Guide complet du projet Java

Ce document représente le guide complet du projet java pour le but de concevoir et développer une application desktop en Java destiné à un client spécifique. Ledit guide met l'accent sur les points essentiels suivants :

- Objectifs de ce projet
- Les thèmes du projet
- Phases de réalisation du projet
- Planification et déroulement du projet
- Le cahier de charges fonctionnel
- Outil de conception d'interface graphique
- Programmation événementielle
- Persistance de données
- Travail à rendre
- Evaluation du projet et calendrier

A. Objectifs de ce projet :

Ce projet permettra aux étudiants de :

- Maîtriser la chaîne de développement logiciel par l'expérience ce qui lui donne du sens.
- Développer le savoir-faire et de compétences.
- Renforcer l'autoformation et l'autonomie.
- Renforcer les habiletés de communication et de collaboration dans ce projet collectif.
- Vivre des situations de réalisation les plus proches possible de la réalité professionnelle.
- Commettre des erreurs qui le feront progresser, à gérer les imprévus.
- Livrer un produit destiné à des clients spécifiques.

B. Les thèmes du projet Java

Dans le but de viser plusieurs types de clients, six thèmes ont été proposés dans ce projet. Chaque groupe d'étudiants est obligé de choisir un thème parmi les six thèmes suivant (date limite pour faire le choix du thème est **03/03/2024**) :

- Thème n°1 : Gestion d'une poissonnerie
- Thème n°2 : Gestion d'un cabinet d'avocat
- Thème n°3 : Gestion d'une clinique dentaire
- Thème n°4 : Gestion d'une salle de fitness
- Thème n°5 : Gestion d'une parapharmacie

- Thème n°6 : Gestion d'un cabinet de kinésithérapie

C. Etapes de réalisation du projet Java :

Le modèle ci-dessous montre les phases qu'une équipe de développement logiciel suit au cours d'un projet

