Dossier d'architecture

DOSSIER ARCHITECTURE
PROJET RESTAURANT – Exia A3

Pons Emilien
Gilly Pierre
Mazurier Pierre
Longatelli Jonathan
Leon Montes Roberto

Exia A3 - Toulouse

Dossier d'architecture

Table des matières

Table des figures :

Projet Restaurant – Exia A3

Dossier d'architecture

1. CADRAGE DU PROJET

CHARTE PROJET

OBJECTIFS

L'objectif de ce projet est de réaliser une simulation d'un restaurant afin de faire des propositions d'améliorations. Ces améliorations devront reposer sur des éléments mesurables. Ces propositions d'améliorations permettront au directeur du restaurant d'optimiser son établissement.

ENJEUX

Les enjeux de ce projet sont des enjeux économiques et humain. Si le projet décèle des optimisations, il se peut que le restaurant gagne plus d'argent. De plus si la simulation détecte une situation ou il y a trop d'employé à un endroit ou pas assez, il se peut que le nombre d'employés fluctue.

ACTEURS

Les parties prenantes de ce projet sont :

- La grande chaîne de restaurant
- Le directeur du restaurant

Les personnes pouvant être impactés par ce projet sont toutes les personnes travaillant dans le restaurant. Il se peut aussi que les autres employés de la chaine soient impactés.

Les prestataires de ce projet sont :

- Emilien Pons
- Pierre Mazurier
- Roberto Montes
- Jonathan Longatelli
- Pierre Gilly

DEFINITION DES RESPONSABILITES

Pour ce projet :

- Emilien Pons (Chef de projet) : sera responsable de la partie Cuisine du projet.
- Pierre Mazurier : sera responsable de la Base de Données ainsi que de son accès.
- Roberto Montes : sera responsable de la Vue de l'application.
- Jonathan Longatelli :sera responsable de la Vue de l'application.
- Pierre Gilly : sera responsable de la partie Salle du projet.

Projet Restaurant – Exia A3

Dossier d'architecture

CAHIER DES CHARGES

BESOINS

L'objectif de ce projet est faire un programme afin de simuler le fonctionnement d'un restaurant afin de détecter des possibles améliorations.

L'entité concernée par ce projet est une chaine de restaurant et plus précisément un restaurant de cette chaine.

Le projet vient de la prise de conscience des incidents survenues dans le restaurant : « file d'attente trop importante, désorganisation du service entre la salle et la cuisine, manque de matériel pour servir ou cuisiner entrainant de très longues attentes en salle pour les clients… »

Les raisons qui ont fait apparaitre ce besoin sont des raisons économiques et de notoriété : si nous arrivons à proposer des possibles améliorations, le restaurant gagnera plus d'argent. De plus si les clients sont plus heureux dans l'établissement, la notoriété du restaurant et plus globalement de la chaine vont grandir.

CONTRAINTES

Il n'y a pas de contraintes techniques, ni humaines, ni économiques, ni organisationnelles.

Il y'a cependant une contrainte de développement qui encourage à coder en TDD.

CRITERES DE REUSSITES

Ce projet pourra être considéré comme réussi si nous arrivons à proposer un programme fonctionnel et si nous arrivons à soumettre des améliorations possibles.

DOCUMENTS & CIE

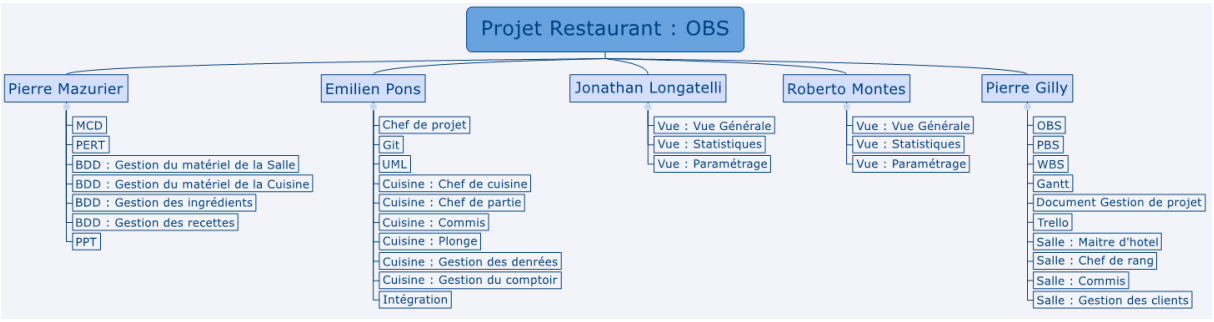
La totalité des documents (images et rapport) seront disponible sur le git du projet à l'adresse suivante : https://github.com/Emilien-Pons/Projet-Restaurant

« Dossier architecture », situé à la racine du master.

