# PROJET : Pixel Together

## 1. Pitch

**Pixel Together** est une plateforme de création collaborative. Chaque utilisateur est "Maitre" de ses propres grilles de Pixel Art. Intégration à des plateforme fortement communautarisée (Facebook, Kongregates)

Le but étant de remplir une grille de pixel pour « créer » quelque chose

- **Typologie des Grilles :**
  - **Privée :** Solo ou sur invitation uniquement.
  - **Publique (Spectateur) :** Tout le monde voit, seul le propriétaire dessine.
  - **Publique (Collaborative) :** Tout le monde peut participer dans la limite de ses pixels disponibles.
- **Modération :** Le propriétaire peut bannir/expulser des utilisateurs de sa grille ainsi que donner des droits d'écriture.
- **Chat en temps réel**
- **Système de galeries, avec vote pour classement hebdo/mensuel/annuel**
- **Système partage hors écosystème**
- **Export sous forme de NFT**

**2. Économie et Progression**

Le jeu intègre une dimension RPG et économique pour fidéliser l'utilisateur (et monétiser dans le futur) : (**Uniquement dans une éventuelle V2, pas dispo dans le jeu de base vu que non-monétisable)**

- **Ressources :**
    - **Pixels :** Stock limité (ex : 100 max au Niv. 1). Régénération automatique (ex : 1 px / min).
    - **XP & Or :** 1 pixel posé = 1 XP + 1 Pièce d'or.
- **Système de Niveaux :** Monter en niveau augmente le stock max de pixels et permet de créer des grilles de plus grand format.
- **Boutique de Couleurs :** * Départ avec le **Pack Basique** (7 couleurs : Noir, Blanc, Rouge, Bleu, Vert, Jaune, Gris).
    - Achat de nouveaux **Packs thématiques** avec l'or accumulé.
    - *Évolution future :* Création de packs personnalisés via sélecteur RGB.

Plus le joueur joue, plus il est récompensé.

**3. Architecture Technique**

- **Frontend :** React (Interface) + HTML5 Canvas (Grilles) + Tailwind (pour le CSS)
- **Backend :** Node.js / Express
- **Temps Réel :** Socket.io pour la synchronisation immédiate des pixels et du chat.
- **Hébergement :** Self-hosting (Machine locale).
- **IDE :** Visual Studio Code.

**4. Infrastructure, Base de données**

**MongoDB** : Persistance, profils utilisateurs, grilles terminée (galerie) ou en pause

## 5. Limitations

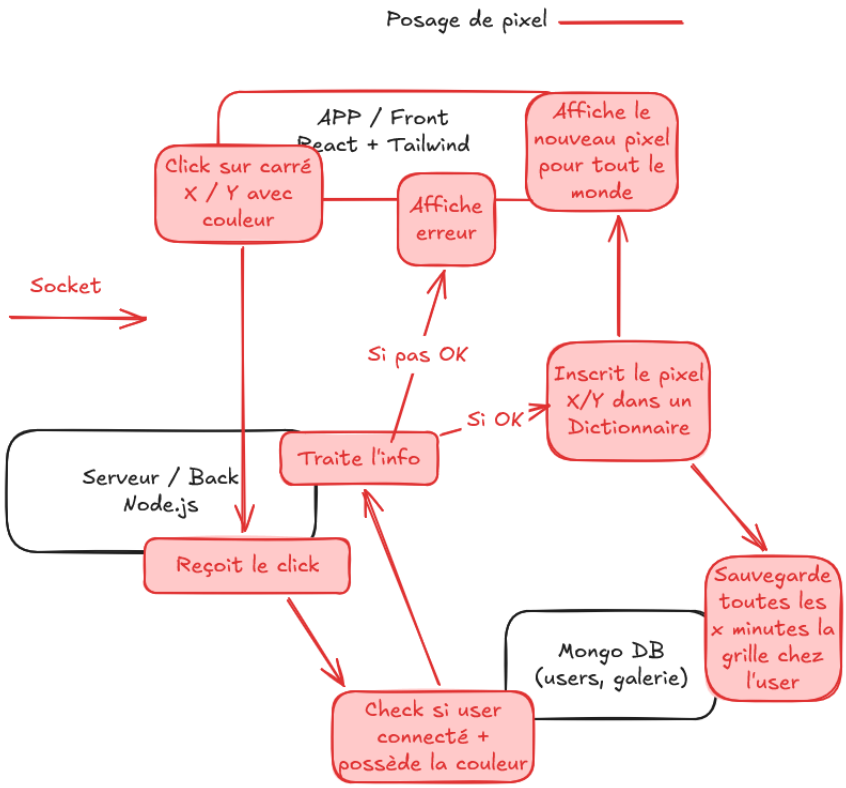
Base de données : Beaucoup de données à stocker, coût proportionnel à l'utilisation (Redis = RAM)

