# One Minute

APPLICATION POUR LA RESTORATION
DOCUMENT POUR LA MOA

## AUTEURS


- Anthony Slimani
- Laurine Drodzinski
- Noura Mares
- Sami Barchid
- Zoé Canoen


## INTRODUCTION

Ce document récapitule la conception du projet de l'application "One Minute". Le but de cette application est de faire gagner du temps au restaurant. Nous nous sommes mis à la place de tous les acteurs du restaurant, mais d'avantage sur la partie service, cuisine et menu du restaurant.

## SOMMAIRE



## RESUME DU PROBLEME

Le but de notre projet est d'informatiser le processus de gestion des commandes d'un restaurant. L'application doit permettre de prendre en charge une commande à partir de la demande du client jusqu'au règlement de l'addition en passant par le traitement en cuisine.

# SCENARIOS

## Liste des scénarios

1. *Ajout d'une préparation dans une commande*

Ajouter une préparation dans une commande déjà en cours.

2. *Établir un menu*

Créer ou modifier la carte de restaurant.

3. *Le client prend sa commande*

Un client prend commande en interagissant avec le système.

4. *Paiement*

Le client a terminé son repas et veut payer l'addition.

5. *Préparation indisponible*

Une préparation n'est plus disponible pour la clientèle et il faut le notifier à tous.

6. Prendre une commande

Un serveur prend la commande pour des clients.

7. Servir une commande

Une partie de la commande est prête à être envoyée au client.

8. Suivre une commande

Un membre du personnel veut suivre l'état d'avancement des commandes.

9. Cuisiner une préparation

Un cuisinier veut préparer une préparation pour un client.

# Description des scénarios

Les acteurs sont ***Bob*** (client), ***Alice*** (serveuse), ***Roger*** (cuisinier) et ***Albert*** (patron).

---

## AJOUTS DE PREPARATIONS DANS UNE COMMANDE

*Description :*

***Alice*** peut ajouter des préparations dans une commande déjà en cours si ***Bob*** le désire.

*Scénario :*

- ***Bob*** veut ajouter une préparation à sa commande en cours.
- ***Bob*** appelle ***Alice*** pour changer sa commande.
  - ***Alice*** n'est pas disponible.
- ***Alice*** prend sa tablette.
- ***Alice*** ajoute la nouvelle préparation à la commande.
  - La préparation n'est plus disponible.
- L'équipe de cuisine est avertie de l'ajout.
  - Problème de notification si problème de connexion à internet.

---

## ETABLIR UN MENU

*Description :*

***Albert*** crée ou modifie la carte du restaurant, avant l'ouverture ou après la fermeture du restaurant.

*Scénario :*

- ***Albert*** commence à créer ou modifier la carte dans l'application.
- ***Albert*** crée ou modifie divers éléments.
- ***Albert*** termine la création ou la modification.
- La carte s'actualise sur les diverses tablettes.
  - Problème d'actualisation si les tablettes ne sont pas connectées à internet.

---

## LE CLIENT PREND SA COMMANDE

La fonctionnalité étant optionnelle, elle sera décrite et travaillée ultérieurement. Nous la prévoyons pour la V2, voir V3. Nous ferons néanmoins le nécessaire pour pouvoir garder l'application la plus généraliste possible pour pouvoir incorporer cette fonctionnalité plus tard.

---

## PAIEMENT

*Description :*

- ***Bob*** a fini de manger et souhaite quitter le restaurant mais avant il doit régler ***l'addition*** de sa commande

*Scénario :*

- ***Bob*** demande ***l'addition*** à ***Alice***
  - ***Alice*** n'est pas disponible
- ***Alice*** recherche ***l'addition*** à partir du numéro de table de ***Bob*** par l'intermédiaire de l'application
  - ***Alice*** n'a pas accès au service de commande sur l'application
- L'application affiche ***l'addition***
- ***Alice*** informe le montant de ***l'addition*** à ***Bob***
- ***Bob*** paie ***l'addition*** par le moyen de paiement choisi
- ***Alice*** indique à l'application que ***l'addition*** a bien été réglée
- L'application archive la commande de ***Bob***

*Questions :*

- Faut-il générer ***l'addition*** de manière physique à ***Bob*** ?
  - ***L'addition*** a besoin d'être généré de manière physique notamment pour les notes de frais

---

## PREPARATION INDISPONIBLE

*Description :*

- ***Roger*** s'aperçoit que la réalisation d'une préparation ne peut être réalisée et doit donc rendre ***indisponible*** cette préparation.

*Scénario :*

- ***Roger*** indique que la préparation est ***indisponible*** sur l'application
    - Des commandes comprenant cette préparation sont en cours
        - L'application annule les commandes comportant cette préparation
        - L'application notifie les serveurs des commandes annulées
- L'application bloque la possibilité de sélectionner cette préparation lors d'une commande

[En attente de ***réapprovisionnement*** des ingrédients]

- ***Roger*** indique que la préparation est de nouveau disponible
- L'application débloque la possibilité de sélectionner cette préparation lors d'une commande

---

## PRENDRE UNE COMMANDE

*Description :*

***Alice*** propose à ***Bob*** de prendre sa première commande.

*Scénario :*

- ***Alice*** propose le menu à ***Bob***.
- ***Bob*** choisit ce qu'il souhaite commander.
- ***Alice***, à partir de sa tablette, transmet la commande et le numéro de table aux cuisiniers.
- Les cuisiniers sont au courant de la commande.

---

## SERVIR UNE COMMANDE

*Description :*

Lorsque qu'une ***partie de la commande*** est prête, il faut la servir à Bob.

*Scénario :*

- ***Alice*** est mise au courant qu'une ***partie de la commande*** est prête.
    - Problème de notification si problème de connexion à internet.
- ***Alice*** indique sur l'application qu'elle va la chercher.
- ***Alice*** récupère la ***partie de la commande***.
- ***Alice*** la ramène à Bob.

---

8. Suivre une commande

## SUIVRE UNE COMMANDE

*Description :*

Le personnel du restaurant peut suivre l'état d'avancement d'une commande en cours ou passée.

*Scénario :*

- **Bob** veut savoir si la commande de spaghetti d'**Alice** est prête.
- **Bob** sélectionne la commande d'**Alice** pour voir son état.
- **Bob** voit que les spaghettis d'**Alice** sont prêts.
- **Bob** voit que son mojito fraise n'est pas prêt.
- **Bob** va chercher les spaghettis en cuisine.
- **Bob** amène les spaghettis à **Alice**.
- **Bob** indique que les spaghettis sont servis.

---

## CUISINER UNE PREPARATION

*Description :*

***Roger*** désire faire une préparation, et doit informer l'application sur quelle(s) préparation(s) il est en train de cuisiner.

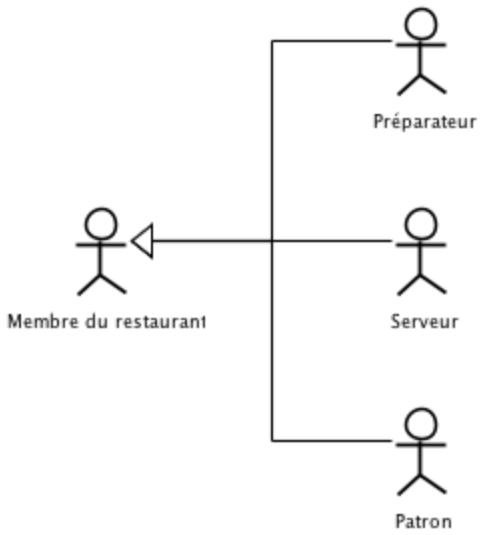
*Scénario :*

- ***Roger*** veut cuisiner une (ou plusieurs) préparation(s).
- ***Roger*** valide son identité pour voir les préparations qui lui sont assignées.
- ***Roger*** choisi une (ou plusieurs) préparation(s) qu'il s'apprête à cuisiner.
- ***Roger*** valide et peut commencer à cuisiner.
- Une fois que ***Roger*** a terminé sa préparation, il s'identifie à nouveau pour dire qu'il a terminé une préparation.
- Ensuite, SOIT
    - sa préparation permet de terminer une ***partie de la commande***, dans ce cas une notification est envoyée aux serveurs pour venir chercher la préparation.
    - il manque une préparation pour la commande, la commande est alors en attente de la préparation manquante. Et attend qu'un cuisinier cuisine la préparation manquante.

