جامعة شعيب الدكالي
Université Chouaïb Doukkali

المدرسة العليا للتكنولوجيا
Ecole Supérieure de Technologie
Sidi Bennour - سيدي بنور

Université Chouaib Doukkali
École Supérieure de Technologie
Sidi Bennour

RAPPORT DE PROJET JAVA

En vue d'obtention du Diplôme Universitaire de Technologie

Filière :

Génie Informatique

GESTION DES NOTES SCOLAIRES EXPORT/IMPORT CSV

Réalisée par :

- Khawla Jaha
- Assia Gouti
- Salma Esselmaoui

Sous la direction du :

PR. Youssef Baddi

Année Universitaire 2025– 2026

Résumé

Ce rapport présente le développement d'une application Java dédiée à la gestion des notes scolaires avec des fonctionnalités d'importation et d'exportation des données au format CSV. L'application permet d'ajouter, modifier, supprimer et consulter les notes des étudiants de manière structurée.

Le rapport commence par une présentation générale du projet et de ses objectifs, suivie d'une description des outils et technologies utilisés. Ensuite, l'architecture de l'application ainsi que les principales fonctionnalités sont détaillées. Une partie est consacrée à la gestion des fichiers CSV, expliquant le processus d'importation et d'exportation des données. Enfin, le rapport se termine par une analyse des résultats obtenus et une conclusion générale mettant en évidence les acquis et les perspectives d'amélioration.

Table des figures

Sommaire

Introduction Générale

Dans le cadre de notre formation, nous avons été amenés à réaliser un projet informatique en langage Java afin de mettre en pratique les connaissances théoriques acquises durant les cours. Ce projet consiste à développer une application de gestion des notes scolaires permettant d'enregistrer, consulter et gérer les résultats des étudiants de manière efficace et organisée.

L'objectif principal de cette application est de faciliter le suivi des notes grâce à une interface simple et fonctionnelle, tout en offrant la possibilité d'importer et d'exporter les données sous forme de fichiers CSV. Cette fonctionnalité permet d'assurer la sauvegarde des informations, leur partage ainsi que leur exploitation dans d'autres outils informatiques.

À travers ce projet, nous avons cherché à renforcer nos compétences en programmation orientée objet, en manipulation des fichiers et en structuration des données, tout en répondant à un besoin réel dans le domaine de la gestion académique.

Chapitre 1 :
Contexte du projet & analyse de besoin

Introduction

Ce chapitre a pour objectif de présenter le contexte général du projet ainsi que l'analyse des besoins nécessaires à sa réalisation. Il permet d'identifier la problématique à l'origine du projet, d'analyser les limites des solutions existantes et de définir clairement les besoins fonctionnels et non fonctionnels auxquels l'application devra répondre. Cette étape constitue une base essentielle pour orienter les choix de conception présentés dans le chapitre suivant.

1. Cadre général du projet

1.1 Contexte du projet

Dans un contexte universitaire, la gestion des notes des étudiants constitue une tâche essentielle pour assurer un suivi pédagogique efficace. Cette gestion implique la saisie des notes, leur organisation par module, ainsi que l'évaluation globale des résultats des étudiants.

Le projet proposé vise à répondre à ce besoin à travers une application permettant de centraliser les informations des étudiants et de simplifier les opérations liées à la consultation et à la gestion des notes.

1.2 Étude de l'existant

Actuellement, le suivi des notes est souvent réalisé de manière manuelle ou à l'aide d'outils bureautiques tels que les tableurs.
Bien que ces solutions soient largement utilisées, elles présentent plusieurs limites :

- Risque élevé d'erreurs de saisie
- Difficulté de gestion lorsque le nombre d'étudiants augmente
- Absence d'automatisation pour certaines opérations, comme le calcul des moyennes
- Manque de fonctionnalités avancées pour la recherche, la modification ou la suppression des données

Ces limites montrent la nécessité d'une solution plus structurée et automatisée.

1.4 Solution proposée

Afin de remédier aux problèmes identifiés, une application dédiée est proposée.
Cette solution vise à offrir un moyen simple et organisé de gérer les informations des étudiants et leurs notes, tout en garantissant une meilleure fiabilité des données et une facilité d'utilisation.

2. Spécification des besoins

2.1 Besoins fonctionnels

Les besoins fonctionnels décrivent les services que l'application doit fournir pour répondre aux attentes des utilisateurs :

Gestion des étudiants :

- Ajouter un étudiant
- Afficher la liste des étudiants

Gestion des notes :

- Associer des notes aux modules suivis par chaque étudiant
- Calculer automatiquement la moyenne des étudiants

Recherche et gestion :

- Rechercher un étudiant par son nom
- Modifier ou supprimer les informations d'un étudiant

Gestion des données :

- Sauvegarder les informations
- Charger les données existantes lors du lancement de l'application

2.2 Besoins non fonctionnels

Les besoins non fonctionnels définissent les contraintes et critères de qualité de l'application :

- **Simplicité et ergonomie** : interface claire et facile à utiliser
- **Robustesse** : gestion des erreurs de saisie (notes invalides, données manquantes)
- **Portabilité** : application exécutable sur tout environnement disposant d'un JDK
- **Fiabilité** : conservation des données entre différentes utilisations

Conclusion:

Ce chapitre a permis de présenter le contexte général du projet et d'analyser les besoins liés à la gestion des notes des étudiants. L'étude de l'existant a mis en évidence les limites des solutions traditionnelles, justifiant ainsi la nécessité d'une application dédiée. L'identification des besoins fonctionnels et non fonctionnels a permis de définir clairement les attentes et les objectifs du système à développer. Ces éléments constituent une base solide pour la phase de conception, qui sera abordée dans le chapitre suivant.

