Projet DA50
- Rapport technique -

InvestMate

Tuteur: M.Galland

Robert Hakobyan
Arthur Aquilano
Hugo Pereira
Thomas Pierron
Adam Bin Azmi
Abderrahman Rezki

- Table des matières -

Lien vers le GitHub: **InvestMate-DA50**

Introduction

Dans le cadre de notre formation en ingénierie informatique à l'UTBM, nous avons réalisé un projet au sein de la filière Développement Avancé (DA). Cette filière forme les étudiants aux pratiques modernes et efficaces du développement logiciel en utilisant des outils adaptés. L'objectif principal de ce projet est de mettre en œuvre les bonnes pratiques enseignées en cours.

Présentation du projet

Le projet, proposé par M. Hakobyan et supervisé par M. Galland, consiste à développer une application de bureau en Java permettant aux utilisateurs de suivre des cours sur l'investissement en bourse.

Ce sujet répond à des problématiques actuelles telles que l'optimisation de la gestion financière, un enjeu pertinent pour de futurs ingénieurs. De plus, ce projet couvre un large éventail de compétences, notamment le développement Java, l'utilisation de frameworks comme Hibernate, et la gestion de tokens de connexion. Ces aspects ont motivé notre choix.

Ce rapport détaille le travail accompli, les choix réalisés, les difficultés rencontrées, ainsi que les compétences acquises.

- La première partie présente l'appropriation du projet, la rédaction du cahier des charges incluant les choix technologiques.
- La deuxième partie décrit la conception de l'architecture logicielle, les étapes de réalisation, et les principaux défis rencontrés.
- Enfin, la dernière partie revient sur les enseignements tirés et les améliorations à apporter pour potentiellement rendre l'application publique.

Remerciements

Nous tenons à exprimer notre profonde gratitude envers toutes les personnes qui ont contribué à la réalisation de ce projet. Ce travail n'aurait pas été possible sans le soutien et les encouragements de nombreuses personnes.

Tout d'abord, nous remercions sincèrement notre superviseur de projet, M. GALLAND, pour sa précieuse guidance et sa disponibilité tout au long de ce projet. Ses conseils avisés ont été essentiels pour atteindre nos objectifs.

Nos remerciements vont également à l’ensemble du corps enseignant de la filière Développement Avancé. En partageant votre savoir et vos expériences vous nous avez permis de réaliser ce projet dans les meilleures conditions, ainsi que votre participation à l’étude et l’évaluation de ce projet au terme de ce semestre de travail.

Nous sommes assurés que les connaissances acquises et les compétences développées lors de ce projet serviront de base solide pour nos futures carrières professionnelles. Nous sommes fiers du travail accompli et reconnaissants envers tous ceux qui ont rendu cela possible.

Merci à tous.

Robert, Arthur, Hugo, Thomas, Adam et Abderrahman.

Rédaction du cahier des charges

L'objectif principal de l'UE projet de Développement Avancée est de penser et concevoir une solution informatique en adoptant des méthodologies professionnelles. Ainsi, il est essentiel de planifier le projet en segments couplés à des objectifs et des dates limites, définir des rôles, choisir les outils et technologies utilisés, anticiper les difficultés et prévoir le budget potentiel que nécessiterait ce projet s'il était réalisé dans une véritable entreprise. Toutes ces étapes ont été réalisées en amont du développement du projet, et consignées dans un cahier des charges.

NB: Le cahier des charges est disponible dans la branche Documentation du Github

Comprendre les attentes

La première étape a consisté à identifier précisément les attentes des parties prenantes. Pour cela, nous avons utilisé trois approches:

1. Discussion avec M. Hakobyan : Nous avons échangé avec le porteur du projet pour comprendre sa vision, ses attentes, et les fonctionnalités qu'il souhaitait intégrer à l'application.
2. Entretien avec M. Galland : En tant que superviseur et examinateur principal, son retour nous a permis de mieux appréhender les critères d'évaluation, le niveau de finalisation attendu et de bénéficier de ses conseils.
3. Analyse des ressources numériques : La lecture des ressources numériques (sur Teams et Moodle) pour clarifier les modalités d'évaluation, les livrables attendus, et les délais imposés.

À l'issue de ces échanges, nous avons établi une première version générale du cahier des charges. La prochaine étape a été de structurer et préciser les éléments du projet.

Formalisation des besoins

Un projet informatique doit essentiellement répondre aux attentes des utilisateurs tout en restant évolutif et modulable. Les exigences initiales des clients évoluent souvent au cours du développement, à mesure que leurs besoins se précisent. Bien que notre projet soit relativement simple sur cet aspect, nous avons pris soin de formaliser les besoins des utilisateurs finaux. Cela nous a permis d'obtenir une vision claire du fonctionnement *end-to-end* (de bout en bout) de l'application, tout en prévoyant la possibilité de modifier ou d'ajouter des fonctionnalités sans impact majeur.

Nous avons structuré ces besoins à l'aide :

- De scénarii utilisateurs (cf. Annexe 1), qui décrivent les interactions principales avec l'outil.
- D'un diagramme UML des cas d'utilisation, illustrant les fonctionnalités clés.

