

GreenRoots
Milan LAMETTE



Sommaire

1

Introduction au projet

- L'équipe projet GreenRoots

2

Vision et cadre méthodologique

- Outils organisationnels
- Méthodologie Agile (SCRUM)
- Suivi collaboratif GIT/GitHub

3

Conception / Modélisation

- Modélisation (MCD, MLD)

4

Architecture et Stack technique

- Sécurité et outils

5

Développements & Technicité

- Arborescence projet
- Fichiers du projet pour en expliquer les choix et les utilités app.js — Serveur Express : cœur du projet

6

Bonnes pratiques, retours et perspectives

- Retour d'expérience
- Axes d'amélioration
- Documentation & maquettes
- Résultats

I. INTRODUCTION AU PROJET

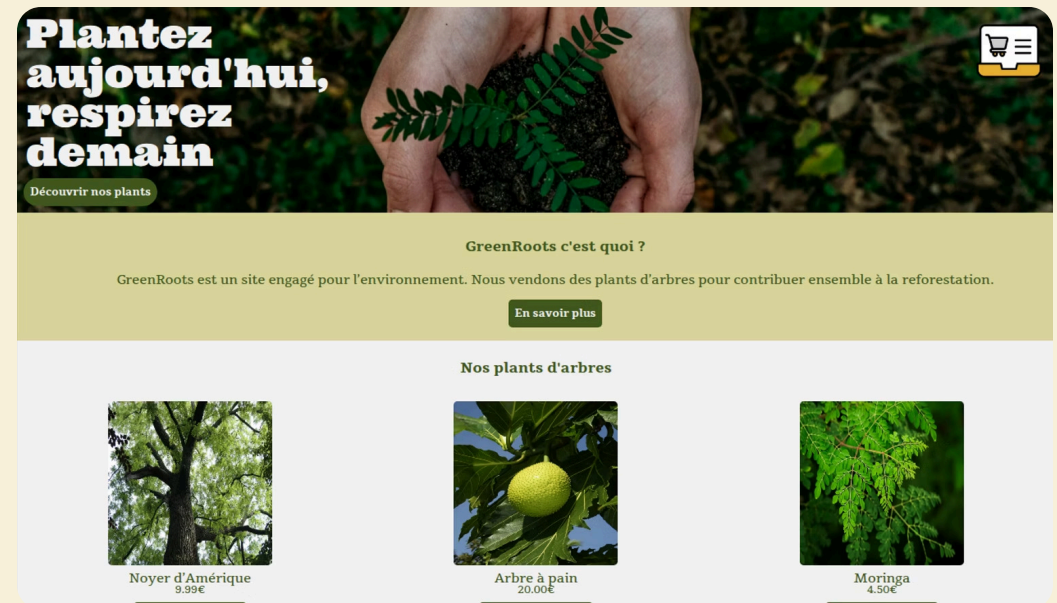
Présentation du projet

GreenRoots : Qu'est ce que c'est ?

- Plateforme e-commerce
- Acheter des arbres à planter par des entreprise partenaires
- Suivre l'évolution des arbres
- Partager son expérience en laissant des avis

✓ Valeur ajoutée

Une démarche concrète et simple : permettre à tous de lutter contre la déforestation de masse.



L'équipe projet GreenRoots



Milan Lamette

Product Owner



Scott Gallagher

Lead Developer



Maxence de Saint Leger

Front-End Developer



Vincent Rousselot

Scrum Master

Chaque rôle a été essentiel, de la conception à la livraison, pour assurer organisation, qualité technique et évolution rapide.

II. VISION ET CADRE MÉTHODOLOGIQUE

Vision

- GreenRoots vise à simplifier et encourager l'engagement de tous dans la reforestation.
- Agir concrètement pour l'environnement en achetant des arbres, en suivant leur impact et en participant activement à la restauration des écosystèmes.

Objectifs

Utilisateurs

Inscription, connexion, gestion de profil personnalisée

E-commerce

Achat d'arbres avec panier, favoris, commandes, avis

Administration

Back-office pour gérer produits, commandes, utilisateurs, avec rôles distincts

Cahier des charges

Users stories =>

Définir MVP + Evolutions

Contraintes Techniques (Technologies et Exigences)

Architecture

User connected	Accéder/Paramétrer son compte	Modifier ses informations (nom, prénom, adresse, ...)	Page Mon compte
	Ajouter au panier	Mettre de côté les arbres qu'ils souhaitent acheter sur une page dédiée	Page produits, détail d'un prod
	Acceder au panier	Visualiser tous les arbres (type d'arbre, qté, prix) ajouter au panier qu'il souhaite acheter	Page Panier, menu header de toutes les pages
	Ajouter à ses favoris	Mettre de côté les arbres qu'ils souhaitent ajouter en favoris sur une page dédiée	Page produits, détails
	Accéder à ses favoris	Visualiser tous les arbres (type d'arbre, prix) ajouter à ses favoris	Page Favoris
	Accéder Historique des commandes passées	Visualiser les commandes passées avec date, prix, articles	Page commande
	Ajouter un avis	Ecrire un commentaire sur un arbre spécifique	Page produits
Visiteur	Accéder aux pages produits, détail produits	Avoir accès aux arbres proposés sans pouvoir ajouter au panier ou favoris	

