Neurona - Apprends, Partage, Progresse

Développeur Web & Web Mobile

Thomas PENA–BERMOND

09/01/2025

Table des matières avec pagination

Introduction

Ce dossier présente la réalisation du projet Neurona, une plateforme collaborative augmentée par l'intelligence artificielle. Ce projet à été développé dans le cadre de la validation du titre professionnel Dwwm (Développeur Web & Web Mobile). Il synthétise les compétences acquises en conception du développement front-end et back-end.

Présentation générale

Contexte

Le développement de l' information évolue à grande vitesse. Les développeurs font face à une documentation souvent trop technique, ou des forums souvent obsolètes. La recherche d'information devient fastidieuse. C'est dans ce contexte qu'est né Neurona, un "hub" de connaissances centralisé maintenue par la communauté et assistée par l'IA pour garantir sa pertinence et sa clarté.

Objectifs

L'objectif principale est de fournir un outil qui simplifie l'apprentissage continue :

- Pour les juniors : Offrir un parcours d'apprentissage progressifs et explicite.
- Pour les seniors : Proposer un espace pour partager l'expertise et rester informer.

Résumé du projet

Neurona est une application web (SaaS) pensée pour les développeurs qui cherchent à apprendre, progresser et partager leurs connaissances dans un environnement collaboratif. Son rôle est de centraliser en un seul endroit ce qui est souvent dispersé : la documentation, les discussions communautaires et les actualités.

Cahier des charges

Besoins fonctionnels

- Authentification & Gestion de compte : Inscription, Connexion, Modification, Suppression.
- Documentation : Création assistée par IA, Recherche, Consultation structurée.
- Communauté : Espace d’entraide permettant de poster des sujets.
- Veille : Intégration d’articles récents via api pour permettre aux utilisateurs de suivre l'actualité.

Besoin techniques

- Front-end : Interface simple, composent réutilisables.
- Back-end : API sécurisé, gestion de base de données.
- Performance : Chargement rapide, optimisation des requêtes IA.

Utilisateurs cibles

- Développeurs en formation ou juniors.
- Développeurs expérimentés : Formateurs ou Mentors Techniques.

Contraintes

- Respect du RGPD : Protection des données personnelles.
- Accessibilité : Respect des normes WCAG.
- Budget : Utilisation de technologies open-source et gratuites.

Gestion de projet

Méthodologie

Le projet a suivi une méthodologie Agile (Scrum) :

- Utilisation de Notion pour la gestion du projet.
- Utilisation de Trello pour le suivi des tâches.
- Utilisation de Clockify pour pour le suivi.
- Utilisation de Canva pour la présentation du projet.

Outils

- Notion : Centralisation des ressources (Note d'intention, cahier des charges, etc.).
- Trello : Gestion des sprints avec tableau (A faire, En cours, En attente, Terminer).
- Clockify : Mesure du temps passé par tâche.
- Canva : Création des supports de présentations

Planning & Répartition des tâches

- Semaine 1 (27/10 - 05/11): Initialisation du projet, mise en place de l'authentification back-end, débogage des fournisseurs OAuth, et début de l'implémentation front-end.
- Semaine 2 (05/11 - 12/11): Développement des fonctionnalités "Membres", intégration de l'IA pour la génération de documents.
- Semaine 3 (12/11 - 19/11): Développement des fonctionnalités communautaire (CRUD), intégration du système veille technologique, et tests sur les fonctionnalités de l'ia.
- Semaine 4 (19/11 - 21/11): Graphique (scss, responsive mobile, dark mode), mise en place de Google Analytic, déploiement sur vercel/render et préparation de l'oral.

Analyse et conception

Diagrammes et architecture

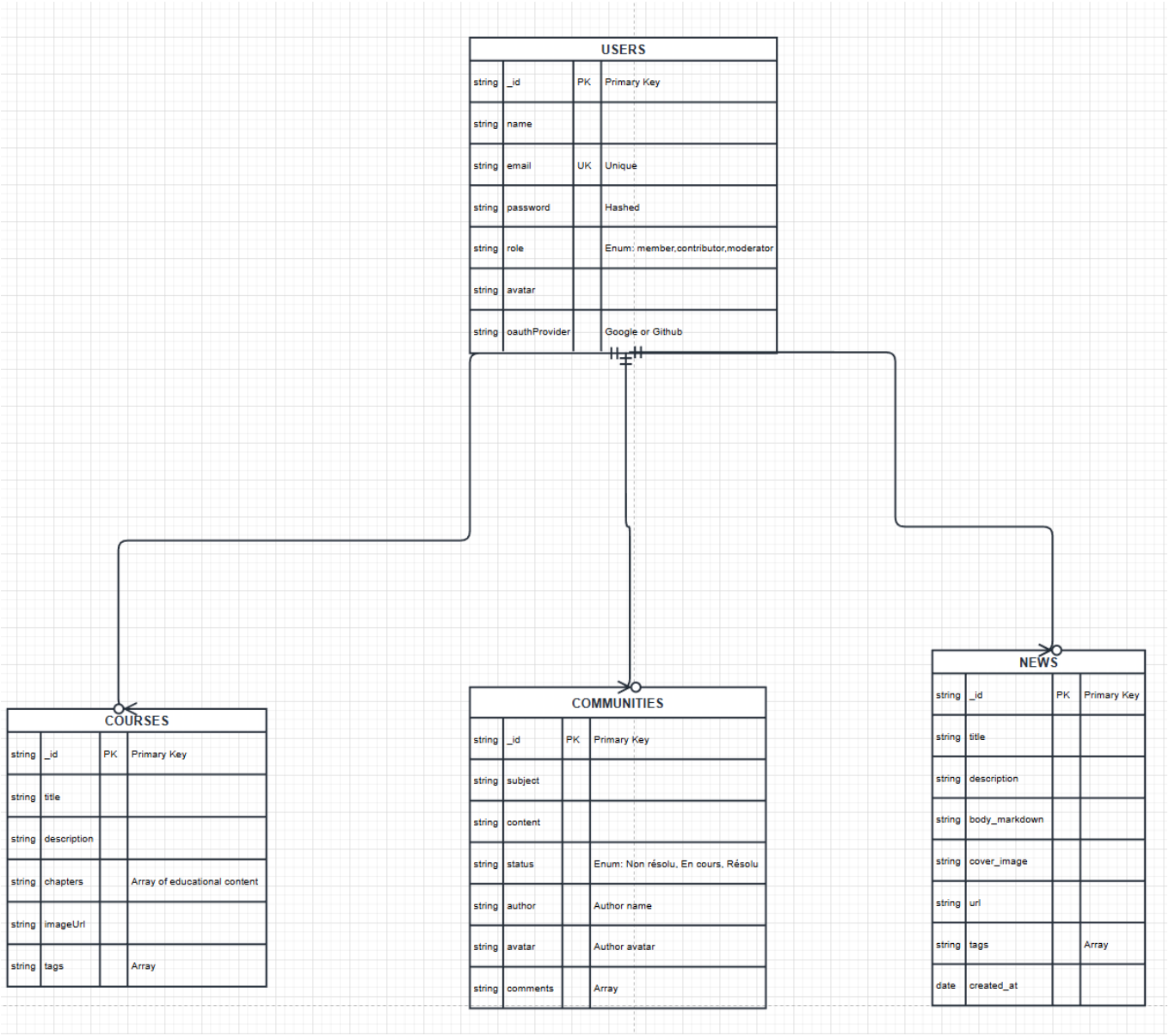
L’architecture retenue est de type MVC (Modèle-Vue-Contrôleur) coté back-end, utilisée via une API REST par le front-end.