Ces outils nous ont aidés à hiérarchiser les besoins, à distinguer les fonctionnalités prioritaires pour la première échéance de celles pouvant être ajoutées dans des versions futures, et à établir une feuille de route préliminaire.

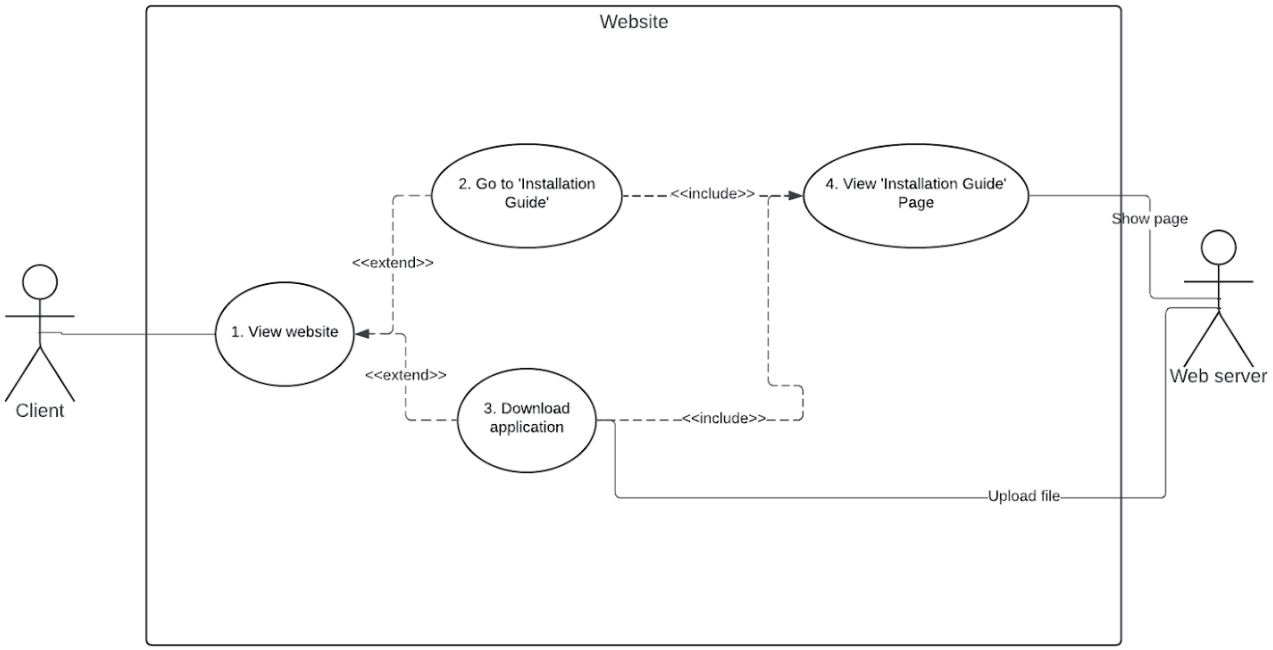

Diagramme des cas d'utilisation du site web (également disponible en grand format sur le Github, branche Documentation)