Outils organisationnels

- Équipe organisée selon la méthodologie SCRUM
- Scrumboard partagé pour le pilotage du projet

Sprint en cours	Dates	Equipe back	Equipe Front	Veille technique
Sprint 0	10/04/2025	Présentation du projet, définition des besoins, attribution des rôles		Figma et Git/Github
Sprint 0	17/04/2025	Création du MCD / MLD	Création des maquettes desktop et mobile	Outils de conception de MCD (Looping)
Sprint 0	18/04/2025	Review du MCD / MLD + dictionnaire de données + réunion de suivi		
Sprint 0	22/04/2025	Réunion de suivi + finalisation du MCD/MLD/Dictionnaire de données + finalisation maquettes mobile + catalogue produits + imports assets back et front		
Sprint 0	23/04/2025	présentation du projet + wireframe mobile		
Sprint 1	24/04/2025	Mise en place du serveur express, création de la BDD, implémentation des produits	Découpage en composants, création des premières pages statiques	git/github, PSQL
Sprint 1	25/04/2025	Création des routes et des controllers	finalisation des pages en statiques et commencement des versions desktop	
Sprint 1	28/04/2025	mise en place de l'authentification, hashage des mdp et conservation en session	finalisation des versions desktop	
Sprint 1	29/04/2025	déploiement du back sur la VM kourou	correction des conflits CSS, dynamisation de la landing page	
Sprint 1	30/04/2025	Présentation fin de sprint 1, correctif des images en back, dynamiser page d'un produit et tout les produits		
Sprint 2	05/05/2025	modales de connexion, menu déroulant, début de l'authentification		React
Sprint 2	06/05/2025	modales d'ajout de produit, modification/suppresion d'un produit		
Sprint 2	07/05/2025	ajouter au panier/ aux favoris, retirer des favoris		
Sprint 2	12/05/2025	système d'inscription, gestion du panier, modification du compte, vérification sécurité et retrospective des bugs		sécurité (xss, sql)
Sprint 2	13/05/2025	dynamisation du panier, création d'une commande, présentation fin de sprint 2		
Sprint 3	14/05/2025	Ajouter au panier depuis landing page/favoris, suppression du panier		
Sprint 3	15/05/2025	dynamisation des commandes, modales de confirmation		docker
Sprint 3	16/05/2025	Correctifs dernier bugs, déploiement du back et du front		surge
Sprint 3	19/05/2025	Préparation présentation		
Sprint 3	20/05/2025	Demos		

Le scrumboard est le cœur de notre organisation agile, de la priorisation à la validation des tâches !

Méthodologie Agile (SCRUM)

