

Présentation d'une création d'application web & mobile

The image displays a multi-device presentation of a web and mobile application for "AstroNex".

- Desktop View:** Shows a woman in a floral dress. Text on the page reads: "Magnétisme & Sophrologie Près De Lens ☀ Bien-Être Et Relaxation". Below this, it says: "Situé à Lila, près de Lens, je propose des séances de magnétisme et sophrologie pour vous accompagner vers un mieux-être global." There are buttons for "PRENDRE RDV", "MES SERVICES", and "CONTACTEZ-MOI".
- Smartphone View:** Shows a sidebar menu with categories like Accueil, À Propos, Services (Human Design, Magnétisme, Sophrologie), Articles, FAQ, and a "Votre" section. It includes buttons for "PRENDRE RDV", "MON PROFIL", and "CONTACTEZ-MOI".
- Tablet View:** Shows a "Choisir votre service" (Choose your service) page. It lists three services with descriptions:
 - Magnétisme:** Séances de magnétisme pour débloquer vos énergies.
 - Sophrologie:** Techniques de relaxation et de bien-être.
 - Human Design:** Découvrir votre design énergétique unique.
- QR Code:** An orange QR code is located on the right side of the image, which likely links to the website.

Sommaire

- Contexte & Objectifs
- Stack technique & Outils
- Conceptions
- Diagramme UML
- Architecture
- Front-end
- Composants réutilisables
- Page d'accueil
- Back-end
- Prisma (ORM)
- Base de données (PostgreSQL)
- API
- Authentification & Sécurité
- Tests (Postman)
- Déploiement (Vercel)
- Conclusion

Contexte & Objectifs



Digitaliser les services de bien-être



Prise de rendez-vous (présentiel / distanciel)



Moderniser l'expérience client



Améliorer visibilité (SEO)



Gestion articles (admin)



Gestion favoris (utilisateur)

Stack Technique & Outils utilisés

Next.js (Framework)

PostgreSQL (Neon Tech)

Prisma (ORM)

Auth : BcryptJs + JWT

Déploiement : Vercel

Visual Studio Code (Dev)

Git/GitHub (versionning & collab)

Postman (tests API)

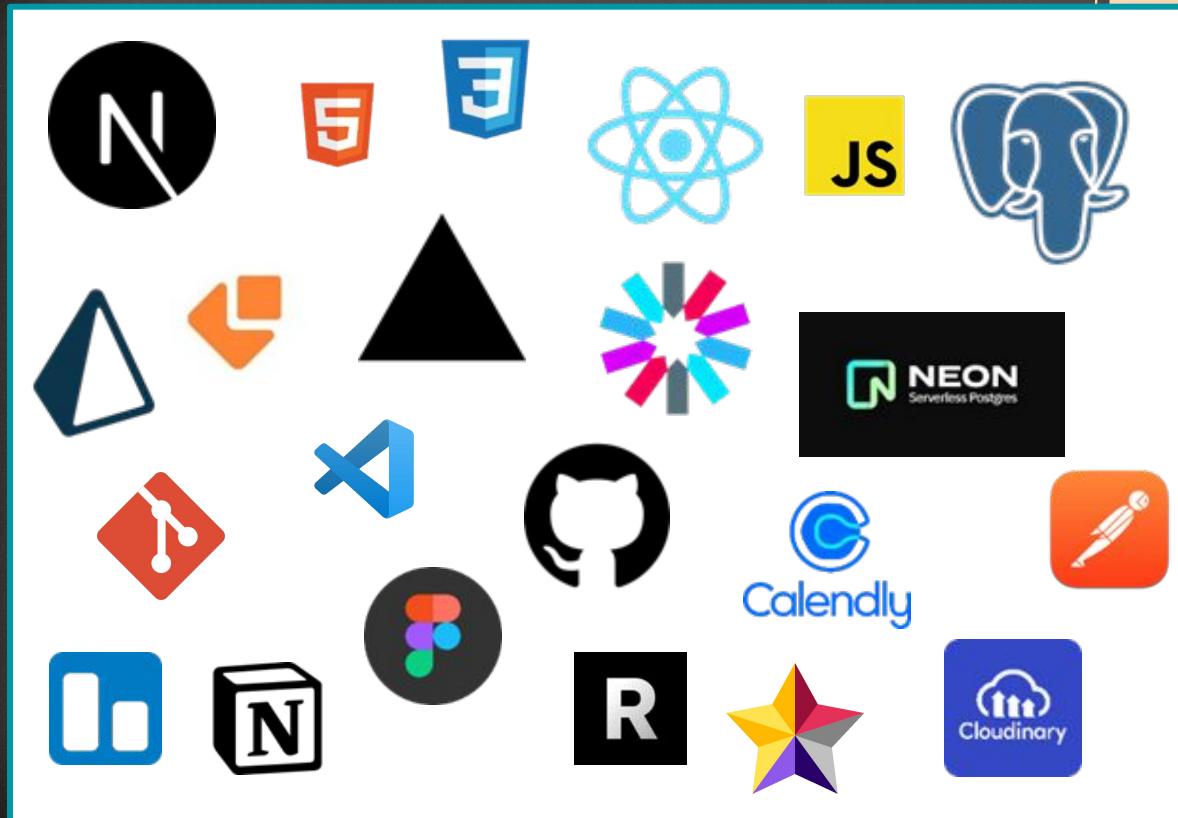
Conceptions :

Figma (maquettes) & StarUML

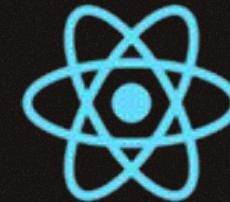
Services :

Calendly, Email.Js, Cloudinary

Trello & Notion (gestion tâches)



React.js



Bibliothèque JavaScript pour construire des interfaces utilisateur.

Principes clés :

- **Composants réutilisables** : Architecture modulaire
- **Virtual DOM** : Mises à jour optimisées
- **JSX** : Syntaxe HTML dans JavaScript
- **Hooks** : useState, useEffect pour gérer l'état et les effets

Next.js

Framework **React** créé par Vercel qui ajoute des fonctionnalités puissantes.

