Brice LIMOUSIN
Cyril BAUSSART
Erwan SALAVILLE
Thibault GARCIA-MEGEVAND

Projet Web Dev J2EE :

4X game

Professeur: Mohamed-Lamine BENZAGOUTA

SOMMAIRE:

INTRODUCTION:

Le projet de développement d'un jeu 4X (eXploration, eXpansion, eXploitation, eXtermination) a pour objectif de mettre en pratique les concepts appris dans le cadre de l'enseignement de la technologie Java Enterprise Edition (JEE). Ce projet vise à développer un jeu stratégique, multijoueur en tour par tour, accessible via une interface web dynamique, tout en respectant les principes de l'architecture MVC.

Les jeux 4X sont caractérisés par une gestion complexe des ressources, des stratégies élaborées et des interactions variées entre joueurs. Dans cette version simplifiée, les joueurs partagent une carte commune où ils peuvent explorer, occuper des territoires, récolter des ressources et engager des combats pour étendre leur domination. Chaque joueur agit à tour de rôle, avec des règles précises pour les déplacements, les actions et les conditions de victoire.

Ce projet repose sur l'intégration de plusieurs technologies et pratiques clé :

- Java Enterprise Edition (JEE) : utilisation de Servlets pour la gestion des requêtes et des JSP (JavaServer Pages) pour l'affichage dynamique des pages web.
- Base de données : stockage des informations relatives aux joueurs, à leurs scores et à l'état de la carte de jeu.
- Architecture MVC : séparation claire des responsabilités entre le modèle, les contrôleurs et les vues pour garantir une bonne organisation du code et faciliter sa maintenance.

Ce rapport présente le déroulement du projet, en mettant en évidence les choix techniques effectués, la structure du code et les fonctionnalités implémentées, ainsi que les résultats obtenus. L'objectif est de décrire les réalisations concrètes et de proposer une analyse critique pour orienter les futures évolutions possibles.

I/ORGANISATION

Afin de mener à bien ce projet, nous avons le choix de séparer celui-ci en différents lots en fonction de leurs priorités et de leurs impacts sur le suivant. Ainsi, nous avons établi 4 lots suivant :

- 1. Gestion de la connexion à la partie et création de la carte de celle-ci avec le système de fin de tour.
- 2. Ajout des soldats et de leur déplacement
- 3. Ajout des combats et autres actions du soldat ainsi que la gestion des scores
- 4. Gestion de fin de partie et phase de test

Afin de faciliter notre coordination, nous avons créé un serveur discord afin de communiquer sur nos avancements et priorités. Nous avons également créé un projet git afin de sauvegarder et de faciliter la gestion des différentes mises à jour.

II/CHOIX TECHNOLOGIQUES

Le projet “4X” à été développé en JEE et conteneurisé dans un serveur Tomcat. Afin de faire communiquer la partie client et la partie serveur, nous avons utilisé les classes servlet issues de JAKARTA. Enfin, nous avons fait le choix d'établir une base de données MySQL dans le but de sauvegarder diverses données relatives à notre application telle que les utilisateurs et les scores issus des différentes parties.

III/ ARCHITECTURE

Le projet 4X respecte l’architecture MVC. Cette structure garantit une séparation claire des responsabilités, ce qui facilite la compréhension, la maintenance et l’évolution du projet. Vous en trouverez les principales composantes ci-dessous les principales :

MODEL (src/main/java/model):

Ce dossier contient tous les objets évoluant dans le jeu :

- Map.java: représente le monde dans lequel se déroule le jeu.
- Autres : Modélise les joueurs, unités, les combats et autres éléments de la carte.

VUE (src/main/webapp):

Ce dossier contient les JSP permettant d'afficher les pages web dynamiques du jeu et d'interagir avec les utilisateurs.

- login.jsp : gère l'authentification des joueurs.-
- game.jsp : affiche la grille de jeu et les actions possibles.
- combat.jsp : montre les détails des combats en cours.
- scoreGame.jsp: affiche les scores des joueurs inclus dans la partie.
- scoreHistory.jps : affiche l'historique des scores d'un utilisateur.
- endGameScore.jsp: écran de fin de partie avec les scores

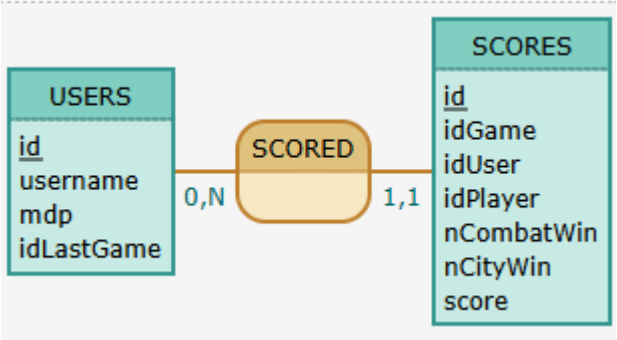
À noter que l'on y trouve un dossier "css" contenant les feuilles de style. Ainsi qu'un dossier où sont stockées les images utilisées par le jeu.