# DIAGRAMMES DE CAS D'UTILISATION

## Acteurs

Description des acteurs de l'application. Il est important de noter que les cuisiniers sont des préparateurs. Nous voulons faire quelque chose de généraliste, pour mettre dans la même classe le barman, le glacier, les cuisiniers...




Nous avons choisi de représenter ces acteurs en fonction du sujet. Nous avons bien conscience que dans certains restaurants, il n'y a pas forcément de barman, mais plutôt des serveurs comme préparateurs. De plus dans les restaurants, il y a souvent des managers, comme dans la restauration rapide par exemple.

*Membre du restaurant*

- Cet acteur est la généralisation des acteurs suivants :

*Patron*

- Gérant du restaurant.
- Possède un compte particulier qui lui permet de :
    - créer de nouveaux menus
    - modifier des menus existants
    - superviser l'état global du restaurant (avancement des commandes etc.)
- Doit être connecté à internet.

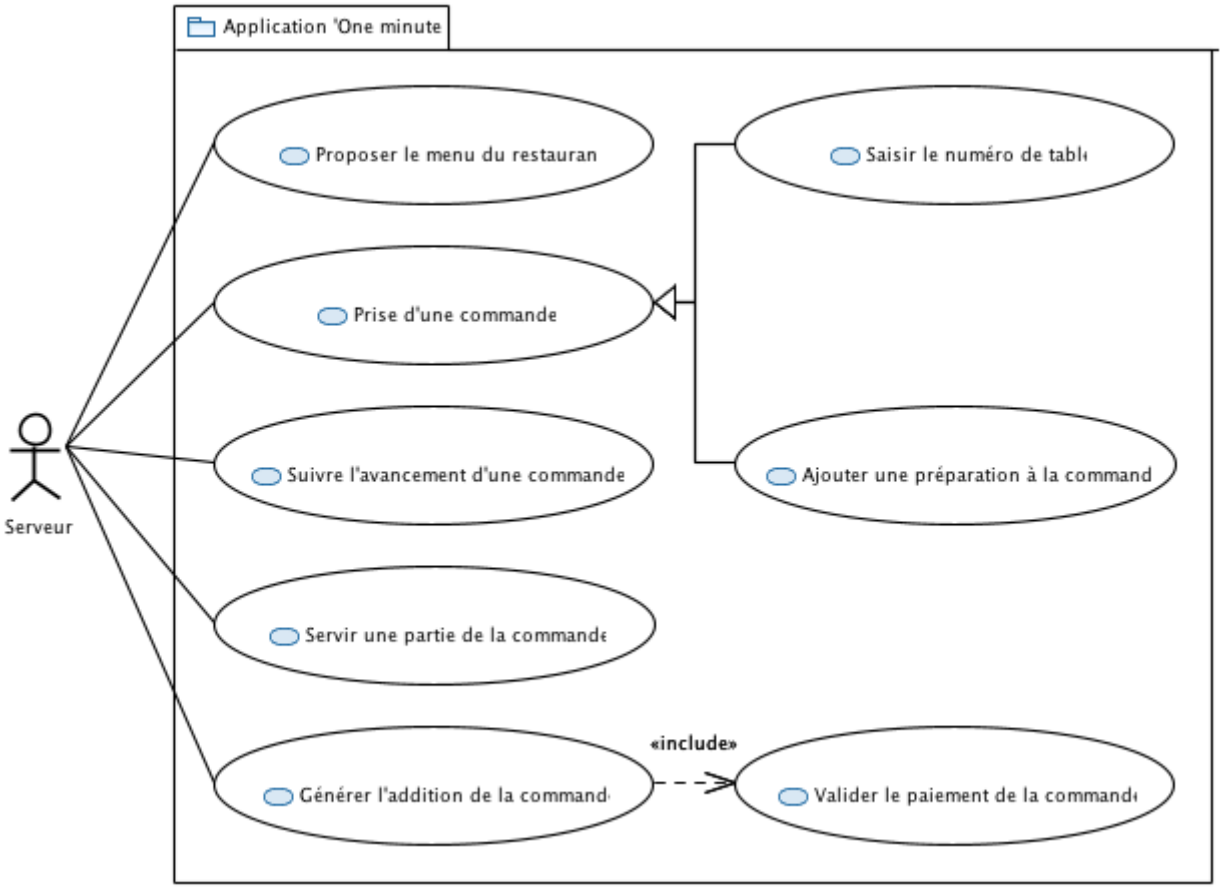
*Serveur*

- Est en contact avec les clients.
- Possède une tablette lui permettant de :
    - proposer le menu aux clients
    - prendre des commandes
    - suivre l'avancement des commandes
    - générer l'addition et valider le paiement
- Doit être connecté à internet.

*Préparateur*

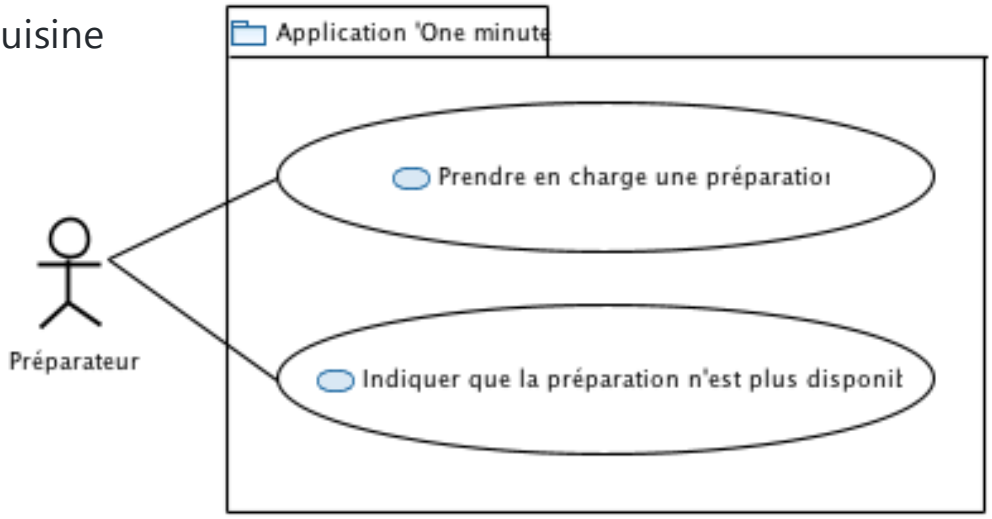
- Peut-être un barman, un glacier, ou simplement un cuisinier etc.
- Ne possède pas une tablette à lui seul (une tablette pour les cuisiniers, une pour les barmans, glaciers etc.).
- Celle-ci lui permet de prendre en charge une préparation et également de gérer la disponibilité des plats.
- Doit être connecté à internet.

