🛠️ Documentation technique - EcoRide

📌 Objectif du projet

EcoRide est une application web de covoiturage orientée écologie, développée dans le cadre du **Titre Professionnel Développeur Web & Web Mobile**. Elle permet aux utilisateurs de proposer ou rechercher un trajet en voiture.

⚙️ Environnement de développement

J'ai choisi un environnement simple mais complet pour bien apprendre à gérer un projet web :

- **Framework Frontend** : Next.js 13 avec le système de pages (pas le dossier app)
- **Langage** : JavaScript avec JSX
- **Styling** : Tailwind CSS, pour une mise en page rapide et responsive
- **Backend** : API Routes de Next.js (simple à intégrer au projet)
- **Stockage** : `db.json` (fichier JSON pour simuler une base de données)
- **IDE** : Visual Studio Code
- **Contrôle de version** : Git avec GitHub (branches par fonctionnalité)
- **Déploiement** : Vercel, qui permet de tester mon app en ligne gratuitement

---

📁 Structure de l'application

```
/ecoride
├── components      → Navbar, Footer, boutons, etc.
├── data            → db.json (données simulées : utilisateurs, trajets...)
├── pages
│   ├── index.js            → Accueil + recherche
│   ├── connexion.js        → Connexion / création de compte
│   ├── profil.js           → Profil utilisateur
│   ├── ajouter-trajet.js   → Chauffeurs : proposer un trajet
│   ├── covoiturages
│   │   ├── index.js        → Liste des trajets
│   │   └── [id].js         → Détail d'un trajet
│   ├── contact.js          → Formulaire de contact
│   └── api                 → Toutes les routes API (utilisateurs, trajets...)
├── public          → Logo, images
├── styles          → Global.css (peu utilisé car Tailwind)
├── scripts         → Fichiers SQL simulés (non utilisés réellement)
```

├── docs → Tous les livrables PDF

🧠 Choix techniques expliqués

Je voulais une stack simple mais complète pour tout apprendre, du front au back :

- **Next.js** : pour avoir le frontend + backend dans le même projet
- **Tailwind CSS** : rapide à prendre en main, très pratique pour le responsive
- **JSON (db.json)** : facile à lire, modifier, et suffisant pour un projet étudiant
- **API Routes** : permet de créer mes propres endpoints sans serveur externe
- **localStorage** : pour simuler une session utilisateur côté client

Tout est pensé pour que je comprenne bien comment ça fonctionne, sans frameworks trop complexes.

🗒️ Modèle de données (simplifié)

> J'ai imaginé une structure de données simple pour gérer les utilisateurs et les trajets. Chaque utilisateur peut créer ou rejoindre un covoiturage.

Utilisateur
- id
- pseudo
- email
- motdepasse
- role (utilisateur / admin / employé)
- trajets[]

Trajet
- id
- départ
- arrivée
- date
- prix
- écologique (true/false)
- conducteurId
- participants[]

🔐 Sécurité & limitations

Comme c’est un projet étudiant, j’ai gardé les choses simples. Je sais qu’il manque des éléments importants côté sécurité, mais je sais aussi ce que je pourrais ajouter ensuite.

- Les mots de passe sont en clair dans le JSON (pas de cryptage)
- Pas de vrai système de session utilisateur
- Les rôles admin/employé ne sont pas protégés côté back
- Les données sont juste enregistrées dans un fichier local, pas dans une vraie base de données

💡 Ce que je pourrais faire plus tard (avec ce que je connais) :

- Ajouter une page de déconnexion pour vider le localStorage
- Empêcher l’accès à certaines pages si l’utilisateur n’est pas connecté
- Vérifier le rôle côté front pour adapter l’affichage (admin, chauffeur...)
- Commencer à utiliser une base de données (ex : SQLite via un outil ou MongoDB avec un modèle simple)

🔄 Fonctionnement de l’app (version simple)

Voici comment l’utilisateur se déplace dans l’application :

Utilisation

Visiteur
├── Page d’accueil
├── Recherche de trajet
├── Connexion / Inscription

Utilisateur
├── Accès au profil
├── Ajout de trajet
├── Historique des covoiturages
└── Participation à un trajet

Séquence (connexion)

Formulaire → API /login → Vérifie l'email + mot de passe
→ Si OK → stocké dans localStorage → redirection vers /profil

---

🚀 Déploiement de l’application

J’ai utilisé Vercel parce que c’est super simple : je connecte mon GitHub et ça déploie tout seul.

- Site en ligne ici : https://ecoride-black.vercel.app
- À chaque push sur la branche `main`, Vercel déploie automatiquement la nouvelle version
- GitHub utilisé : https://github.com/Nico-Slrn/ecoride

✅ Conclusion

J’ai voulu un projet simple, complet, et compréhensible. Je suis parti de zéro, j’ai appris à gérer le front, le back, les routes API, les composants, et même la structure d’un projet réel.

Il me reste encore plein de choses à apprendre, mais ce projet m’a permis de bien comprendre les bases, et surtout comment tout fonctionne ensemble. C’est une première vraie application que je peux montrer avec fierté.