Documentation Développeur

Projet : Gestion de Projets des étudiants

Binôme :

Yanis MEHMEL

Yani LHADJ

Enseignantes :

Emna MATHLOUTHI

Khaddouja ZELLAMA

Sommaire

1. Introduction

Bienvenue dans la documentation pour le développeur du projet. Ce document vise à fournir une compréhension approfondie des choix architecturaux, des défis surmontés, de la répartition du travail entre les membres du binôme, et des réflexions stratégiques qui ont façonné le développement de ce projet.

1.1 Contexte du Projet

Tout au long de ce document, nous explorerons les décisions prises tout en justifiant nos choix, les difficultés techniques rencontrées et les solutions élaborées. De plus, nous détaillerons la répartition des tâches au sein du binôme et ferons le point sur les fonctionnalités implémentées, ainsi que sur celles qui n'ont pas été intégrées, avec une explication des raisons.

Nous vous invitons à plonger dans cette documentation pour acquérir une vision approfondie du processus de développement, des considérations architecturales et des défis résolus au cours de ce projet.

2. Choix d'Architecture

2.1 Justification des Choix

Nous avons délibérément opté pour l'adoption du design pattern DAO (Data Access Object) afin de structurer notre application de manière modulaire et de garantir une séparation claire entre la logique métier et l'accès aux données. Le principe fondamental du DAO est de centraliser toutes les opérations liées à la base de données dans des classes spécifiques, offrant ainsi une interface standardisée pour interagir avec ces données.

Packages Principaux

1. Package "classe"

Dans ce package, nous regroupons les entités métier de notre application, telles que les classes Admin, Binome, Etudiant, EtudiantBinome, Formation, Projet, et VueData. Chacune de ces classes représente une table de la base de données.

L'utilisation de ces classes permet de modéliser clairement les objets métier et d'encapsuler leurs comportements spécifiques.

2. Package "dao"

Le package dao est essentiel à la mise en œuvre du pattern DAO. Chaque entité métier a sa classe DAO associée (par exemple, AdminDAO, BinomeDAO, etc.). Ces classes sont responsables de toutes les opérations liées à la base de données concernant leur entité respective.

Le DAO encapsule les détails de l'accès aux données, fournissant une interface standardisée pour effectuer des opérations CRUD (Create, Read, Update, Delete). Cette approche facilite la maintenance, car toute modification de la structure de la base de données peut être traitée à un seul endroit.

3. Package "interfaceG"

Le package interfaceG gère les interfaces graphiques de l'application. Il contient des classes telles que Identification, App, AppBinome, etc. Ces interfaces utilisent les classes du package dao pour accéder aux données de manière transparente.

Nous avons respecté la bonne pratique de gestion de projet en évitant les interdépendances entre les packages, suivant ainsi les méthodes d'élimination des cycles de dépendance.

4. Package "exception”

Ce projet comprend également un package "exception" qui contient une exception "PlusDeDeuxEtudiants". Cette exception est levée lorsque l'utilisateur tente de créer un binôme avec plus de deux étudiants. Toutes les autres exceptions susceptibles d'être déclenchées sont de type SQLException, une exception prédéfinie dans la JDBC.

Structure de la Base de Données

Nous avons mis en place une structure de base de données relationnelle avec les tables suivantes :

- Formation
- Etudiant

- Projet
- Binome
- EtudiantBinome
- Admin (utilisé pour l'authentification)
- VueData (vue regroupant les informations nécessaires à l'interface graphique)

Chaque table est soigneusement définie avec des clés primaires, des clés étrangères, et des contraintes assurant la cohérence des données.

Il convient de noter que, tout au long du développement, nous avons utilisé une base de données locale pour faciliter le processus de conception et de test. Cependant, afin de permettre à l'application de s'exécuter sur n'importe quelle machine, nous avons migré la base de données vers un serveur en ligne.

Il est important de souligner que l'utilisation d'un serveur gratuit a entraîné des temps de réponse plus longs lors de l'accès à la base de données, impactant ainsi la performance de certaines fonctionnalités de l'application. Les connexions à distance avec le serveur gratuit ont introduit des délais qui peuvent rendre certaines opérations plus lentes que lors de l'utilisation de la base de données locale.