Sprints courts

Projet morcelé en sprints courts, pour améliorer en continu et réagir rapidement



Stand-ups quotidiens

Stand-ups quotidiens pour faire le point, levée de blocages

Revue de code

Pull requests sur GitHub pour revue de code, validation collective

Suivi collaboratif GIT/GitHub



Branches

Travailler en branches pour chaque ajout de feature



Pull requests

Pull requests et merge pour garantir la qualité du code livré



Historique

Historique traçable à tout moment, même en équipe répartie

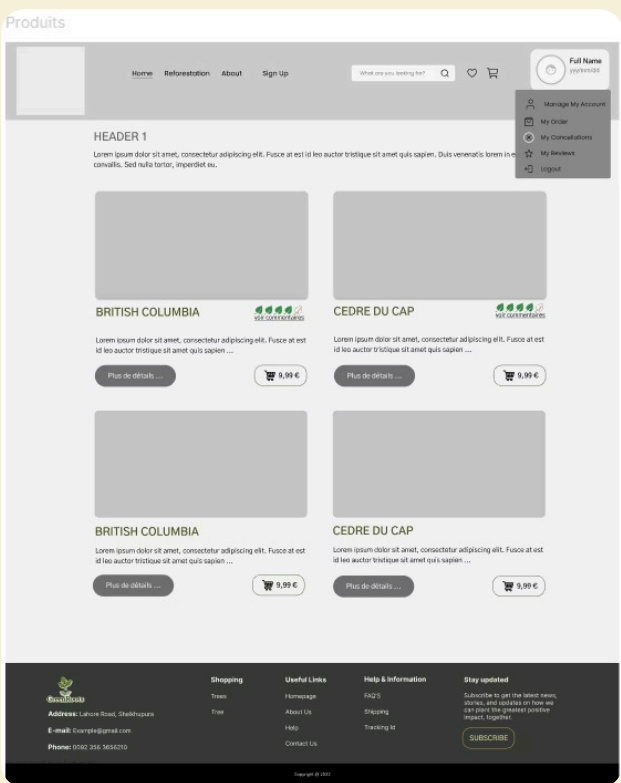
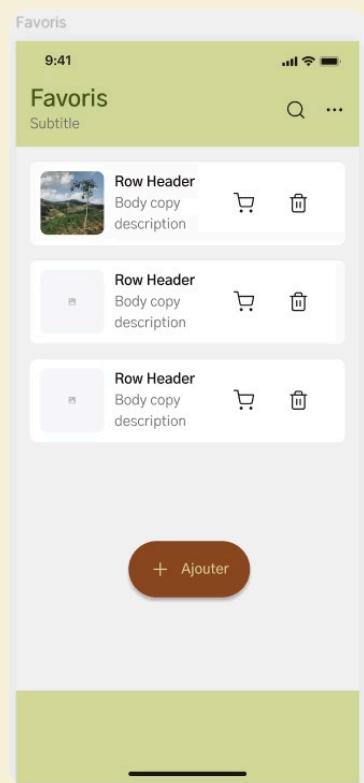
modification du session.controller.js pour récupérer le id_user correctement. LametteMilan
installation de bcrypt et JWT, création du controller de session LametteMilan
Merge pull request #3 from O-clock-Samoussas/test/http Lamette Milan
ajout du fichier test.http et de quelques request , puis modif du fichier user.model.js: supprimer ...
Merge pull request #2 from O-clock-Samoussas/router ScottGallagher93
routes finished ScottGallagher93
correction bokmark to bookmark ScottGallagher93
Merge pull request #1 from O-clock-Samoussas/branch1 ScottGallagher93
controllers done ScottGallagher93
break ScottGallagher93
controller branch 1 in progress ScottGallagher93
test package.json LametteMilan
type module added ScottGallagher93
ddb created and seeded ScottGallagher93
tables created ScottGallagher93
test model 2 ScottGallagher93
Add Sequelize models ScottGallagher93
Delete node_modules directory ScottGallagher93
Main setup finished, switching to dev branch ScottGallagher93
Main setup finished, switching to dev branch ScottGallagher93
init test ScottGallagher93

File Edit View Repository Branch Help
Current repository: green-roots-back | Current branch: dev | Pull origin: Last fetched 3 minu...
Changes | History | Merge pull request #6 from O-clock-Samoussas/ModifPassword
Select branch to compare...
Merge pull request #6 from O-clock-Sa... Lamette Milan • 3 months ago
modification du 'zip_code' en base de ... LametteMilan • 3 months ago
modification du zip code INTEGER en V... LametteMilan • 3 months ago
récuperer panier d'un user connecté : ... MaxenceSL • 3 months ago
modif pour accéder aux favoris, pour aj... MaxenceSL • 3 months ago
ajout middleware multer pour upload L... MaxenceSL • 3 months ago
-modification du script dans package.js... LametteMilan • 3 months ago
fixed db create and reset ScottGallagher93 • 3 months ago
create-table fixed ScottGallagher93 • 3 months ago
minor corrections renaming webp images ScottGallagher93 • 3 months ago
cors added
4 changed files
controllers/user.controller.js
@@ -17,6 +17,7 @@ export const userController = {
 res.status(404).json({ error: "User not found"
});
}
},
+
 async createUser(req, res) {
 try {
 const userData = req.body;
 @@ -30,7 +31,7 @@ export const userController = {
 return res.status(400).json({ error: "Le mo
t de passe doit faire au moins 8 caractères" });
 }
 }
+
 userData.password = await bcrypt.hash(password,
 10);
+ userData.password = await bcrypt.hash(userData.
password, 12);
 userData.user_role = userData.user_role || ROLE
S.CUSTOMER;
 const newUser = await User.create(userData);
 @@ -51,52 +52,49 @@ export const userController = {