GESTION DE PROJET

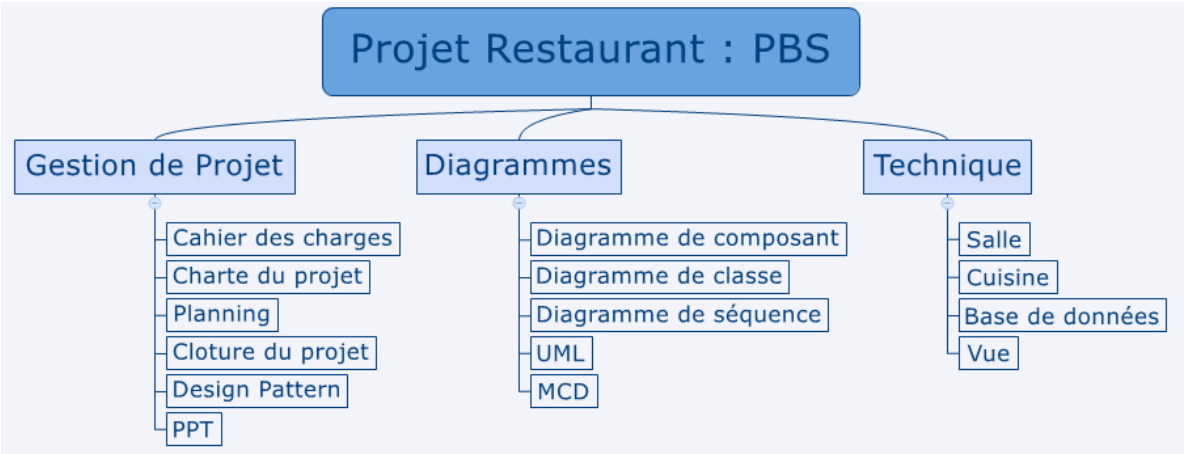
OBS / PBS / WBS

Voici le schéma OBS qui représente l'organisation des taches au sein du groupe. Il divise les tâches et responsabilités aux différents acteurs du projet.



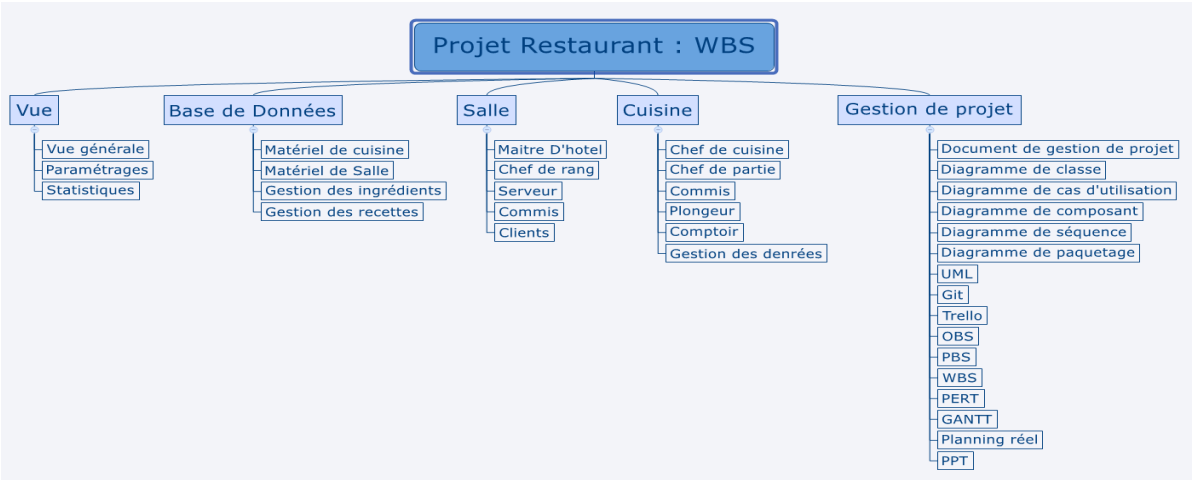
OBS

Voici le schéma PBS qui représentes les livrables du projet.



PBS

Voici le schéma WBS qui représente la liste des taches à effectuer pour construire le projet :

