Application Web du jeu Skull

Septembre 2026

Consignes préparatoires : Ce projet a pour objectif de vous placer dans une situation proche d'un contexte professionnel. Chaque groupe devra travailler en collaboration, définir clairement les rôles de chacun et progresser de manière régulière tout au long du semestre. Le sujet étant volontairement ouvert, il est attendu des étudiants qu'ils fassent preuve d'initiative et de créativité afin de mobiliser et mettre en valeur l'ensemble de leurs connaissances et compétences dans la réalisation d'un projet complet. Attention, les fonctionnalités demandées devront être impérativement implémentées.

1 Contexte

Le projet porte sur l'implémentation du jeu Skull. Dans ce contexte, des joueurs pourront se connecter afin de jouer à une partie de Skull en ligne, en compétition avec d'autres joueurs ou des IAs. Attention, un administrateur peut se reserver le droit de bannir un joueur quand celui-ci est signalé. De plus, Skull est un jeu au design particulier, mettant en avant plusieurs cultures du monde via des crânes et des fleurs représentatives des quatre coins du monde. Un marketplace devra être accessible afin d'enrichir votre jeu et son design.
L'application finale sera développée à l'aide de web services, en respectant une architecture basée sur **SpringBoot**. L'authentification des utilisateurs devra être implémentée avec **SpringSecurity** et **JWT** (JSON Web Token). L'interface utilisateur devra être dynamique ; et pourra être implémentée avec **JSP** (JavaServer Pages), **JavaFX** ou n'importe quelle autre technologie vue pendant votre parcours.

Remarque importante : Les données pourront être stockées de n'importe quelle manière, mais devront rester persistantes entre deux utilisations de l'application.

2 Règles du jeu

Skull est un jeu de bluff et d'enchères, largement inspiré du Perudo. Les règles complètes officielles du jeu sont disponibles ici : `https://fr.scribd.com/document/741077569/Skull-Rules-Fr`. Dans une partie de Skull, chaque joueur possède **4 disques : 3 fleurs et 1 crâne**. Le but du jeu est d'être le premier à **réussir 2 défis**. Le jeu se déroule en manches, jusqu'à ce qu'un joueur réussisse ses défis, ou bien que tous les joueurs, sauf un, soient éliminés.

2.1 Déroulement d'une manche

Au début d'une manche, chaque joueur **pose secrètement un disque face cachée** devant lui. Personne d'autre ne sait donc s'il s'agit d'une fleur ou d'un crâne. Puis, le jeu se déroule par tour, joueur après joueur, dans le sens horaire. Durant son tour, un joueur a deux possibilités :

1) **Ajouter un disque** face cachée sur sa pile.

2) **Lancer une enchère**, en pariant sur le nombre de fleurs autour de la table. Un joueur ne pouvant plus poser de disque doit obligatoirement lancer une enchère.

Une fois une enchère lancée, les joueurs suivants ne peuvent que surenchérir ou se retirer de la manche. Si un joueur se retire, ses disques restent cependant face cachée sur la table. Les enchères sont proposées jusqu'à ce qu'il ne reste plus qu'un seul joueur, ou alors que l'enchère correspond au nombre total de disques sur la table. Le joueur qui a alors annoncé le nombre le plus élevé doit tenter son défi.

2.2 Le défi

Lorsqu'un joueur tente un défi, il doit **retourner autant de disques fleurs que sa dernière enchère**. Il retourne les disques un par un, en dépilant les piles de chaque joueur. Attention : un joueur réalisant un défi doit impérativement **commencer par dépiler intégralement sa propre pile** avant de dépiler celles des autres. Si le joueur réussit à **dépiler autant de fleurs que son enchère**, alors le défi est réussi et il marque 1 point. S'il tombe sur un **crâne**, le défi est raté. Il perd aléatoirement un de ses disques jusqu'à la fin de la partie.