Illustration

La classe "EtudiantDAO.java" accède à la base de données pour effectuer soit une recherche, soit une mise à jour. Par exemple, elle permet de renvoyer tous les étudiants de la table sous la forme d'une collection de type "Etudiant.java". La classe "AppEtudiant.java" mettra à jour l'interface graphique en utilisant les informations de chaque étudiant stocké dans la collection. En adoptant cette approche, chaque classe a un rôle spécifique en fonction de son package d'appartenance : une classe du package "classe" a pour rôle de stocker les différentes informations, une classe du package "dao" a pour rôle d'accéder à une table spécifique de la base de données et d'effectuer des opérations sur ces dernières (mise à jour, recherche), et une classe du package "interfaceG" a pour objectif d'afficher ces informations.

3. Difficultés Rencontrées

3.1 Difficultés Techniques

Au cours du projet, plusieurs difficultés techniques ont émergé, nécessitant une réflexion approfondie et des ajustements.

3.1.1 Conception de la Base de Données

L'une des premières difficultés a résidé dans l'établissement d'une base de données finale, pratique et cohérente. Les multiples changements dans la conception initiale ont soulevé des défis quant à la définition d'un modèle entités-associations optimal. Après plusieurs ajustements, la consolidation autour d'un modèle cohérent a grandement facilité la suite du travail.

3.1.2 Intégration avec les Entités Java

L'intégration efficace entre la base de données et les entités Java a été une autre problématique. Trouver une approche pratique et cohérente pour interagir avec la base de données tout en utilisant les entités Java a nécessité des itérations et des ajustements. La synchronisation entre la logique métier et la persistance des données a été cruciale pour assurer la cohérence de l'application.

3.1.3 Implémentation de l'Interface Graphique

Une difficulté majeure a émergé lors de l'implémentation de l'interface graphique. Initialement, nous avions envisagé d'utiliser JavaFX, assistés par un outil de génération d'interfaces tel que Scene Builder. Cependant, nous avons rapidement constaté que l'utilisation d'outils externes pouvait introduire des erreurs difficilement résolubles, car le code généré n'était pas entièrement maîtrisé. Nous avons donc fait le choix de Swing, bien que son utilisation ne soit pas intuitive, nécessitant une courbe d'apprentissage. L'implémentation de l'interface graphique a ainsi constitué une étape complexe, demandant un investissement significatif en temps et en apprentissage.

3.2 Solutions Apportées

3.2.1 Stabilisation de la Conception de la Base de Données

Pour surmonter les défis liés à la conception de la base de données, nous avons adopté une approche itérative. La recherche d'un modèle entités-associations cohérent a impliqué des discussions approfondies et des ajustements continus. Une

fois le modèle établi, le processus de développement a été considérablement simplifié.

3.2.2 Synchronisation Entre Java et la Base de Données

L'intégration entre les entités Java et la base de données a été optimisée en assurant une synchronisation précise entre la logique métier de l'application et la persistance des données. Des tests approfondis ont été réalisés pour garantir la stabilité et la cohérence de cette interaction cruciale.

3.2.3 Transition vers Swing pour l'Interface Graphique

Face aux défis liés à l'interface graphique, la transition vers Swing a été accompagnée d'une phase d'apprentissage intensive. Des ressources en ligne et des exemples pratiques ont été explorés pour maîtriser cette technologie. Malgré sa complexité, cette décision a permis de garantir un contrôle total sur le code de l'interface graphique, évitant ainsi des erreurs potentielles induites par l'utilisation d'outils externes.

En surmontant ces difficultés, le projet a progressé de manière significative, démontrant notre capacité à résoudre des problèmes techniques complexes et à prendre des décisions judicieuses pour assurer le succès du développement.

4. Répartition du Travail

4.1 Tâches et Responsabilités

La répartition des tâches au sein du binôme a été soigneusement planifiée pour maximiser l'efficacité et garantir une progression harmonieuse du projet.

4.1.1 Utilisation de Trello et Git

Pour organiser notre travail, nous avons opté pour l'utilisation conjointe de Trello et Git. Trello a été instrumental pour la gestion des tâches, permettant une visualisation claire des objectifs et des étapes à franchir. Chaque membre a pu suivre et mettre à jour l'avancement de ses propres tâches, facilitant ainsi la coordination.