- Client : React + Vite
- Serveur : Node + Express
- Base de données : MongoDB

Schéma de base de données

Utilisation de Mongoose pour modéliser les données.

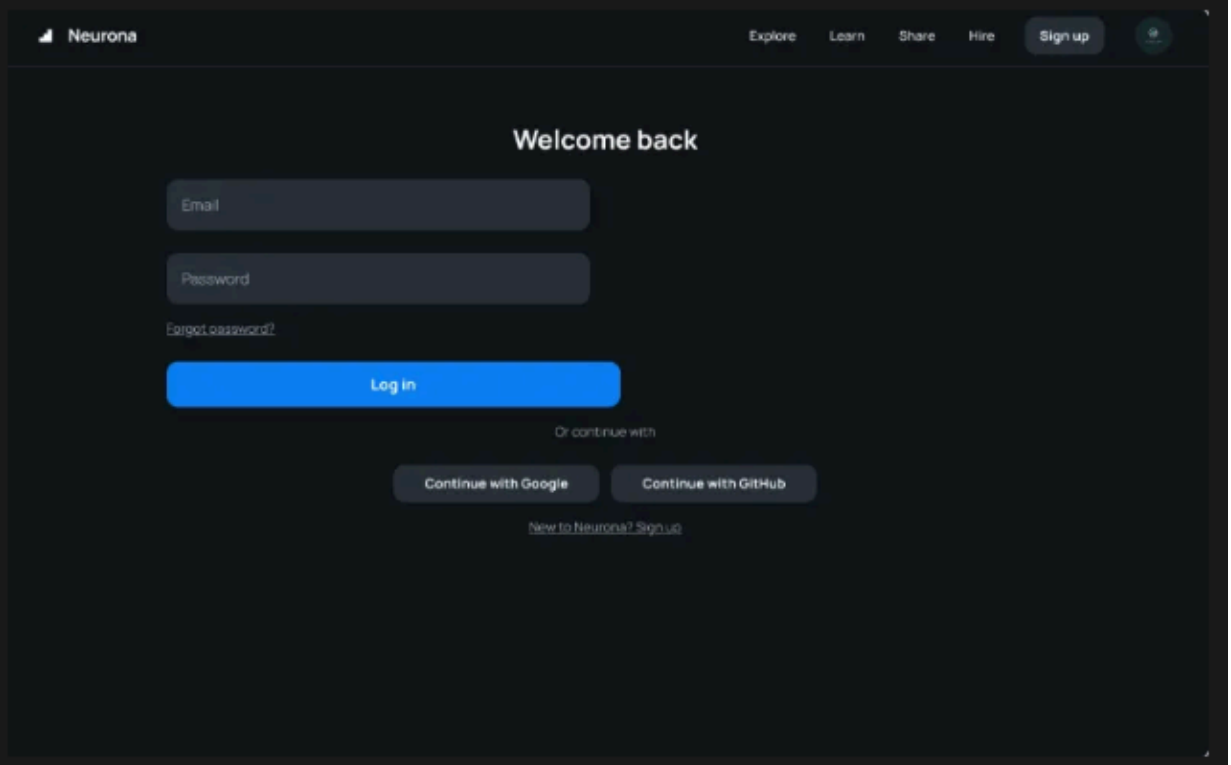
- User : avatar, name, email, password, role.
- Course : title, descriptions, chapters, imgURL, tags.
- Community : author, subject, content, status, comments.
- Veille : title, description, body_markdown, cover_image, tags, url.



