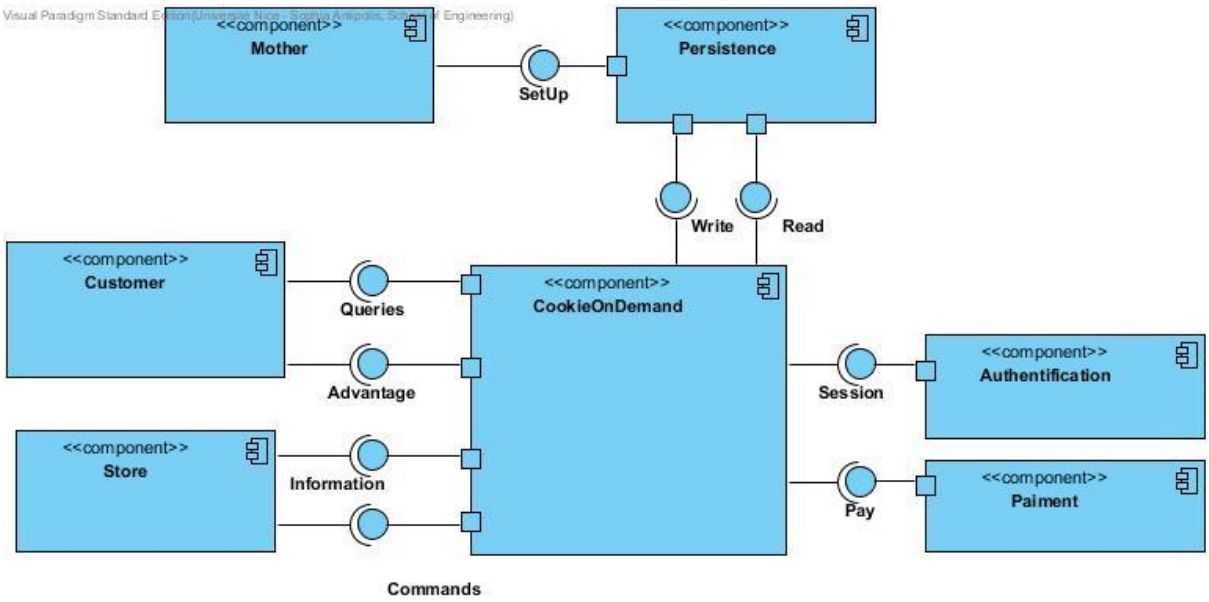

Figure 6 : Diagramme de composants général du CoD

Nous pouvons distinguer dans ce diagramme les différents composants de la plateforme Cookie On Demand.
Le composant Customer propose les actions décrites pour l'utilisateur et l'inscrit dans le diagramme de cas d'utilisation.

Le composant Store propose les actions décrites pour le responsable et l'employé.

Le composant Mother propose l'action décrite pour le créateur.

Le composant TheCookieFactory représente le programme qui va effectuer la partie métier de la plateforme. Il sera décrit plus en détail dans ce document.

Le composant Persistence permet le stockage des données de la plateforme et contient les accès à la base de données.

Les composants Authentification et Payment s'interfacent avec les composants préexistants de l'entreprise pour permettre l'interopérabilité.