Chapitre2 :

Conception d'application

Introduction

Ce chapitre est consacré à l'analyse et à la conception du système de gestion des notes des étudiants. Il vise à modéliser les fonctionnalités et la structure de l'application à l'aide du langage UML, afin de représenter de manière claire les interactions entre l'utilisateur et le système, ainsi que l'organisation interne des différentes composantes. Cette étape permet de traduire les besoins identifiés dans le chapitre précédent en modèles conceptuels facilitant la phase de réalisation.

1. UML comme outil de modélisation

1.1. Présentation

UML (Unified Modeling Language) est utilisé dans ce projet comme outil de modélisation pour représenter les fonctionnalités principales de l'application et les relations entre ses classes. Il permet de structurer le système et de faciliter la transition entre l'analyse des besoins et la phase de réalisation. Les diagrammes ont été conçus à l'aide de l'outil StarUML.

2. Modélisation UML

2.1. Le diagramme de cas d'utilisation

Le diagramme des cas d'utilisation présente les fonctionnalités principales de l'application de gestion des notes des étudiants et l'interaction avec l'acteur principal , l'Administrateur.

L'administrateur a la possibilité de gérer les étudiants en important ou exportant des fichiers CSV, en ajoutant de nouveaux étudiants, et en affichant la liste complète des étudiants.

Il peut également gérer les notes, c'est-à-dire visualiser les notes existantes et les modifier si nécessaire.

Ce diagramme permet de comprendre rapidement les fonctionnalités offertes par le système et comment l'administrateur peut interagir avec chacune d'elles pour assurer la gestion efficace des informations des étudiants et de leurs résultats académiques.

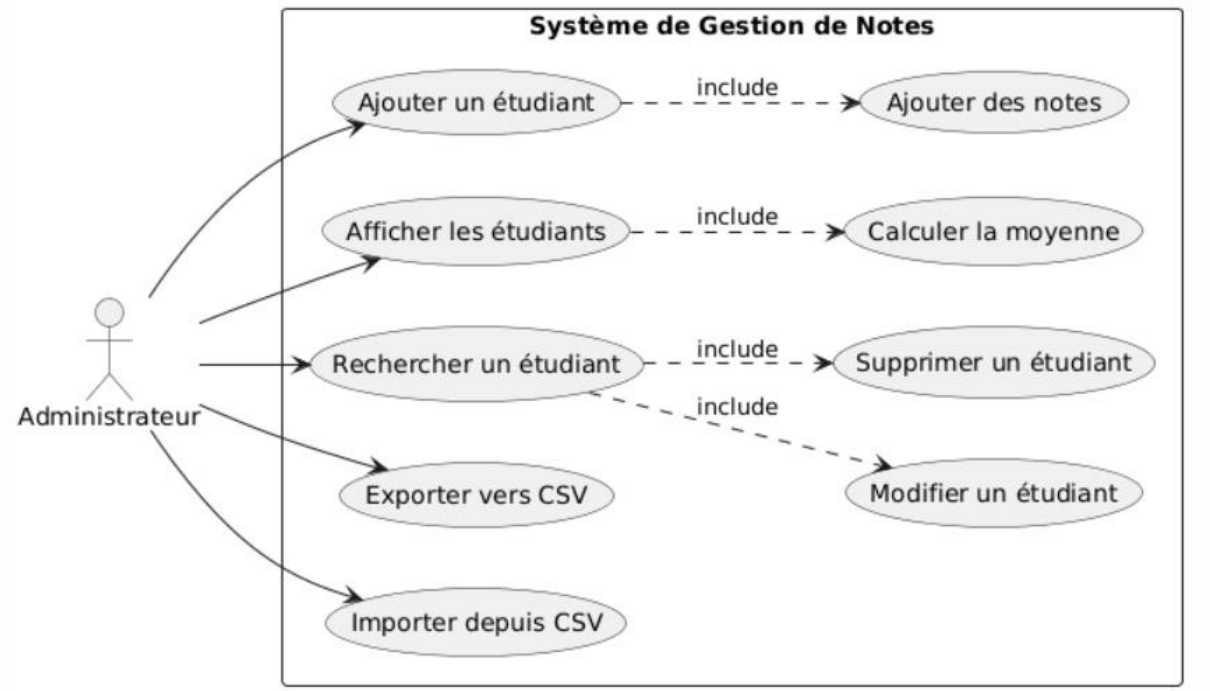

Figure 1: Diagramme de cas d'utilisation

2.2. Le diagramme de classe

Le diagramme de classes représente la structure statique de l'application. Il met en évidence les principales classes du système, leurs attributs et les relations existantes entre elles. Ce diagramme facilite la compréhension de l'organisation interne du système et sert de référence pour l'implémentation.

