Progressive Web App :

Une **PWA** est une application web améliorée qui s’installe comme une app classique sur l’écran d’accueil, mais fonctionne dans un navigateur.

* technologies modernes (HTML, CSS, JS, Service Workers)
* Frontend **:** React,
* Backend: Node.js, Flask,
* Service Worker **:** script qui gère le cache, la connexion hors-ligne, les push notifications, etc.
* Manifest JSON **:** fichier qui décrit l’icône, nom, couleur, orientation de l’app
* Déploiement **:** hébergée comme un site web
* Installation **:** le navigateur propose “Ajouter à l’écran d’accueil”

**Avantages :**

* Un seul code pour tous les appareils (Android, iOS, PC)
* Facile à mettre à jour : une simple publication en ligne
* Aucune validation par les stores
* Peut être utilisée hors-ligne (avec cache)

**Inconvénients :**

* Support NFC très limité (voire inexistant sur iOS)
* Fonctionne moins bien sur iPhone (restrictions Apple)
* Moins performant que les apps natives pour le rendu temps réel
* Stockage local limité (mais j’ai pas réussi a comprendre pourquoi )
* Notifications Push non fiables selon le navigateur

**Application Mobile Native**

Conçue directement pour Android ou iOS.

Accès complet au matériel (ce qui va nous aider avec les NFC)

**Implémentation :**

Backend: API REST ou GraphQL (Node.js, Flask…)

NFC :

* + Utilisation de modules comme react-native-nfc-manager
  + Nécessite des autorisations spécifiques selon l’OS

Déploiement: Google Play & App Store

**Avantages :**

* Plein accès au matériel : NFC
* Expérience utilisateur fluide
* Accès hors-ligne robuste
* Peut stocker beaucoup de données

**Inconvénients :**

* Plus complexe à développer
* Requiert des builds Android/iOS séparés (même avec React Native)
* Nécessite de suivre les règles des stores (délai de validation mais je n’ai pas très bien compris)
* Mises à jour moins instantanées

En conclusion a part si on utilise un lecteur NFC externe a travers Bluetooth ce qui peut être cher mais sinon avec le PWA Android supporte très mal et IOS absolument pas.

Manière dont effet peux être intégrer au jeux

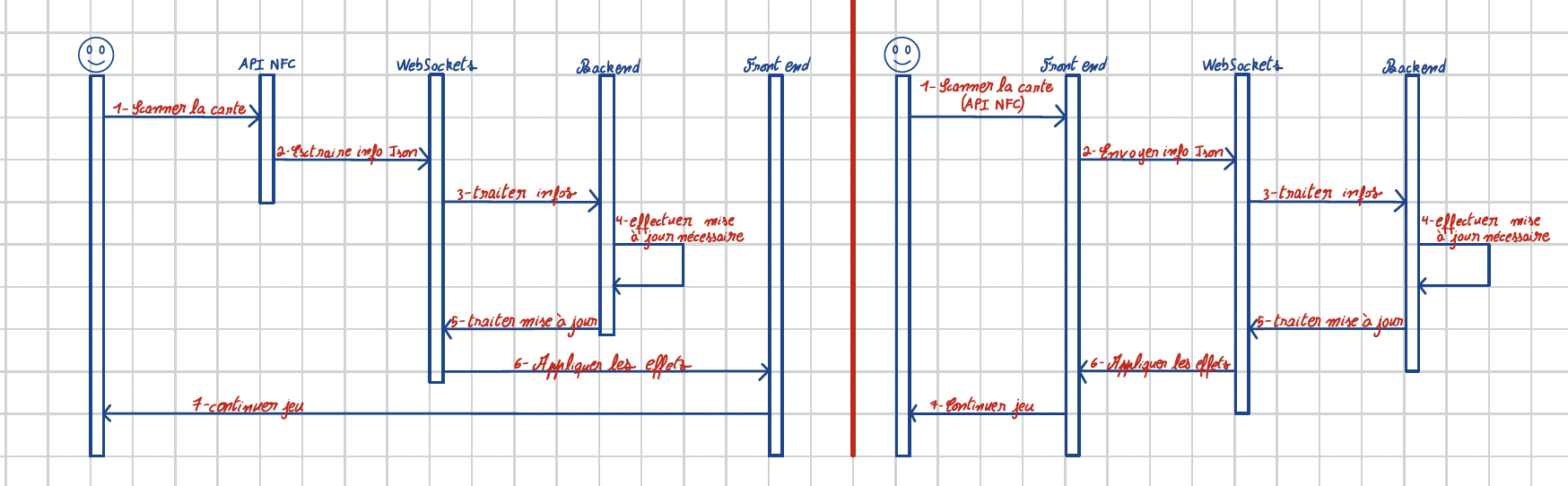
Liste des objets prédéfinis

Définir modèle générale du jeu (ensemble des classes et propriétés)

Replit

**Manière dont les effets peuvent être intégrer aux jeux :**

Le joueur scanne une carte NFC via son téléphone (React Native via react-native-nfc-manager)



* L’API NFC lit un identifiant stocké dans la carte (souvent un ID ou un JSON contenant l’effet).
* L’effet est envoyé via WebSocket ou API REST au backend (Node.js ou Flask selon les cas).
* Le backend traite l’effet : il regarde dans la base de données (NoSQL/SQL) à quoi correspond cette carte.
* Il renvoie une réponse contenant les propriétés de l’effet (type, durée, cumulabilité, etc.).
* Le frontend applique l’effet visuellement en temps réel :
  + Boost de vitesse : vitesse du joueur augmentée
  + Double saut : ajout d’un second saut dans la mécanique
  + Obscurité champ de vision réduit, effet visuel ajouté

Ces effets sont temporaires, parfois cumulables, et sont réinitialisés à chaque activation

**Liste des objets prédéfinis disponibles pour la création du monde**

| **Type d'objet** | **Détails** |
| --- | --- |
| **Blocs** | Sol, mur, plateformes (taille personnalisable) |
| **Objets interactifs** | Porte, objets à ramasser |
| **Monstres** | Oiseaux, loups, requins, etc. (liste visible dans le prototype papier) |
| **Éléments de gameplay** | Départ, arrivée |

**Caractéristiques** :

* Dimension ajustable : largeur / hauteur
* Position libre sur grille
* Couleurs modifiables
* Suppression simple