Dauphine | PSL
UNIVERSITÉ PARIS

documentation de développeur

Détails techniques et architecturaux du projet

Nom des Auteurs: Yingxuan LI

Shengqi MA

Date de Soumission: 28/11/2023

Table des Matières

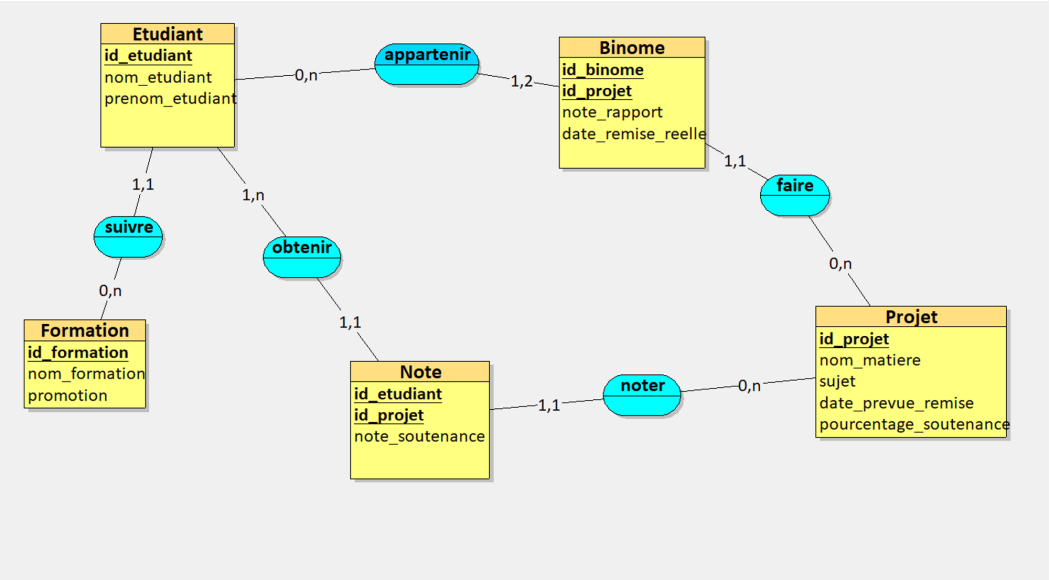
I. Outil et Architecture du Projet

1. Environnement de Développement

- IDE : Utilisation d'IDEA pour le développement principal et de DataGrip pour la gestion des bases de données.
- Version Java : Le projet est développé en Java 19.
- Bases de Données : Utilisation de SQLite comme système de gestion de base de données.
- Framework Java pour la Manipulation des Bases de Données : MyBatis est utilisé pour l'intégration et la manipulation des bases de données.
- Interface Graphique : JavaFX est employé pour développer l'interface utilisateur.
- Plateforme de Gestion de Version : Le code source est géré et partagé via GitHub.

2. Architecture de la base de données

L'architecture de la base de données est définie par un modèle entité-association qui décrit les relations entre les différentes entités de notre système. Les principales entités et leurs relations sont détaillées comme suit :



- Étudiant : Chaque étudiant est identifié par un id_etudiant unique et possède des attributs tels que nom_etudiant et prenom_etudiant.
- Formation : Décrit les programmes de formation avec un id_formation, un nom_formation, et une promotion spécifique.
- Binome : Représente les équipes de deux étudiants, avec une particularité importante : chaque binôme possède une double clé primaire composée de

id_binome et id_projet, ce qui permet d'avoir des numéros de binôme identiques pour des projets différents, assurant ainsi que le binôme N°1 pour le projet "Entrepôt de données" peut être différent du binôme N°1 pour le projet "J2EE".

- Projet : Contient les détails des projets tels que id_projet, nom_matiere, sujet, date_prevue_remise, et pourcentage_soutenance.
- Note : Stocke les évaluations des étudiants avec une relation directe avec l'entité Étudiant et Projet.

Pour notre système de gestion de base de données, nous avons opté pour une stratégie non conventionnelle : l'absence de clés étrangères. Nous nous reposons uniquement sur les clés primaires, et toute gestion des contraintes de clés étrangères est assurée par le code Java. Cette décision est motivée par plusieurs raisons :

- Souplesse : Gérer les relations directement dans le code Java offre plus de souplesse pour les cas où les contraintes de base de données standard sont trop rigides ou ne correspondent pas à nos cas d'usage complexes.
- Performance : L'absence de vérification des contraintes de clés étrangères par le système de gestion de base de données peut entraîner une amélioration des performances, en particulier pour les opérations d'écriture volumineuses.
- Contrôle des transactions : Nous avons un contrôle plus granulaire sur les transactions, permettant de gérer les exceptions et les cas d'erreur de manière plus précise.
- Portabilité : Cette méthode rend l'application moins dépendante des spécificités des systèmes de gestion de base de données, facilitant ainsi la migration entre différentes bases de données si nécessaire.

En conclusion, l'architecture de notre base de données est pensée pour une optimisation de la flexibilité et des performances, tout en garantissant l'intégrité des données à travers une gestion de contraintes personnalisée et maîtrisée dans le code Java.

