Rapport détaillé pour le projet Capi

LORTEAU Chelo, LECOMTE Thibaud

M1 Informatique, Université Lumières LYON 2

1. Présentation générale du projet

Capi est une application web collaborative et interactive qui utilise Flask et Flask-SocketIO pour son développement.

Proposée comme une plateforme de gestion participative ou de jeu, elle se démarque par ses caractéristiques en temps réel, offrant aux utilisateurs la possibilité de se connecter à des salles, d'envoyer des messages et de participer à des activités interactives. Son design est conçu pour garantir une compatibilité avec différentes plateformes, offrant une expérience fluide sur les ordinateurs, les tablettes et les smartphones.

Le but principal est de proposer une plateforme légère, facile à utiliser et performante, qui répond aux exigences des utilisateurs dans un contexte de travail en équipe.

---

2. Motivations des choix technologiques

- **Framework Flask** :
    - Choisi pour sa légèreté et sa simplicité, il permet de développer rapidement des applications web évolutives. Il est particulièrement adapté pour notre projet nécessitant des interactions côté serveur et côté client.
- **Flask-SocketIO** :
    - Ce module permet d'intégrer facilement des fonctionnalités WebSocket, essentielles pour des communications en temps réel. Il est idéal pour notre application collaborative où les utilisateurs doivent interagir simultanément.
- **Multiplateforme** :
    - L'interface utilisateur est optimisée pour s'adapter aux écrans de différentes tailles. Le choix de technologies web standards (HTML, CSS, JavaScript) garantit une compatibilité avec les navigateurs sur PC, tablettes et téléphones.
- **Sessions sécurisées** :
    - Utilisation d'une clé secrète pour garantir que les sessions utilisateur sont sécurisées contre les manipulations malveillantes.
- **Structure modulaire** :
    - Les fonctionnalités sont bien séparées entre les gestionnaires d'événements WebSocket, les routes HTTP, et les fonctions utilitaires. Cela facilite la maintenance et l'évolution future de l'application.

---

3. Fonctionnalités détaillées

a. Gestion en temps réel

Les fonctionnalités WebSocket offrent des interactions dynamiques :

- Envoi et réception de messages dans des salles spécifiques.
- Notifications en temps réel des utilisateurs rejoignant ou quittant les salles.
- Maintien de logs de chat spécifiques à chaque salle.

b. Gestion des parties

- **Création de parties** :
    - Les utilisateurs peuvent créer une partie en choisissant un mode de jeu et un pseudonyme. Une clé unique est générée pour chaque partie.
- **Participation à des parties** :
    - Les utilisateurs rejoignent des parties existantes via une clé.
    - La liste des participants est partagée et mise à jour en temps réel.

c. Multiplateforme

- L'application est développée pour offrir une compatibilité avec différents appareils grâce à un design adaptatif. Les utilisateurs peuvent jouer ou collaborer via un ordinateur, une tablette, ou un smartphone.

---

4. Diagrammes

a. Diagramme de classe

Ce diagramme illustre les relations entre les principales classes et entités de l'application, y compris les joueurs, les salles, les messages, et la gestion des sessions.

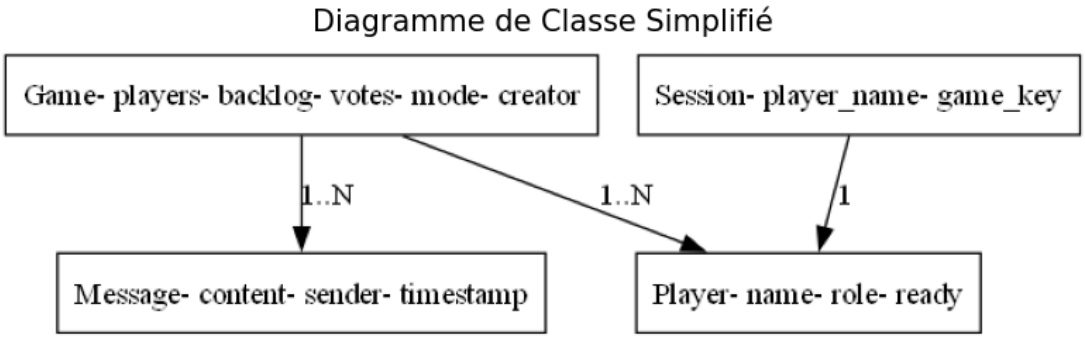

Figure 1 : Diagramme de Classe Simplifié

b. Diagramme de séquence

Ce diagramme montre le déroulement des interactions utilisateur, comme la création d'une partie, l'envoi d'un message, et la mise à jour en temps réel.