III. CONCEPTION / MODÉLISATION

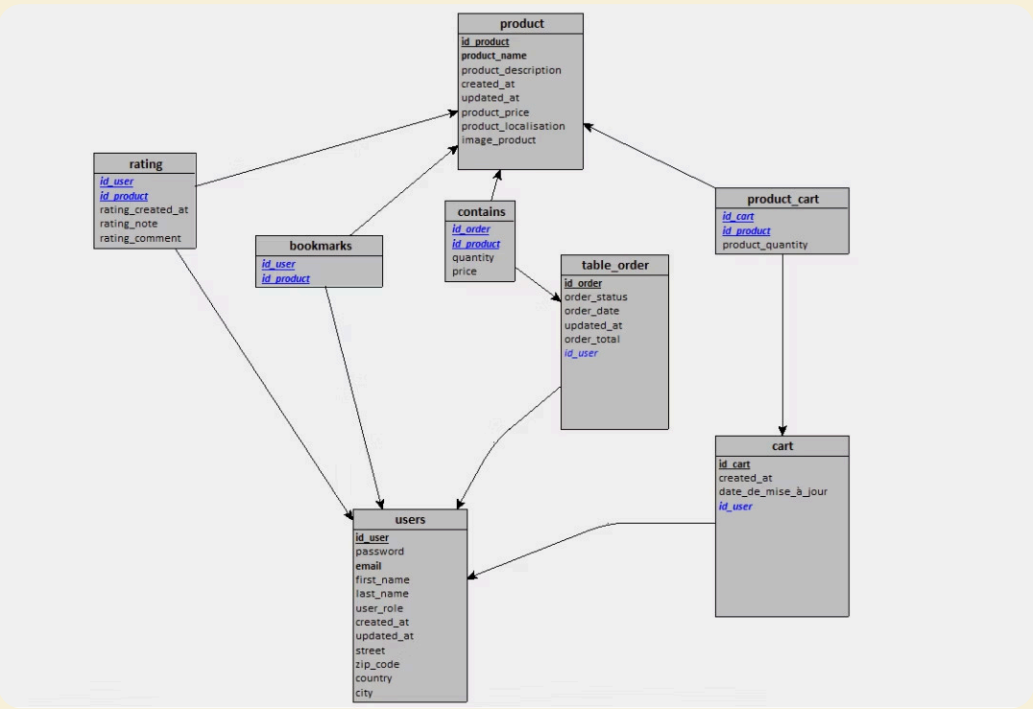
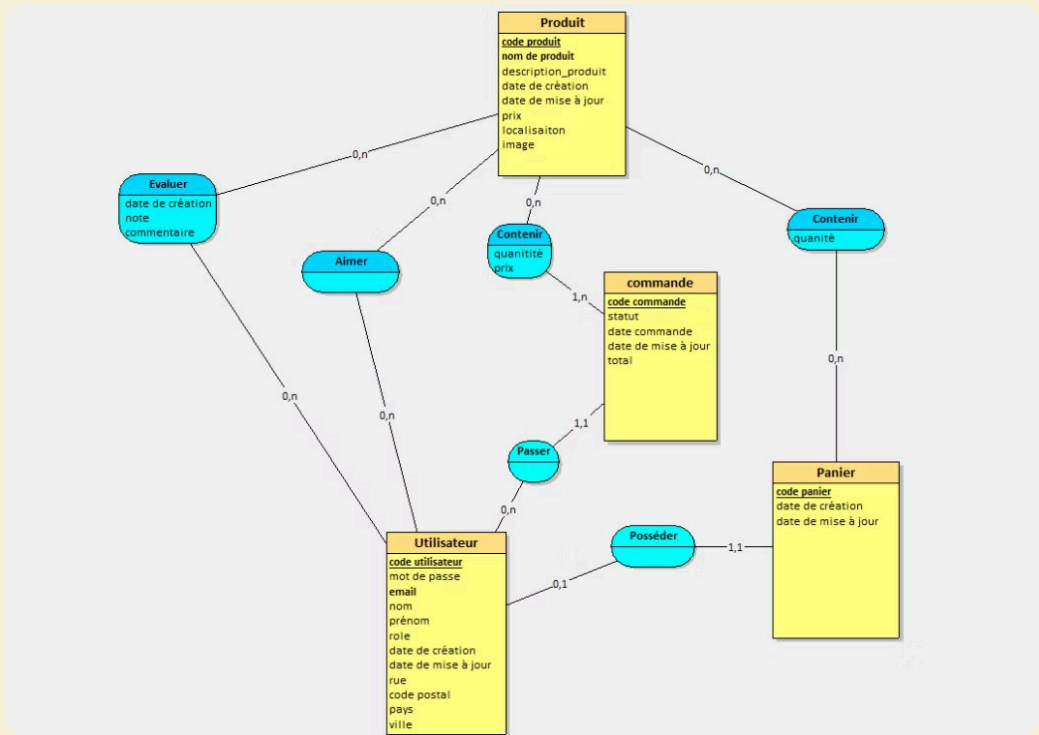
Maquettage

Lien Figma



L'expérience visuel utilisateur ayant été validée , nous avons pu nous lancé dans la modélisation !

Modélisation (MCD, MLD)



MCD

Décrit toutes les entités

MLD

Structure SQL exhaustive

Looping nous a permis de définir les relations métiers.

IV. ARCHITECTURE ET STACK TECHNIQUE

Stack technique & justification

Back-end :



Node.js

- API performante
- Asynchronicité



Express

- Gestion des routes et middlewares



Sequelize

- CRUD
- Gestion des migrations et des associations



PostgreSQL

- Gestion des données, relationnelles et jointures

Front-end

React 19 + TypeScript

- Construction d'interfaces utilisateur dynamiques.
- Détection précoce des erreurs.

