

GUIGON Dorian	<i>PO</i>
HEILMANN Hugo	<i>DevOps</i>
HESCHUNG Erwan	<i>DevOps</i>
MAGNIN Mathis	SA
ROQUES Maxence	QA

Rapport backend

Team Q

I Périmètre fonctionnel	3
I.I Hypothèses de travail	3
I.II Les limites identifiées	3
II Conception UML	4
II.I Glossaire :	4
II.II Diagramme de cas d'utilisation	5
II.III Diagramme de classe	5
II.IV Design Patterns	6
II.IV.I Façades	6
II.IV.II Stratégies	6
II.IV.III Singleton	7
II.IV.IV Joshua Bloch's Builder	7
II.IV.V Design pattern non implémenté	8
II.V Diagramme de séquence	8
III Maquette	11
IV Qualité des codes et gestion de projets	17
IV.I Tests	17
IV.II Vision qualitative	17
IV.III Gestion du projet	18
V Rétrospective et auto-évaluation	18
V.I Bilan	18
V.II Missions personnelles	18
V.III Auto-évaluation	19

I Périmètre fonctionnel

I.I Hypothèses de travail

- o **Absence de traitement des paiements en interne** : Il est supposé que le traitement des paiements ne sera pas réalisé par les moyens internes du système.
- o **Proximité des restaurants aux points de livraison** : Il est supposé que chaque restaurant se situe à environ 15 minutes des points de livraison. En conséquence, le temps de livraison ne sera pas géré directement par le système, et l'estimation du délai repose sur cette proximité supposée.
- o **Menus à article unique sans condiments** : Il est supposé que chaque menu se compose uniquement d'un seul article principal, sans possibilité d'ajouter des condiments. Cette structure fixe permet une gestion simplifiée des commandes et du suivi des articles.
- o **Application de la réduction en fin de commande** : Il est supposé que toute réduction sera appliquée uniquement à la fin de la commande, une fois le montant total calculé. Cela permet de simplifier la gestion des remises en les traitant globalement, sans affecter le calcul individuel des articles.

I.II Les limites identifiées

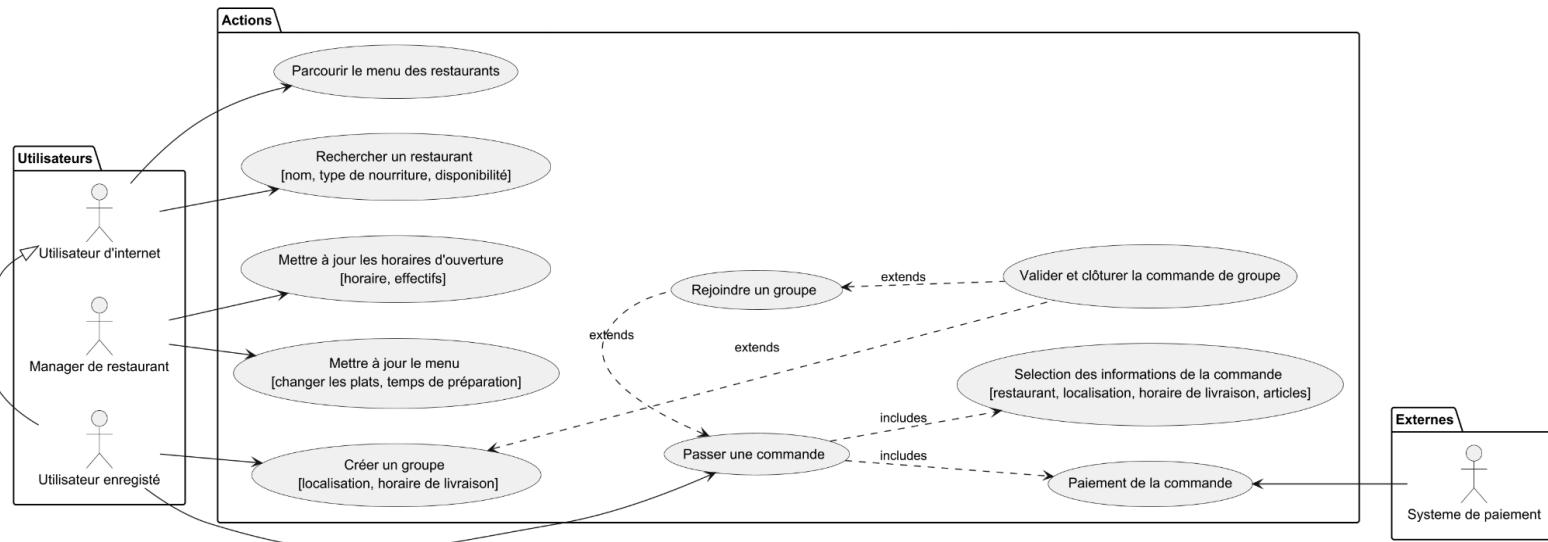
- o **Limitation de la gestion des remises** : Les réductions appliquées en fin de commande ne peuvent pas être combinées avec d'autres promotions ou offres spéciales (la plus importante sera gardée). Le système ne gère qu'une seule réduction par commande.
- o **Absence de personnalisation des articles** : Les menus étant composés d'un seul article sans condiments optionnels, le système ne permet aucune personnalisation d'article.
- o **Aucune gestion de stock** : Le système ne gère pas en temps réel le stock des restaurants. Il est supposé que chaque article est disponible, à condition que le personnel du restaurant soit en mesure de le préparer.