3. Architecture du Projet

Diagramme de Classes

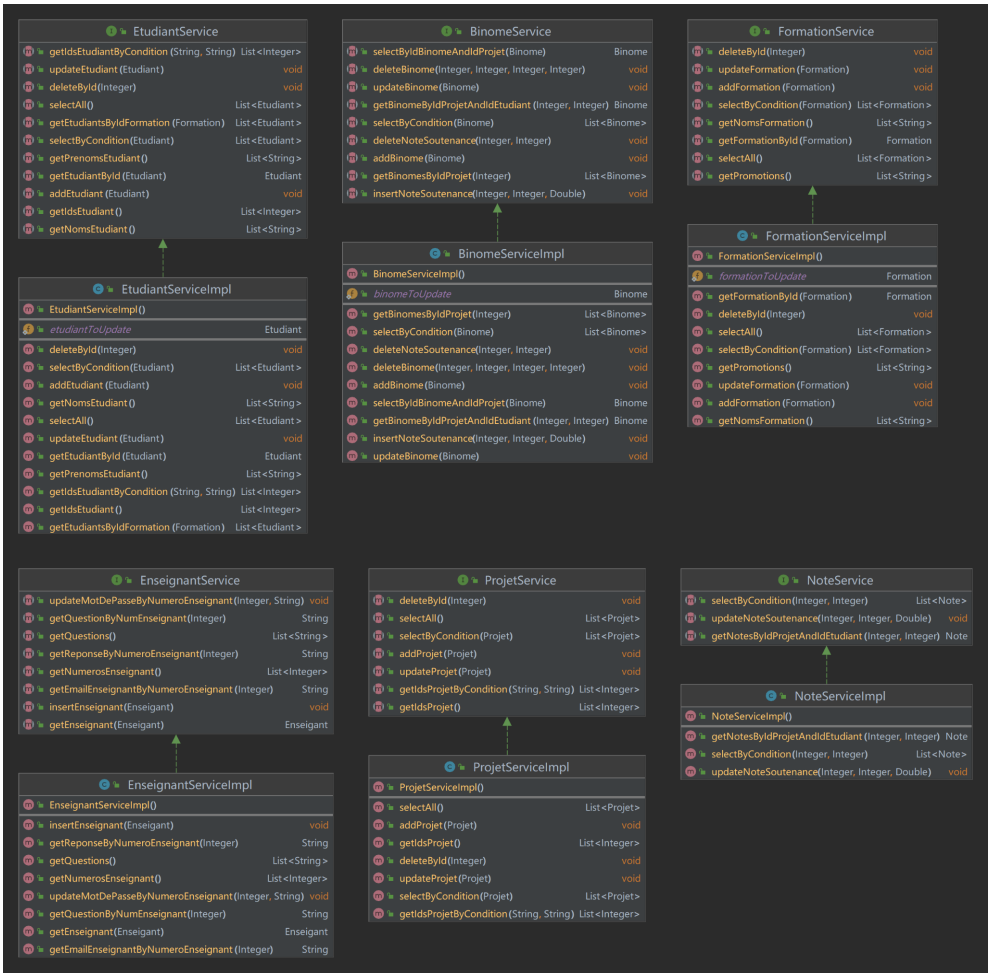
Le diagramme de classes représente la structure de notre modèle de données et la conception orientée objet de notre système. Il décrit les entités clés telles que Étudiant, Enseignant, Formation, Binome, Note et Projet, ainsi que leurs attributs et méthodes respectifs.

- Étudiant : Stocke les informations sur les étudiants et offre des méthodes pour interagir avec ces données.
- Enseignant : Gère les détails relatifs aux enseignants, y compris leurs questions de sécurité pour la récupération de mots de passe.

- Formation : Contient des informations sur les formations proposées et les promotions.
- Binome : Décrit les groupes de deux étudiants et est lié à des projets spécifiques, permettant une distinction entre les binômes du même numéro pour des projets différents.
- Note : Enregistre les notes associées à des étudiants et des projets.
- Projet : Contient les détails sur les projets attribués aux étudiants.

Ce diagramme met en évidence la manière dont les objets interagissent entre eux et avec la base de données, reflétant notre approche de conception qui privilégie la cohérence et la maintenance facilitée.

Diagramme des Services



Le diagramme des services met en exergue la couche intermédiaire de notre architecture, jouant le rôle de médiateur entre l'interface utilisateur et les mécanismes d'accès aux données. Pour chaque entité de notre modèle de données, un service dédié est instauré, tels que EtudiantService, EnseignantService, FormationService, BinomeService, NoteService et ProjetService. Ces services renferment la logique métier de l'application et communiquent avec la base de données essentiellement au moyen de mappers, préférant une approche moderne et flexible aux DAOs traditionnels.

Les implémentations spécifiques de ces services, comme EtudiantServiceImpl ou EnseignantServiceImpl, concrétisent les opérations commerciales de chaque entité. Ainsi, BinomeServiceImpl prend en charge la gestion des binômes, y compris les opérations de récupération, de mise à jour, de suppression et l'ajout de notes de soutenance.

Les choix de conception tels que l'héritage et le polymorphisme sont exploités pour consolider notre architecture. L'héritage nous a permis de généraliser des comportements communs dans des classes de base, tandis que le polymorphisme facilite la gestion de divers comportements spécifiques à chaque entité via des interfaces communes. Ce paradigme assure une plus grande flexibilité et réutilisation du code dans le cadre de l'extension des fonctionnalités ou de l'ajout de nouvelles entités.

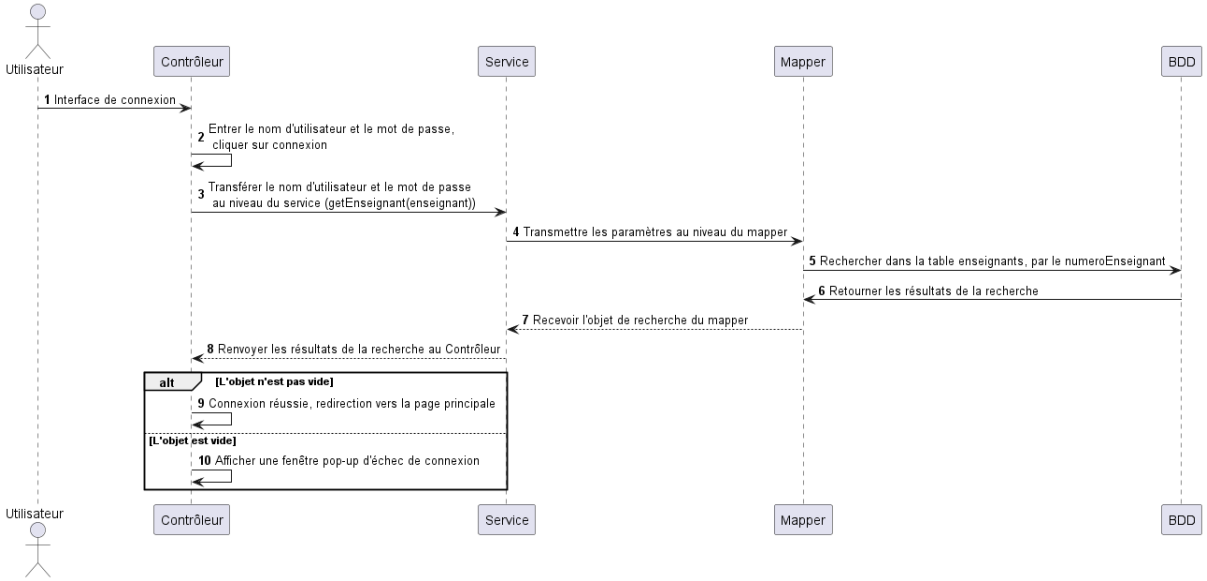
La structure actuelle offre plusieurs avantages distincts :

- Mappers : Ils permettent de définir les interactions avec la base de données de manière déclarative, s'appuyant sur MyBatis. Ce choix diminue le code standardisé et améliore la lisibilité, simplifiant la maintenance et les évolutions futures.
- Services : Ils centralisent la logique d'affaires et isolent les appels aux mappers, ce qui permet de délester les contrôleurs de toute logique métier complexe.
- Contrôleurs : Ils dirigent les requêtes des utilisateurs vers les services appropriés et présentent les réponses via des vues adaptées, se concentrant uniquement sur la gestion des flux d'entrée-sortie de l'utilisateur.