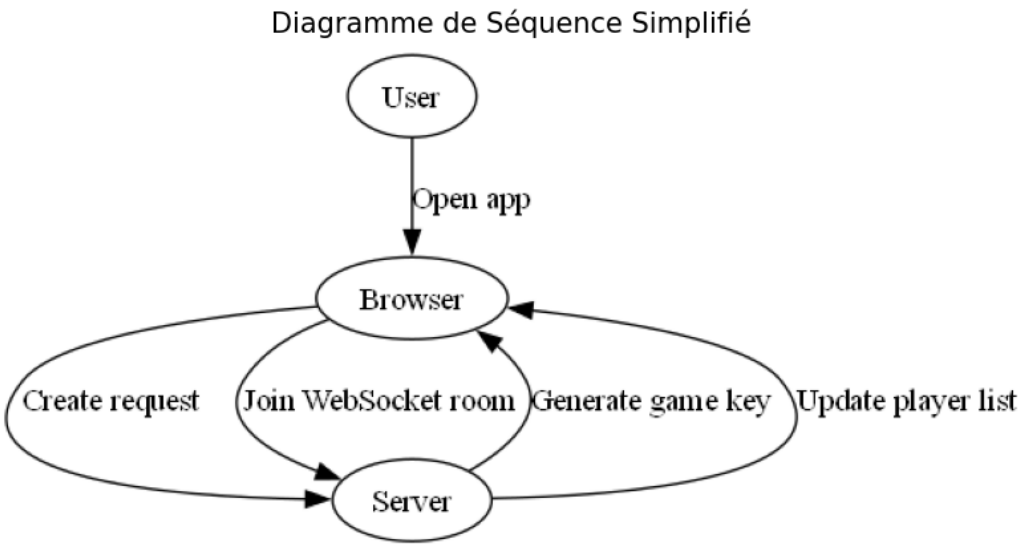

Figure 2 : Diagramme de Séquence Simplifié

c. Diagramme d'architecture

Il décrit les différents composants de l'application, y compris le serveur Flask-SocketIO, les clients web (navigateur), et les modules utilitaires.

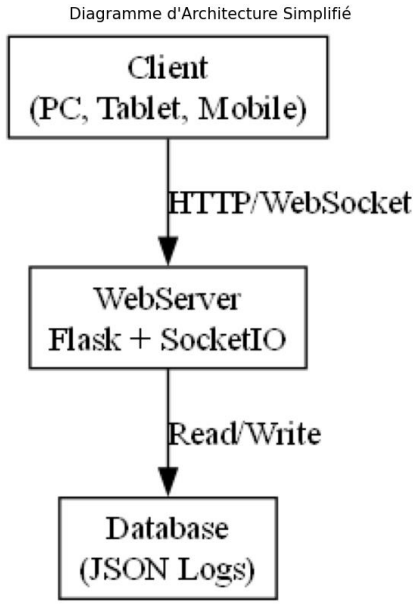

Figure 3 : Diagramme d'Architecture Simplifié

---

5. Intégration continue

L'application utilise une stratégie d'intégration continue pour garantir la qualité du code :

- **Tests unitaires** :
    - Des tests automatisés sont définis dans des fichiers comme test_app.py et les fichiers JavaScript correspondants.

- **Pipeline CI/CD** :
    - Les tests sont exécutés automatiquement sur chaque commit.
    - Un outil comme GitHub Actions ou GitLab CI est utilisé pour assurer le déploiement automatique en cas de succès des tests.

---

6. Conclusion

La solution collaborative robuste du projet Capi est mise en place grâce à une architecture performante et des choix technologiques motivés. La modularité et les fonctionnalités en temps réel offrent une possibilité d'extension à l'avenir tout en garantissant une expérience utilisateur fluide.