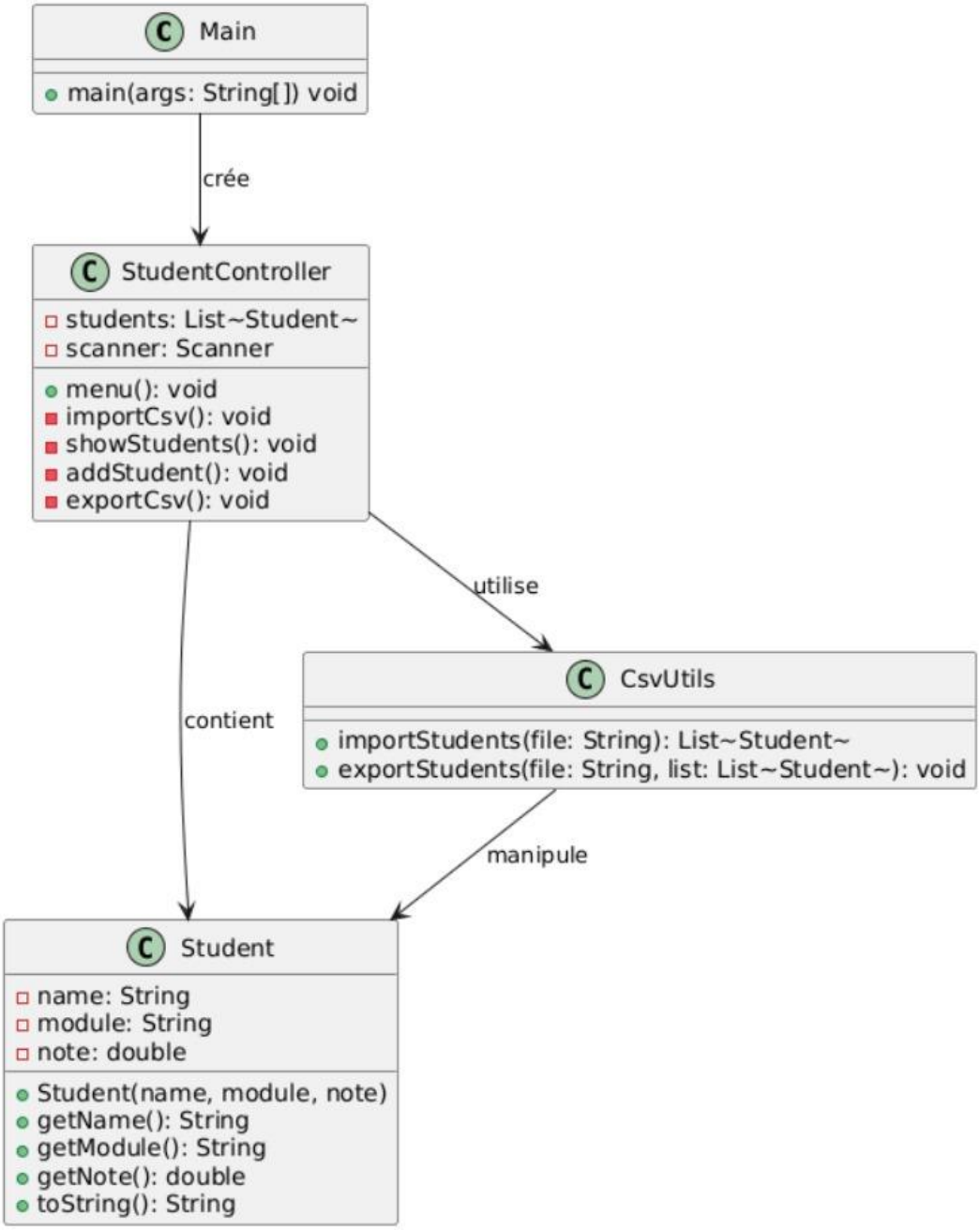

Figure 2 : Diagramme de classe

2.3. Les diagrammes de séquences

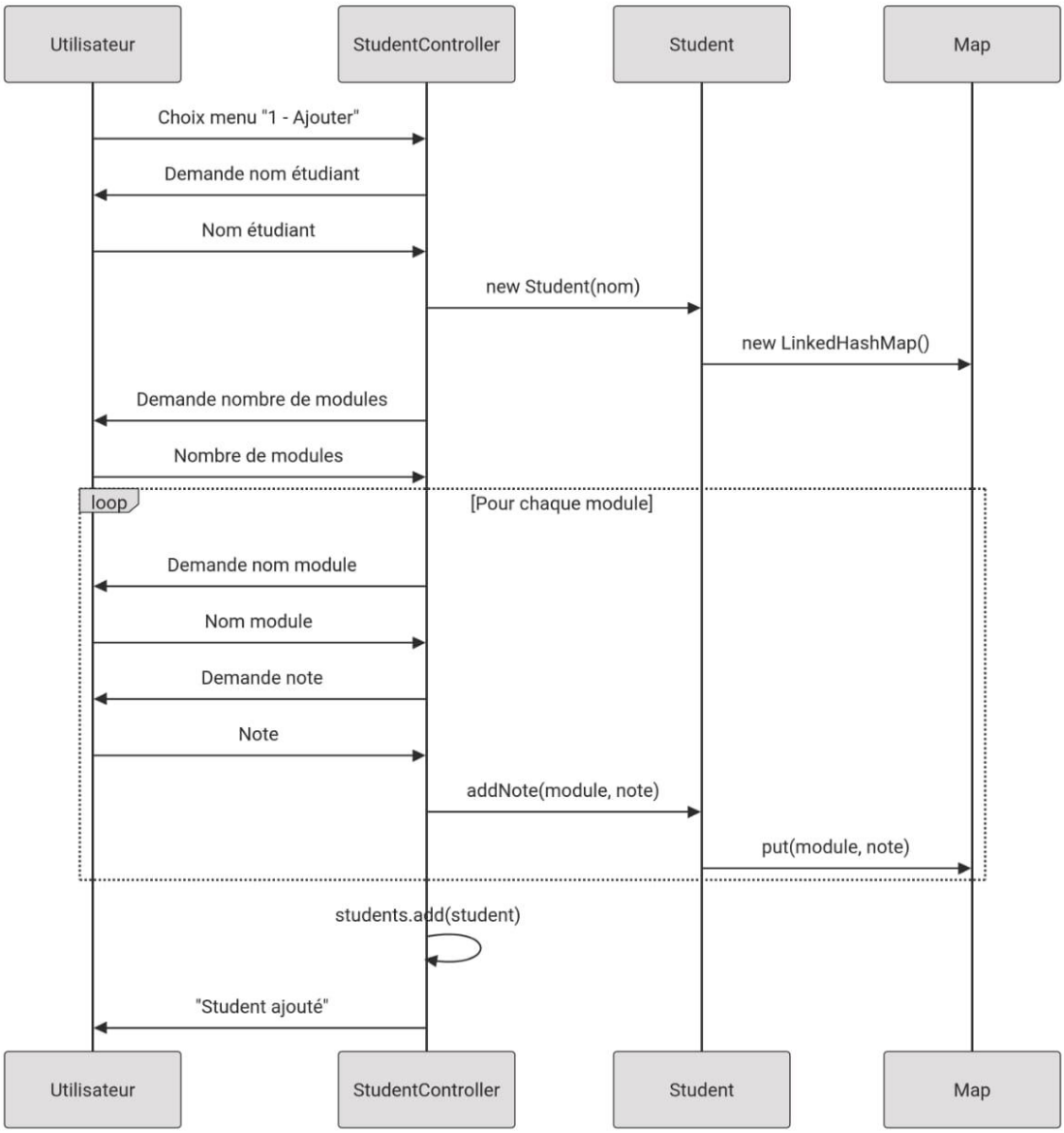

Figure 3 : Diagramme de séquence – Ajout étudiant

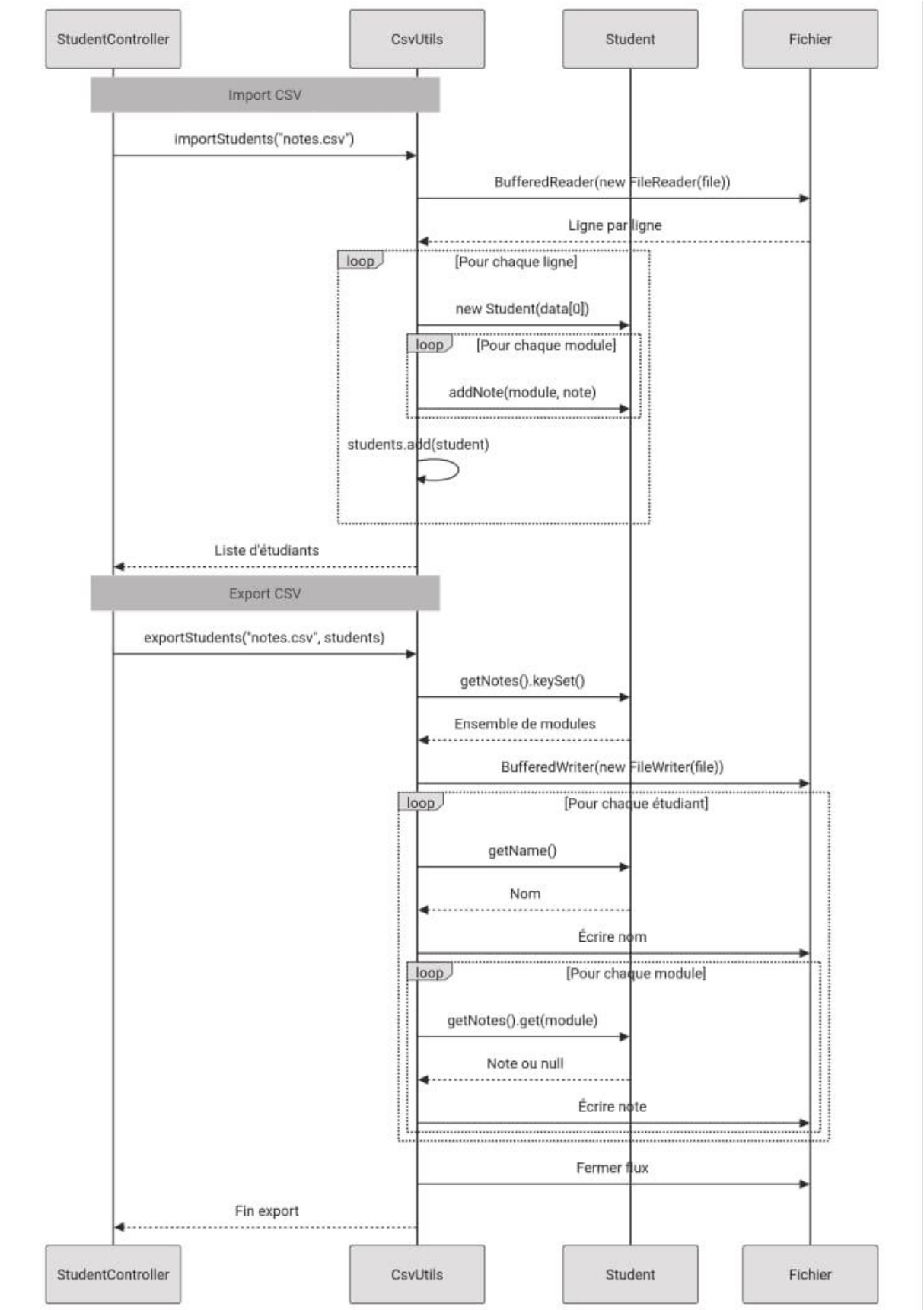

Figure 4 : Diagramme de séquence – Import/Export CSV

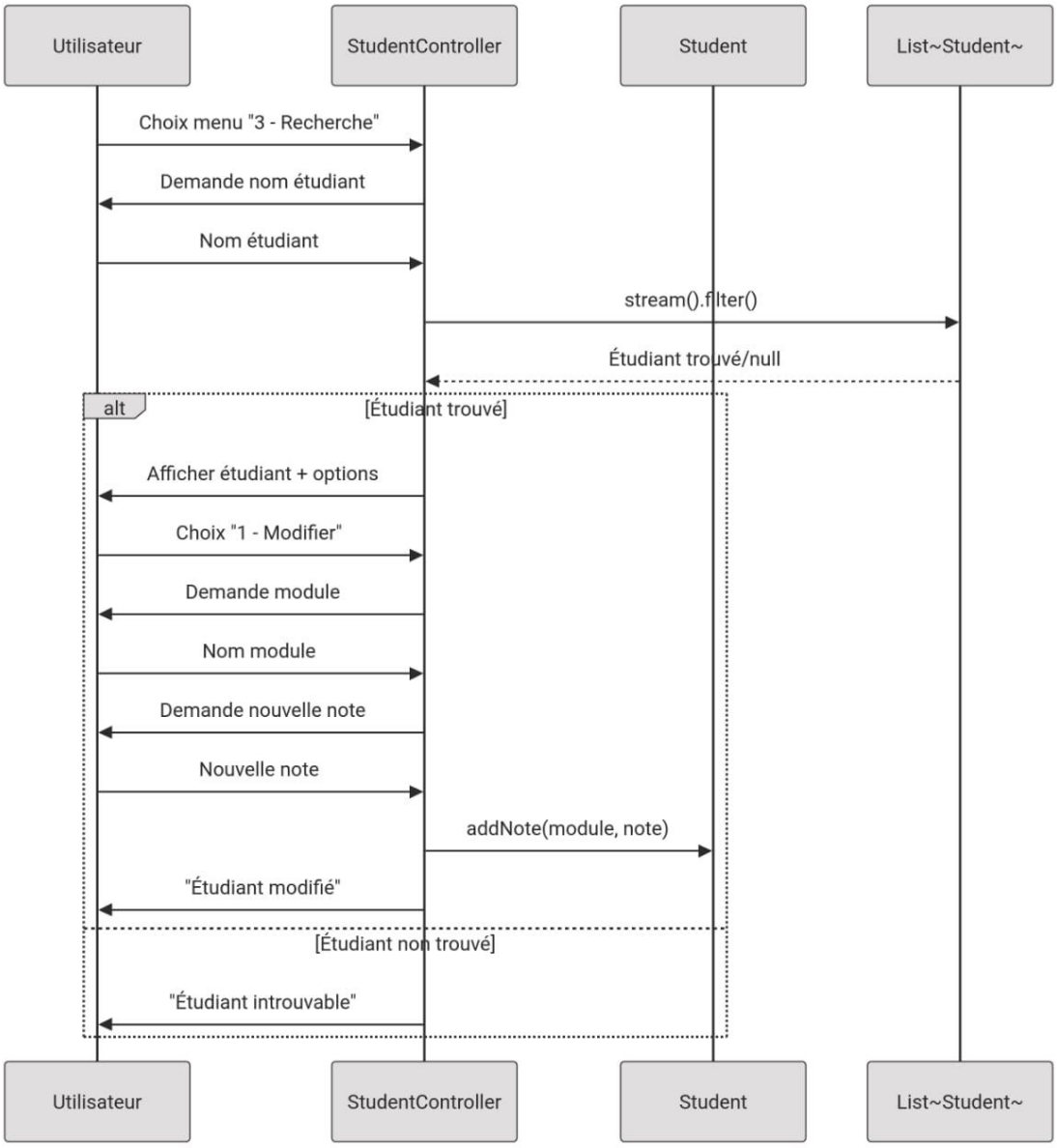

Figure 5 : Diagramme de séquence – Recherche et Modification

Conclusion:

Ce chapitre a permis de modéliser le système de gestion des notes des étudiants à l'aide du langage UML. Les différents diagrammes présentés ont facilité la compréhension des fonctionnalités du système et de sa structure globale. Cette étape de conception constitue une base essentielle pour la phase de réalisation, qui sera présentée dans le chapitre suivant.

