# RAPPORT
# SUR LES DP UTILISES


**MEMBRES DU GROUPE**

**BAYA Monera**

**MOMBOULI Ben Paossi**

**NKOUKA Sorel**

**TCHUENTE Nadia**

# TABLE DES MATIERES

# 1. Design Patterns Utilisés

## 1.1. Factory Method

**Salle de restauration : IngredientFactory**

Description :

La classe IngredientFactory implémente le pattern Factory Method, permettant de créer des objets Ingredient de manière centralisée. Cela offre plusieurs avantages :

Encapsulation de la logique de création : Si la manière de créer un Ingredient devait changer (par exemple, en fonction de certains critères ou en utilisant des sous-classes), cette logique serait contenue dans la factory.

## 1.2. MVC (Model-View-Controller)

Type : Architecture globale

Implémentation spécifique :

Modèle : La classe Ingredient ainsi que d'autres représentent les modèles des données (attributs, getters, setters).

Vue : Les classes comme mainWindow, controlDialog, stockWindow agissent en tant que vue, fournissant une interface utilisateur avec des champs pour saisir des informations.

Contrôleur : Les méthodes de stockWindow : onCommandButtonClicked) sert de contrôleur, manipulant le modèle et mettant à jour la vue en réponse aux événements.

Explications :

Le pattern MVC permet de séparer les préoccupations :

- Le modèle contient les données brutes et la logique métier.
- La vue affiche ces données et recueille les entrées utilisateur.
- Le contrôleur agit comme un intermédiaire qui lie la vue et le modèle.

## 1.3. Singleton

**Utilisation de QSqlDatabase**

Description :

Dans Qt, l'accès à la base de données se fait généralement via une instance unique gérée par QSqlDatabase :addDatabase et QSqlDatabase::database.

Cette approche suit le pattern Singleton, qui garantit qu'une seule instance d'une ressource partagée (la connexion à la base de données, ici) est utilisée dans l'application.

Avantages :

- Évite les collisions ou les problèmes liés à l'ouverture multiple de connexions.
- Simplifie la gestion des ressources partagées.

## 1.4. DAO (Data Access Object)

**Gestion de la base de données (CommandeManager, Ingredient)**

Description : Les classes et méthodes encapsulent les interactions avec la base de données (comme getAllIngredients, insertIngredientIntoDatabase) et centralisent les opérations de persistance et d'accès aux données.

Avantages :

- Encapsulation des détails techniques : La logique SQL est isolée des autres parties de l'application.
- Réutilisabilité : Les méthodes DAO sont réutilisées dans le programme.

## 1.5. Composite

**Gestion des commandes (CommandeManager, OrderItem)**

Description : Le programme utilise les structures de données comme **OrderItem** pour la composition des commande.

## I.6. Signaux et Slots comme Observer

**Système de signaux et slots dans Qt**

Description du pattern : Le pattern Observer consiste à maintenir une liste d'observateurs (ou écouteurs) qui sont notifiés automatiquement lorsqu'un état change.

Dans Qt, les signaux sont émis par un objet lorsqu'un événement se produit (par exemple, un clic sur un bouton dans le programme). Les slots sont des méthodes qui réagissent à ces signaux. Dans Qt on connecte les deux dynamiquement pour faire le lien événement ensuite mise à jour de l'interface.

Cas concret dans le programme :

Signal : Le bouton "Commander" émet un signal lorsqu'il est cliqué.

Observateur : La méthode onCommandButtonClicked agit comme un observateur, répondant à cet événement en exécutant la logique métier.

Fonctionnement :

Signal : Émis lorsqu'un événement spécifique se produit.

Slot : Réagit au signal, exécutant une logique appropriée.

Connexion : L'association entre le signal et le slot est établie dynamiquement via QObject::connect.

Pourquoi c'est un Observer ?

Propagation des événements :

Les slots sont automatiquement notifiés lorsqu'un signal est émis. Cela reflète directement le comportement attendu dans le pattern Observer.

Dynamisme :

Les connexions peuvent être établies ou supprimées à l'exécution, permettant une grande flexibilité dans la gestion des événements.

Cas concret dans le projet :

Le clic sur un bouton ("Commander") émet un signal. Ce signal est observé par le slot onCommandButtonClicked, qui exécute la logique pour insérer une commande.

## Conclusion

Le projet utilise les DP suivants :

- Factory Method pour la création d'objets (IngredientFactory).
- MVC pour structurer l'architecture.
- Singleton pour la gestion des connexions à la base de données.
- Observer pour la gestion des événements via les signaux et slots.
- DAO pour encapsuler les opérations liées à la base de données.

## Sources et références bibliographiques

Robert, J. (2024, 25 novembre). Data Access Object (DAO) : Qu'est-ce que c'est ? À quoi ça sert ? Formation Data Science | DataScientest.com. https://datascientest.com/data-access-object-tout-savoir#:~:text=Les%20Data%20Access%20Object%20ou%20DAO%20est%20un,il%20reste%20pertinent%20dans%20les%20architectures%20modernes%20%21

Cyrille, H. (s. d.). Le pattern DAO. Developpez.com. https://cyrille-herby.developpez.com/tutoriels/java/mapper-sa-base-donnees-avec-pattern-dao/

Programmation Qt/Signaux et slots — Wikilivres. (s. d.). https://fr.wikibooks.org/wiki/Programmation_Qt/Signaux_et_slots

Signals & Slots | QT Core 6.8.1. (s. d.). https://doc.qt.io/qt-6/signalsandslots.html