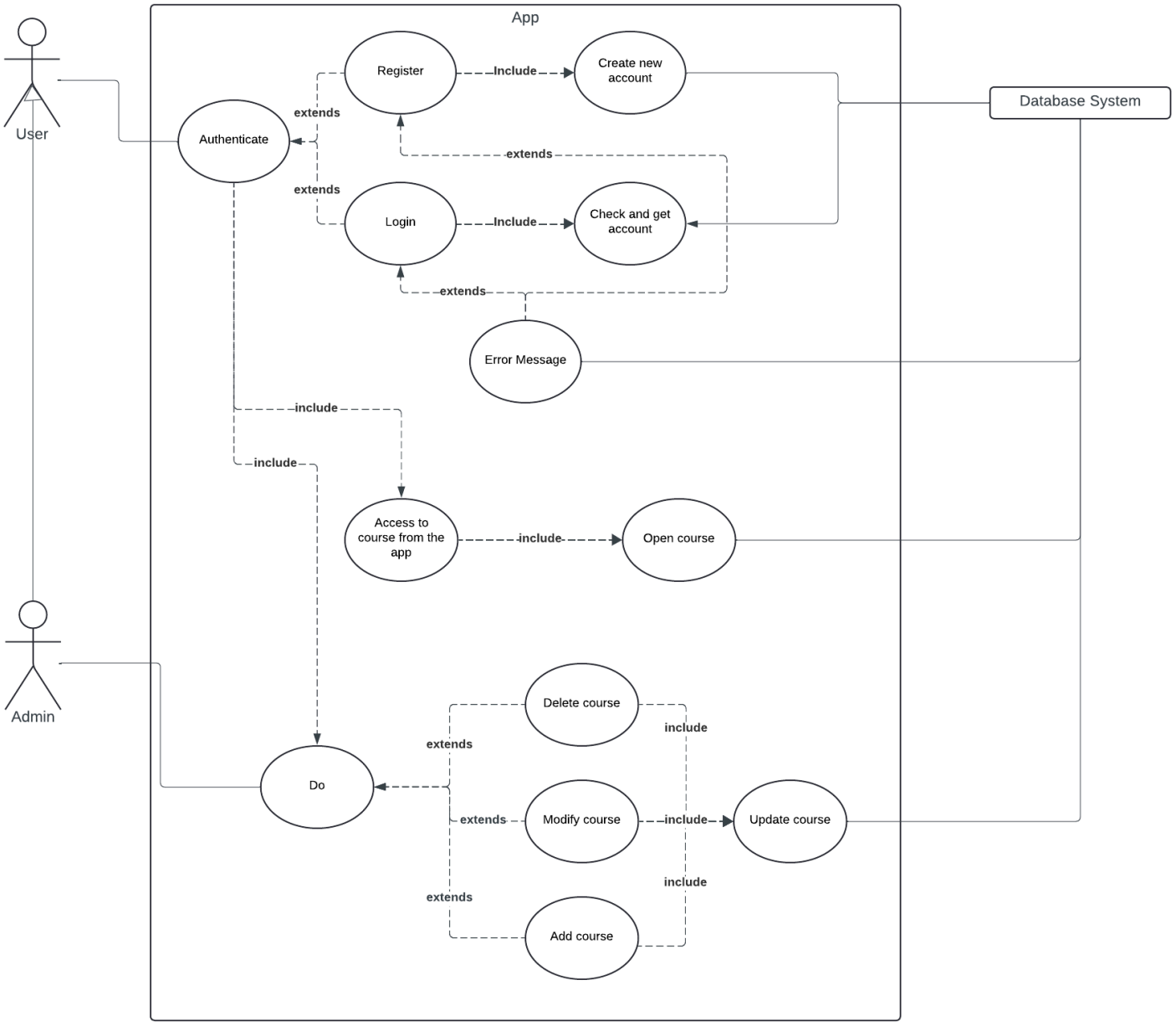

Diagramme des cas d'utilisation de l'application (également disponible en grand format sur le Github, branche Documentation)
(L'absence des numéros et des quiz est dû à la perte de la version finale à cause de la fin de la période d'essaie de l'outil)

Ainsi, voici comment nous planifions le fonctionnement de bout en bout de l'application:

Un utilisateur arrive sur le site internet, clique sur le bouton Download présent sur la première page ou sur la page *Installation Guide*. Le téléchargement se lance automatiquement (ou possibilité de forcer le téléchargement automatique).

Une fois installé, l'utilisateur lance l'outil et arrive sur un menu de connexion. S'il s'agit d'un nouveau client, il doit créer un compte via le formulaire de création de compte, directement accessible dans le menu d'authentification, qui demande un nom d'utilisateur (unique), une adresse mail et un mot de passe. En cas de réussite, le rôle "Lecteur" sera attribué à l'utilisateur. Puis on revient sur la page de connexion, l'utilisateur s'identifie et arrive sur le menu principal de l'application.

Sur le menu principal sont présents plusieurs éléments: Un bouton *Profile* pour changer ses informations, un bouton *Quiz* pour accéder à la liste des quiz, un bouton *Logout* pour se déconnecter, et la liste de l'ensemble des leçons.

En tant qu'utilisateur avec le rôle "Lecteur", il a la possibilité de consulter les leçons, quiz, et de changer ses informations personnelles.

Si un utilisateur avec le rôle "Admin"navigue sur l'outil, il verra différents boutons supplémentaires, comme ceux liés à la création de leçon et de quiz (respectivement sur le menu principal et le menu des quiz), et les boutons de modifications des quiz (présents sur chaque élément leçon ou quiz affiché).

La création de leçon fonctionne de la manière suivante: Menu spécifique où l'utilisateur choisit un titre, sélectionne un type d'élément à afficher parmi Sous-titre, Paragraphe, Image et Vidéo, renseigne les éléments à intégrer, puis sauvegarde la leçon pour la publier.

La création de quiz fonctionne de la manière suivante: Menu spécifique où l'utilisateur choisit si le quiz est un QCM à bonne(s) réponse(s) unique ou multiples. Il ajoute une question, les réponses possibles, sélectionne les bonnes réponses et sauvegarde le quiz pour le publier.

Choisir les technologies

Dès le début, le choix de Java comme langage principal était imposé par les consignes de l'UE. Cependant, le projet ne se limite pas à un simple programme. Il nécessite une réflexion approfondie sur plusieurs aspects techniques, notamment :

- L'intégration d'outils de gestion et d'automatisation adaptés aux projets Java d'envergure.
- Le stockage des données utilisateur et des cours, avec potentiellement une solution de persistance.
- La sécurisation de l'authentification et des interactions utilisateur.
- La sélection des bibliothèques pour la conception de l'interface utilisateur.

Ces choix technologiques influencent la direction globale du projet, son coût potentiel, les compétences nécessaires et les contraintes qu'ils imposent. Chaque technologie retenue ouvre des possibilités tout en limitant d'autres options.

Voici les technologies sélectionnées (et éventuellement ajustées) conformément au cahier des charges :

Site internet

- Frontend : Vite, React.js et Tailwind CSS.
    - *Vite* garantit une construction rapide et optimisée.
    - *React.js* permet des composants dynamiques et interactifs.