Vite

- Serveur de développement Hot Reload.

Axios

- Appel Api.
- Gestion d'erreur.

React Router / Sass

- Gestion SPA.
- Structuration et optimisation des styles CSS.

Ce socle donne performance, sécurité, évolutivité.

Sécurité et outils

bcrypt

- hachage des mots de passe

JWT

- méthode encodage des informations dans un jeton

multer

- middleware Node.js qui permet d'uploader et gérer des images.

```
async createUser(req, res) {
  try {
    const userData = req.body;
    // Déstructuration des champs attendus dans userData
    const { email, password, first_name, last_name } = userData;

    // Vérification que tous les champs obligatoires sont présents
    if (!email || !password || !first_name || !last_name) {
      return res.status(400).json({ error: "Tous les champs sont requis" });
    }

    // Vérification de la longueur minimale du mot de passe
    if (password.length < 8) {
      return res.status(400).json({ error: "Le mot de passe doit faire au moins 8 caractères" });
    }

    // Hachage du mot de passe avec bcrypt, facteur de salage 12
    userData.password = await bcrypt.hash(userData.password, 12);

    // Affectation d'un rôle par défaut si non précisé
    userData.user_role = userData.user_role || ROLES.CUSTOMER;

    // Création d'un nouvel utilisateur en base de données avec les données validées
    const newUser = await User.create(userData);
  } catch (error) {
    console.error('Erreur lors de la création de l\'utilisateur:', error);
    return res.status(500).json({ error: "Erreur serveur" });
  }
}
```

Administration

Gestion des commandes et des produits en toute simplicité

Commandes

Produits

Ajouter produit

Ajouter un produit

Résumé produit:

Nom du produit:

Titre du produit

Localisation:

Nom

Localisation

Description:

Description

Titre de la description

Image:

Image principale

Image de détail

Informations:

email

password

first_name

last_name

user_role

street

zip_code

country

city

Créer le produit

```
const authMiddleware = async (req, res, next) => {
  // Récupération du token JWT depuis le header Authorization (format "Bearer TOKEN") ou dans les cookies
  const token = req.headers.authorization?.split(' ')[1] || req.cookies?.token;
  console.log("Token reçu:", token);

  if (!token) {
    return res.status(401).json({ error: 'Token manquant' });
  }

  // Vérification et décodage du token JWT avec la clé secrète définie dans les variables d'environnement
  const decoded = jwt.verify(token, process.env.JWT_SECRET);
  console.log("Token décodé:", decoded);

  // Recherche de l'utilisateur en base via la clé primaire récupérée dans le token
  const user = await User.findById(decoded.userId);
  console.log("Utilisateur trouvé:", user ? user.get() : null);

  if (!user) {
    console.log("Échec: userId du token ne correspond à aucun utilisateur");
    return res.status(401).json({ error: 'Utilisateur introuvable' });
  }

  // Injection des informations utilisateur dans l'objet requête pour les étapes suivantes
  req.userId = user.id;
  req.userRole = user.user_role;
}
```

```
POST http://localhost:3000/users
{
  "title": "Produit",
  "last_name": "Dupont",
  "user_role": "admin",
  "street": "2 rue de la Paix",
  "zip_code": "75001",
  "country": "France",
  "city": "Paris"
}

## 1. Connexion
POST http://localhost:3000/sessions
Content-Type: application/json
{
  "email": "admin@green-roots.com",
  "password": "admin123"
}

## 2. Connexion
POST http://localhost:3000/sessions
Content-Type: application/json
{
  "email": "admin@green-roots.com",
  "password": "admin123"
}
```

Query Results (Previous)

