Fic’Tif – Centre médical

Application web de prise de rendez-vous médicaux  
par Nicolas BERNARD

Projet Réalisé dans le cadre de la formation Développeur Web et Web Mobile réalisée à l’organisme de formation SOFIP Rouviginie

Table des matières

[Présentation personnelle 1](#_Toc185341968)

[Mon parcours 1](#_Toc185341969)

[Mon entrée en formation 1](#_Toc185341970)

[Le projet Fic’Tif - Concept 1](#_Toc185341971)

[Introduction 1](#_Toc185341972)

[Services délivrés par l’application 1](#_Toc185341973)

[ Prise de rendez-vous 1](#_Toc185341974)

[ Suivi des rendez-vous 2](#_Toc185341975)

[ Gestion du compte utilisateur 2](#_Toc185341976)

[Besoins fonctionnels 2](#_Toc185341977)

[ Quand je suis patient 2](#_Toc185341978)

[ Quand je suis docteur 3](#_Toc185341979)

[Problématiques techniques 3](#_Toc185341980)

[Solutions envisagées 3](#_Toc185341981)

[Fic’Tif - Conception 3](#_Toc185341982)

[Maquettage de l’application 3](#_Toc185341983)

[ La théorie des couleurs 3](#_Toc185341984)

[ Wireframes 5](#_Toc185341985)

[Mise en place de l’environnement 8](#_Toc185341986)

[ Les repos git 8](#_Toc185341987)

[Mise en place de la base de données 8](#_Toc185341988)

[ MCD 8](#_Toc185341989)

[Les dépendances indispensables 9](#_Toc185341990)

[Création du back-end 10](#_Toc185341991)

[ Technologies employées 10](#_Toc185341992)

[ Création de la base MongoDB 11](#_Toc185341993)

[ Modules et dépendances 11](#_Toc185341994)

[ Structure 12](#_Toc185341995)

[ Création des premières routes 12](#_Toc185341996)

[ Sécurité 17](#_Toc185341997)

[Simulation des routes backend 17](#_Toc185341998)

[ Installation de PostMan 17](#_Toc185341999)

[ Tests des routes backend 17](#_Toc185342000)

[ Premiers peuplements de la base MongoDB 17](#_Toc185342001)

[Création du front-end 17](#_Toc185342002)

[ Technologies employées 17](#_Toc185342003)

[ Structure 17](#_Toc185342004)

[ Création des routes de navigations 17](#_Toc185342005)

[ Création d’un layout 17](#_Toc185342006)

[Fonctionnalités 17](#_Toc185342007)

[ Theme toggler 17](#_Toc185342008)

[ Prise de rendez-vous 17](#_Toc185342009)

[ Login/Logout 17](#_Toc185342010)

[ Dashboard 17](#_Toc185342011)

[ Recherche de médecin 17](#_Toc185342012)

[Tests 17](#_Toc185342013)

[Documentation 17](#_Toc185342014)

[Déploiement 17](#_Toc185342015)

[La trêve de novembre 18](#_Toc185342016)

[Problèmes rencontrés 18](#_Toc185342017)

[MCD 18](#_Toc185342018)

[Conception du backend 18](#_Toc185342019)

[Login/Logout 18](#_Toc185342020)

[Conclusion 18](#_Toc185342021)

[Technologies balayées par le projet 18](#_Toc185342022)

[ Frontend 18](#_Toc185342023)

[ Backend 18](#_Toc185342024)

[Axes d’amélioration 18](#_Toc185342025)

[Et si c’était à refaire ?! 18](#_Toc185342026)

[Bilan 18](#_Toc185342027)

[ Remerciements 18](#_Toc185342028)

[ Le mot de la fin 18](#_Toc185342029)

[Test 18](#_Toc185342030)

[Titre secondaire 18](#_Toc185342031)

[ Titre section 18](#_Toc185342032)

# Présentation personnelle

## Mon parcours

Ayant toujours baigné dans le monde du numérique et de l’informatique, j’ai toujours eu une appétence pour le développement. Lors de mes années d’études, j’ai d’ailleurs entrepris un cursus informatique à l’université du Mont Houy à Valenciennes (DEUST IOSI). J’ai alors découvert de nombreux domaines liés à l’informatique comme le réseau, la programmation et aussi le développement web. Au cours de ce cursus, c’est d’ailleurs ce dernier point qui m’a le plus inspiré quand il été question de réalisé un projet de fin d’étude au technologies et thèmes ouverts.

