Note de Cadrage 

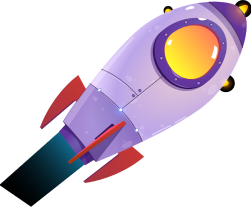
## 

## **Contexte et objectifs**

Le projet **Space Operators** s'inscrit dans le cadre du projet fil rouge de la formation **M2i M1 DEV 2024-2025**.

L’objectif est de concevoir et développer une application en **JAVA**, sous forme d’application **desktop.**

## **Description du projet**

* Nom du projet : Space Operators
* Type d'application : Jeu vidéo coopératif, développé sous forme d’une **application desktop**.
* Public cible : Tous les **joueurs PC** souhaitant vivre une expérience de jeu collaborative en mode **coopératif**.

## **Fonctionnalités minimales requises**

### **1. Authentification utilisateur par API**

* Mise en place d’un système sécurisé d’authentification.
* Gestion des comptes utilisateurs avec création, connexion et déconnexion.
* Possibilité de réinitialiser les mots de passe ou de modifier les informations du profil.

### **2. Manipulation de jeux de données (BDD, web services)**

* Intégration d’une **base de données locale ou distante** pour la gestion des données du jeu (profils, scores, missions, etc.).
* Option d’interfaçage avec des **webservices** pour récupérer ou synchroniser des données en temps réel (ex. mises à jour de contenu).

### **3. Actions CRUD complètes (Create, Read, Update, Delete)**

* Gestion des données liées aux utilisateurs et aux éléments du jeu :
  + Création de comptes de type utilisateur.
  + Création de sessions de partie.
  + Affichage de l’historique des parties
  + Suppression de profil.

### **Affichage dynamique du niveau**

* Conception d’une **interface dynamique** permettant de visualiser en temps réel les statistiques du vaisseau (état des systèmes, ressources, objectifs de mission).
* Mise à jour automatique des données affichées selon les actions des joueurs.

### **Multi-profils utilisateurs**

* **Gestion des profils utilisateurs** distincts (différents rôles ou niveaux d’accès selon les joueurs).

### 

### 

### 

### **Partie web : Interface d'administration (questions + utilisateurs)**

Une interface web sera mise en place spécifiquement pour les administrateurs du jeu afin de :

* Créer de nouvelles questions et missions (textes, choix, niveau de difficulté, type…).
* Lire la liste complète des questions existantes.
* Mettre à jour les questions (contenu, catégorie, paramètres de difficulté…).
* Supprimer les questions obsolètes ou inadaptées.
* Gérér les utilisateurs

## **4. Fonctionnalités facultatives**

### **Utilisation de l’intelligence artificielle pour la génération des questions/tâches**

* Implémentation d’un agent de l’intelligence artificielle Open AI capable de générer de nouvelles questions.
* Adaptation des tâches en temps réel en fonction des performances des joueurs, créant ainsi une difficulté évolutive et personnalisée.
* Possibilité de proposer des missions aléatoires ou scénarisées selon les décisions des joueurs, enrichissant ainsi la rejouabilité du jeu.

### **Création d’un chat vocal in-game**

* Intégration d’un **système de communication vocale en jeu** pour permettre aux joueurs de collaborer efficacement en temps réel.
* Gestion du chat vocal avec des fonctionnalités telles que :
  + Activation/désactivation du micro.
  + Détection automatique de la voix (push-to-talk ou voix continue).
  + Ajustement du volume individuel des coéquipiers.

## **5. Contraintes technologiques**

### **1. Spring (Java SE)**

* **Injection de dépendances** : facilite la gestion des objets et leurs dépendances, garantissant un code plus propre et maintenable.
* **Architecture MVC (Modèle-Vue-Contrôleur)** : sépare la logique métier de l’interface utilisateur, améliorant la modularité.
* **Gestion des services backend** : Spring gère les processus liés à l’authentification, à la gestion des données et aux services métiers.
* **Sécurité intégrée** : possibilité d'utiliser **Spring Security** pour la gestion des connexions sécurisées.
* Possibilité d’utiliser de créer un websocket permettant de rendre la communication en temps réel.

### **2. Maven**

* **Gestion des bibliothèques** : automatise le téléchargement des bibliothèques nécessaires, garantissant une configuration cohérente pour tous les développeurs.
* **Cycle de vie automatisé** : prend en charge les étapes du build, de la compilation au déploiement, en passant par les tests.
* **Structure normalisée** : impose une organisation standard du projet, facilitant la collaboration en équipe.

### **3. JUnit**

* **Tests automatisés** : permet de vérifier la fiabilité du code en exécutant des tests sur les fonctions critiques.

### **4.** **OpenJFX (JavaFX)**

* **Conception d’interface dynamique** : permet de créer des interfaces avec des composants graphiques avancés tels que tableaux, graphiques, et animations.
* **Gestion des événements en temps réel** : idéal pour une mise à jour dynamique des tableaux de bord et des éléments interactifs dans un jeu.

### **5.** **Web (Spring Web)**

* Interface d’administration web.

## **6. Calendrier prévisionnel**

* Novembre-Décembre 2024 :
  + Conception
  + Planification
  + Spécifications
  + Prototype application desktop (V1)
* Avril-Juillet 2025 :
  + Codage
  + Tests
  + Déploiement
  + Présentation / démo de l'application web (V2)

## **7. Livrables attendus**

* Note de cadrage (PDF)
* Spécifications fonctionnelles détaillées
* Spécifications techniques
* Dossier de conception avec diagrammes
* Code source sur GitHub
* Diagramme de Gantt / planning de projet

## **8. Equipe de suivi**

* Romain SESSA : Responsable du développement JAVA
* François SCHOSSIG : Responsable du suivi de projet et des livrables