Spécifications Techniques Détaillées - Application Cultures Sauvages

1. Choix de Stack Technologique Raisonnée

Philosophie de Sélection

Pour Cultures Sauvages, nous privilégions des technologies matures, bien documentées et adaptées à une équipe associative qui pourrait évoluer. Pensons à choisir des outils comme on choisit des essences d'arbres pour un jardin : certaines sont robustes et demandent peu d'entretien, d'autres offrent des fruits savoureux mais nécessitent plus de soins.

Stack Recommandée : Python/Django + SQLite + Vue.js

Backend : Python avec Django Django est l'équivalent technique d'un couteau suisse de qualité professionnelle. Cette framework nous offre tout ce dont nous avons besoin pour construire rapidement et solidement votre application, sans nous obliger à réinventer la roue à chaque fonctionnalité.

Avantages spécifiques pour Cultures Sauvages :

- **Django Admin automatique** : Interface d'administration générée automatiquement à partir de vos modèles de données. C'est comme avoir un tableau de bord complet sans programmer une seule ligne d'interface admin.
- **Système d'authentification intégré** : Gestion des utilisateurs, permissions, sessions déjà implémentées et sécurisées par défaut.
- **ORM puissant** : Traduction automatique entre vos objets Python et votre base SQLite, avec protection contre les injections SQL.
- **Écosystème riche** : Packages existants pour calendriers, notifications, exports PDF, intégrations de paiement.

Base de Données : SQLite avec Migration vers PostgreSQL SQLite pour commencer, c'est comme planter des graines dans des pots avant de les transplanter en pleine terre. Cette approche progressive nous permet de démarrer rapidement tout en préparant l'évolution.

SQLite convient parfaitement pour vos premiers mois d'utilisation (jusqu'à plusieurs centaines d'utilisateurs et milliers d'événements). Quand votre communauté grandira, la migration vers PostgreSQL se fera naturellement grâce aux outils Django.

Frontend : Vue.js avec Nuxt.js Vue.js est l'outil idéal pour créer une interface utilisateur réactive et intuitive. Contrairement à React qui peut être intimidant, ou Angular qui est complexe, Vue.js offre une courbe d'apprentissage douce tout en restant très puissant.

Nuxt.js ajoute la structure et les bonnes pratiques automatiquement, comme avoir un jardinier expérimenté qui organise votre potager selon les meilleures méthodes.

2. Architecture Technique Détaillée

Structure des Répertoires et Modularité

```
cultures_sauvages/
├── backend/
│   ├── config/                # Configuration Django
│   ├── apps/
│   │   ├── authentication/    # Module auth et permissions
│   │   ├── users/             # Gestion des utilisateurs
│   │   ├── events/            # Événements et inscriptions
│   │   ├── projects/          # Projets à long terme
│   │   ├── notifications/     # Système de notifications
│   │   └── api/               # API REST centralisée
│   ├── utils/                 # Outils communs
│   └── tests/                 # Tests automatisés
├── frontend/
│   ├── components/            # Composants Vue réutilisables
│   ├── pages/                 # Pages de l'application
│   ├── store/                 # Gestion d'état Vuex
│   ├── plugins/               # Extensions Nuxt
│   └── assets/                # Resources statiques
└── docs/                      # Documentation du projet
```

Cette organisation en modules séparés facilite la maintenance et permet à plusieurs développeurs de travailler simultanément sans se marcher sur les pieds.

Gestion des Permissions Avancée

Le système de permissions mérite une attention particulière car il détermine qui peut faire quoi dans votre application. Plutôt que d'utiliser un système binaire (autorisé/interdit), nous implémentons un système de permissions contextuelles.

Exemple concret : Un bénévole peut modifier les événements qu'il a créés, mais pas ceux des autres. Un membre actif peut modifier tous les événements de sa thématique. Un admin peut tout modifier. Cette granularité respecte la hiérarchie naturelle de votre association tout en évitant la rigidité excessive.

python

```python
# Exemple de logique de permission contextuelle
class EventPermission:
    def can_edit_event(self, user, event):
        if user.is_admin:
            return True
        if user.is_membre_actif and event.thematique in user.thematiques_responsal
            return True
        if event.organisateur == user:
            return True
        return False
```

Système de Notifications Intelligent

Les notifications dans votre application ne doivent pas être du spam, mais des informations utiles au bon moment. Nous implémentons un système qui apprend des préférences utilisateur et adapte sa fréquence.