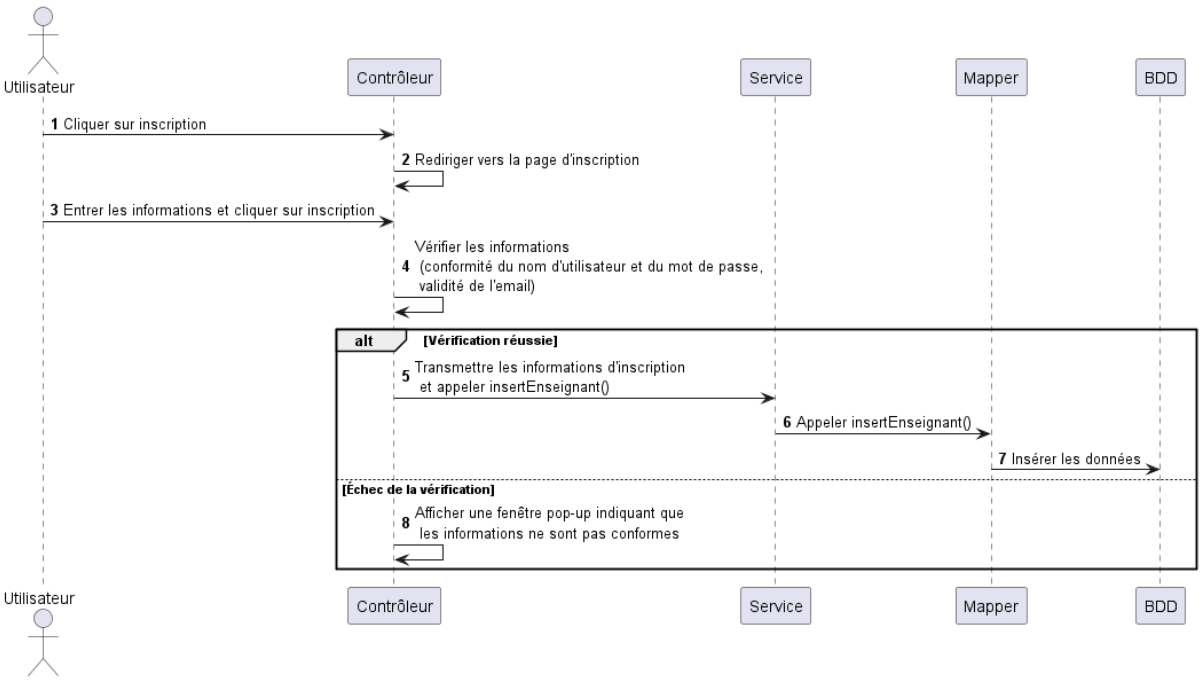
Ce découpage stratégique favorise la testabilité, la maintenabilité et l'évolutivité du système. Il témoigne de notre engagement à adhérer à des principes de conception soignés et à des pratiques d'ingénierie logicielle rigoureuses, démontrant notre

capacité à mener une réflexion approfondie sur l'architecture de l'application et sur la manière dont les composants interagissent de manière cohérente et efficiente.

