1 Présentation Globale

L'objectif de l'application est de permettre à des joueurs de faire une partie de planning poker, en respectant les règles vues en cours.

L'application peut être à distance ou locale.

Un menu permet de décider du nombre de joueurs et de rentrer un pseudo pour chacun des joueurs. Le menu doit aussi permettre de choisir parmi différentes règles de planning poker (règles strictes, moyenne, médiane, etc.)

On doit pouvoir entrer une liste de fonctionnalités ou backlog.

Une fois que chacun à voté, l'application valide ou non la fonctionnalité en fonction des règles choisies via le menu. Si la fonctionnalité n'est pas validée, on recommence le vote.

Lorsque tout le backlog est validé, l'application enregistre un fichier JSON avec, pour chaque fonctionnalité, la difficulté estimée par l'équipe.

2 Choix des languages

Nous avons opté pour le framework **Express** et le langage **JavaScript** pour développer l’API backend, pour les raisons suivantes :

- **Documentation simplifiée** : grâce à l’intégration facile de SwaggerDoc.
- **Tests optimisés** : avec des outils performants tels que Jest et Supertest.

Pour le front-end, notre choix s’est porté sur **Godot**, un moteur spécialement conçu pour la création de jeux vidéo. Sa spécialisation le rend bien plus adapté à notre projet qu’un framework web classique.

3 Structure du projet

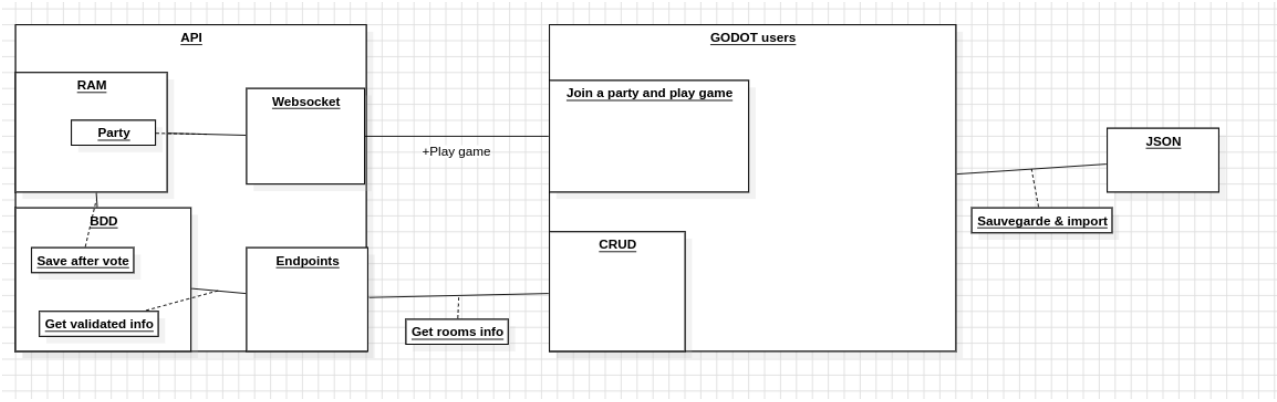

Figure 1: Structure du projet

Comme présenté ci-dessus, le projet est structuré en deux parties principales :

1. **Le backend**
    - **Endpoints REST** : Permettent de modifier directement les données de la base de données.

- **WebSocket** : Gère les interactions en temps réel, notamment pour les fonctionnalités de vote.

2. **Le front-end**
   Développé avec l'application **Godot**, il permet de lancer des sessions de planning poker et de consulter les informations des différentes rooms.

Vous trouverez ci-contre le schéma de la base de données utilisée par l'API.

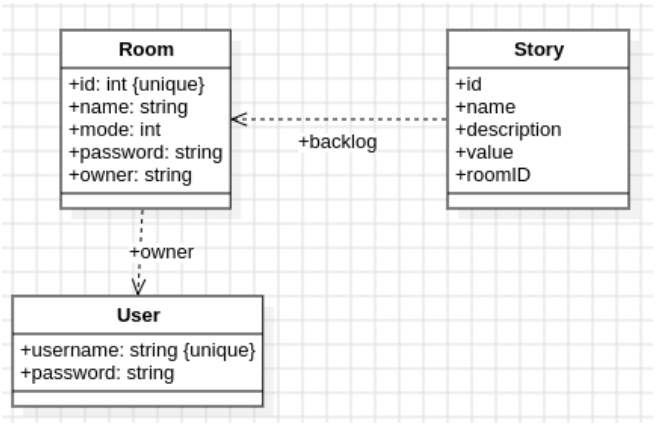

Figure 2: Structure de la BDD

4 Intégration continue et suivi de projet

Pour les tests, nous avons utilisé les modules **Jest** et **Supertest**, qui permettent de tester efficacement les endpoints d'une API Express. Conformément aux indications fournies dans le document qui nous a été remis, nous avons rédigé les tests en respectant les bonnes pratiques.

