

# DOSSIER DE SOUTENANCE POUR LE TITRE DE CONCEPTEUR DÉVELOPPEUR APPLICATION

# TABLE DES MATIÈRES

<b>INTRODUCTION AU PROJET</b>	<b>4</b>
1. Présentation personnelle	4
2. Présentation du projet en anglais	4
3. Compétences couvertes par le projet	5
<b>ORGANISATION ET CAHIER DES CHARGES</b>	<b>6</b>
1. Analyse de l'existant	6
2. Les utilisateurs du projet	6
3. Contexte technique	6
4. Les fonctionnalités attendues	6
1. Application mobile	6
2. Application web	6
<b>CONCEPTION DU PROJET</b>	<b>7</b>
1. Choix de développement	7
1. Choix des langages	7
2. Choix des frameworks	7
3. Logiciels et autres outils	7
2. Organisation du projet	7
3. Architecture logicielle	7
<b>CONCEPTION DU FRONT-END DE L'APPLICATION</b>	<b>8</b>
1. Arborescence du projet	8
2. Charte graphique	8
3. Maquettage	8
<b>CONCEPTION DU BACK-END DE L'APPLICATION</b>	<b>9</b>
1. La base de données	9
1. Concevoir une base de données	9
2. Mettre en place une base de données	9
3. Modèle conceptuel de données	9
4. Modèle logique de données	9
2. DÉVELOPPEMENT DU BACKEND DE L'APPLICATION	9
1. Organisation	9
2. Arborescence	9
3. Fonctionnement de l'API	9
4. Middleware	9
5. Routage	9
6. Controller	9
7. Service	9
8. Model	9

---

9. Sécurité	9
1. Chiffrement des données sensibles	9
2. JWT	9
3. Gestion des Droits	9
10. Problématique rencontrée	9
11. Problématique rencontrée	9
12. Recherches anglophones	9
13. Exemple :	9
14. Documentation	9
15. Tests	9
1. Postman	9
2. Newman	9
DÉVELOPPEMENT DU FRONT-END DE L'APPLICATION	11
1. Arborescence	11
2. Pages et composants	11
3. Sécurités	11
4. Problématiques rencontrées	11
5. Exemple navigation	11
6. Exemple de formulaire de mise à jour du profil	11
CONCEPTION DE L'ESPACE ADMINISTRATEUR	12
1. Conception de la partie administration	12
2. User Story	12
3. Choix du langage et frameworks	12
4. Conception du frontend du site web	12
1. Charte graphique	12
2. Maquettage	12
5. Conception du back end du site web	12
CONCLUSION	13
ANNEXES	14

---

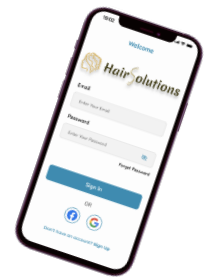
# INTRODUCTION AU PROJET

## 1. Présentation personnelle

Je m'appelle Mériem BARKA. J'ai découvert la programmation suite à une formation de designer web que j'ai suivie avec ACOPAD; Cela m'a permis de m'ouvrir aux métiers liés au web et plus particulièrement le développement. J'aime créer et trouver des solutions aux problèmes rencontrés dans tous les domaines. J'ai suivi le cursus de la coding School à la plateforme en 2022 et j'ai obtenu mon titre de développeur web et web mobile. Aujourd'hui je suis en Coding School 2 afin de préparer mon titre de concepteur et développeur d'application web en étant en alternance au sein de l'entreprise DEKI.

## 2. Présentation du projet en anglais

### HairSolution



We have developed a mobile application called "HairSolution" specifically designed to meet the needs of hair care enthusiasts.

We have observed that many people are in need of advice and guidance for their hair care.

Whether you are seeking personalized advice, solutions for specific hair issues, or simply engaging discussions with other hair care enthusiasts, our application is the ideal tool to guide you in your hair care routine.

With Hair Solution, you can expand your knowledge of hair care by joining a dynamic and committed community.

Share your experiences, ask questions, receive expert advice, and discover new tips to improve the health of your hair.

Our application provides a friendly space where users can interact with each other, exchange tips, and discuss the latest trends in hair care.

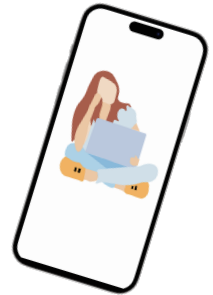
Download Hair Solution now and join a passionate and caring community!

---

### 3. Compétences couvertes par le projet

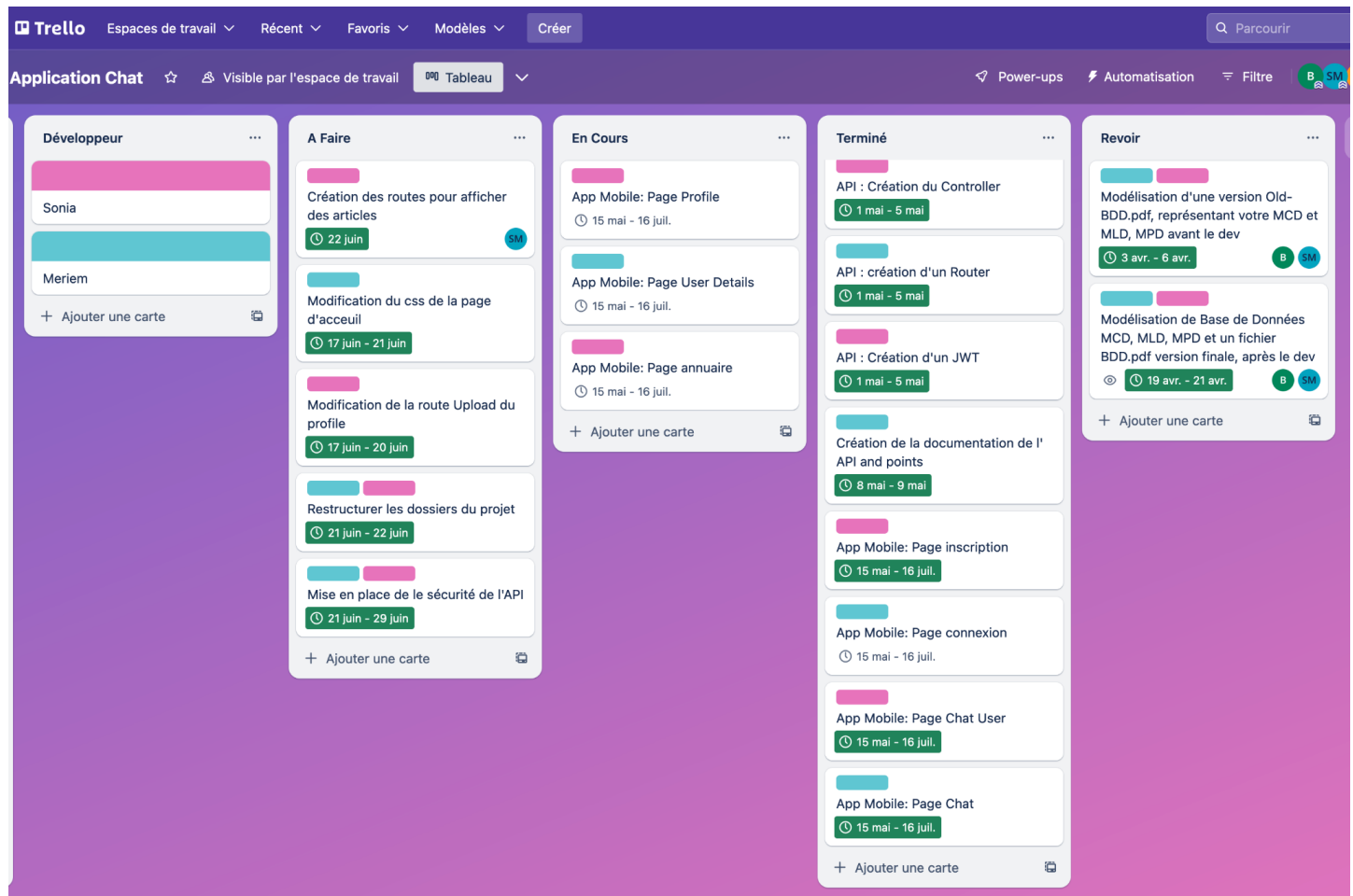
*Ce projet couvre les compétences du titre suivantes :*

- Maquetter une application
- Développer des composants d'accès aux données
- Développer la partie front-end d'une interface utilisateur web
- Développer la partie back-end d'une interface utilisateur web
- Concevoir une base de données
- Mettre en place une base de données
- Développer des composants dans le langage d'une base de données
- Concevoir une application
- Développer des composants métier
- Construire une application organisée en couches
- Développer une application mobile
- Préparer et exécuter les plans de tests d'une application
- Préparer et exécuter le déploiement d'une application



# ORGANISATION

Pour ce projet nous étions deux dans le groupe et nous nous sommes réparti les tâches de la conception à la livraison du projet. Nous nous sommes organisés de la manière suivante :



## 1. Définition du projet :

- Nous avons discuté ensemble de l'objectif et des spécifications du site web que nous souhaitons créer.
- Nous avons identifié les fonctionnalités clés et les exigences techniques.

## 2. Conception :

- On préfère le faire ensemble pour la conception graphique et l'élaboration de la charte graphique.
- Ainsi que sur la conception de l'architecture de l'information et la création des wireframes.

## 3. Développement :

- J'ai assumé la responsabilité du développement front-end, en utilisant react Native, expo.. pour créer l'interface utilisateur.

---

- J'ai pris en charge le développement back-end, "les routes" en utilisant des langages de programmation tels que node js, express pour mettre en place la logique et les fonctionnalités du site.

#### 4. Intégration et tests :

- Nous avons collaboré étroitement pour intégrer les éléments front-end et back-end et nous assurer que tout fonctionnait correctement.
- Nous avons effectué des tests réguliers pour détecter et corriger les bugs et les problèmes de performance.

#### 5. Révision et finalisation :

- Nous avons revu ensemble le site web dans son ensemble, en vérifiant la cohérence visuelle, la fluidité de la navigation et le bon fonctionnement de toutes les fonctionnalités.
- Nous avons apporté les dernières modifications nécessaires avant de considérer le projet comme terminé.

#### 6. Livraison :

- Ensemble, nous avons préparé les fichiers et les ressources nécessaires pour la mise en ligne du site web.
- Nous avons coordonné la mise en place du site sur un serveur ou une plateforme d'hébergement.

Dans cette organisation à deux personnes, nous avons tiré parti de nos compétences complémentaires pour mener à bien toutes les phases du projet, en assurant une communication régulière et une répartition équilibrée des responsabilités.

## CAHIER DES CHARGES

### 1. Analyse de l'existant

Nous avons effectué des recherches, mais nous n'avons pas trouvé d'application mobile qui propose ce que nous souhaitons offrir aux utilisateurs.

### 2. Les utilisateurs du projet

Le projet est coupé en deux parties : une application mobile pour les utilisateurs et une interface web qui est destiné à la personne qui va gérer l'application mobile .

Bienvenue dans l'application mobile dédiée aux soins capillaires ! Notre objectif principal est d'offrir aux utilisateurs une expérience enrichissante en lui fournissant des astuces capillaires de qualité.

L'application Mobile sera utiliser par les hommes et les femmes, jeunes ou plus âgé(e)s, quels que soient vos types de cheveux (bouclés, lisses, frisés, longs ou courts), que vous cherchiez à résoudre des problèmes spécifiques liés à vos cheveux ou simplement à découvrir de nouvelles astuces pour les sublimer partager vos propres astuces.