II Conception UML

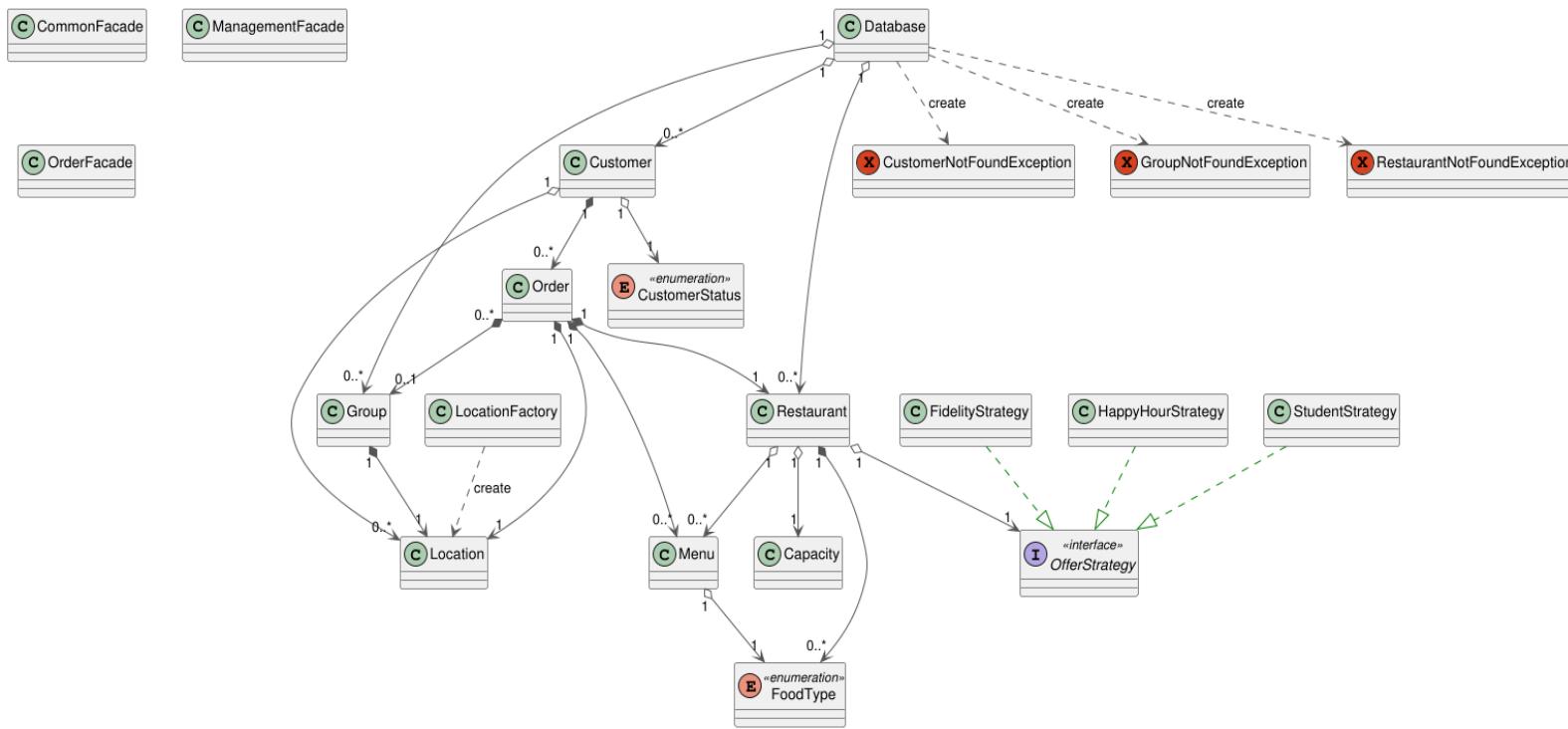
II.I Glossaire :

Manager de restaurant	Personnel responsable de la modification des informations du restaurant sur l'application
Administrateurs du Campus	Représentent les personnes occupant des rôles administratifs et responsables de la supervision et de la gestion des activités du campus. Par exemple, ils peuvent collaborer pour répartir les heures de cours sur le campus afin d'éviter les congestions au Restaurant Universitaire.
Utilisateur d'internet	Fait référence à n'importe quel utilisateur disposant d'internet.
Utilisateur Enregistré	Fait référence à un utilisateur avec un compte enregistré.
Commande de Groupe	Commande constituée de "sous-commandes", diffusées aux personnes concernées grâce à un identifiant. La localisation et l'heure de livraison (pouvant être enregistré ultérieurement) ne sont pas modifiables et aucun paiement n'est impliqué.
Capacité de production	Nombre d'articles disponibles pour un créneau horaire spécifique.
Restaurant	Restaurants disponibles sur l'application.
Localisation pré-enregistrée	Localisations enregistrées en amont de la commande.
Capacité du restaurant	Faisabilité d'une commande dans un temps donné
Remise	Appliquée à partir de 10 articles pour une commande (groupée ou non). Elle s'applique aux commandes et sous commandes.
Créateur d'une commande groupée	Personne responsable de : diffuser l'identifiant de la commande, enregistrer la localisation et l'heure de livraison de manière compatible avec les sous-commandes, et de fermer la commande.
Sous-commande	Commande inclue dans une commande groupée.
Commande individuelle	Commande liée à un utilisateur. Un paiement est impliqué !

