**CARTHAGREEN**

GROW THE GREEN SOUL

**1. Architecture Backend (Nest.js)**

Votre backend Nest.js est actuellement à l'état initial. Pour un système hydroponique intelligent, je recommande l'architecture suivante:

**Modules à développer:**

1. **Module Capteurs**
   * Gestion des différents capteurs (pH, température, humidité, lumière)
   * Endpoints pour recevoir et stocker les données des capteurs
   * Service de validation et traitement des données
2. **Module Dispositifs**
   * Gestion des actionneurs (pompes, éclairage, ventilateurs)
   * Endpoints pour contrôler les dispositifs
   * Logique d'automatisation
3. **Module Alertes**
   * Système de détection d'anomalies
   * Service de notification (push, email)
   * Configuration des seuils d'alerte personnalisables
4. **Module Utilisateurs**
   * Authentification et autorisation
   * Gestion des profils et préférences
   * Rôles et permissions
5. **Module Plantes**
   * Base de données des plantes et leurs besoins spécifiques
   * Recommandations personnalisées
   * Suivi de croissance
6. **Module Statistiques**
   * Agrégation des données
   * Rapports de performance
   * Prédictions et optimisations

**2. Intégration IoT**

* **Protocole MQTT** pour la communication entre les capteurs et le backend
* **Websockets** pour les mises à jour en temps réel sur l'application mobile
* **API RESTful** pour les interactions non temps réel

**3. Base de données (MongoDB)**

* Collections pour: Utilisateurs, Dispositifs, Lectures de capteurs, Plantes, Alertes, Configurations
* Utilisation d'index pour optimiser les requêtes fréquentes
* Implémentation de TTL (Time To Live) pour gérer les données historiques

Recommandations techniques pour le backend Nest.js :

1. **Structure des dossiers  
   Une image contenant texte, capture d’écran, menu, Police

   Le contenu généré par l’IA peut être incorrect.**
2. **Intégration MongoDB**
   * Utiliser @nestjs/mongoose pour l'intégration avec MongoDB
   * Définir des schémas clairs pour chaque collection
   * Implémenter des repositories pour l'accès aux données
3. **Authentification et Sécurité**
   * JWT pour l'authentification
   * Stratégies de protection des routes
   * Validation des données entrantes
4. **Communication IoT**
   * Intégrer un broker MQTT comme mqtt.js ou aedes
   * Implémenter des services d'écoute pour les messages des capteurs
   * Mettre en place un système de file d'attente pour gérer les pics de données
5. **API Documentation**
   * Utiliser Swagger pour documenter l'API
   * Faciliter l'intégration avec le frontend Flutter

**Fonctionnalités innovantes à considérer :**

1. **Apprentissage automatique pour l'optimisation**
   * Prédiction des besoins en eau et nutriments
   * Détection précoce des maladies
   * Optimisation des cycles de croissance
2. **Contrôle adaptatif**
   * Ajustement automatique des paramètres en fonction des conditions
   * Modes d'économie d'énergie et d'eau
3. **Marketplace de recettes hydroponiques**
   * Partage de configurations optimales pour différentes plantes
   * Communauté d'utilisateurs
4. **Intégration avec des services météorologiques**
   * Ajustement des paramètres en fonction des prévisions
   * Optimisation de la consommation d'énergie

**Prochaines étapes pour le développement backend :**

1. **Configuration de la base de données**
   * Installer les dépendances: @nestjs/mongoose, mongoose
   * Configurer la connexion à MongoDB
2. **Création du module Capteurs**
   * Définir les schémas pour les différents types de capteurs
   * Créer les endpoints pour recevoir les données
3. **Mise en place de l'authentification**
   * Implémenter le module utilisateurs
   * Configurer JWT et les stratégies de protection
4. **Intégration MQTT**
   * Configurer un broker MQTT
   * Développer les services d'écoute

**USER MANAGEMENT**

****

**Fonctionnalités implémentées**

**1. Authentification et gestion des utilisateurs**

* **Inscription des utilisateurs :**
  + **Création de compte avec nom, email et mot de passe**
  + **Hachage sécurisé des mots de passe avec bcrypt**
  + **Vérification de l'unicité des emails**
  + **Upload de photo de profil lors de l'inscription (optionnel)**
* **Vérification d'email :**
  + **Génération de tokens de vérification**
  + **Envoi d'emails de vérification avec lien**
  + **Page HTML de confirmation de vérification**
* **Connexion des utilisateurs :**
  + **Authentification par email/mot de passe**
  + **Génération de tokens JWT**
  + **Protection des routes avec JwtAuthGuard**
* **Authentification Google OAuth :**
  + **Intégration de la stratégie Google OAuth2**
  + **Création ou mise à jour des utilisateurs via Google**
  + **Gestion des informations de profil Google**
* **Gestion de profil :**
  + **Mise à jour des informations utilisateur**
  + **Changement de mot de passe**
  + **Upload et mise à jour de photo de profil**
* **Récupération de mot de passe :**
  + **Génération et envoi d'OTP (One-Time Password)**
  + **Vérification d'OTP**
  + **Réinitialisation sécurisée du mot de passe**

**2. Gestion des fichiers**

* **Upload de photos de profil :**
  + **Configuration de Multer pour la gestion des fichiers**
  + **Validation des types de fichiers (images uniquement)**
  + **Limitation de taille (5MB max)**
  + **Génération de noms de fichiers uniques**
* **Serveur de fichiers statiques :**
  + **Configuration pour servir les fichiers uploadés**
  + **Accès aux images via URL**

**3. Envoi d'emails**

* **Service d'email :**
  + **Configuration de Nodemailer**
  + **Templates d'emails pour la vérification et la réinitialisation**
  + **Support HTML avec style intégré**

**Technologies et dépendances principales**

* **NestJS : Framework backend**
* **MongoDB : Base de données NoSQL**
* **Mongoose : ODM pour MongoDB**
* **Passport.js : Authentification**
* **JWT : Tokens d'authentification**
* **Multer : Gestion des uploads de fichiers**
* **Nodemailer : Envoi d'emails**
* **bcrypt : Hachage de mots de passe**
* **class-validator : Validation des données**

**Prochaines étapes potentielles**

1. **Modules IoT :**
   * **Création des schémas pour les appareils et capteurs**
   * **API pour l'enregistrement et la gestion des appareils**
   * **Stockage et traitement des données des capteurs**
2. **Système d'alertes :**
   * **Configuration des seuils d'alerte**
   * **Notifications en temps réel**
   * **Historique des alertes**
3. **Documentation API :**
   * **Intégration de Swagger pour la documentation**
   * **Description des endpoints et des modèles**
4. **Tests :**
   * **Tests unitaires pour les services**
   * **Tests d'intégration pour les contrôleurs**
   * **Tests end-to-end**

**Le backend CarthaGreen a déjà une base solide avec un système d'authentification complet et une gestion des utilisateurs robuste. Les fonctionnalités actuelles permettent de gérer l'ensemble du cycle de vie des utilisateurs, depuis l'inscription jusqu'à la gestion de profil, avec plusieurs méthodes d'authentification et de récupération de compte.**