Avantages principaux :

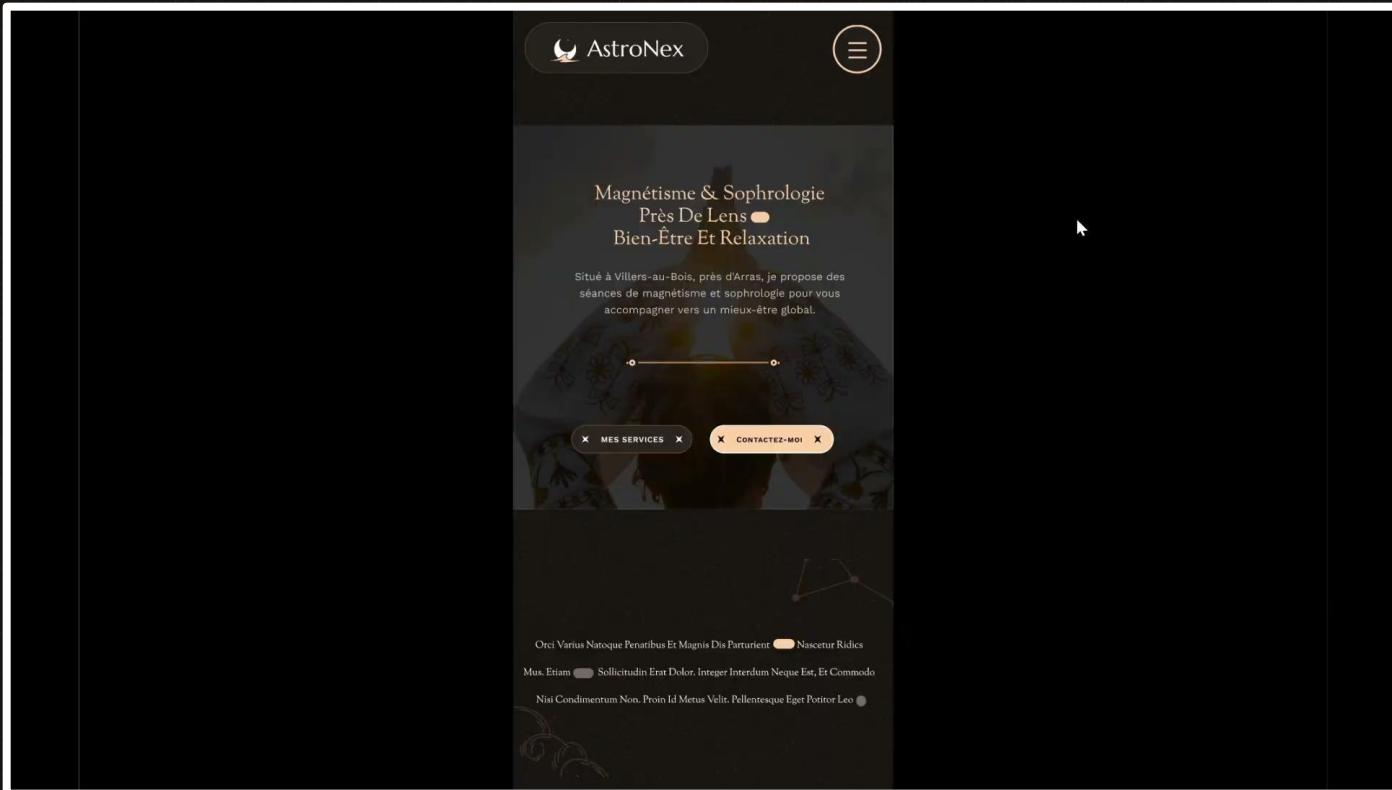
- **SSR (Server-Side Rendering)** : Rendu serveur ou génération statique pour performances et SEO
- **Routing automatique** : Basé sur la structure des fichiers
- **API Routes** : Backend intégré

API => Application Programming Interface , il sert d'intermédiaire entre deux programme

API Routes => inclut un **backend minimal** permettant de créer facilement des **points d'accès API** directement dans le projet

Routing automatique : Le **système de routes** est basé directement sur la **structure des fichiers** , sans configuration supplémentaire.

Conception | Maquette (Figma)



Conception | Charte Graphique

Magnétisme & Sophrologie Près De Lens Bien-Être Et Relaxation

AstroNex

BOUTON PRINCIPAL

BOUTON SECONDAIRE

Sorts Mill Goudy

Work Sans

Aa

Aa

abcdefghijklmnopqrstuvwxyz
1234567890

abcdefghijklmnopqrstuvwxyz
1234567890

AstroNex

The screenshot shows the AstroNex website's home page. It features a dark background with a subtle texture. At the top, there's a header with text and a small circular icon. Below the header is a row of five colored circles in shades of orange, brown, and grey. The main title "AstroNex" is displayed in a large, bold, serif font. Below the title are two buttons labeled "BOUTON PRINCIPAL" and "BOUTON SECONDAIRE". The text "Sorts Mill Goudy" and "Work Sans" are listed below the respective fonts. Two large letters "Aa" are shown, each with a different font style. At the bottom, there are two rows of lowercase letters and numbers, followed by a small image of a person in a brown robe sitting at a desk.

ACCESIBILITÉ RGAA 4.1

Contraste : Ratio minimum 4.5:1 (AA) - Beige/Noir = 7:1 (AAA ✓)

Navigation clavier : Focus visible, ordre de tabulation logique

Tailles : Corps 16px min, titres hiérarchisés (H1>H2>H3)

Images : Textes alternatifs descriptifs, icônes avec aria-label

Structure : Balises sémantiques HTML5, formulaires avec labels

7:1

Beige/Noir - AAA

7:1

Noir/Beige - AAA

PSYCHOLOGIE DES COULEURS

Beige (#E6B88C) : Chaleur, réconfort, élégance - parfait pour le bien-être

Marron foncé (#2C2520) : Stabilité, ancrage, nature - évoque la sérénité

Marron moyen (#8B6F47) : Authenticité, terre, connexion naturelle

Cette palette véhicule des valeurs de calme, de professionnalisme et d'harmonie, essentielles pour une activité de bien-être et de développement personnel.

Responsive



Cahier des charges | Notion

Cahier des charges

Cahier des Charges — Application Web Bien-être & Formations

Objectif général

Développer une application web comportant :

1. Un site vitrine présentant les services de bien-être : **magnétisme, sophrologie, human design**.
2. Une prise de rendez-vous via Calendly, avec choix du mode (présentiel/distanciel).
3. Un espace de formation en ligne, avec accès sécurisé pour les participants.
4. Une interface d'administration pour gérer les utilisateurs, articles, et contenu de formation.

Structure du site

1. Site vitrine (public)

Pages à développer :

- Accueil
 - Présentation générale, appel à l'action, description des services, témoignages, mes articles.
- Qui suis-je ?
 - Biographie, parcours, vision.
- Services
 - Une page par service :
 - Magnétisme
 - Sophrologie
 - Human Design
- Articles

- Liste paginée ou filtrable d'articles.
- Chaque article a sa **page dédiée** (route dynamique).

FAQ

- Liste de questions/réponses fréquentes.

Contact

- Formulaire de contact.
- Bouton "Prendre rdv" :
 - Choix entre **distanciel** ou **présentiel**.
 - Redirection vers le bon lien **Calendly**.