3 Objectifs

L'objectif du projet est de créer une application web, reposant sur des services, qui permet :

- La gestion des utilisateurs : Un nouveau joueur peut s'inscrire ; et une fois un compte créé, le joueur pourra se reconnecter afin de conserver son historique de partie. La gestion des authentifications doit être faite pour empêcher un utilisateur d'outrepasser ses permissions. Chaque joueur a un espace privé.
- La gestion des parties dans un salon : Un ensemble de parties ouvertes doit être accessible aux joueurs connectés, afin d'en rejoindre une.
- Jouer une partie de Skull : Le but est avant tout de permettre à des joueurs de jouer au jeu. Le jeu se jouant avec un minimum de 3 joueurs, des IA pourront compléter une partie.
- La conservation d'un historique de parties : un système d'historique persistant pour classer les joueurs en fonction de leur nombre de victoires et/ou du ratio victoires sur nombre de parties jouées.
- Activer une nouvelle apparence depuis une marketplace.

4 Architecture générale

4.1 Front-End

Le choix de la technologie pour développer le front-end de l'application est libre. Il est cependant fortement recommandé d'utiliser une technologie vue en cours (**JavaFX, JSP**), ou avec laquelle vous êtes à l'aise et autonome. Le front-end fera office de client, et devra être instanciable plusieurs fois afin de permettre à plusieurs joueurs de se connecter.
Note : L'aspect principal de ce projet est le développement de web services ; le front-end ne doit donc pas représenter la majorité de votre production.

4.2 Back-End

L'utilisation de services interfacés via des **APIs REST** est obligatoire. Vous devrez définir vous-même les endpoints REST pour respecter les fonctionnalités demandées. La majorité des services doivent être développés et gérés à l'aide du framework **SpringBoot**, afin de gérer les requêtes HTTPS et les sessions utilisateur. Il est toutefois possible de développer un (et un seul) service en utilisant un langage et un framework autres que Java et SpringBoot (par ex. Flask en Python). Chaque service devra bien différencier ses couches en suivant le modèle MVC (Modèle - Vue - Contrôleur) lorsque c'est pertinent. Bien que l'architecture générale des services soit libre, je vous recommande d'avoir au moins 4 services : (i) `PlayerService`, pour la gestion des utilisateurs ; (ii) `LadderService`, pour la gestion des historiques de parties ; (iii) `GameService`, pour la gestion des parties de jeu ; et (iv) `MarketService`, pour la gestion de l'achat de nouvelles apparences.
L'ensemble des données manipulées par l'application (par ex. informations relatives aux utilisateurs, historique des parties) devra être stocké de manière **persistante**. Plusieurs approches sont possibles :

- L'utilisation d'une base de données relationnelle (par ex. MongoDB), permettant de garantir l'intégrité et la pérennité des données.
- Un stockage en mémoire (par ex. Map, List) avec sérialisation régulière dans des fichiers au format `JSON`, afin d'assurer une persistance entre deux exécutions de l'application.

Votre back-end devra être impérativement livré avec un environnement Docker composé d'un fichier docker-compose.yaml et des Dockerfile pour chacun de vos services. Le client, quant à lui, peut être conteneurisé à part, ou bien livré sous la forme d'un « fat JAR » contenant toutes les dépendances du projet.

4.3 Authentification

L'application doit implémenter l'authentification par **JWT** avec les dernières versions de **SpringBoot** et **SpringSecurity**. Si toutefois un service est implémenté avec autre chose que **SpringBoot** et **SpringSecurity**, l'utilisation de **JWT** reste obligatoire.

5 Fonctionnalités à implémenter

L'intégralité des fonctionnalités suivantes devra être implémentée. Les technologies pour le faire sont exposées plus haut dans la section Architecture générale.

5.1 Gestion des utilisateurs

- **Inscription** : Implémentez une page permettant à un utilisateur de se créer un compte. Le compte administrateur sera un compte particulier créé au premier démarrage de l'application. L'utilisateur devra renseigner son email et un mot de passe.
- **Connexion** : Implémentez une page permettant à un utilisateur de se connecter. Un formulaire demandera les informations d'authentification (email, mot de passe).
- **Tableau de bord** : Implémenter une page pour que l'utilisateur connecté ait accès à une liste des parties joignables, et à son historique de jeu. Un bouton de déconnexion doit aussi être accessible.
- **Bannir un joueur** : Un joueur peut en signer un autre au cours d'une partie, ou à posteriori, en cas de suspicion de triche. Dans ce cas, l'administrateur doit pouvoir le bannir depuis son interface personnalisée.

