Cahier des Charges Technique

Projet : Plateforme éducative interactive pour enfants

---

1. Description Générale du Projet

La plateforme vise à offrir une expérience éducative ludique pour les enfants, avec des jeux et des activités dans des domaines comme les mathématiques, le français et l'introduction au codage. Les parents pourront suivre la progression de leurs enfants, et les jeux seront adaptatifs pour offrir des niveaux de difficulté croissants.

2. Spécifications Techniques

2.1 Environnement de Développement

- **Front-End** : Vue.js (Composition API et `<script setup>`)
- **Back-End** : Spring Boot avec Java, intégrant Spring Security pour la gestion de la sécurité
- **Base de Données** : MySQL pour la gestion relationnelle des données utilisateurs, scores et statistiques
- **API Documentation** : Swagger pour décrire et tester les endpoints de l'API
- **Authentification** : JWT pour les sessions sécurisées
- **Tests** : Utilisation de Postman pour l'API et Jest ou Mocha pour le front-end
- **CI/CD** : GitHub Actions pour les tests et déploiements automatiques (optionnel)

---

3. Structure de l'Application

3.1 Front-End

1. **Structure des Composants Vue.js** : Suivant une architecture en composants modulaires avec un fichier principal `App.vue` pour les routes principales et des composants nommés selon le schéma `{Nom}Comp.vue` pour une meilleure gestion.
2. **Gestion de l'État** : Utilisation de Pinia pour gérer l'état global et synchroniser la progression, les scores et les badges des utilisateurs entre les composants.
3. **Routing** : Utilisation de `vue-router` pour naviguer entre les sections comme l'accueil, les jeux, le dashboard, et l'espace parent.

3.2 Back-End

1. **Contrôleurs REST** : Organisation en trois contrôleurs principaux :
    - **UserController** : Inscription, connexion, gestion de profil.
    - **GameController** : Récupération des jeux et progression de l'utilisateur.
    - **ProgressController** : Gestion et enregistrement des scores et des progrès par jeu.

2. **Services** : Chaque contrôleur repose sur des services pour l'exécution de la logique métier.
3. **Gestion des erreurs** : Middleware de gestion d'erreurs pour retourner des messages clairs à chaque requête.
4. **Sécurité** : Spring Security avec JWT pour garantir l'accès sécurisé aux endpoints selon le type d'utilisateur (enfant, parent, admin).

---

4. Fonctionnalités Spécifiques

4.1 Front-End

1. **Page d'Accueil** : Introduction des jeux, options de matière et explications pour les parents.
2. **Jeux Éducatifs** : Interface avec animation et chronomètre intégrés pour chaque jeu, ajustée pour répondre aux spécificités de chaque matière.
    - **Mathématiques** : Jeux basés sur des calculs et des puzzles logiques.
    - **Français** : Jeux d'orthographe, de grammaire et de conjugaison.
    - **Codage** : Introduction aux concepts basiques de la logique de programmation.
3. **Dashboard Utilisateur** : Suivi en temps réel des badges, de la progression, des scores.
4. **Espace Parents** : Vue des performances et des progrès de l'enfant par matière.
5. **Interface Responsive** : Optimisation pour les appareils mobiles et tablettes.

4.2 Back-End

1. **API Utilisateurs** :
    - Inscription, Connexion, et gestion du profil utilisateur.
    - Gestion des rôles avec accès aux contrôles parentaux pour les comptes parent.
2. **API Progression** :
    - Suivi et enregistrement des scores par jeu et par session.
    - Historique de progression permettant de visualiser l'évolution de chaque utilisateur.
3. **API Classement et Récompenses** :
    - Calcul automatique des classements par score et niveau.
    - Enregistrement des badges gagnés en fonction des performances.

---

5. Base de Données

Modélisation des Tables

1. **Table Utilisateurs**
    - **Champs** : `id`, `nom`, `email`, `mot_de_passe`, `role`, `date_creation`, `avatar`
    - **Description** : Stocke les informations de chaque utilisateur et leur rôle (enfant, parent, admin).
2. **Table Jeux**
    - **Champs** : `id`, `nom`, `description`, `matière`, `niveau_difficulté`

- **Description** : Liste des jeux disponibles avec un champ pour le niveau de difficulté.

3. **Table Progression**
    - **Champs** : `id_progression`, `id_utilisateur`, `id_jeu`, `score`, `date`, `niveau_atteint`
    - **Description** : Enregistre les scores et le niveau atteint par l'utilisateur pour chaque jeu.
4. **Table Badges**
    - **Champs** : `id_badge`, `nom`, `description`, `points`, `niveau_requis`
    - **Description** : Contient les différents badges et conditions pour les obtenir.
5. **Table Classement**
    - **Champs** : `id`, `matière`, `id_utilisateur`, `rang`, `date_mise_a_jour`
    - **Description** : Affiche les classements des utilisateurs par matière.

---

6. Sécurité

1. **Authentification JWT** : Gestion des sessions pour les utilisateurs authentifiés via JWT, avec des rôles spécifiques pour l'accès aux informations et fonctionnalités.
2. **Contrôles d'accès** : Les contrôles d'accès sont appliqués selon le type de compte (enfant, parent, admin).
3. **Protection CSRF et Validation des Données** : Filtrage des entrées pour prévenir les failles de sécurité.

---

7. API REST et Documentation

1. **Endpoints API** :
    - `/api/users/register` : Inscription utilisateur
    - `/api/users/login` : Connexion utilisateur
    - `/api/games/list` : Récupère la liste des jeux disponibles
    - `/api/progress/update` : Mise à jour de la progression de l'utilisateur
    - `/api/rankings/list` : Affiche les classements
    - `/api/parent/reports` : Informations spécifiques pour le suivi parental
2. **Documentation** : Swagger pour documenter tous les endpoints disponibles.

---

8. Exigences de Performance

1. **Performance de l'API** : Temps de réponse de moins de 200 ms.
2. **Évolutivité** : La structure doit permettre l'ajout de nouveaux jeux et de matières.
3. **Sécurité et Tests** : Mise en place de tests de sécurité, de charge, et d'intégration pour garantir la fiabilité du système.

---

9. Plan de Développement et Maintenance

1. **Étapes du Projet**
    - **Phase de Maquettage** : Création des maquettes avec Figma.
    - **Développement du Front-End** : Mise en place des composants Vue.js et intégration du design responsive.
    - **Développement du Back-End** : Création de l'API REST, intégration des services, et mise en place de Spring Security.
    - **Intégration et Tests** : Tests unitaires, d'intégration et tests utilisateurs.
2. **Maintenance et Améliorations Futures**
    - **Maintenance Mensuelle** : Suivi des performances et mises à jour de sécurité.
    - **Améliorations Futures** : Ajouter de nouveaux jeux, améliorer le système de progression, intégrer de nouvelles matières.