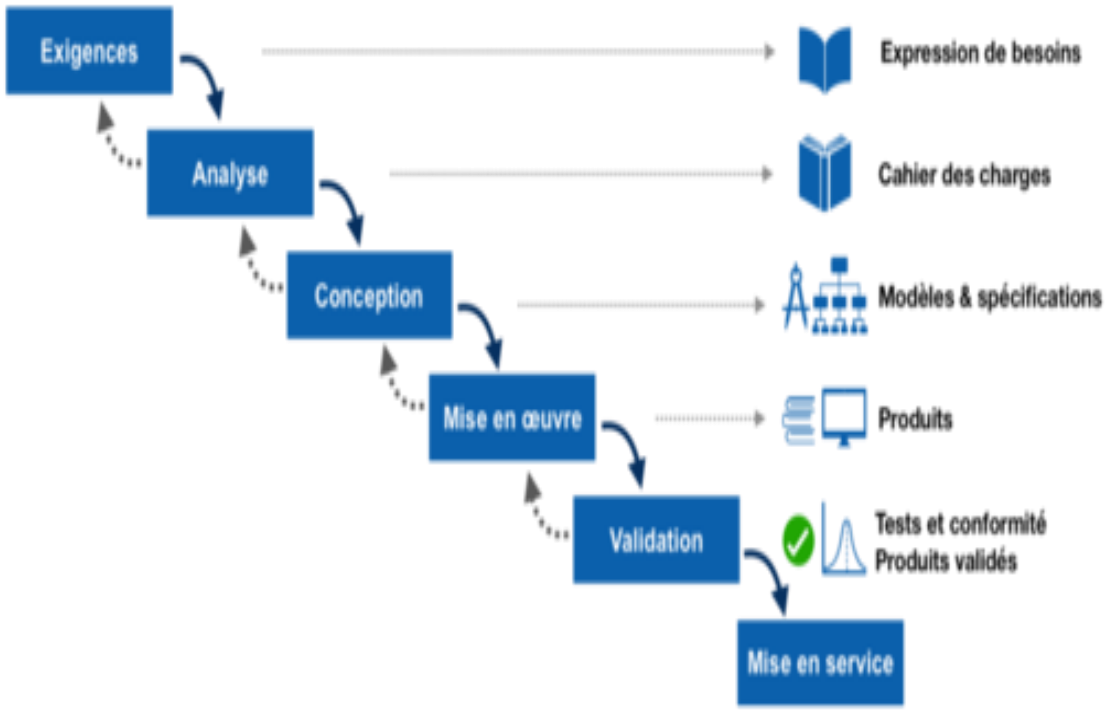

Fig 1 : Modèle de développement en cascade

Le modèle en cascade comprend les phases et les livrables suivants :

1. **Exigences** : les exigences font l'objet d'une expression des besoins. Cette phase vous incite à contacter le client (pharmacien, médecin, restaurateur, …) choisi selon le thème de votre projet pour recueillir ces besoins ;
2. **Analyse** : Ces besoins sont analysés pour établir un cahier des charges fonctionnel ;
3. **Conception** : le produit (Application desktop) est conçu et spécifié de sorte à pouvoir être réalisé ;
4. **Mise en œuvre** : le produit (Application desktop) est réalisé sur la base des spécifications; c'est la traduction dans un langage de programmation (Java dans le cas de ce projet) des fonctionnalités définies lors de phases de conception

5. **Validation** : le produit (Application desktop) est testé et vérifié et sa conformité aux exigences est validée ;
6. **Mise en service** : le produit est installé, les préparatifs pour sa mise en service sont organisés, puis le produit est utilisé.

D. Planification et déroulement du projet

- Les membres du groupe doivent au premier coup contacter à plusieurs reprises le client pour recueillir ses besoins de façon exhaustive et capturer l'image du produit (prototype des différentes interfaces de l'application) qu'il souhaite obtenir après achèvement du projet.
- Après avoir établi le portefeuille qui exprime les besoins du client, les membres du groupe doivent établir un planning pour les prochaines phases afin de mieux gérer le projet.
- Les membres du groupe doivent tous contribuer à la phase de la rédaction du cahier des charges fonctionnel et la phase de conception en planifiant des réunions journalières. Pour la phase de mise en œuvre de l'application (codage), il sera mieux que vous répartissiez les tâches entre vous pour le but d'accélérer la réalisation de l'application.
- Après la réalisation de l'application, vous procédez à la phase de validation du produit et sa mise en service.

E. *Cahier des charges fonctionnel (CDCF) :* spécifications fonctionnelles

Le but d'établir le cahier des charges fonctionnel est de décrivez les besoins d'un client en terme de fonctionnalités. En quelques lignes, expliquez **ce que doit faire** votre application. C'est l'outil de base pour la réalisation.

Par où commencer cette analyse ? Vous pouvez procéder de la sorte :

Partez des fonctions principales et déclinez-les.

Par exemple :

Fonction principale : Enregistrer le contact avec le client

Sous-fonctions :

- créer une nouvelle fiche
- modifier une fiche existante,
- etc.

Autre exemple pour un site internet de remboursement de soins :

Fonction principale : Faire une simulation de remboursement

Sous-fonctions :

- ajouter un acte
- supprimer un acte
- mémoriser sa simulation

Pour chaque fonction, vous pouvez adopter une grille précisant : l'objectif, la description de la fonctionnalité, les contraintes / règles de gestion et le niveau de priorité. Voir l'exemple ci-dessous :

Fonction : enregistrer le contact avec le client / créer une nouvelle fiche	
Objectif	Accéder facilement à une interface de saisie comprenant les informations essentielles à demander
Description	L'ouverture d'une interface est possible à travers un raccourci sur le bureau. Il comprend les informations suivantes : compte client, Nom du client, Nom du contact (...), etc.
Contraintes / règles de gestion	Le service n'est valable que pour les clients, un mode dégradé sera prévu plus tard pour les prospects. La fiche ne peut être fermée que par son créateur, etc.
Niveau de priorité	Priorité haute

La difficulté de cette tâche est de décrire les fonctions attendues précisément pour permettre le travail de réalisation sans entrer dans un niveau détail trop élevé.