— 2. Espace formation en ligne (privé après paiement)

Fonctionnalités :

- Achat de la formation via **Stripe**.
- Après paiement :
 - Création automatique d'un compte utilisateur.
 - Envoi d'un **email** via **Resend** contenant les identifiants.
- Espace personnel :
 - Connexion avec **BcryptJs + JsonWebToken**
 - Modification du **mot de passe** avec **Resend** et du **nom d'utilisateur**.
 - Affichage des **chapitres de formation** :
 - Menu vertical à gauche (liste des chapitres),
 - Affichage du contenu à droite :
 - Texte
 - Vidéos
 - Audios
 - Fichiers téléchargeables

3. Espace d'administration (privé)

Accessible uniquement à l'administrateur du site.

Fonctionnalités :

- **Gestion des Utilisateurs** (CRUD)
- **Gestion des chapitres de formation** (CRUD)
 - Titre
 - Contenu : **texte, audio, vidéo, fichiers**.
- **Gestion des articles** (CRUD)

Stack technique

Élément	Technologie
Framework	Next.js
Base de données	Neon (PostgreSQL)
ORM	Prisma
Authentification	BcryptJs + JsonWebToken
Paiement	Stripe
Emails/Resend	Email.js (coté client) / Resend (coté serveur)
Icons	React Icons
Formulaires	React Hook Form + Zod
Déploiement	Vercel

Contraintes & exigences techniques

- **Site responsive** (desktop / mobile)
 - **SEO friendly** (balises meta, titres, performances)
 - Sécurisation des pages privées.
- Utilisation de **Stripe Webhooks** pour débloquer l'accès formation
- Expérience utilisateur fluide, interface apaisante, moderne et lisible

Diagramme de classe - UML

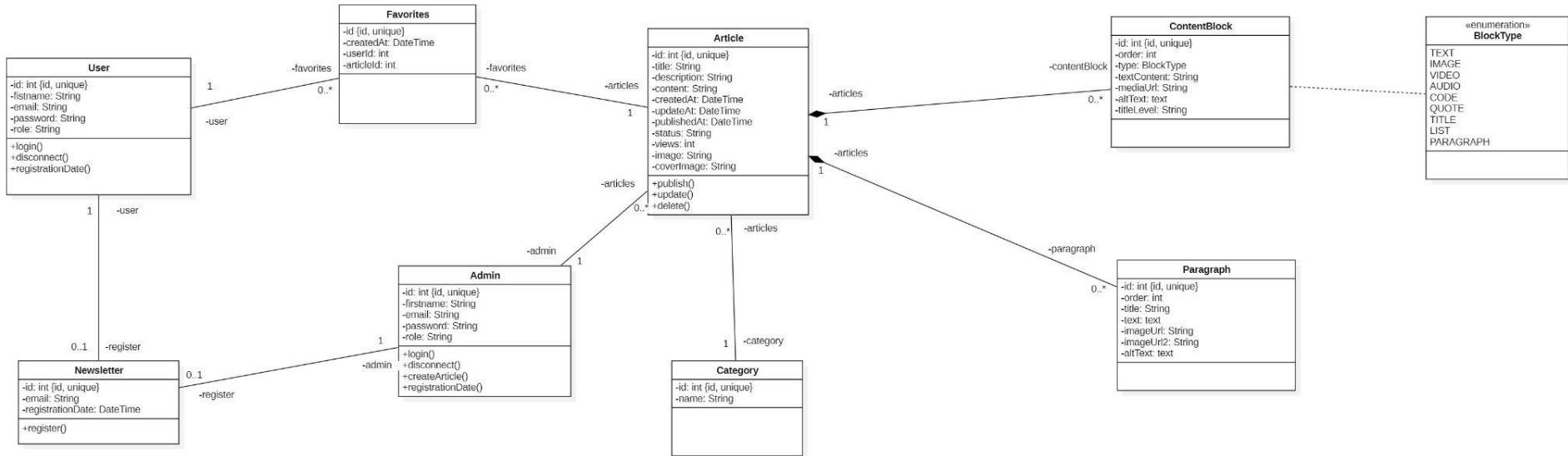


Schéma de base de données PhyMyAdmin

2. Relation Article ↔ ContentBlock (1:N)

Article (1) ----< ContentBlock (N)

- Clé étrangère : `articleId` dans la table `content_blocks`
- Référence : `id` dans la table `articles`
- CASCADE : Si un article est supprimé, tous ses blocs de contenu sont supprimés
- Signification : Un article peut avoir plusieurs blocs de contenu

3. Relation Article ↔ Paragraph (1:N)

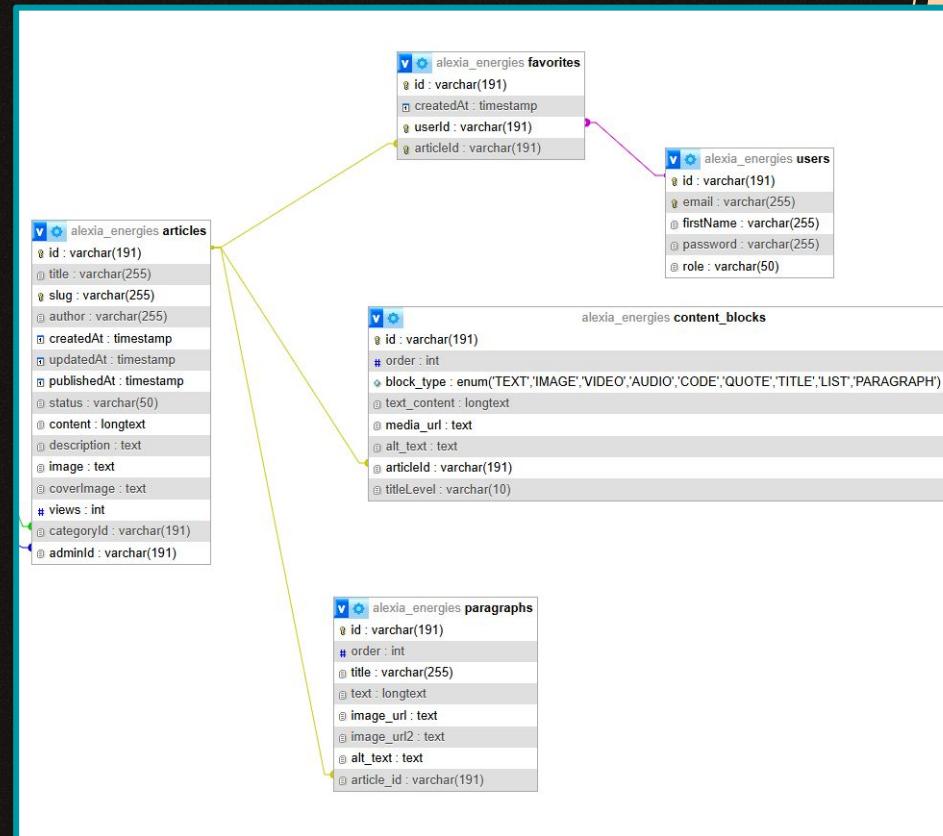
Article (1) ----< Paragraph (N)