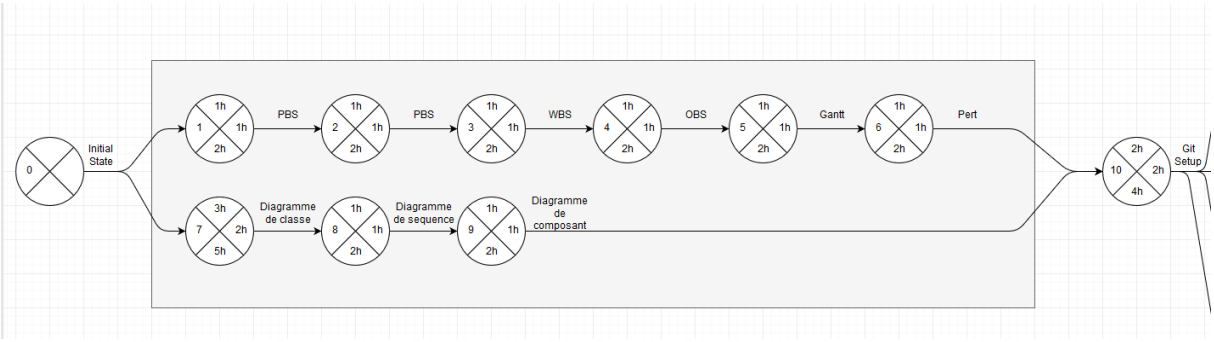


WBS

PERT

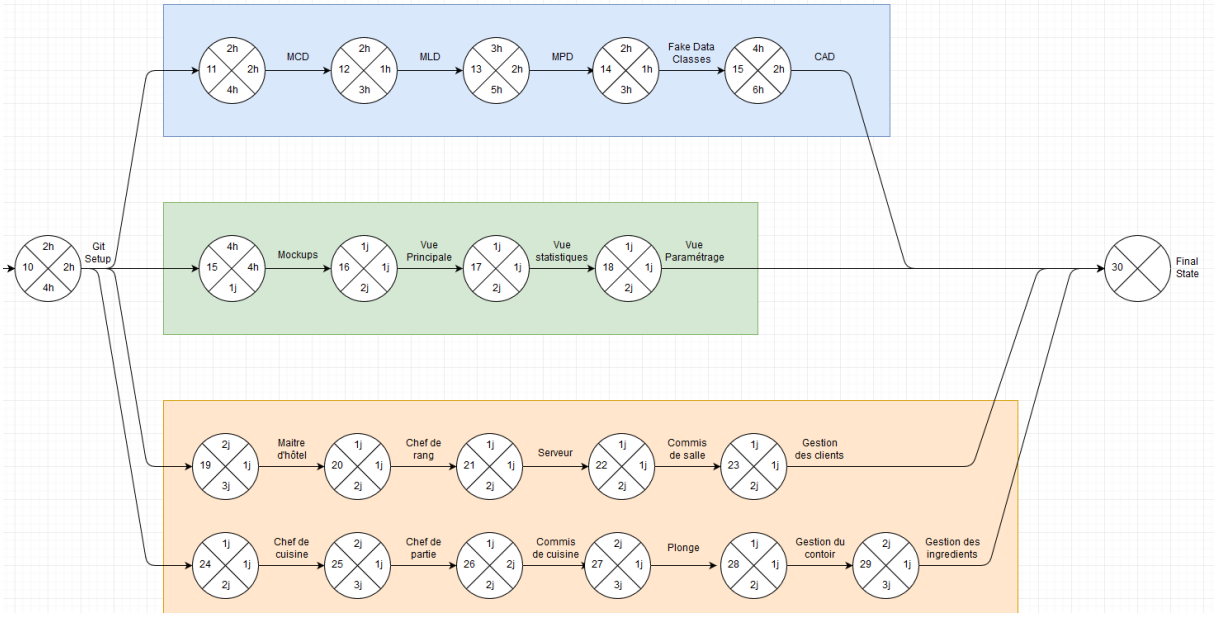
Pour une meilleure lisibilité, le PERT à été découpée en deux phases :

- Conception & Gestion de projet
- Déploiement



PERT – Conception & Gestion de projet

Ce PERT nous indique les éléments qui ont été constitués durant les premiers jours. La réalisation se compte en heure, bien évidement ces éléments étaient prévisionnels, et le diagramme de classe a pris bien plus de temps que prévu.