II.II Diagramme de cas d'utilisation



II.III Diagramme de classe



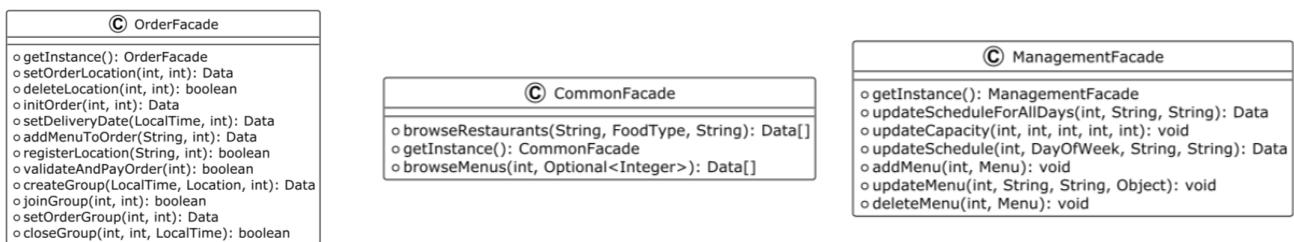
Ce diagramme de classe représente toutes les relations et classes de notre projet. Afin que le diagramme soit lisible, celui-ci ne possède pas les attributs, les méthodes de classe ainsi que les sous-classes.

Ce diagramme peut être divisé en plusieurs parties :

- Les façades permettant d'effectuer les différentes actions.
- La base de données ainsi que les différentes exceptions.
- Les restaurants avec leur stratégie, menu et capacité.
- Les commandes avec la notion de groupe et de customer.

II.IV Design Patterns

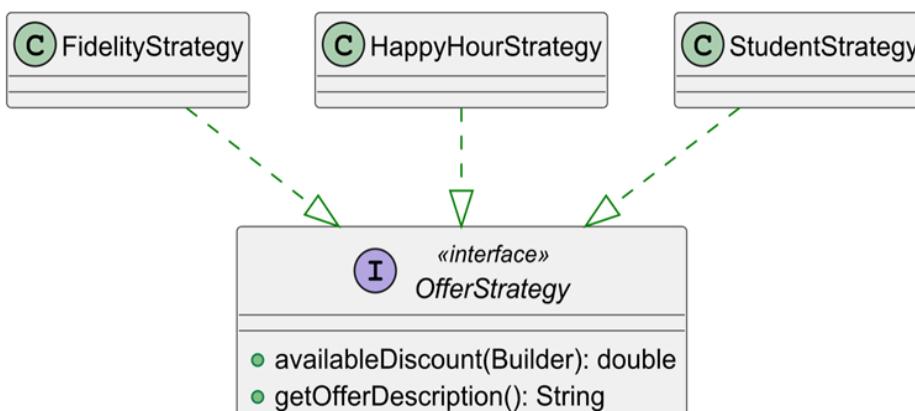
II.IV.I Façades



Chaque façade représente un type d'action, vous retrouverez la *OrderFacade* pour toutes les actions liées au passage d'une commande pour un utilisateur enregistré. La *CommonFacade* permet de réaliser toutes les actions possibles pour un simple utilisateur d'internet. Pour finir, la *ManagementFacade* permet de faire toutes les actions liées au management d'un restaurant par un manager.

Nos façades n'ont pas de statut et sont des singletons (voir [II.IV.III Singleton](#)).

II.IV.II Stratégies



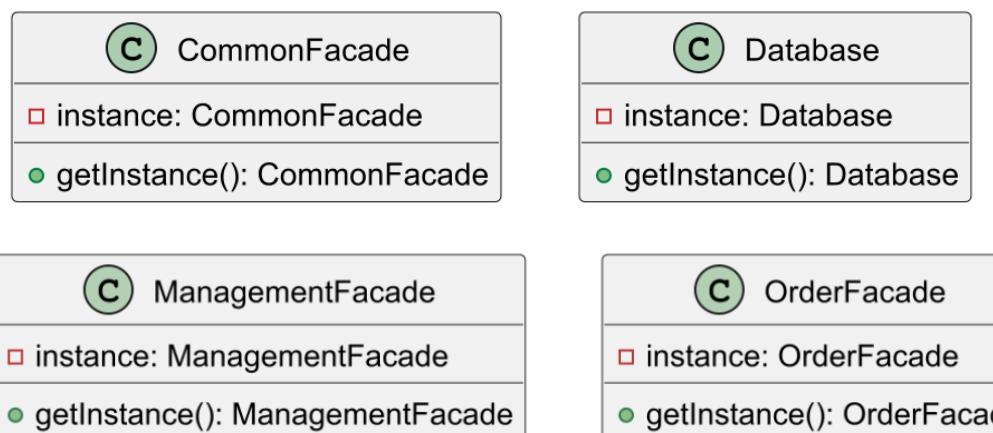
Chaque restaurant peut choisir d'avoir une stratégie. Cette potentielle stratégie sera appliquée à la validation d'une commande dans le restaurant associé.