- Clé étrangère : `articleId` dans la table `paragraphs`
- Référence : `id` dans la table `articles`
- Signification : Un article peut avoir plusieurs paragraphes

4. Relation User ↔ Favorite ↔ Article (N:M - Plusieurs vers Plusieurs)

User (N) ----< Favorite >---- Article (N)

- Table de liaison : `favorites`
- Clés étrangères :
 - `userId` → référence `users.id`
 - `articleId` → référence `articles.id`
- CASCADE : Si un utilisateur ou un article est supprimé, les favoris associés sont supprimés
- Contrainte unique : `@@unique([userId, articleId])` empêche les doublons



main 2 Branches 0 Tags

Go to file

t

Add file

Code

fannysaez Merge branch 'fanny' ✘

b455c92 · 3 months ago

410 Commits

prisma

Maj(fichiers): modified: prisma/schema.prisma

3 months ago

public

Layout(Ajout): Ajout img OpenGraph Astronex

3 months ago

src

Layout(Ajout): Ajout img OpenGraph Astronex

3 months ago

.gitignore

CONTRÔLE DE CODE SOURCE

MODIFICATIONS	
GRAPHIQUE	
• Maj(Responsive,Pagination): Ajout Pagination admin/dashboard car formulaire trop long, pagination 3cards/page pour(Modifier et ...)	Automatique
• Merge branch 'fanny' saez.fanny.63@gmail.com	4 months ago
• Page et Style(Contact): Responsive mobile pour la page contact saez.fanny.63@gmail.com	4 months ago
• Merge branch 'fanny' saez.fanny.63@gmail.com	4 months ago
• Page(contact): amélioration saez.fanny.63@gmail.com	4 months ago
• Merge branch 'fanny' saez.fanny.63@gmail.com	4 months ago
• Maj+tests(responsive partout): Responsive sur le header, dashboard, contact, login, articles etc... tests, image couverture, principale...	dm... 3 months ago
• Merge branch 'fanny' saez.fanny.63@gmail.com	dm... 3 months ago
• Finitions(Pages, style des route.js) : modified: src/app/admin/articles/ArticleForm.jsx saez.fanny.63@gmail.com	dm... 3 months ago
• Merge branch 'fanny' saez.fanny.63@gmail.com	dm... 3 months ago
• CRUD(Articles(slug)): Le CRUD des articles fonctionne correctement, rajouter dans les blocksType, reste a rajouter le textera pour a...	dm... 3 months ago
• Merge branch 'fanny' saez.fanny.63@gmail.com	dm... 3 months ago
• Maj(fichiers): modified: prisma/schema.prisma saez.fanny.63@gmail.com	dm... 3 months ago
• Merge branch 'fanny' saez.fanny.63@gmail.com	dm... 3 months ago
• Ajustements aux complets(fichiers): ajuster modified: prisma/schema.prisma saez.fanny.63@gmail.com	dm... 3 months ago
• Merge branch 'fanny' saez.fanny.63@gmail.com	dm... 3 months ago
• Header(Page): Amélioration du boutton connexion/mon profil saez.fanny.63@gmail.com	dm... 3 months ago
• Merge branch 'fanny' saez.fanny.63@gmail.com	dm... 3 months ago
• Maj(admin articles form, articles, categories): Ajout d'une nouvelle catégorie via admin dashboard pour filtrer par catégorie ma pa...	dm... 3 months ago
• Merge branch 'fanny' saez.fanny.63@gmail.com	dm... 3 months ago
• Articles(slug)(Page): Ajustements de la page routing dynamique saez.fanny.63@gmail.com	dm... 3 months ago
• Merge branch 'fanny' saez.fanny.63@gmail.com	dm... 3 months ago
• Page(Articles(slug)): Ajustements de la page puis dans admin/dashboard, afin d'ajouter, modifier et supprimer ! reste une autre par...	dm... 3 months ago
• Merge branch 'fanny' saez.fanny.63@gmail.com	dm... 3 months ago
• Page(Articles(slug)): Ajustements de la page puis dans admin/dashboard, afin d'ajouter, modifier et supprimer ! reste une autre par...	dm... 3 months ago
• Merge branch 'fanny' saez.fanny.63@gmail.com	dm... 3 months ago
• Page(Articles(slug)): Ajustements de la page puis dans admin/dashboard, afin d'ajouter, modifier et supprimer ! reste une autre par...	dm... 3 months ago
• Merge branch 'fanny' saez.fanny.63@gmail.com	dm... 3 months ago
• CRUD(Articles): Gestion des articles dans admin dashboard, à vérifier tt de même saez.fanny.63@gmail.com	dm... 3 months ago

GitHub | Plateforme

Base du projet dès le départ (3 mois) de travail

alexia-energies Public

main 2 Branches 0 Tags

Go to file t Add file Code

fannysaez initial commit ✓ 7cf7382 · 2 weeks ago 190 Commits

prisma initial commit last month

public initial commit 3 weeks ago

src initial commit 2 weeks ago

.gitignore initial commit 3 months ago

README.md initial commit last month

cahier-des-charges.md initial commit 2 months ago

eslint.config.mjs initial commit 3 months ago

jsconfig.json initial commit 3 months ago

next.config.mjs initial commit 2 months ago

package-lock.json initial commit 2 months ago

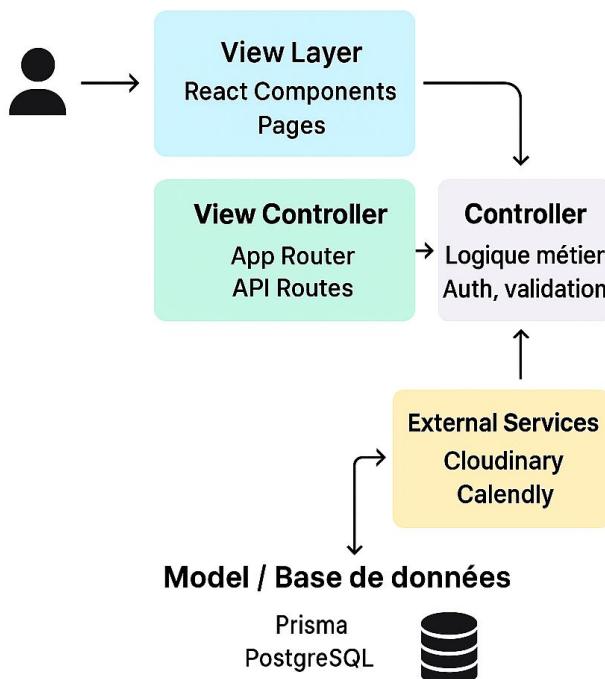
package.json initial commit 2 months ago

12 / 30

Git (versionning)

Architecture MVVC

(Model View View Controller) de Next.js



👤 **Utilisateur** → utilise l'interface.