    - *Tailwind CSS* propose une approche utilitaire pour un design moderne et réactif.
- Environnement local : MAMP ou XAMPP pour le développement et le test du site.

Application (Frontend)

- Technologies : Java et JavaFX.
    - L'application de bureau a été privilégiée pour permettre une utilisation hors ligne des cours et une gestion de la sécurité indépendante d'une infrastructure web.

Base de données (Backend)

- Technologies :
    - J2EE pour la gestion des utilisateurs, de l'authentification et des cours.
    - Hibernate pour l'interaction avec la base de données.
    - MySQL comme système de gestion relationnelle, dédié au stockage des informations utilisateur, des cours, des configurations des bots de trading, et des données d'historique.

Sécurité

- Système d'authentification par nom d'utilisateur et mot de passe (6 caractères ASCII minimum).
- Utilisation du protocole HTTPS pour sécuriser les échanges.
- Génération et vérification de tokens JWT (JSON Web Tokens) pour chaque requête.
- Gestion des permissions selon deux rôles distincts :
    - *Lecteur* : accès limité.
    - *Admin* : rôle étendu, incluant les droits du lecteur.
- Fonctionnalités additionnelles :
    - Journalisation des accès et activités utilisateur.
    - Déconnexion automatique après inactivité.
    - Historique des connexions, des cours suivis, et des temps de connexion.

Planification, budgétisation et anticipation des risques

La rédaction du cahier des charges s'est conclue par une analyse approfondie des risques et contraintes, afin de définir un plan d'action clair et d'estimer les coûts potentiels du projet dans un contexte professionnel.

Analyse des risques

Les risques identifiés sont regroupés en quatre catégories :

- Humains : fluctuations de la charge de travail étudiante, coordination entre les membres de l'équipe.
- Technologiques : limites des outils gratuits, compatibilité des technologies choisies.

- Environnementaux : contraintes de temps liées aux échéances académiques.
- Matériels : ressources limitées (infrastructure de test et développement).

Des solutions potentielles ont été proposées pour anticiper ces risques, minimiser leur impact et garantir un déroulement optimal.

Planification et budget

Un calendrier a été établi pour structurer les étapes clés du projet, comprenant :

1. La définition des besoins et de l'architecture.
2. Le développement des modules principaux.
3. Les tests et validations.
4. La livraison et l'évaluation.

Bien que les coûts estimés restent purement théoriques (le projet étudiant se limitant à des outils gratuits et des données à petite échelle), cette analyse permet d'établir une base comparative avec la réalité des résultats obtenus.

Une fois le cahier des charges rédigé, validé et compris par toutes les parties prenantes, nous avons pu entamer la phase de réalisation en suivant la planification établie.

Réalisation du projet

Cette section présente le déroulement du développement du projet, en abordant successivement :

- La conception de l'architecture logicielle.
- La répartition des tâches.
- Une description détaillée de l'application et de son fonctionnement.

Conception du squelette de l'application

La première étape a consisté à définir l'architecture générale de l'application en s'appuyant sur un diagramme de classes. Ce diagramme, disponible sur le dépôt GitHub, a permis de :

- Visualiser clairement les classes nécessaires et leurs relations.
- Déterminer les dépendances et les interactions entre les composants.
- Structurer les fonctionnalités à développer selon un modèle standardisé, en l'occurrence le **modèle MVC** (Modèle - Vue - Contrôleur).

NB: Le diagramme des classes est disponible dans la branche Documentation du Github

Application du modèle MVC
Le modèle MVC a été choisi pour sa clarté et sa modularité :

- **Modèle** : représente les entités principales du projet (par ex. : *Lesson*, *Title*, *Paragraph*, *User*). Ces éléments centralisent les données partagées entre les vues et la base de données.
- **Vue** : constitue l'interface utilisateur, rassemblant les fonctionnalités accessibles. Les vues communiquent avec les contrôleurs pour mettre à jour les données des modèles et les transmettre à la base.
- **Contrôleur** : joue un rôle d'intermédiaire. Il vérifie les données envoyées par les utilisateurs via les vues, agit sur les modèles et interagit avec la base de données. Les contrôleurs garantissent également l'intégrité et la conformité des données affichées dans les vues après récupération depuis la base.

Cette organisation modulaire garantit une séparation nette des responsabilités, facilitant la maintenance et l'évolution de l'application.

Structure globale de l’application

L’application a été divisée en trois modules principaux :

1. **Authentification**
    - Fonctionnalités :
        - Connexion et création de compte.
        - Récupération de mot de passe oublié.
        - Chiffrement des données sensibles dans la base de données.
        - Gestion d’un token d’authentification pour permettre un accès hors ligne sécurisé.
2. **Gestion des leçons**
    - Fonctionnalités pour les comptes *Admin* :
        - Création et modification des leçons, composées de titres, sous-titres, paragraphes, introduction, conclusion, ainsi que l’intégration d’images et vidéos.
    - Fonctionnalités pour tous les utilisateurs :
        - Lecture des leçons disponibles.
3. **Gestion des quiz**
    - Fonctionnalités pour les comptes *Admin* :
        - Création et modification des quiz, comprenant des questions à choix multiples (QCM) avec une ou plusieurs réponses possibles.
    - Fonctionnalités pour tous les utilisateurs :
        - Réalisation des quiz disponibles.