Hébergement : Va devoir encaisser de nombreuses interactions, un simple hébergement web classique et statique insuffisant. Connections persistantes

Taille serveur : adaptabilité, selon le flux, certaines heures plus de demande que d'autre. Scalabilité à anticiper

Juridique : Chat + liberté de création = cocktail explosif. Prévoir base juridique solide + outils de modération. CGU, RGPD, propriété intellectuelle, responsabilité personnelle

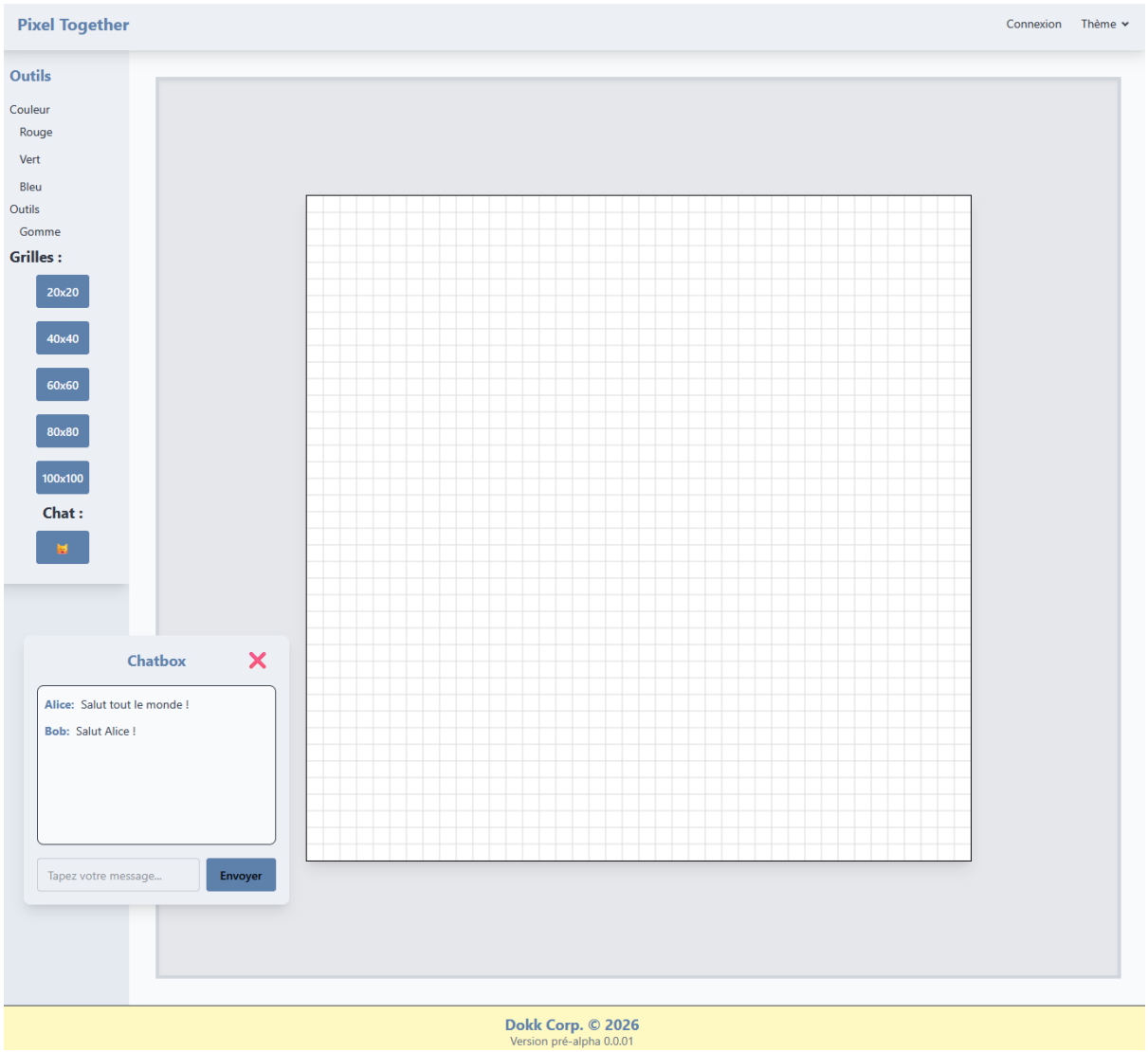
## 6. Schéma infrastructure



## 7. Interface

Simple, centrée sur la grille, ton des couleurs clair, tendance à s'effacer pour mettre en valeur la grille.
Modules adaptables : Chat, palette de couleurs, outils peuvent se déplacer, fermer
Grilles : Zoomable avec molette (hover sur la div), on peut se placer dessus avec click droit maintenu, possible de passer en plein écran (efface l'interface)

Premier jet :



**8. Road map**

**Étape 1** : Le Moteur de Jeu **(1 semaine)**
Objectif : La base du gameplay, colorier un pixel

1. **Installation**
    - Créer le projet (React pour le visuel, Node.js pour le cerveau).
    - Faire en sorte que les deux communiquent
2. **Le Dessin**
    - Afficher la grille blanche.
    - Créer la palette de couleurs (7 + blanc(gomme))
    - Changer de couleur un pixel
3. : **Le Multijoueur (V1)**
    - Connecter le clic au serveur.
    - Tester avec deux onglets ouverts

**Étape 2** : L'Architecture des Salons **(1 semaine)**
**Objectif :** Création des rooms

1. **Le Lobby (Accueil)**
    - Créer une page d'accueil simple.
    - Ajouter un bouton "Créer une grille"
    - Ajouter un champ pour donner un nom à sa grille.
2. **Les Rooms**
    - Modifier le serveur pour qu'il gère plusieurs grilles en mémoire