## Service



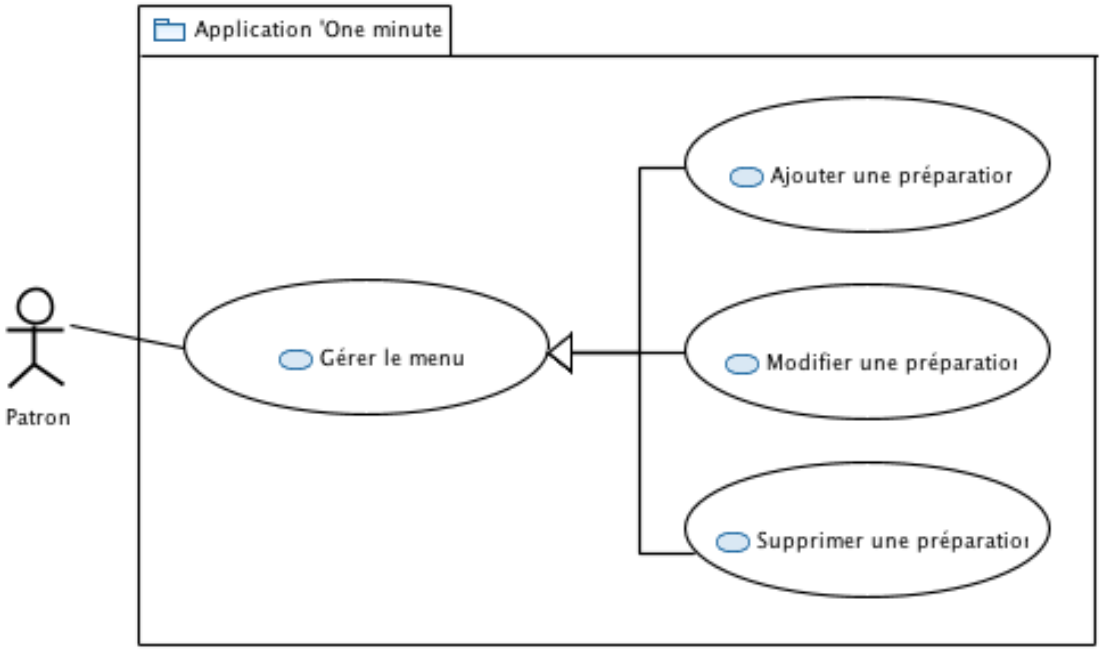

Le serveur a la possibilité de proposer le menu au client. Il pourra notamment prendre la commande de celui-ci. Pour cela, il devra saisir le numéro de table ainsi qu'ajouter une ou plusieurs préparations à la commande. Lorsqu'une préparation est ajoutée dans une commande, les préparateurs sont directement notifiés. Les serveurs ont également la possibilité de servir une partie de la commande dès lors qu'ils ont été notifiés par les préparateurs. Il peut également suivre l'état d'avancement de la commande. Dès que celle-ci est finie, le serveur pourra générer l'addition puis indiquer si le règlement a bien été réalisé.

## Cuisine



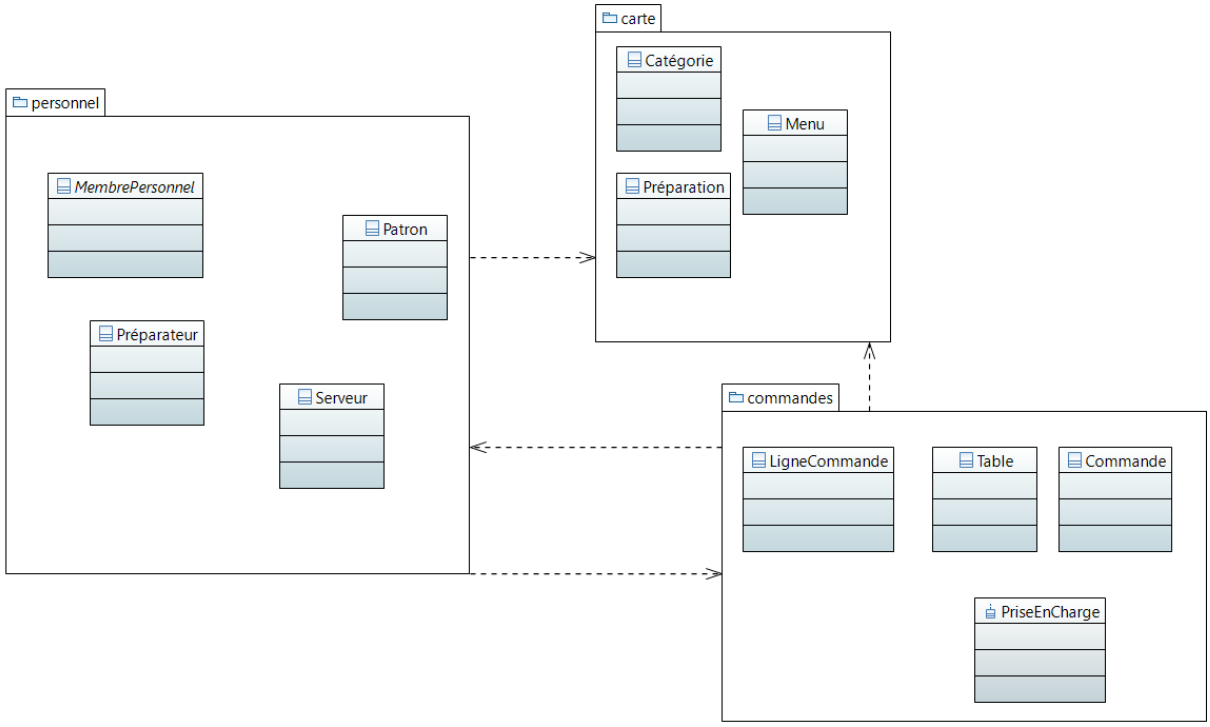

Le préparateur a la possibilité d'indiquer sur l'application la prise en charge d'une préparation. Dès qu'il a fini, il doit indiquer l'accomplissement de cette dernière. Cela a comme conséquence de notifier tous les serveurs. Il a également la possibilité d'indiquer qu'une préparation est indisponible.

## Gérer le menu




Le patron a la possibilité de gérer le menu, c'est-à-dire d'ajouter, de modifier et de supprimer une préparation d'un menu.

DIAGRAMME DE PACKAGES




# Explication du diagramme de packages

Ce document a pour but d'éclaircir les choix qui ont été fait pour le diagramme de packages.

*Division en trois packages*

La logique de l'application "One Minute" peut se diviser en trois grands "domaines" : la logique relative au personnel du restaurant, la logique relative à la gestion des préparations que propose le restaurant et la logique relative à la gestion des commandes.

Pour respecter cette séparation de logique, il a été choisi de décomposer également les classes en trois grands paquetages : le **personnel**, la **carte** et les **commandes**.

*Personnel*

Ce package contient toutes les classes liées aux utilisateurs de l'application : les employés du restaurant. Ce package est dépendant des deux autres packages car le personnel manipule les données du système.