PERT – Déploiement

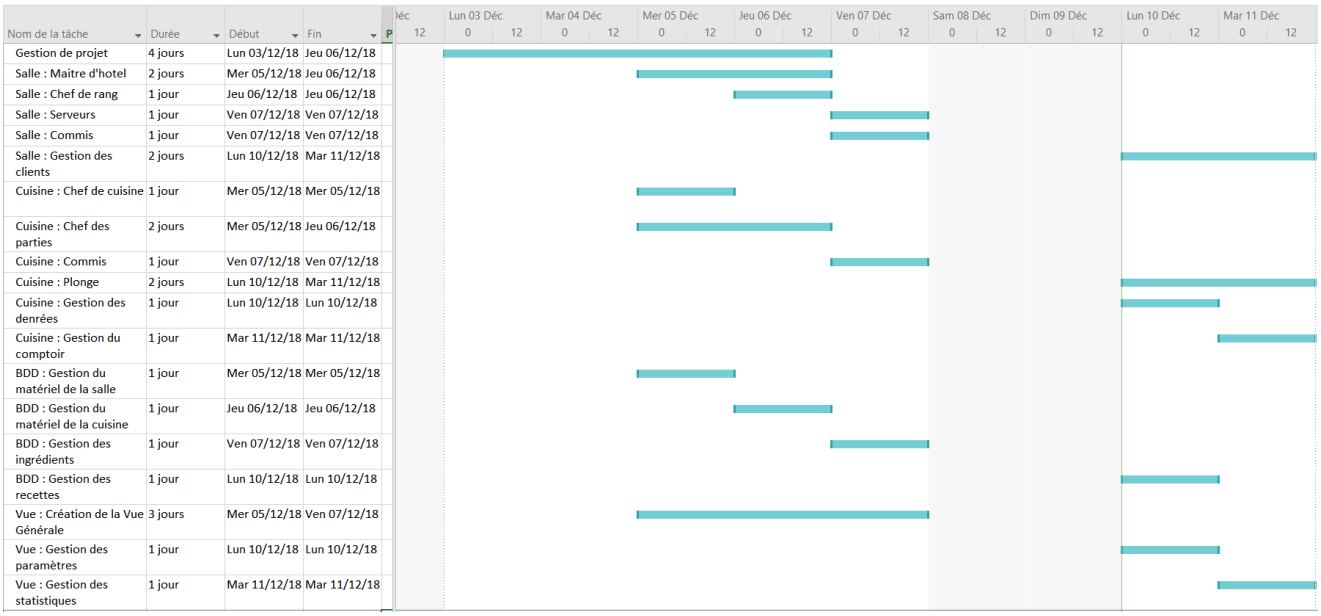
La seconde partie du PERT indique le temps prévu pour chaque tâche (technique), il est découpé en trois lignes, représentant les trois vecteurs du projet : La CAD (Bleu), Les vues (Vert), et les contrôleurs et modèles (Jaune). Ces éléments seront détaillés dans la suite de ce rapport.

GANTT

Dossier d'architecture

Voici le schéma GANTT qui représente la liste des tâches à effectuer pour le projet ainsi que de leurs estimations dans le temps. Le Gant est encore une fois prévisionnel, et se concentre sur l'implémentation des différents éléments sur le plan technique mais aussi la gestion de projet.

La première phase était la gestion de projet (commune) pour s'organiser, puis viens ensuite deux jours après le début des premières implémentations (en vue de la longueur du sujet, qui s'est annoncée très hâtive).



Nom de la tâche	Durée	Début	Fin
Gestion de projet	4 jours	Lun 03/12/18	Jeu 06/12/18
Salle : Maitre d'hotel	2 jours	Mer 05/12/18	Jeu 06/12/18
Salle : Chef de rang	1 jour	Jeu 06/12/18	Jeu 06/12/18
Salle : Serveurs	1 jour	Ven 07/12/18	Ven 07/12/18
Salle : Commis	1 jour	Ven 07/12/18	Ven 07/12/18
Salle : Gestion des clients	2 jours	Lun 10/12/18	Mar 11/12/18
Cuisine : Chef de cuisine	1 jour	Mer 05/12/18	Mer 05/12/18
Cuisine : Chef des parties	2 jours	Mer 05/12/18	Jeu 06/12/18
Cuisine : Commis	1 jour	Ven 07/12/18	Ven 07/12/18
Cuisine : Plonge	2 jours	Lun 10/12/18	Mar 11/12/18
Cuisine : Gestion des denrées	1 jour	Lun 10/12/18	Lun 10/12/18
Cuisine : Gestion du comptoir	1 jour	Mar 11/12/18	Mar 11/12/18
BDD : Gestion du matériel de la salle	1 jour	Mer 05/12/18	Mer 05/12/18
BDD : Gestion du matériel de la cuisine	1 jour	Jeu 06/12/18	Jeu 06/12/18
BDD : Gestion des ingrédients	1 jour	Ven 07/12/18	Ven 07/12/18
BDD : Gestion des recettes	1 jour	Lun 10/12/18	Lun 10/12/18
Vue : Création de la Vue Générale	3 jours	Mer 05/12/18	Ven 07/12/18
Vue : Gestion des paramètres	1 jour	Lun 10/12/18	Lun 10/12/18
Vue : Gestion des statistiques	1 jour	Mar 11/12/18	Mar 11/12/18

GANTT

Nb : Vous trouverez également dans les Outils (dernière page) différents liens, dont le **Trello**, qui permet de suivre l'avancement des tâches de chacun en temps réel.

