Rapport final conception logicielle

CookieFactory-22-23-L

Groupe

Team-L

Étudiants

BAILET Ludovic, CHABANIER Aurélia, DUBOIS Quentin, HERRMANN Matis, LATAPIE Florian

17 décembre 2022

Université Côte d'Azur - Polytech Nice – SI4-Conception Logicielle

2022 – 2023

Table des matières

Introduction

Le projet *The Cookie Factory* est un projet ayant pour but de créer un service de commande en ligne pour une chaîne de magasins de cookies, *The Cookie Factory* (TCF). Ce service porte le nom de *Cookie on Demand* (CoD). Un client de la chaine de magasin peut constituer un panier en ligne afin de commander deux types de cookies : des cookies classiques et des Party Cookies.

Une fois le panier rempli et le créneau choisi, le client paye sa commande en ligne. La commande sera ensuite assignée à un cuisinier du magasin. Lorsque celle-ci sera préparée, le client en sera informé par le système via courriel et SMS. Si le client ne vient pas récupérer sa commande, alors elle sera considérée comme oubliée. Pour éviter le gaspillage, notre système crée des paniers « surprise » composés des cookies de ces commandes oubliées. Afin de les écouler le plus efficacement possible, notre application utilise le service externe *Too Good To Go*. Les clients peuvent ainsi récupérer un ensemble de cookies aléatoires moins cher grâce au système *Too Good To Go*, dépendant des commandes oubliées.

Diagrammes UML

Diagramme de cas d'utilisations

SVG en ligne

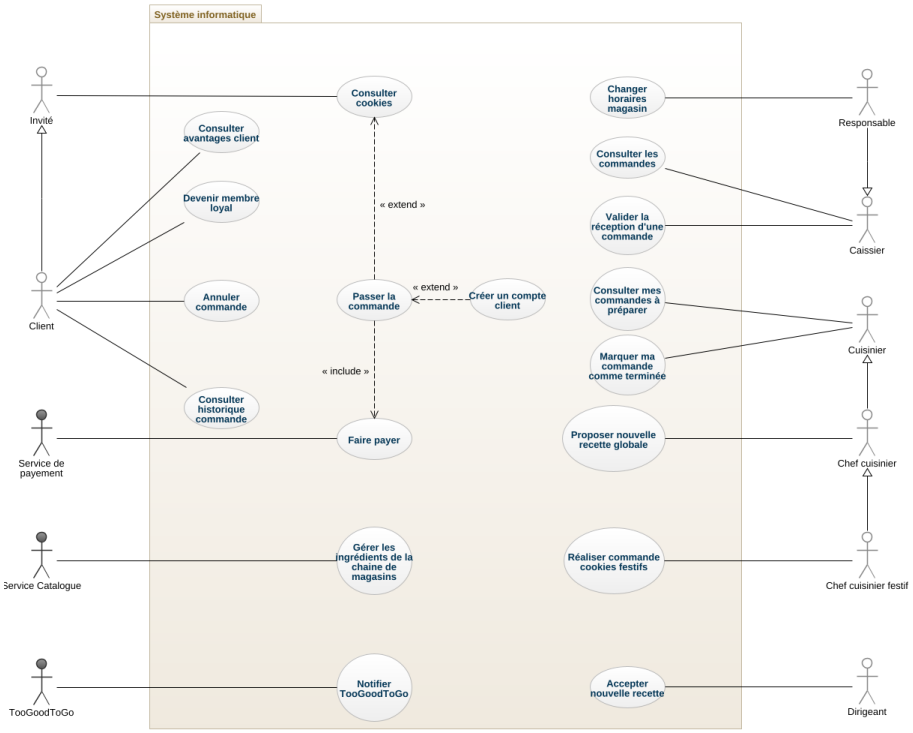


Diagramme de classes

SVG en ligne (nous vous conseillons d'ouvrir ce fichier depuis le web)