Coordination du travail en équipe

La division de l’application en modules distincts a permis une répartition efficace des tâches entre les membres de l’équipe. Chaque groupe a travaillé en parallèle sur :

- La gestion des données dans la base de données.
- Les fonctionnalités propres à chaque module.
- L’interface graphique.

Cette organisation a favorisé une progression cohérente et simultanée sur l’ensemble des aspects du projet, tout en minimisant les interférences entre les tâches individuelles.

La description détaillée des contributions individuelles et du déroulement des étapes sera abordée dans la section suivante.

Répartition des rôles

Nous sommes un groupe de 6, donc pour éviter de travailler en même temps sur la même chose, simplifier les échanges, et faire avancer le projet au rythme souhaité, il nous a fallu répartir les tâches:

Les tâches sur l'analyse des besoins ont été faites par tous les membres du groupe. Puis:

- Adam s'est dans un premier temps occupé du site internet car il maîtrisait déjà cette partie du projet puis il s'est occupé de la partie création, gestion et réalisation des quiz.

- Abderrahman s'est directement occupé de la partie création, gestion et réalisation des quiz.
- Robert s'est dans un premier temps concentré sur la rédaction du cahier des charges, puis sur la partie authentification et le système pour rester authentifié même hors ligne et finalement la mise en commun de l'ensemble des parties du projet.

- Thomas s'est dans un premier temps concentré sur la rédaction du cahier des charges, puis sur la partie authentification et le système pour rester authentifié même hors ligne et finalement la mise en commun de l'ensemble des parties du projet.

- Hugo s'est d'abord concentré sur la rédaction du cahier des charges, puis la conception du diagramme de classe puis sur la partie création, gestion et lecture des leçons, et finalement la rédaction du rapport technique du projet.

- Arthur s'est d'abord concentré sur la rédaction du cahier des charges, puis la conception du diagramme de classe, puis sur la partie création, gestion et lecture des leçons, et finalement la rédaction du rapport technique du projet.

Maintenant que les rôles ont été définis, l'avantage du projet est qu'il nous est possible de travailler chacun sur notre partie en parallèle.

Site internet et téléchargement de l'application

Conception initiale

Avant de commencer le développement, nous avons réalisé oralement une ébauche du site car il ne comprend pas beaucoup de fonctionnalités. Le développement a commencé par la page d'accueil, dont le but principal est de proposer le téléchargement de l'application.

Approche modulaire du développement

Pour garantir une construction efficace et bien organisée, nous avons adopté une approche modulaire en divisant la page d'accueil en différents composants. Ces composants incluent notamment :

- La barre de navigation, permettant de guider les utilisateurs à travers le site.
- La section héros, qui met en avant les principaux atouts de l'application.
- Le pied de page, offrant des informations complémentaires et des liens utiles.

Chaque composant a été développé et stylisé individuellement avec Tailwind CSS, ce qui a permis de gagner du temps grâce à ses classes utilitaires prédéfinies.

Mise à disposition et instructions d'installation

Après compilation du projet, nous avons généré un fichier exécutable que nous avons ajouté sur le site web. Afin de faciliter l'expérience utilisateur, nous avons également intégré une section détaillant les étapes d'installation de l'application. Ces instructions sont présentées de manière claire et accessible pour que les utilisateurs puissent suivre le processus sans difficulté.

Authentification et système hors connexion

L'authentification est assurée par la classe *UserController* en vérifiant les informations d'identification de l'utilisateur par rapport à la base de données.
La classe *ErrorHandler* valide les champs de connexion et de création de compte, en appliquant des restrictions telles qu'un mot de passe minimum de 6 caractères.
La classe *TokenManager* gère les jetons d'authentification, en les générant, en les enregistrant et en vérifiant leur validité.
Les jetons sont utilisés pour maintenir la connexion des utilisateurs.
Aussi, nous avons mis en place les logs avec la classe *LogController*, qui permet de renseigner dans la base de données les activités effectuées sur l'application.
De plus, nous avons ajouté une sécurité sur les tentatives de login forcées avec la classe *LoginAttempt* qui permet de limiter le nombre d'erreurs à la suite dans la tentative de connexion. Au bout de 5 erreurs, l'utilisateur se doit d'attendre 1 min.

L'authentification des utilisateurs repose sur la classe `UserController`, qui vérifie les informations d'identification fournies lors de la connexion en les comparant à celles enregistrées dans la base de données.

Afin d'améliorer la qualité des données saisies, nous utilisons la classe `ErrorHandler` pour valider les champs lors de la connexion ou de la création de compte. Cette classe impose des règles strictes, comme un mot de passe comportant au minimum six caractères dans la table ASCII, afin d'assurer un niveau de sécurité suffisant.

La gestion des sessions est assurée par la classe `TokenManager`, qui gère les jetons d'authentification. Cette classe s'occupe de générer des jetons uniques pour chaque utilisateur, de les enregistrer en toute sécurité et de vérifier leur validité à chaque requête. Les jetons permettent de maintenir la connexion des utilisateurs sans qu'ils aient à s'identifier à nouveau lors de chaque action.