🎨 **View Layer** → affiche les données.

🧭 **View Controller** → gère la navigation.

💼 **Controller** → applique la logique métier.

🗄️ **Model (Schéma prisma - ORM)** → définit la structure et envoie les données.

⌚ **PostgreSQL** → stocke les données.

☁️ **Services externes** → ajoutent images, mails, rendez-vous.

MVVC = Model – View – View Controller – Controller.

C'est une architecture pensée pour Next.js, qui a besoin d'une couche spéciale (le **View Controller**) parce que son **App Router** et ses **API intégrées** jouent un rôle unique entre la Vue et le Controller.

Front-end

1. Importation des composants modulaire (réutilisables)

```
19 import HeroSection from "@/app/components/accueil/hero/section1";
20 import About from "@/app/components/accueil/about/section2";
21 import Marque from "@/app/components/marque/marque";
22 import Process from "@/app/components/accueil/process/section3";
23 import Testimonials from "@/app/components/testimonials/testimonials";
24 import MesArticles from "@/app/components/accueil/mesArticles/mesArticles";
```

2. Structure du fichier page.js / App Router

L'**App Router** est la nouvelle architecture de Next.js (depuis la v13) basée sur les **React Server Components** et une structure de fichiers dans le dossier **app/**. à **réutiliser les composants** et à **améliorer les performances** grâce au rendu côté serveur. Dans ce projet, il permet une **navigation fluide** et une **organisation claire du code** .

```
26 export default function Home() {
27   return (
28     <>
29       <HeroSection />
30       <About />
31       <Process />
32       <Marque />
33       <Testimonials />
34       <MesArticles />
35     </>
36   );
37 }
```

Composants réutilisables

3. Structure du composant réutilisable avec des props [button.jsx](#) et le [button.module.css](#)

C'est quoi ?

Un composant réutilisable est un bloc d'UI autonome (structure, style, comportement) configurable via des props, ex. Button + CSS Modules.

À quoi ça sert ?

À factoriser et uniformiser l'interface en utilisant le même élément partout, plus vite et avec une meilleure accessibilité/testabilité.

Pourquoi ?

Pour une meilleure maintenabilité et cohérence: un changement unique se propage, on évite la duplication (DRY) et on gagne en productivité.

```
/* button.module.css */  
.primary {  
  background-color: var(--secondary-color);  
  color: var(--tertiary-color);  
  border-radius: 30px;  
}  
  
.secondary {  
  background-color: transparent;  
  color: var(--primary-color);  
  border: 1px solid var(--primary-color);  
}
```

Système de stylisation
modulaire (code CSS
maintenable)

```
export default function Button({  
  text,  
  link,  
  onClick,  
  variant = "primary",  
  className = "",  
  leftVector,  
  rightVector,  
  type = "button",  
  isReserveButton = false,  
  ariaLabel  
}) {  
  You, il y a 3 sema
```

15 / 30

Props = paramètres passés à un composant React. Elles le rendent **flexible** et **réutilisable** (apparence ou comportement modifiable sans réécrire le code).

Page d'accueil

```
export default function HeroSection() {
  return (
    <section className={style.section}>
      <div className={style.container}>
        <h1>Magnétisme & Sophrologie Près De Lens
          <span className={style.line}></span>
        Bien-Être Et Relaxation
        </h1>
        <p>Situé à Lilas, près de Lens, je propose...</p>

        <Button
          text="Découvrir mes services"
          link="/services"
          variant="primary"
          rightVector={<Image src={StarWhite} />}
        />
      </div>
    </section>
  );
}

:root {
  --primary-color: #FED1A7;
  --secondary-color: #2C2520;
  --background: #181411;
  --text: #CAC3BC;
}
```

```
src > app > components > accueil > hero > style.module.css > ...
saez.fanny.63@gmail.com, il y a 2 mois | 1 author (saez.fanny.63@gmail.com)
1  /* Page accueil | Section 1 */
2  .section {
3    position: relative;
4    /* padding: 200px 0px;
5    padding-top: 250px; */
6    width: 100%;
7    margin: auto;
8    background-image: url('/img/accueil/HeroSection/section1-accueil.webp');
9    background-size: cover;
10   background-position: center;
11   display: flex;
12   align-items: center;
13   justify-content: center;
14   overflow: hidden;
15   min-height: 100vh;
16 }
17
18 .overlay {
19   position: absolute;
20   top: 0;
21   left: 0;
22   width: 100%;
23   height: 100%;
24   background-color: rgba(0, 0, 0, 0.5);
25   /* overlay noir transparent */
26   z-index: 1;
27   /* sous le contenu */
28   background-color: rgba(0, 0, 0, 0.5);
29   /* overlay noir transparent */
30   z-index: 1;
31   /* sous le contenu */
32 }

/* Mobiles (320px - 480px) */
33 @media (max-width: 480px) {
34   .section {
35     padding: 100px 10px;
36     margin-top: 40px;
37   }
38
39   .container {
40     max-width: 95%;
41     padding: 5px;
42   }
43
44   .container h1 {
45     font-size: 28px;
46     margin-bottom: 15px;
47     line-height: 1.0;
48   }
49
50   .container p {
51     margin-bottom: 30px;
52     line-height: 24px;
53   }
54 }
```

Page d'accueil

Magnétisme & Sophrologie Près De Lens Bien-Être Et Relaxation

Situé à Lilas, près de Lens, je propose des séances de magnétisme et sophrologie pour vous accompagner vers un mieux-être global.

 MES SERVICES 

 CONTACTEZ-MOI 

Back-end

```
src/
  └── lib/
    ├── prisma.js      # 🌐 Connexion base de données
    ├── auth.js         # 🔒 Vérification utilisateurs
    └── emailTemplates.js # 📧 Templates d'emails

  └── app/
    └── api/
      ├── articles/      # 📄 CRUD Articles publics
      ├── admin/articles/ # 🏛️ CRUD Articles admin
      ├── favorites/     # ❤️ Système de favoris
      ├── auth/           # 🔑 Authentification
      ├── login/          # 🚀 Connexion
      ├── register/       # 📝 Inscription
      └── middleware/     # 🛡️ Protection des routes
```

endpoints: Un endpoint ou point de terminaison est une **URL spécifique** de mon API qui répond à des **requêtes HTTP** pour effectuer une action précise.

J'ai organisé mon code de manière modulaire avec **Next.js/App Router**.

Chaque dossier à une responsabilité précise **lib/** pour les utilitaires, **api/** pour les endpoints et **middleware/** pour la sécurité.

middleware: Un **middleware** contrôle l'accès: il vérifie l'authentification, applique les rôles, rejette les jetons invalides/expirés, redirige vers /login pour les pages et renvoie 401/403 pour les API.