## Mon entrée en formation

Après une longue période dans le milieu du help desk, j’ai décidé, à l’âge de 34 ans, d’effectuer une reconversion professionnelle. Le milieu dans lequel le réalisé était clair, le développement web ! C’est alors que j’ai appris l’existence de la formation DWWM dispensée par la SOFIP, je m’y suis directement inscris et j’ai eu la chance d’y participer. Depuis le 29 Mai, je la suis avec la plus grande assiduité et tout le sérieux dont je puisse faire preuve.

# Le projet Fic’Tif - Concept

## Introduction

Afin de mettre en avant les compétences que j’ai acquises depuis le début ma formation, j’ai voulu les illustrées dans un projet complet, ambitieux et passionnant. Je devais alors trouver un concept qui puisse cocher toutes les cases tout en symbolisant un challenge certain et être un prétexte à l’apprentissage.

Pour l’aspect complet, il me fallait un sujet qui introduise la gestion de données et leurs manipulations, un affichage dynamique qui soit simple à comprendre, en somme : une interface destinée à n’importe quel type de publique. Pour l’aspect ambitieux je souhaitais implémenter de nombreuses fonctionnalités et essayer de créer une structure conséquente. Et enfin, pour le passionnant, un sujet dans lequel je puisse m’y retrouver.

A l’intersection de ses trois prérequis que je me suis fixé, j’ai trouvé le domaine médical. Je me devais alors de réaliser un site pour tout le monde tout en prenant en compte les règles d’accessibilité. Le sujet est suffisamment vaste pour que je puisse y trouver de nombreux besoins auxquels il faudrait répondre. Et bien que le sujet ne fasse pas partie de mes passions ou que mon quotidien n’est pas réellement attaché au milieu médical, j’aimais l’idée de créer quelque chose que tout le monde pourrait utiliser un jour ou l’autre. Et puis ça me permet de garder à l’esprit que l’application que je développe ne doit pas que me convenir, mais dois aussi être pensée pour convenir au plus grand nombre.

C’est alors que l’idée d’un site de prise de rendez-vous médical m’est venue. Mon premier projet personnel de développement web et web mobile (Hors projet étudiant) sera alors un site de prise de rendez-vous médical !

## Services délivrés par l’application

### Prise de rendez-vous

La principale fonctionnalité de mon application serait alors de prendre rendez-vous avec un médecin particulier en fonction de ses disponibilités. Etant la fonctionnalité principale, je voulais qu’elle soit facilement accessible et simple d’usage.

### Suivi des rendez-vous

Une fois un ou plusieurs rendez-vous effectué/s. Je voulais qu’ils soient visibles dans une page dédiée afin que l’utilisateur puisse voire l’état de son/ses rendez-vous et également avec qui, ou et quand ce/ces rendez-vous aura/auront lieu.

### Gestion du compte utilisateur

Je me suis rendu compte très rapidement que j’aurais besoin d’un système d’identifiant. J’aurais besoin d’implémenter un système de connexion utilisateur avec la possibilité de se déconnecter. Mais je voulais aussi que l’utilisateur puisse avoir le contrôle sur ses informations de compte, qu’il puisse en changer ou encore supprimer son compte.

D’autres fonctionnalités ont germé dans mon esprit au cours du développement de l’application mais ces trois là représentent la base de ma réflexion initiale.

## Besoins fonctionnels

Ces différents services génèrent déjà plusieurs besoins avant d’être pleinement conçu. J’ai appris, après le début de mon projet, que ces réflexions conceptuelles sont à l’origine de la rédaction d’une série de besoins métier qu’un Product Owner se doit de centraliser et verbaliser afin de les distribuer dans des User Story.

En ce sens, j’ai divisé mes besoins fonctionnels entre le profil patient et le profil docteur.