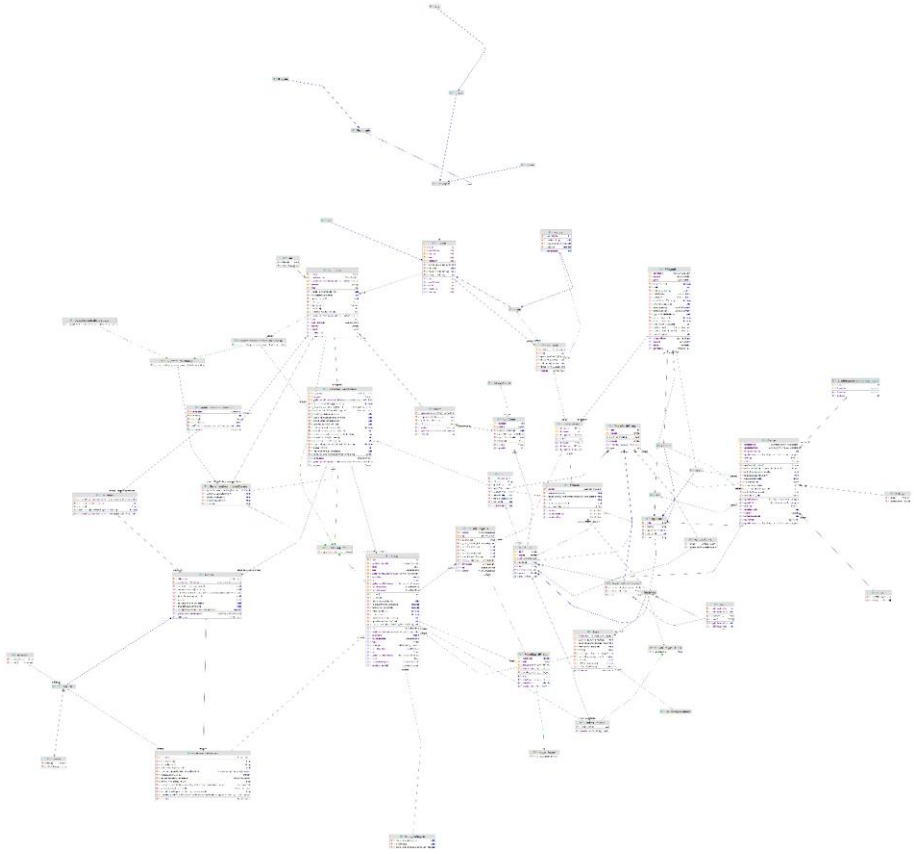
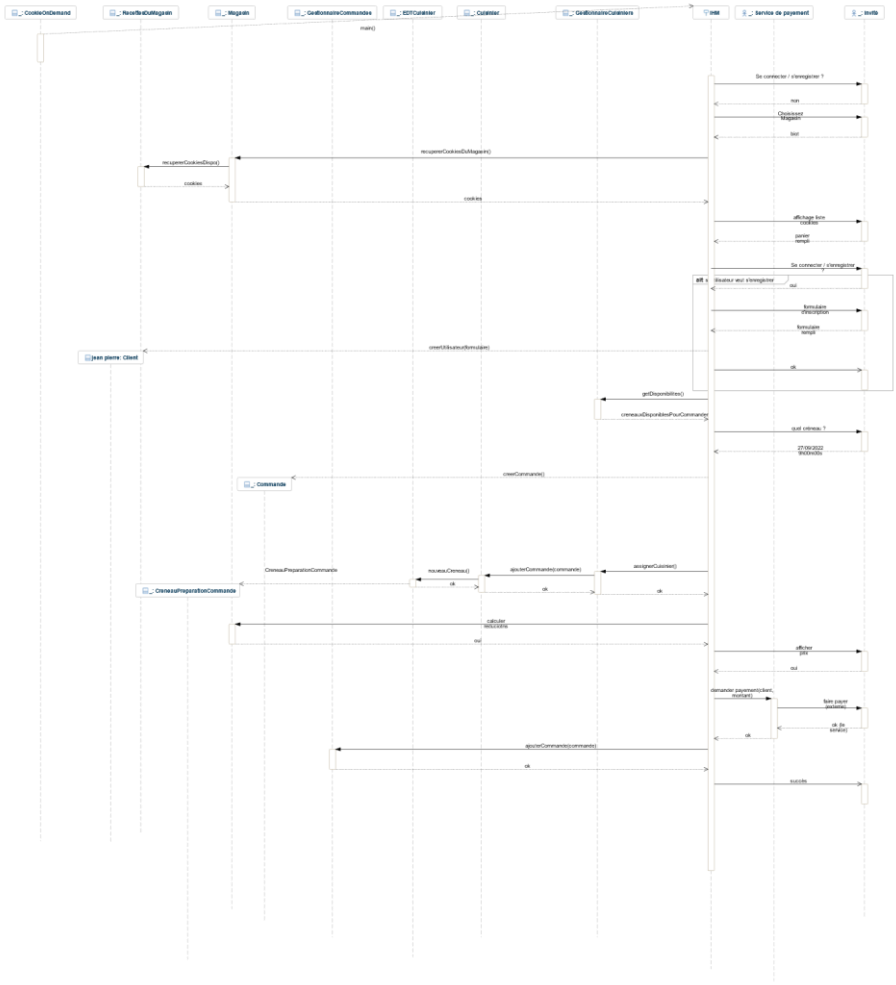