Si vous avez en tête une idée du design final, faites une maquette.

L'exercice peut paraître quelque peu rébarbatif, mais il est essentiel pour la compréhension de votre besoin. De plus, il vous oblige à réfléchir sur ce que vous voulez vraiment.

F. *Cahier des charges technique (CDCT)* : spécifications techniques

Le cahier des charges technique (ou CDCT) détermine les moyens pour arriver à un résultat espéré sur le projet. Il spécifie ainsi les contraintes et spécificités techniques nécessaires pour répondre aux besoins fonctionnels exprimés dans le cahier des charges fonctionnel.

Tous les éléments inscrits au CDCT devront être respectés par le maître d'œuvre en charge de la réalisation du projet.

On y liste donc le choix des technologies, les exigences de performance, les critères d'acceptation, les exigences de compatibilité, les caractéristiques techniques à respecter, etc.

G. Outil de conception d'interface graphique : JavaFX Scene Builder

JavaFX Scene Builder est un outil de conception visuelle qui permet aux utilisateurs de concevoir rapidement les interfaces utilisateur de l'application JavaFX, sans avoir besoin d'écrire le code. Vous pouvez faire glisser et déposer (drag and drop) les composants de l'interface d'utilisateur à une zone de

travail. Vous pouvez changer leur nature, appliquer le style et FXML de Layout nouvellement créée automatiquement ci-dessous. Le résultat a été eu est un fichier FXML qui peut être combiné avec un projet Java en reliant l'interface avec la logique de l'application.