### Quand je suis patient

* Je dois être capable de me connecter POUR accéder aux services requérant une connexion
* Je dois être capable de m’inscrire POUR obtenir des informations de connexion
* Je dois être capable de me déconnecter POUR des raisons de sécurité ET POUR permettre à d’autres utilisateurs d’utiliser l’application
* Je dois être capable de reprendre ma session en cours POUR ne pas avoir à me reconnecter si la période de session n’est pas dépassée
* Je dois être capable de consulter mes informations de compte POUR vérifier qu’elles sont à jour
* Je dois être capable de modifier mes informations de compte POUR les mettre à jour
* Je dois être capable de supprimer mon compte POUR retirer mes informations du site
* Je dois être capable de voire la liste des médecins POUR connaitre la totalité des services proposés
* Je dois être capable, si je suis connecté, d’atteindre la page de rendez-vous POUR prendre rendez-vous avec un médecin
* Je dois être capable, sur la page de rendez-vous, de filtrer les médecins par Spécialité POUR obtenir un rendez-vous qui correspond à mes besoins
* Je dois être capable, sur la page de rendez-vous, de sélectionner un créneau horaire POUR réserver ce créneau pour mon rendez-vous
* Je dois être capable, sur la page de rendez-vous, d’être confronter à une étape de confirmation POUR ne pas prendre rendez-vous par accident
* Je dois être capable de consulter les rendez-vous en cours qui me concerne POUR ne pas les oublier

### Quand je suis docteur

* Je dois être capable de me connecter POUR accéder aux services requérant une connexion
* Je dois être capable de me déconnecter POUR des raisons de sécurité ET POUR permettre à d’autres utilisateurs d’utiliser l’application
* Je dois être capable de reprendre ma session en cours POUR ne pas avoir à me reconnecter si la période de session n’est pas dépassée
* Je dois être capable de consulter mes informations de compte POUR vérifier qu’elles sont à jour
* Je dois être capable de modifier mes informations de compte POUR les mettre à jour
* Je dois être capable de supprimer mon compte POUR retirer mes informations du site
* Je dois être capable de consulter les rendez-vous en cours qui me concerne POUR ne pas les oublier

## Problématiques techniques

Afin de répondre efficacement à ses besoins, j’ai des impératifs techniques :

* J’ai besoin d’un front end pour le visuel de mon application tout en veillant à l’accessibilité et le backend pour gérer la communication avec les données stockées.
* J’ai besoin d’une base de données pour stocker les données.
* Ma page de prise de rendez-vous doit accueillir un calendrier à l’affichage dynamique pour organiser de façon claire et visuelle les créneaux disponibles pour que le patient puisse prendre son rendez-vous.

Ces points ont constitué le constat initial de mon application.

## Solutions envisagées

Afin de surmonter ces problématiques, j’ai opté pour l’usage de Node.js et React pour le front end, cela me permettra une flexibilité pour m’adapter aux problématiques que je rencontrerais au cours du développement et me permettra plus facilement de mettre en place une structure conséquente. J’ai décidé d’utiliser Express en complément de Node.js pour effectuer le back-end. Ce choix s’est effectué de part la simplicité d’usage, la quantité d’informations présente sur internet concernant Express et surtout par sa capacité à fournir un backend robuste.

# Fic’Tif - Conception

## Maquettage de l’application

### La théorie des couleurs

Lors de la conception de ma maquette, j’ai été préoccupé par l’identité de mon application. La couleur y jouant un rôle capital, je me suis intéressé aux différentes nuances disponibles et également à leur signification.