La *Fidelity* stratégie applique une réduction (20%) si l'utilisateur a déjà trois commandes réalisées dans ce restaurant et n'a pas bénéficié de réductions sur ces 3 commandes.

La *HappyHour* stratégie permet d'appliquer une réduction (25%) entre 17H (inclus) et 19h (exclu).

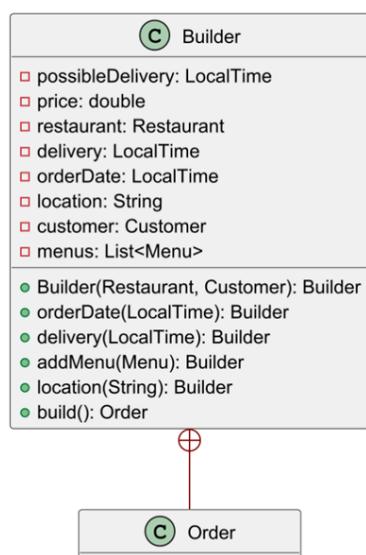
La *Student* stratégie permet d'appliquer une réduction (10%) aux utilisateurs avec le statut étudiant.

II.IV.III Singleton



Ce design pattern nous est utile pour récupérer une instance unique d'un objet. Il est très utile pour la pseudo 'Database' qui regroupe les données nécessaires (Restaurants, Menus, Utilisateurs et Groupes). Les façades peuvent ainsi être facilement récupérées pour les tests.

II.IV.IV Joshua Bloch's Builder



Le builder que nous avons intégré dans notre classe Order permet de construire une Order étape par étape (ajout d'un menu, spécifier la localisation en début de commande, construire la commande ...).