4.1.2 Collaboration sur Git

Git a joué un rôle central dans la répartition des responsabilités. Chaque fonctionnalité ou tâche a été traitée dans une branche dédiée, favorisant un développement parallèle sans compromettre l'intégrité du code. Les pulls réguliers et les résolutions de conflits ont été gérés de manière transparente, préservant la cohérence du code source.

4.1.3 Répartition des Tâches

Les responsabilités ont été attribuées en fonction des compétences et des préférences de chaque membre. Par exemple, la conception de la base de données a été confiée à un membre avec des compétences avancées en modélisation, tandis que l'implémentation de l'interface graphique a été prise en charge par celui ayant une affinité particulière avec les technologies graphiques.

4.2 Collaboration

La collaboration a été un pilier essentiel du succès du projet, favorisant un environnement de travail efficace et harmonieux.

4.2.1 Réunions Hebdomadaires et Virtuelles

Nous avons maintenu une communication régulière à travers des réunions hebdomadaires virtuelles via Zoom. Ces sessions ont permis de discuter de l'avancement global du projet, d'échanger des idées et de résoudre rapidement les éventuels obstacles. La flexibilité des réunions en ligne a facilité la participation de chacun malgré les contraintes d'emploi du temps.

4.2.2 Réunions en Présentiel

En complément des réunions virtuelles, des rencontres en présentiel ont été organisées lorsque cela était possible. Ces réunions ont renforcé la cohésion de l'équipe, favorisant un échange plus informel et stimulant la créativité collective.

4.2.3 Gestion Agile avec Trello

L'approche Agile, facilitée par l'utilisation de Trello, a permis une adaptation constante aux changements et aux retours d'expérience. Les ajustements ont été intégrés de manière itérative, garantissant une flexibilité essentielle dans un environnement de développement dynamique.

En combinant l'efficacité des outils tels que Trello et Git avec une communication transparente, la répartition des tâches et la collaboration ont été gérées de manière optimale, contribuant au succès global du projet.

5. Fonctionnalités

5.1 Fonctionnalités Implémentées

5.1.1 Opérations de Recherche et d'Affichage

Notre application offre des fonctionnalités de recherche et d'affichage exhaustives. Les utilisateurs peuvent lister les étudiants, les formations, les projets, les binômes, et accéder aux notes associées à chaque étudiant pour un projet spécifique. Cette fonctionnalité fournit une vue détaillée des données stockées dans la base de données, offrant ainsi une expérience utilisateur complète et informative.

5.1.2 Opérations de Mise à Jour

L'application prend en charge des opérations de mise à jour dynamiques. Les utilisateurs ont la possibilité d'ajouter ou de modifier les informations relatives à un étudiant, une formation, un projet, ou un binôme. Cette flexibilité permet une gestion complète et personnalisée des données, adaptée aux besoins évolutifs de l'utilisateur.

5.1.3 Gestion des Notes

Les utilisateurs peuvent attribuer et consulter les notes associées à chaque étudiant pour un projet spécifique.

Les utilisateurs ont la possibilité d'attribuer et de consulter les notes associées à chaque étudiant pour un projet spécifique. Cette gestion des notes inclut la capacité pour un utilisateur de donner une note de rapport au binôme dans son ensemble et une note de soutenance distincte pour chaque étudiant du binôme, le tout spécifique à un projet donné.

La note finale attribuée au projet est calculée automatiquement en utilisant une formule qui prend en considération à la fois la note de soutenance, la note de rapport, et la date de remise effective du projet. Cette approche permet d'obtenir une

évaluation globale prenant en compte plusieurs aspects de la performance du binôme et des étudiants individuels.

5.2 Fonctionnalités Non Implémentées

Malgré nos efforts pour concevoir une application complète, certaines fonctionnalités n'ont pas été implémentées pour diverses raisons.

5.2.1 Gestion des Rapports Statistiques

Une fonctionnalité de génération de rapports statistiques n'a pas été intégrée dans cette version. Cette décision découle de contraintes de temps et de priorisation des fonctionnalités. Nous reconnaissons l'importance des rapports statistiques.

5.2.2 Interface Utilisateur Avancée

Bien que l'interface utilisateur actuelle réponde aux besoins de base, une interface plus avancée avec des fonctionnalités conviviales aurait pu être développée. Cependant, nous avons choisi de privilégier la stabilité et la performance dans cette version initiale, en réservant l'amélioration de l'interface utilisateur pour des itérations futures.