    - Faire marcher la commande socket.join(roomId) : séparer les joueurs dans des canaux différents.
3. **La Liste Publique**
    - Afficher sur l'accueil la liste des grilles créées par les autres.
    - Ajouter le bouton "Rejoindre" qui connecte le joueur à la bonne Room.

**Étape 3 :** Identité & Social **(1 semaine)**
**Objectif :** Les comptes, le tchat, le côté social

1. **Les Comptes (Base de données)**
    - Installer et connecter MongoDB.
    - Créer le formulaire d'inscription / connexion simple (Pseudo + Mot de passe).
    - Lier le créateur de la grille (Owner) à son compte en base de données.
2. **Le Chat (Interface)**
    - Créer la fenêtre de discussion à côté du dessin
    - Afficher les messages envoyés par les autres joueurs.
3. **Le Chat (Logique)**
    - Connecter le chat aux Rooms : seuls les joueurs de la *même* grille voient les messages.
    - Ajouter les couleurs aux pseudos dans le chat (ex: l'Owner en doré).

**Étape 4 :** Sauvegarde & Finitions **(1 semaine)**
**Objectif :** La persistance + finitions

1. **La Galerie**
    - Créer le bouton "Terminer la grille" (visible uniquement par le chef/Owner).
    - Sauvegarder le dessin final dans MongoDB
    - Créer une page "Galerie" pour voir les anciens dessins figés.
2. **La Modération (Kick/Ban)**
    - Ajouter une liste des joueurs connectés à droite de l'écran.
    - Ajouter un petit bouton "Kick" à côté des pseudos (visible uniquement par l'Owner).
    - Faire en sorte que le joueur kické soit renvoyé au menu principal.
3. **Mise en Ligne (Déploiement)**
    - Nettoyer le code
    - Test en direct

Temps total estimé : 1 mois pour un projet propre, utilisable en l'état. C'est possiblement ambitieux vu que je n'ai aucune expertise en web moderne et que le JS est beaucoup moins intuitif que python (à mon avis)