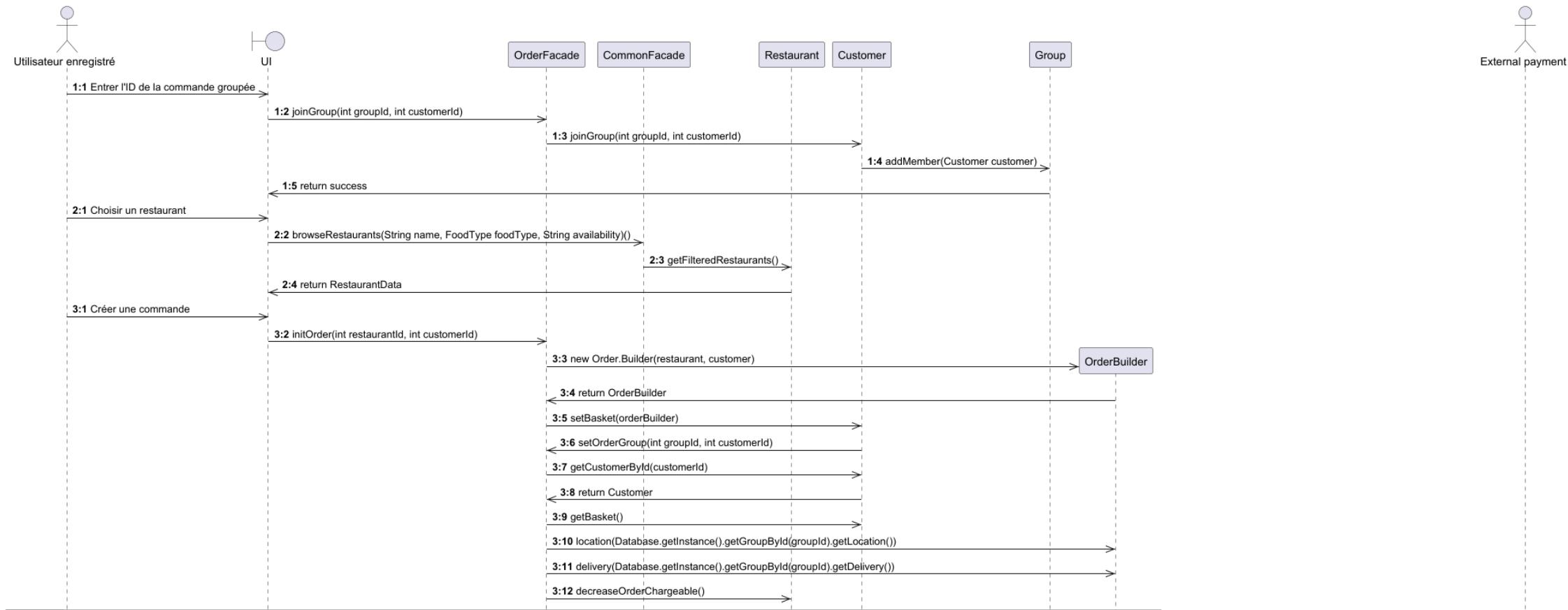
II.IV.V Design pattern non implémenté

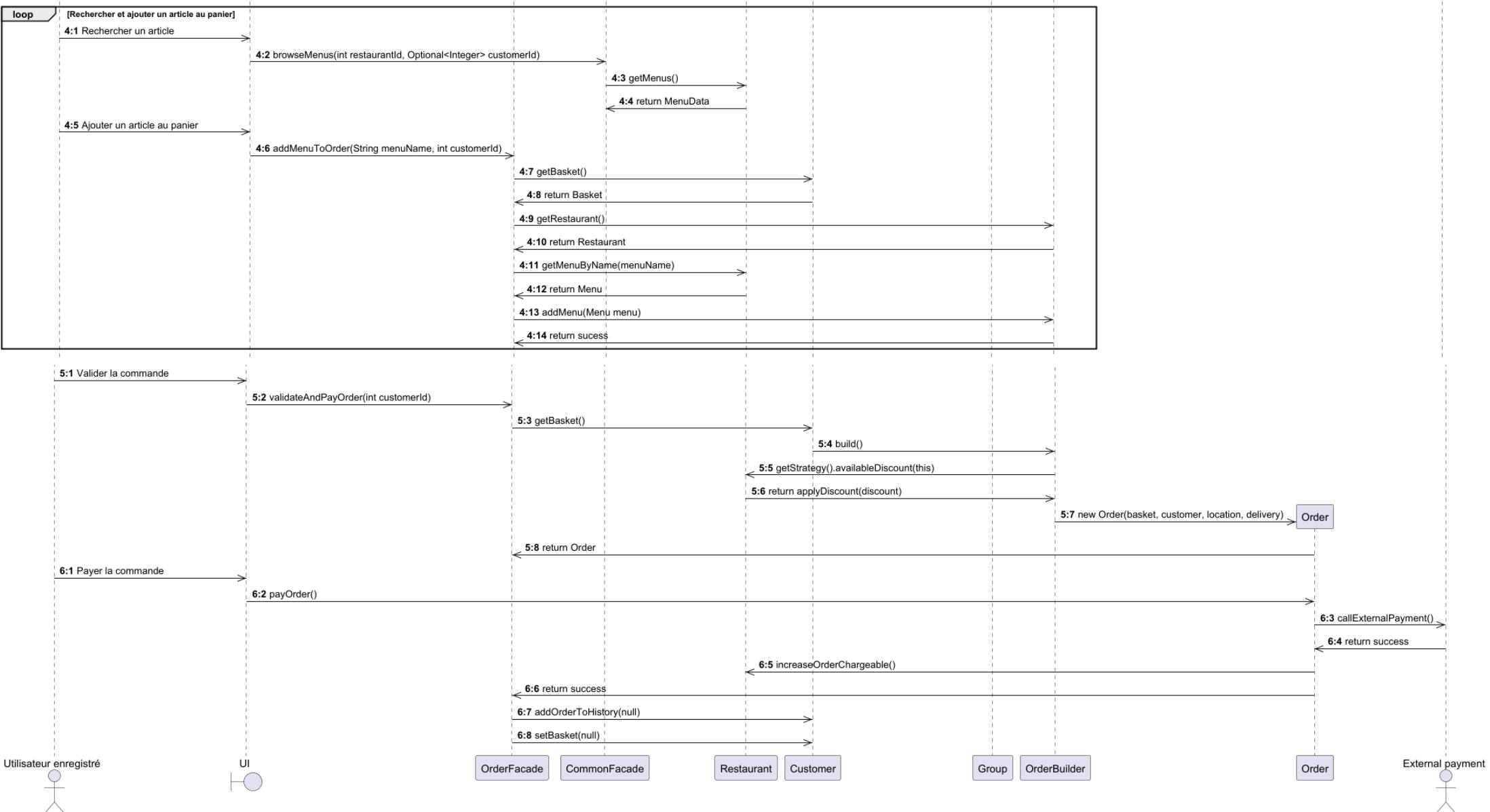
Lors de la conception du système, nous avons envisagé l'intégration de **proxies**, mais nous n'avons pas eu le temps de les mettre en œuvre. L'utilisation de proxies aurait apporté une sécurité renforcée en vérifiant les permissions d'accès avant d'interagir avec les objets réels, protégeant ainsi les services critiques.

II.V Diagramme de séquence

Hypothèses du diagramme de séquence :

- L'utilisateur est déjà connecté
- Le groupe qu'il rejoint est déjà créé





III Maquette

Le Meilleur site

Rechercher un restaurant

Catégories

- List item

Régime alimentaire

- List item

Mega burger
Body text.

Sushii master
Body text.

Tacoland
Body text.

Mexicano
Body text.

HellSI
Body text.

Authentica
Body text.

Do

Do

The screenshot shows a mobile application interface for a food delivery service. At the top left is a user profile icon and a search bar with placeholder text "Rechercher un restaurant". On the right are icons for account settings and a shopping cart. The main content area features a large image of a cheeseburger. Below it, there are two sections: "Catégorie 1" and "Catégorie 2", each containing four items, represented by icons with the text "Item price".

Catégorie 1

Catégorie 2

Item price Item price Item price Item price

Item price Item price Item price Item price

Item price Item price Item price Item price

This screenshot is similar to the one above, showing the same food delivery app interface. However, a modal dialog box has appeared in the center-right area. The dialog has a red close button in the top right corner and contains the text "Entrer le code de groupe". Inside the dialog is a numeric keypad with the number "1" repeated five times, followed by a green "Valider" button.