Types de notifications gérées :

- **Immédiates** : Confirmation d'inscription, modification d'événement auquel on participe
- **Quotidiennes** : Résumé des nouvelles opportunités correspondant au profil
- **Hebdomadaires** : Bilan d'activité, rappel des événements à venir
- **Sur-mesure** : Notifications basées sur les compétences et disponibilités déclarées

Gestion des Fichiers et Media

Votre éco-tiers-lieu produira beaucoup de contenu visuel : photos d'événements, plans de projets, documents de formation. Le système de gestion des médias doit être à la fois simple pour les utilisateurs et optimisé pour les performances.

Stratégie d'optimisation : Compression automatique des images, génération de miniatures, stockage organisé par date et type d'événement. Pour l'évolutivité, préparation à l'intégration avec des services cloud (AWS S3, Cloudinary).

3. Algorithmes Métier Critiques

Algorithme de Matching Compétences/Besoins

C'est le cœur intelligent de votre application, celui qui transforme une base de données statique en outil de mise en relation dynamique. L'algorithme analyse les compétences déclarées par chaque utilisateur et les met en correspondance avec les besoins exprimés dans les événements et projets.

Principe de fonctionnement : L'algorithme utilise un système de scoring multicritères qui prend en compte :

- **Correspondance des compétences** : Pourcentage de match entre compétences possédées et recherchées
- **Disponibilités temporelles** : Croisement entre les créneaux libres de la personne et les besoins de l'événement
- **Historique d'engagement** : Bonus pour les personnes actives, malus temporaire après participation récente (éviter le burn-out)
- **Proximité géographique** : Pour les événements nécessitant une présence physique
- **Préférences thématiques** : Priorité aux domaines d'intérêt déclarés

Exemple pratique : Quand quelqu'un crée un événement "Atelier construction éco-habitat", l'algorithme identifie automatiquement les membres ayant des compétences en menuiserie, maçonnerie écologique, ou simplement marqué leur intérêt pour les "Travaux". Il leur envoie une notification personnalisée expliquant pourquoi cet événement pourrait les intéresser.

Algorithme de Planification Intelligente

La planification des événements dans une association est complexe car elle doit tenir compte de multiples contraintes : disponibilités des animateurs, saisonnalité des activités, éviter les conflits avec d'autres événements importants.

Notre algorithme de suggestion de créneaux analyse :

- **Historique de participation** : Quels jours et heures attirent le plus selon le type d'activité
- **Conflits potentiels** : Autres événements, jours fériés, vacances scolaires
- **Contraintes météorologiques** : Pour les activités extérieures en Guadeloupe
- **Ressources disponibles** : Salles, matériel, animateurs qualifiés

Système de Réputation et Engagement

Plutôt qu'un simple compteur de participation, nous développons un système qui valorise la qualité et la diversité de l'engagement. Cette approche encourage l'implication durable plutôt que la participation mécanique.

Métriques d'engagement calculées :

- **Assiduité** : Ratio présence/inscription avec pondération selon la prévisibilité
- **Diversité thématique** : Bonus pour les personnes qui s'engagent sur plusieurs domaines
- **Contribution au collectif** : Mesure des retours positifs des autres participants
- **Initiative** : Valoration des propositions d'événements ou d'améliorations

4. Optimisations de Performance Critiques

Stratégie de Cache Multi-Niveaux

Votre application doit rester fluide même quand votre communauté grandit. Le système de cache que nous implémentons fonctionne comme la mémoire musculaire d'un artisan : les actions fréquentes deviennent automatiques et instantanées.

Cache applicatif : Les requêtes fréquentes (liste des événements à venir, profils des organisateurs actifs) sont mises en cache pour éviter de solliciter la base de données à chaque chargement de page.

Cache de session : Les informations spécifiques à chaque utilisateur connecté restent en mémoire durant sa navigation, rendant l'interface très réactive.

Cache de contenu statique : Images, CSS, JavaScript sont optimisés et mis en cache côté navigateur pour des temps de chargement rapides, crucial pour l'accès mobile en Guadeloupe où la connexion peut être variable.