La partie web sera utilisée uniquement par la personne qui a le droit administrateur. cet espace lui permettra de poster des articles, de supprimer et de voir la liste des utilisateurs

### 3. Contexte technique

Pour l'application mobile on a fait le choix d'utiliser react native car c'est ce qui est le mieux adapté pour aller dans multiplateforme.

1. Langage de programmation : Le choix du langage de programmation que nous avons choisi est react native, car il offre plusieurs avantages significatifs :
  - **Multiplateforme** : React Native permet de développer une seule base de code qui fonctionne à la fois sur iOS et Android. Cela réduit considérablement les efforts de développement et de maintenance, car une grande partie du code peut être partagée entre les deux plateformes.
  - **JavaScript** : React Native utilise JavaScript, l'un des langages de programmation les plus populaires et répandus. Il est largement connu et maîtrisé par de nombreux développeurs, ce qui facilite le recrutement et la collaboration au sein de l'équipe de développement.
  - **Écosystème React** : React Native est basé sur React, une bibliothèque JavaScript bien établie pour la création d'interfaces utilisateur. Cela signifie que les développeurs qui connaissent déjà React pour le développement Web peuvent facilement transférer leurs compétences pour créer des applications mobiles avec React Native.
  - **Performances** : Bien que React Native ne soit pas aussi rapide que le développement natif (Kotlin pour Android, Swift pour iOS), il se rapproche beaucoup des performances natives. React Native utilise des composants natifs pour les parties critiques de l'application, ce qui améliore considérablement les performances par rapport aux frameworks hybrides traditionnels.
  - **Communauté active** : React Native bénéficie d'une large communauté de développeurs qui contribuent à son amélioration continue. Cela signifie qu'il existe de nombreuses bibliothèques open-source, des modules complémentaires et des ressources disponibles pour accélérer le développement.



- **Rapidité de développement** : Le fait de partager une grande partie du code entre les plateformes permet d'accélérer le processus de développement. Les équipes peuvent donc créer des applications plus rapidement, ce qui est un avantage essentiel dans le domaine des technologies mobiles.

- **Hot Reloading** : React Native propose une fonctionnalité appelée "Hot Reloading" qui permet aux développeurs de voir instantanément les changements apportés au code sans avoir à recompiler l'application. Cela facilite le processus de débogage et améliore la productivité des développeurs.

2. Environnement de développement nous avons intégré L'IDE Visual Studio Code c'est un IDE dédié pour écrire, tester et déboguer le code de l'application.  
Les IDE populaires incluent Xcode pour iOS et Android Studio pour Android.
4. Conception de l'interface utilisateur (UI) / Expérience utilisateur (UX) : La conception de l'interface utilisateur a été réalisée en accord avec les principes de l'expérience utilisateur pour créer une application attrayante et facile à utiliser.
5. Base de données : Pour stocker les informations telles que les profils des utilisateurs, les astuces capillaires, les préférences des utilisateurs, etc., l'application mobile nécessite une base de données.  
Nous avons opté pour l'utilisation d'une base de données NoSQL car cette technologie offre une grande flexibilité et évolutivité.  
Avec une base de données NoSQL, nous pouvons gérer facilement des données non structurées ou semi-structurées, ce qui correspond parfaitement aux besoins variables d'une application de soins capillaires.  
De plus, la capacité de mise à l'échelle horizontale des bases de données NoSQL nous permettra de gérer efficacement la croissance future de l'application et d'assurer des performances optimales, même avec un grand nombre d'utilisateurs et de données.
6. API : Si l'application se connecte à un serveur ou à des services tiers, des API (Interfaces de Programmation Applicative) seront utilisées pour faciliter les échanges de données.
7. Sécurité : Des mesures de sécurité ont été mises en place pour protéger les données des utilisateurs, telles que le chiffrement des données, l'authentification sécurisée, etc.
8. Notifications push : Pour envoyer des notifications aux utilisateurs, il faudra mettre en place un système de notifications push.
9. Test et débogage : Des tests rigoureux ont été effectués pour identifier et corriger les bugs avant le lancement de l'application.

---

avec l'outil postman et le console log pour débogue ou pour résoudre les problèmes techniques éventuels.

10. Déploiement : Une fois l'application prête, elle sera soumise aux magasins d'applications (App Store pour iOS, Google Play Store pour Android) pour être disponible en téléchargement.
11. Maintenance : Après le lancement, l'application nécessitera des mises à jour régulières pour ajouter de nouvelles fonctionnalités, corriger les bugs et s'adapter aux évolutions des plateformes mobiles.

## 12. Les fonctionnalités attendues

### 1. Application mobile

#### UTILISATEURS NON AUTHENTIFIÉ

**Page d'accueil :** l'utilisateur dispose de deux boutons lui permettant d'accéder à la page de connexion ou la page d'inscription.

**Page d'inscription :** l'utilisateur saisit les informations demandées dans le formulaire puis valide ce dernier. Si tout est correct, l'utilisateur est redirigé vers la page de connexion.

**Page de connexion :** l'utilisateur saisit son adresse email et son mot de passe. Si tout est correct, il est alors redirigé vers son espace personnel regroupant ses discussions.

#### UTILISATEURS AUTHENTIFIÉ

Les fonctionnalités attendues par les utilisateur de l'application mobile qui est dédiée aux soins et astuces capillaires :

**Page d'inscription :** Lorsque un nouvel utilisateur veut s'inscrire sur l'application mobile HairSolution.

**Page de connexion :** La page de connexion permet aux utilisateurs d'accéder à leur compte et d'utiliser les fonctionnalités qui leur sont réservées. L'envoi du message, poster des articles

**Tchat en direct :** Une fonctionnalité de messagerie instantanée qui permet aux utilisateurs d'échanger des messages en direct avec d'autres membres de la communauté.

---

**Forum de discussion** : Un espace où les utilisateurs peuvent créer des sujets de discussion, poser des questions, partager des astuces et des expériences, et interagir avec d'autres passionnés de soins capillaires.

**Partage d'astuces et conseils** : Les utilisateurs peuvent publier leurs propres astuces, conseils et routines capillaires pour les partager avec la communauté.

**Gestion de profil utilisateur** : Les utilisateurs peuvent créer et personnaliser leur profil, ajouter des informations sur leur type de cheveux, leurs préférences capillaires, etc.

Notifications personnalisées : Les utilisateurs peuvent choisir de recevoir des notifications sur les nouveaux messages, les réponses à leurs publications ou les sujets qui les intéressent.

Suivi des sujets favoris : Les utilisateurs peuvent marquer des sujets ou des discussions comme favoris pour y accéder facilement ultérieurement.

Système de likes et de commentaires : Les utilisateurs peuvent liker les publications d'autres membres et laisser des commentaires pour engager des discussions.

Recherche avancée : Une fonctionnalité de recherche qui permet aux utilisateurs de trouver des sujets spécifiques, des astuces ou des membres de la communauté.

Modération et rapport : Un système de modération pour gérer les contenus inappropriés et la possibilité pour les utilisateurs de signaler des publications problématiques.

Synchronisation des données : Les données de l'application, telles que les discussions et les profils des utilisateurs, peuvent être synchronisées pour une utilisation hors ligne.

En intégrant ces fonctionnalités de tchat et de forum, l'application mobile permettra aux utilisateurs d'échanger des informations, de partager leurs connaissances et de bénéficier d'une communauté engagée dans les soins et astuces capillaires.

## 2. Application web (administration)

Voici les fonctionnalités attendues pour l'application web destinée à l'administration et à la gestion de l'application dédiée aux soins capillaires :

**Page de connexion** : La page de connexion permet à l'admin d'accéder à son panel admin pour qu'il puisse utiliser les fonctionnalités attribuées pour l'admin

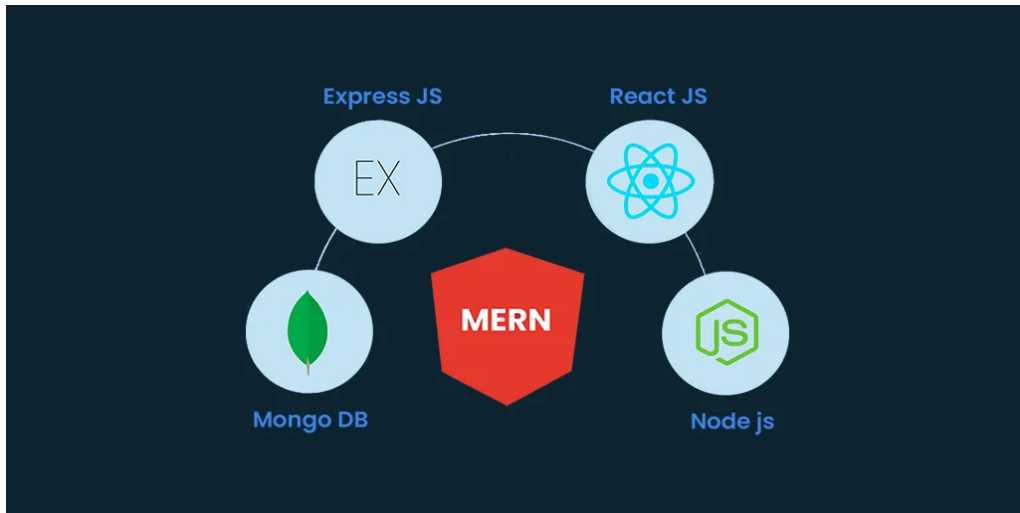
1. **Tableau de bord d'administration** : Un tableau de bord convivial et sécurisé pour les administrateurs, offrant un aperçu des principales activités et statistiques de l'application.
2. **Gestion des utilisateurs** : La possibilité pour les administrateurs de gérer les comptes des utilisateurs, la possibilité de les supprimer ou les bloquer et de gérer les autorisations.
3. **Modération des contenus** : Un système de modération qui permet aux administrateurs de surveiller et de modérer les publications, les commentaires et les discussions pour assurer le respect des règles de la communauté.
4. **Gestion des astuces et des tutoriels** : Les administrateurs peuvent ajouter, éditer ou supprimer des astuces, des conseils et des tutoriels pour les partager avec les utilisateurs.
5. **Gestion du contenu du forum** : La possibilité de gérer les sujets de discussion, d'approuver ou de supprimer des publications et de surveiller l'activité du forum.
6. **Statistiques et analyses** : Un système d'analyse qui fournit des rapports sur l'utilisation de l'application, l'engagement des utilisateurs, les sujets les plus populaires, etc.
7. **Gestion des notifications** : La possibilité de gérer les notifications envoyées aux utilisateurs, y compris les notifications push et les e-mails.
8. **Gestion des rapports d'utilisateurs** : La possibilité de gérer les rapports soumis par les utilisateurs concernant des contenus inappropriés ou des problèmes techniques.
9. **Système de sauvegarde et de restauration** : Un mécanisme de sauvegarde régulière des données de l'application pour la prévention des pertes de données et la possibilité de restaurer des sauvegardes si nécessaire.
10. **Paramètres du site** : Les administrateurs peuvent gérer les paramètres globaux de l'application, tels que les paramètres d'affichage, les options de langue, etc.

En intégrant ces fonctionnalités pour l'administration de l'application web, les administrateurs pourront gérer efficacement l'ensemble de l'application dédiée aux soins capillaires, assurer la qualité du contenu et offrir une expérience utilisateur optimale aux utilisateurs de la communauté.

## CONCEPTION DU PROJET

### 1. Choix de développement

On a choisi de d'utiliser MERN pour le développement est une excellente décision, car cette technologie est populaire et puissante pour la création d'applications web modernes. MERN est un acronyme qui représente les technologies suivantes :



1. MongoDB : Une base de données NoSQL orientée documents, très adaptée pour stocker des données JSON.
2. Express : Un framework JavaScript pour le développement d'applications web côté serveur basées sur Node.js. Express facilite la création de routes, la gestion des requêtes et des réponses, ainsi que la configuration du serveur.
3. React : Une bibliothèque JavaScript pour la création d'interfaces utilisateur interactives et réactives. React permet de construire des composants réutilisables qui facilitent la création d'expériences utilisateur dynamiques.
4. Node.js : Un environnement d'exécution JavaScript côté serveur qui permet d'utiliser JavaScript pour créer des applications web côté serveur.