Pour suivre les activités effectuées sur l'application, nous avons intégré la classe `LogController`. Cette classe enregistre dans la base de données des informations sur les actions réalisées par les utilisateurs, ce qui permet de faciliter le débogage et de renforcer la traçabilité des événements. Nous avons fait le choix de placer les logs dans la base de données afin de préparer ce système pour les futurs logs liés aux placements en bourse et la gestion des porte-monnaies qui nécessiteront une surveillance accrue et automatique.

En termes de sécurité, nous avons également développé la classe `LoginAttempt`, qui surveille et limite les tentatives de connexion échouées. Si un utilisateur échoue cinq fois consécutives à se connecter, il devra patienter une minute avant de réessayer. Ce système de délai vise à protéger l'application contre les attaques par force brute tout en offrant une expérience utilisateur fluide.

Création et lecture des leçons

Le principal défi est de développer un générateur de leçons permettant d'ajouter ou de supprimer des éléments dynamiquement dans la vue, tels que des titres, des paragraphes, des images ou des vidéos. Chaque élément doit être configurable pour permettre une personnalisation complète de la leçon. Par exemple, les titres peuvent être associés à différents niveaux hiérarchiques, les paragraphes peuvent inclure des styles spécifiques, et les images ou vidéos doivent permettre l'intégration des téléchargements locaux.

Nous avons également conçu une interface utilisateur intuitive, `CreationLessonView`, qui permet d'ajouter ou de modifier les différents composants d'une leçon. Cette interface offre une flexibilité suffisante pour que l'utilisateur puisse visualiser immédiatement les modifications apportées à la leçon. Une fois que l'utilisateur a terminé la création, nous devions gérer l'envoi de la structure complète de la leçon construite sur le front-end vers le back-end.

Comme nous utilisons le modèle MVC, nous avons créé un contrôleur `LessonController` qui récupère toutes les entrées saisies dans la vue et génère un objet Lesson. Ce contrôleur se charge ensuite de valider les données pour s'assurer de leur conformité et de leur cohérence avant de procéder à leur enregistrement dans la base de données.

Pour répondre à ces besoins, nous avons également dû intégrer des validations côté client et côté serveur afin de garantir que les leçons enregistrées soient conformes aux attentes en termes de contenu et de structure. Cela inclut la vérification de la cohérence des types de données, la gestion des erreurs lors de l'envoi des données, et l'assurance que les éléments de la leçon respectent un format standardisé.

Après avoir implémenté ces fonctionnalités de création, nous avons travaillé sur l'intégration des leçons dans la base de données et développé le reste des fonctionnalités de CRUD (Create, Read, Update, Delete). Cela nous a permis de compléter les pages associées, comme `LessonView` pour la visualisation des leçons, et une page de modification permettant de rééditer une leçon existante.

Les leçons sont affichées sur le menu principal `MainMenuView` via un composant `LessonComponent`. Ce composant permet de un accès direct à la page `LessonView` pour consulter le contenu complet d'une leçon.

De plus, pour faciliter la gestion et l'accès aux leçons, nous avons mis en place un système de navigation qui permet de trier et de rechercher les leçons. Une barre de recherche est disponible pour filtrer les leçons par titres, tandis qu'un menu déroulant (dropbox) permet de trier les leçons par tags. Ces fonctionnalités offrent une expérience utilisateur optimisée pour naviguer parmi un grand nombre de leçons.

Création et réponses aux quiz

Pour construire le générateur de quiz, nous avons débuté par la création des modèles principaux : `Quiz`, `Question`, et `Option`. Ces modèles servent de base pour structurer les données nécessaires à la construction d'un quiz interactif. Une fois les modèles définis, nous avons implémenté les opérations CRUD (Create, Read, Update, Delete) correspondantes dans le contrôleur, avant de développer la vue associée.

Le principal défi de ce générateur résidait dans la possibilité d'ajouter ou de supprimer dynamiquement des questions et des options directement depuis la vue. Pour cela, nous avons conçu une interface utilisateur intuitive qui permet de manipuler facilement les éléments du quiz. Une fois le quiz composé, nous avons mis en place une logique pour transmettre les données du front-end vers le back-end sous forme d'un objet `Quiz`, lequel est ensuite traité par un contrôleur qui se charge de son enregistrement dans la base de données.

Pour la fonctionnalité de mise à jour, nous avons dû comparer l'ensemble des questions existantes avec les nouvelles entrées. Nous avons utilisé la méthode `removeIf()` combinée à `contains()` pour identifier et supprimer les questions obsolètes tout en ajoutant les nouvelles.

Après avoir finalisé le générateur, nous avons intégré une fonctionnalité de validation des réponses. Bien que, pour la version actuelle de l'application, les scores ne soient pas enregistrés dans la base de données, cette fonctionnalité permet à l'utilisateur de s'entraîner. La validation est effectuée côté client : la vue calcule le score, affiche le résultat de chaque question, et indique les bonnes réponses en cas d'erreur.