Chapitre 3 :

Réalisation et Résultats

Introduction

Ce chapitre est consacré à la phase de réalisation de l'application de gestion des notes des étudiants. Il présente l'environnement de développement utilisé, l'implémentation des différentes fonctionnalités définies dans les chapitres précédents, ainsi que les résultats obtenus après les tests et l'exécution de l'application.

1. Environnement de développement

La réalisation de ce projet a été effectuée en utilisant les outils et technologies suivants :

- **Langage de programmation** : Java
- **Environnement de développement (IDE)** : IntelliJ IDEA / Eclipse
- **Gestion des données** : fichiers CSV
- **Système d'exploitation** : Windows
- **Architecture** : MVC simplifiée

L'utilisation des fichiers CSV permet de stocker les données de manière persistante sans recourir à une base de données, ce qui rend l'application légère et facile à manipuler.

2. Architecture de l'application

L'application est organisée selon une architecture modulaire inspirée du modèle MVC :

- **Model** : représente les données métier à travers la classe *Student*, qui contient le nom de l'étudiant, la liste des modules et les notes associées.
- **Controller** : la classe *StudentController* gère les interactions avec l'utilisateur, le menu principal et la logique de gestion des étudiants.
- **Utility** : la classe *CsvUtils* assure l'importation et l'exportation des données vers le fichier CSV.
- **Main** : point d'entrée de l'application.