CONTROLER (src/main/java/controler):

Ce dossier contient les contrôleurs qui gèrent les actions des utilisateurs :

- FrontControllerServlet.java : point d'entrée principal redirigeant les actions vers les contrôleurs appropriés.
- LoginController.java : gère l'authentification et la gestion des comptes utilisateurs.
- ActionsController.java : traite les actions liées au jeu (déplacements, combats, récolte de ressources).
- ScoresController.java : gère les scores et leur affichage.

Nous avons également créé un dossier "utils" contenant notamment la class "Constantes.java" qui permet de modifier facilement les paramètres du jeu, tel que le nombre de joueurs ou la taille de la carte. Nous avons également un dossier "dao" contenant toutes les classes nécessaires à la bonne communication entre notre application et notre base de données MYSQL dont vous trouverez le MCD ci-dessous :



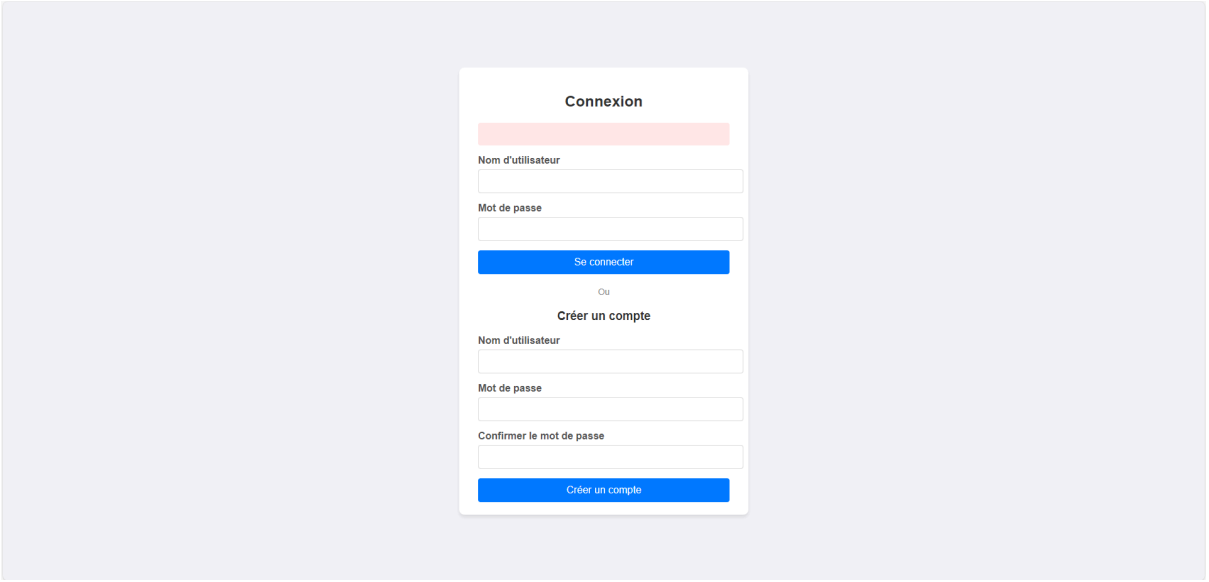
MCD de la base de donnée MySQL du projet

IV/ LE JEU

Cette section illustre le fonctionnement du jeu à l'aide de captures d'écran, mettant en avant les principales interfaces et fonctionnalités.

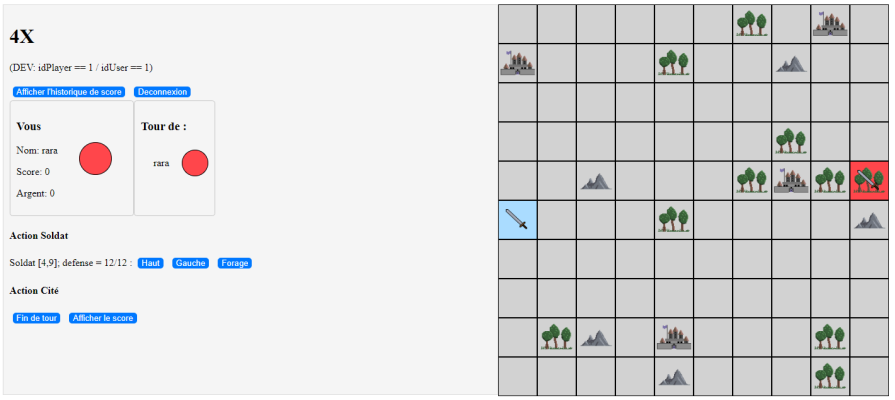
1. Écran de connexion (login.jsp):

Cet écran permet aux joueurs de s'inscrire ou de se connecter avec leurs identifiants pour accéder au jeu. Voici une capture d'écran de cette interface :



2. Jeu (game.jsp)

Cet écran est divisé en deux parties. À droite, nous affichons la grille de jeu affiche la carte partagée par tous les joueurs, avec ses différentes tuiles (ville, forêt, montagne, etc.) et les unités présentes. À gauche, nous avons une section présentant diverses informations sur la partie en cours et toutes les actions réalisables par l'utilisateur.