Organisation et stockage des données

Pour gérer efficacement les informations liées à l'application, nous avons conçu une base de données nommée `da50`. Cette base contient plusieurs tables, chacune destinée à un aspect précis des données :

- **elements** : Gère les éléments présents dans les leçons..
- **lessons** : Stocke les leçons disponibles pour les utilisateurs.
- **logs** : Enregistre les activités des utilisateurs pour assurer un suivi des actions.
- **options** : Contient les options associées aux questions des quiz.
- **paragraphs** : Gère les paragraphes de contenu des leçons.
- **pictureintegrations** : Stocke les images utilisées dans les leçons.
- **questions** : Contient les questions des quiz, avec leurs attributs spécifiques.
- **quiz** : Gère les informations globales des quiz créés.
- **titles** : Stocke les titres utilisés dans les leçons.
- **users** : Contient les informations des utilisateurs, y compris leurs identifiants et leurs préférences.
- **videointegrations** : Stocke les vidéos intégrées dans les leçons.

Cette structure de base de données permet une organisation claire des informations et facilite leur manipulation lors des interactions entre le front-end et le back-end. Chaque table a été conçue pour optimiser l’efficacité des requêtes et garantir une scalabilité conforme aux besoins futurs de l’application.

En réalisant le projet de la façon décrite au-dessus, nous avons respecté la vision que nous avions du fonctionnement end-to-end de notre outil.

Ainsi, dans la dernière partie du rapport nous allons parler des résultats et des compétences acquises.

Résultats et retour d’expérience

Un point essentiel dans la réalisation de tout projet est de mener une phase d’analyse et de critique du travail accompli. Cette démarche permet d’identifier les forces et faiblesses, afin d’améliorer les futurs outils développés en adoptant des méthodes et approches plus performantes.

Nous avons structuré cette réflexion en deux sous-parties principales : critique de l’outil et compétences acquises.

Critique de l’outil

Plusieurs éléments d’améliorations ont été noté:

Nous avons principalement concentré nos efforts sur la mise en place des fonctionnalités de base telles que la création et la lecture de leçons ou de quiz sans accorder assez d’importance à l’intégration d’éléments innovants qui auraient pu enrichir l’expérience utilisateur. Par exemple, il aurait été pertinent d’ajouter :

- Différents types de quiz/test (Simulation d’investissement, etc.).
- Intégrer des éléments d’intelligence artificielle à la rédaction des leçons, comme des suggestions personnalisées.

Certaines fonctionnalités secondaires ont été omises ou simplifiées, comme :

- L’utilisation de Markdown pour formater les contenus de manière flexible.
- La séparation des logs entre la base de données et un fichier JSON pour limiter la place prise dans la base de données et tout de même améliorer l’analyse des activités.

Bien que notre application soit fonctionnelle, l’usage d’outils et de frameworks plus avancés aurait permis d’optimiser le développement et les performances. Par exemple, en suivant une période de montée en compétence dans l’optique de maîtriser Swing aurait pu améliorer l’interface utilisateur.

Compétences acquises

Ce projet nous a permis de développer des compétences aussi bien techniques que méthodologiques avec:

La découverte et maîtrise de technologies clés :

- Hibernate pour la gestion de la persistance des données.

- Maven pour la gestion des dépendances et l'automatisation des builds.
- Le système de Token pour sécuriser les sessions utilisateur.

Perfectionnement de l'utilisation de GitHub :

- Mise en place d'un suivi de projet efficace avec des outils comme les tableaux Kanban et le diagramme de Gantt.

Conception fonctionnelle :

- Meilleure compréhension de la distinction entre fonctionnalités principales et secondaires.
- Prise en compte de l'importance de l'innovation dans la conception d'un outil informatique.

Amélioration de la rédaction du cahier des charges :

- Avoir une rédaction claire et précise pour éviter toute ambiguïté.
- Définition rigoureuse des objectifs et des échéances.
- Justification approfondie des choix technologiques pour assurer leur pertinence.

Conclusion

Ce projet a représenté une opportunité précieuse de développer des compétences techniques et méthodologiques tout en relevant des défis concrets. Bien que certains aspects aient révélé des limites dans notre approche, notamment en termes d'innovation et d'exploitation des technologies, ces expériences constituent des enseignements précieux pour nos futurs projets d'ingénieurs.

Nous avons pu renforcer notre maîtrise d'outils essentiels comme Hibernate, Maven et les systèmes de gestion de projet collaboratif, tout en affinant notre manière de concevoir des applications informatiques. Les axes d'amélioration identifiés, qu'ils concernent l'intégration de fonctionnalités novatrices ou une meilleure planification des tâches, serviront de base pour des projets plus ambitieux et mieux structurés à l'avenir. Ce travail nous a également permis de mieux comprendre l'importance d'un cahier des charges clair et précis, élément clé pour garantir la réussite et la pérennité d'un projet informatique.

Annexes

Annexe 1: Liste des principaux scénarii

Note : les scénarios ont été réalisés au début du projet, certains éléments peuvent donc différer des choix prit au cours du développement.

