Cahier des charges

Projet Application web 212-Colors

Table des matières

1. Introduction

Contexte du projet

212 Colors est une application web dédiée à la gestion d'identité visuelle. Elle vise à offrir aux utilisateurs un espace centralisé et intuitif pour définir, organiser, et maintenir les règles visuelles de leurs projets créatifs, qu'il s'agisse de sites web, de marques, ou d'autres types de productions visuelles.

Dans un monde où la cohérence de l'image de marque est cruciale pour se démarquer, la gestion des éléments visuels (couleurs, polices, logos) est souvent fragmentée et éparpillée entre divers outils ou équipes. Cette complexité peut entraîner des incohérences ou des erreurs coûteuses, tant en termes d'image que de temps.

212 Colors se positionne comme une solution innovante permettant de centraliser toutes ces ressources visuelles au même endroit. L'application répond à un besoin croissant d'harmonisation et de simplification de la gestion de l'identité visuelle, que ce soit pour les agences de design, les entreprises ou les indépendants.

Avec l'essor des projets digitaux et la multiplication des supports (web, mobile, print), la nécessité de conserver une identité visuelle cohérente à travers ces plateformes est devenue un enjeu stratégique pour de nombreuses entreprises. **212 Colors** propose ainsi un outil flexible, facile à utiliser, et capable de s'adapter aux besoins spécifiques de chaque utilisateur.

Objectifs généraux

L'objectif de **212 Colors** est de devenir un outil de référence pour la gestion d'identité visuelle, offrant aux utilisateurs un espace unique et centralisé pour définir, organiser et visualiser les éléments graphiques de leurs projets. Plus spécifiquement, le projet vise à :

- **Assurer la cohérence visuelle** : Aider les utilisateurs à maintenir une identité visuelle homogène à travers différents canaux.
- **Simplifier la gestion des éléments graphiques** : Proposer une interface intuitive pour gérer efficacement les couleurs, typographies, logos et autres composants graphiques.
- **Accélérer la productivité des équipes** : Offrir une solution qui centralise toutes les ressources visuelles, facilitant la collaboration entre designers, développeurs et autres parties prenantes.

- **Renforcer la flexibilité et la personnalisation** : Permettre aux utilisateurs d'adapter les ressources visuelles en fonction des besoins spécifiques de chaque projet, qu'il s'agisse de la création de nouveaux thèmes ou de l'ajustement des éléments graphiques.
- **Intégrer des technologies avancées** : Utiliser l'intelligence artificielle pour aider les utilisateurs à mieux contextualiser leurs projets, en générant des recommandations pertinentes en matière de couleurs, typographies, et autres aspects visuels.

Public cible

L'application **212 Colors** s'adresse à un large éventail de professionnels et de créateurs qui ont besoin de gérer efficacement leur identité visuelle. Le public cible comprend notamment :

- **Designers graphiques** : Professionnels responsables de la création et de la gestion des chartes graphiques, qui ont besoin d'un outil centralisé pour organiser les palettes de couleurs, typographies, et logos de leurs projets.
- **Agences de communication et de branding** : Entreprises spécialisées dans la gestion de l'image de marque, cherchant une solution pour centraliser et structurer les éléments visuels de leurs clients afin de garantir une cohérence sur tous les supports.
- **Développeurs web et mobile** : Professionnels techniques qui doivent appliquer les éléments visuels définis par les designers de manière uniforme dans leurs projets de développement, qu'ils travaillent sur des sites web, des applications mobiles, ou d'autres plateformes numériques.
- **Entreprises et startups** : Organisations qui souhaitent maintenir une identité visuelle cohérente à travers différents projets et supports, en facilitant la communication entre les équipes de design, marketing, et développement.
- **Indépendants et créateurs de contenu** : Freelancers ou petits créateurs qui gèrent eux-mêmes leur identité visuelle et cherchent un outil pour organiser leurs choix graphiques de manière structurée.