Prisma (ORM)

```
model Favorite {
    id      String @id @default(cuid())
    createdAt DateTime @default(now())
    userId   String
    articleId String

    user   User   @relation(fields: [userId], references: [id], onDelete: Cascade)
    article Article @relation(fields: [articleId], references: [id], onDelete: Cascade)

    @@unique([userId, articleId])
    @@map("favorites")
}
```

L'annotation `@@unique([userId, articleId])` permet d'éviter les doublons : un même utilisateur ne peut pas aimer deux fois le même article.

L'annotation `@@map("favorites")` indique le nom réel de la table dans la base de données, ici au pluriel. Le modèle Prisma, lui, reste au singulier : *Favorite*.

L'option `onDelete: Cascade` signifie que si un utilisateur est supprimé, tous ses favoris le seront automatiquement.

1. Qu'est-ce que c'est ?

Prisma ORM est un outil qui fait le pont entre mon code JavaScript et ma base de données. et évite d'écrire du SQL manuellement.

2. À quoi ça sert ?

Prisma permet de gérer les données sans SQL et simplifie les opérations CRUD, le schéma, le client TypeScript et les migrations.

3 Pourquoi ?

Prisma simplifie le développement, limite les erreurs et s'adapte à plusieurs bases de données.

Prisma, qui définit la structure d'une table de favoris dans la base de données. C'est une relation many-to-many entre utilisateurs et articles.

Prisma (ORM) | Exemples

Récupérer les Favoris d'un Utilisateur

SQL Traditionnel (DANGEREUX - Injection possible)

```
```sql
-- Si quelqu'un envoie slug = ''; DROP TABLE articles; --
const query = "SELECT * FROM articles WHERE slug = '" + slug + "'";
-- Résultat : SELECT * FROM articles WHERE slug = ''; DROP TABLE articles; --
````
```

103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120

Code Prisma (SÉCURISÉ automatiquement)

```
```javascript
// Dans votre /api/favorites/route.js
const user = await prisma.user.findUnique({
 where: { email: userEmail },
 include: {
 favorites: {
 include: {
 article: true,
 }
 },
 },
});
````
```

```
const articlesFavoris = user.favorites.map(fav => fav.article);
````
```

### POURQUOI PRISMA ?

- 1.Sécurité automatique (zéro injection SQL)
- 2.Code lisible et maintenable
- 3.Relations simplifiées
- 4.Type Safety (détection d'erreurs)

### POURQUOI l'injection SQL est IMPOSSIBLE avec Prisma ?

### ### SQL Sécurisé avec Requêtes Préparées

```
```sql
-- Requête complexe avec plusieurs JOINs
SELECT
    a.id, a.title, a.slug, a.description, a.image,
    c.name AS category_name,
    f.createdAt AS favorite_date
FROM articles a
INNER JOIN favorites f ON a.id = f.articleId
INNER JOIN users u ON f.userId = u.id
LEFT JOIN category c ON a.categoryId = c.id
WHERE u.email = 'fanny.saez.0486@gmail.com'
    AND a.status = 'published'
ORDER BY f.createdAt DESC;
````
```

Prisma utilise des requêtes préparées automatiques qui isolent complètement les données utilisateur du code SQL, rendant l'injection impossible.

# PostgreSQL (Neon Tech)

```
COLUMNS Add column
id TEXT PRIMARY KEY
createdAt TIMESTAMP NOT NULL DEFAULT 'CURRENT_TIMESTAMP'
userId TEXT NOT NULL
articleId TEXT NOT NULL

CONSTRAINTS Add constraint
CONSTRAINT favorites_articleId_fkey FOREIGN KEY (articleId) REFERENCES public.articles (id) ON UPDATE CASCADE ON DELETE CASCADE
CONSTRAINT favorites_userId_fkey FOREIGN KEY (userId) REFERENCES public.users (id) ON UPDATE CASCADE ON DELETE CASCADE
CONSTRAINT favorites_pkey PRIMARY KEY (id)

INDEXES Add index
UNIQUE INDEX favorites_pkey ... USING BTREE (id)
UNIQUE INDEX favorites_userId_articleId_key ... USING BTREE (userId, articleId)
```

**Neon Tech** = une version **cloud et serverless** de **PostgreSQL**, qui se scale automatiquement et permet de créer des **branches de base** comme avec Git.

## 1. Qu'est-ce que c'est ?

**PostgreSQL** est une base de données relationnelle **open-source, performante et fiable**.

## 2. À quoi ça sert ?

Assurer la cohérence des données, gérer les accès simultanés et optimiser les requêtes.

## 3. Pourquoi ?

Pour sa fiabilité, ses relations Many-to-Many natives et son automatisation (cascades, scalabilité).

# API

## 1. Qu'est-ce que c'est ?

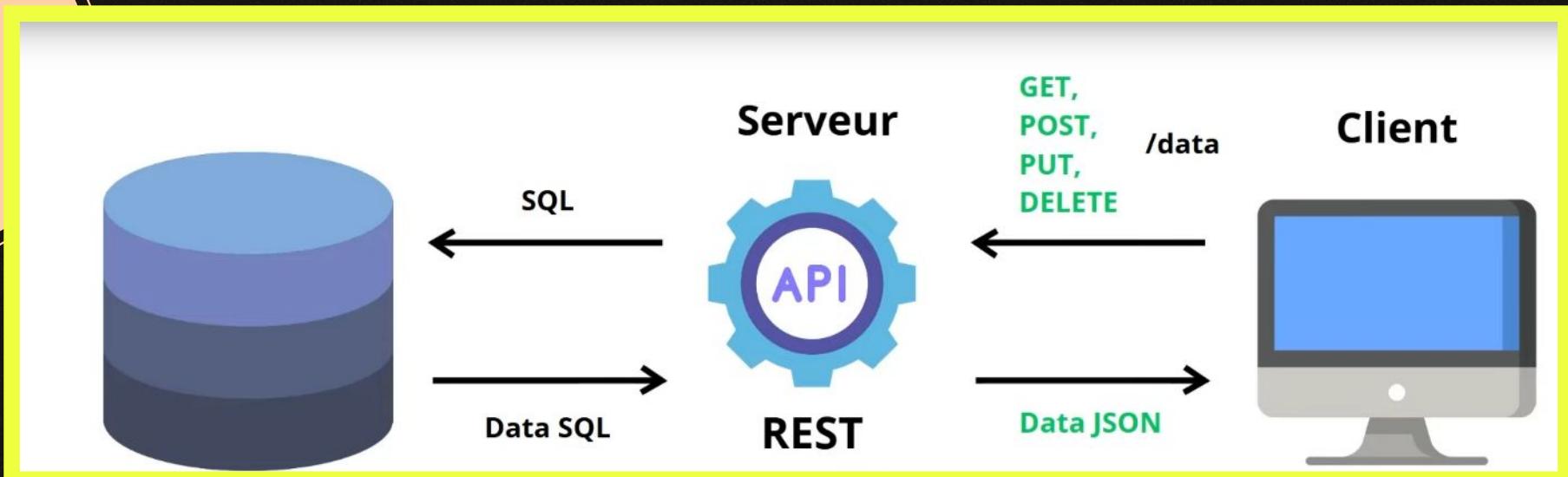
Une API REST qui relie le frontend (React/Next.js) au backend (API Routes + Prisma qui communique avec la bdd PostgreSQL Neon).

## 2. À quoi ça sert ?

Structurer et sécuriser les échanges de données : articles, favoris, administration.

## 3. Pourquoi ?

Séparer frontend/backend, assurer la sécurité, la cohérence et faciliter la maintenance.



# API (exemples) | CRUD des articles/[slug]