Diagramme de séquence

Passer une commande

SVG en ligne



Patrons de Conception

Too Good To Go

L'extension *Too Good To Go* consiste à créer des paniers « surprise » basés sur les commandes oubliées. Jeter alors que la nourriture est encore consommable est un véritable gâchis, surtout les délicieux cookies de notre Cookie Factory.

Tout d'abord, nous sommes partis de l'hypothèse que le client était notifié par le service externe *Too Good To Go* des paniers « surprise » qu'il pourrait récupérer. Nous avons donc traité le service externe comme un service de courriel ou de SMS.

Pour réaliser cette fonctionnalité, nous avons utilisé le patron de conception *Singleton*. Nous avons fait ce choix, car le service de message, Message Service, était déjà implémenté de cette manière. Néanmoins, nous pensons que ce choix est judicieux puisque nous avons besoin d'une seule instance afin de s'adresser aux services externes. Il a alors un comportement similaire à ce qu'un composant Spring pourrait fournir comme fonctionnalité. De plus, l'utilisation du patron de conception singleton permet de réduire le couplage en accédant directement à la classe de manière statique.

Party Cookies

L'extension Party Cookies offre la possibilité aux clients de constituer un panier avec des cookies festifs, qui sont des cookies de grandes tailles possédant des thèmes spécifiques. Ces cookies sont confectionnés par des chefs cuisiniers spéciaux ayant les compétences pour réaliser cette tâche.

Commenté [FL1]: Pas sur du terme

Pour réaliser cette extension, nous avons dû créer les nouveaux types de cookies en utilisant l'héritage. Pour ce faire, nous avons simplement utilisé l'héritage. Un cookie festif étant un cookie classique ayant un certain nombre d'attributs et de méthodes identiques, nous n'avons alors pas utilisé de patron de conception en particulier.