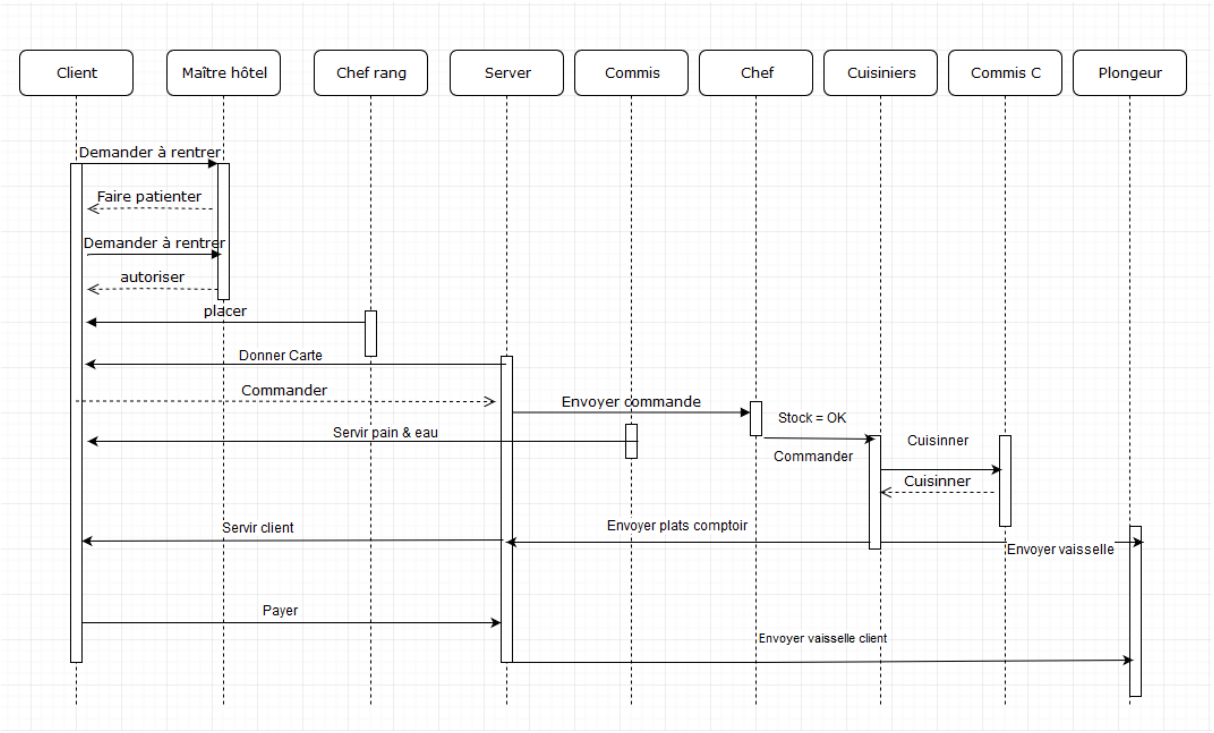
2. ARCHITECTURE

DEFINITION DE L'ARCHITECTURE

DIAGRAMME DE SEQUENCE

Ces diagrammes vont nous permettre de visualiser les scénarios pour des cas spécifiques. Nous vous en présentons trois. Ceux-ci ne rentrent pas encore dans la technique, mais permettent de visualiser la trame de différents évènements.