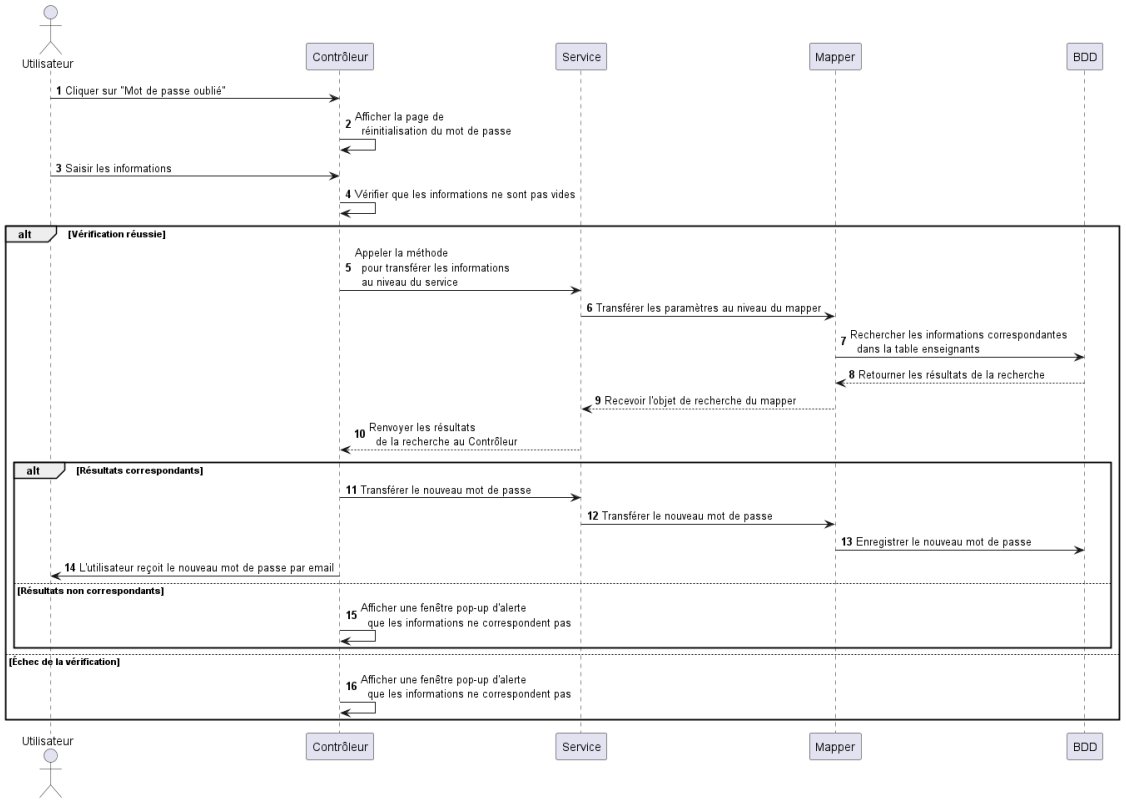
4. Diagrammes de Séquence



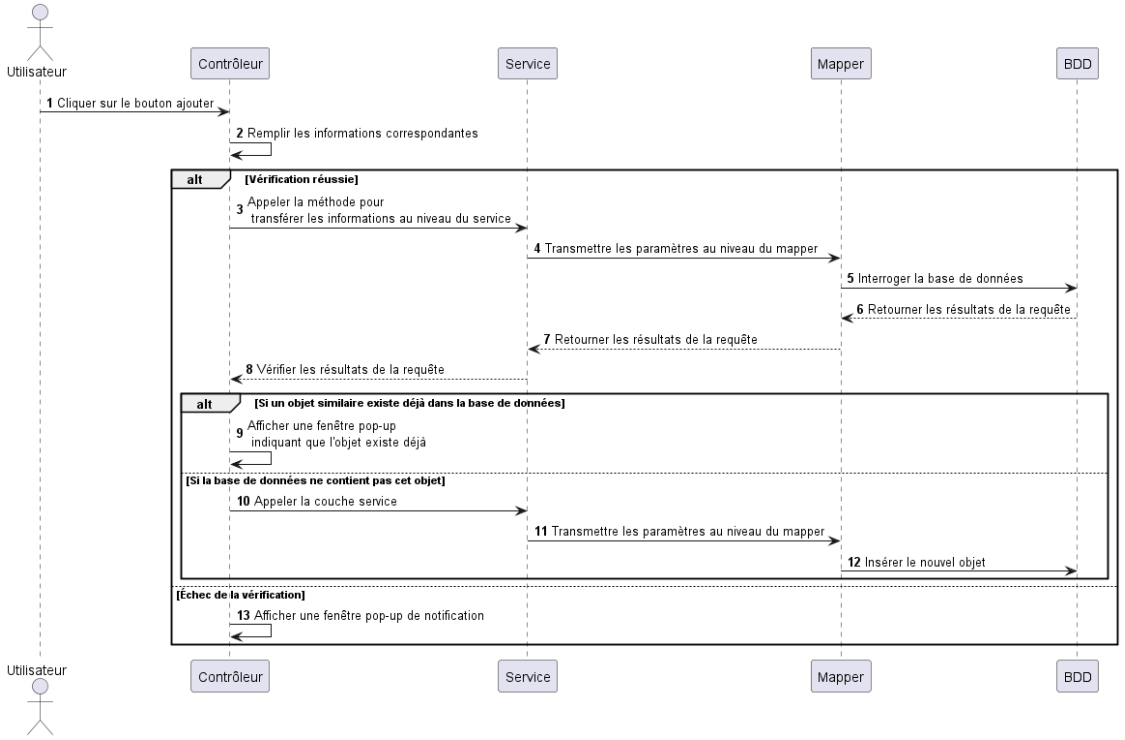
Connexion Utilisateur : Ce diagramme montre comment un utilisateur se connecte à l'application. Il illustre le processus depuis l'entrée des identifiants jusqu'à la validation de l'authentification et la redirection vers la page principale ou l'affichage d'un message d'erreur.



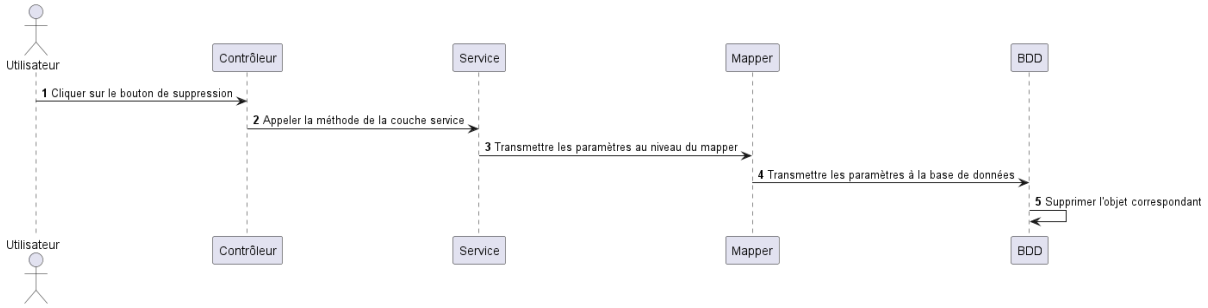
Inscription d'un Nouvel Utilisateur : Ici, nous décrivons les étapes de l'inscription d'un nouvel utilisateur, depuis la saisie des informations jusqu'à leur insertion dans la BDD, en passant par la validation des données et le traitement des erreurs.



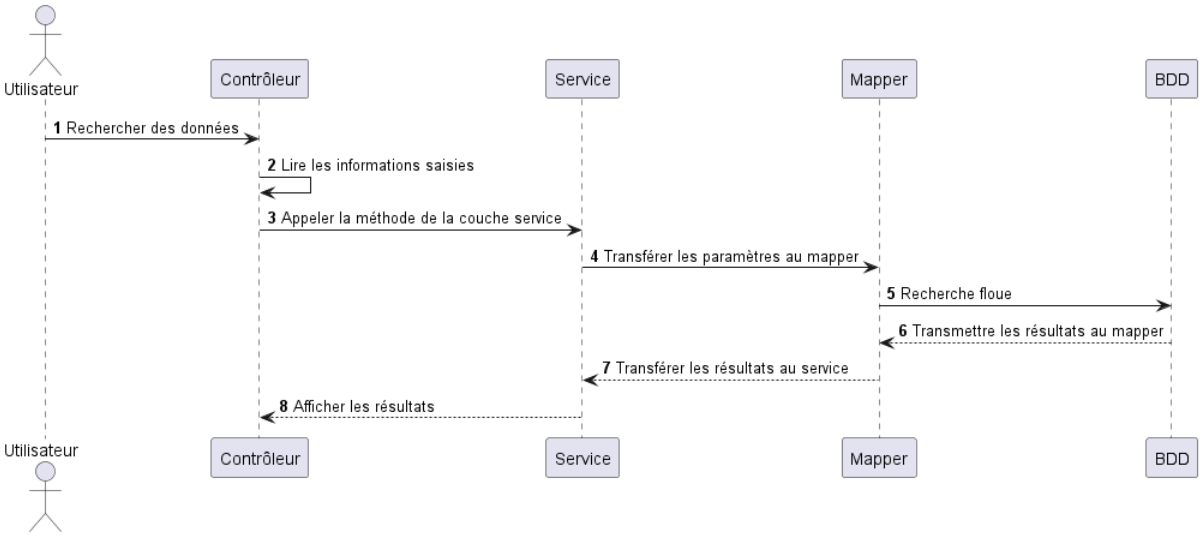
Réinitialisation du Mot de Passe : Ce diagramme explique le processus de réinitialisation du mot de passe pour un utilisateur ayant oublié ses identifiants, en incluant la validation de sécurité et l'envoi du nouveau mot de passe par email.