```
1. POST `/api/articles`

But : Créer un nouvel article
Qui : Admin uniquement (JWT requis)
Exemple fetch côté frontend :

```javascript  
const res = await fetch('/api/articles', {  
    method: 'POST', // Méthode HTTP pour créer  
    headers: {  
        'Content-Type': 'application/json', // Format des données envoyées  
        'Authorization': `Bearer ${token}` // Token JWT pour l'authentification admin  
    },  
    body: JSON.stringify({  
        title, slug, author, description, image, content, // Champs principaux de l'article  
        categoryId, contentBlocks, paragraphs // Relations et structures complexes  
    })  
});
```

1. Explication : Vérification du token admin, Prisma crée un nouvel article en base avec les données requises.

```
## 4. DELETE `/api/articles/[slug]`  
  
### **But** : Supprimer un article (suppression)  
### **Qui** : Admin uniquement (JWT requis)  
### **Exemple fetch côté frontend** :  
  
```javascript  
const res = await fetch(`/api/articles/${slug}`, {
 method: 'DELETE', // Méthode HTTP pour supprimer
 headers: {
 'Authorization': `Bearer ${token}` // Token JWT admin obligatoire
 }
});
```

**4. Explication :**

Le serveur vérifie le rôle admin, puis supprime l'article avec Prisma

```
2. GET `/api/articles/[slug]`

But : Récupérer les détails d'un article publié (lecture)
Qui : Tout utilisateur (public)
Exemple fetch côté frontend :

```javascript  
const res = await fetch(`/api/articles/${slug}`); // Requête GET vers l'API avec le slug  
const article = await res.json(); // Conversion de la réponse en JSON  
```
```

**2. Explication :** Prisma récupère l'article dont le slug correspond, uniquement si son statut est 'publié'. Retourne l'article avec les blocs de contenu, paragraphes et catégorie.

```
3. PUT `/api/articles/[slug]`

But : Modifier un article existant
Qui : Admin uniquement (JWT requis)
Exemple fetch côté frontend :

```javascript  
const res = await fetch(`/api/articles/${slug}`, {  
    method: 'PUT', // Méthode HTTP pour modifier  
    headers: {  
        'Content-Type': 'application/json', // Format des données  
        'Authorization': `Bearer ${token}` // Token JWT admin requis  
    },  
    body: JSON.stringify(updatedData) // Données mises à jour  
});
```

3. Explication : Vérification du token admin, puis mise à jour de l'article filtré, description, contenu, image, blocs, paragraphes, etc.) via Prisma.

Démo en live | Créer & modifier un article

AstroNex

Accueil À Propos Services > Articles FAQ

PRENDRE RDV MON PROFIL

Admin Panel

ARTICLES UTILISATEURS NEWSLETTER

Gestion des articles

Ajouter Modifier Supprimer

 Retrouver l'harmonie intérieure
Reconnectez vous à votre énergie...
par Fanny
harmonie-énergie
 Publié
Modifier

 S'écouter
Apprendre à écouter ses besoin...
par Alexia
s-écouter
 Publié
Modifier

 Poser l'intention sur ses objectifs
Méthode en 4 étapes pour tran...
par Alexia
poser-intention-objectifs
 Publié
Modifier

Précédent Page 1 / 5 Suivant

SE DÉCONNECTER

Authentification & Sécurité

```
#### *Chiffrement des mots de passe*
```javascript
// src/app/api/register/route.js
import bcrypt from 'bcryptjs';

const salt = bcrypt.genSaltSync(10);
const hashedPassword = bcrypt.hashSync(password, salt);
```

```

1

```
#### **JWT (JSON Web Token)**
```javascript
// src/app/api/login/route.js
import jwt from 'jsonwebtoken';

const payload = {
 id: user.id,
 email: user.email,
 firstname: user.firstname || user.firstName,
 role: user.role || userType
};

const token = jwt.sign(
 payload,
 process.env.JWT_SECRET
) // Durée optionnelle
```

```

2

```
68 #### *Validation forte des mots de passe*
69 ```javascript
70 // src/app/api/register/route.js
71 function isStrongPassword(password) {
72     const lengthCheck = password.length >= 12;
73     const lowercaseCheck = /[a-z]/.test(password);
74     const uppercaseCheck = /[A-Z]/.test(password);
75     const digitCheck = /\d/.test(password);
76     const specialCharCheck = /[#$!%*&]/.test(password);
77
78     return {
79         isValid: lengthCheck && lowercaseCheck && uppercaseCheck && digitCheck && specialCharCheck,
80         lengthCheck,
81         lowercaseCheck,
82         uppercaseCheck,
83         digitCheck,
84         specialCharCheck
85     };
86 }
```

```

3

```
193 #### *Gestion de l'expiration des tokens*
194 ```javascript
195 // Vérification côté client
196 export function isLoggedIn() {
197 if (typeof window === 'undefined') return false;
198
199 const token = window.localStorage.getItem('token');
200 if (!token) return false;
201
202 try {
203 const payload = JSON.parse(atob(token.split('.')[1]));
204 // Vérification de l'expiration
205 const currentTime = Math.floor(Date.now() / 1000);
206 return payload.exp > currentTime;
207 } catch (error) {
208 return false;
209 }
210 }
```

```

4

```
106 #### **Vérification des accès - Middleware JWT**
107 ```javascript
108 // src/app/api/middleware/route.js
109 export function verifyJWT(req) {
110     try {
111         const authHeader = req.headers.get('authorization');
112         if (!authHeader || !authHeader.startsWith('Bearer ')) {
113             return {isValid: false, error: 'Token manquant'};
114         }
115         const token = authHeader.split(' ')[1];
116         const decoded = jwt.verify(token, SECRET);
117         req.user = decoded;
118     } catch (err) {
119         return {isValid: true, user: decoded};
120     }
121     return {isValid: false, error: 'Token invalide'};
```

```

1