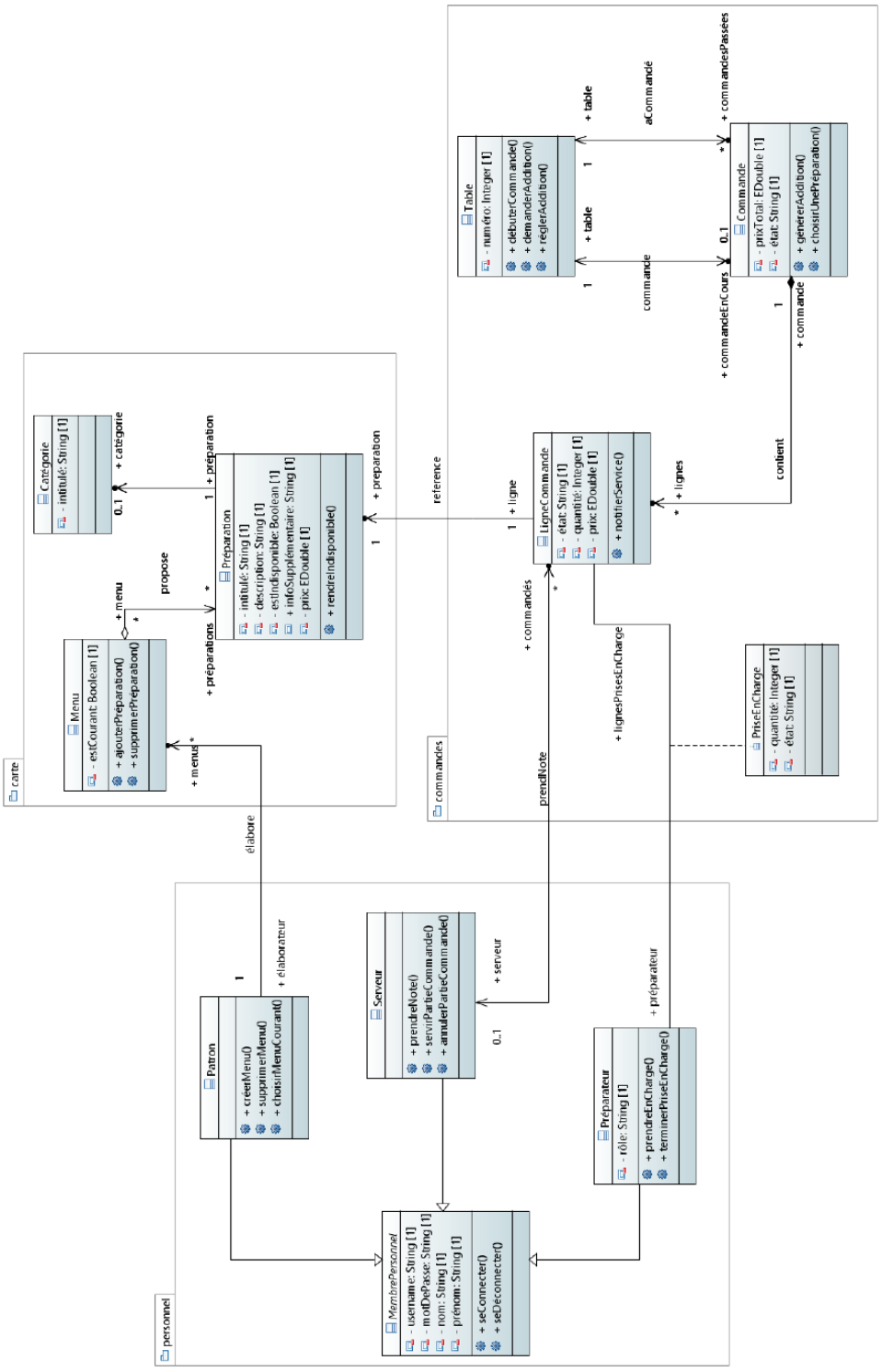
*Carte*

La carte est le package représentant les plats, desserts, préparations etc du restaurant. Ce package ne dépend d'aucun autre car il représente avant tout les classes qui seront manipulées par les autres packages.

*Commandes*

Le package des commandes représente toute la logique des commandes, de sa création en table à l'addition. Les commandes sont dépendantes de la carte ainsi que du personnel.

# DIAGRAMME DE CLASSES



# Explication du diagramme de classe

## Ce document a pour but d'éclaircir ou de justifier certains choix dans le diagramme.

*Les utilisateurs*

- **MembrePersonnel** représente tout utilisateur connecté de l'application et contient les infos de ces utilisateurs.
- **Patron**, **Serveur** et **Préparateur** représentent les différents utilisateurs pouvant se connecter et héritent donc de **MembrePersonnel**
- **Préparateur** représente tout membre du restaurant faisant partie des cuisines
    - Exemple : glacier, barman, cuisinier...

*Classe "Table"*

- Les clients, qui ne manipulent pas directement l'application, s'installe à des **Tables** qui seront utilisées pour la gestion des commandes dans le système.
- Les clients n'ont donc pas de représentation dans le système puisqu'ils n'interagissent pas avec ce dernier.
- Chaque table possède un **numéro** unique.
- Lorsque le client commence sa commande, celle-ci devient la **commandeEnCours** de la table.
- Lorsque l'addition est réglée, la **commandeEnCours** s'ajoute aux **commandesPassées** de la table.
    - Le champ **commandesPassées** est additionnel et existe à des fins d'historique non nécessaires dans l'immédiat.

*Gestion du menu*

- Une **Préparation** représente un plat, une entrée, un dessert, etc
- Une **Préparation** peut être reliée à une **Catégorie**.
- **Catégorie** représente une catégorie de préparation telle que Poisson, viande, dessert, boisson, etc
- Le **prix** d'une **Préparation** représente le prix courant de cette préparation à la carte.
- Un **Menu** propose plusieurs **Préparations** et une **Préparation** peut être proposée par plusieurs **Menus** différents.
- Il n'existe qu'un seul **Menu** courant qui est celui sur lequel on peut commander durant le service.
- Un **Patron** a tous les droits sur tous les **Menus**, il peut donc les créer/modifier/supprimer/etc.
- Une **Préparation** peut être temporairement **indisponible** (on ne peut plus la commander).

*Gestion de la commande*

- Une **Commande** possède plusieurs **états** :
  - Créée, En cours, Annulée, Terminée
- Afin d'éviter des calculs inutiles, la **Commande** retient son **prix total**
- Une **Commande** possède plusieurs **LigneCommande**.
- Une **LigneCommande** représente une **Préparation** choisie par le client ainsi que la quantité de cette **Préparation** à faire (ex : 2 spaghettis)
  - **Explication** : comme un client ne commande pas toujours tout ce qu'il veut en un coup au restaurant, il est plus sage de décomposer la commande en plusieurs lignes correspondant à ses demandes.
- Une **LigneCommande** possède elle-même plusieurs états :
  - En attente, En cours, Terminée, Servie, Annulée
- La **LigneCommande** est la donnée la plus manipulée par le système puisqu'elle sera :
  - créée par un serveur quand il **prend note** des demandes d'une table
  - notifiée en cuisine et en attente d'une prise en charge en cuisine.
  - en cours quand elle sera prise en charge en cuisine.
  - notifiée en service et terminée quand la prise en charge en cuisine est finie.
  - servie à la table par le serveur.
  - annulée si un problème survient.
- Une **LigneCommande** possède un **prix** (le prix total de la ligne s'obtient en faisant `prix * quantité`)
  - **Explication** : on ne peut pas se référer au prix courant dans la classe **Préparation** car ce prix courant change au cours du temps. Cependant, si je veux consulter l'état d'une commande d'il y a un mois, peut-être que le prix de mes spaghettis ont changé. Ceci m'amènerait à voir un prix qui n'est pas celui qui était correct un mois auparavant.
- Un **Préparateur** qui prend en charge une **LigneCommande** va générer une **PriseEnCharge**.
  - **Explication** : si une **LigneCommande** demande 64 spaghettis, il est peu probable qu'un seul **Préparateur** s'en charge. Donc chaque **Préparateur** choisit la **quantité** de spaghettis dont il s'occupera.
- Une **PriseEnCharge** a trois **états** :
  - En cours, Terminée, Annulée

## PREMIERES VERSIONS DES MAQUETTES

Ces maquettes seront amenées à changer, il s'agit de nos premières versions.

L'application aura plusieurs vues en fonction des utilisateurs. Chaque vue sera sélectionnable à gauche de l'écran. Nous souhaitons laisser à tous les membres du restaurant la possibilité de voir les différentes parties de l'application. A priori, les accès seront donnés par des managers. Le mieux sera de le faire en début de chaque jour, avant le service.