Commenté [FL2]: Pabo

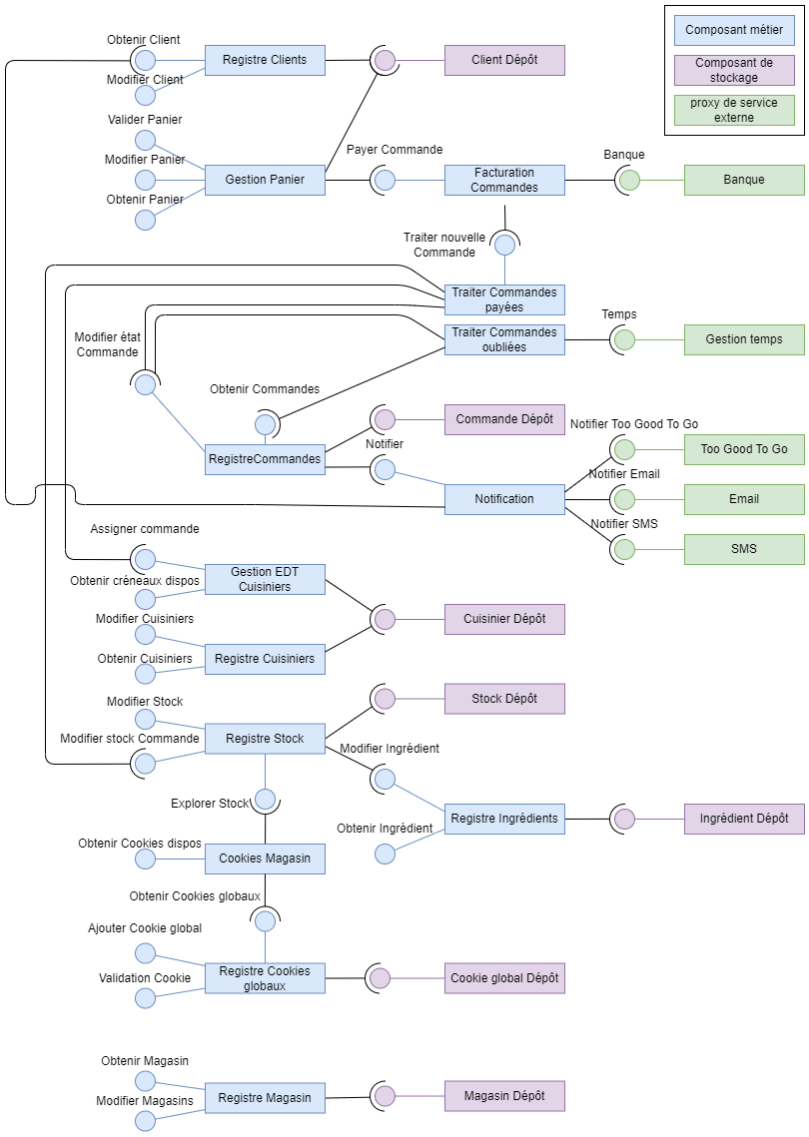
Le seul patron de conception qui aurait correspondu serait le patron *Décorateur* qui permet d'ajouter de nouvelles fonctionnalités de manière dynamique à un objet. Cela permet notamment d'éviter une explosion du nombre de classes, dans le cas où de nombreux objets similaires possèdent différents comportements.

Dans notre cas, nous n'avons que deux objets semblables qui existent au sein de notre application, Cookie Festif et Cookie. Ainsi, le patron *Décorateur* a peu d'intérêt pour nous. C'est la raison pour laquelle nous avons décidé de ne pas l'utiliser.

Nous avons néanmoins utilisé le design patron *Factory Method* dans le but de simplifier la création des Cookies et des Cookies Festifs. Cela permet également de réduire le couplage de nos classes à Cookie et Cookie Festif, puisque la création des objets est déléguée à nos Factory.

Migration vers Spring

Tout d'abord, notre architecture initiale contenait déjà des classes gestionnaires qui avaient déjà le rôle de composants métier. Nous les avons donc conservés, mais les avons renommés, car leur appellation manquait de signification ou leur nommage ne convenait pas aux conventions présentées en cours. L'objectif a été de créer une nouvelle architecture orientée composants qui se rapprochait de notre architecture déjà existante.



Nous allons maintenant rentrer plus en détail sur la responsabilité respective de chaque composant. Pour commencer, les composants Registre Clients, *Registre Magasins*, *Registre Cuisiniers*, *Registre Ingrédients*, *Registre Cookies Globa*ux ainsi que de *Registre Commande* possèdent deux interfaces permettant respectivement d'obtenir et de modifier la ressource. Nous avons ici une notion de CRUD : ces composants créent, lisent, mettent à jour et suppriment ou au moins l'un des quatre.

Commenté [FL3]: Pabo c'est du parlé ca

Commenté [FL4]: Ajout

Ensuite, le composant *Gestion Panier* ressemble aux composants précédents. Il permet de gérer le panier d'un client. Il possède deux interfaces, l'une pour modifier le panier et l'autre pour valider le panier, une fois celui-ci complété.

Le composant *Facturation Commande* possède la responsabilité d'encaisser la commande d'un client. Il prend un panier et renvoyer un objet commande une fois, le paiement validé auprès du service externe *Banque*.

Le composant *Traiter Commandes Payées* offre une seule interface *Traiter nouvelle commande*, permettant une fois la commande créée de retirer les ingrédients du stock et d'assigner un cuisinier. Dans le cas où aucun cuisinier n'est disponible ou que les ingrédients du stock sont insuffisants, une erreur survient et la commande est annulée. Ce cas pourrait arriver lors du traitement de plusieurs commandes en parallèles.

