

Génie Logiciel : TD 2

Exercice 1 :

Une entreprise souhaite développer une application pour la gestion des achats, des ventes, des fournisseurs et des clients.

- Achats : gestion des commandes fournisseurs et des stocks.
- Ventes : suivi des commandes clients, factures et paiements.
- Fournisseurs : gestion des fiches et des performances.
- Clients : suivi des historiques d'achats et des interactions.

Le projet suit les étapes suivantes :

1. Analyse des besoins des départements concernés.
2. Développement et tests de chaque module (achats, ventes, fournisseurs, clients) de manière indépendante.
3. Intégration progressive des modules et mise en production étape par étape.
4. Améliorations continues basées sur les retours des utilisateurs finaux.

Question :

Quel modèle de cycle de vie serait le plus adapté pour ce projet ? Justifiez votre réponse.

➔ Le modèle de cycle de vie **évolutif**.

- Le projet est divisé en plusieurs parties (achats, ventes, fournisseurs, clients).
- **Chaque partie est développée et testée indépendamment, puis intégrée une à une.**
- Cela correspond à une approche évolutive, où chaque étape ajoute de nouvelles fonctionnalités ou améliorations.
- **Les retours des utilisateurs sont pris en compte pour améliorer le système en continu.**
- Ce modèle permet de livrer les fonctionnalités progressivement tout **en s'adaptant aux besoins des utilisateurs.**

Exercice 2 :

Une université souhaite développer un logiciel de gestion intégrée pour centraliser ses processus administratifs et académiques. Ce logiciel devra inclure les fonctionnalités suivantes :

- Étudiants : gestion des inscriptions, des fiches personnelles, et des parcours académiques.
- Enseignants : gestion des fiches des enseignants, de leurs cours et de leurs emplois du temps.
- Notes : saisie et consultation des notes par cours et par semestre, calcul automatique des moyennes et des classements.
- Emplois du temps : création et gestion des emplois du temps pour les étudiants et les enseignants.

Le projet suit les étapes suivantes :

- Analyse initiale : Collecte des besoins auprès des responsables académiques, des enseignants, et des étudiants.
- Développement par modules : o Le module de gestion des étudiants est développé et testé en premier. o Ensuite, les modules enseignants, notes, et emplois du temps sont développés progressivement.
- Intégration : Les modules sont reliés entre eux pour assurer une gestion centralisée.
- Déploiement pilote : Le logiciel est testé dans une faculté pour ajuster les fonctionnalités.
- Déploiement complet : Le logiciel est mis en œuvre dans toute l'université avec des sessions de formation pour les utilisateurs.

Question : Dans le cadre de ce projet, quelle méthode agile serait la plus appropriée : XP (Programmation Extrême) ou FDD (Développement Dirigé par les Fonctionnalités) ? Justifiez votre choix

1- La méthode agile **FDD**

2-

- Le projet est organisé autour de plusieurs fonctionnalités bien définies.
- La méthode FDD permet de développer les modules un par un, de manière structurée.
- Elle garantit une bonne organisation et une intégration progressive des fonctionnalités.
- FDD est idéale pour un projet de grande envergure comme celui-ci, où la cohérence entre les modules est essentielle.

Exercice 3 :

Une entreprise de commerce électronique souhaite développer un logiciel de gestion des achats en ligne pour son site web. Ce logiciel devra permettre aux clients de rechercher, ajouter et acheter des produits, tout en offrant aux administrateurs la possibilité de gérer les stocks, les commandes et les paiements. Les principales fonctionnalités attendues sont les suivantes :

- Gestion des produits : ajout, modification, suppression des produits, suivi des stocks.
- Gestion des commandes : création, suivi et traitement des commandes clients.
- Gestion des paiements : validation et traitement des paiements sécurisés.
- Gestion des utilisateurs : création et gestion des comptes clients (inscription, connexion, historique des commandes).
- Interface administrateur : gestion des produits, des commandes et des rapports de vente.

Un Client peut passer plusieurs commandes, et chaque commande peut contenir plusieurs produits. De plus, chaque produit peut être commandé plusieurs fois par différents clients. En ce qui concerne le paiement, chaque commande est associée à un paiement spécifique, chaque commande nécessitant un paiement unique pour sa validation. Quant à l'administrateur, il gère les produits et les commandes au sein du système, et il a également la possibilité de générer des rapports sur les ventes et les stocks.

Questions :

1. Proposez et justifiez le modèle de développement le plus approprié pour ce logiciel. Expliquez pourquoi ce modèle convient mieux dans ce cas spécifique.
2. Identifiez les étapes principales pour développer ce logiciel, en respectant le modèle de développement choisi.
3. Modélisez les entités principales du système et leurs relations sous forme d'un diagramme de classe **UML**.

1- Le modèle de développement le plus approprié pour ce logiciel est le **modèle FDD**

- Justification :

- Le projet est structuré autour de plusieurs fonctionnalités bien définies, telles que la gestion des produits, des commandes, des paiements, et des utilisateurs.
- Le projet est concentré sur le développement progressif par fonctionnalités, ce qui permet de livrer chaque partie du logiciel de manière incrémentale, tout en maintenant une organisation claire.
- Ce modèle est particulièrement adapté à des projets de grande envergure comme celui-ci, où les fonctionnalités doivent être intégrées progressivement pour assurer leur cohérence et leur bon fonctionnement.

2-

• **Analyse des besoins et définition des fonctionnalités**

- Identifier les fonctionnalités principales : gestion des produits, des commandes, des paiements, des utilisateurs, et l'interface administrateur.
- Comprendre les relations entre les entités :
 - Un client peut passer plusieurs commandes.

- Une commande contient plusieurs produits.
- Chaque commande est liée à un paiement unique.
- Préciser les attentes pour chaque fonctionnalité, comme le suivi des stocks ou la validation des paiements sécurisés.
- **Conception générale du système**
 - Définir l'architecture du logiciel, y compris la séparation entre les modules pour chaque fonctionnalité.
 - Prévoir l'interaction entre les modules, comme la liaison entre les commandes et les paiements, ou entre les stocks et les produits.
- **Développement des fonctionnalités par module**
 - Commencer par des modules prioritaires, comme la gestion des produits ou des utilisateurs.
 - Développer chaque module en incluant ses détails spécifiques (exemple : le suivi des stocks dans la gestion des produits).
- **Intégration progressive**
 - Relier les modules pour assurer leur cohérence. (s'assurer que la gestion des paiements s'intègre correctement avec la gestion des commandes.)
 - Tester chaque nouvelle intégration pour identifier et corriger les éventuels problèmes.
- **Validation et ajustements**
 - Effectuer des tests sur le logiciel pour s'assurer que toutes les fonctionnalités fonctionnent correctement.
 - Vérifier les scénarios spécifiques, comme la génération de rapports par l'administrateur ou la consultation de l'historique des commandes par un client.
- **Déploiement final et suivi**
 - Mettre le logiciel en production et fournir une formation pour les administrateurs.
 - Recueillir les retours des utilisateurs pour apporter des améliorations si nécessaire.

3- Diagramme de Classe