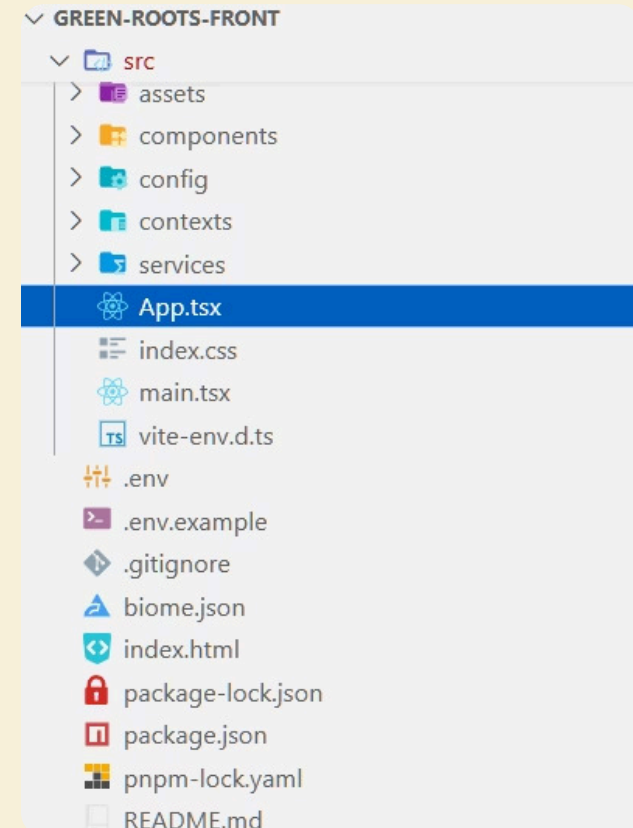
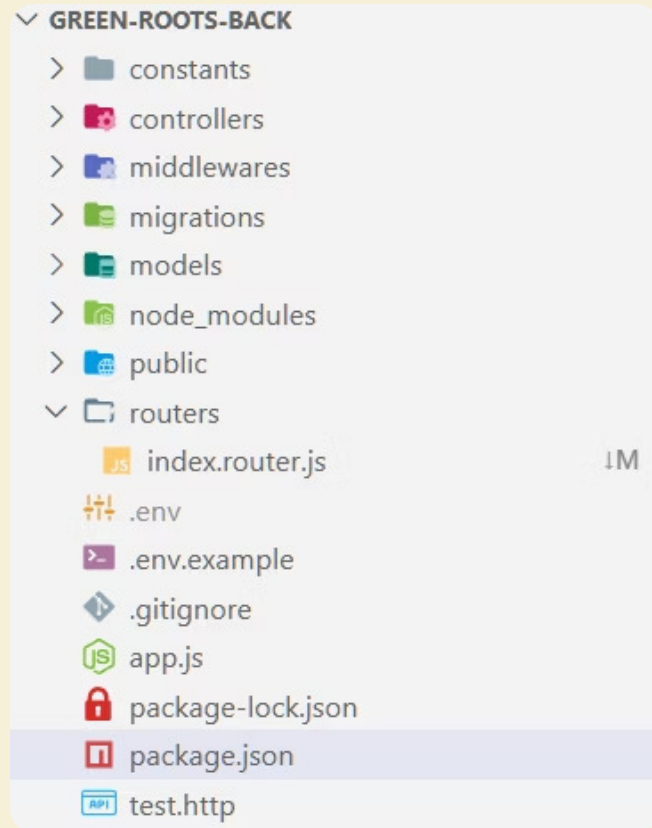
2025-07-30T21:12:17.093Z

Le résultat de la requête est le suivant :

```
{
  "email": "admin@green-roots.com",
  "password": "admin123",
  "first_name": "Admin",
  "last_name": "Dupont",
  "user_role": "admin",
  "street": "2 rue de la Paix",
  "zip_code": "75001",
  "country": "France",
  "city": "Paris",
  "created_at": "2025-07-30T21:12:17.093Z"
}
```

V. DÉVELOPPEMENTS & TECHNICITÉ


Arborescence projet



tsconfig.app.json
tsconfig.json
tsconfig.node.json
vite.config.ts

Chaque dossier — que ce soit côté back-end ou front-end — est conçu pour séparer les rôles et faciliter les évolutions.

Frontend React (Products.tsx) — gestion état

```
<div className="modal-message-content">
  
  <strong className="error">Oopss ... </strong>
  <p className="error-message">Vous devez être connecté</p>
  <button type="button" className="button-retry" onClick={closeModal}>
     retour
  </button>
```

```
utilisateur authentifié : false
Error: Failed to load resource: the server responded with a status of 401 (Unauthorized)
Error lors de l'ajout au panier : ▶ AxiosError
    at Landing.tsx:85
```

GreenRoots c'est quoi ?

Oopss ...

Vous devez être connecté

 retour

Un ajout panier quand l'utilisateur est non connecté. Une requête = un affichage instantané côté utilisateur.



Routes et contrôleurs (index.router.js)

```
import express from 'express';
import { userController } from '../controllers/user.controller.js'; // Toute la
logique création user
import { authMiddleware, roleMiddleware } from
'../middlewares/auth.middleware.js';

const router = express.Router();
router.post('/users', userController.createUser); // Ouvert à tous
(inscription)
router.get('/users', authMiddleware, roleMiddleware('admin'),
userController.getAllUsers); // Vifier attribut user_role === admin

export { router };
```

Chaque changement de politique sécurité est local.