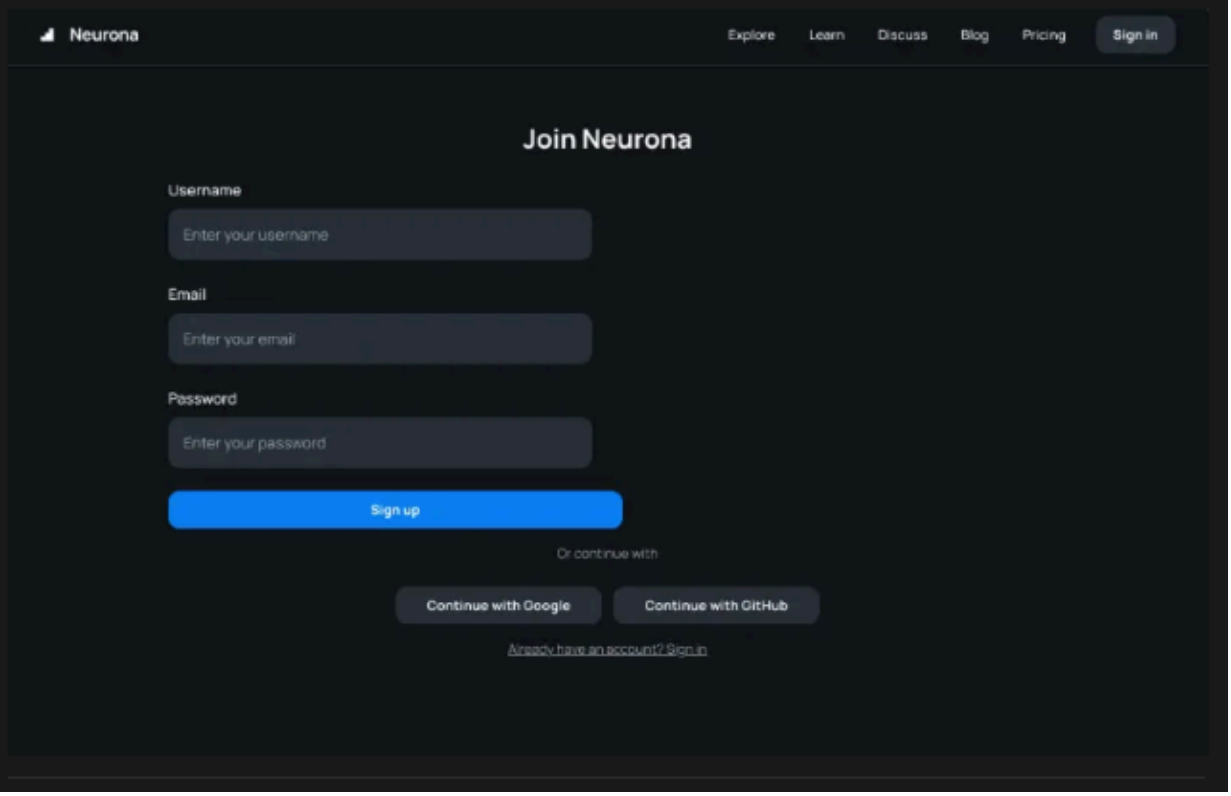
Maquettes / wireframes

Les maquettes ont été réalisées avant le développement pour valider l'UX/UI. L'accent a été mis sur la lisibilité qui est essentielle pour une plateforme de lecture longue.

