Prosit Aller - Restauration

Infos et rôles

Date : 31/01/2025
Début : 11h04
Fin : 11h44
Durée : 40 minutes

Tuteur : Thibault
Animateur : Mathéo
Secrétaire : Lucas
Scribe : Clément
Gestionnaire : Ludovic

Mots-clés

- Régression logiciel
- ASP.NET
- MVC/MVVM
- Évolutivité
- Migrer
- Modularité
- Logique Métier
- .NET 6.0
- Design Pattern avancé
- Couche de présentation

- Maintenabilité
- C#
- Gestion de versions
- POO Avancé
- UML
- Principes SOLID
- Communication technique

Contexte

2F Roaming a résolu des problèmes techniques mais fait face à des coûts élevés et une architecture rigide. (Ludovic)

Ils souhaitent redéfinir l'architecture de l'application. (Jérémy)

Problématique

Comment séparer les différents couches d'une application tout en garantissant une maintenabilité et la ré usabilité.

Contraintes

- Imposition des Frameworks modernes
- Évolutivité/Maintenabilité
- Réduction des coûts au maximum
- POO Avancé

Livrable(s)

- UML
- Code (Design Pattern)
- Plan formation

Généralisation

- Modèle d'architecture d'application

Hypothèses

- MVC —> Model View Controller ?
- MVVM —> Model View View Model ?

Permettrai d'architecturer le code et la communication entre les controller (cerveau), les model et la view (affichage) ?

L'intérêt est de sectoriser le code ?

Pourquoi deux méthodes différents :

- 1 pour la technique et 1 pour le client ?
- 1 plus ancien et 1 plus récent ?

- 3 couches :
  - Présentation
  - Logique métier
  - Accès donné

- Principe SOLID pour :
  - La modification ne doit pas entrainer de "casser" les bonnes pratiques.

- ASP .NET quésaco ?
  - Permet de faire du backend et de l'applicatif ?

Pourquoi être sur une ancienne version de .NET ?

- .NET 6.0 présente des avantages par rapport à .NET 8.0
- Problème de fiabilité ? → Une ancienne version est potentiellement travaillé pendant plus longtemps ?
- Versions plus récentes → Plus de sécurité ?
- Pas de rétrocompatibilité ? Nouvelles versions peuvent changer l'utilisation complète de la techno.
- Tant que cela fonctionne on ne change pas ? Devoir former toute une team de Senior qui ont l'habitude d'une certaine techno → Coûts temps migration et formation

---

UML à faire ? Lesquels ?

- Classe ? Si l'archi est à refaire ? Peut-être des bonnes pratiques à respecter qui ne le sont pas
- Séquence ? Nouveau membre dans l'équipe
- USER CASE ? Présentation pour une personne extérieure, pas trop de détail ni trop exhaustif.

Plan d'action

- Tableau comparatif
  - MVC et MVVM
  - .NET 6.0 et .NET 8.0

- Voir la compatibilité de plusieurs technos
  - ASP. NET Core, Blazore, Razore...

- Voir qu'est ce que la POO Avancé
  - Faire un petit projet entre différentes versions de .NET en incluant la POO et le Design Pattern

- UML
  - Faire des exemples de schémas selon les pistes de solutions (Classe, Séquence, USER CASE)

- Plan formation

Remarques :

Mots-clés sont avant tout des mots importants à comprendre plus que des définitions —> utiliser au sein du plan d'actions et des hypothèses pour mieux définir la théorie et/ou la pratique (Remarque Tuteur)

Restructuration du template "Prosit Aller"

Mots-clés → Contexte → Problématique → Contraintes → Livrables → Généralisation → Hypothèses/Pistes de solutions → Plan d'action

(Remarque Tuteur) Changement de version d'applications = Big deal pour une entreprise (Coûts temps migration et formation), des fois devoir refaire une appli entière pour une version de différence. Pas qu'une difficulté technique, difficulté humaine également