Catégorie 1

Catégorie 2

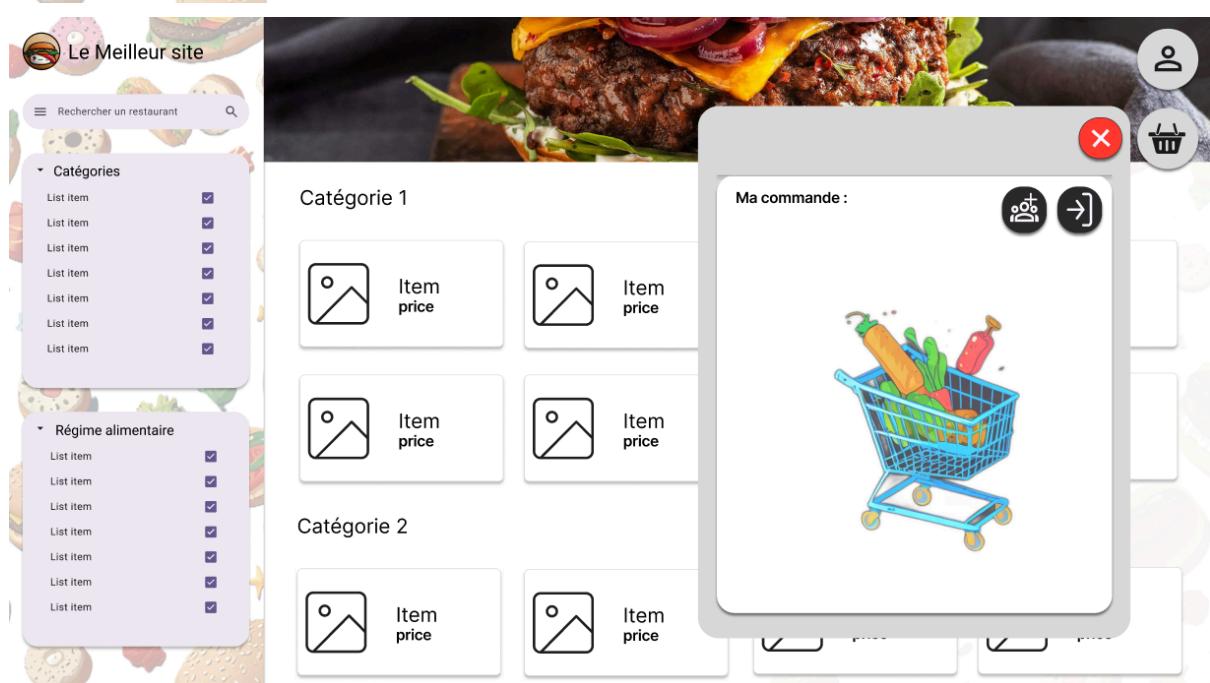
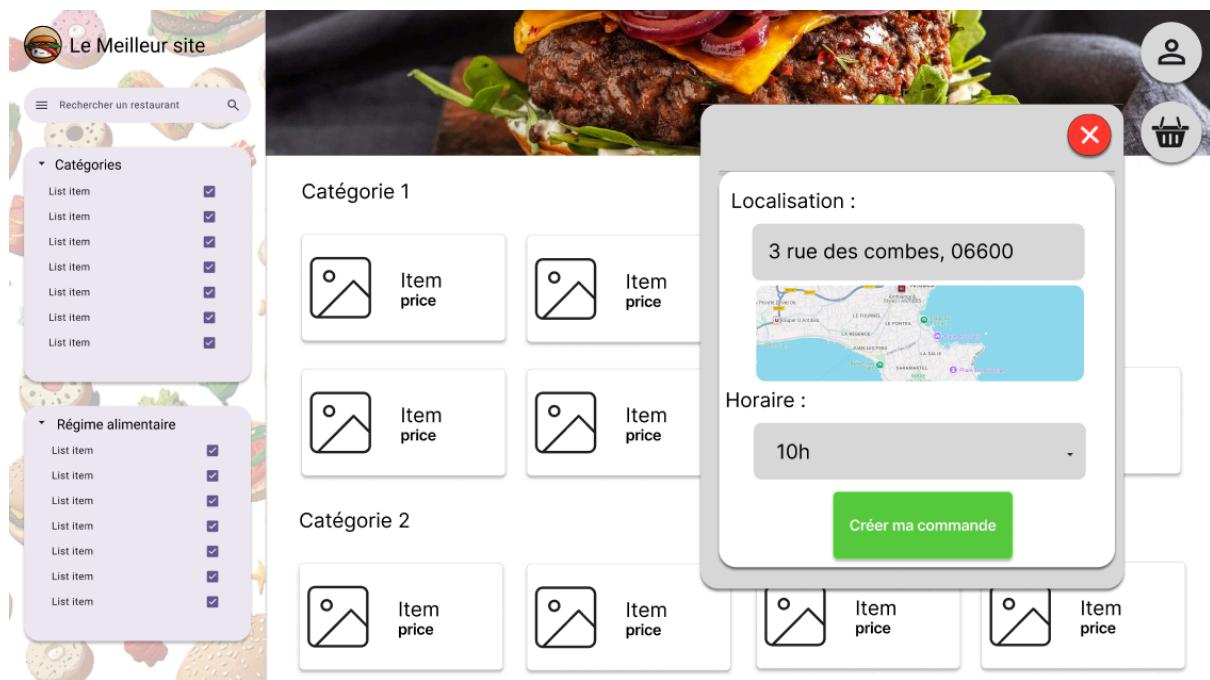
Item price Item price