Commenté [FL5]: Pourrait au lieu de peu

Le composant *Traiter Commandes oubliées* permet de modifier le statut des commandes qui n'ont pas été récupérées au bout de deux heures. Il ne possède pas d'interface, car c'est un composant indépendant. Il ne traite que, n'utilise que les informations des commandes en fonction et le service externe de gestion du temps. Il est utilisé par aucuns autres composants. Pour la gestion du temps, il utilise le service externe *Gestion temps*.

Ensuite, le composant *Registre Stock* permet de gérer les stocks du magasin via l'interface *Modifier Stock*. Il permet également de diminuer les quantités des ingrédients via l'interface *Gérer Stock commande*, utilisée lors de la validation d'une commande ou de son annulation.

Commenté [FL6]: Phrase modifiée

Le composant *Gestion EDT Cuisiniers* permet de gérer le planning d'un cuisinier. Il possède deux interfaces. La première *Assigner commande* permet d'assigner une commande à un certain créneau à un des cuisiniers disponibles. La deuxième interface *Obtenir créneaux disponibles* permet de récupérer tous les créneaux disponibles pour les afficher au client via l'interface web.

Le composant *Message Service* permet de notifier le client par SMS et courriel lors du changement d'état de sa commande. Il permet aussi de contacter le service externe *Too Good To Go* pour lui transmettre les paniers « surprise » formés à partir des commandes oubliées.

Finalement, le composant *Cookie Magasin* permet de récupérer les cookies disponibles dans un certain magasin via l'interface *Consulter cookies disponibles*. Le calcul des cookies disponibles est dynamique.

Concernant le code, nous avons migré qu'une petite partie de notre application vers Spring. Nous avons ainsi migré un *MVP*, permettant à un client de créer un compte, de passer une commande et de la réceptionner en magasin. Le but étant d'avoir une application de bout en bout relativement simple afin de faciliter la migration. Nous avons aussi migré les tests concernant ces parties.

En conclusion, nous avons rempli les demandes de l'application tout en utilisant des composants Spring. Nous pensons tout de même que notre architecture bien qu'aboutie présente quelques faiblesses. Nous ne maîtrisons pas encore parfaitement l'architecture en composants. Nous avons notamment des problèmes à discerner les différentes connexions entre les composants et de savoir si ces connexions sont légitimes au non (par exemple doit on passer une référence ou rechercher l'objet via son identifiant ou encore des composants trop couplés). Néanmoins, la partie sur la migration à Spring, nous aura permis à tous de découvrir le Framework Spring et l'architecture en composants. Cette partie a été très instructive. Cela nous inspire a l'utiliser dans de prochains projets, aborder un gros projet via ce type d'architecture nous montre de grandes forces au niveau de la mise à l'échelle d'une application.

Commenté [FL7]: Modif ici

Commenté [FL8]: Ajout

Commenté [FL9]: Ajout

Auto-évaluation

Cette partie concerne l'attribution des points et de la répartition des tâches pour chaque membre de l'équipe.

Quentin Dubois – 115 points

Durant la première partie du projet, j'ai travaillé activement sur les diagrammes de cas d'utilisation, de classes et de séquences. Essayant au maximum de créer un ensemble cohérent et harmonieux.

Commenté [FL10]: Ptdrr feng shui aussi tant qu'on y est

Ensuite, durant la deuxième partie du projet concernant le code, j'ai réalisé les tâches suivantes :

- Passer une commande par un client
- Bénéficier d'une réduction sur une commande
- Gérer les ingrédients en stock
- Notifier le client lorsque l'état de sa commande change
- Gérer les ingrédients
- Gérer les magasins
- Ajout des cookies festifs (Extension : Party Cookies)

Finalement, sur la partie Spring du projet, j'ai beaucoup travaillé sur l'architecture en composants et donc sur le diagramme de composants. J'ai également participé à la migration du code sur Spring. Notamment les tests.

Florian Latapie – 115 points :

Pendant la première partie, j'ai travaillé sur les différents diagrammes. Principalement sur le diagramme de classes et use-case. Je me suis aussi occupé de la mise en place de l'environnement de développement (pom.xml).