Le but de l'application est de gagner du temps. Notre application ne doit donc pas alourdir la charge de travail des préparateurs. Nous ne connaissons pas le nombre de tablettes disponibles dans le restaurant, nous en comptons une par serveur pour le moment, et une ou deux en cuisine. Il sera tout à fait possible d'en mettre moins, car nous sommes bien conscients qu'une tablette coûte assez chère. Cela pose donc certains problèmes, en effet, les préparateurs ne doivent pas perdre de temps en se connectant. C'est pourquoi, nous vous proposons d'utiliser notre application avec un compte pour la cuisine. Chaque préparateur doit juste s'identifier en cliquant sur un bouton avec son nom pour que l'on sache qui cuisine quoi. Cela leur permet de ne pas avoir à rentrer leur mot de passe.

### Maquette de la partie cuisine

*Première version de la maquette Cuisine*

Dans cette version pour pouvons observer qu'il y aura deux vues pour les préparateurs. Une première vue avec les catégories de plats, et une seconde en fonction des commandes. Le but des catégories était de faire gagner du temps aux préparateurs. Etant donné le fait que la vue des commandes est aussi importante, nous allons aussi la laisser.

*Deuxième version de la maquette Cuisine*


Suite à la création de cette maquette nous ai venu l'idée de laisser la possibilité aux préparateurs de ne pas perdre de temps en s'identifiant. C'est pourquoi ils auront juste à appuyer sur leur photo à droite, valider leur nom et choisir le plat qu'ils veulent s'approprier ou valider. Pourrons pas exemple tous se connecter sur le même compte, un compte spécialement réservé à la partie cuisine.

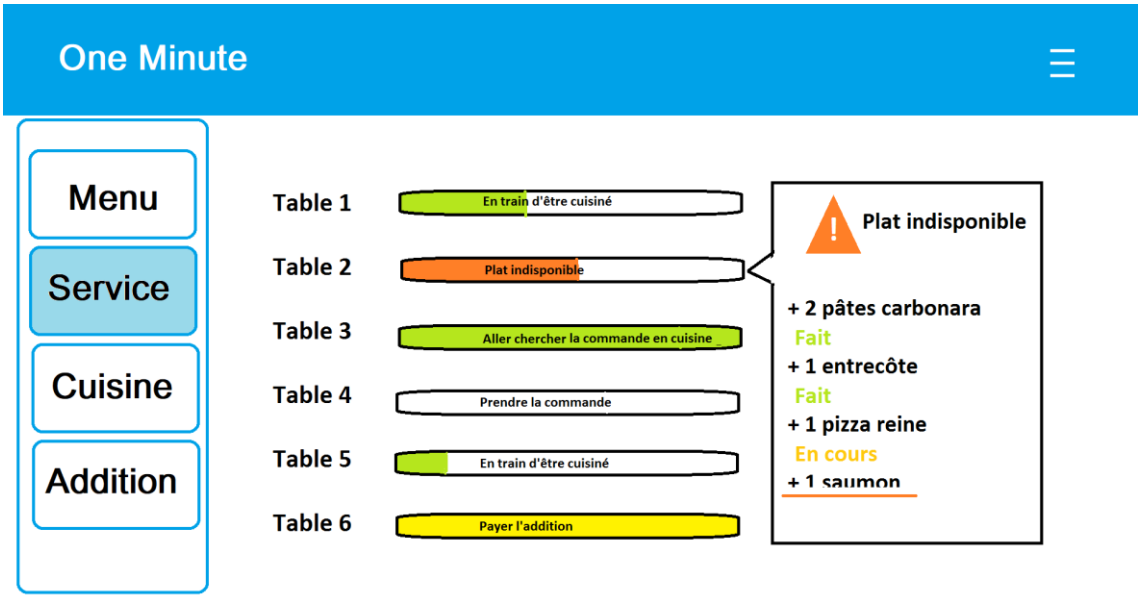
## Maquette de la partie service

Dans cette première version, l'idée était de permettre aux serveurs de recréer le restaurant sous la forme d'une vue. Ceci avait pour but de savoir quelle table était laquelle. Malheureusement cette solution était assez technique et allait prendre du temps.

*Première version de la maquette Service*


La seconde solution était plus portée sur quelque chose de plus simple techniquement. Mais, cela a permis de voir l'avancement des commandes plus facilement.

*Deuxième version de la maquette Service*




Nous pouvons observer ici la fonctionnalité de voir chaque commande, leurs états, les plats indisponibles s'il y en a, et s'il est temps de payer l'addition.

# GLOSSAIRE METIER

- ***Addition*** : Facture de la commande indiquant la liste des préparations commandées ainsi que le montant total de la commande.
- ***Bar*** *:* Endroit où sont préparées les boissons.
- ***Boisson*** : Préparation.
- ***Commande*** : Note qui sert à savoir ce que le client a choisi. Elle est prise par un serveur, et est transmise aux cuisines.
- ***Cuisine*** : Endroit où sont préparées les préparations.
- ***État*** : État d'avancement de la commande.
- ***Indisponible*** : État d'une préparation qui ne peut pas/plus être réalisée par manque de un (ou plusieurs) ingrédient(s) la constituant.
- ***Note*** : Liste des plats qu'écrit le serveur
- ***Partie de la commande*** : pack de boissons, de plats ou de desserts etc.
- ***Préparation*** : Plutôt que plat, nous employons le terme préparation qui est plus générique. Il contient les plats, les boissons, les desserts…
- ***Réapprovisionner*** : Action de recevoir un nouvel apport de marchandises afin de renouveler les stocks.
- ***Service*** : Fonction à gérer pour les serveurs du restaurant. Cela consiste à prendre la commande, amener les plats, et régler la commande.
- ***Stock*** : Ingrédients disponibles pour préparer les commandes.

# ANNEXES

Les annexes sont tous les documents que la MOA n'a pas besoin. En effet, pour vous permettre de juger au mieux du travail effectué, nous vous invitons à lire nos annexes qui contiennent :

- Un retroplanning du rendu 1
- Les synthèses de nos séances
- Le tableau types-concepts

# RETROPLANNING

## Retroplanning de Laurine Drodzinski

*Travail effectué*

Au cours de cette première itération j'ai eu l'occasion de travailler sur les scénarios. Après en avoir réalisé, j'ai pu retravailler le reste des scénarios afin qu'ils soient cohérents tant au niveau des acteurs, de la présentation, du vocabulaire etc. J'ai également travaillé sur le tableau regroupant divers termes utilisés au cours de cette itération. Ensuite, j'ai décrit les différents acteurs de l'application en indiquant leurs possibilités d'actions ainsi que quelques contraintes les concernant. Par ailleurs, j'ai pu accompagner Anthony sur les diagrammes de cas d'utilisation. Puis, pour finir, j'ai corrigé les fautes d'orthographes des différents fichiers du git ainsi que remis en forme divers tableaux et dossiers.

*Difficultés rencontrées*

J'ai rencontré quelques problèmes de compréhension au début de projet concernant l'organisation du projet. En effet nous avions éparpillé des données sur diverses plateformes. Cependant ce problème fut très rapidement réglé car dès la deuxième séance nous nous sommes réorganisés et nous avons regroupés les informations sur git.Le second souci a été une confusion à propos de certaines informations du cours. Notamment au niveau des diagrammes de cas d'utilisation et des différentes flèches à utiliser. Afin de régler ce problème nous nous sommes rapprochés de Cédric Dumoulin qui nous a réexpliqué leur sens.