En combinant ces quatre technologies, MERN offre une stack complète pour le développement web full-stack en JavaScript. Cela permet une meilleure cohérence du code entre le côté client et le côté serveur, ce qui peut accélérer le processus de développement.

---

## 1. Choix des langages



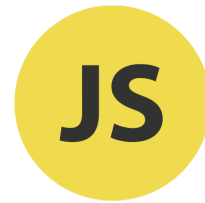
Nous avons fait le choix d'utiliser les langages suivants : ReactJS, React Native et Node.js est cohérent et complémentaire pour le développement d'applications web et mobiles complètes et réactives.

1. ReactJS : React est une bibliothèque JavaScript pour la création d'interfaces utilisateur interactives et réactives côté client. Avec React, vous pouvez construire des composants réutilisables qui facilitent la création d'une expérience utilisateur fluide et dynamique pour les applications web.
2. React Native : React Native est une extension de React qui permet de développer des applications mobiles natives pour iOS et Android en utilisant JavaScript. Grâce à React Native, vous pouvez partager une grande partie du code entre les plateformes tout en offrant une expérience utilisateur proche de celle des applications natives.
3. Node.js : Node.js est un environnement d'exécution côté serveur basé sur JavaScript. Il permet d'utiliser JavaScript pour développer des applications web côté serveur. Node.js est connu pour sa rapidité et son évolutivité, ce qui le rend idéal pour gérer des applications en temps réel et des API.

Ces langages sont tous basés sur JavaScript, ce qui permet une expérience de développement unifiée, car l'équipe n'a besoin de maîtriser qu'un seul langage pour l'ensemble du projet.

---

## 2. Choix des frameworks



Nous avons fait le choix d'utiliser le framework Express est une décision solide pour le développement d'applications web côté serveur basées sur Node.js. Express est l'un des frameworks les plus populaires et les plus largement utilisés dans l'écosystème Node.js en raison de sa simplicité, de sa flexibilité et de sa robustesse.

Voici quelques raisons pour lesquelles on a choisie Express :

1. **Simplicité** : Express est conçu pour être simple et facile à utiliser. Sa syntaxe claire et sa mise en œuvre intuitive permettent aux développeurs de démarrer rapidement et de créer des applications sans trop de complexité.
2. **Flexibilité** : Express est très flexible, ce qui signifie que vous pouvez choisir comment structurer votre application et quelles fonctionnalités ou middleware intégrer selon vos besoins spécifiques.
3. **Large adoption et communauté active** : Express est l'un des frameworks les plus populaires, avec une large adoption parmi la communauté des développeurs. Cela signifie qu'il existe de nombreuses ressources, des tutoriels, des plugins et des middleware tiers disponibles pour faciliter le développement.
4. **Middleware** : L'une des forces d'Express réside dans son système de middleware. Les middleware sont des fonctions qui peuvent être utilisées pour effectuer des tâches telles que l'authentification, la gestion des erreurs, la journalisation, etc. Ils offrent une flexibilité supplémentaire pour personnaliser le comportement de votre application.
5. **Performance** : Express est bien connu pour ses performances élevées, ce qui en fait un choix approprié pour les applications en temps réel et les API.
6. **Large écosystème** : Express est soutenu par un écosystème riche qui comprend de nombreux modules, bibliothèques et outils complémentaires pour étendre ses fonctionnalités.

**Express** : Express est un framework JavaScript pour Node.js qui simplifie le développement d'applications web côté serveur. Il fournit des fonctionnalités de routage, de gestion des requêtes et des réponses, ainsi que la possibilité d'utiliser des middleware pour gérer des tâches spécifiques comme l'authentification et l'autorisation.

### 3. Logiciels et autres outils



Nous avons fait le choix d'utiliser les outils suivants qui sont pertinents pour le développement et la gestion de projets numériques. Chacun de ces outils offre des fonctionnalités uniques et complémentaires pour faciliter le travail d'équipe, la gestion de projets et la conception d'interfaces utilisateur.

#### avantages de chaque outil :

1. **GitHub** : GitHub est un outil essentiel pour la collaboration dans le développement de logiciels. Il permet de gérer le code source de manière centralisée, de suivre les changements, d'effectuer des révisions de code (pull requests), de gérer les problèmes et les demandes de fonctionnalités, et de faciliter la collaboration entre les développeurs. GitHub offre une plateforme robuste pour la gestion du code, ce qui facilite le travail en équipe et la contribution à des projets open source.
2. **Trello** : Trello est un outil de gestion de projet basé sur des tableaux Kanban qui permet d'organiser et de suivre les tâches de manière visuelle. Il offre une vue d'ensemble de l'état d'avancement du projet, permettant à l'équipe de suivre les tâches en cours, les tâches terminées et celles en attente. Trello est idéal pour la planification des projets, le suivi des tâches et la collaboration entre les membres de l'équipe.
3. **Figma** : Figma est un outil de conception et de prototypage d'interfaces utilisateur collaboratif basé sur le cloud. Il permet aux concepteurs de travailler ensemble en temps réel, de créer des maquettes de conception interactives et de partager facilement des conceptions avec les parties prenantes. Figma offre une approche fluide de la conception et permet de collaborer efficacement pour créer des interfaces utilisateur attrayantes et conviviales.



4. **Canva** : Canva est un outil de conception graphique en ligne convivial, idéal pour créer des graphiques, des présentations, des publications sur les réseaux sociaux, des affiches et bien plus encore. Avec ses modèles prédéfinis et ses bibliothèques d'images, Canva facilite la création de contenus visuels de qualité, même pour les personnes n'ayant pas de compétences en conception.

En combinant ces outils, nous avons pu bien travailler en collaboration, gérer les projets de manière organisée et créer des interfaces utilisateur et des contenus visuels attrayants. Cette combinaison complète vous permettra de gagner du temps, d'améliorer la communication et la productivité, et de fournir des résultats de haute qualité dans le développement et la gestion de vos projets numériques.

## 2. Organisation du projet

Pour l'organisation du projet nous nous sommes organisés de manière suivante :

Chacune de nous a d'abord essayé de prendre en main les nouvelles technologies pour pouvoir comprendre et apprendre cette nouvelle technologie.

Lorsque nous avons adopté cette nouvelle **stack** technologique.

Pour nous organiser nous avons utilisé des outils tels que **GitHub** et **Trello** pour faciliter notre travail **collaboratif** et améliorer notre productivité.

Nous avons également utilisé **Figma** pour la **conception** et le **prototypage**, ce qui nous a permis de collaborer de manière fluide sur l'aspect visuel de notre projet.

**GitHub** a été notre principal outil pour la gestion du code source. Nous avons créé un dépôt commun pour le projet, où nous avons une branche "main" représentant la version stable du code.

Chacun de nous avait également sa propre branche de développement, nommée "dev", où nous pouvions travailler individuellement sur des fonctionnalités spécifiques ou des corrections sans perturber le code principal.

Pour organiser notre travail et suivre notre progression, nous avons utilisé Trello. Nous avons créé un tableau Trello dédié au projet, avec des listes représentant différentes étapes du développement telles que "à faire", "en cours" et "terminé".

Chaque tâche à accomplir était représentée par une carte dans le tableau, et nous pouvions les déplacer d'une liste à une autre au fur et à mesure de notre avancement.

Cela nous a permis de garder une vue d'ensemble de toutes les tâches et de nous assurer que rien n'était oublié.

Concernant notre environnement de développement, nous avons utilisé une nouvelle stack technologique pour ce projet.

Nous avons choisi **MongoDB** pour la gestion de la base de données, **React Native** et **Expo** pour le développement mobile, et **React.js** avec **Express** pour le développement Web. Pour interagir avec MongoDB, nous avons utilisé Mongoose pour faciliter la modélisation des données et les opérations **CRUD**.

Malgré que nous étions que deux dans l'équipe, nous avons réussi grâce à une collaboration étroite et une communication régulière.

En utilisant les outils GitHub et Trello, ainsi que la nouvelle stack technologique, nous avons développé efficacement notre projet tout en apprenant et progressant ensemble.

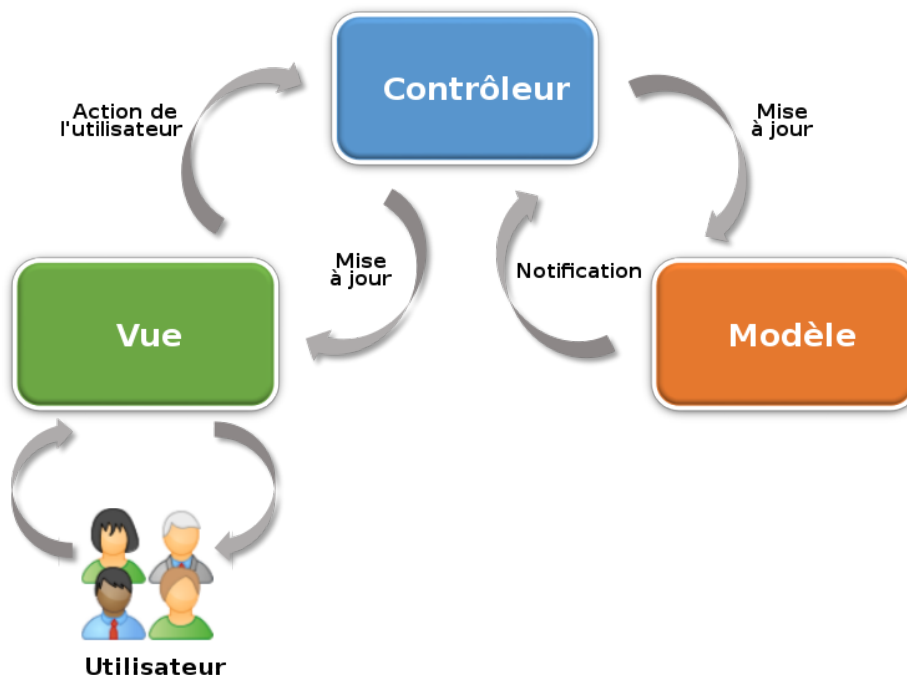
Notre approche collaborative nous a permis de surmonter les défis et de répondre aux attentes de l'équipe et des utilisateurs.

Nous avons pu travailler de manière autonome tout en partageant nos connaissances sur GitHub et en coordonnant nos efforts via Trello. Cette approche nous a permis de progresser rapidement dans l'apprentissage de la nouvelle technologie tout en maintenant une communication et une coordination efficaces au sein de l'équipe.

### 3. Architecture logicielle

Nous avons fait le choix de partir sur une architecture logicielle pour faire de manière propre, on a choisi de faire du MVC (Model,View,Controller) en séparant les différents composants du logiciel. Les composants sont organisés de manière à qu'ils interagissent entre eux et communiquent pour accomplir les tâches spécifiques du programme.

L'objectif principal de l'architecture logicielle est de créer une base solide pour le développement, la maintenance et l'évolutivité du logiciel.



---

**La Scalabilité** : D'une bonne architecture logicielle permet à l'application de s'adapter facilement aux changements de charge et de croissance en termes de volume d'utilisateurs ou de données.

**Dans l'architecture logicielle**, on cherche à bien séparer les différentes parties du logiciel (par exemple, la partie qui gère les règles métier de celle qui s'occupe de l'interface utilisateur). Cela permet de rendre le code plus clair et plus facile à comprendre, ce qui facilite aussi sa maintenance.