3. Interface de combat (combat.jsp)

Cette interface présente les détails d'un combat en cours, notamment les unités engagées, les points d'attaque, et l'évolution des points de défense.

4. Résumé des scores (endGameScore.jsp)

À la fin de chaque partie, un tableau récapitulatif présente les scores et statistiques des joueurs, permettant de comparer leurs performances. Cette page attend un certain temps avant

d’afficher le bouton “Relancer une partie” afin de laisser le temps à tous les joueurs d'apparaître sur cette page.

Score de fin de partie

Victoire de : rara

Joueur	Combats Gagnés	Cités Gagnées	Score Total
rara	0	1	4
michel	0	0	0

Refaire une partie

5. Résumé des score (scoreGame.jsp)

Cette interface résume les scores obtenus par les différents joueurs en cours de la partie en cours.

Liste des Scores

Joueur	Combats Gagnés	Cités Gagnées	Score Total
rara	0	0	0
michel	0	0	0

Retour au Plateau

6. Historique des scores (scoreGame.jsp)

Cette interface montre l’historique des parties précédentes, avec les scores obtenus par chaque joueur, permettant un suivi des performances dans le temps.

Liste des Scores:

Retour au Plateau

ID Partie	Combats Gagnés	Cités Gagnées	Score Total
1	0	1	4
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	1	4
15	0	1	4
16	0	0	0
17	0	0	0
18	0	1	4

7. Condition de victoire:

La partie se termine quand il ne reste qu'un joueur avec encore des soldats sur le plateau. Ou lorsque qu'un joueur atteint un certain score. Ce score est augmenté lors des conquêtes de cité et les victoires face à des soldats adverses. Toutes ces valeurs sont modifiables dans le fichier (src/main/java/utils/constantes.java). Par défaut, le score de victoire est 16 sachant qu'une conquête de cité vaut 4 points et qu'une victoire face à un soldat adverse 1 point.

V/DIFFICULTÉ RENCONTRE

La plus grande difficulté rencontrée lors du développement de ce projet a été la question de l'UI. En effet, il a été difficile de trouver un compromis entre : limitation technique et confort du joueur. Je pense notamment à la gestion de l'affichage des actions du joueur qui a souvent changé au cours du projet avant d'arriver à sa forme actuelle.

La question de la gestion des parties en cours à également été un questionnement tout au long du projet. Nous avions pensé à sauvegarder les parties dans la base de données de l'application, mais cela aurait donné des tables trop lourdes et incompréhensibles. Ainsi, la solution du singleton a été adoptée, mais permet à l'application de ne gérer qu'une seule partie à la fois.

Enfin, les différentes versions des bibliothèques utilisées par les membres de l'équipe ont entraîné des incompatibilités qui ont nécessité une synchronisation minutieuse des environnements de travail. Ce qui nous a fait perdre un temps précieux au début du projet.

VI/ PISTE D'AMÉLIORATION

Il reste des perspectives intéressantes que nous pourrions explorer afin d'améliorer ce projet. Nous pensons notamment à ajouter des animations fluides pour les déplacements des soldats, les combats ou les actions comme le forage permettrait d'améliorer l'expérience utilisateur ; Implémenter une gestion en temps réel plus performante, en utilisant par exemple WebSockets, pour minimiser les délais entre les actions des joueurs ; où encore proposer différentes cartes ou des missions spécifiques pour diversifier les parties. Rendre l'interface plus moderne et intuitive, en intégrant des éléments visuels interactifs.

CONCLUSION

En conclusion, le projet de développement d'un jeu 4X en Java Enterprise Edition a permis de mettre en pratique des concepts avancés de programmation tout en respectant les principes de l'architecture MVC. Malgré les défis rencontrés, notamment en termes d'interface utilisateur et de gestion des parties en cours, l'équipe a su trouver des solutions efficaces et a livré un produit fonctionnel et évolutif. Les pistes d'amélioration identifiées, telles que l'ajout d'animations, l'implémentation de gestion en temps réel et l'amélioration de l'interface, ouvrent la voie à des évolutions futures prometteuses pour enrichir l'expérience utilisateur et la fonctionnalité du jeu.