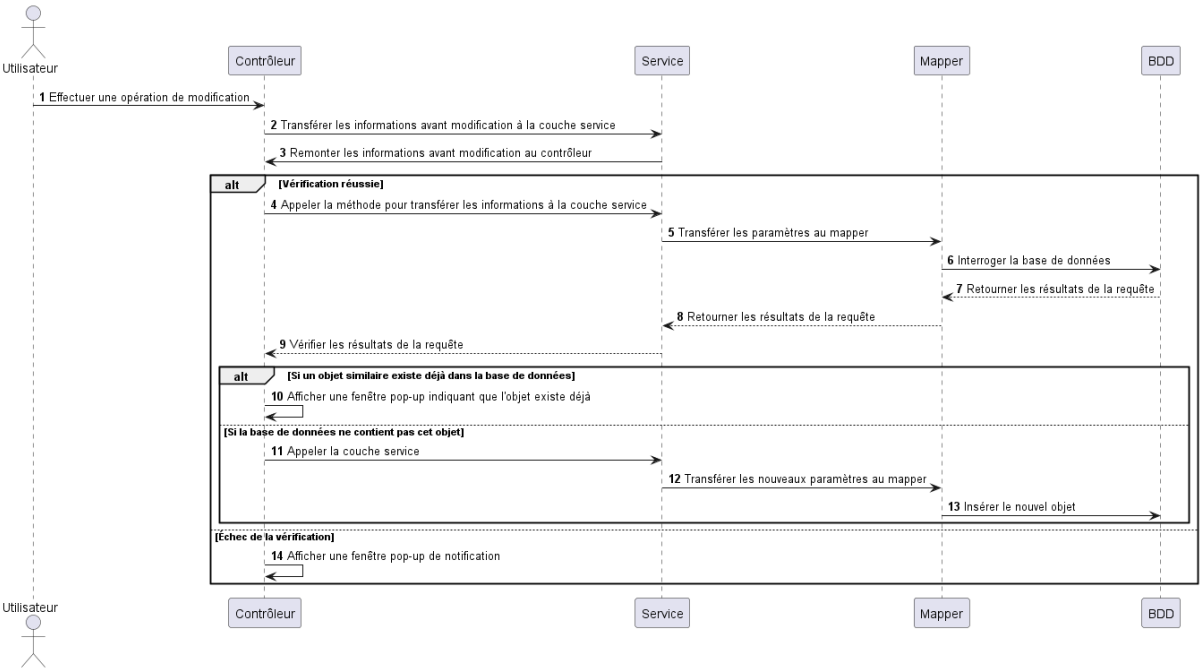
Ajout de Données : Cet enchaînement d'actions montre comment un utilisateur peut ajouter des données à la base de données, et comment l'application valide et insère ces données après vérification qu'elles n'existent pas déjà.



Suppression de Données : Ce diagramme décrit le flux permettant à un utilisateur de supprimer des données de la BDD via l'interface utilisateur, avec l'intervention du service et du mapper pour exécuter l'opération.



Recherche de Données : Ce diagramme illustre la fonctionnalité de recherche floue, démontrant comment les requêtes sont traitées et comment les résultats correspondants sont retournés à l'utilisateur.



Modification de Données : Nous présentons le processus de modification des données, y compris la validation en amont et la mise à jour en aval dans la BDD, en soulignant les vérifications pour prévenir les erreurs.

II. Difficultés

1. Difficultés des organisations

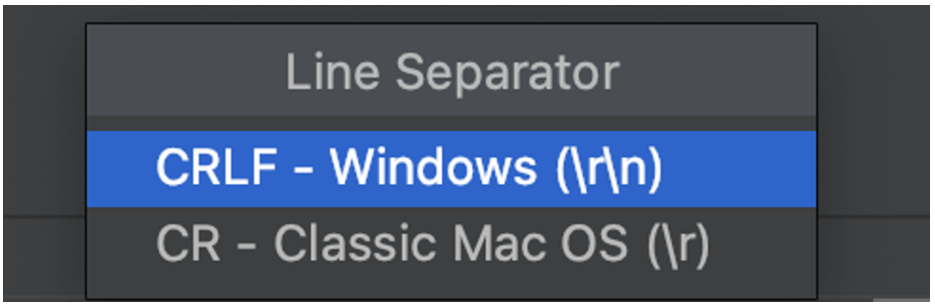
a.Gestion des Conflits de Code et Collaboration d'Équipe

Le projet n'était pas assez important, la technologie n'était pas assez mûre, il n'y avait pas de chef de projet, etc... Nous ne pouvions pas séparer complètement le code dont nous étions responsables, ce qui signifiait que git avait souvent quelques fusions.

Et la résolution de ces problèmes fait perdre beaucoup de temps, plus encore que l'écriture du code. Mais nous avons trouvé une solution : lorsque nous modifions un code, nous nous assurons que l'autre partie ne l'a pas modifié.

b. Compatibilité Inter-Systèmes et Gestion des Encodages

L'un des membres de notre équipe utilise Mac OS tandis que l'autre opère sous Windows, ce qui entraîne des différences d'encodage et de formatage de fichiers susceptibles de causer des bogues occasionnels. Cependant, cette diversité d'environnements a également servi de test grandeur nature, nous permettant d'assurer que notre code est exécutable avec succès sur les deux systèmes.