**Modularité** : L'architecture logicielle favorise la conception modulaire du logiciel, ce qui permet de réutiliser des composants et de faciliter le travail collaboratif entre les développeurs.

Une architecture logicielle bien conçue est essentielle pour le succès et la durabilité d'un projet logiciel. Elle aide à éviter les problèmes de maintenance, à améliorer la qualité du code et à faciliter l'ajout de nouvelles fonctionnalités tout au long du cycle de vie du logiciel.

# CONCEPTION DU FRONT-END DE L'APPLICATION

## 1. Arborescence du projet

L'arborescence du projet est l'organisation hiérarchique des dossiers et fichiers au sein du projet. Elle représente la structure du projet, où différents fichiers sont regroupés dans des dossiers en fonction de leur fonctionnalité et de leur rôle. Une arborescence bien organisée facilite la gestion, la compréhension et la maintenance du projet, en permettant aux développeurs de trouver rapidement les fichiers dont ils ont besoin.

```
2
3  application_mobile_front/
4  ├── assets/
5  │   ├── images/
6  │   └── fonts/
7  ├── config/
8  │   ├── config.js
9  │   └── firebaseConfig.js
10 ├── src/
11 │   ├── client/
12 │   │   └── client/
13 │   ├── components/
14 │   │   ├── Header.js
15 │   │   ├── Logo.js
16 │   │   ├── Message.js
17 │   │   ├── NewMessageForm.js
18 │   │   ├── SendNewMessage.js
19 │   │   ├── TabNavigator.js
20 │   │   └── ...
21 │   ├── context/
22 │   │   ├── AuthContext.js
23 │   │   └── ...
24 │   ├── routes/
25 │   │   ├── index.js
26 │   │   └── ...
27 │   ├── screen/
28 │   │   ├── UsersScreen.js
29 │   │   ├── RegisterScreen.js
30 │   │   ├── LoginScreen.js
31 │   │   └── ...
32 │   └── App.js
33 ├── package.json
34 ├── package-lock.json
35 ├── .env
36 ├── .gitignore
37 └── README.md
```

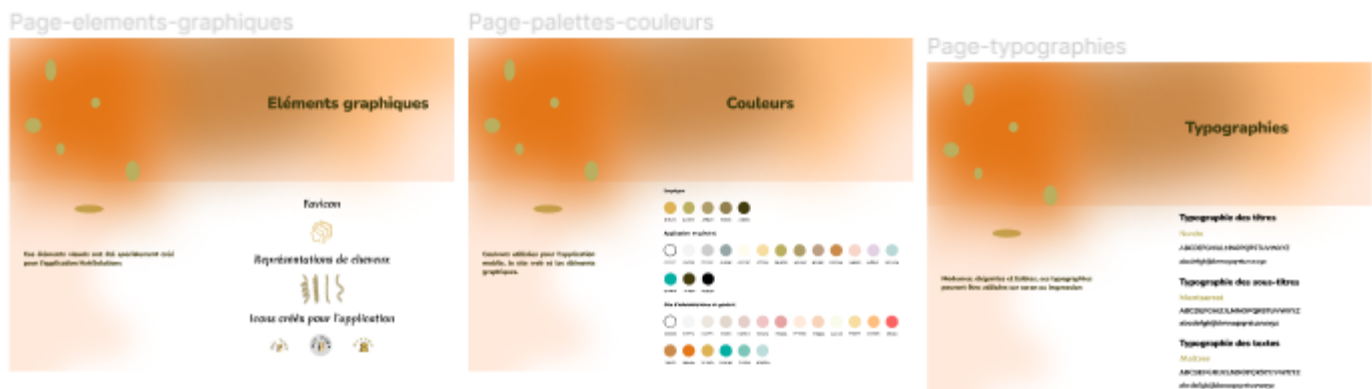
1. **\*\*assets/\*\*** : Ce répertoire contient les ressources statiques utilisées par l'application, telles que des images et des polices.

- 
2. **\*\*config/\*\*** : Ce répertoire contient les fichiers de configuration de l'application. Plus précisément :
    - **\*config.js\*** : Ce fichier contient probablement des paramètres de configuration généraux de l'application.
    - **\*firebaseConfig.js\*** : Il s'agit probablement de la configuration nécessaire pour se connecter et interagir avec Firebase, un service de développement d'applications mobiles.
  3. **\*\*src/\*\*** : Le répertoire principal de code source de l'application.
    - **\*\*client/\*\*** : Ce répertoire contient des fichiers associés au client de l'application. Cela pourrait inclure des modules de communication avec des serveurs distants ou des services externes.
    - **\*\*components/\*\*** : Ce répertoire regroupe les différents composants réutilisables de l'interface utilisateur de l'application. Chaque composant est généralement défini dans un fichier séparé pour faciliter la maintenance et la réutilisation du code.
    - **\*\*context/\*\*** : Ici, on retrouve probablement des fichiers liés à la gestion du contexte de l'application, ce qui peut inclure des données partagées entre différents composants.
    - **\*\*routes/\*\*** : Ce répertoire contient les fichiers liés à la gestion des routes de l'application, permettant de définir les chemins (URL) associés à chaque écran.
    - **\*\*screen/\*\*** : Les écrans de l'application sont généralement regroupés dans ce répertoire. Chaque fichier représente une vue spécifique de l'application.
  4. **\*\*App.js\*\*** : Ce fichier est généralement le point d'entrée de l'application. Il peut être utilisé pour définir la structure de l'application, configurer des routes et des contextes, et gérer l'état global de l'application.
  5. **\*\*package.json\*\*** et **\*\*package-lock.json\*\*** : Ces fichiers sont spécifiques à Node.js et sont utilisés pour décrire les dépendances du projet ainsi que leurs versions spécifiques. Ils permettent également de définir des scripts pour automatiser certaines tâches liées au développement et au déploiement.
  6. **\*\*env\*\*** : Ce fichier est souvent utilisé pour stocker des variables d'environnement qui peuvent être utilisées dans l'application. Cela permet de séparer les données sensibles (par exemple, des clés d'API) du code source.
  7. **\*\*gitignore\*\*** : Ce fichier spécifie les fichiers et dossiers à ignorer lors de la gestion des versions avec Git. Il est utilisé pour exclure les fichiers générés ou les fichiers contenant des données sensibles du contrôle de version.
  8. **\*\*README.md\*\*** : Un fichier Markdown contenant des informations sur le projet, son but, sa configuration, et des instructions pour le développement et le déploiement.

En résumé, l'arborescence du projet sert à organiser et structurer le code, les ressources et la configuration de l'application de manière cohérente. Une bonne organisation facilite le développement, la maintenance, la collaboration entre les membres de l'équipe et le partage du code avec la communauté.

## 2. Charte graphique

La charte graphique, également appelée charte graphique d'entreprise, est un ensemble de règles et de lignes directrices visuelles qui définissent la conception de base de données est un ensemble d'étapes qui aident à créer, mettre en œuvre et maintenir les systèmes de gestion de données d'une entreprise. l'identité visuelle d'une marque, d'une entreprise ou d'une organisation. Elle constitue un document de référence pour la conception graphique de tous les supports de communication de l'entité concernée, que ce soit en ligne ou hors ligne. Pour faire la charte graphique on a utilisé l'outil Figma.



La charte graphique définit les règles pour les éléments visuels de l'entreprise :

- Le logo, utilisé de manière cohérente sur tous les supports, avec différentes versions autorisées.
  - Les couleurs officielles de la marque, avec des utilisations spécifiques pour les éléments graphiques.
  - Les polices de caractères à utiliser pour les titres, les sous-titres et le texte, pour une identité visuelle cohérente.
  - Les directives sur l'utilisation des images, icônes, illustrations et photographies.
  - Des lignes directrices pour la mise en page, la hiérarchie visuelle et l'utilisation d'une grille.
  - Les icônes et pictogrammes spécifiques peuvent être définis.
1. - Comment les éléments visuels doivent être utilisés sur différents supports de communication.

### 3. Maquettage

Le maquettage a pour objectif de visualiser et valider rapidement les idées de conception, l'architecture et les fonctionnalités de l'interface utilisateur. Il permet une collaboration efficace entre concepteurs, développeurs et parties prenantes pour avoir une compréhension commune du produit final.



Les maquettes sont créées simplement avec des outils de conception graphique pour définir la disposition des éléments, les interactions et la navigation. Elles n'incluent généralement pas de détails visuels.

Les maquettes permettent de définir la hiérarchie visuelle en organisant les éléments pour guider l'utilisateur dans sa navigation. On peut définir la taille, la position et les relations entre les éléments.

Elles peuvent inclure des interactions simples pour montrer le flux de navigation et l'expérience utilisateur, comme des liens entre les pages ou des boutons interactifs.

Les maquettes permettent une validation rapide des idées de conception avant le développement, ce qui économise du temps et des ressources en évitant des modifications coûteuses plus tard.

Elles sont utilisées comme base pour créer le design final et facilitent la communication des idées d'interface au sein de l'équipe et avec les parties prenantes et les clients.

---

# CONCEPTION DU BACK-END DE L'APPLICATION

## 1. La base de données

### 1. Concevoir une base de données

L'objectif principal de la conception d'une base de données est de produire des modèles physiques et logiques de conception pour le système de base de données proposé.

### 2. Mise en place une base de données

Pour mettre en place une base de données pour un chat forum, nous avons conçu un schéma de données qui permettra de stocker les messages des utilisateurs, les informations des utilisateurs et d'autres détails pertinents.

### 3. Modélisation de la base de données

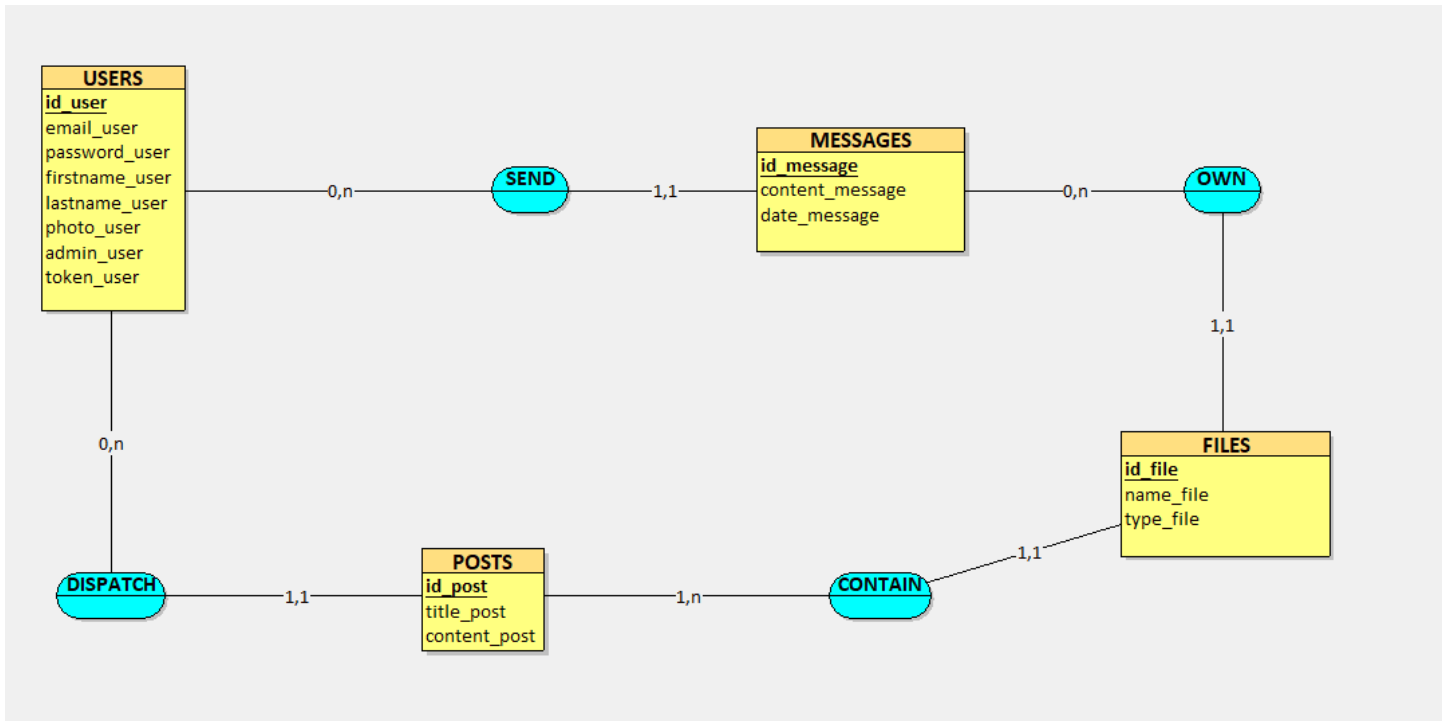
MERISE est une méthode structurée et hiérarchique qui facilite la compréhension et la communication entre les différents acteurs impliqués dans le développement de systèmes d'information. Elle est largement utilisée dans les projets de développement informatique, en particulier dans les contextes d'entreprise où la gestion des données est cruciale.