Cette organisation permet une meilleure lisibilité du code et facilite les évolutions futures.

3. Réalisation des fonctionnalités

3.1 Chargement automatique des données (Import CSV)

Au démarrage de l'application, les données des étudiants sont automatiquement chargées depuis un fichier CSV. Cette fonctionnalité permet de restaurer l'état précédent de l'application sans intervention manuelle de l'utilisateur et garantit la persistance des données entre les différentes exécutions.

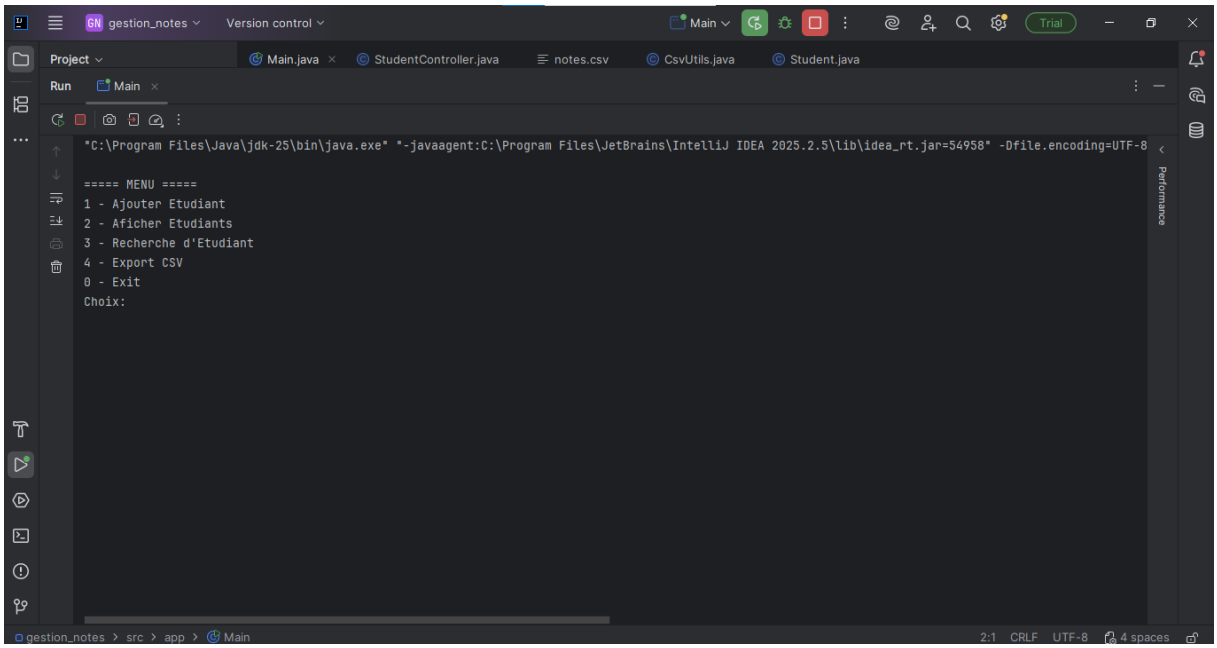

Figure 6 : Lancement de l'application

3.2 Ajout d'un étudiant

L'application permet d'ajouter un nouvel étudiant en saisissant son nom, puis les modules suivis avec leurs notes respectives.
Un contrôle de saisie est appliqué afin de garantir que les notes sont comprises entre **0 et 20**, ce qui assure la validité des données enregistrées.