2. Difficultés techniquement

a. Choix Technologiques et Courbe d'Apprentissage

Nous avons choisi mybatis et javaFX, les technologies dominantes de l'entreprise, pour notre projet, même si nous n'avions que peu de connaissances préalables de ces technologies. En effet, ce projet nous a

permis d'apprendre ces technologies afin de pouvoir les utiliser dans notre travail par la suite.

b. Défis et Solutions de la Persistance des Données avec SQLite et JAR

Nous avons initialement choisi SQLite pour stocker la base de données directement dans le projet car cela présentait un avantage : le projet pouvait être exécuté n'importe où tant que l'utilisateur avait accès au fichier de la base de données. Cependant, nous avons rencontré un obstacle : les fichiers JAR sont en lecture seule, ce qui signifie que nous ne pouvions pas modifier les données qu'ils contiennent, rendant ainsi impossible toute écriture ou mise à jour dans la base de données.

Après avoir exploré plusieurs méthodes, nous avons trouvé une solution en plaçant le fichier de base de données et le JAR dans le même répertoire sur le système de fichiers de l'utilisateur. Cette approche a permis de contourner la limitation en lecture seule des JAR. En exécutant le JAR, l'application peut accéder au fichier de base de données situé dans le même répertoire, ce qui lui permet de lire mais aussi de modifier les données si nécessaire. Cela a rendu notre application portable facile à déployer, tout en conservant la fonctionnalité de persistance des données.

c. Optimisation de la Fonctionnalité de Réinitialisation du Mot de Passe

Lors de la mise en place de la fonctionnalité de réinitialisation de mot de passe, nous avions prévu d'envoyer le nouveau mot de passe généré par e-mail aux utilisateurs. Cependant, bien que le code ne présentait aucune erreur, les e-mails n'étaient pas reçus. Cette tentative étant nouvelle pour nous et le projet n'étant pas encore entièrement conforme à nos exigences, nous avons décidé de suspendre temporairement cette méthode au profit d'une solution alternative : afficher le nouveau mot de passe dans une fenêtre pop-up. Après avoir satisfait à toutes les exigences du projet, nous avons revisité la fonction d'envoi d'e-mails pour comprendre l'échec initial. Nous avons découvert que le service n'était pas activé dans votre compte de messagerie et qu'un mot de passe d'application n'avait pas été configuré. Une fois ces ajustements effectués, nous avons réussi à envoyer les e-mails avec succès.

3. Difficultés des codes

a. Le Défi des Expressions Régulières

Lors de la mise en place de la fonctionnalité d'inscription des utilisateurs, nous avons utilisé des expressions régulières pour valider les mots de passe et les adresses e-mail. Cependant, lors des tests avec nos propres adresses e-mail, nous avons rencontré des difficultés à trouver une expression régulière qui fonctionne correctement. Malgré de nombreux essais infructueux avec différentes expressions régulières, ce n'est qu'après avoir consulté abondamment la documentation et révisé méticuleusement notre expression régulière actuelle que nous avons finalement mis au point une solution qui valide efficacement les adresses e-mail.

b. Résolution d'un Bug dans le Module de Notation

Alors que nous étions sur le point de finaliser la fonctionnalité du module 'Note', nous avons découvert un bug : lorsque nous modifions les notes de soutenance et actualisons la page, la note finale ne change pas. Au début, nous pensions que le problème venait du fait que les notes modifiées n'étaient pas correctement sauvegardées dans la base de données. Cependant, après vérification, nous avons confirmé que les modifications étaient bien enregistrées. Nous avons ensuite décidé de réécrire la méthode d'actualisation de la page. Comme le score final n'était pas stocké dans la base de données, nous avons suspecté que le problème provenait du fait que le calcul du score final n'était pas effectué lors de l'actualisation. Malgré l'ajout de cette étape de calcul, le problème persistait. Finalement, nous avons eu l'idée d'utiliser la méthode `refresh` de l'objet `TableView`, ce qui a finalement résolu le problème.

c. Optimisation de la Structure des Tables Binôme et Étudiant

Lors de la conception initiale de notre base de données, nous avions choisi de stocker les informations des deux étudiants de chaque binôme directement dans la table `binômes`. Cependant, au cours du développement, nous avons réalisé que les binômes pourraient comporter un ou deux étudiants et avons identifié une relation de type plusieurs-à-plusieurs entre les binômes et les étudiants. Par conséquent, nous avons décidé d'extraire la relation entre les binômes et les étudiants dans une table séparée, nommée `appartenir`. Cette table sert à stocker les relations entre ces deux entités. L'avantage de cette approche est qu'elle résout directement le problème potentiel où un même étudiant pourrait apparaître deux fois dans un même binôme pour un projet.

d. Défi de la Gestion des Clés Multiples avec Mybatis

Le plus grand défi auquel nous avons été confrontés concernait les pages des modules 'Binôme' et 'Note'. Après avoir implémenté les méthodes de requête pour ces deux modules, nous avons constaté que les résultats affichés ne correspondaient pas à nos attentes. Initialement, nous avons suspecté une erreur dans les requêtes SQL, nous les avons donc exécutées directement dans la base de données et avons constaté que les résultats étaient corrects. Nous avons ensuite envisagé que le problème pourrait résider dans le processus de mappage des données en objets. Lors de la conception des tables de la base de données, nous avons utilisé des clés primaires doubles pour la table 'Binôme' et triples pour la table 'Note', et avons choisi Mybatis comme framework de persistance pour notre projet. Cependant, étant donné que Mybatis était un nouveau framework pour nous, nous n'étions pas au courant que Mybatis ne reconnaissait pas les multiples clés primaires. Ce n'est qu'après avoir imprimé les objets mappés sur la console et les avoir examinés que nous avons pris conscience de ce problème. Finalement, nous avons résolu ce problème en effectuant plusieurs requêtes, fixant la valeur d'une clé primaire à chaque fois, pour contourner le problème de mappage des objets.

e. Amélioration de la Fonctionnalité d'Ajout et de Mise à Jour dans le Module Binôme

Dans la fonctionnalité d'ajout de binômes dans le module binôme, nous avons dû effectuer de nombreuses vérifications sur les saisies des utilisateurs. Ces vérifications incluent l'impossibilité de choisir deux fois le même étudiant, de ne pas ajouter un étudiant déjà présent dans un binôme d'un projet donné dans un autre binôme du même projet, de s'assurer que le premier étudiant n'est pas vide, et que la note rapport doit être une valeur entre 0 et 20. Cependant, lors de la consultation des informations des binômes, nous avons constaté une impossibilité de récupérer les données des binômes. Nous avons ensuite découvert que cela était dû à des valeurs nulles dans les champs note rapport et date réelle remise. Par conséquent, nous avons modifié le code de la fonctionnalité d'ajout pour insérer des valeurs par défaut dans ces champs lors de l'insertion d'un binôme. De plus, nous avons ajouté une fonctionnalité pour que si ces champs contiennent des valeurs par défaut, les champs correspondants dans l'interface de mise à jour du binôme soient définis comme vides.