2. Fonctionnalités

Fonctionnalités principales

Les fonctionnalités principales de 212 Colors sont conçues pour offrir une gestion centralisée et intuitive des éléments visuels d'un projet. Elles comprennent :

- **Centralisation des éléments graphiques** : L'application regroupe dans une interface unique les ressources visuelles comme les couleurs, les polices et les logos, permettant une gestion intuitive et organisée.
- **Contextualisation de projet** : Permet à l'utilisateur de définir des informations clés sur son projet (nom, description, thème, etc.), facilitant ainsi la gestion des ressources visuelles.
- **Gestion des couleurs** : L'utilisateur peut définir une palette de couleurs pour chaque projet, classée en trois catégories : couleurs principales, secondaires, et spéciales (succès, erreur, avertissement, etc.).
- **Création de thèmes** : L'utilisateur peut créer des thèmes (clair ou sombre) et attribuer des couleurs spécifiques à différents types de couleurs dans chaque thème.
- **Suggestion IA** : L'IA de l'application propose des suggestions adaptées pour les couleurs, les noms de couleurs, et d'autres éléments visuels en fonction du contexte défini par l'utilisateur.
- **Gestion de la typographie** : L'utilisateur peut choisir une typographie principale et, si nécessaire, une typographie secondaire, permettant de visualiser le rendu selon divers scénarios.
- **Gestion des logotypes** : L'utilisateur peut définir différentes versions de son logo (monochrome, avec nom, avec fond) et visualiser le rendu selon divers scénarios (différentes tailles et fonds).
- **Gestion des bannières** : L'application permet à l'utilisateur de créer et de gérer des bannières, en spécifiant différents types, ratios et positions du logo.

MVP (Minimum Viable Product)

Le **MVP** de **212 Colors** se concentrera sur les fonctionnalités essentielles pour offrir une première version utilisable de l'application. Les fonctionnalités incluses dans cette première version sont :

- **Création et gestion de compte** : Inscription via email (avec possibilité de changement de mot de passe) et via OAuth (GitHub).

- **Création et gestion de projet** : Les utilisateurs peuvent créer des projets, définir leur nom, description, et activer ou désactiver certaines sections.
- **Vue globale du projet** : Un aperçu des informations clés du projet, y compris les sections activées.
- **Gestion des couleurs** : Possibilité de définir les couleurs principales, secondaires et spéciales avec un nom, une couleur en HEX et une description.
- **Gestion des thèmes** : Possibilité de définir les types de couleurs pour chaque thèmes (couleur de fond, couleur de contenu, etc.) ainsi que des thème de type clair ou sombre avec un nom, une description et une liste de couleurs pour chaque types

3. Spécifications techniques

Architecture du projet

L'architecture du projet est structurée de façon simple autour de trois pages principales : **Profile**, **Réglages** et **Projet**.

- **Profile** : Cette section gère les informations de l'utilisateur, permettant de modifier le mot de passe, d'afficher les détails du compte, et d'effectuer des actions telles que la déconnexion.
- **Paramètres** : Permet aux utilisateurs de personnaliser leur expérience avec des options comme pour changer la langue de l'application et choisir le thème (clair ou sombre).
- **Projet**
    - **Vue d'ensemble** : Offre une présentation synthétique de toutes les données et visuels liés au projet, permettant une visualisation rapide des éléments clés.
    - **Informations** : Contient les détails spécifiques du projet, avec des options pour supprimer le projet et choisir les sections à activer ou désactiver (comme les couleurs ou les thèmes).
    - **Section Couleurs** : Permet aux utilisateurs de créer, gérer, réordonner et visualiser les palettes de couleurs associées au projet.
    - **Section Thèmes** : Gère les thèmes visuels du projet, permettant de créer et d'appliquer des thèmes basés sur les couleurs définies.