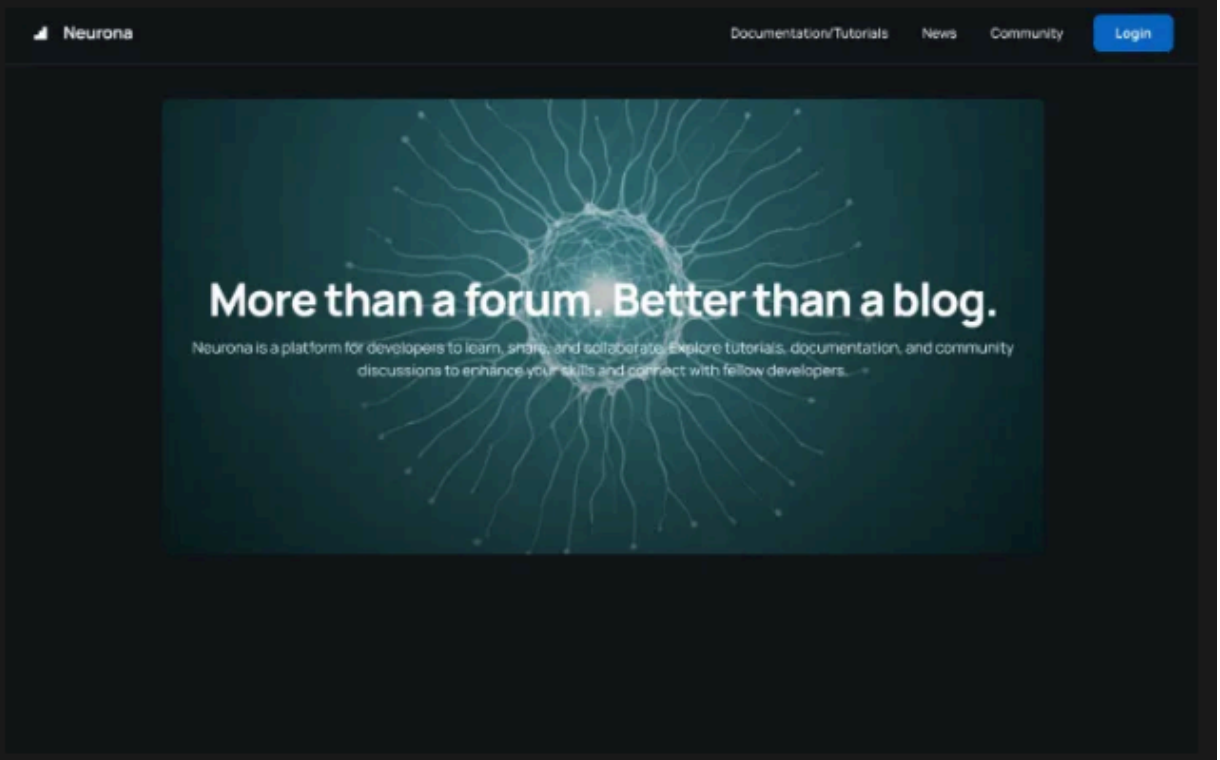
Connection



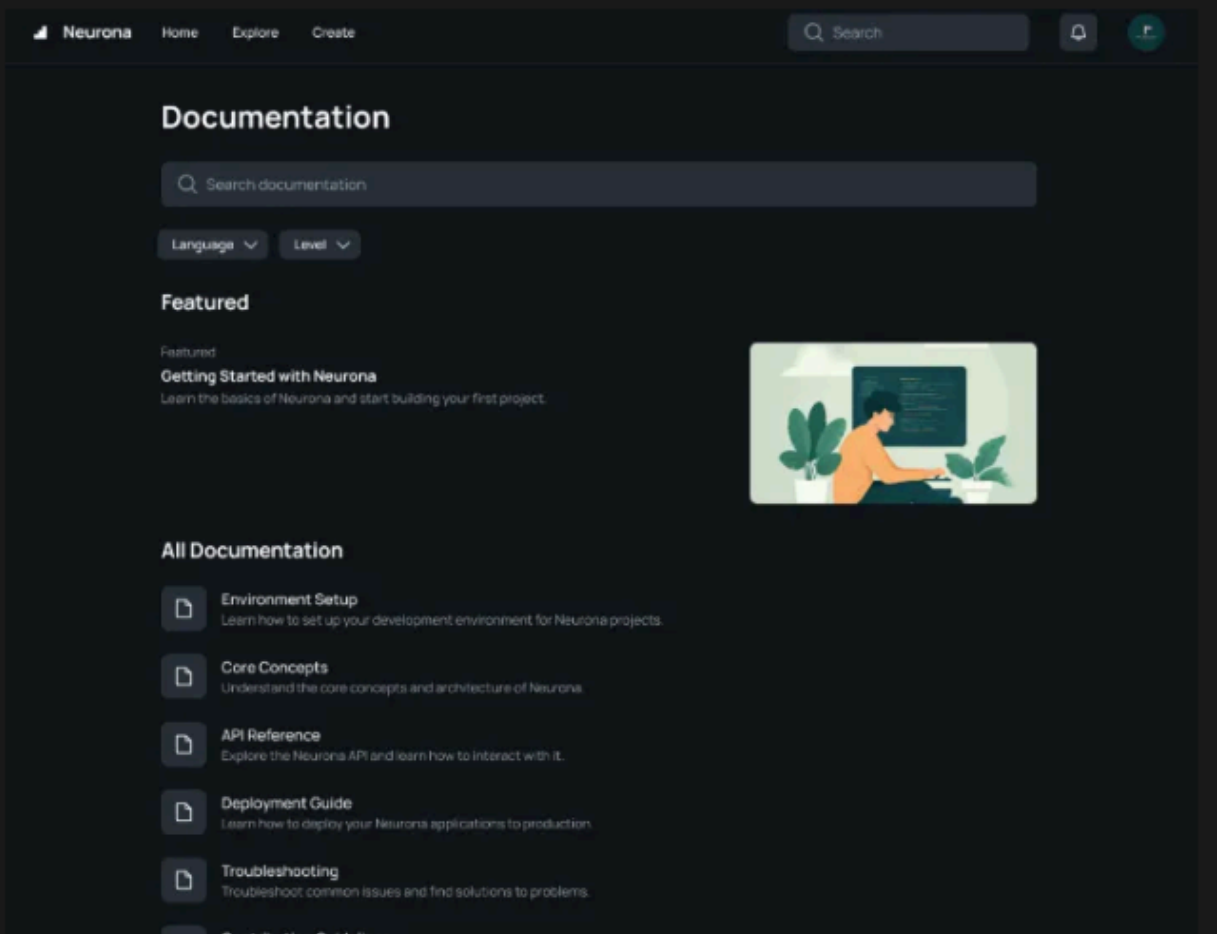
Inscription



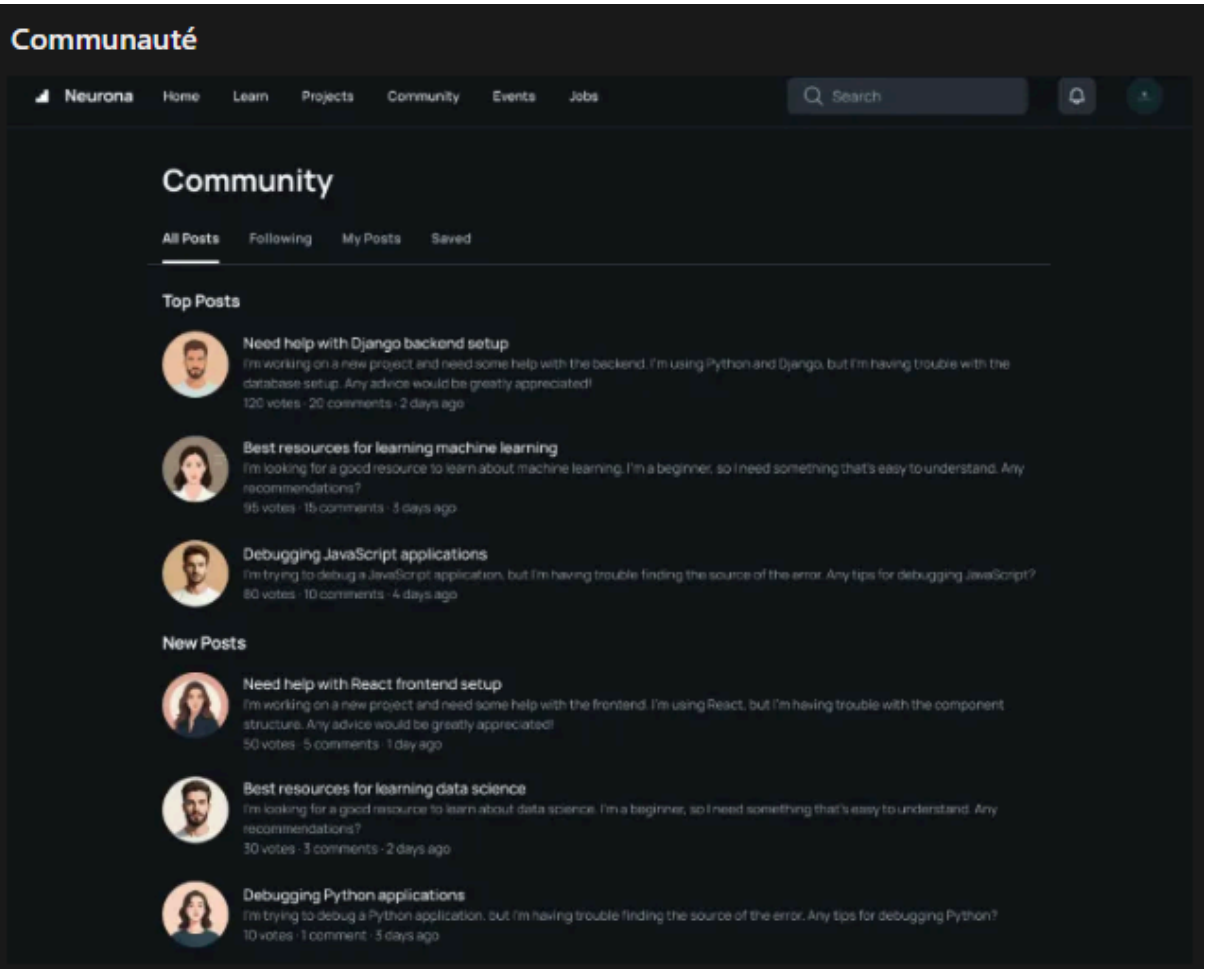
Page d'accueil



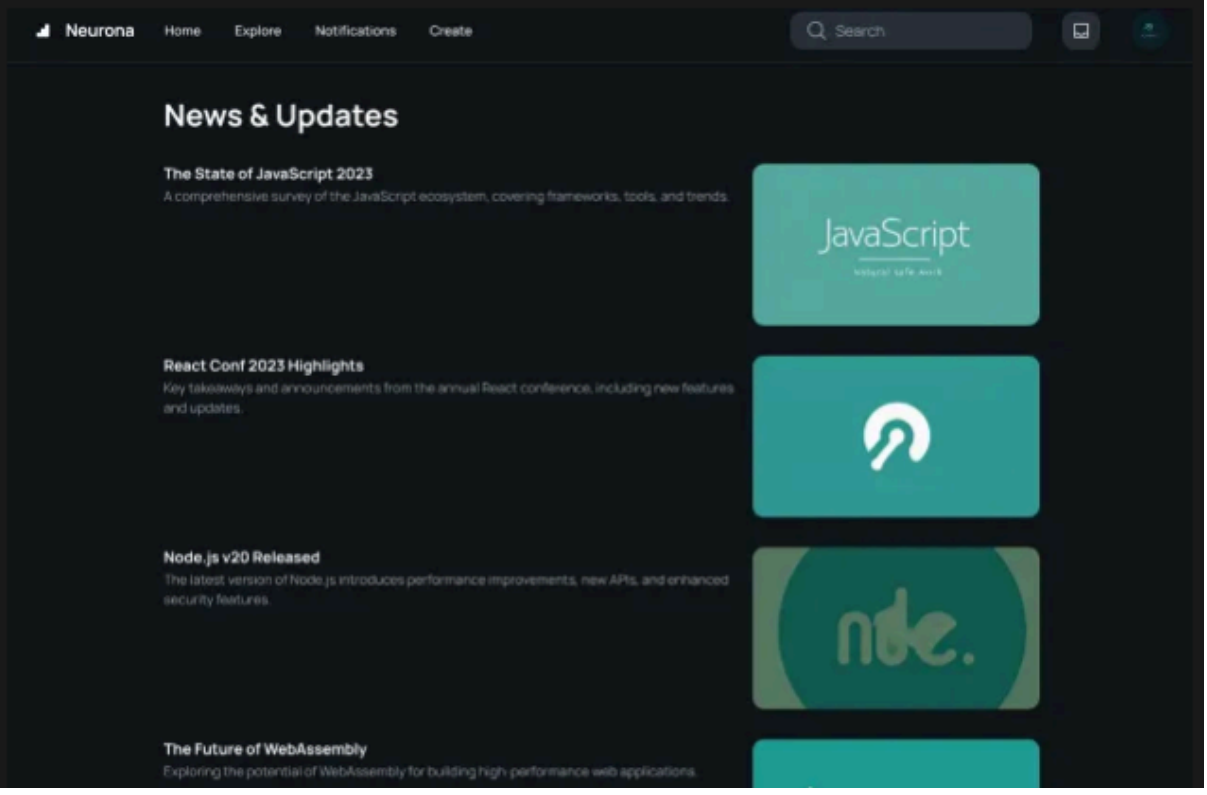
Recherche



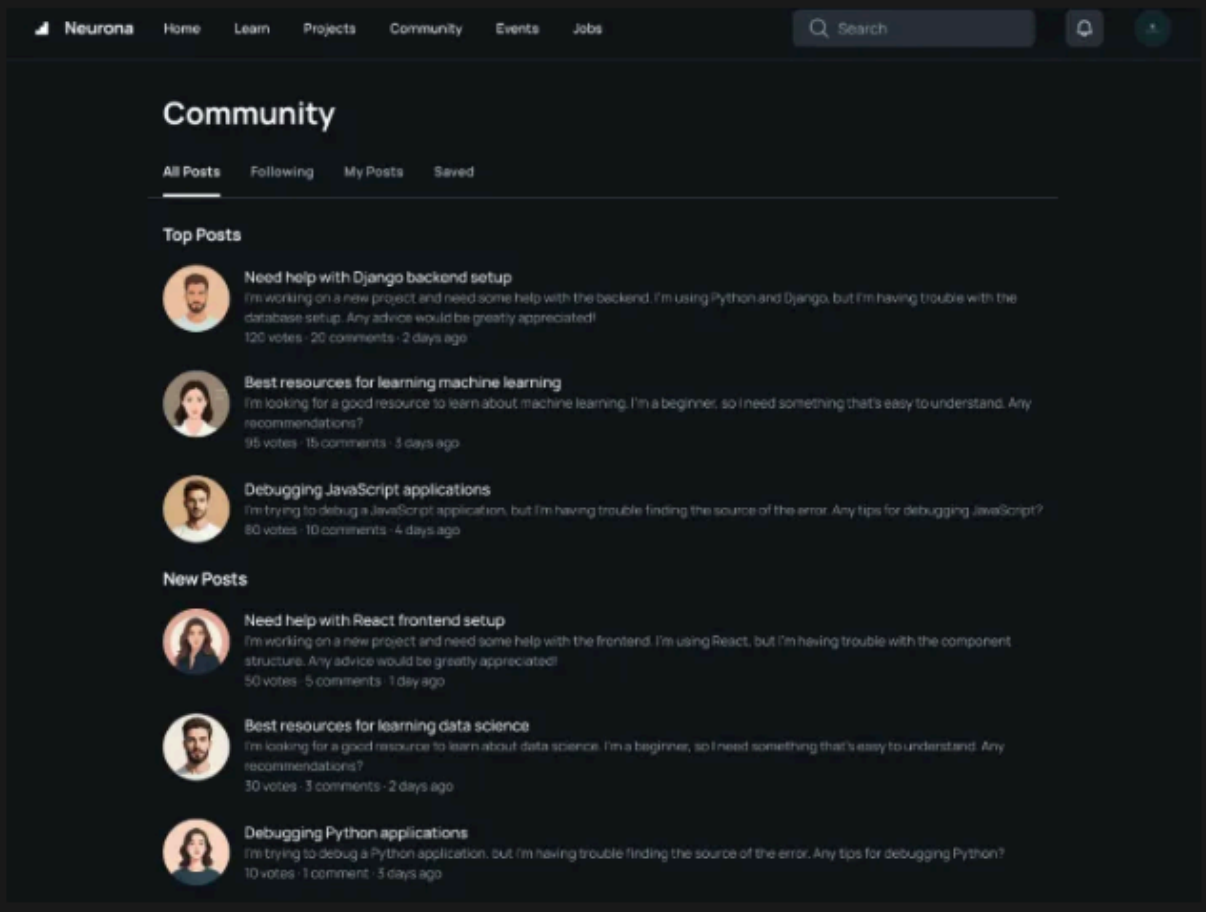
Communauté



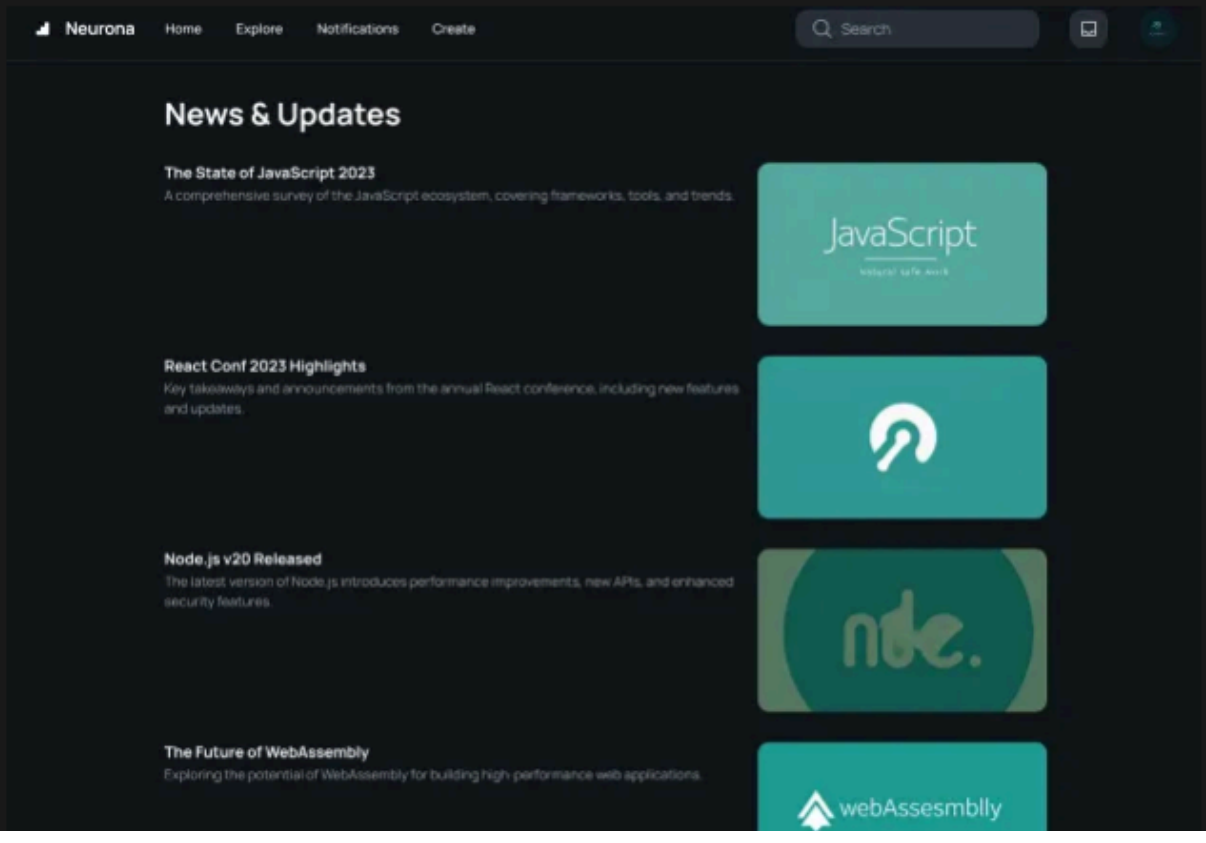
Documentation