Méthode MERISE en bref :

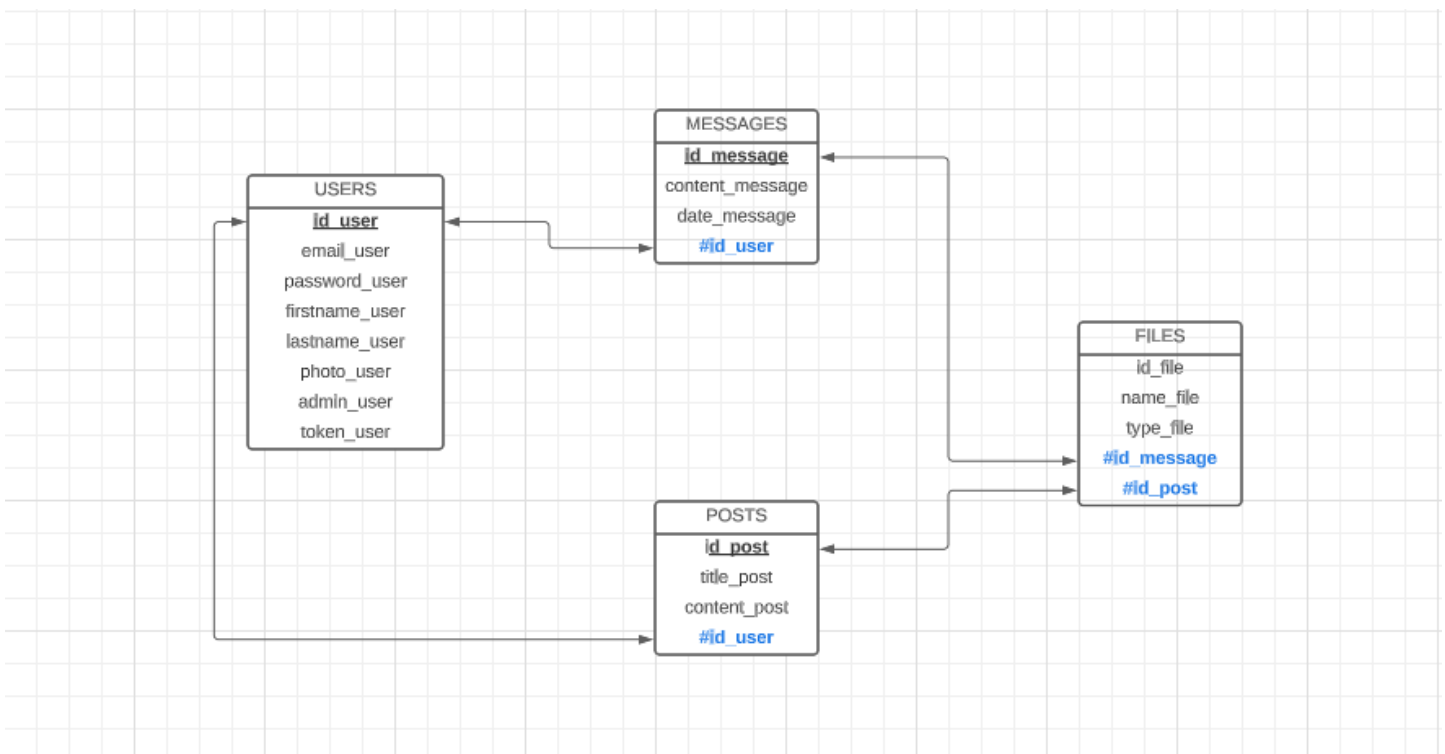
La méthode MERISE se compose de trois niveaux de modélisation :

1. **\*\*Modèle Conceptuel des Données (MCD) :\*\*** C'est la première étape où l'on identifie les principales entités du domaine d'application et les relations entre elles. On utilise généralement des diagrammes entité-association pour représenter ces entités et leurs liens.





2. **\*\*Le Modèle Logique de Données (MLD)** est une étape intermédiaire entre le Modèle Conceptuel de Données (MCD) et le Modèle Physique de Données (MPD) dans le processus de conception d'une base de données. Le MLD est une représentation plus concrète des données, où les entités du MCD sont transformées en tables et les relations en contraintes de clé étrangère.



3. **\*\*Modèle Physique des Données (MPD) :\*\*** Cette dernière étape consiste à transformer le MCD en un modèle physique, spécifique à un système de gestion de base de données (SGBD) donné. On définit les tables, les colonnes, les clés primaires et étrangères, et les contraintes pour représenter la structure concrète de la base de données.

#### SCREEN MPD

Finalement, pour ma conception dans MongoDB en NoSQL, j'ai converti les entités en collection, les tables en documents et les relations en références. avec mongoose

#### SCREEN mongodb

Quelque exemple de collections :

Users : Cette collection stockera les informations relatives aux utilisateurs du chat forum.

- ID (identifiant unique)
- Nom d'utilisateur (string)
- Mot de passe (hashé)
- Adresse e-mail (string)
- Date d'inscription (timestamp)
- Autres informations de profil (âge, photo de profil, etc.)

Messages : Cette collection stockera les messages envoyés par les utilisateurs dans les différentes salles de chat.

- ID (identifiant unique)
- Contenu du message (string)
- ID de l'utilisateur (référence à un utilisateur dans la collection "Utilisateurs")
- ID de la salle de chat (référence à une salle de chat dans la collection "Salles de chat")

Sécurité :

Nous devons mettre en place un système d'authentification pour les utilisateurs afin de sécuriser l'accès à la base de données et garantir que seuls les utilisateurs enregistrés peuvent envoyer des messages et participer aux discussions.

Dénormalisation (en option) :

En fonction des besoins en performances, nous pouvons envisager de dénormaliser certaines données pour éviter les opérations de jointure coûteuses, par exemple, en stockant le nom d'utilisateur dans la collection "Messages" au lieu de faire référence à la collection "Utilisateurs" à chaque fois que nous avons besoin du nom d'utilisateur.

---

## 2. DÉVELOPPEMENT DU BACKEND DE L'APPLICATION

Le développement du backend d'une application est une partie essentielle du processus de création d'une application informatique. Le backend est responsable de la gestion des données, de la logique métier et de la communication entre le serveur et le client. Il constitue le cœur de l'application, permettant au frontend (interface utilisateur) de communiquer avec les bases de données, les services externes et d'exécuter les fonctionnalités spécifiques de l'application.

### 1. Organisation

Mon back-end a pour but d'être utilisé à la fois sur mon application web et mobile

Une API (Interface de Programmation d'Application) offre de nombreux avantages pour le développement des applications. Voici pourquoi elle est utile :

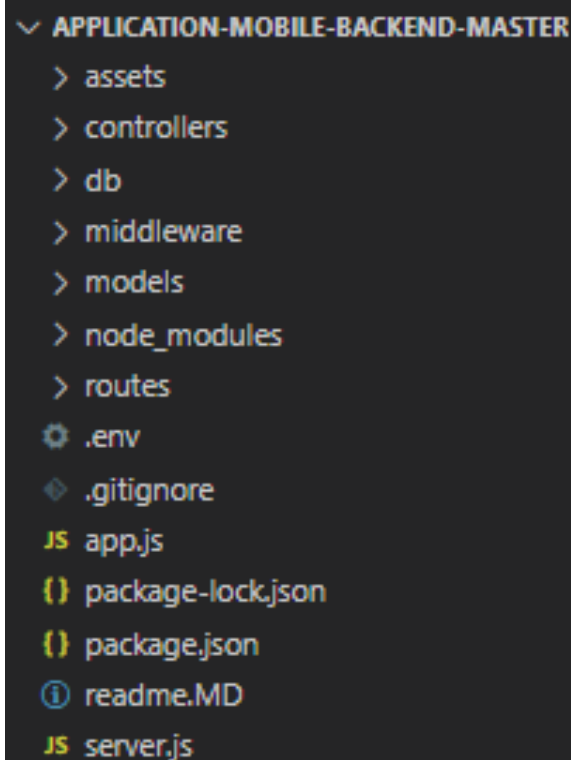
1. **\*\*Séparation des tâches\*\*** : Une API sépare les parties du programme qui gèrent les données (backend) de celles qui gèrent l'affichage (frontend). Cela facilite la collaboration entre les équipes de développement.
2. **\*\*Communication facile\*\*** : Une API permet à différentes applications de se parler. Cela rend l'intégration avec d'autres services plus simples.
3. **\*\*Réutilisation du code\*\*** : L'API permet de réutiliser certaines fonctions dans plusieurs endroits de l'application ou même dans d'autres applications, ce qui économise du temps et des efforts.
4. **\*\*Gestion de la demande\*\*** : Une API bien conçue permet de faire évoluer l'application plus facilement, en ajoutant des ressources supplémentaires au besoin.
5. **\*\*Compatibilité avec tous les appareils\*\*** : L'utilisation d'une API permet aux applications d'être utilisées sur différents appareils et plateformes.
6. **\*\*Sécurité\*\*** : Une API contrôle qui peut accéder aux données et aux fonctions de l'application, protégeant ainsi les informations sensibles.
7. **\*\*Facilité de mise à jour\*\*** : Les mises à jour du backend peuvent être effectuées sans perturber l'interface utilisateur.
8. **\*\*Collaboration aisée\*\*** : L'utilisation d'une API définit clairement les règles à suivre, ce qui facilite la communication entre les équipes.

Une API rend l'application plus flexible, sécurisée et facile à améliorer, en permettant à différentes parties de l'application de communiquer entre elles et de s'intégrer avec d'autres services.

## 2. Arborescence

J'ai choisi d'utiliser L'architecture n-tiers, est une méthode pour organiser une application en différentes parties, chacune ayant un rôle spécifique. Chaque partie accomplit certaines tâches et communique avec les autres parties de manière bien définie.

Cette organisation rend l'application plus ordonnée, facile à entretenir et à faire évoluer. Chaque partie est autonome, ce qui permet aux développeurs de travailler sur des parties spécifiques sans perturber le reste de l'application. Cela facilite aussi la collaboration entre les équipes de développement, car chacune peut se concentrer sur sa propre partie sans interférer avec les autres.



```

▼ APPLICATION-MOBILE-BACKEND-MASTER
  > assets
  > controllers
  > db
  > middleware
  > models
  > node_modules
  > routes
  ⚙ .env
  💎 .gitignore
  JS app.js
  {} package-lock.json
  {} package.json
  ⓘ readme.MD
  JS server.js

```

---

### 3. Fonctionnement de l'API

L'API (Interface de programmation d'application) d'un chat forum est une interface qui permet aux développeurs de créer et de gérer des fonctionnalités spécifiques d'un forum de discussion en utilisant du code. Elle définit les règles et les méthodes pour communiquer avec le forum et accéder à ses fonctionnalités.

