

RAPPORT CONCEPTION ARCHITECTURE

PROJET PROGRAMMATION SYSTEME 05/12/2018

Tom BRUNETTI | Alexandra DESWAEME | Sofack NANA | Victor SORRENTI | Guillaume THIBAULT

TABLE DES MATIÈRES

Contenu

PROJET PROGRAMMATION SYSTEME _____ 1














Design Patterns utilisés _____ 2

PROJET PROGRAMMATION SYSTEME

CONTEXTE DU PROJET

Une grande chaîne internationale de restaurants souhaite s'équiper d'une nouvelle application informatique pour améliorer l'accueil du public, le remplissage des salles, la gestion des réservations et l'organisation du travail en cuisine.

RAPPEL DE LA DEMANDE DU PROJET

-  Diagrammes de composants et classes de l'application.
-  Diagramme de séquence.
-  Au moins un IPC par type (synchro, échange de données ou les deux) doit être utilisé. Les threads seront indispensables également.
-  Utilisation des pools.
-  Utilisation des sockets pour les échanges entre la salle de resrauration et la cuisine.
-  Utilisation du langage C#.NET.
-  Base de données de stocks en SQL Serveur, actualisée avec les livraisons et mise à jour en temps-réel en fonction des commandes.
-  Utilisation des Design Pattern (au moins 5 au choix – Observer, Strategy, Builder , Factory, Decorator, Singleton,... - en plus du MVC qui est obligatoire).
-  Utilisation de Git et du TDD pendant tout le développement du projet.
-  **(BONUS 1)** Chaque tâche de chaque processus ou thread doit être horodaté dans un log (en BDD ou en fichier log unique).
-  **(BONUS 2)** Utilisation de 2 machines Windows distinctes : une pour la cuisine et une pour la salle de restauration .
-  **(BONUS 3)** Utilisation de 2 machines distinctes pour la cuisine et la salle de restauration : une sur Windows avec .NET et l'autre sur Linux en Java.
-  **(BONUS 4)** Possibilité de sauvegarde une situation déterminée dans un service et de pouvoir la « **rejouer** » plus tard. Cela peut être très utile en complément du mode **PAUSE**.

Design Patterns utilisés

Les Design Patterns représentent chacun un ensemble de bonnes pratiques diffusées sous forme de schéma de classes pour la résolution d'un ou de plusieurs problèmes de conception. Il existe plusieurs design patterns tous aussi importants les uns que les autres.

Nous avons choisis d'utiliser dans notre conception de diagramme de classes cinq design patterns différents en plus du design pattern MVC.

1. MVC

MVC signifie « Modèle Vue Contrôleur ».

La partie Modèle gère les interactions entre l'application et la base de données.

La partie Vue se concentre sur l'affichage. Elle ne fait aucun calcul et se contente de récupérer les données pour savoir ce qu'elle doit afficher.

La partie Contrôleur gère le traitement des données. C'est l'intermédiaire entre le Modèle et la Vue, le Contrôleur va demander au Modèle les données, les traiter et renvoyer les informations à afficher à la Vue.

Ce design pattern nous permet de structurer une conception claire et efficace, grâce à la séparation des données de la Vue et du Contrôleur, un gain de temps de maintenance et d'évolution de l'application, ainsi qu'une plus grande souplesse pour organiser le développement du programme entre différents développeurs.

2. Observer

Ce design pattern permet au personnels de salle de réagir aux différents états du client.

L'objet observé (client) pourra notifier aux observateurs (le personnels de salle) un changement d'état, ainsi les observateurs pourront agir en conséquence.

3. Strategy

Le design pattern strategy est un patron de conception de type comportemental grâce auquel des algorithmes peuvent être sélectionnés à la volée au cours du temps d'exécution selon certaines conditions.

DESIGN PATTERNS UTILISES

Dans notre cas, ce design pattern est utilisé afin de gérer les différents types de comportements des clients, comme par exemple, un client qui est pressé (doit manger en maximum 30 minutes), un client normal et un client « cool » qui est disposé à prendre son temps pour manger.

4. Builder

Builder est un patron de conception utilisé pour la création d'une variété d'objets complexes à partir d'un objet source. Dans notre cas, nous allons utiliser ce design pattern afin de préparer un plat pour le client.

5. Mediator

Mediator a pour but d'encapsuler dans un objet, le médiateur, les interactions entre les différents objets d'un sous-système. Il est utilisé pour réduire le couplage entre plusieurs classes et permettre ainsi une réutilisation directe de ces classes indépendamment de leurs interactions (sans héritage). Nous avons utilisé ce design pattern pour gérer les différents ustensiles présents dans le restaurant (petits et gros ustensiles de cuisine, fourchettes etc...).

6. Singleton

Ce design pattern garantit une classe ne possède qu'une seule instance, et fournit un seul point d'accès à celle-ci. Dans notre cas, nous avons utilisé ce design pattern afin d'avoir qu'une seule instance pour certains personnels dont : maître de rang et chef de cuisine.