Optimisation des Requêtes Base de Données

La performance d'une application repose largement sur l'efficacité de ses interactions avec la base de données. Nous implémentons plusieurs stratégies d'optimisation :

Requêtes préventives (Prefetching) : Quand un utilisateur charge la liste des événements, nous chargeons aussi les informations des organisateurs en une seule requête plutôt qu'une par événement.

Pagination intelligente : Au lieu de charger tous les événements passés, nous utilisons une pagination qui charge les contenus à la demande, maintenant des temps de réponse constants même avec des milliers d'événements archivés.

Index stratégiques : Les recherches fréquentes (événements par date, utilisateurs par statut) sont optimisées par des index de base de données soigneusement placés.

5. Sécurité et Protection des Données

Architecture de Sécurité Défensive

La sécurité de votre application ne repose pas sur un seul mécanisme mais sur une stratégie de défense en profondeur, comme les multiples barrières protectrices d'un écosystème naturel.

Authentification renforcée : Au-delà du simple mot de passe, nous implémentons :

- Politique de mots de passe adaptatifs (plus stricts pour les admins)
- Limitation des tentatives de connexion avec blocage temporaire
- Sessions sécurisées avec expiration automatique
- Option d'authentification à deux facteurs pour les comptes sensibles

Protection contre les attaques communes :

- **Injection SQL** : Prévenue par l'ORM Django qui échappe automatiquement les requêtes
- **Cross-Site Scripting (XSS)** : Protection par échappement automatique des contenus dans les templates
- **Cross-Site Request Forgery (CSRF)** : Tokens de sécurité intégrés à tous les formulaires
- **Brute Force** : Limitation du taux de requêtes par IP et par utilisateur

Conformité RGPD Intégrée

Votre association collecte des données personnelles sensibles, nous intégrons donc la conformité RGPD directement dans l'architecture technique :

Consentement granulaire : Les utilisateurs peuvent choisir précisément quelles données ils partagent et pour quels usages. Par exemple, accepter les notifications d'événements mais refuser les statistiques d'utilisation.

Droit à l'oubli technique : Fonctionnalité permettant la suppression complète et vérifiable des données d'un utilisateur, avec gestion des références dans l'historique des événements (anonymisation plutôt que suppression pour préserver l'intégrité des bilans).

Audit trail automatique : Journalisation de tous les accès aux données personnelles, permettant de répondre aux demandes de transparence des utilisateurs.

6. Monitoring et Maintenance Préventive

Système de Surveillance Proactif

Une application pour une association ne peut pas se permettre de tomber en panne pendant un événement important. Notre stratégie de monitoring anticipe les problèmes avant qu'ils n'affectent les utilisateurs.

Métriques surveillées en temps réel :

- Temps de réponse des pages critiques (inscription, connexion)

- Taux d'erreur des opérations sensibles (paiements, envoi de notifications)
- Utilisation des ressources serveur (CPU, mémoire, espace disque)
- Qualité de l'expérience utilisateur (temps de chargement par type de connexion)

Alertes intelligentes : Le système distingue les anomalies temporaires des problèmes structurels. Une montée de trafic lors d'un événement populaire ne déclenche pas les mêmes alertes qu'une réelle dégradation de performance.

Stratégie de Sauvegarde et Récupération

Vos données associatives sont irremplaçables : années d'historique d'engagement, relations entre membres, mémoire collective des projets. La stratégie de sauvegarde doit être aussi rigoureuse que la conservation des archives d'une bibliothèque.

Sauvegardes automatisées multi-niveaux :

- **Quotidiennes** : Sauvegarde complète de la base de données avec rétention de 30 jours
- **Hebdomadaires** : Archive complète incluant les fichiers média, conservée 6 mois
- **Mensuelles** : Sauvegarde à long terme avec compression, conservation illimitée

Tests de récupération : Procédure mensuelle de test de restauration sur environnement de développement pour vérifier l'intégrité des sauvegardes.

Cette approche technique peut sembler complexe, mais chaque élément répond à un besoin concret de votre association. Souhaiteriez-vous que nous approfondissions certains aspects particuliers, ou préférez-vous que nous abordions maintenant la planification du développement et la mise en œuvre pratique de ces spécifications ?