1. Authentification

Scénario réussi : Connexion utilisateur

- **Préconditions :** L'utilisateur accède à l'application et lance la page de connexion.
- **Actions :**
    - L'utilisateur renseigne les champs « nom d'utilisateur » et « mot de passe ».
    - L'utilisateur valide les informations.
- **Résultat attendu :**
    - L'utilisateur est redirigé vers la page d'accueil correspondant à son rôle (« administrateur » ou « utilisateur classique »).

Scénario échoué : Connexion refusée

- **Préconditions :** L'utilisateur accède à l'application et tente de se connecter.
- **Actions :**
    - L'utilisateur renseigne des informations incorrectes.
    - L'utilisateur valide les informations.
- **Résultat attendu :**
    - La connexion est refusée.
    - Le champ mot de passe est réinitialisé.

Scénario : Changement de mot de passe

- **Préconditions :** L'utilisateur a accès à la fonctionnalité de changement de mot de passe.
- **Actions :**
    - L'utilisateur demande un changement de mot de passe.
    - L'utilisateur entre l'adresse mail qu'il pense avoir inscrit lors de la création de son compte
    - Un mail contenant un code de confirmation est envoyé (simulé)
    - L'utilisateur renseigne le code reçu.
    - Si le code est valide, l'utilisateur entre un nouveau mot de passe et le confirme.
- **Résultats attendus :**
    - **Code valide :** Le mot de passe est mis à jour avec validation des critères de sécurité.
    - **Code invalide :** Accès refusé et demande de recommencer l'opération.

---

2. Création de Compte

Scénario réussi : Création d'un nouveau compte

- **Préconditions :** Aucun compte n'est enregistré.
- **Actions :**
    - L'utilisateur accède à la page de connexion.
    - L'utilisateur sélectionne « Créer un compte ».
    - Un formulaire contenant les champs nom, prénom, adresse mail, mot de passe s'affiche.
    - L'utilisateur remplit le formulaire et confirme.
- **Résultat attendu :**
    - Un message confirme la création du compte.
    - L'utilisateur est redirigé vers la page d'authentification.

Scénario échoué : Création refusée

- **Préconditions :** L'utilisateur tente de créer un compte.
- **Actions :**
    - L'utilisateur remplit incorrectement le formulaire (par exemple : mot de passe non conforme, champs incomplets, ou erreur de connexion à la base de données).
- **Résultat attendu :**
    - Un message d'erreur décrit le problème (par exemple : « Mot de passe non valide », « Tous les champs doivent être remplis »).
    - Les données déjà renseignées restent présentes dans le formulaire.

---

3. Gestion des Cours

Création de Cours par un Administrateur

Scénario réussi : Création ou modification de cours

- **Préconditions :** L'administrateur accède au menu principal.
- **Actions :**
    - L'administrateur clique sur le bouton « Ajouter un cours ».
    - Une liste de leçons existantes (en cours de création ou publiées) s'affiche.
    - L'administrateur choisit une leçon existante ou crée une nouvelle leçon.
    - Il peut :
        - Donner un titre.
        - Ajouter des paraphrase.
        - Ajouter des images (vérification du format, taille et dimensions).
        - Intégrer des vidéos.
    - L'administrateur sauvegarde ou publie la leçon.
- **Résultat attendu :**
    - Enregistrement ou publication réussie.

Scénario échoué : Erreurs lors de la création

- **Préconditions :** L'administrateur tente d'ajouter un contenu.

- **Actions :**
    - Une erreur survient (par exemple : image non conforme, échec de la sauvegarde en base de données, échec de la publication).
- **Résultat attendu :**
    - Un message d'erreur précis est affiché.
    - Les modifications sont sauvegardées localement en cas d'échec de l'enregistrement en base de données.

Accès aux Cours par un Utilisateur

Scénario réussi : Consultation des cours

- **Préconditions :** L'utilisateur accède à la page de présentation des cours.
- **Actions :**
    - L'utilisateur utilise la barre de recherche ou des filtres pour trouver un cours.
    - L'utilisateur sélectionne un cours.
- **Résultat attendu :**
    - Le cours s'affiche correctement.

Scénario échoué : Erreurs d'accès

- **Préconditions :** L'utilisateur tente d'accéder à un cours.
- **Actions :**
    - La recherche renvoie un résultat vide.
    - Le cours ne se charge pas correctement.
- **Résultat attendu :**
    - Un message d'erreur précise la cause (« Aucun résultat correspondant », « Échec de chargement »).
    - L'utilisateur est redirigé vers le menu principal.

---

4. Téléchargement de l'Application

Scénario réussi : Téléchargement réussi

- **Préconditions :** L'utilisateur accède au site de l'application.
- **Actions :**
    - L'utilisateur clique sur le bouton de téléchargement.
    - Le site redirige vers la page du guide d'installation.
    - Le téléchargement commence automatiquement ou un lien manuel est proposé.
- **Résultat attendu :**
    - L'application est téléchargée avec succès.

Scénario échoué : Téléchargement échoué

- **Préconditions :** L'utilisateur tente de télécharger l'application.
- **Actions :**
    - Aucune action ne se produit après le clic sur le bouton.

    - Le téléchargement automatique ou manuel échoue.
- **Résultat attendu :**
    - Un message d'erreur invite l'utilisateur à réessayer ou à consulter un autre lien (par exemple : page GitHub).