Migration SQL — Initialisation automatisée

```
// Définition des requêtes de création de tables
const tables = [
  {
    name: 'users',
    query: `
      CREATE TABLE users(
        id_user SERIAL PRIMARY KEY,          -- Identifiant unique auto-incrémenté
        password VARCHAR(50) NOT NULL,       -- Mot de passe (hashé), obligatoire
        email TEXT NOT NULL UNIQUE,          -- Email unique et obligatoire
        first_name VARCHAR(100) NOT NULL,    -- Prénom obligatoire
        last_name VARCHAR(100) NOT NULL,     -- Nom obligatoire
        user_role VARCHAR(50) NOT NULL,      -- Rôle utilisateur (ex: admin, customer), obligatoire
        created_at TIMESTAMP WITH TIME ZONE NOT NULL, -- Date de création
        updated_at TIMESTAMP WITH TIME ZONE NOT NULL, -- Date de dernière mise à jour
        street VARCHAR(100) NOT NULL,        -- Rue de l'adresse, obligatoire
        zip_code VARCHAR(10) NOT NULL,       -- Code postal, obligatoire
        country VARCHAR(50) NOT NULL,        -- Pays, obligatoire
        city VARCHAR(50) NOT NULL           -- Ville, obligatoire
      );
    `,
  },
  // la définition des autres tables sur le même modèle
];
```

```
// Chargement des variables d'environnement depuis le fichier .env
dotenv.config();

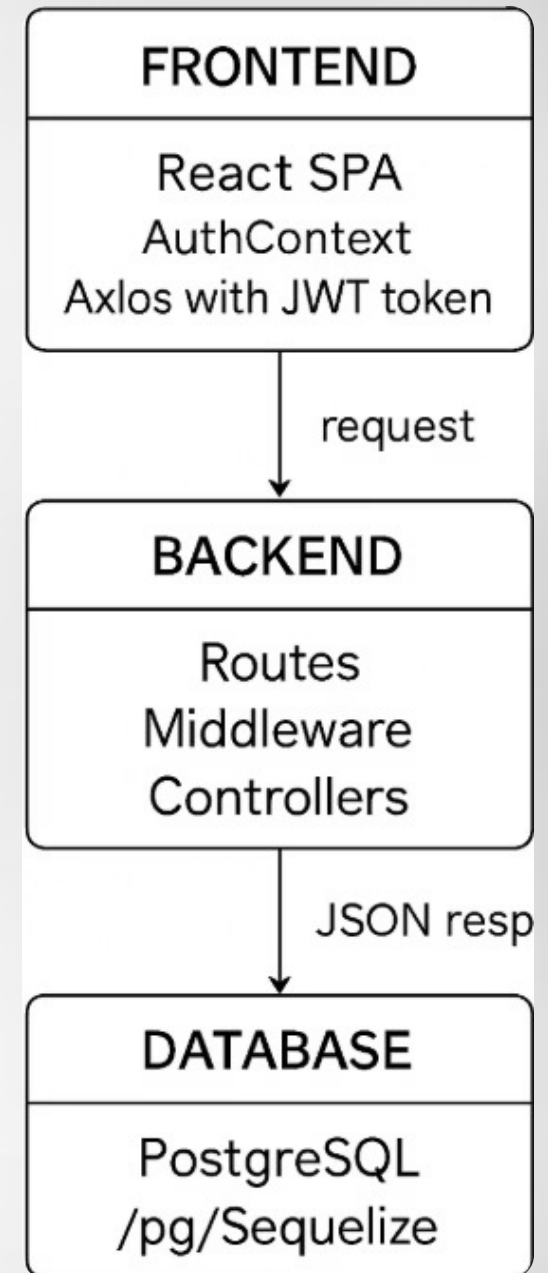
// Connexion à la base de données via les variables .env
const client = new Client({
  user: process.env.DB_USER,
  host: process.env.DB_HOST,
  database: process.env.DB_NAME,
  password: process.env.DB_PASSWORD,
  port: process.env.DB_PORT,
});

// Fonction principale pour créer les tables
const createTables = async () => {
  try {
    // Connexion à la base
    await client.connect();
    console.log('Connected to the database');

    // Liste des tables à supprimer si elles existent déjà
    const tablesToDrop = [
      'contains',
      'bookmarks',
      'rating',
      'product_cart',
      'table_order',
      'cart',
      'product',
      'users'
    ];
  }
};
```

Avec ce script, chaque dev/tuteur peut régénérer la base à l'identique.

Cheminement des données



VI. BONNES PRATIQUES

Récapitulatif sécurité & bonnes pratiques

- ☐ **Protection des routes**
Middleware auth/role partout où besoin
- ☐ **Sécurité des données**
Mot de passe hashés stocké dans un token
- ☐ **Validation**
Modèles et scripts testés / validés avant tout merge



Retour d'expérience

Collaboration

Partage d'expériences, prise en charge spontanée des obstacles

Rigueur organisationnelle

Scrumboard validé quotidiennement

Points forts

Évolution technique

Prise en main de technologies modernes (Node.js, React 19, TypeScript, JWT)

Axes d'amélioration (en cours)