D’après le site [Adobe.com](https://www.adobe.com/fr/creativecloud/design/discover/color-symbolism.html), *(*[*https://www.adobe.com/fr/creativecloud/design/discover/color-symbolism.html*](https://www.adobe.com/fr/creativecloud/design/discover/color-symbolism.html)*)*  
  
***« Le Bleu***

***Associé à****: la mer et le ciel*

***Effet psychologique****: stabilité, harmonie, paix, calme et confiance*

***Utilisation par les marques et dans le design****: le bleu peut aider à positionner la marque comme digne de confiance, fiable et relaxante. C'est le cas d'enseignes de grande distribution où l'on peut faire ses courses en un seul endroit pratique. Le secteur de la santé utilise généralement le bleu dans leur image de marque pour aider les gens à associer la marque à un produit de qualité, fiable et sûr.*

***Le Orange***

***Associé à****: la créativité, l'aventure, l'enthousiasme, le succès et l'équilibre.*

***Effet psychologique****: ajoute un peu de plaisir.*

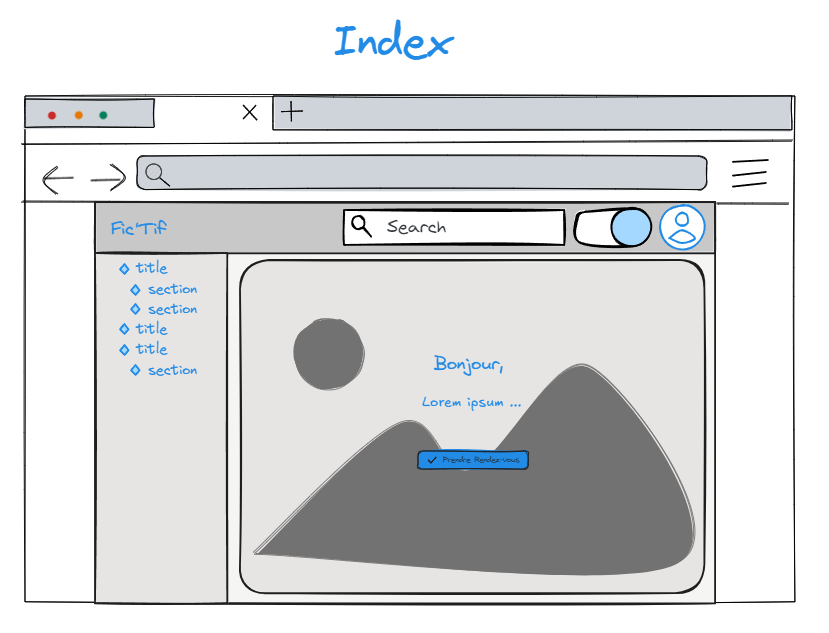
***Utilisation par les marques et dans le design****: dans les chaînes de télévision pour enfants pour représenter la créativité, l'enthousiasme et le côté ludique. De nombreuses enseignes de bricolage utilisent la couleur orange pour représenter la créativité et la rénovation de la maison. L'orange est une couleur forte qui convient le mieux aux marques tout aussi bruyantes et énergiques. Si votre marque est quelque chose d'un peu plus traditionnel, l'orange n'est probablement pas la voie à suivre. »*

Pour le domaine médical, le bleu est très régulièrement utilisé. Afin que l’utilisateur ne soit pas perturbé par une charte de couleur différente de celle qu’il a l’habitude de voir dans ce domaine, j’ai opté pour le bleu. Aussi, je voulais obtenir un visuel attrayant et dynamique, de ce fait j’ai sélectionné la couleur orange, d’autant plus qu’elle s’associe bien au bleu comme le montre ce second article

D’après le site [Adobe.com](file:///C:\Users\SOFIP\AppData\Roaming\Microsoft\Word\Adobe.com), *(*[*https://www.adobe.com/fr/creativecloud/design/discover/complementary-colors.html*](https://www.adobe.com/fr/creativecloud/design/discover/complementary-colors.html)*)*