Contraintes techniques

Le projet doit respecter les contraintes suivantes :

- **Sécurité** : Mise en place de protocoles de sécurité robustes, incluant l'authentification des utilisateurs, le chiffrement des mots de passe, et des mesures de prévention contre les attaques courantes.
- **Accessibilité** : Respect des normes d'accessibilité (WCAG) pour garantir que l'application soit utilisable par tous, y compris les personnes en situation de handicap.
- **Multilinguisme** : L'application doit prendre en charge plusieurs langues pour répondre à un public diversifié. Les utilisateurs pourront choisir leur langue préférée et l'application s'adaptera en conséquence.
- **Thème simple** : Adoption d'un design minimaliste et épuré pour assurer une expérience utilisateur intuitive. Les éléments de l'interface doivent être clairement identifiables et faciles à utiliser.
- **Performance** : L'application doit être optimisée pour offrir un temps de chargement rapide et une fluidité dans l'interaction, y compris l'utilisation d'images et de ressources légères.
- **Compatibilité** : Garantir que l'application fonctionne de manière optimale sur les navigateurs modernes et sur les appareils mobiles.
- **Scalabilité** : Prévoir une architecture capable de supporter une augmentation du nombre d'utilisateurs et des fonctionnalités futures sans dégradation des performances.

Technologies utilisées

- **TypeScript** : Choisi comme langage principal pour garantir la cohérence et la maintenabilité du projet. TypeScript permet de partager des types et des interfaces entre le front-end et le back-end, facilitant ainsi le développement et réduisant les erreurs.
- **Next.js** : Utilisé comme framework pour le front-end et le back-end, offrant une unification de l'architecture de l'application. Next.js permet également un rendu côté serveur, améliorant les performances et le référencement.

Technologies du front-end :

- **Next.js Server Actions** : Cette fonctionnalité moderne permet d'exécuter des fonctions côté back-end directement depuis un composant front-end,

simplifiant les appels API et améliorant la performance.

- **React** : Utilisé pour construire des interfaces utilisateur dynamiques et réactives. React facilite la création de composants réutilisables, ce qui accélère le développement.
- **Ant Design** : Une bibliothèque de composants qui offre des éléments UI simples et élégants, tels que des boutons et des champs de saisie. Cela permet d'assurer une cohérence visuelle et de gagner du temps dans la conception des interfaces.
- **Antd Zod Validation** : Intégrée pour appliquer des validations aux formulaires en utilisant les schémas Zod. Cela améliore la sécurité et l'expérience utilisateur en fournissant des retours d'erreur clairs.
- **Zustand** : Un gestionnaire d'état léger et simple qui permet de partager des états dans l'application sans configuration complexe. Cela facilite la gestion de l'état global tout en gardant le code propre et maintenable.
- **colord** : Bibliothèque pour manipuler les couleurs, offrant une API simple et performante. Elle est essentielle pour la gestion des palettes de couleurs dans l'application.
- **Dart Sass** : Utilisé pour styliser l'application de manière plus pratique et dynamique. Sass permet de créer des styles plus modulaires et maintenables.
- **Tabler Icons** : Fournit une large collection d'icônes modernes et gratuites, enrichissant l'interface utilisateur et améliorant l'expérience visuelle.
- **dnd-kit** : Une bibliothèque robuste pour gérer le glisser-déposer dans React. Elle permet d'implémenter des interactions utilisateur intuitives et engageantes.

Technologies du back-end :

- **Prisma** : Un ORM qui permet d'interagir avec la base de données en utilisant une syntaxe intuitive. Prisma simplifie les opérations de base de données et améliore la sécurité des requêtes.
- **NextAuth.js** : Gère l'authentification par OAuth et par Credentials (email et mot de passe). Cela permet d'implémenter facilement des mécanismes d'authentification sécurisés.
- **bcrypt.js** : Utilisé pour hacher les mots de passe des utilisateurs, garantissant ainsi la sécurité des données sensibles.

- **Zod** : Permet de définir des schémas et de les valider sur les entrées utilisateur. Cela augmente la sécurité et la robustesse des données traitées par l'application.

Stockage :

- **Base de données** : PostgreSQL, choisie pour sa fiabilité et ses fonctionnalités avancées, offrant une gestion efficace des données.
- **Object Store** : Blob, utilisé pour le stockage de fichiers binaires. Cela permet de gérer les fichiers de manière scalable.
- Les deux sont hébergés sur Vercel, assurant une unification de l'architecture et une intégration transparente.