Chaque décision de non-implémentation a été prise après une évaluation minutieuse des priorités et des contraintes, visant à garantir une application robuste et fonctionnelle.

6. Résolution de Problèmes

6.1 Gestion des Bugs

6.1.1 Gestion des Exceptions liées à la Base de Données

Durant le développement de l'application, nous avons identifié et résolu plusieurs bugs liés à la gestion de la base de données. Un exemple notable était la tentative d'ajout d'un étudiant avec un identifiant déjà existant, entraînant une SQLException due à la violation de la contrainte de clé primaire. Pour remédier à cela, nous avons implémenté une gestion d'exceptions qui informe désormais l'utilisateur de

l'impossibilité d'effectuer une telle opération. Ces pratiques suivent les principes enseignés lors des cours sur la gestion d'exceptions en Java.

6.1.2 Gestion des Binômes

Un autre bug significatif concernait la gestion des binômes. L'application limite le nombre d'étudiants qu'un utilisateur peut ajouter à un binôme à deux. Cependant, cette opération générait initialement une exception OutOfBoundException. Pour résoudre ce problème, nous avons créé une exception personnalisée, PasPlusDeDeuxEtudiant, permettant de signaler de manière claire et précise cette contrainte à l'utilisateur.

6.1.3 Résolution des Problèmes d'Interface Graphique

Nous avons également rencontré plusieurs problèmes lors de l'implémentation de l'interface graphique. La lecture de la documentation Java Swing nous a permis de résoudre ces problèmes avec succès. La compréhension approfondie de la documentation a été cruciale pour surmonter les difficultés liées à l'utilisation de Swing.

6.2 Évitement de la Redondance

Chaque classe du package DAO a été conçue avec un rôle spécifique, permettant ainsi une organisation claire et modulaire du code. Par exemple, les classes DAO accèdent à la base de données pour effectuer des opérations de recherche ou de mise à jour, tandis que les classes du package classe stockent les informations et les classes du package interfaceG sont responsables de l'affichage des informations. Cette division des responsabilités contribue à maintenir un code propre, évitant les redondances inutiles et facilitant la compréhension et la maintenance du système.

7. Conclusion

7.1 Bilan Global

Le projet aboutit à la création d'une application fonctionnelle répondant aux besoins spécifiques de gestion des étudiants, formations, projets et binômes. En adoptant le design pattern DAO, nous avons établi une architecture modulaire et bien structurée,

favorisant la maintenance et l'évolutivité du système. Les choix techniques, tels que l'utilisation de Git et Trello pour la gestion des tâches, ont contribué à l'efficacité de la collaboration au sein du binôme.

L'interface graphique, bien que présentant des défis, a été mise en œuvre avec succès en optant pour Swing et en surmontant les problèmes rencontrés grâce à une lecture approfondie de la documentation Java. Les difficultés techniques liées à la conception de la base de données ont été résolues de manière itérative, conduisant à un modèle stable et cohérent.

Le projet atteint ses objectifs initiaux et fournit une base solide pour d'éventuelles améliorations futures. Les difficultés rencontrées ont été des opportunités d'apprentissage, contribuant à renforcer nos compétences techniques et notre compréhension des bonnes pratiques de développement.

7.2 Leçons Apprises

7.2.1 Importance de la Planification

La planification initiale du projet, incluant la conception de la base de données et la répartition des tâches, a joué un rôle essentiel.

7.2.2 Flexibilité et Adaptabilité

L'approche Agile, soutenue par l'utilisation de Trello, a démontré l'importance de la flexibilité et de l'adaptabilité dans le processus de développement. La capacité à ajuster les priorités en fonction des retours d'expérience a contribué à une progression plus efficace du projet.

7.2.3 Maîtrise des Outils

La maîtrise des outils tels que Git et la compréhension approfondie de la documentation Java ont été des éléments clés pour résoudre les problèmes rencontrés. Investir du temps dans l'apprentissage des outils utilisés a payé en termes de résolution rapide des obstacles.

7.2.4 Collaboration Efficace

La combinaison de réunions virtuelles régulières, de réunions en présentiel, et de l'utilisation d'outils de collaboration a favorisé une collaboration efficace au sein du binôme.

En conclusion, ce projet a été une expérience enrichissante, combinant des aspects techniques et de gestion de projet. Les compétences acquises et les leçons apprises seront précieuses pour nos projets futurs en tant que développeurs.