Item price Item price Item price Item price

Item price Item price Item price Item price

The screenshot shows a mobile application interface for a food delivery service. At the top, there's a header with a logo, a search bar labeled "Rechercher un restaurant", and user icons for profile and cart. Below the header, a large image of a cheeseburger is displayed. To the left of the main content area, there's a sidebar with two expandable sections: "Catégories" and "Régime alimentaire". The "Catégories" section lists "List item" multiple times with checkboxes. The "Régime alimentaire" section also lists "List item" multiple times with checkboxes. The main content area is divided into two categories: "Catégorie 1" and "Catégorie 2", each containing four items, represented by icons and the text "Item price".

This screenshot is similar to the one above, showing the same app interface. However, a modal window has been overlaid on the right side of the screen. The modal contains a shopping cart icon filled with vegetables, a message "Pas de commandes répertoriées", and a green button labeled "Commander dans ce restaurant". The rest of the interface, including the sidebar and the grid of items, remains visible.



The screenshot shows a mobile application interface for a food delivery service. At the top, there is a header with a logo, a search bar labeled "Rechercher un restaurant", and user icons for profile and cart. Below the header, there are two expandable dropdown menus:

- Catégories**:
 - List item
 - List item
- Régime alimentaire**:
 - List item
 - List item

The main content area displays a large image of a cheeseburger. Below the image, there are two sections labeled "Catégorie 1" and "Catégorie 2", each containing four items represented by icons and labels.

Image	Item price
	Item price

This screenshot shows the same food delivery application with a modal window overlaid on the screen. The modal is centered and contains the following elements:

- A placeholder image area with a circular crop mark.
- A "Catégorie" label.
- A "Description" label.
- A green button labeled "Ajouter cette article".

The background of the app shows the same header, search bar, and category menus as the first screenshot. Below the modal, there are four items displayed in a grid.

Image	Item price
	Item price

The screenshot shows a mobile application interface for food ordering. At the top, there's a header with a logo and the text "Le Meilleur site". Below the header is a search bar with the placeholder "Rechercher un restaurant" and a magnifying glass icon. To the right of the search bar are two circular icons: one with a "D" and another with a shopping basket.

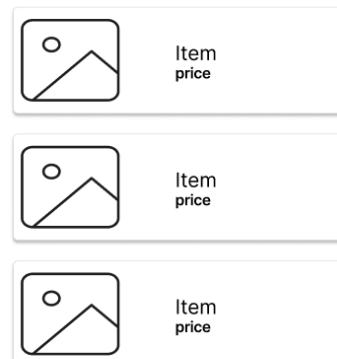
On the left side of the screen, there's a sidebar with two sections:

- Catégories**: A list of items under "List item" with several checkboxes checked.
- Régime alimentaire**: A list of items under "List item" with several checkboxes checked.

The main content area features a large image of a cheeseburger. Below the image, there are two sections of items:

- Catégorie 1**: Contains four items, each represented by a small image icon and the text "Item price".
- Catégorie 2**: Contains four items, each represented by a small image icon and the text "Item price".

A modal window is open in the center-right, containing the message "Vous êtes dans le groupe de X:" followed by two items from Catégorie 1. A green button at the bottom right of the modal says "Valider la commande".



total price : X

Payer

IV Qualité des codes et gestion de projets

IV.I Tests

En ce qui concerne les tests, nous avons effectué des tests unitaires avec JUnit et des tests d'acceptation avec Cucumber.

Les tests unitaires réalisés avec JUnit garantissent que chaque classe fonctionne correctement et assurent une qualité élevée grâce à des assertions précises.

Nous avons également développé 62 scénarios d'acceptation en Cucumber, validant le comportement global de l'application selon les exigences fonctionnelles.

La réalisation de ces tests nous ont apporté une couverture globale de 83%, ce qui assure une bonne fiabilité et confiance dans ce projet que nous avons développé.

Coverage java in demo			
Element ^	Class, %	Method, %	Line, %
> sophiaeats	88% (24/27)	83% (163/195)	86% (504/580)