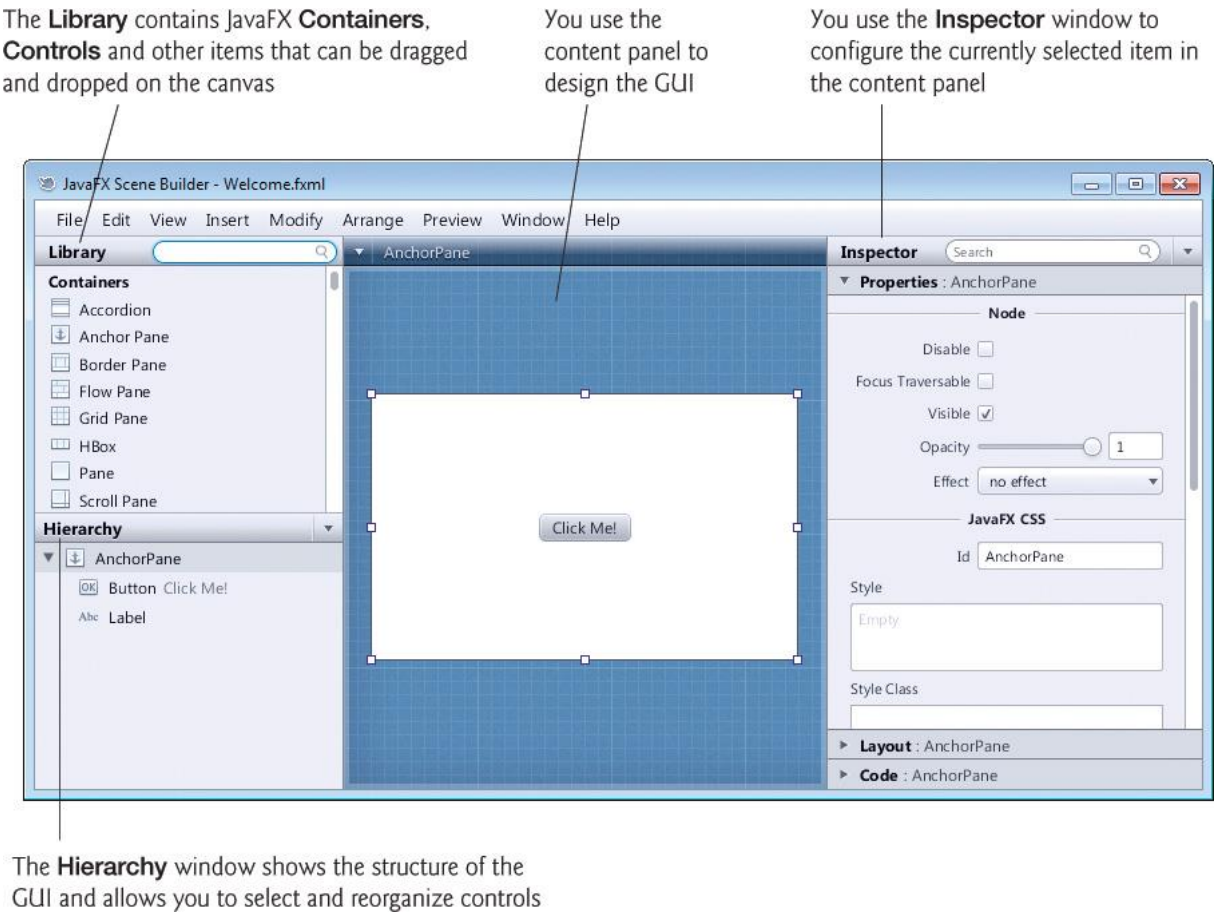

Fig 2: JavaFX Scene Builder displaying the default GUI in Welcome.fxml

Lien d'installation de scene builder : https://gluonhq.com/products/scene-builder/

JavaFX Scene Builder peut être intégré à des IDE tels qu'Eclipse et Intellij Idea.

- Procédure de l'intégration de scene builder dans Intellij Idea (accéder au lien suivant) :

https://www.jetbrains.com/help/idea/opening-fxml-files-in-javafx-scene-builder.html

- Procédure de l'intégration de Scene Builder dans Eclipse (accéder au lien suivant) :

https://docs.oracle.com/javafx/scenebuilder/1/use_java_ides/sb-with-eclipse.htm

H. *Organisez votre code Java à l'aide de l'architecture MVC*

Il existe différentes manières de structurer le code des applications interactives (celles qui comportent une interface utilisateur). Une des architectures, communément utilisée, et qui comporte de nombreuses variantes, est connue sous l'acronyme **MVC** qui signifie **M**odel - **V**iew - **C**ontroller.

Dans cette architecture on divise le code des applications en entités distinctes (***modèles***, ***vues*** et ***contrôleurs***) qui communiquent entre elles au moyen de divers mécanismes (invocation de méthodes, génération et réception d'événements, etc.).

Dans l'architecture MVC, les rôles des trois entités sont les suivants.

- modèle : données (accès et mise à jour)
- vue : interface utilisateur (entrées et sorties)
- contrôleur : gestion des événements et synchronisation

Rôle du modèle

Le modèle contient les données manipulées par le programme. Il assure la gestion de ces données et garantit leur intégrité. Dans le cas typique d'une base de données, c'est le modèle qui la contient.

Le modèle offre des méthodes pour mettre à jour ces données (insertion suppression, changement de valeur). Il offre aussi des méthodes pour récupérer ses données. Dans le cas de données importantes, le modèle peut autoriser plusieurs vues partielles des données. Si par exemple le programme manipule une base de données pour les emplois du temps, le modèle peut avoir des méthodes pour avoir, tous les cours d'une salle, tous les cours d'une personne ou tous les cours d'un groupe de TD.

Rôle de la vue

La vue fait l'interface avec l'utilisateur. Sa première tâche est d'afficher les données qu'elle a récupérées auprès du modèle. Sa seconde tâche est de recevoir tous les actions de l'utilisateur (clic de souris, sélection d'une entrées, boutons, …). Ses différents événements sont envoyés au contrôleur.

La vue peut aussi donner plusieurs vues, partielles ou non, des mêmes données. Par exemple, l'application de conversion de bases a un entier comme unique donnée. Ce même entier est affiché de multiples façons (en texte dans différentes bases, bit par bit avec des boutons à cocher, avec des curseurs). La vue peut aussi offrir la possibilité à l'utilisateur de changer de vue.