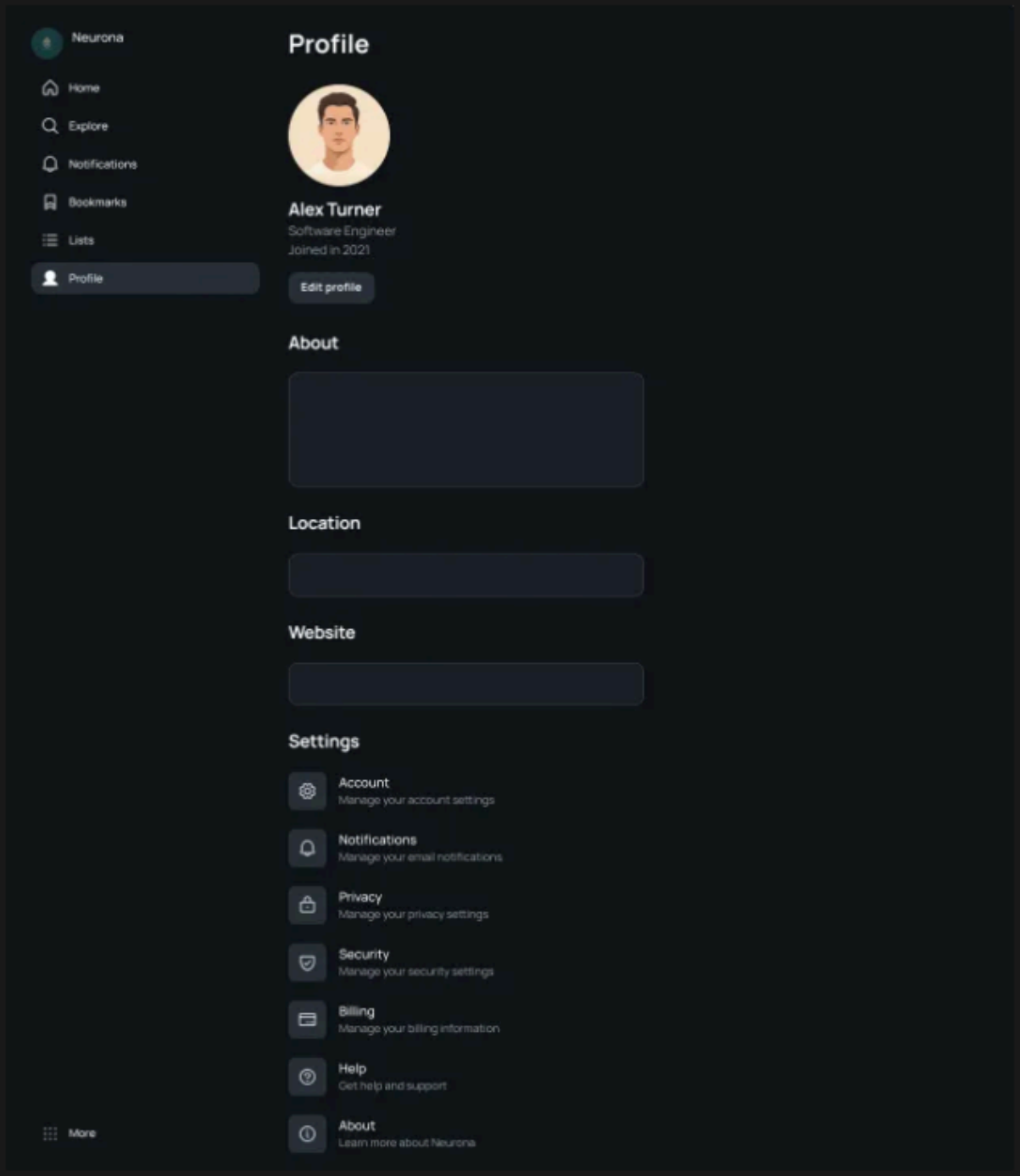
Communauté



Documentation / veille



Profil


Neurona
Home
Explore
Notifications
Bookmarks
Lists
Profile
More
Profile
Alex Turner
Software Engineer
Joined in 2021
Edit profile
About
Location
Website
Settings
Account
Manage your account settings
Notifications
Manage your email notifications
Privacy
Manage your privacy settings
Security
Manage your security settings
Billing
Manage your billing information
Help
Get help and support
About
Learn more about Neurona

Choix techniques et justification

Le projet a suivi une technique moderne et cohérente : MERN (MongoDB, Express, React, Node). Ce choix se justifie par l'utilisation du langage JavaScript (ou TypeScript) à la fois côté client et serveur, ce qui facilité le développement et la maintenance. MongoDB est idéal pour stocker des documents JSON flexibles comme ceux générés par l'IA. Tailwind est utilisé spécifiquement pour l'intégration des composants shadcn/ui, tandis que le scss a été retenu pour le styling personnalisé afin de garantir une structure propre et modulaire.