5.2 Gestion des parties

- **Créer une partie** : Un utilisateur peut créer une nouvelle partie, en spécifiant le nombre de joueurs maximum, et le nombre d'IA autour de la table. L'utilisateur rejoint automatiquement cette partie.
- **Rejoindre une partie** : Un utilisateur peut rejoindre une partie qui n'est pas encore lancée. Les motifs utilisés pour les fleurs et crâne du joueur peuvent être choisis ou définis aléatoirement. Deux joueurs peuvent avoir les mêmes motifs.
- **Quitter une partie** : Un joueur peut quitter une partie. Si la partie était lancée, alors une IA prend sa place. Si elle ne l'était pas, alors une
- **Jouer une partie** :

- Un joueur peut joueur durant une partie en cours, lorsque c'est son tour, en respectant les règles du jeu. Une fois la partie finie, le résultat est enregistré, et le joueur peut retourner sur la page d'acceuil. Attention, si une partie n'est pas terminée, et qu'un joueur quitte la table, alors il est automatiquement considéré comme perdant, peu importe le résultat que fera son IA ensuite.
- Une IA de jeux pour les personnages non-joueurs devra être implémentée avec une logique simple. En voilà une qui repose sur quelques règles de décision :
  * Lorsque c'est à elle de jouer, l'IA a 80% de chance d'ajouter une nouvelle tuile (aléatoire), et 20% d'entamer une enchère en annoncant son nombre de tuile comme défi.
  * Lorsqu'elle a la possibilité de surencherir, et qu'elle n'a posé que des fleurs, l'IA surenchéri à 70% de chance, avec une enchère à +1 de la précédente, et se retire à à 30%. Si l'IA a posé un crane, elle surenchérie à 20% de chance, avec une enchère à +1 de la précédente, et se retire à à 80%.

- **Signaler un joueur** : En cours de partie, un joueur peut en signaler un autre en cas de suspicion de triche. La partie continue normalement cependant.

5.3 Gestion de l'historique

- **Accéder à son historique** : Un joueur peut accèder au détail de ses parties précédentes : joueurs, score de chacun, nombre de tuiles restantes, etc. Il peut, depuis cette interface, signaler un autre joueur en cas de suspicion de triche.
- **Accéder au classement général des joueurs** : Une vue des meilleures joueurs, en fonction d'un ou des critères que vous définirez (par ex. nombre de victoires, ratio victoires / défaites, etc), doit être accessible aux joueurs.

5.4 Gestion des illustrations

- **Activer un nouveau set d'illustrations** : Un utilisateur peut décider d'*acheter* une nouvelle apparence pour les tuiles (cranes et fleurs). L'achat doit se dérouler sous la forme d'un paiement par résolution mathématique d'une operartion simple. Lorsqu'un joueur achète un modèle de tuile, un calcul aléatoire est généré, et l'utilisateur doit le resoudre correctement pour valider la transaction. Cette apparence est ensuite disponible pour tous les autres joueurs.

6 Déroulement

Dans ce projet, vous allez devoir faire preuve de professionnalisme. Vous allez devoir scinder vos projets en lots distincts. Pour chaque lot, vous devrez établir : un descriptif fonctionnel du lot, un plan de tests, une date de livraison. Au dernier lot, l'application devra donc être complète.

Petits conseils :

- Vous comprendrez qu'un certain nombre de technologies nécessaires à l'élaboration de ce projet sont en cours d'apprentissage. Par exemple, Spring Security arrive assez tard dans le programme du module de Web Services. Il faudra donc établir vos lots en fonction de ces paramètres.
- Le projet se décompose facilement en plusieurs tâches pouvant être réalisées et testées indépendamment. Leur répartition en fonction de leurs dépendances est donc une tâche initiale importante. Il est aussi conseillé de prévoir vos interfaces (au sens Java) en amont, pour permettre une intégration des services sans leurs implémentations complètes.

De même pour le test structurel spécifique aux services REST. On ne demande pas d'automatisation obligatoire de vos tests fonctionnels, mais des preuves comme quoi vous les avez faits. Tout ceci devra être documenté à chaque livraison. Vous devrez à minima avoir trois lots distincts. Vous devrez annoncer votre planning prévisionnel avant le 30 septembre, délai de rigueur.

7 Livrables

- Pour chaque livraison, **un document décrivant le lot** (pdf) et la **qualification du lot** (fonctionnelle / structurelle) devront être fournis. Le **code source et un rapport des tests** devront aussi être fournis. Toute documentation supplémentaire sera appréciée.