DevOps :

- **Documentation** : Storybook, utilisé pour documenter et visualiser les composants de l'interface utilisateur. Cela facilite la collaboration et l'intégration des nouvelles fonctionnalités.
- **Gestionnaire de paquets** : pnpm, offrant une gestion rapide et efficace des dépendances, réduisant les temps d'installation.
- **Tests** : Vitest, un framework moderne pour effectuer des tests unitaires et d'intégration, garantissant la qualité du code et le bon fonctionnement de l'application.

Schéma de la base de données

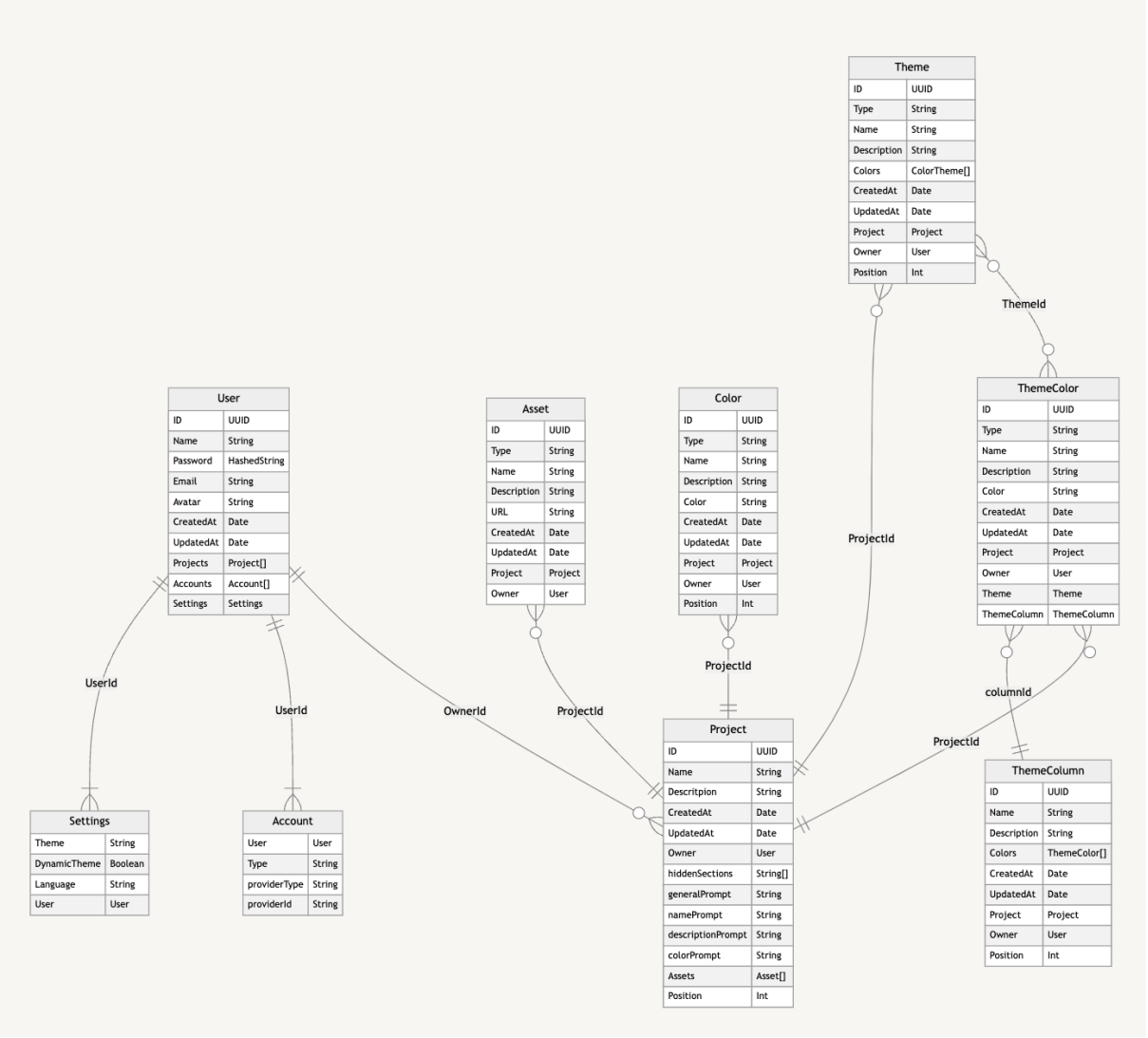
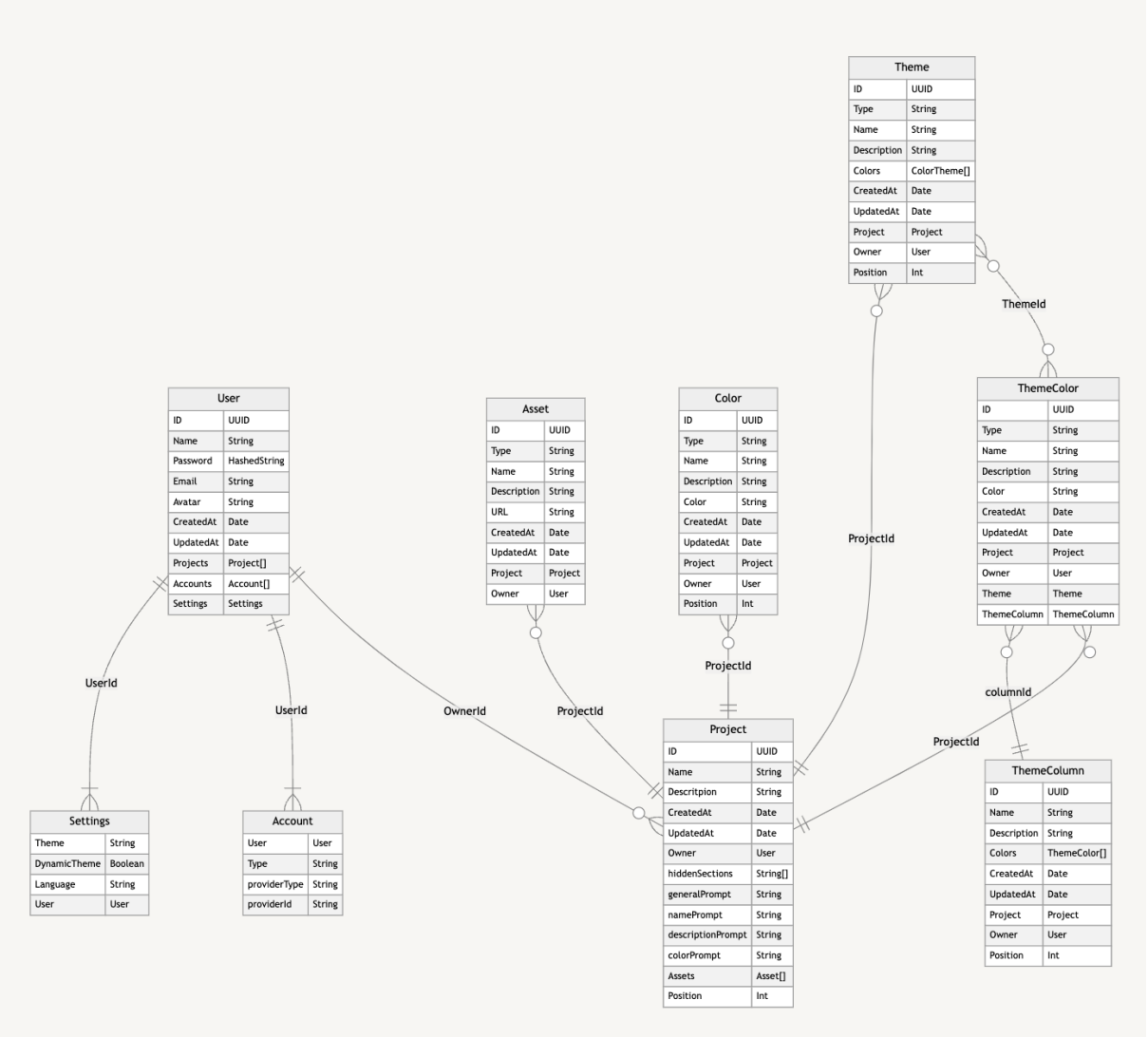