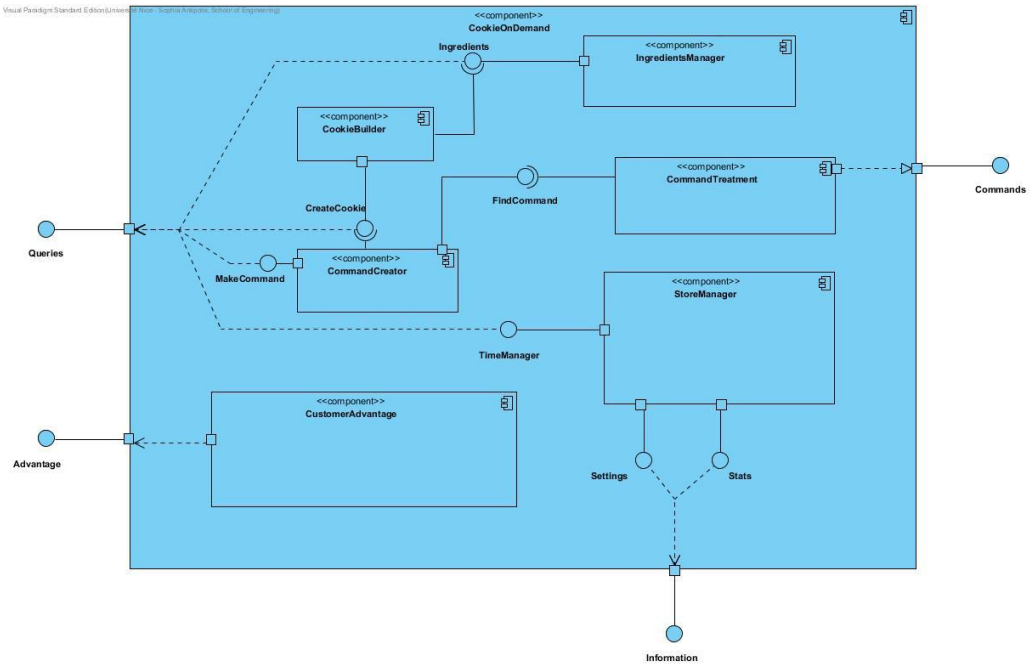

Figure 7 : Diagramme de composant détaillé du composant CoD

Le composant CookieOnDemand est ici redécoupé en sous-composants. C'est sans doute celui dont l'architecture évoluera potentiellement le plus durant le projet.

IngredientManager peut être amené à disparaître pour se répartir dans les autres composants.

Tous les composants découpent les fonctionnalités afin de les rendre plus modulables.

Interfaces

Les interfaces des diagrammes sont décrites ci-dessous :

Queries

MakeCommand

List<Store> getStores()
Command createCommand(Map<Recipe, Integer> m, Store st)
void command(Command c, Slot sl)

CreateCookie

Recipe createCookie(String dough, String flavour, String topping, boolean mixed, boolean crunchy)

Ingredients

List<String> getDoughs()
List<String> getFlavours()
List<String> getToppings()
void suggestDough(String d)
void suggestFlavour(String f)
void suggestTopping(String t)

TimeManager

List<Slot> slotList(Command c, Calendar d)

Advantage

void subscribe(User u)
void makeGift(User giver, User recever, Integer amount)

Information

Settings

setHours(List<Calendar> starts, List<Calendar> ends)
setToday(Recipe : today)

Stats

List<PickUpStats> getStats()

Commands

Store getStore(int id)
List<Command> getCommands(Store s)

SetUp

List<Ingredients> notAccepted()
acceptIngredients(Ingredient i, double price)
rejectIngredients(Ingredient i)

Read

Store findById(int)
Recipe findById(int)
Command findById(int)
Store findByStore(int)
Person findById(int)
List<Ingredient> findByType(IngredientType)
List<Ingredient> getUnvalidated()

Write

insert(Store s)
update(Store s)
delete(Store s)
insert(Receipe r)
update(Receipe r)
delete(Receipe r)
insert(Command c)
update(Command c)
delete(Command c)
insert(Person p)
update(Person p)
delete(Person p)
Ingredient add(IngredientType t, double prix)
addFavoutite(Receipe r, Registered rd)
addFavoutite(Store s, Registered rd)

Diagramme de déploiement

Voici le diagramme de déploiement de notre architecture.