Lors de la mise à jour des binômes, nous avons remarqué que les utilisateurs peuvent modifier simultanément les étudiants dans le binôme ainsi que la note rapport et la date réelle remise, ce qui n'a pas de sens pratique. Par conséquent, en plus de conserver les vérifications mentionnées ci-dessus, nous avons ajouté une fonctionnalité empêchant la modification de la note rapport et de la

date réelle remise si les étudiants du binôme sont modifiés, et vice versa. En outre, si les étudiants d'un binôme sont modifiés, la note rapport et la date réelle remise de ce binôme sont réinitialisées aux valeurs par défaut.

f. Intégration des Fonctionnalités de Note dans le Module Binôme

Initialement, pour le développement du module note, nous avions suivi la logique appliquée dans le module binôme. Cependant, nous avons rapidement réalisé que, d'un point de vue logique, l'ajout d'un nouveau binôme devrait automatiquement entraîner la création d'une entrée correspondante dans note, et que la suppression d'un binôme devrait également entraîner la suppression des notes associées. Par conséquent, nous avons décidé d'intégrer les fonctionnalités d'ajout, de suppression et de modification (à l'exception des notes de soutenance) de note directement dans le module binôme.

g. Résolution des Erreurs de Caractères Vides dans les Requêtes Binôme

Lors de l'exécution des requêtes de consultation pour le module binôme, nous avons rencontré des erreurs liées à la présence de caractères vides. En examinant le problème, nous avons découvert que cela était dû au fait que certaines données, qui ne pouvaient être renseignées dès le début, comme les notes de rapport, les notes de soutenance et la date de remise finale, étaient laissées vides lors de l'ajout des données. Pour remédier à cette situation, nous avons décidé d'insérer des valeurs par défaut pour ces données, ce qui a permis de résoudre le problème.

h. Optimisation de la Saisie des Dates dans le Module Binôme

Lors de la modification de la date de remise finale dans le module binôme, nous avons rencontré un problème lié à notre valeur par défaut fixée au 11/11/1111. Le composant datepicker se positionne automatiquement sur cette date, rendant la saisie d'une nouvelle date difficile pour les utilisateurs. Pour résoudre ce problème, nous avons ajouté un événement de clic sur le datepicker. Ainsi, lorsque l'utilisateur clique sur le composant, la date actuelle est automatiquement récupérée, facilitant grandement la saisie des nouvelles dates.

III. Organisation des tâches

Creation de BDD: Shengqi MA

Classe entity : Yingxuan LI

Couche de service(Optimisation): Yingxuan LI

Mappers:

Etudiant Mapper	Shengqi MA
Formation Mapper	Shengqi MA
Projet Mapper	Shengqi MA
Binomes Mapper	Yingxuan LI
Note Mapper	Shengqi MA

Test:

Test Etudiant	Shengqi MA
Test Formation	Shengqi MA
Test Projet	Shengqi MA
Test Binomes	Yingxuan LI
Test Note	Shengqi MA

Interface:

Interface Etudiant	Yingxuan LI
Interface Formation	Yingxuan LI
Interface Projet	Shengqi MA
Interface Binomes	Yingxuan LI
Interface Note	Shengqi MA
Optimisation de l'Interface Utilisateur	Shengqi MA

Autres fonctionnalités:

Fonctionnalité Login	Yingxuan LI

Fonction d'importation de fichiers	Yingxuan LI
Fonction de téléchargement de la page actuelle	Yingxuan LI

Documentation:

Documentation de développeur	Yingxuan LI et shengqi MA
Documentation d’utilisateur	Yingxuan LI et shengqi MA

IV. Fonctionnalités et Limitations

Fonctionnalités Implémentées

- **Recherche Floue dans les Tableaux**

  Notre application permet une recherche multicritère au sein de tous les tableaux. Les résultats affichent toutes les entrées contenant les termes de la recherche. Par exemple, il est possible de rechercher tous les étudiants dont le nom contient la lettre "a".

- **Suppression de Données**

  Les utilisateurs peuvent sélectionner et supprimer des entrées directement dans les tableaux, garantissant une interaction intuitive avec les données.

- **Ajout de Données**

  La fonctionnalité d'ajout de données est présente pour tous les tableaux, avec des vérifications en place pour assurer la validité des données. Par exemple, il est impossible d'ajouter des étudiants absents des binômes.

- **Modification de Données**

La modification des données est implémentée de deux manières :

Pour les vues Note et Projet, un double-clic sur une entrée du tableau permet de l'éditer directement.

Pour les autres vues, une fenêtre contextuelle traditionnelle est utilisée pour modifier les données.

- **Intégrité des Données**

Toutes les fonctionnalités sont conçues pour préserver l'intégrité de la base de données et la validité des données. Le programme fonctionne de manière fluide et sans bugs.

- **Interface Graphique Adaptable**

L'interface utilisateur s'adapte dynamiquement aux changements de taille de la fenêtre, offrant une expérience utilisateur cohérente et responsive.

- **Fonction de Connexion**

Un système de connexion sécurisé a été implémenté. Les utilisateurs, notamment les enseignants, doivent s'identifier avec leur identifiant et un mot de passe crypté avant d'accéder à l'application.

En cas d'oubli du mot de passe, les enseignants peuvent utiliser la fonction de récupération de mot de passe. Ils seront alors invités à répondre à des questions de sécurité définies lors de l'enregistrement initial.

Nous avons également intégré une fonctionnalité d'envoi d'e-mails, permettant d'envoyer automatiquement le nouveau mot de passe à l'adresse e-mail préalablement configurée par l'enseignant.

- **Exportation des Données**

Les utilisateurs peuvent exporter des données des tableaux en formats Excel ou PDF, facilitant le partage et l'analyse hors ligne des données.

- **Importation des Données**

  Les utilisateurs ont la possibilité d'importer des données dans le système en utilisant le format Excel. Cette fonctionnalité est conçue pour faciliter l'intégration rapide et en masse de données, comme l'ajout de listes d'étudiants ou d'autres informations pertinentes. Cette option d'importation rend le processus de mise à jour et de gestion des données plus efficace, en réduisant le besoin de saisie manuelle et en permettant une mise à jour globale des données avec simplicité et précision.

- **Fonction de la Barre de Menu**

  La barre de menu offre des raccourcis vers différentes fonctionnalités et paramètres de l'application.