Diagramme de classe



Conception et interface utilisateur

Design et ergonomie

L'accent sera mis sur une interface utilisateur simple et intuitive, avec des éléments cohérents en termes de style, de couleurs, de tailles et de marges. L'ergonomie sera pensée pour minimiser les actions nécessaires à l'utilisateur pour accomplir ses tâches, rendant ainsi l'expérience utilisateur fluide et agréable.

La palette de couleurs sera définie comme suit pour assurer une expérience visuelle agréable en mode sombre comme en mode clair :

- Couleurs de fond
  - Mode sombre : #0f1319
  - Mode clair : #f5f5f5

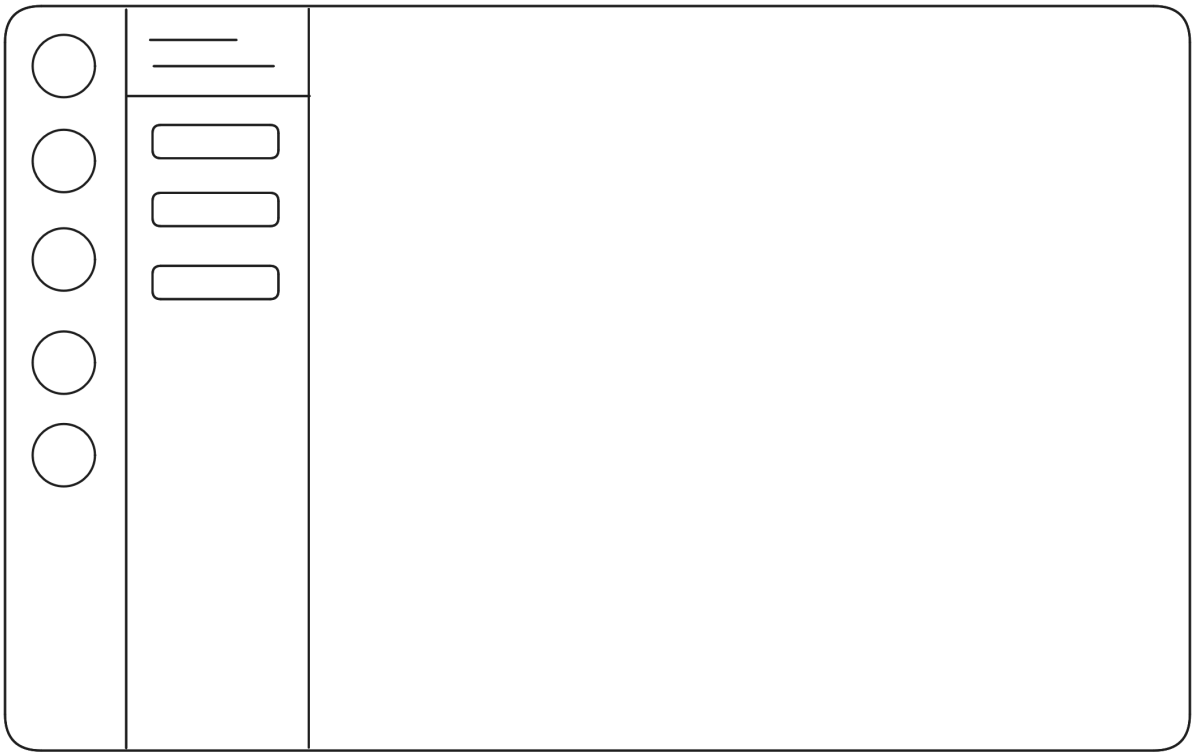
- Couleurs de contenu :
  - Mode sombre : #1c1b22
  - Mode clair : #ffffff
- Couleurs de contenu secondaire
  - Mode sombre : #13161c
  - Mode clair : #fafafa
- Couleur principale : #ff1818

Les transitions et animations seront subtiles pour maintenir l'interaction fluide et agréable sans distraire l'utilisateur.