Voici comment fonctionne l'API d'un chat forum :

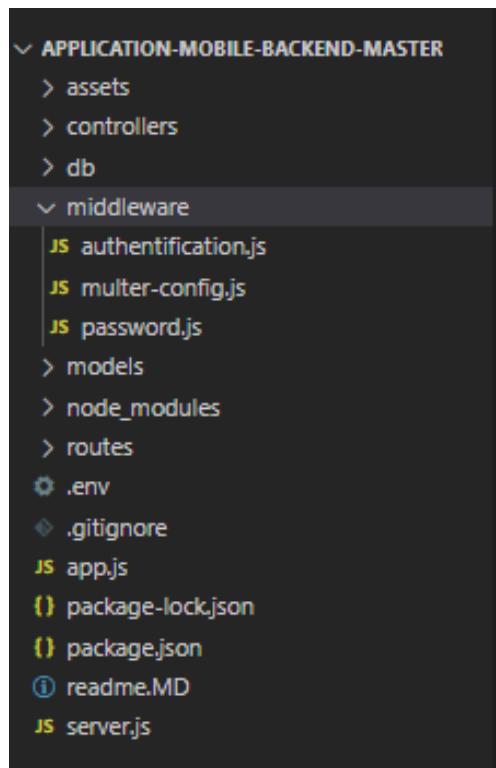
1. **Authentification** : Avant d'accéder aux fonctionnalités du forum, les utilisateurs doivent s'authentifier en fournissant leurs identifiants (par exemple, nom d'utilisateur et mot de passe) à l'API. Cela permet de vérifier leur identité et de les autoriser à utiliser les différentes fonctionnalités du forum.
2. **Récupération des données** : Une fois authentifiés, les utilisateurs peuvent utiliser l'API pour récupérer les données du forum, telles que les messages, les discussions, les utilisateurs en ligne, etc. L'API fournira les données sous une forme structurée, souvent au format JSON (JavaScript Object Notation) ou XML (eXtensible Markup Language).
3. **Envoi de messages** : Les utilisateurs peuvent utiliser l'API pour envoyer de nouveaux messages ou répondre à des discussions existantes. L'API prendra en charge la réception de ces messages et les enregistrera dans la base de données du forum.
4. **Gestion des discussions** : Les utilisateurs peuvent également créer de nouvelles discussions, les modifier ou les supprimer en utilisant l'API. Cela permet de gérer l'organisation et la structure du forum.
5. **Gestion des utilisateurs** : L'API permettra également aux administrateurs du forum de gérer les comptes des utilisateurs, tels que la création de nouveaux comptes, la mise à jour des informations de profil et la gestion des autorisations d'accès.
6. **Notifications** : L'API peut également prendre en charge les notifications, en informant les utilisateurs des nouvelles activités sur le forum, telles que les messages reçus, les réponses à leurs messages ou les modifications apportées aux discussions auxquelles ils participent.
7. **Sécurité** : L'API doit être conçue de manière sécurisée pour empêcher tout accès non autorisé aux données sensibles du forum. Cela peut inclure l'utilisation de jetons d'authentification, de mécanismes de chiffrement et de vérification des autorisations d'accès.

En résumé, l'API d'un chat forum permet aux développeurs d'interagir avec le forum de discussion de manière programmable, en accédant aux données, en envoyant des messages, en gérant les discussions et les utilisateurs, le tout de manière sécurisée. Cela facilite l'intégration du forum dans

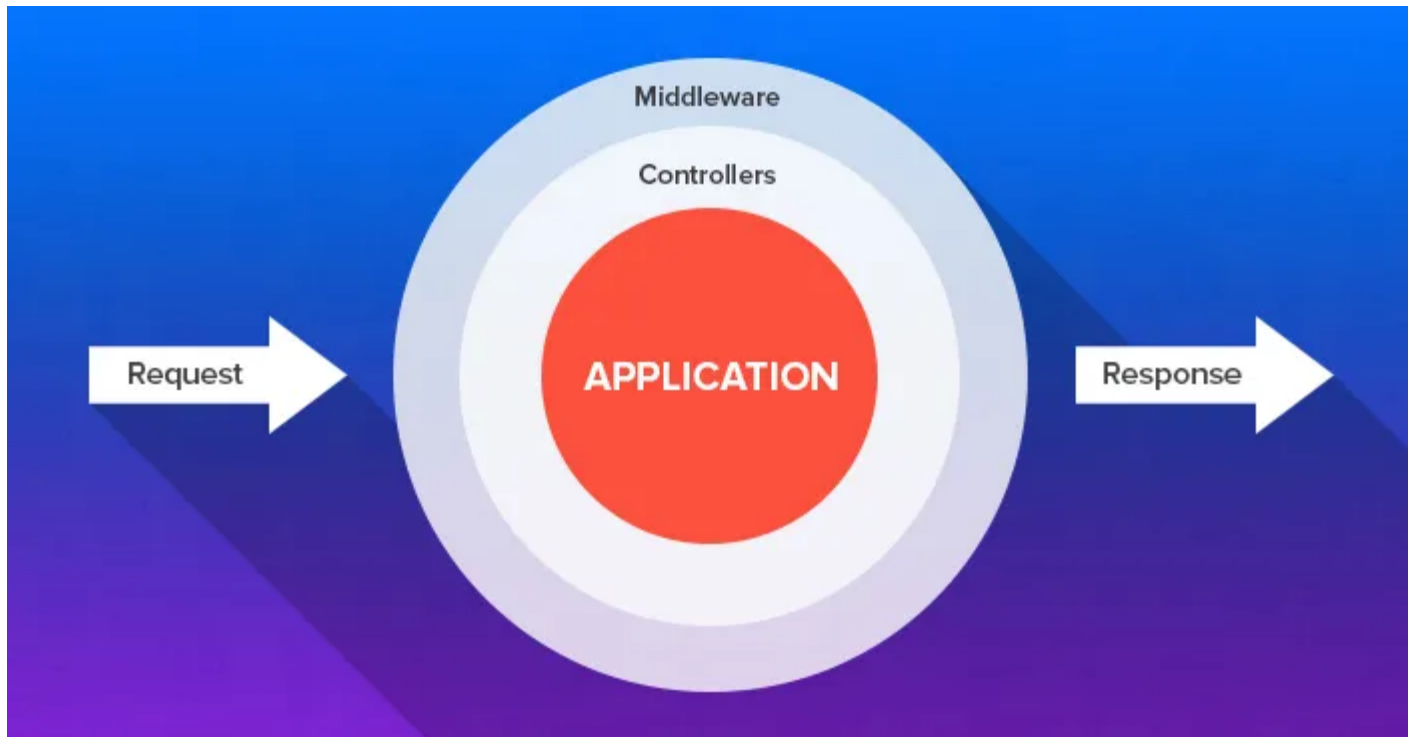
d'autres applications ou plateformes et offre une plus grande flexibilité pour personnaliser les fonctionnalités selon les besoins spécifiques des utilisateurs.

## 4. Middleware

On a utilisé des middleware entre différentes parties d'un système informatique. Il facilite leur communication en gérant les tâches importantes telles que la sécurité, la gestion des erreurs et des données en mémoire, et la coordination des sessions. En gros, il assure que toutes les parties du système peuvent travailler ensemble de manière harmonieuse, ce qui rend le système plus organisé, souple et capable de s'adapter aux évolutions.



### Fonctionnement du middleware:



## 5. Routage

Les routes permettent la communication entre une requête d'un client vers un serveur est le processus de détermination de la destination appropriée

```
18 // Routes de l'API
19 // Accès à l'application
20 // Route pour s'inscrire
21 router.post("/signup", password, ctrlUsers.signup);
22 // Route pour se connecter
23 router.post("/login", ctrlUsers.login);
24 // Route pour se déconnecter
25 router.post("/logout", auth, ctrlUsers.logout);
26
27 // Profil utilisateur
28 // Route pour voir ses informations personnelles
29 router.get("/profil", auth, ctrlUsers.getProfil);
30 // Route pour modifier ses informations personnelles
31 router.put("/profil", auth, imageMulter, ctrlUsers.updateProfil);
32 // Route pour modifier ses informations personnelles
33 router.delete("/profil", auth, ctrlUsers.deleteProfil);
34
```

## 6. Controller

Un "controller" dans une API est comme un "chef d'orchestre" qui écoute les demandes des utilisateurs, les traite en fonction de ce qui est demandé, récupère les informations nécessaires et renvoie une réponse claire et organisée. Il joue un rôle essentiel pour rendre l'API utile et permettre aux utilisateurs d'obtenir les informations ou les services qu'ils souhaitent de manière simple et efficace.

```

2 // Fonction pour l'inscription
3 exports.signup = (req, res) => {
4   // Chiffrement l'email
5   const emailcryptoJS = cryptoJS
6     .HmacSHA256(req.body.email, `${process.env.SECRET_CRYPTOJS_EMAIL}`)
7     .toString();
8   // Hachage du mdp
9   bcrypt
10    .hash(req.body.password, 10)
11    .then((passwordHash) => {
12      // Supprimer l'id du front end car nous l'avons avec mongoDb
13      // delete req.body._id; // Supprimer Id de la bd
14      // delete req.body._userId; // Supprimer Id du client
15
16      const user = new Users({
17        ...req.body, // Copie de tous ce qu'il y a dans les req.body
18        // userId: req.auth.userId,
19        email: emailcryptoJS,
20        password: passwordHash,
21      });
22
23      user
24        .save()
25        .then(() => {
26          res.status(201).json({ message: "Utilisateur créé avec succès." })
27        })
28        .catch((error) => {
29          res.status(401).json({ message: "Utilisateur déjà existant." })
30        });
31    })
32    .catch((error) => res.status(500).json({ error }));
33 };
34

```



## 7. Model

Dans MongoDB, le terme "Model" fait référence à la façon dont les données sont structurées et organisées dans la base de données. MongoDB est une base de données NoSQL de type document, ce qui signifie qu'elle stocke les données sous forme de documents JSON (JavaScript Object Notation). Ce modèle flexible et évolutif facilite le stockage et la récupération de données de différentes structures, ce qui en fait une solution populaire pour les applications qui nécessitent une grande souplesse dans la gestion des données. Exemple :

```
// module.exports = UserModel

const mongoose = require('mongoose');
// const uniqueValidator = require('mongoose-unique-validator');

const Users = mongoose.Schema({
  email: {type: String, require: true, trim: true, unique: true },
  password: {type: String, require: true, trim: true },
  firstname: {type: String, require: true, trim: true, minlength: 2, maxlength: 100},
  lastname: {type: String, require: true, trim: true, minlength: 2, maxlength: 100},
  photo: { type: String, trim: true, default: false },
  admin: {type: Boolean, require: true, default: false },
  // isConnected: {type: Boolean, require: true, default: false },
  token: { type: String, trim: true }
}, { timestamps:true});

// Package pour ne pas avoir le même login lors de l'inscription
// userSchema.plugin(uniqueValidator);

module.exports = mongoose.model('users', Users);
```

## 8. Sécurité

Pour la sécurité dans MongoDB on a créé des Authentification grâce aux middleware, on a défini les rôles, on a crypté le mot de passe et l'adresse mail car c'est un aspect crucial pour protéger les données stockées dans la base de données contre les accès non autorisés.