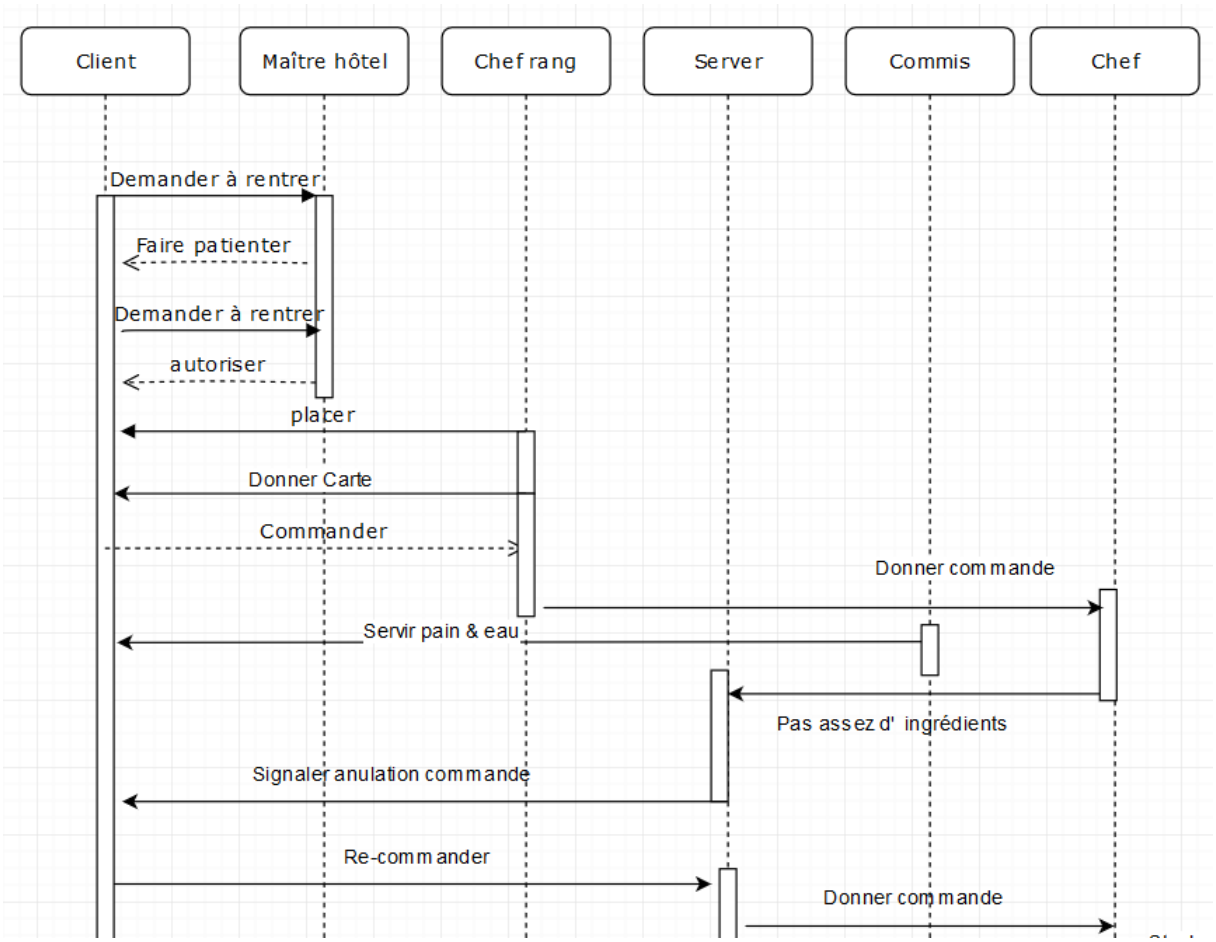
Premier cas : « La totalité du processus »



Séquence 1 – Cheminement complet

Dans ce diagramme, nous pouvons voir les processus qui seront présent (les acteurs), et ce qu'ils échangent entre eux. Cela débute par un client qui demande à rentrer, et se termine par la paye de celui-ci.

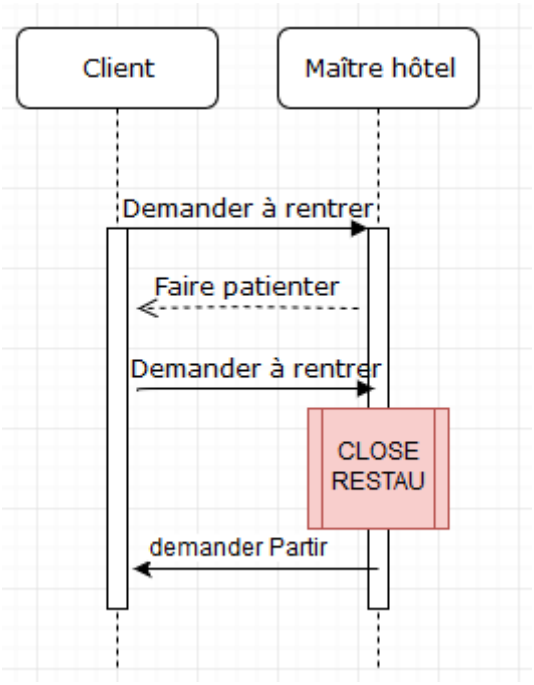
Second cas : « L'indisponibilité d'ingrédients pour le plat »



Séquence 2 – NotEnoughIngredients

Cas similaire au premier, à l'exception que le chef qui vérifie le stock détecte qu'il n'a pas assez d'ingrédients. Il est donc dans la nécessité de signaler l'annulation au serveur, et lui-même va demander au client de recommander.

Troisième cas : « Fermeture du restaurant »



Séquence – Fermeture du restaurant

Troisième et dernier cas que nous présentons, il y' a en effet le système d'ouverture et fermeture du restaurant à prendre en compte, via une horloge. C'est ce diagramme qui nous a d'ailleurs fais penser à créer une horloge par la suite.