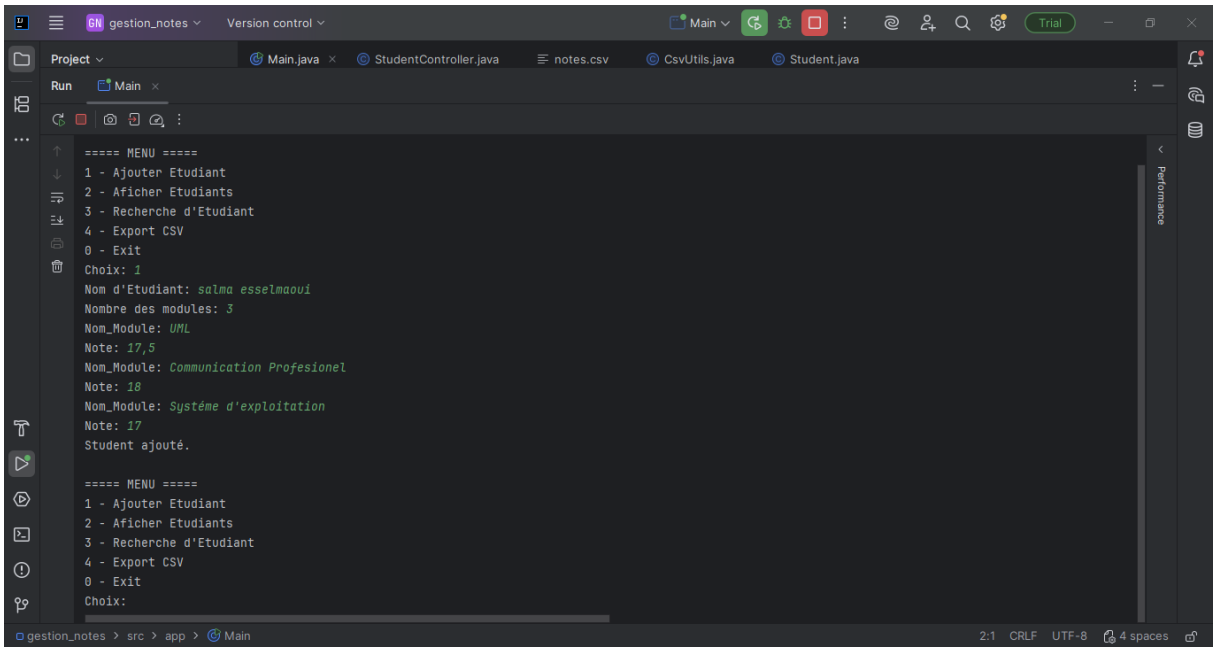

Figure 7 : Ajout des étudiants

3.3 Gestion des modules et des notes

Chaque étudiant peut avoir plusieurs modules. Pour chaque module, une note est associée.
Cette approche permet une gestion réaliste des résultats académiques et reflète le fonctionnement réel d'un système universitaire.

3.4 Calcul automatique de la moyenne

La moyenne générale de chaque étudiant est calculée automatiquement à partir des notes obtenues dans les différents modules.
Cette moyenne est affichée lors de la consultation des informations de l'étudiant, ce qui facilite l'évaluation de ses performances.

3.5 Affichage des étudiants

L'option « Afficher Etudiants » permet d'afficher :

- le nom de l'étudiant
- la liste des modules
- les notes obtenues
- la moyenne générale

Cette fonctionnalité offre une vue claire et complète sur les résultats de chaque étudiant.

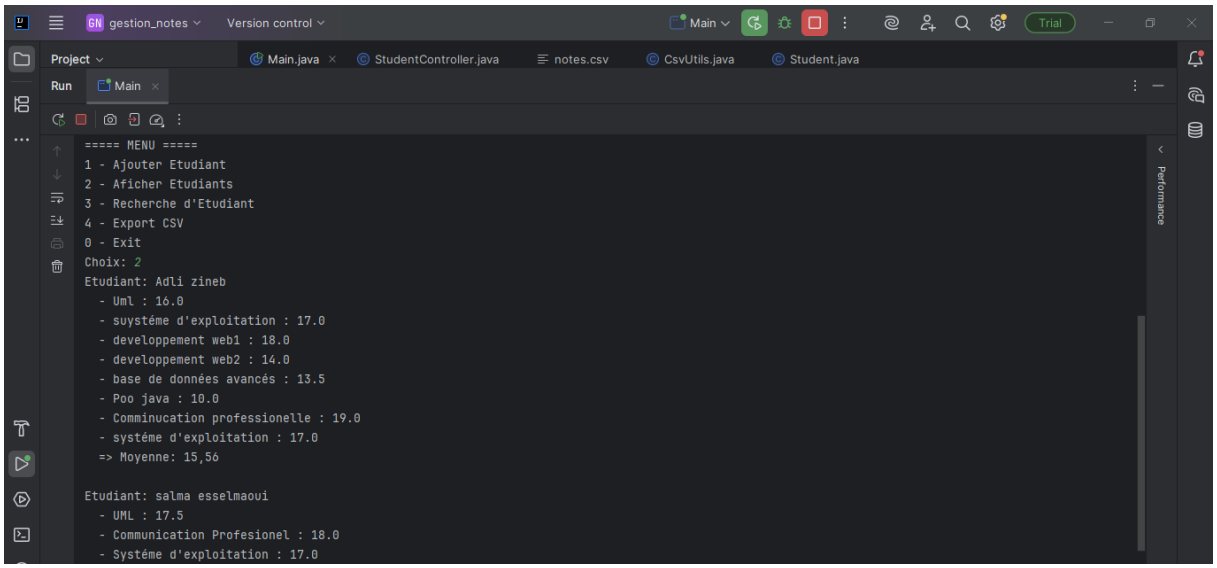

Figure 8: Affichage des étudiants

3.6 Recherche, modification et suppression d'un étudiant

L'application intègre une fonctionnalité de recherche permettant de retrouver un étudiant à partir de son nom.
Après la recherche, l'utilisateur peut :

- modifier les notes ou les modules de l'étudiant
- supprimer définitivement l'étudiant du système

Ces opérations assurent une gestion flexible et dynamique des données.

