Rapport

sur l'évolution du projet bancaire

🔷 Approche de conception progressive

Le projet a été structuré de manière incrémentale pour illustrer l'apport de plusieurs design patterns fondamentaux en génie logiciel.

Étape	Pattern	Objectif principal
1	Version de base	Implémentation simple pour établir les fondations
2	Singleton	Garantir une instance unique du compte
3	Observateur	Découpler le modèle des vues
4	MVC (Modèle-Vue-Contrôleur)	Séparation claire des responsabilités

Cette démarche progressive démontre comment chaque pattern améliore la structure, la maintenabilité et l'extensibilité du système.

🔷 Contribution de chaque pattern

1. Version initiale

- **Approche** : Implémentation directe pour définir les bases fonctionnelles.
- **Apport principal** : Simplicité et rapidité de développement.
- **Contrainte** : Fort couplage entre la logique métier et l'interface, limitant l'évolutivité.

2. Singleton

- **Approche** : Centralisation de l'état via une instance unique de CompteSingleton.
- **Apport principal** : Cohérence et intégrité des données financières.
- **Points forts** :

1. Évite les incohérences entre multiples instances
2. Centralise la gestion de l'état global

Considération : Réduit la flexibilité pour la gestion simultanée de plusieurs comptes

3. Observateur

- **Approche** : Mécanisme de notification automatique des vues (Observable).
- **Apport principal** : Découplage entre modèle et observateurs.
- **Avantages techniques** :

1. Extensible (ajout/suppression d'observateurs sans modifier le modèle).
2. Architecture réactive à faible couplage.

- **Considération** : Complexité accrue et gestion nécessaire des cascades de notifications.

4. MVC

- **Approche** : Séparation architecturale en trois couches distinctes.
- **Apport principal** : Organisation structurée et maintenabilité accrue.
- **Rôles et responsabilités** :

Composant	Responsabilité	Avantage
Modèle	Gestion des données et logique métier	Pureté fonctionnelle
Vue	Affichage et interactions utilisateur	Interface isolée
Contrôleur	Coordination Modèle-Vue	Séparation des préoccupations

- **Considération :** Structure plus élaborée, justifiée pour des applications évolutives

🔷 Avantages architecturaux obtenus

▢ Organisation optimisée

Chaque pattern introduit une structuration plus adaptée et professionnelle.

▢ Extensibilité renforcée

- Ajout de fonctionnalités sans impact sur le noyau métier
- Intégration simplifiée d'observateurs, vues et contrôleurs

▢ Réutilisabilité

Architecture MVC permettant la réutilisation du modèle avec différentes interfaces :

- Application console
- Interface graphique
- Solution web

▢ Robustesse systémique

Pattern	Contribution à la robustesse
Singleton	Cohérence des données
Observateur	Réactivité garantie
MVC	Maintenabilité à long terme

▢ Progression pédagogique

Chaque étape démontre une amélioration mesurable par rapport à la version initiale, validant la valeur des patterns dans un contexte réel.

▢ Synthèse et perspectives

L'évolution architecturale du projet bancaire valide l'importance stratégique des design patterns :

Pattern	Valeur ajoutée	Impact
Singleton	Cohérence systémique	Intégrité des données
Observateur	Flexibilité et découplage	Architecture réactive
MVC	Organisation et maintenabilité	Qualité logicielle

- **Conclusion** : Cette progression démontre qu'une approche pattern-driven transforme une implémentation basique en une architecture professionnelle, scalable et maintenable, tout en servant de référence pédagogique pour l'application des bonnes pratiques en génie logiciel.