MongoDB propose plusieurs mécanismes de sécurité pour garantir la confidentialité et l'intégrité des données.

```
middleware > JS authentication.js > <unknown> > exports
1 // Importation du token pour l'authentification
2 const jwt = require('jsonwebtoken');
3 // Importer package pour variables d'environnement
4 require('dotenv').config();
5
6 module.exports = (req, res, next) => {
7   try {
8     // Récupérer le token dans le headers authorization : bearer token
9     // On récupère le token grâce à la methode split de js pour enlever le mot Bearer
10    // qui se trouve au début du token dans le req.headers.authorization
11    const token = req.headers.authorization.split(' ')[1];
12    // Décoder le token
13    const payload = jwt.verify(token, `${process.env.JWT_KEY_TOKEN}`);
14
15    // Objet request qui va être transmit au route appelé par la suite
16    req.auth = {
17      userId: payload.userId,
18      // admin: payload.admin,
19    };
20    next();
21  } catch (error) {
22    res.status(401).json({ error: "JWT Error" });
23  }
24 }
```

Voici les mesures de sécurité de MongoDB :

1. Authentification : MongoDB prend en charge l'authentification des utilisateurs avant qu'ils puissent accéder à la base de données. Les utilisateurs doivent fournir des identifiants (nom d'utilisateur et mot de passe) pour se connecter et accéder aux données. Cela empêche les accès non autorisés à la base de données.
2. Contrôle d'accès basé sur les rôles : MongoDB permet de définir des rôles avec des privilèges spécifiques pour les utilisateurs. Par exemple, un utilisateur peut être autorisé à lire et écrire des données, tandis qu'un autre peut être limité à la lecture seule. Ceci permet de contrôler finement les autorisations d'accès aux différentes parties de la base de données.

---

En mettant en œuvre ces mesures de sécurité, les administrateurs peuvent assurer la protection des données stockées dans MongoDB contre les accès non autorisés et les menaces potentielles, tout en préservant la confidentialité et l'intégrité des informations de l'application.

### 1. Chiffrement des données sensibles

Le chiffrement des données sensibles dans MongoDB est une mesure de sécurité importante pour protéger les informations confidentielles stockées dans la base de données. Il rend les données illisibles sans une clé spéciale, ce qui les protège contre les accès non autorisés.

MongoDB utilise le chiffrement pour sécuriser les données lorsqu'elles sont transférées sur le réseau (chiffrement en transit) et lorsqu'elles sont stockées sur le disque (chiffrement au repos). Le chiffrement garantit que même si quelqu'un accède physiquement au disque, les données restent illisibles sans la clé appropriée.

En utilisant le chiffrement, on s'assure que les données sensibles, comme les informations personnelles ou les mots de passe, sont protégées contre les menaces potentielles, ce qui contribue à renforcer la sécurité globale de l'application.

### 2. JWT

Le chiffrement des données sensibles dans MongoDB est une mesure de sécurité importante pour protéger les informations confidentielles stockées dans la base de données. Il rend les données illisibles sans une clé spéciale, ce qui les protège contre les accès non autorisés. Quand l'utilisateur se connecte, celui-ci génère un token.

### 3. Gestion des Droits

Pour la gestion des droits, on utilise un booléen dans le schéma de l'utilisateur, et du côté de la base de données, c'est nous qui avons choisi d'ajouter une ligne supplémentaire pour l'administrateur avec la valeur true.

## 9. Problématique rencontrée

### Comment créer des collections avec mongoose ?

## 10. Recherches anglophones

### How to create collections with mongoose ?

```
const mapSchema = new Schema({
  name: {
    type: String
  },
  description: {
    type: String
  },
  rate: {
    type: Number,
    default: 0
  },
  sumOfRates: {
    type: Number,
    default: 0
  },
  places: [
    {
      name: String,
      description: String,
      difficulty: { type: String, default: 0 },
      sumOfRates: { type: String, default: 0 },
      lat: Number,
      lng: Number
    }
  ]
}, {
  timestamps: true
})
```

## 11. Documentation aller sur l'annexe ()

---

## 12. Tests

### 1. Postman

Nous avons testé nos routes sur l'outil Postman pour vérifier que les routes fonctionnent correctement et qu'elles récupèrent les bonnes informations. Ces tests nous ont permis de valider le bon fonctionnement de notre système et de s'assurer que les réponses fournies par les routes sont cohérentes et conformes aux attentes.

Grâce à ces tests, nous avons pu identifier et corriger d'éventuels problèmes avant le déploiement, garantissant ainsi une meilleure expérience utilisateur et une application plus fiable.

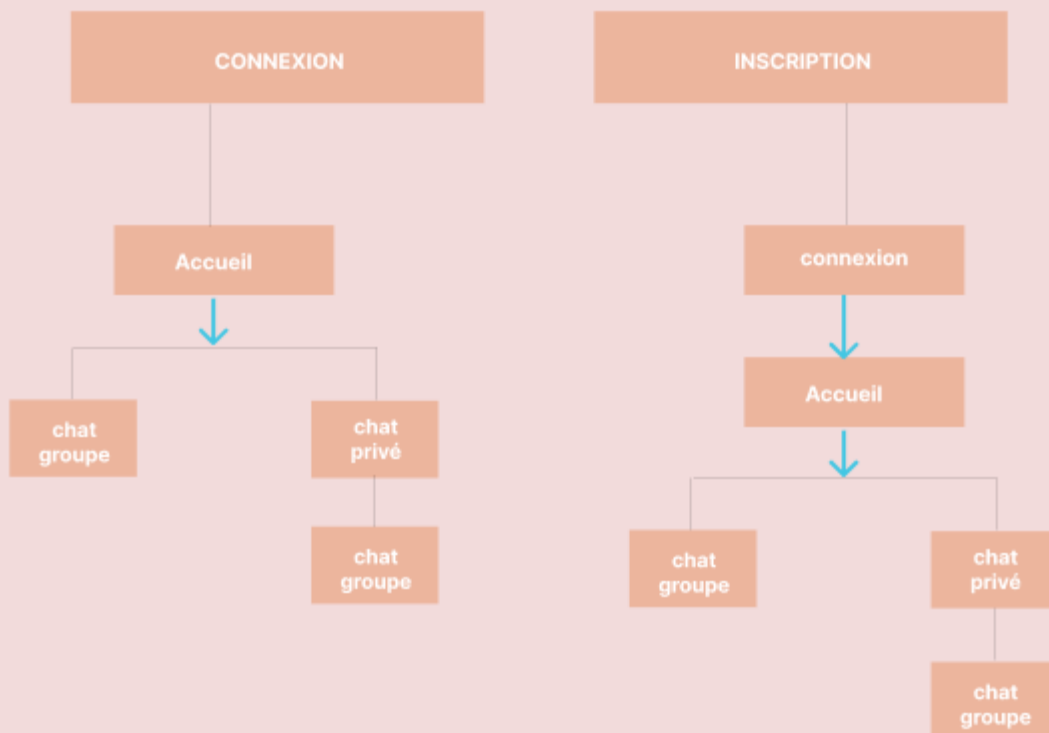
### 2. Newman

Newman est un outil open-source développé par Postman qui fonctionne en ligne de commande. Il est utilisé pour automatiser l'exécution de collections de tests d'API créées dans Postman. Newman permet ainsi de valider automatiquement le bon fonctionnement des API en effectuant les requêtes et en exécutant les tests définis.

Cela garantit que les réponses des API sont conformes aux attentes, ce qui est essentiel pour assurer la fiabilité et la qualité des applications qui utilisent ces API. En résumé, Newman facilite l'automatisation des tests d'API, contribuant ainsi à un développement plus efficace et fiable.

# DÉVELOPPEMENT DU FRONT-END DE L'APPLICATION

## 1. Arborescence



## 2. Pages et composants

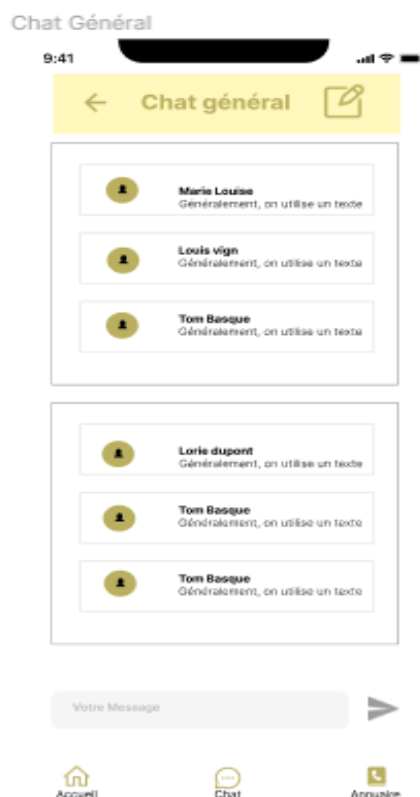
On a créé un composant "header" est généralement la partie supérieure d'une page web, qui contient souvent le logo, le menu de navigation et d'autres éléments essentiels. Voici une explication des avantages et des inconvénients de créer un tel composant :

```

1  import React from "react";
2  import { View, Text, StyleSheet, TouchableOpacity } from "react-native";
3  import { Ionicons } from "@expo/vector-icons";
4
5  const Header = ({ onRetour }) => {
6    return (
7      <View style={styles.header}>
8        <TouchableOpacity onPress={onRetour} style={styles.touchableOpacity}>
9          <Ionicons name="arrow-back" size={24} color="#BAAF5E" />
10        </TouchableOpacity>
11        <Text style={styles.titleHeader}>Retour</Text>
12      </View>
13    );
14  };
15

```

### Résultat du composant côté front-end



### 3. Sécurités :

Pour la sécurisation du côté front on a utiliser des regex pour valider une adresse e-mail et le hachage du mot de passe en React Native :

voici un petit extrait de code

#### 1. **\*\*Validation d'une adresse e-mail avec une regex\*\*** :

```
45 // Vérifications des champs
46 const validateEmail = () => {
47   if (!email) {
48     setEmailError("L'email est requis");
49     return false;
50   }
51   setEmailError("");
52   return true;
53 };
54
55 const validatePassword = () => {
56   if (!password) {
57     setPasswordError("Le mot de passe est requis");
58     return false;
59   }
60   setPasswordError("");
61   return true;
62 };
63
64 // Fonction qui fait appel a l'authentification dans le contexte
65 const loginUser = async () => {
66   if (validateEmail() && validatePassword()) {
67     await authLogin(email, password);
68   }
69 };
70
```



## 2. **Hachage du mot de passe** :

Note : Pour le hachage du mot de passe, il est préférable de réaliser cette opération côté serveur plutôt que côté front-end. Cependant, pour illustrer le processus, voici un exemple côté front-end utilisant une bibliothèque de hachage appelée "bcryptjs" :

```

1  const passwordValidator = require('password-validator');
2
3  const passwordSchema = new passwordValidator();
4
5  passwordSchema
6    .is().min(8)           // Minimum length 8
7    .is().max(100)        // Maximum length 100
8    .has().uppercase()    // Must have uppercase letters
9    .has().lowercase()    // Must have lowercase letters
10   .has().digits(1)       // Must have at least 1 digits
11   .has().not().spaces()  // Should not have spaces
12   .is().not().oneOf(['Passw0rd', 'Password123']); // Blacklist these values
13
14  module.exports = (req, res, next) => {
15    if(passwordSchema.validate(req.body.password)) {
16      next();
17    }
18    else {
19      return res
20        .status(400)
21        .json({error : `Mot de passe pas assez fort ${passwordSchema.validate('req.body.password', { details : true })}`});
22    }
  }

```

## 4. Problématiques rencontrée

C'est la nouvelle syntaxe de react native

---

# CONCEPTION DE L'ESPACE ADMINISTRATEUR

## Conception de la partie administration

Dans la phase de conception, nous allons aborder les principaux aspects de la partie administration et définir les fonctionnalités clés qui garantiront une gestion efficace et sécurisée.

