Atelier 2

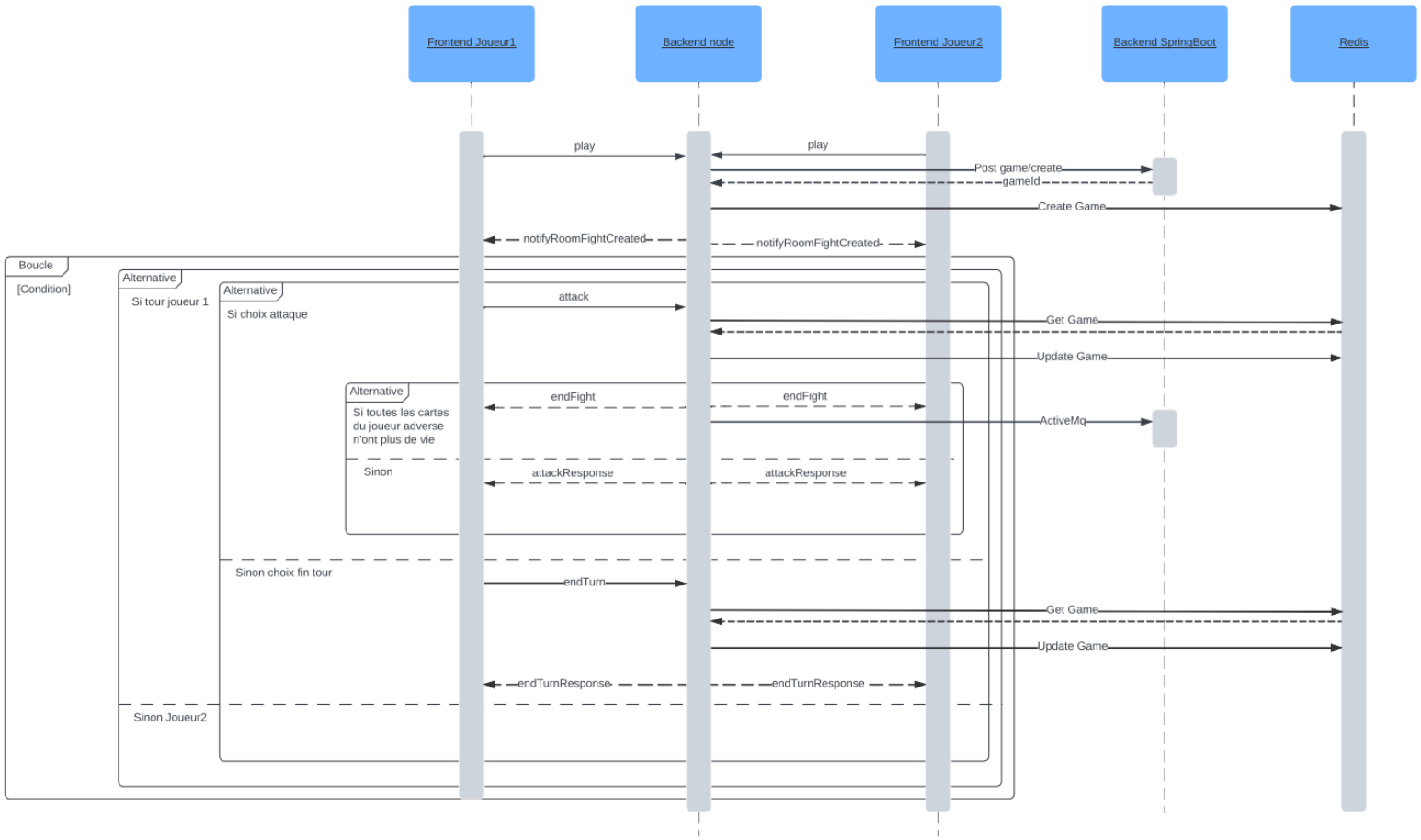
Avantages de Node.js pour un chat et un système de combat

1. **Temps réel natif avec Socket.IO**
   - **Chat :** Node.js, associé à des bibliothèques comme **Socket.IO**, facilite la mise en place d'une communication bidirectionnelle en temps réel, essentielle pour les chats entre utilisateurs.
     - **Écoute et émission d'événements :** Les messages peuvent être envoyés et reçus sans délai, offrant une expérience fluide.
     - **Scalabilité :** Node.js gère efficacement des milliers de connexions WebSocket simultanées grâce à son modèle événementiel non bloquant.
   - **Système de combat :** Les interactions dans un combat (attaques, défense, mises à jour en temps réel des points de vie, etc.) peuvent être transmises instantanément entre joueurs. Cela garantit une synchronisation précise des actions entre les participants.

2. **Performances pour les connexions simultanées**
   - **Gestion massive d'utilisateurs :** Node.js est conçu pour gérer des connexions simultanées, ce qui est crucial dans un jeu où plusieurs joueurs interagissent en temps réel.
     - Le modèle **non bloquant** assure que chaque joueur peut envoyer/recevoir des données (messages, actions de combat) sans perturber les autres.

3. **Synchronisation et événements asynchrones**
   - Node.js permet de gérer plusieurs événements en parallèle grâce à son modèle événementiel :
     - Dans un chat : réception et diffusion des messages en temps réel.
     - Dans un combat : gestion simultanée des actions des joueurs (attaque, défense) avec une faible latence.

Diagramme de séquences des interactions entre Node.js et le FrontEnd lors d'un jeu entre deux joueurs



Différences principales entre Docker et la virtualisation classique

Caractéristique	Conteneurs Docker	Virtualisation classique
Niveau d'isolation	Isolation au niveau du système d'exploitation	Isolation au niveau de la machine virtuelle (hyperviseur)
Système d'exploitation	Partage le noyau du système hôte, ce qui réduit les ressources.	Chaque VM embarque son propre système d'exploitation complet
Performances	Plus légers et rapides à démarrer	Plus lourds et lents à démarrer
Taille	Images légères -> Mo	VMs volumineuses -> Go
Gestion des ressources	Consommation minimale grâce au partage de ressources du système hôte	Plus gourmandes en CPU, RAM et stockage
Isolation forte	Moins isolés (partagent le noyau, donc potentiellement vulnérables).	Isolation complète entre VM et hôte grâce à l'hyperviseur
Flexibilité	Idéal pour des microservices ou des applications conteneurisées.	Mieux adapté pour exécuter plusieurs systèmes d'exploitation différents.