# Retroplanning de BARCHID Sami

*Tâches faites*

- Élaboration des diagrammes de classe
- Élaboration du diagramme de package
- Élaboration de scénarios concrets

*Méthode de travail*

Ma manière de travailler personnelle dans le projet a été de profiter des TPs pour mixer un maximum des jugements de chacun sur tous les travaux à réaliser (diagrammes, scénarios, etc) afin d'être directement fixé sur la façon de remplir mes tâches. Personnellement, je pense que les deux heures de TPs par semaine sont plus utiles si on les utilise pour entrechoquer nos idées que réellement écrire quelque chose qui ne satisfait pas tout le monde.

*Difficultés*

La principale difficulté vue pour ce projet est l'absence totale de client à qui poser des questions, laissant les ambiguïtés de "l'appel d'offre" à notre appréhension personnelle.

Une difficulté, plus secondaire, est la différence de compréhension des standards UML entre tous (élèves, professeurs et manuels compris), qui souligne bien l'importance d'établir des conventions entre les différents membres de l'équipe. Cette difficulté est surmontée grâce aux différents débats entre nous lorsqu'une ambiguïté était rencontrée.

## Retroplanning de Zoé Canoen

*Tâches faites*

- Mise en place des différents outils lors du début du projet
- Gestion de la répartition des charges au sein de l'équipe
- Synthèse des séances
- Rendu du lot 1 (format markdown et pdf)
- Premières maquettes de l'application (format fixe)

*Méthode de travail*

Pour moi il est important d'avoir une vision globale du sujet. Cela permet de gérer au mieux les délais. La répartition des tâches et la confiance en chacun des membres de l'équipe est primordiale. J'aime apporter des nouvelles idées et je pense que la communication dans l'équipe est importante. Au début je parlais à chacun, mais maintenant nous faisons des réunions tous ensemble pour une mise en commun.

*Difficultés*

Je pense que nous avançons très bien, et que nous faisons face à tous les évènements sans trop de difficultés. La capacité d'adaptation du groupe est très importante. Je suis bien consciente que ce n'est pas le seul projet sur lequel nous devons travailler, et que parfois certaines personnes dans le groupe n'ont pas le temps de travailler GL. Mais notre groupe se rééquilibre toujours pour continuer à avancer. Si une personne travaille beaucoup plus que les autres, je veille à ce que ces tâches soient minimisées. Et à l'inverse, si une personne n'a pas le temps de réaliser une tâche, je m'arrange pour que la tâche soit redistribuée si les délais sont trop courts. Nous nous faisons confiance sur le travail effectué, chaque personne travaille dans la mesure de son temps disponible, et fini toujours par rattraper son retard quand elle le peut. Nous pouvons tous compter sur les autres si besoin, et donc avancer à notre rythme.

*Ressenti*

J'aurai adoré n'avoir que GL à travailler ce semestre. C'est réellement une matière fortement intéressante. Au début nous devions trouver notre rythme de croisière et je crois que nous l'avons trouvé. Les réunions de groupe lors des tps sont vraiment très utiles et nous avançons beaucoup. Chacun peut ainsi avoir une vision globale du projet.

## Retroplanning of Noura

*Done tasks*

- Description of case of use Package diagram elaboration of one version

*Work's Method*

For me, working within a team is a greatfull experience ; during the "TP", I try usually to understand the ideas of my colleagues and mainly try to explain mine. The most important thing that I try to get it after each meeting is to gather what Mr Michaël Launay tells us and what we sugggess to do as a srategy. Even between us, we have really many points of view, so that to make a collective decision is really so interesting.

*Difficulties*

If I will speak about difficulties , i will start by the difference of language with our responsible and with my colleagues. After each meetiig , and when I come back to home I try to remember the most important suggession, sentences and even words and I start "traduction service", it takes much time. Besides, one of the most things that was difficult for me, is that our project seems to be clear in his glabal view:"take an order inside a restaurant", but some details about at which level we are limited to designing this application ; should we just take into consideration "the service give by the waiter" or may be be so selective when we had to choose the performance evaluation criterion (good service/client satisfaction/price/time taken by the application), make it a bit complicated.

# Retroplanning de Anthony SLIIMANI

*Tâches faites*

- Élaboration de scénarios concrets & un alternatif
- Élaboration des diagrammes des cas d'utilisations
- Participation à l'élaboration du diagramme de classes

*Difficultés*

On a pu constater un début difficile au niveau de l'organisation mais nous avons pu nous harmoniser au fil des semaines. Nous avons des idées intéressantes mais parfois qui s'écarte un peu trop du sujet principal ou encore n'a pas la nécessité d'être élaborer pour cette première itération. Nous avons eu quelques difficultés sur les standards UML en constatant diverses différences avec le cours, les livres de référence ou encore des documents sur Internet.

*Ressenti*

Cette itération m'a montré que toute l'équipe était motivée pour ce début de projet. J'espère que cette motivation va perdurer durant les prochaines itérations. Une bonne cohésion d'équipe s'est formée durant ces quelques semaines. Tout le monde essaye de s'adapter aux autres, à son rythme. J'aime beaucoup la manière que chacun appréhende le sujet, de voir la diversité de nos idées et l'ouverture d'esprit de chacun lors des échanges de nos idées pour parvenir à une solution qui correspond le plus aux besoins du client.

# Synthèse des séances

## Synthèse de séance 01

**Semaine du 10 Septembre**

15 minutes de lecture du sujet 15 minutes pour poser le problème 2h15 pour faire les scénarios ensemble 15 minutes pour savoir qui fera quoi hors séance, et comment nous allons travailler (github, trello..)

METHODE DE TRAVAIL

## En séance

Après une rapide lecture du sujet, la majorité des idées concordaient entre nous au niveau du choix des scénarios. Il a été décidé de répartir alors les différents scénarios trouvés entre nous. Zoé CANOEN avait pour rôle de noter nos diverses réflexions sur papier et de répartir les autres membres du groupe sur les scénarios à faire.

En parallèle, lorsque l'un de nous avait une question, nous essayions d'y répondre ensemble pour clarifier au mieux le scénario et trouver des problèmes auxquels nous n'aurions pas pensé au départ (voir les points discutés au chapitre référé).

*L'écriture des scénarios*

L'écriture des scénarios a été faite par chacun, mais nous avons tout regroupé vers le milieu de la séance pour voir ce que les autres avaient fait. Nous nous sommes aperçus qu'il y avait des points à discuter. En effet, le sujet ayant des zones d'ombre, nous devions pouvoir en parler avec le client. Mais puisque nous ne pouvions pas parler au client, nous avons allions devoir trancher, maintenant ou plus tard.

Il est important de noter que chaque scénario a été créé atomiquement, car le sujet rend assez bien compte des fonctionnalités de l'application pour chaque acteur. C'est pourquoi les scénarios se limitent au point de vue d'un acteur.

### La caisse

Nous ne sommes pas sûrs que l'API doivent gérer le paiement ou si elle doit juste servir d'affichage de l'addition. Nous nous préoccuperons de cela plus tard. Pour le moment elle ne fera qu'afficher la note.

**Le visuel du client / serveur / cuisinier / patron **

L'application sera à priori fournie soit sous 4 applications différentes, soit sur une seule avec 4 visuels différents. En effet, le client ne doit pas pouvoir modifier les plats, avoir accès à l'addition, et les cuisiniers ne peuvent pas modifier une commande.

### Ce que peut faire le client :

- Remplir le numéro de sa table
- Prendre une commande
- Suivre la commande

### Ce que peut faire le serveur :

- Gérer l'ensemble des tables du restaurant (ajouter ou enlever des clients d'une table, ajouter ou supprimer une table)
- Créer une commande pour un client
- Modifier une commande pour un client
- Gérer l'addition d'une table pour régler la commande

### Ce que peut faire un cuisinier :

- Voir la liste des plats qui le concerne
- Réserver un plat qu'il est en train de faire, pour ne pas que d'autres personnes le prenne
- Informer le client qu'un plat est en train d'être fait, ou est fini
- Donner une information sur le fait qu'un plat est en rupture de stock

### La selection des plats par les cuisiniers

Nous avons commencé à voir comment serait l'affichage des plats pour les cuisiniers.