### 1. User Story

Parfait ! Les User Stories sont un moyen populaire et efficace de décrire les fonctionnalités d'un projet ou d'un système du point de vue des utilisateurs. Si vous avez déjà créé une User Story pour la partie administration, voici comment cela pourrait ressembler :

**\*\*User Story** : Gestion des utilisateurs dans la partie administration**\*\***

En tant qu'administrateur système,  
je souhaite pouvoir gérer facilement les comptes des utilisateurs  
afin de garantir un accès sécurisé et personnalisé au système.

**\*\*Critères d'acceptation : \*\***

1. En tant qu'administrateur, je peux créer un nouveau compte utilisateur en fournissant son nom, son adresse e-mail et en définissant un mot de passe sécurisé.
2. Je peux modifier les informations d'un utilisateur existant, y compris son nom, son adresse e-mail et son rôle dans le système.
3. En cas de besoin, je peux réinitialiser le mot de passe d'un utilisateur et lui envoyer les instructions de réinitialisation par e-mail.
4. Lors de la création ou de la modification d'un compte utilisateur, je peux attribuer un rôle spécifique à l'utilisateur parmi les rôles prédéfinis (administrateur, modérateur, utilisateur, etc.).
5. Je peux désactiver un compte utilisateur si nécessaire, ce qui empêchera l'utilisateur de se connecter, tout en conservant ses données dans le système.
6. En tant qu'administrateur, je peux afficher la liste complète des utilisateurs enregistrés avec leurs détails respectifs.
7. Je peux rechercher un utilisateur spécifique par son nom ou son adresse e-mail pour accéder rapidement à son compte.

8. Les mots de passe des utilisateurs doivent être stockés de manière sécurisée en utilisant des méthodes de hachage et de salage.

9. Lorsqu'un utilisateur se connecte, le système doit effectuer une vérification appropriée de son identité pour éviter les accès non autorisés.

## 2. Choix du langage et framework

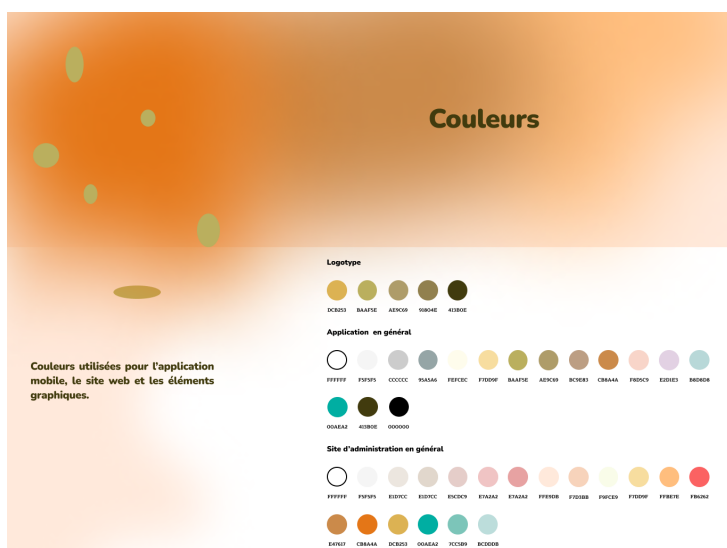
Nous avons fait le choix d'utiliser React comme langage et le choix du framework pour développer un panneau d'administration en React dépend de plusieurs facteurs tels que les compétences de l'équipe de développement, les exigences spécifiques du projet, les performances attendues et les contraintes techniques.

## 3. Conception du frontend du site web

### 1. Charte graphique

La conception du frontend du site web repose sur une charte graphique soigneusement élaborée. Cette charte graphique détermine l'identité visuelle du site et définit les éléments de design qui seront utilisés de manière cohérente sur toutes les pages.

Les couleurs principales de la charte graphique ont été choisies pour refléter l'image de marque de l'entreprise, en utilisant des tons harmonieux qui attirent l'attention tout en créant une ambiance conviviale et professionnelle.

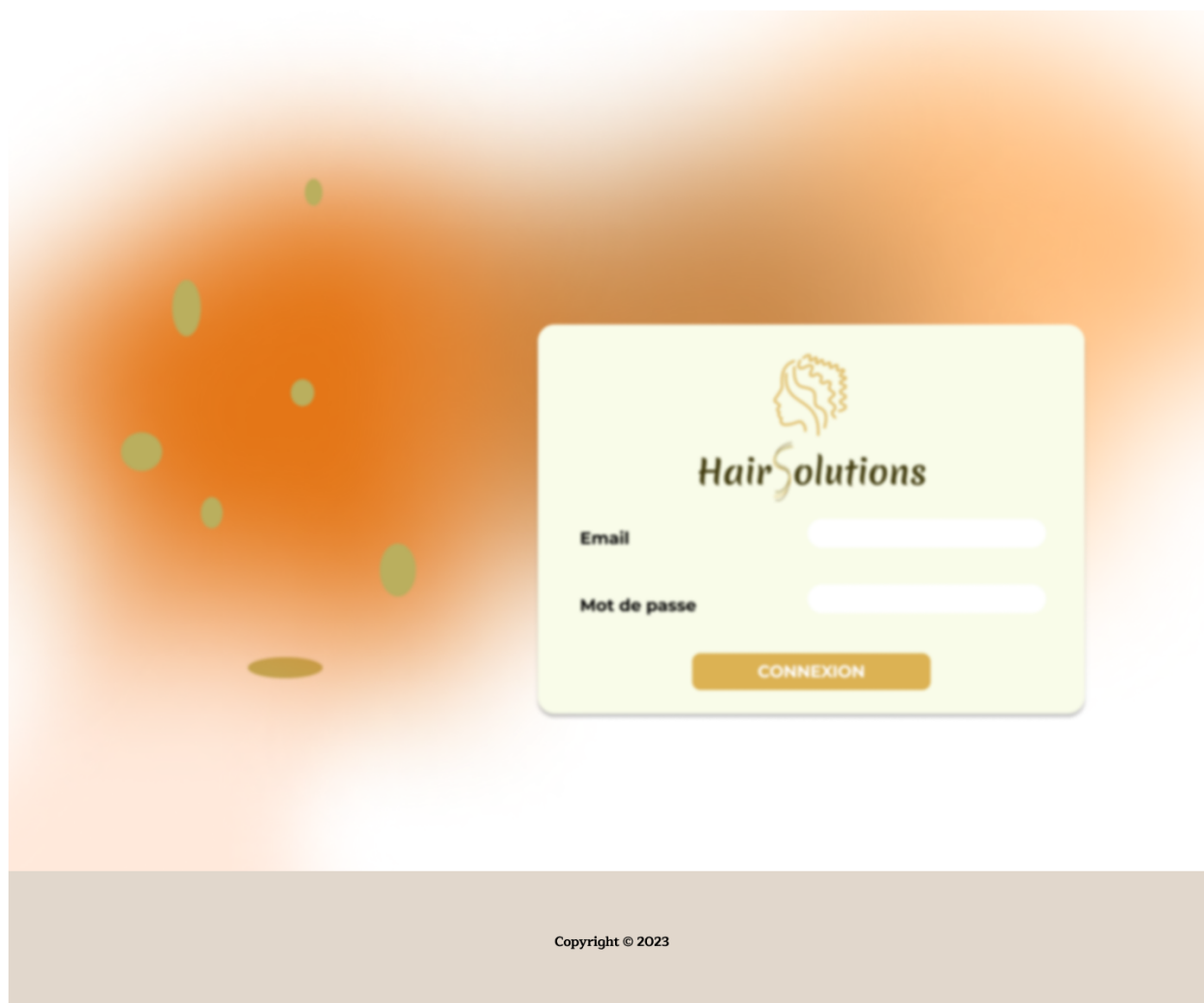


Les typographies sélectionnées assurent une lisibilité optimale, avec une combinaison équilibrée de polices pour les titres, les sous-titres et le contenu afin de rendre la navigation agréable pour les visiteurs.



Grâce à cette charte graphique, le frontend du site web offre une apparence professionnelle, cohérente et attrayante qui reflète l'identité de l'entreprise et favorise une navigation intuitive pour les visiteurs.

## 2. Maquettage



---

## 4. Conception du back end du site web

"Pour la conception du backend de notre projet, nous avons opté pour l'utilisation de la stack MERN, qui combine MongoDB en tant que système de gestion de base de données **NoSQL**, Express.js pour la création d'une API robuste, React pour l'interface utilisateur côté client et Node.js pour la gestion du côté serveur.

La décision d'utiliser la stack **MERN** s'est avérée bénéfique, car elle offre une architecture complète et cohérente pour le développement d'applications web modernes. MongoDB, en tant que base de données NoSQL, nous permet de stocker des données de manière flexible et évolutive, sans avoir à définir de schéma rigide, ce qui facilite l'adaptation aux évolutions futures.

Grâce à Express.js, nous avons pu créer facilement des points **d'API** pour gérer les opérations CRUD (Create, Read, Update, Delete) et répondre aux requêtes du côté client de manière efficace et sécurisée.

L'utilisation de React dans la stack MERN nous a permis de développer une interface utilisateur réactive et interactive, offrant ainsi une expérience utilisateur fluide et agréable.

Enfin, Node.js a été le choix idéal pour gérer les opérations côté serveur, en utilisant le même langage de programmation (JavaScript) pour le frontend et le backend, ce qui simplifie la communication et le partage de logique entre les deux parties de l'application.

Avec la modélisation NoSQL, nous avons pu concevoir une structure de données flexible, adaptée aux besoins changeants du projet, tout en maintenant des performances élevées pour les opérations de lecture et d'écriture.

---

## CONCLUSION

En conclusion, ce dossier de soutenance met en lumière le parcours passionnant que j'ai suivi en tant que concepteur/développeur d'application. Tout au long de ce processus, j'ai acquis des connaissances approfondies en matière de conception et de développement d'applications, ainsi que des compétences techniques variées.

Le dossier présente en détail les différentes étapes du développement d'une application, allant de la phase de conception à la réalisation concrète. J'ai exploré les principaux langages de programmation, les frameworks et les outils essentiels utilisés dans le domaine du développement d'applications, tels que React, Node.js, Express.js et MongoDB.

J'ai également abordé les bonnes pratiques de développement, y compris la gestion de version, les tests automatisés et l'optimisation des performances. Ces aspects sont cruciaux pour assurer la qualité et la fiabilité de chaque projet que j'entreprends.

Le dossier de soutenance a également mis en évidence mon engagement à rester constamment à jour avec les dernières technologies et tendances du secteur. Je me suis impliqué dans des projets réels et j'ai acquis une expérience précieuse grâce à des stages et des projets collaboratifs.

Enfin, ce dossier met en avant ma capacité à travailler en équipe, à communiquer efficacement avec les parties prenantes du projet et à trouver des solutions créatives aux défis rencontrés.

Je suis convaincu que cette formation de concepteur/développeur d'application a solidement préparé le terrain pour ma future carrière dans le domaine de la technologie. Je suis enthousiaste à l'idée de continuer à évoluer dans ce domaine en mettant en pratique les compétences et les connaissances acquises.

Je tiens à exprimer ma gratitude envers mes formateurs, mes mentors et mes camarades de classe qui ont tous contribué à mon apprentissage et à ma croissance personnelle tout au long de ce parcours.

En conclusion, je suis prêt(e) à relever de nouveaux défis et à apporter une contribution significative dans le monde du développement d'applications. Je suis confiant(e) que ma passion pour la technologie et ma détermination à apprendre continueront de me guider vers des réalisations futures exceptionnelles.

---

## ANNEXES

DOCUMENTATION API

CAHIER DES CHARGES