IV.II Vision qualitative

Dans l'ensemble, le code reste plutôt lisible et clair notamment grâce aux commentaires explicatifs présents au-dessus de la définition des fonctions. Cependant, certaines parties pourraient être mieux gérées. Par exemple, dans la classe Capacity, chaque élément du tableau (représentant la capacité du restaurant pour un jour et une demi-heure donnés) est représenté par son jour et la demi-heure de la journée qu'il représente. Cependant, cela revient à devoir compter l'heure correspondante à chaque fois que l'on relit le code, ou à devoir mettre des commentaires explicatifs en plein milieu du code. Une bonne manière de faire aurait été de créer un dictionnaire statique associant pour un horaire donné, le numéro de demi-heure correspondante, et d'utiliser des horaires dans le code.

```
for (int halfHour = 0; halfHour < 48; halfHour++) {
    if ((halfHour >= 22 && halfHour <= 34) || (halfHour >= 40 && halfHour <= 46)) {
        // Heures de pointe : 11h-14h et 18h-21h
        // 5 à 10 membres du personnel => 300 à 600 secondes de préparation possible
        staff[day][halfHour] = (5 + (int) (Math.random() * 6)) * PREPARATION_TIME_FOR_STAFF_MEMBER;
    } else if ((halfHour >= 16 && halfHour <= 20) || (halfHour >= 36 && halfHour <= 38)) {
        // Heures intermédiaires : 8h-10h et 15h-17h
        // 2 à 5 membres du personnel => 120 à 300 secondes de préparation possible
        staff[day][halfHour] = (2 + (int) (Math.random() * 4)) * PREPARATION_TIME_FOR_STAFF_MEMBER;
    } else {
        // Heures creuses
        // 1 à 2 membres du personnel => 60 à 120 secondes de préparation possible
        staff[day][halfHour] = (1 + (int) (Math.random() * 2)) * PREPARATION_TIME_FOR_STAFF_MEMBER;
    }
}
```

IV.III Gestion du projet

En ce qui concerne l'automatisation et la gestion des branches, nous avons imposé des règles sur github afin de nous épargner certains éléments clés en ce qui concerne la gestion.

Nous avons imposé de devoir faire des pull request pour chaque nouvelle implémentation en interdisant de push directement sur les branches principales (main et develop).

Et lors de l'ouverture d'une pull request, l'intégralité des tests unitaires et Cucumber sont automatiquement lancés, refusant le merge de la pull request si l'un d'eux échoue.

V Rétrospective et auto-évaluation

V.I Bilan

Dans l'ensemble, le bilan est satisfaisant. Nous avons implémenté la quasi-totalité des fonctionnalités attendues et la majorité de notre code nous semble propre, l'utilisation de nombreux design patterns participant grandement à ce résultat. Les tests cucumber ont été correctement implémentés et utilisés, couvrant une grande partie du code. Nous sommes aussi fiers de notre utilisation de github et de ses outils. D'autre part, chaque membre de l'équipe a participé activement à la conception du projet, et a rempli sa mission vis-à-vis du rôle qu'il a choisi.

En revanche, l'organisation a parfois laissé à désirer, notamment au niveau des délais à respecter et des user stories à définir. Cela nous a rappelé l'importance de produire des fonctionnalités concrètes et testables afin de ne pas stagner, mais au contraire faire évoluer constamment le produit.

V.II Missions personnelles

PO

GUIGON Dorian

- Écriture des scénarios et user stories pour guider l'équipe et s'assurer que les fonctionnalités requises sont implémentées.
- Accompagnement de l'équipe dans la prise de décisions pour garantir la cohérence et la qualité du projet.
- Vérification de l'exhaustivité des documents et respect des délais de rendu pour une gestion rigoureuse des livrables.

Ops

HEILMANN Hugo

- Labellisation des issues avec la méthode MoSCoW afin de mieux estimer la valeur cliente de chacune d'elles.
- Suivi rigoureux des pull request pour assurer l'intégration progressive des tests.

HESCHUNG Erwan

- Mise en place des branches sur le repository et les sécurités associées (push protection sur les branches principales develop et main)
- Mise en place du Kanban
- Mise en place du WebHook discord pour traquer les pull request (un message est envoyé pour chaque pull request ouverte et fermée)
- Mise en place des critères de pull request (au moins une review et passage des tests)

SA

MAGNIN Mathis

- Accompagnement de l'équipe dans la considération des différents patrons de conception.
- Mise en place de la façade et des différents singltons.
- Mise en place du diagramme de classe
- Vérification de la structuration des codes

QA

ROQUES Maxence

- Mise en place de tests unitaires pour s'assurer du bon fonctionnement des fonctions les plus complexes du projet avant leur utilisation dans des scénarios Cucumber.
- Mise en place de tests end-to-end pour vérifier le bon fonctionnement du projet face à une utilisation normale par un utilisateur.
- Mise en place de tests pour les exceptions afin de vérifier le bon fonctionnement de la gestion des erreurs.

V.III Auto-évaluation

Selon nous, tous les membres de l'équipe ont contribué de manière égale au projet, mais nous considérons tout de même qu'Erwan a fourni un effort supplémentaire ! Nous avons chacun mis autant d'efforts dans les tâches assignées, ce qui nous amène à une distribution avoisinant les 100 points pour chaque membre, en reconnaissant l'apport précieux de chacun.

Voici un tableau récapitulatif représentant le ressentie de chacun par rapport à l'implication de chaque membre dans ce projet :

Évalué → Evaluateur ↓	GUIGON Dorian	HEILMANN Hugo	HESCHUNG Erwan	MAGNIN Mathis	ROQUES Maxence
GUIGON Dorian	90	90	100	90	90
HEILMANN Hugo	90	90	100	95	90
HESCHUNG Erwan	90	90	100	85	90
MAGNIN Mathis	85	90	100	80	85
ROQUES Maxence	90	95	100	90	85
Moyenne	89	91	100	88	88