## Présentation Projet Fil Rouge

## Choix Technique et fonctionnels du projet

Dans le cadre de notre projet de site **Cyna**, nous avons fait des choix techniques précis afin d'assurer une bonne performance, une facilité de développement et une évolutivité du site. Voici les principales technologies et outils utilisés :

**Frontend**

* **React.js**  
  Nous avons utilisé React.js pour construire l’interface utilisateur. Ce framework JavaScript permet de créer des composants réutilisables et dynamiques, facilitant la création d’une expérience utilisateur fluide et réactive.

**Backend**

* **Next.js**  
  Le backend de l’application repose sur Next.js, un framework basé sur React qui permet à la fois le rendu côté serveur (SSR) et la génération de pages statiques. Ce choix nous permet d’optimiser le SEO et d’améliorer la vitesse de chargement des pages.

**ORM & Base de données**

* **Prisma**  
  Pour interagir avec la base de données, nous avons opté pour Prisma, un ORM (Object-Relational Mapping) moderne et performant. Prisma facilite la gestion des modèles de données, les migrations, et l’écriture de requêtes sécurisées et typées.
* **SQLite**  
  Nous avons choisi SQLite comme moteur de base de données pour sa simplicité d'utilisation et sa légèreté, particulièrement adaptée au développement local et aux projets à petite échelle. La base de données est facilement intégrée avec Prisma.

**Raisons de ces choix**

* **Simplicité de mise en place** : les outils choisis sont bien documentés, largement utilisés dans la communauté et rapides à mettre en œuvre.
* **Productivité** : l’association React.js / Next.js permet un développement rapide avec une architecture claire.
* **Flexibilité** : Prisma facilite l’évolution du modèle de données tout au long du projet.
* **Légèreté** : SQLite est suffisant pour un projet en phase de développement ou de démonstration.

## Méthode de gestion de Projet

Pour la réalisation du projet **Cyna**, nous avons adopté une **méthode de gestion Agile**, adaptée aux projets collaboratifs en équipe réduite et permettant une progression itérative et flexible.

**Organisation de l’équipe**

Notre groupe est composé de **trois membres**, chacun ayant un rôle bien défini afin de répartir efficacement les tâches :

* **Développeur Frontend** : chargé de la conception et de l’implémentation de l’interface utilisateur à l’aide de **React.js**.
* **Développeur Backend** : responsable de la logique serveur avec **Next.js**, de la mise en place de la base de données via **SQLite**, et de la gestion des modèles avec **Prisma**.
* **Responsable documentation et support technique** : en charge de la **rédaction de la documentation** (technique et utilisateur), de l’organisation du travail et de la coordination entre les membres. Il a également :
  + **Contribué au code** sur certaines parties, notamment pour des ajustements mineurs ou des éléments secondaires,
  + **Intégré des éléments liés au RGPD** (Règlement Général sur la Protection des Données), notamment en ce qui concerne **l’accès, la modification et la suppression des données personnelles** des utilisateurs sur le site.

#### **Fonctionnement Agile**

Nous avons appliqué les principes de l’**Agilité** à notre échelle :

* **Travail en itérations courtes**, avec des objectifs définis à chaque étape (ex. mise en place du front, intégration des routes backend, connexion à la base de données...).
* **Flexibilité** dans la répartition des tâches, selon les compétences et la charge de chacun.
* **Tests et retours continus**, permettant d’améliorer progressivement les fonctionnalités et l’interface.

Cette approche nous a permis de **collaborer efficacement**, d’avoir une bonne **vision de l’état d’avancement** du projet à tout moment, et de **réagir rapidement** en cas de difficultés techniques.

## Planning Détaillé du projet

## Les outils utiliser pour travailler en équipe

Pour assurer une bonne communication et une coordination efficace entre les membres du groupe tout au long du développement du projet **Cyna**, nous avons utilisé plusieurs outils collaboratifs :

* **Discord** : notre principal outil de communication. Il nous a permis d’échanger en temps réel, de planifier nos séances de travail, de partager des idées et de résoudre rapidement les problèmes rencontrés.
* **GitHub** : utilisé pour le **versionnage du code** et la collaboration technique. Chaque membre poussait régulièrement ses modifications, ce qui nous a permis de suivre l’évolution du projet, d’éviter les conflits de code et de travailler en parallèle sur différentes fonctionnalités.
* **Microsoft Word** : utilisé pour la **rédaction et la mise en forme de la documentation** du projet (choix techniques, gestion de projet, planning, etc.).
* **JetBrains IDEs** (WebStorm, IntelliJ) : environnement de développement professionnel pour le backend, avec autocomplétion avancée, refactoring et intégration Git.
* **Visual Studio Code** : éditeur léger et extensible pour le frontend, avec de nombreuses extensions (Tailwind CSS IntelliSense) et intégration de débogueur.