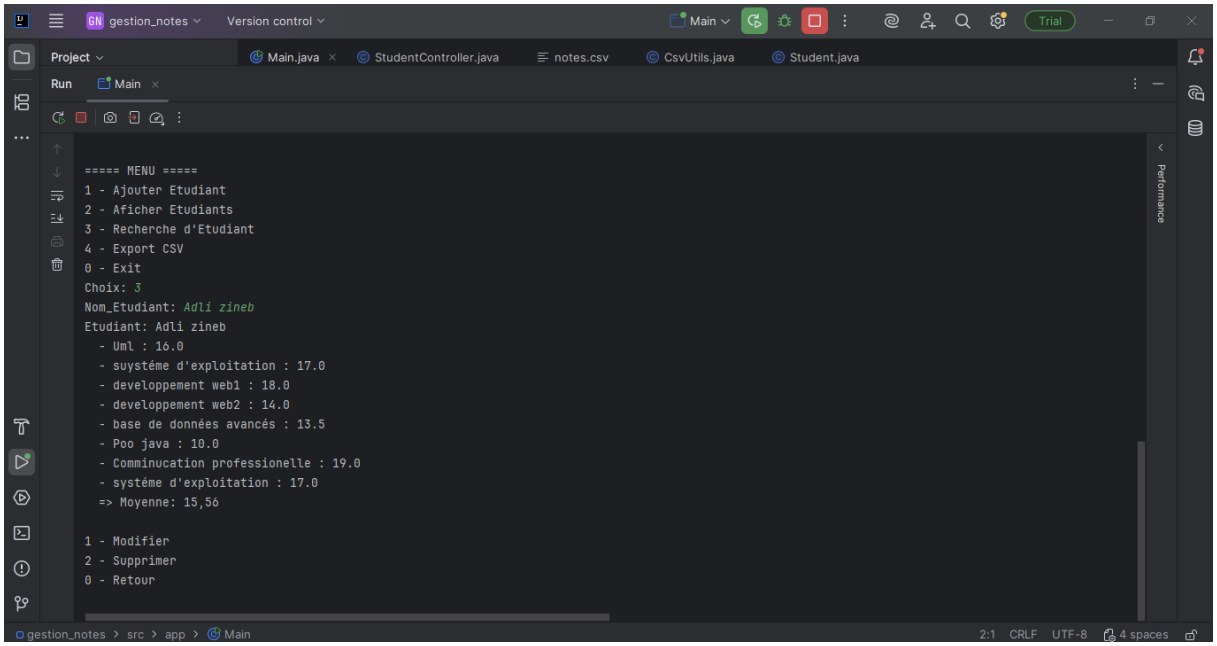

Figure 9: Recherche, modification et suppression d'un étudiant

3.7 Exportation des données vers un fichier CSV

Les données mises à jour peuvent être exportées vers un fichier CSV.

Ce fichier contient la liste des étudiants, leurs modules et leurs notes, ce qui permet de conserver les modifications et de les réutiliser ultérieurement.

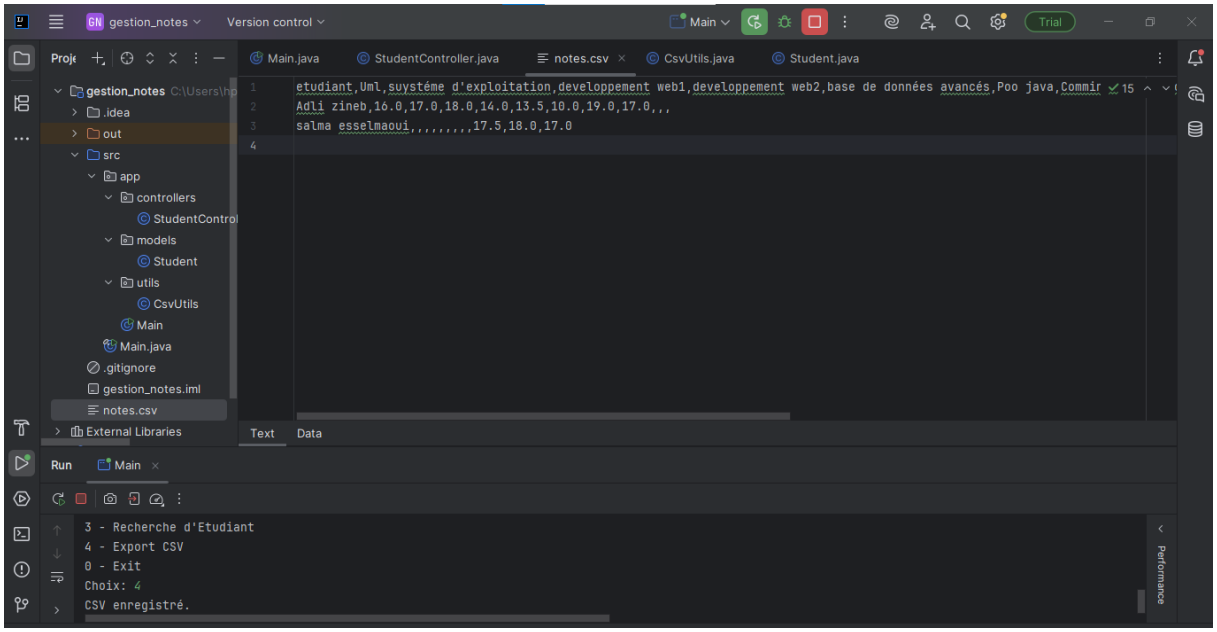

Figure 10: Exportation des données vers un fichier CSV

Conclusion:

Ce chapitre a présenté la réalisation de l'application ainsi que les résultats obtenus. Les fonctionnalités développées permettent une gestion complète et cohérente des notes des étudiants.

Dans la suite du travail, des améliorations pourraient être envisagées, telles que l'ajout d'une interface graphique ou l'utilisation d'une base de données relationnelle.

Conclusion Général

La réalisation de ce projet de gestion des notes en Java nous a permis de consolider nos connaissances en programmation et d'acquérir une expérience pratique dans le développement d'une application complète. L'intégration des fonctionnalités d'importation et d'exportation CSV a apporté une valeur ajoutée importante en facilitant la gestion, la sauvegarde et le transfert des données.

Ce projet nous a également permis de mieux comprendre l'importance de l'organisation du code, de la gestion des fichiers et de la fiabilité des données. Malgré certaines difficultés rencontrées, notamment lors de la manipulation des fichiers CSV, les objectifs fixés ont été atteints avec succès.

En perspective, plusieurs améliorations peuvent être envisagées, telles que l'ajout d'une interface graphique plus avancée, la gestion des utilisateurs ou encore l'intégration d'une base de données. Ce projet constitue ainsi une base solide pour de futures évolutions et pour le développement d'applications plus complexes.