Sécurité et accessibilité

- Protection des données (xss, csrf, stockage token)
- Contraste



Déploiement

Continuer docker




Expérience utilisateur

Optimiser UX

Documentation du projet & maquettes




Git Hub

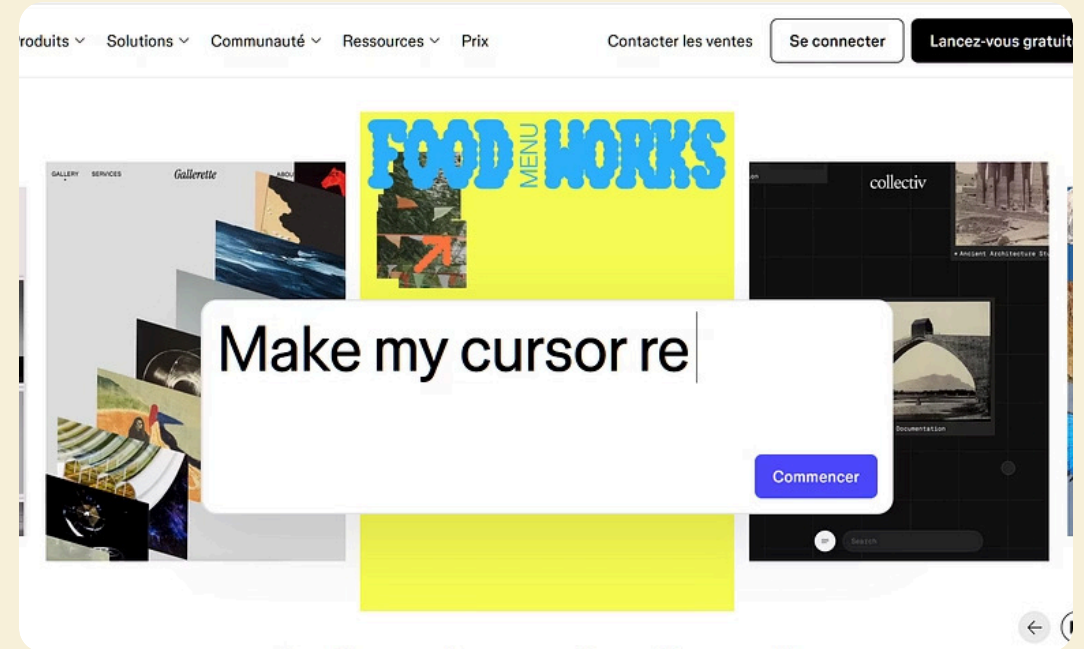


GitHub


LametteMilan – Overview

LametteMilan has 5 repositories available.
Follow their code on GitHub.






Figma



Figma

Mobile Templates (Community)

Created with Figma



Documentation et veille technologique

Stack front-end :

Mdn SQL

Sequelize

Node.js

Express

Stack back-end :

React v19

Typescript

Axios

Sass

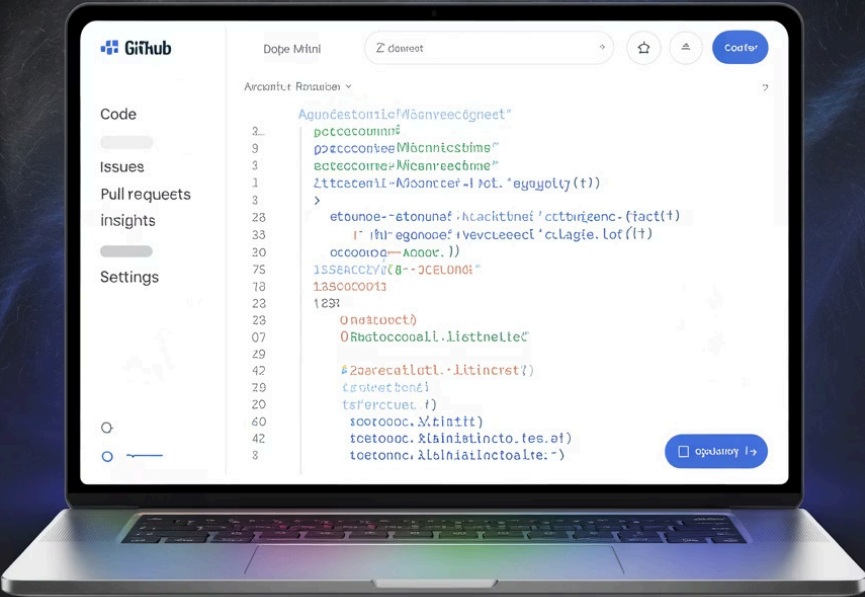
React-router

Dépôt GitHub & Résultats

Dépôt GitHub

Green-Roots_back-end

Green-Roots_front-end



"Open source project: Advanced machine learning algorithms"

Explore repository

Résultats et perspectives

- Plateforme à 100% conforme au MVP
- Code structuré facilitant évolutivité