Développement

Fonctionnalité principale

- Authentification & Gestion de compte : Système sécurisé par Token JWT & hachage du mot de passe.
- Documentation : Générateur de documentation via Google Gemini.
- Communauté : CRUD (Create, Read, Update, Delete).
- Veille : Système d’affichage d'articles récupérés via api.

Exemple de code & API

```
const { GoogleGenAI } = require("@google/genai");
const { z } = require("zod");
const { zodToJsonSchema } = require("zod-to-json-schema");
const Course = require("../models/docModel");

// 1. Définition de la structure exacte que l'IA doit renvoyer (Validation Zod)
const docSchema = z.object({
 imageKeywords: z
   .string()
   .describe("Mots-clés pour trouver une image d'illustration"),
 title: z
   .string()
   .describe("Titre principal du cours"),
 // ... (autres champs définis comme difficulté, chapitres, etc.)
 chapters: z.array(z.object({
   title: z.string(),
   content: z.string().describe("Contenu éducatif du chapitre")
 })).describe("Liste des chapitres du cours")
});

exports.getDocumentation = async (req, res) => {
 try {
   const subject = req.body.subject; // Sujet demandé par l'utilisateur
   // 2. Création du prompt pour l'IA
   const prompt = `
   Génère une documentation technique complète sur le sujet suivant : "${subject}".
   Retourne un JSON valide respectant ce schéma.
   `;
   // 3. Connexion à l'IA (Google Gemini)
   const ai = new GoogleGenAI({ apiKey: process.env.GEMINI_API_KEY });
   // 4. Génération du contenu
   const request = await ai.models.generateContent({
     model: "gemini-2.0-flash-lite",
     contents: prompt,
     config: {
       responseMimeType: "application/json", // On force le format JSON
       responseJsonSchema: zodToJsonSchema(docSchema), // On impose notre structure
     },
   });

   // 5. Validation et nettoyage des données reçues
   const doc = docSchema.parse(JSON.parse(request.text));

   // 6. Récupération d'une image gratuite via l'API Pexels basée sur les mots-clés de l'IA
   const imgRes = await fetch(
     `https://api.pexels.com/v1/search?query=${doc.imageKeywords}&per_page=1`,
     { headers: { Authorization: process.env.PEXELS_API_KEY } }
   );
   const imgData = await imgRes.json();
   const imageUrl = imgData.photos?.[0]?.src?.large || "url_par_defaut.jpg";

   // 7. Sauvegarde du cours complet en base de données
   const course = await Course.create({ ...doc, imageUrl });
   // 8. Réponse au client (Frontend) avec le cours créé
   res.status(201).json(course);
 } catch (e) {
   res.status(500).json({ message: e.message });
 }
};
```

Sécurité

- Mots de passe : Hachage avec bcrypt avant stockage.
- Authentication : Token JWT pour sécuriser les routes.
- Protection HTTP : Helmet pour les headers de sécurité et Cors pour contrôler l'accès aux ressources.
- Validation : Zod pour valider les données entrantes.

Accessibilité & Responsive design

- Utilisation de balises sémantiques HTML5.
- Contraste des couleurs pour le confort de lecture.

Tests et Validation

Stratégie de tests

- Tests Unitaires : Vérification des fonctions isolées.
- Tests d'Intégration : Test des endpoints avec Thunder Client.
- Tests End-To-End : Test du parcours utilisateurs.

Exemples de cas

- Tentative d'inscription avec email existant : L'API doit retourner une erreur 401 "Email déjà utilisé".
- Accès à une route protégée sans token : Le middleware d'auth doit bloquer la requête et renvoyer une 401.
- Génération de doc avec prompt vide : Le contrôleur renvoie une 400.

Difficultés rencontrées

Problèmes techniques et organisationnels

- Organisationnel : Sous-estimation du temps nécessaire pour finaliser l'application.
- CORS : La gestion des cookies entre le front-end et le back-end à demandé une configuration spéciale.
- IA : Le format de réponse de l'IA n'était pas toujours un json valide ce qui faisait planter le front-end.

Résolutions des problèmes

- Organisationnel : Priorisation des fonctionnalités MVP (Minimum Viable Product) en reportant les fonctionnalités complexes.
- CORS : Configuration avec origin et credentials.
- IA : Utilisation de la librairie zod pour une schématisation du json avant de l'envoyer au front-end.

Bilan et perspective

Bilan

Bien que l'application ne soit pas entièrement finalisée, l'objectif pédagogique était avant tout de montrer ce dont je suis capable. La contrainte de travailler en totale autonomie, due à une situation personnelle complexe, m'a poussé à dépasser mes limites techniques. Ce qui m'a permis de consolider mes compétences sur la stack MERN et de gérer seul les imprévues d'un projet réel. Aujourd'hui, je suis capable de concevoir, développer, et déployer une application sécurisée en toute indépendance. Actuellement en stage sur un nouveau projet, de nouveau en autonomie, je continue de mettre à profit cette expérience pour m'améliorer de jour en jour.

Perspective

- Finaliser l'application : Terminer l'intégration des pages et les fonctionnalités.

Axes d'amélioration

- SEO : Améliorer le référencement naturel via du Server Side Rendering (SSR).

Evolutions possibles

- Premium : Intégration d'un mode premium permettant à l'utilisateur de générer ses propres documentations.

Annexes

- Lien application : https://neurona-xi.vercel.app/