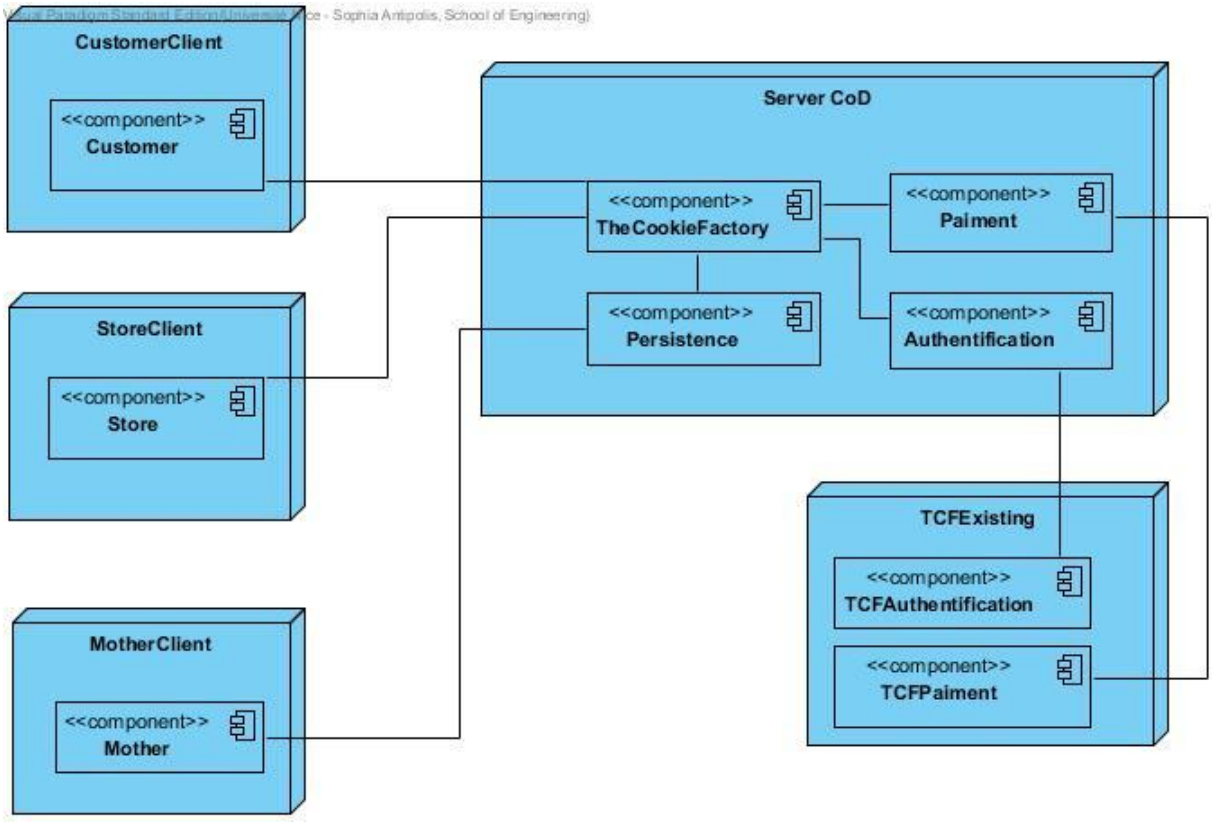

Figure 8 : Diagramme de déploiement du CoD

Nous pouvons distinguer dans ce diagramme de déploiement 5 entités qui viennent compléter le premier schéma (Figure1) et le diagramme de composants.

La première entité « CustomerClient » correspond aux machines depuis lesquelles les simples utilisateurs et les utilisateurs inscrits vont se connecter pour commander leurs cookies.

L'entité « StoreClient » correspond aux machines installées dans les différentes boutiques pour gérer les commandes et les paramètres de la boutique.

L'entité MotherClient est réservée à la maison mère qui gère les recettes.

Ces trois entités seront « clientes » du server « ServerCoD ». Le ServerCoD est installé sur une machine et comprend le serveur qui va effectuer les services, la base de données ainsi que les interfaces d'interactions avec les systèmes .Net préexistants de l'entreprise.

Enfin l'entité TCFExisting représente les deux systèmes .Net préexistants et sont déjà déployés dans l'entreprise.

Cette architecture de déploiement n'est pas prévue pour être opérationnel en cas d'une panne sur le serveur CoD. Il privilégie plutôt la simplicité de déploiement et est moins coûteux en nombre de serveurs à déployer.