DIAGRAMME DE COMPOSANT

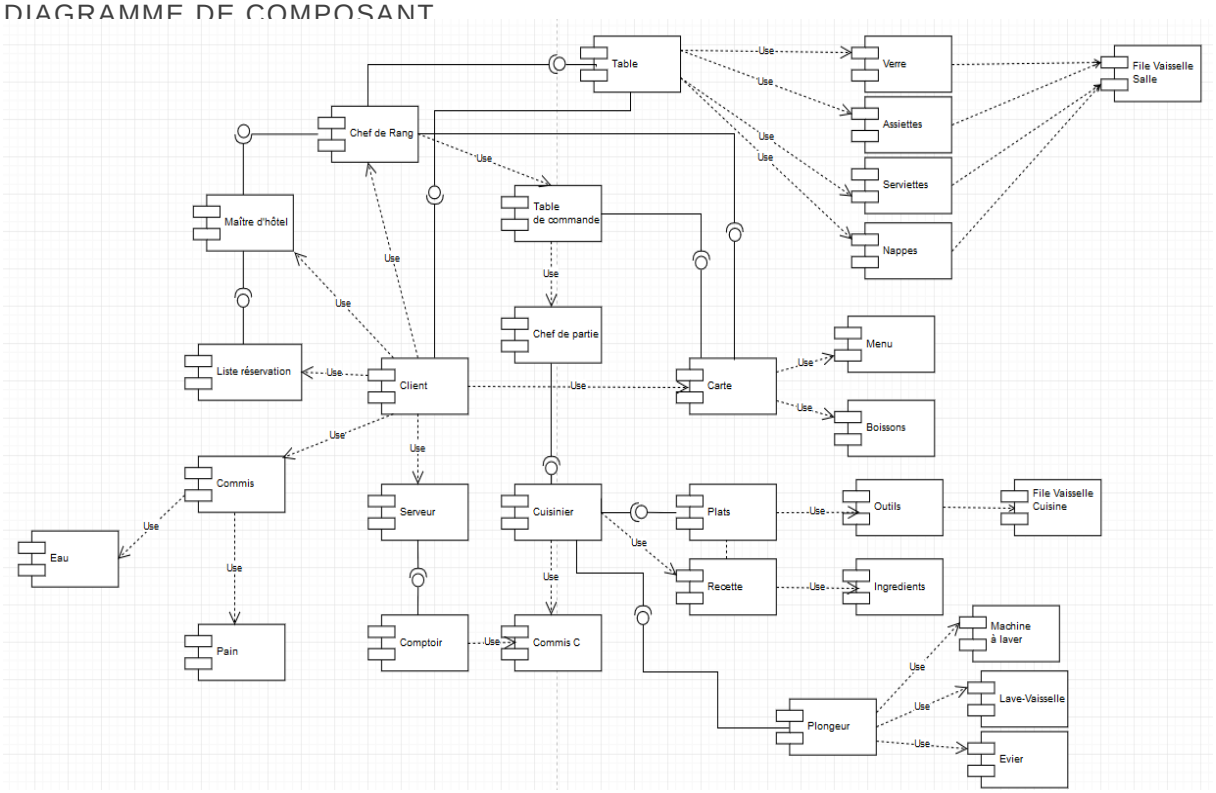


Diagramme de composant

Le diagramme de composant identifie tous les objets et leurs liaisons entre elles. On retrouve donc qui va utiliser quoi, et qui est utilisé par qui. On remarque que le client est au centre de nombreux composants, et que les composants tiers « pain, eau »{…} sont en bout de chaîne.