## Respect du RGPD

(**Montrer en Live**)

## Choix de la licence

(**Montrer le fichier LICENCE.txt)**

**Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.**

## Crédits des ressources utilisées

* **React.js : framework JavaScript utilisé pour le développement de l’interface utilisateur (frontend).**
* **Next.js : framework basé sur React, utilisé pour le backend et le rendu côté serveur.**
* **JetBrains IDEs (WebStorm, IntelliJ) : environnement professionnel pour le développement backend, avec autocomplétion, refactoring et intégration Git.**
* **Visual Studio Code : éditeur léger et extensible pour le frontend, équipé d’extensions Tailwind CSS IntelliSense**
* **Prisma : ORM utilisé pour gérer les interactions avec la base de données.**
* **SQLite : base de données légère utilisée dans le projet.**
* **Documentation et standards web :** [**W3C**](https://www.w3.org/) **pour les bonnes pratiques d’accessibilité, de structure HTML et de conformité RGPD.**
* **Outils de gestion de projet :**
  + **Discord pour la communication entre les membres de l’équipe.**
  + **GitHub pour le versionnage du code et la collaboration.**
  + **Microsoft Word pour la rédaction de la documentation.**

## Liste des sources de chaque contenu

**1. Code / Librairies utilisées**

| **Contenu** | **Source / Lien** | **Remarques** |
| --- | --- | --- |
| **React.js** | [**https://reactjs.org**](https://reactjs.org) | **Utilisé pour l’interface utilisateur** |
| **Next.js** | [**https://nextjs.org**](https://nextjs.org) | **Utilisé pour le backend (API + SSR)** |
| **Prisma** | [**https://www.prisma.io**](https://www.prisma.io) | **ORM pour la base de données** |
| **SQLite** | [**https://www.sqlite.org**](https://www.sqlite.org) | **Base de données légère locale** |
| **Tailwind CSS** | [**https://tailwindcss.com**](https://tailwindcss.com) | **Framework utilitaire pour le stylisme et la gestion des thèmes** |
| **Global CSS** | **./styles/globals.css** | **Fichier de styles globaux personnalisés (reset, variables, etc.)** |
| **Stripe** | [**https://stripe.com**](https://stripe.com) | **Intégration de paiement sécurisé** |

**2. Polices d’écriture**

| **Nom de la police** | **Source / Lien** | **Licence** |
| --- | --- | --- |
| sans-serif | Par défaut du navigateur | Standard web, aucune licence requise |

**3. Documentation et normes**

| **Contenu** | **Source / Lien** | **Utilité** |
| --- | --- | --- |
| Normes HTML/CSS | <https://www.w3.org> | Conformité W3C, bonnes pratiques Web |
| RGPD | <https://www.cnil.fr> | Références pour la gestion des données |

**4. Ressources d’apprentissage / Assistance**

| **Ressource** | **Lien / Plateforme** | **Utilisation** |
| --- | --- | --- |
| Stack Overflow | <https://stackoverflow.com> | Résolution de bugs et questions techniques |
| Documentation Prisma | https://www.prisma.io/docs | Création des modèles et requêtes |
| Documentation Next.js | https://nextjs.org/docs | Gestion des routes, API et SSR |
| ChatGPT | <https://chat.openai.com> | Aide à la rédaction, corrections, structuration du rapport |

## Normes d’accessibilité

**« Prouver que le texte est lisible (rapport de contraste ≥ 4.5:1) »**

Nous avons vérifié les contrastes de texte sur la page d'accueil du site Cyna pour assurer sa conformité aux normes **WCAG 2.1 niveau AA** :

* En **mode sombre**, tous les textes (blanc ou gris clair) sur fond noir ont un **rapport supérieur à 7:1**, ce qui est **totalement conforme**.
* En **mode clair**, le texte principal présente un **dégradé violet / bleu** qui, selon les nuances, peut descendre en dessous du **contraste minimal de 4.5:1**. Des ajustements de teintes plus foncées ont été envisagés pour corriger cela.
* Les paragraphes et boutons ont tous un **contraste suffisant**, supérieur à 5:1 dans les deux thèmes.

L’ensemble de la page respecte donc les **principes d’accessibilité** en matière de lisibilité pour les utilisateurs.

**« Faire une démo de navigation sans souris (juste avec Tab) » et « Démontrer le style visible pour les éléments focusables »**

(Montrer en live)

**« Vérifier que chaque image a un attribut alt »**

Exemple :

Une image contenant texte, ordinateur, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.

**« Montrer que la structure HTML est claire »**

(Montrer en live)