Rôle du contrôleur

Le contrôleur est chargé de la synchronisation du modèle et de la vue. Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer. Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle et ensuite avertit la vue que les données ont changé pour que celle-ci se mette à jour. Certains événements de l'utilisateur ne concernent pas les données mais la vue. Dans ce cas, le contrôleur demande à la vue de se modifier.

Dans le cas d'une base de données des emplois du temps. Une action de l'utilisateur peut être l'entrée (saisie) d'un nouveau cours. Le contrôleur ajoute ce cours au modèle et demande sa prise en compte par la vue. Une action de l'utilisateur peut aussi être de sélectionner une nouvelle personne pour visualiser tous ses cours. Ceci me modifie pas la base des cours mais nécessite simplement que la vue s'adapte et offre à l'utilisateur une vision des cours de cette personne.

Le contrôleur est souvent scindé en plusieurs parties dont chacune reçoit les événements d'une partie des composants. En effet si un même objet reçoit les événements de tous les composants, il lui faut déterminer quelle est l'origine de chaque événement. Ce tri des événements peut s'avérer fastidieuse et

peut conduire à un code pas très élégant (un énorme switch). C'est pour éviter ce problème que le contrôleur est réparti en plusieurs objets.

Interactions

Les différentes interactions entre le modèle, la vue et le contrôleur sont résumées par le schéma de la figure suivante.

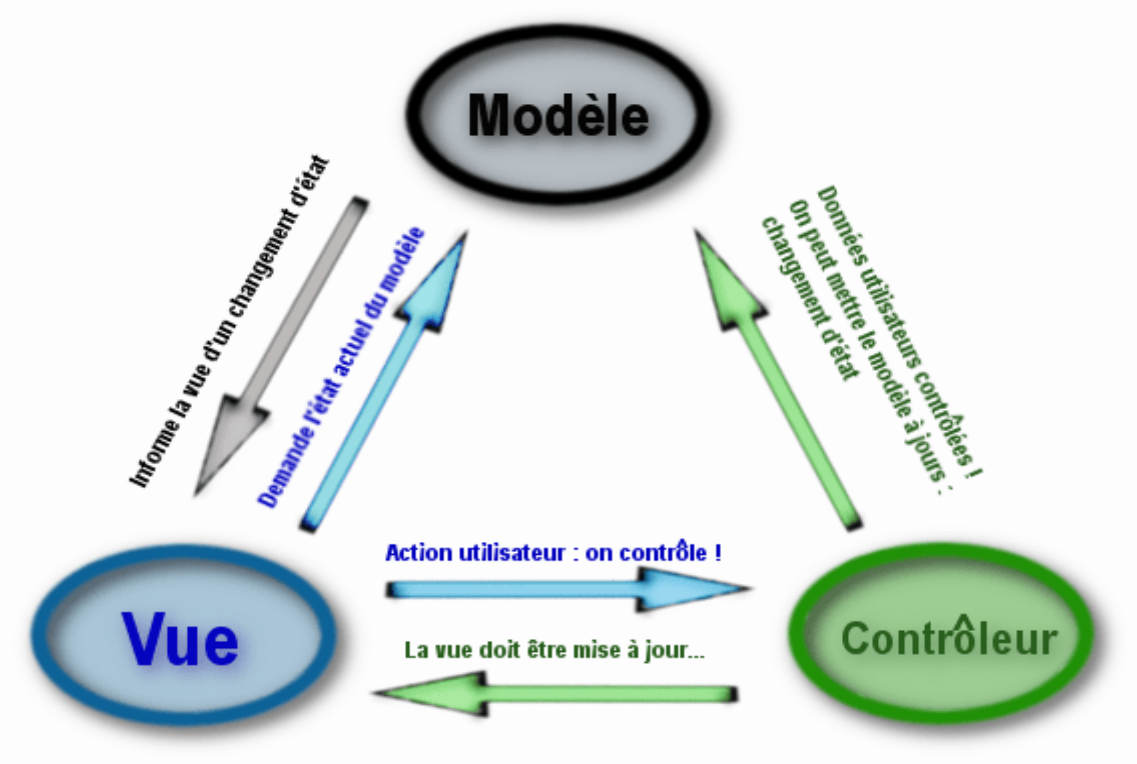

Fig 3: Modèle MVC

I. Programmation événementielle

La programmation des applications avec interfaces graphiques est généralement basée sur un paradigme nommé **programmation événementielle** (Event Programming).