- **Calcul Automatique du Score Total**

  La note finale est calculée en tenant compte des pourcentages respectifs pour le rapport et la soutenance, ainsi que des pénalités pour retard de remise du projet, avec une réduction de 0,1 point pour chaque jour de retard.

- **Alertes en cas d'Erreurs de Saisie**

  Afin d'assurer une expérience utilisateur sans faille, notre application est équipée de messages d'alerte contextuels. Ces notifications apparaissent sous forme de fenêtres pop-up pour informer l'utilisateur en cas de saisie incorrecte ou invalide et lui fournir des indications claires sur la manière de corriger les erreurs. Cela permet de prévenir les erreurs de saisie et assure que les données entrées dans l'application sont correctes avant leur soumission.

Fonctionnalité pas réussi

- **Fonction de rafraîchissement automatique après ajouter des donnée**

Après avoir ajouté et modifié de nouvelles données, vous devez appuyer manuellement sur le bouton d'actualisation pour afficher les nouvelles données ; l'actualisation automatique n'a pas pu être mise en œuvre pour des raisons de temps.

V. Optimisation

1.Optimisation du Code :

Dans le cadre de l'amélioration continue de notre architecture logicielle, nous avons effectué une optimisation majeure de la gestion des données. Initialement, toutes les opérations de récupération des données étaient centralisées dans la couche contrôleur. Cependant, pour adhérer au principe de responsabilité unique et faciliter la maintenance du code, nous avons introduit une couche de service.

Avant l'Optimisation :

Le contrôleur était chargé à la fois de la logique d'affaires et de l'interaction avec la base de données, ce qui le rendait lourd et difficile à gérer. Par conséquent, nous avons décidé de les séparer, en ajoutant une couche de service dédiée exclusivement à l'interaction avec les données.

Après l'Optimisation :

Nous avons créé une couche de service distincte, dédiée à l'interaction avec la base de données. Cette abstraction supplémentaire permet une séparation claire entre la logique d'affaires et l'accès aux données, ce qui présente plusieurs avantages :

1. Déplacement des Fonctionnalités : Toutes les fonctionnalités liées à la récupération des données ont été transférées du contrôleur vers la couche de service. Cette restructuration a permis de simplifier les contrôleurs, les rendant plus concis et plus faciles à tester.
2. Responsabilité Unique : Chaque service est conçu pour gérer un ensemble spécifique de tâches liées aux données. Par exemple, un service pour la gestion des étudiants, un autre pour les projets, etc. Cela respecte le principe de responsabilité unique et augmente la cohésion du code.
3. Testabilité et Maintenance : Cette séparation permet de tester individuellement la logique d'affaires et l'accès aux données avec plus de facilité. La maintenance du code est également simplifiée car les modifications dans la logique d'accès aux données ne nécessitent pas de changements dans la couche de présentation et vice versa.

En conclusion, l'introduction de la couche de service dans notre architecture représente un pas vers une plus grande modularité, une meilleure stabilité et une maintenance simplifiée. Elle aligne également notre projet sur les meilleures pratiques industrielles de conception de logiciels.

2.Optimisation de l'Interface Utilisateur

Dans le cadre de notre engagement continu envers l'amélioration de notre application, une optimisation significative a été réalisée sur l'interface utilisateur. Initialement, nous utilisons le modèle standard de TableView fourni par JavaFX, mais pour améliorer l'attrait et l'accessibilité de notre application, des changements étaient nécessaires.

Avant l'Optimisation :

Le modèle initial de TableView était fonctionnel mais manquait de personnalisation et d'attrait esthétique. Cette présentation basique limitait l'expérience utilisateur et l'engagement avec l'application.

Après l'Optimisation :

Nous avons apporté plusieurs améliorations clés à l'interface utilisateur :

1. Personnalisation visuelle : Nous avons redessiné l'interface de TableView pour la rendre plus moderne et visuellement attrayante. Cela inclut l'ajustement des couleurs, des polices et des bordures pour créer une expérience utilisateur plus engageante.
2. Feedback Visuel et Interactivité : L'intégration d'éléments de feedback visuel, tels que des surlignages et des animations, à améliorer l'interaction avec l'interface, rendant les actions de l'utilisateur plus claires et intuitives.

Ces optimisations ont non seulement amélioré l'esthétique de notre application mais ont également rendu l'interface plus intuitive et agréable à utiliser. Ces changements ont eu un impact positif sur l'expérience globale de l'utilisateur et ont contribué à la réussite du projet.

VI. Conclusion

En conclusion de notre projet, il est essentiel de souligner l'évolution et les acquis significatifs obtenus au cours de cette aventure. Le projet a été un véritable défi, tant sur le plan organisationnel que technique, mais il a permis de développer une compréhension approfondie des enjeux liés à la conception et au développement de logiciels complexes.

Sur le plan technique, nous avons relevé des défis majeurs, notamment dans la gestion des données avec SQLite et JAR, l'optimisation de l'interface utilisateur avec

JavaFX, et l'adoption de Mybatis comme framework de persistance. Ces choix technologiques, bien que difficiles au début en raison de notre manque d'expérience préalable, se sont avérés être des décisions judicieuses qui ont renforcé la robustesse et l'efficacité de notre application.

L'organisation des tâches entre les membres de l'équipe a également été cruciale. Malgré les difficultés initiales liées à la gestion des conflits de code et à la compatibilité inter-systèmes, nous avons réussi à instaurer un système de travail collaboratif efficace. Cela a non seulement amélioré la qualité de notre code, mais a aussi renforcé notre cohésion d'équipe et notre capacité à travailler ensemble vers un objectif commun.

Les fonctionnalités implémentées, allant de la recherche floue dans les tableaux à la gestion avancée des notes et des binômes, illustrent notre engagement à fournir une application non seulement fonctionnelle mais aussi intuitive et conviviale. L'attention portée aux détails, comme les alertes en cas d'erreurs de saisie et l'adaptabilité de l'interface, témoigne de notre volonté de créer une expérience utilisateur optimale.

En somme, ce projet a été une opportunité exceptionnelle de croissance et d'apprentissage. Les défis rencontrés et les solutions apportées ont non seulement prouvé notre capacité à surmonter les obstacles, mais ont aussi enrichi notre expertise technique et notre esprit d'équipe. Ce projet représente non seulement une réalisation technique significative, mais aussi un voyage personnel et professionnel enrichissant pour chaque membre de l'équipe.