```
139 #### **protection contre les attaques - Prisma ORM**
140 ```javascript
141 // Utilisation dans les routes API
142 import { PrismaClient } from '@prisma/client';
143
144 const prisma = new PrismaClient();
145
146 // Requête sécurisée avec Prisma
147 const user = await prisma.user.findUnique({
148 where: { email
149 });
```

```

2

```
166 #### **Audit régulier avec npm audit**
167 ```bash
168 # Commande d'audit de sécurité
169 npm audit
170
171 # ...
```

```

3

```
227 #### **Gestion des erreurs sécurisée**
228 ```javascript
229 // Messages génériques pour éviter les fuites d'information
230 catch (error) {
231 console.error('Erreur login:', error); // Log interne détaillé
232 return NextResponse.json({
233 message: 'Erreur interne du serveur' // Message utilisateur générique
234 }, { status: 500 });
235 }
```

```

Tests (jest)

```
PS C:\Users\desig\Desktop\Next.Js - Projet TP DWM\alexia-energies> npm test auth.test.js

> site-vitrine@0.1.0 test
> jest auth.test.js

PASS  src/lib/_tests_/auth.test.js
  Auth Utilities
    isLoggedIn
      ✓ returns false when window is undefined (SSR) (1 ms)
      ✓ returns true when token cookie exists (2 ms)
      ✓ returns true when token in localStorage exists (1 ms)
      ✓ returns false when no token found
      ✓ returns true when both cookie and localStorage have tokens
    isAdmin
      ✓ returns false when window is undefined (SSR)
      ✓ returns false when no token in localStorage (1 ms)
      ✓ returns true when token contains admin role
      ✓ returns false when token contains user role (1 ms)
      ✓ returns false when token is malformed (5 ms)
      ✓ handles JSON parsing errors gracefully (1 ms)
    isUser
      ✓ returns false when not logged in
      ✓ returns false when user is admin
      ✓ returns true when user is logged in and not admin (1 ms)
  Edge cases
    ✓ handles empty cookie string
    ✓ handles multiple cookies without token
    ✓ handles token cookie with spaces

Test Suites: 1 passed, 1 total
Tests:       17 passed, 17 total
Snapshots:  0 total
Time:        0.719 s, estimated 1 s
Ran all test suites matching /auth.test.js/i.
```

Tests unitaires

(Jest) : est un outil qui permet d'automatiser les tests du code JavaScript pour vérifier qu'il fonctionne correctement.

auth.test.js : teste les fonctions d'authentification (connexion, rôles, gestion des tokens, cas limites...)

Un **test unitaire** : vérifie une fonction ou un module isolé, pour s'assurer qu'il renvoie bien le résultat attendu dans différents cas.

Tests avec Postman

The screenshot shows the Postman application interface. At the top, there's a search bar and a 'Getting started' button. Below that, the URL is set to 'http://localhost:3000/api/admin/login'. The method is 'POST' and the target URL is 'http://localhost:3000/api/admin/login'. The 'Headers' tab is selected, showing '(8)' entries. The 'Body' tab is also visible. In the main area, there's a table for 'Query Params' with two rows: 'Key' and 'Value'. Below this, the 'Body' tab is expanded, showing a JSON response with a status of '200 OK' and a response time of '280 ms'. The response body is a JSON object:

```
1 {  
2   "message": "Connexion réussie",  
3   "token": "eyJhbGciOiJIUzI1NiRzI6IkpXVCJ9.  
4     eyJpZCI6ImNtY3p2M0MsD4wvDkakBzr21bDZjxA1lC3lbWFpbCI6ImZhbh56LnNhZXouMDQ4NbnbWFpbC5jb28lCjpc0FkbWluIjp0cnVlLC3maXjzxE5hbhU1OjJGyW5ueSIsImhdCI6MTc1MjUwNTU4Miwi  
5     iZKhwijoxhzu2MTewMzg/yfQ.L8lUpPdnf02TY1l0XwvdfzofQKPaQ211UI5_NL1okw",  
6   "admin": {  
7     "id": "cmcvzb834p0800dhkkb916cmp",  
8     "email": "fanny.saez.0406@gmail.com",  
9     "isAdmin": true,  
10    "firstName": "Fanny"  
11  }  
12 }
```

1. Qu'est-ce que c'est ?

Postman est un outil qui simule des requêtes HTTP pour tester les endpoints de l'API.

2. À quoi ça sert ?

Vérifier les réponses du serveur (**statut, JSON, authentification**) et s'assurer que l'API fonctionne correctement côté client.

3. Pourquoi ?

Pour garantir la **fiabilité, la sécurité**

Vercel

créer un fichier
.env.local => propre
au serveur par contre
.env (production)

```
# Base de données
DATABASE_URL="postgresql://username:password@host:port/database"

# Authentification
JWT_SECRET="your-super-secret-jwt-key-minimum-32-characters"

# Cloudinary (images)
CLOUDINARY_CLOUD_NAME="your-cloud-name"
CLOUDINARY_API_KEY="your-api-key"
CLOUDINARY_API_SECRET="your-api-secret"

# Email (Resend)
RESEND_API_KEY="re_your-resend-api-key"

# NextAuth (optionnel)
NEXTAUTH_SECRET="your-nextauth-secret"
NEXTAUTH_URL="https://your-domain.com"
```

Le déploiement est automatisé via
GitHub & Vercel

Déploiement en Production

Prérequis

- Node.js 18+
- NPM ou Yarn
- Compte Vercel (déploiement)
- Base de données PostgreSQL (Neon)

| | | | | |
|-----------------------------|------------------|-------|---------------|-----|
| RESEND_API_KEY | All Environments | | Added Aug 9 | ... |
| NEXT_PUBLIC_BASE_URL | All Environments | | Added Aug 9 | ... |
| EMAILJS_SERVICE_ID | All Environments | | Updated Aug 9 | ... |
| EMAILJS_TEMPLATE_ID_CONTACT | All Environments | | Updated Aug 9 | ... |
| EMAILJS_TEMPLATE_ID_RESET | All Environments | | Added Aug 9 | ... |
| EMAILJS_USER_ID | All Environments | | Added Aug 9 | ... |
| CLOUDINARY_CLOUD_NAME | Production | | Added Aug 4 | ... |
| CLOUDINARY_API_KEY | Production | | Added Aug 4 | ... |
| CLOUDINARY_API_SECRET | Production | | Added Aug 4 | ... |
| DATABASE_URL | All Environments | | Added Jul 27 | ... |
| JWT_SECRET | All Environments | | Added Jul 27 | ... |

Vercel est une plateforme cloud qui déploie et héberge automatiquement les applis web (Next.js, React...) via GitHub et les outils de performance.

Démo | Système de favoris



Conclusion

Ce stage en freelance m'a vraiment fait progresser. En tant qu'autodidacte, j'ai pu gérer le projet de A à Z et relever des défis qui m'ont fait gagner en compétences et en confiance.

Merci pour votre attention !