Un **événement** (event) constitue une notification qui signale que quelque chose s'est passé (un fait, un acte digne d'intérêt).

Un événement peut être provoqué par :

- Une **action de l'utilisateur (**Un clic avec la souris, La pression sur une touche du clavier, Le déplacement d'une fenêtre ...etc.)
- **Un changement provoqué par le** système (Une valeur a changé (propriété), Un timer est arrivé à échéance, Un processus a terminé un calcul, Une information est arrivée par le réseau . . . etc.)

En ***JavaFX*** les événements sont représentés par des objets de la classe **Event** ou, plus généralement, d'une de ses sous-classes. De nombreux événements sont prédéfinis (MouseEvent, KeyEvent, DragEvent, ScrollEvent, …) mais il est également possible de créer ses propres événements en créant des sous-classes de Event.

J. Persistance de données (objets)

Différentes solutions peuvent être utilisées pour la persistance des objets en Java :

- **JDBC Java DataBase Connectivity**

  C'est une API (Application Programming Interface) qui permet l'accès à des données de type table de bases de données SGBD relationnelles :

  - ouvrir une connexion avec le SGBD
  - envoyer des requêtes SQL au SGBD
  - récupérer des données retournées par des requêtes SQL
  - et traiter ces données "table"
  - gérer les erreurs retournées par des requêtes au SGBD

- **La sérialisation** *:* un mécanisme permettant de stocker des objets dans des fichiers sans se soucier du format qui sera utilisé.

K. Travail à rendre

- Un court rapport (version papier) d'une longueur comprise entre 15 et 20 pages comportant (à rendre le jour de l'évaluation) :
  - Page de garde
  - Une introduction
  - Contexte et définition du problème
  - Objectif du projet
  - Description fonctionnelle des besoins
  - Partie conception UML : les diagrammes de cas d'utilisation, diagramme de classe et diagrammes de séquence.
  - l'organisation et la répartition des tâches au sein du groupe durant la durée du projet (brièvement).
  - bilan qualitatif du travail, difficultés rencontrées, points qui vous ont paru intéressants.
  - Une conclusion sur l'apport du projet en termes technique, scientifique, humain.
  - Le code source de votre application ne doit pas faire partie du rapport.
- Le code source complet et les ressources de votre application (y compris un archive jar exécutable) doivent être envoyé par mail à votre professeur avant deux jours de la date de l'évaluation.

L. Evaluation du projet et calendrier :

Durée de préparation du projet **: 2 mois et demi à compter du 04/03/2024**.

- Le planning de l'évaluation de vos projets vous sera communiqué ultérieurement.
- L'évaluation consiste à consacré 25 min pour chaque groupe.
- L'évaluation sera basée sur les critères suivants :
  - Le périmètre fonctionnel de l'application
  - La conception UML de l'application (Diagrammes de cas d'utilisation, Diagramme de classes, Diagrammes de séquence …).

- Qualité technique du code et test de fiabilité de l'application et persistance des données.
- La prise en considération des besoins non fonctionnel (sécurité, performance,…etc).
- L'ergonomie des interfaces.
- Lisibilité du code : présentation du programme, usage de variables ayant des noms explicites, commentaires.
- Démonstration de l'application et la pertinence des réponses aux questions posées lors de l'évaluation du projet réalisé.

N.B : Dans la situation où je recevrai des travaux qui représentent un cas de plagiat (copie intégrale ou partielle), la note finale sera divisée par le nombre de groupes qui ont rendu ces travaux identiques.