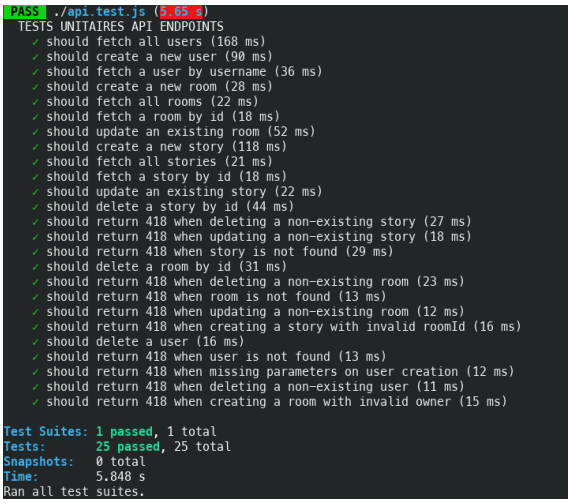

Figure 3: Résultat des tests

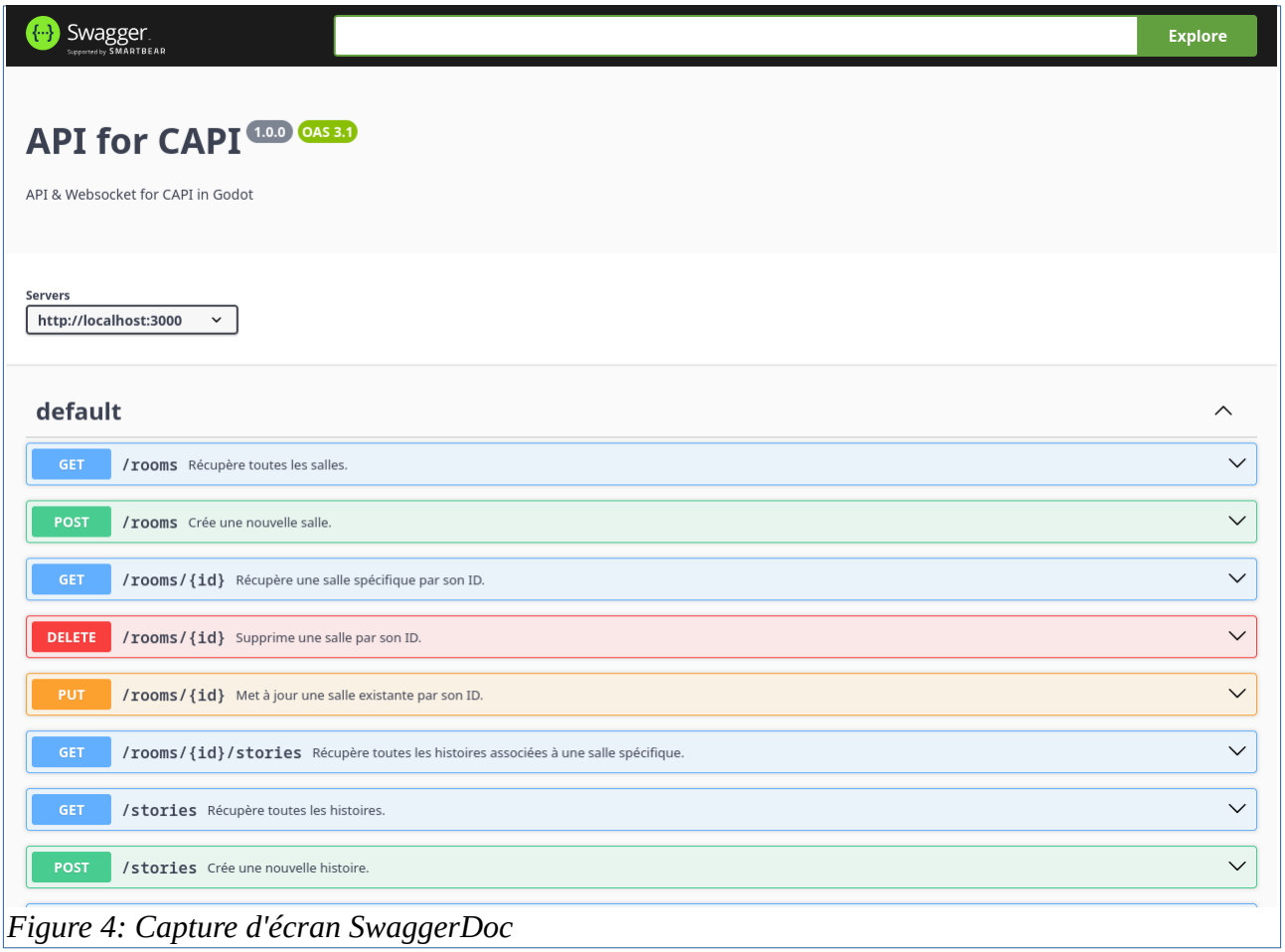

Figure 4: Capture d'écran SwaggerDoc

Pour mettre en place la documentation, nous avons utilisé le module **Swagger**, qui permet de générer automatiquement une page HTML interactive. Cette page offre la possibilité de tester les différents endpoints de l'API, de visualiser leurs fonctionnalités, ainsi que les valeurs attendues en entrée.

Vous pouvez accéder à cette page via l'endpoint : **`/api-docs`**
URL : http://{IP de l'API ou 127.0.0.1 pour localhost}:3000/api-docs/

Nous avons tenté de mettre en place des tests d'intégration continue avec **GitHub Actions**, mais nous n'avons pas réussi à les implémenter. En effet, l'API nécessite l'accès à la base de données ainsi qu'au fichier **.env**, qui ne sont pas présents dans le repository GitHub.