## Hors séance

- Pour organiser le travail à faire pour chaque semaine, il a été décidé d'utiliser **Trello**.
- Les diagrammes seront réalisés sur le logiciel "**draw.io**", permettant de créer des diagrammes collaboratifs (utilise des fichiers XML).
- Le dépôt Git de l'école étant encore indisponible pour la majorité des alternants suite à des retards administratifs, un deuxième `remote` sur **Github** a été installé temporairement à cette adresse.
- Pour faciliter les discussions moins formelles entre nous, nous utilisons un réseau de chat en ligne.
- Les scénarios alternatifs seront à faire plus tard.

# Synthèse de séance 02

SEMAINE DU 18 SEPTEMBRE

- (1h50 de TD sur le sujet TinCar)
- 1h10 de TP sur notre projet

METHODE DE TRAVAIL

## Avant la séance

***Anthony*** s'est renseigné sur le fonctionnement d'une cuisine de restaurant.

- Liens :
- http://technoresto.org/environnement-technologique/lorganisation-du-travail-au-restaurant/
- http://technoresto.org/tr/brigade/

## En séance

Nous avons chacun écrit sur un Google Doc le glossaire, pour mettre en commun plus facilement ce que nous avions. Nous avons pu redéfinir les termes que nous allions utiliser pour l'application. Ceci a permis de mettre en évidence les différents futurs **cas d'utilisation**. Nous avons donc pu prendre en main le logiciel **Eclipse Papyrus** qui servira aux diagrammes d'utilisation.

Dans un premier temps, le glossaire a été rempli à la manière d'un brainstorming, permettant par la suite de débattre ensemble pour définir les termes que nous allions utiliser pour la suite du projet.

Nous avons également déjà pensé à quelques classes pour élaborer une première ébauche d'un diagramme de classes.

## Le début du glossaire

Nous avons commencé un glossaire, en trouvant les mots importants qu'il fallait définir, ainsi que les cas d'utilisation qu'il allait falloir faire.

APRES LA SEANCE

- Les diagrammes ne sont plus réalisés sur "**draw.io**" mais **Eclipse Papyrus**. Chacun va réaliser des diagrammes de cas d'utilisation, qui se rapprochent des scénarios associés.
- ***Sami*** et ***Anthony*** vont commencer les diagrammes de classes
- ***Laurine*** et ***Noura*** vont répertorier toutes les spécifications de chaque acteur de l'application.
- ***Zoé*** va se renseigner sur comment fonctionne techniquement l'envoie des notifications sur une application et trouver tous les cas d'utilisation à faire.

JUSTIFICATION DES CHOIX

## Diagrammes de classes

- Tous les employés du restaurant auront, à l'avenir, plusieurs attributs similaires. C'est pourquoi il est décidé de faire hériter tout membre du personnel du restaurant de la classe abstraite : **MembrePersonnel**.
- Les préparateurs du restaurant peuvent avoir plusieurs rôles différents : barman, glacier, cuisinier etc. Pour l'instant, chaque rôle de préparateur est une classe héritant de la classe **Préparateur**. Par la suite, il pourrait s'avérer qu'une distinction en classes de tous les préparateurs ne soit pas utile. C'est pourquoi ce choix est temporaire.

# Synthèse de séance 03

**Semaine du 25 Septembre**

- 2h00 de TD
- 1h00 de TP

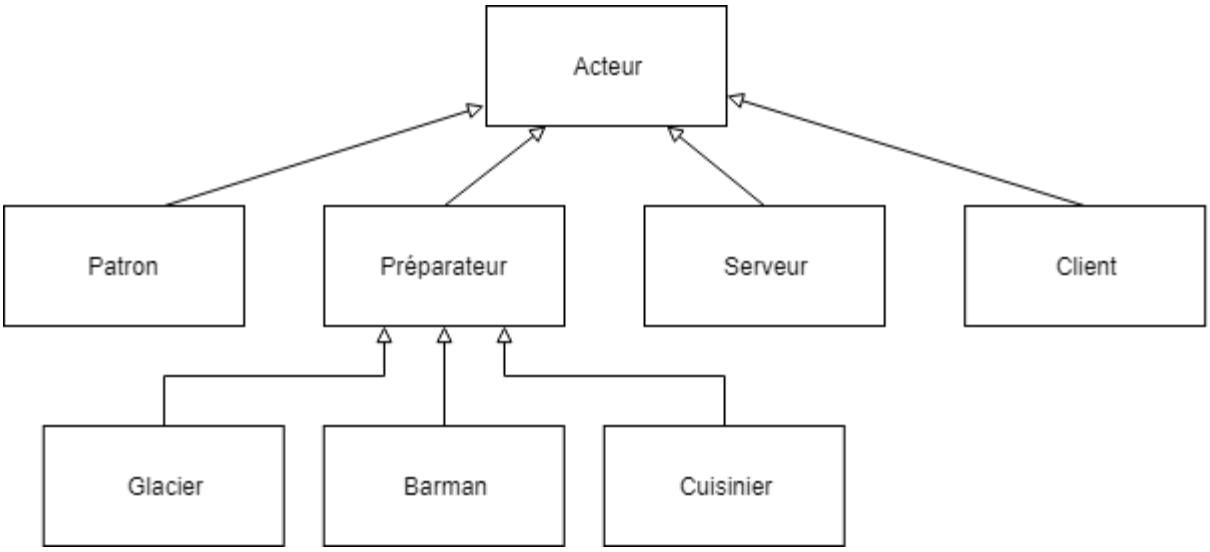
METHODE DE TRAVAIL

## En séance

Nous avons fait un bilan de tout ce qui a été fait. Et nous nous sommes séparés en deux groupes. Sami et Noura se sont occupés des diagrammes de classes. Laurine et Anthony se sont occupés de finir les diagrammes de CU.

*Bilan de ce qui a été fait*

Lors des précédentes séances nous avions pensé à faire cela :




**Mais !** Nous ne nous sommes pas rendu compte que les classes comme "glacier" ou "barman" ne sont que des instances de la classe. Donc nous allons simplement avoir des "préparateurs", qui seront la classe qui s'occupera de préparer les commandes en cuisine. Nous avons donc 3 classes membres du restaurant :

- Le patron, qui a la gestion administrative de l'application. C'est lui qui peut gérer le menu.

- Le serveur, qui s'occupe de prendre la commande au client. Il pourra modifier la commande du client.
- Les préparateurs, qui s'occuperont de valider les plats effectués, et indiquer les plats indisponibles.

Il faut savoir que nous allons vouloir faire une interface graphique différente pour chaque partie du restaurant. En effet, les besoins de toutes les équipes sont différents. Il faut que l'application soit pratique pour chaque personne. Des maquettes ont déjà été discutées lors de la première séance.

*Diagrammes de CU*

Anthony s'est occupé du diagramme de CU, avec tous les CU de notre projet

*Diagrammes de classe*

Sami s'est occupé de faire les diagrammes pour toutes les classes de notre projet. Il a été aidé par Noura qui s'est occupé de remplir les classes.

# Synthèse de séance 04

**Semaine du 2 Octobre**

- 1h30 de TD
- 1h30 de TP

METHODE DE TRAVAIL

## En séance

Les séances servent de plus en plus à faire le bilan des semaines tous ensemble. Cela est mieux comme ça, nous avançons tous ensemble. De plus, nous avons pu voir le cas de Noura, qui commence réellement à s'intégrer au groupe. Son arrivée a été compliquée pour elle car il y avait la barrière de la langue. Avec ce fonctionnement, cela lui permet de suivre tout ce qu'on fait et de réagir et de proposer ses idées. Nous arrêtons de nous séparer en petits groupes pour qu'elle puisse au maximum avoir une vision globale du sujet.