Structure des pages

L'application sera structurée de manière uniforme sur toutes les pages pour maintenir la cohérence et faciliter la navigation.

- **Menu de navigation principal** : Situé à gauche de l'écran, ce menu vertical permettra à l'utilisateur de basculer entre ses différents projets ainsi que d'accéder à son compte et à ses paramètres.
- **Sous-menu de navigation** : Juste à côté du menu principal, un autre menu vertical affichera les différentes sections d'un projet (par exemple : couleurs, thèmes, typographie), permettant une navigation rapide au sein du projet en cours.



Maquettes

Les maquettes suivront un processus en deux étapes : d'abord des wireframes basse fidélité pour valider la structure et l'agencement des pages, puis des maquettes haute fidélité pour finaliser les détails visuels. Ces maquettes permettront de prévisualiser l'expérience utilisateur et d'identifier les ajustements à faire avant l'implémentation.

- **Wireframes** : Excalidraw sera utilisé pour créer des esquisses rapides et simples, parfaites pour les phases initiales de conception.
- **Maquettes haute fidélité** : Figma sera utilisé pour créer des maquettes détaillées avec tous les éléments visuels définitifs (couleurs, typographie, icônes, etc.). Ces maquettes serviront de guide lors du développement de l'interface.

Planification et gestion

Méthode et étape de développement

Le projet suivra une méthodologie de développement **agile** pour garantir flexibilité et adaptation rapide aux besoins et aux retours utilisateurs. Les sprints auront une durée fixe (2 semaines par exemple) et incluront la

planification des tâches, leur **implémentation** ainsi que la **révision des progrès** réalisés à la fin de chaque cycle.

Les principales étapes seront les suivantes :

1. **Recherche et planification**

Analyse des besoins, création du cahier des charges détaillé, définition des technologies, des outils, et des exigences du produit minimal viable (MVP). Cette étape inclura également la réalisation des premiers schémas d'architecture du projet.

2. **Réalisation des maquettes**

Création de l'ensemble des composants graphiques et des maquettes de toutes les pages via Figma. Cette phase permettra de valider l'aspect visuel et l'expérience utilisateur avant d'entrer dans la phase de développement.

3. **Initialisation du projet**

Mise en place du projet : création du dépôt GitHub, initialisation du répertoire, installation des dépendances, mise en place de la base de données, définition des schémas d'authentification, mise en place des types, routage, et intégration de Storybook pour faciliter le développement des composants.

4. **Création des composants**

Développement des composants UI un à un, à l'aide de Storybook pour garantir leur cohérence et leur réutilisabilité à travers l'application.

5. **Développement du MVP**

Implémentation des fonctionnalités essentielles définies lors de la phase de planification pour constituer le MVP. Cela inclura la gestion des projets, l'édition des couleurs, la navigation, etc.

6. **Ajout de l'IA**

Intégration des fonctionnalités d'intelligence artificielle permettant de suggérer des palettes de couleurs ou du texte basées sur les données existantes.

7. **Ajout des fonctionnalités supplémentaires**

Ajout des fonctionnalités non incluses dans le MVP, telles que la gestion de la typographie, du logotype et ed la bannière, ainsi que les fonctionnalités de personnalisation avancées.

Organisation

Tous les documents de projet, y compris le cahier des charges, les schémas, les notes et les maquettes, seront centralisés dans un répertoire **Notion**. Le **versioning** du code source sera géré avec Git, avec des branches distinctes pour chaque sprint ou fonctionnalité majeure. La gestion des tâches se fera également sur **Notion**, en utilisant un tableau Kanban pour visualiser l'avancement de chaque tâche, divisé par sprint.