Voici la liste des taches que j'ai effectué au cours de la partie code orienté objet du projet :

- Marquer la commande en entrée et sortie de la cuisine
- Changer les horaires du magasin
- Une commande par cuisinier, plusieurs cuisiniers puis Plusieurs commandes par cuisinier, plusieurs cuisiniers : c'est ici que j'ai développé un algorithme pour vérifier comment assigner les cuisiniers en fonction des commandes et de l'emploi du temps

- Gestion des classes « temps réel » : principalement GestionnaireDeCommandesOubliees
- Corriger les erreurs (fautes de français, reformat code …)
- Readme, documents de rendus (numérisation de tous les diagrammes, mise en forme du rapport)

Avec Quentin nous nous sommes occupés de la migration du projet vers Java SpringBoot. La plupart de notre travail s'est axée sur le diagramme de composants puis nous avons codé une version MVP. J'en ai profité pour modifier le code que je trouvais de plus faible qualité en plus haute qualité.

Matis Herrmann – 100 points

- Readme
- Programme de fidélité : Gestion des points
- Les acteurs (Clients et Employées)
- Payement d'une commande
- La première version des recettes et des ingrédients
- [US] Obtenir une réduction quand le client attend le quota de fidélité
- [US] Payer lors d'une commande
- [US] Souscription au programme de fidélité
- [US] Créer un compte client
- [US] Consulter les commandes en attente de retrait

Ludovic Bailet – 85 points

Pendant la première partie du projet, j'ai participé à la création de cas d'utilisation et a certain diagramme de séquences. J'apportais aussi mes idées sur le diagramme de classe ou quand d'autres membres voulaient avoir l'avis du groupe.

Lors de la deuxième phase du projet, je me suis occupé des taches suivantes :

- Consulter les commandes à préparer
- Consulter le catalogue d'un magasin
- Soumettre et accepter des nouvelles recettes

J'ai également réalisé quelques refactor du code concernant :

- Les Cookies et les Recettes en implémentant un Builder de Recette
- Les recettes de la marque et des magasins en implémentant un Pattern Observer
- La responsabilité de la vérification des commandes

Pour finir, j'ai peu participé à la migration vers Spring. En effet, je finalisais quelques refactor. J'ai quand même participé lors de la réflexion pour le choix des composants.

Aurélia Chabanier – 85 points

Pendant la première partie du projet, j'ai participé à la création de cas d'utilisation et j'ai apporté mes idées pour les autres diagrammes.

Concernant les fonctionnalités réalisées :

- Valider une commande délivrée
- Refactor du code afin que seul le gestionnaire de commande soit responsable de la vie d'une commande
- Annuler sa commande
- Vendre ses cookies TTC (ajout de la taxe sur les prix)

Enfin, après l'ajout des nouvelles fonctionnalités, j'ai effectué les tâches suivantes :

- Commander des Party Cookies
- Ajout de la quantité dans les recettes
- Modification de la quantité des recettes par les Party Cookies
- Devenir Chef Cookie Festif
- Assigner les Chefs Cookie Festif à une commande Party Cookies

Pour ce qui est de la partie Spring, je n'ai pas vraiment contribué. J'ai continué de développer sur la branche main, ce qui a permis de terminer plus d'issues que ce que nous avions prévu lorsqu'on a commencé la migration vers Spring.

Commenté [FL11]: Je viens d'ajouter ca

Conclusion

Nous sommes globalement fiers de nous. Nous avons réussi à réaliser la majorité des fonctionnalités attendues par les utilisateurs. Évidemment, nous n'avons pas réalisé toutes les fonctionnalités attendues. Néanmoins, nous nous sommes focalisés sur les plus importantes, en gardant à l'esprit d'avoir toujours un système fonctionnel et cohérent.

Ce projet a été fort instructif : il nous a permis de découvrir, apprendre et maîtriser de nouveaux principes et technologies. Tout d'abord, nous avons découvert le Behavior-Driven Development (BDD) avec l'utilisation des technologies Gherkin et Cucumber. Afin de nous focaliser sur les fonctionnalités essentielles de notre application, nous avons utilisé des histoires utilisateurs (user-story). Nous avons aussi découvert les principes d'une architecture modulaire et avons pu mettre en pratique nos connaissances dans notre projet. Enfin, nous avons utilisé le Framework Spring pour migrer une partie de notre application, un MVP d'une architecture orientée objets vers une architecture orientée composants. Cela n'a pas été une mince affaire, mais nous avons réussi.