- **Rendu final**
  - **Un rapport** contenant l'histoire de votre projet et vos choix de conception. Un template LaTeXest disponible ici.
  - **Une spécification de vos points d'entrée** REST (OpenAPI ou Swagger) ;
  - **Le code source** final de l'application, incluant les éléments nécessaires à sa conteneurisation ;
  - **L'intégralité des prompts** fournis à des IAs génératives.

8 Note sur l'utilisation d'IA générative

L'utilisation d'IA génératives et de modèles de langage (LLM) pour assister le développement n'est pas prohibée. **Cependant, chaque membre de l'équipe doit être capable d'expliquer, de justifier et de reproduire l'intégralité du code rendu, indépendamment de l'outil ayant contribué à sa production**. L'utilisation de ces outils doit également rester critique : les modèles peuvent reproduire ou amplifier certains biais présents dans leurs données d'entraînement [1, 4], et une délégation excessive du raisonnement à l'IA peut affecter la mobilisation de l'esprit critique [3]. Des travaux montrent également les effets de l'utilisation des LLM sur l'activité cérébrale, le développement du cerveau [5] et l'engagement cognitif [2]. Pour approfondir ces questions, référez-vous aux articles scientifiques cités. Pour une présentation plus vulgarisée, je vous recommande la vidéo *La Fabrique à Idiots*, de Micode [1].

9 Évaluation du projet

L'évaluation du projet reposera à la fois sur les livrables intermédiaires (15% de la note), sur le rendu final (60% de la note) et sur une présentation finale devant l'enseignant (25% de la note).
Les critères d'évaluation principaux sont les suivants :

- **Application des notions de cours** : programmation de services comme vu en cours, sécurité via authentification, architecture de l'application, etc.
- **Respect de la spécification** : conformité à l'architecture définie, gestion correcte des utilisateurs, respect des règles du jeu, etc.
- **Gestion du projet** : organisation du travail en équipe, répartition des rôles, respect du calendrier et progression régulière.
- **Qualité du code** : structure, lisibilité, respect des bonnes pratiques Java/Spring, ainsi que la présence et la qualité des tests (obligatoires).
- **Interface utilisateur** : simplicité d'utilisation, ergonomie et cohérence générale.

À l'issue de la présentation finale, chaque équipe passera également un entretien individuel avec l'enseignant.
Une note globale de projet sera attribuée à l'ensemble du groupe. Toutefois, dans le cas d'une participation inégale entre les membres de l'équipe, un **coefficient individuel** (compris entre 0 et 1.4) pourra être appliqué. La note finale individuelle sera alors calculée comme suit :

$$\text{Note individuelle} = \text{Note du projet} \times \text{Coefficient individuel}$$

10 Groupes

Merci de remplir au plus vite (max 30 septembre) :
`https://docs.google.com/spreadsheets/d/12O5Q7oW6nBWIHkO6QNVN8vKCcOVQScozSibvQzOMwN4/edit?usp=sharing`

Références

[1] Isabel O. Gallegos, Ryan A. Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K. Ahmed. Bias and fairness in large language models : A survey, 2024.

1. `https://www.youtube.com/watch?v=4xq6bVbS-Pw`

[2] Nataliya Kosmyna, Eugene Hauptmann, Ye Tong Yuan, Jessica Situ, Xian-Hao Liao, Ashly Vivian Beresnitzky, Iris Braunstein, and Pattie Maes. Your brain on chatgpt : Accumulation of cognitive debt when using an ai assistant for essay writing task, 2025.

[3] Hao-Ping (Hank) Lee, Advait Sarkar, Lev Tankelevitch, Ian Drosos, Sean Rintel, Richard Banks, and Nicholas Wilson. The impact of generative ai on critical thinking : Self-reported reductions in cognitive effort and confidence effects from a survey of knowledge workers. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, New York, NY, USA, 2025. Association for Computing Machinery.

[4] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Comput. Surv.*, 54(6), July 2021.

[5] Samson Nivins, Bruno Sauce, Magnus Liebherr, Nicholas Judd, and Torkel Klingberg. Long-term impact of digital media on brain development in children. *Scientific Reports*, 14 :13030, 2024.