Dossier d'architecture

DIAGRAMME DE CLASSE

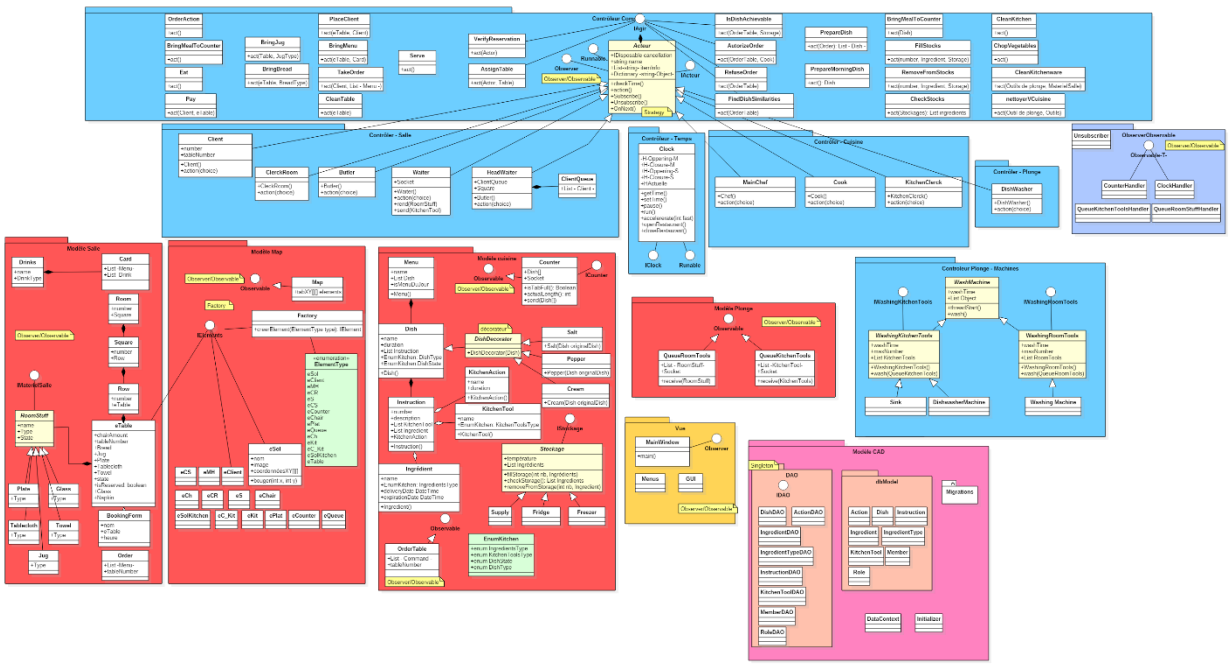


Diagramme classe – Globalité

***/ ! *** : *Il est conseillé d'enregistrer le diagramme sur son PC (à récupérer sur le git par exemple) pour pouvoir zoomer et voir les élements de près.*

L'architecture globale est en MVC (Modèle vue contrôleur).

En rouge se trouve le modèle. (Les données à manipuler). Il est décomposé entre la salle, la vue, la cuisine et la plonge. On retrouve le D.P « Factory », et de nombreux élements surveillé par « Observer/Observable ».

En jaune se trouve la vue. (Les données à afficher), celle-ci sera un observer, qui devra se mettre à jour à chaque fois que certaines données du modèle sont mises à jour.

En bleu se trouve le contrôleur. (Ce qui va manipuler les données). En haut, se trouve un « Strategy », avec des fonctions communes pour tous les acteurs. Les acteurs sont des observer, mais aussi des threads. On retrouve au centre une « Clock » qui sera le moteur des contrôleurs. A chaque action, il faudra regarder un certain nombre de fois la clock pour effectuer l'action.

Nous avons également des aspects techniques :

En rose la CAD (couche d'accès aux données), qui va accéder aux données, mais également fournir un mappage entre les données de la BDD et notre modèle. (Appelé « ORM »). On retrouve un « Syngleton » dessus.

En bleu pale se trouve tous des élements de l'Observer/Observable ; nécessité en C#. Ce sont des « Handler » qui seront responsable de surveiller des élements de notre modèle, et de remonter l'information vers des contrôleurs, comme par exemple les acteurs. C'est sur les acteurs qu'on effectue ensuite le traîtement, une fois l'information reçue.

En vert les énumérations, qui vont servir à gérer les états de nombreux composants.

En jaune pale les classes abstraites.

Récapitulatif et intérêt des Design patterns :

Dans ce projet nous avons dû utiliser 6 designs patterns :

- **Stratégy** : Il permet de changer un comportement pour un élément. Dans notre cas, n'importe quel acteur pourra adopter le comportement de n'importe quel autre acteur si on le souhaite, et ceci, y fournir sa propre manière de le traiter.
- **Décorator** : Il permet d'attacher dynamiquement de nouvelles responsabilités à un objet. Dans notre cas il nous permet de pouvoir construire des plats (entrée, plat ou dessert) en attachant de nouveaux aliments. Nous nous en servons pour rajouter du sel, du poivre, ou de la crème sur nos plats ; ceci modifiant la description initiale du plat à chaque ajout.
- **MVC** : Il permet dans notre cas de construire proprement notre application :
    - Le contrôleur va gérer modifier les données du modèle.
    - Le modèle va gérer les données ainsi que de la logique en rapport avec les données.
    - La vue va permettre à l'utilisateur d'interagir et d'avoir un retour du programme.
- **Factory** : La factory va nous permettre d'instancier les objets du restaurant en passant par un seul objet. C'est comme un peu comme une usine qui fait sortir les élements qu'on lui dit de bâtir.
- **Singleton** : Le singleton permet d'instancier une classe si elle n'existe pas et de la retourner si elle existe déjà. On s'en sert pour la CAD. Elle garantit également l'unicité de l'objet. (Pas de double instance), ce qui porte tout l'intérêt de le mettre ici.
- **Observable / Observeur** : Ce Design Pattern permet d'envoyer des notifications entre des modules observer et des modules observables. Les observables sont observés par des « handler » (en c#), et lors d'une modification, les « handler » envoie les notifications vers les observer qui se seront au préalable enregistré auprès d'eux.

3. ANNEXES

OUTILS

VERSIONNING

Github : https://github.com/Emilien-Pons/Projet-Restaurant

GESTION DES EXIGENCES

Trello : https://trello.com/b/Wx6FJaTq/projet-programmation-syst%C3%A8me

PLANNIFICATION

MS : Project 2018