De plus, il est maintenant plus facile de garder les idées et remarques de chacun.

## Choses à faire pour le rendu de VENDREDI 05 SEPTEMBRE A 20H

Voici toutes les remarques que nous avons notées rapidement :

**Sami**

- DIAGRAMME DE CLASSES :
  - Rajouter les packages sur le diagramme de classes
  - Modifier des flèches droites
- DIAGRAMME DE PACKAGES :
  - Rajouter "prise en charge", comme l'a fait Noura.
- DES PETITS DIAGRAMMES DE CLASSES :
  - Des petites images / diagrammes de classes : découpage des packages (en plus)

### Anthony

- DIAGRAMMES CU :
    - Modifier les titres de type "Service", "Cuisine" en "Application"
    - Supprimer les notifications des diagrammes **OU** modifier en commentaire
    - Ajouter une description "notification" dans les descriptions des CU, si ce n'est pas fait.
    - CUISINE : supprimer notification, mettre un commentaire pour l'include ("il faut avoir fini la commande")
    - Générer l'addition PROBLEME d'include (à supprimer), rajouter la fonctionnalité "Générer l'addition"

### Noura

- Rapport_final.md
    - Add this in the glossary :
        - "service" (for the waiters)
        - "cuisine" (for the cookers)

### Zoé

- MAQUETTES :
    - ajouter les acteurs cuisiniers dans la cuisine
    - modifier l'interface du service. Faire une liste de table plutôt qu'une sorte de modélisation.
    - Modifier le texte : "voir les avancements" => "voir les commandes"
- DOCUMENT PDF
    - Mise en forme du Markdown sous Word avec un joli Template

### Laurine

- changer les scénarios : de plat en préparation
- Mettre à jour le tableau suite aux modifications faites
- Vérifier chaque fichier pour les accents et les fautes d'orthographe

## Discussions

Une discussion a commencé à émerger à la fin de la séance. Notre application est très peu permissive. Nous avons besoin d'avoir le point de vue du client. En effet, dans cette conception, les serveurs n'ont pas accès à la cuisine. Nous allons chercher, dans le futur, à la rendre plus permissive. Mais il ne faut peut-être pas

arriver au cas extrême où nous n'avons aucune spécification sur serveur / préparateur / patron.

## Définition des rôles

Au fur et à mesure, chacun a obtenu un rôle assez précis. Cela est visible dans le récapitulatif de ce que chacun doit faire. **Anthony** et **Sami** sont devenus nos experts de Papyrus. Nous avons choisi de fonctionner ainsi pour éviter les conflits entre tous les ordinateurs.

**Anthony** a également la vision de faisabilité technique. **Zoé** aime vouloir ajouter des nouvelles fonctionnalités. **Anthony** juge régulièrement ce qui est faisable rapidement ou ce qui va prendre du temps.

**Laurine** s'occupe de la vision avec la MOA et la livraison à faire. Et elle s'occupe de la faisabilité en termes de temps. Elle n'a pas de rôle défini, ce qui nous permet d'avoir toujours de l'aide si besoin. Elle a la capacité de s'adapter à toute situation si on lui assigne une tâche, ou de prendre les tâches elle même si elle n'a rien à faire.

**Sami** a le rôle d'expert technique, et est plus notre architecte technique. C'est lui qui produit les différents documents techniques, avec les fluxs de données. Et c'est lui qui est expert pour savoir quelle classe communique avec quoi.

**Noura** s'intégre de plus en plus. Les choses qui lui sont demandées peuvent paraître faible. Cependant, il faut savoir qu'elle traduit tout ce que nous faisons. Car nous produisons tous nos documents en français. C'est pourquoi, nous essayerons d'écrire au maximum en anglais pour l'aider. Son travail lui prend du temps en plus des cours, donc elle a eu beaucoup de mal au début, mais elle reste toujours active en séance.

**Zoé** a la vision d'équipe pour que chacun puisse faire quelque chose. Elle se fait aider de **Sami** pour savoir les tâches faisables pour **Noura** ou **Laurine** en fonction de l'aide dont il a besoin. Elle doit limiter le travail de Sami et Anthony qui sont moteurs de l'équipe. Donc elle doit dispatcher leur travail sur **Noura** et **Laurine**. Elle s'occupe aussi du rendu, d'écrire les synthèses des séances, et de regrouper tout ce qui a été fait. Elle doit être à l'écoute de tout le monde.

# Tableau TYPES-CONCEPTS

| SUJET | TYPE | CONCEPT |
|---|---|---|
| Bob | Acteur / (Classe dans le futur) | Client |
| Alice | Acteur / Classe | Serveur du restaurant |
| Albert | Acteur / Classe | Patron du restaurant |
| Roger | Acteur / Classe | Préparateur / Cuisinier |
| L'équipe de cuisine | Acteur**s** | Ensemble des préparateurs |
| Table | Classe | Table du client |
| Menu | Classe | Carte du restaurant |
| Préparation | Classe | Entrée / Plat / Dessert / Boisson |
| Commande | Classe | Ensemble de toutes les préparations, commandées par le client |
| Partie de commande | Donnée / Classe | Ensemble de boissons, entrées, plats ou desserts.<br>Donnée de la commande |
| Addition | Donnée (Décimal) | Donnée de la commande |
| Numéro de table | Donnée (Entier) | Donnée de la table |
| Glacier | Objet | Préparateur |
| Barman | Objet | Préparateur |
| Proposer le menu | Action / Objectif / Cas d'utilisation | Le serveur propose le menu au client |
| Ajouter une préparation | Action / Objectif / Cas d'utilisation | Le serveur ajoute une ou des préparations dans une commande en cours, sous demande du client |
| Saisir le numéro de table | Action / Cas d'utilisation | Le serveur saisi le numéro d'une table pour référencer une commande |

| | | |
|---|---|---|
| Servir une partie de commande | Action / Cas d'utilisation | Le serveur amène au client une partie de la commande et indique que la livraison va être réalisée |
| Valider le paiement | Action / Objectif / Cas d'utilisation | Le serveur indique sur l'application que l'addition a bien été réglée |
| Suivre l'avancement d'une commande | Action / Objectif / Cas d'utilisation | Les serveurs peuvent suivre l'état et l'avancement d'une commande |
| Créer/Modifier le menu | Action / Objectif / Cas d'utilisation | Le patron apporte des modifications au menu ou en crée un nouveau |
| Valider le menu | Action / Cas d'utilisation | Le patron valide le menu après avoir effectué des modifications |
| Prendre en charge une préparation | Action / Objectif / Cas d'utilisation | Le préparateur sélectionne la préparation dont il souhaite s'occuper |
| Valider une préparation après l'avoir cuisiné | Action / Cas d'utilisation | Le préparateur valide la préparation une fois qu'il a terminé de la cuisiner |
| Rendre une préparation indisponible | Action / Cas d'utilisation | Le préparateur indique qu'une préparation est indisponible si les ingrédients ne sont pas suffisants |
| Demander l'addition | Action / Objectif | Le client demande l'addition à l'un des serveurs |
| Régler l'addition | Action | Le client paie l'addition quand il souhaite partir |
| Notification | Objet de médiation | Message destiné aux serveurs ou aux préparateurs |
| Affiche l'addition | Objet de médiation | Message destiné au client |
| Tablette | Support | Support utilisé par les membres du restaurant |
| OneMinute | Application | Application utilisée par les membres du restaurant |