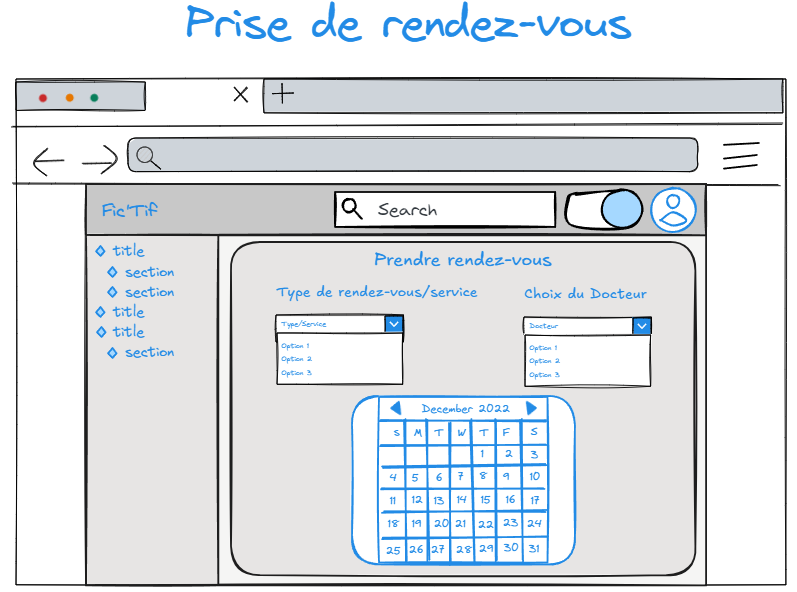
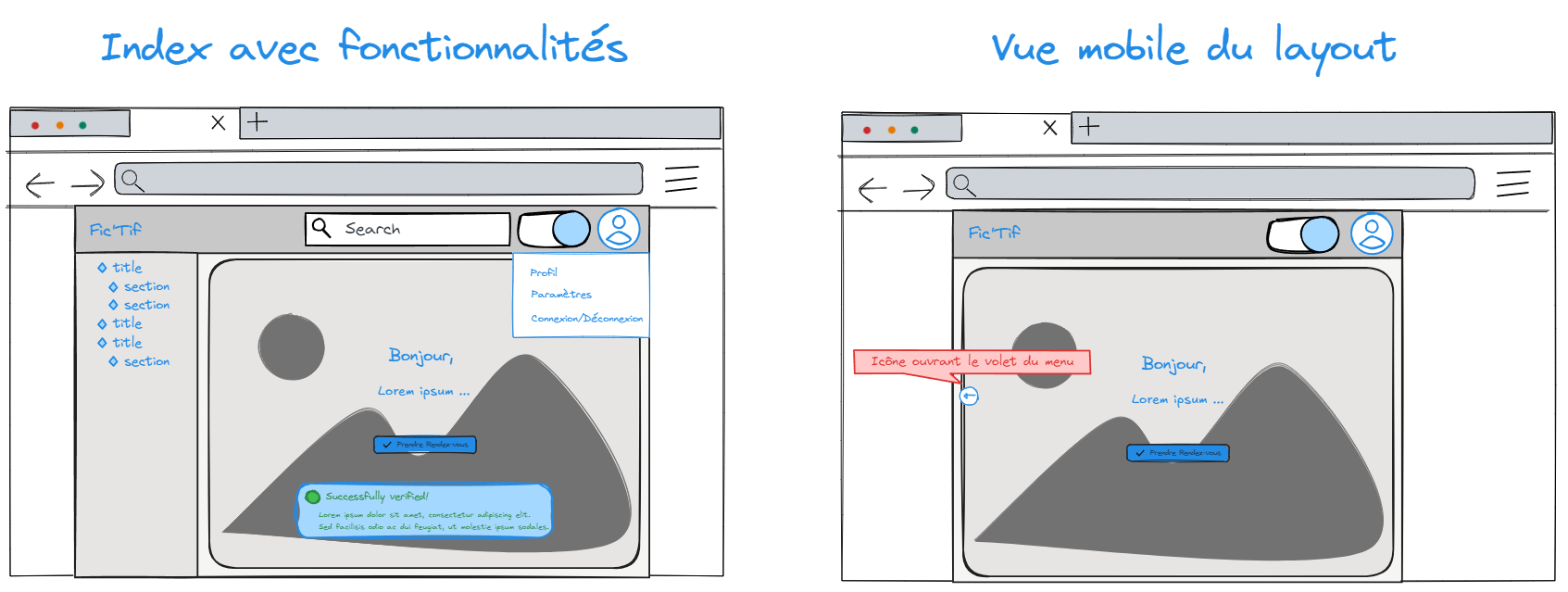
***« Couleurs complémentaires avec une couleur primaire***

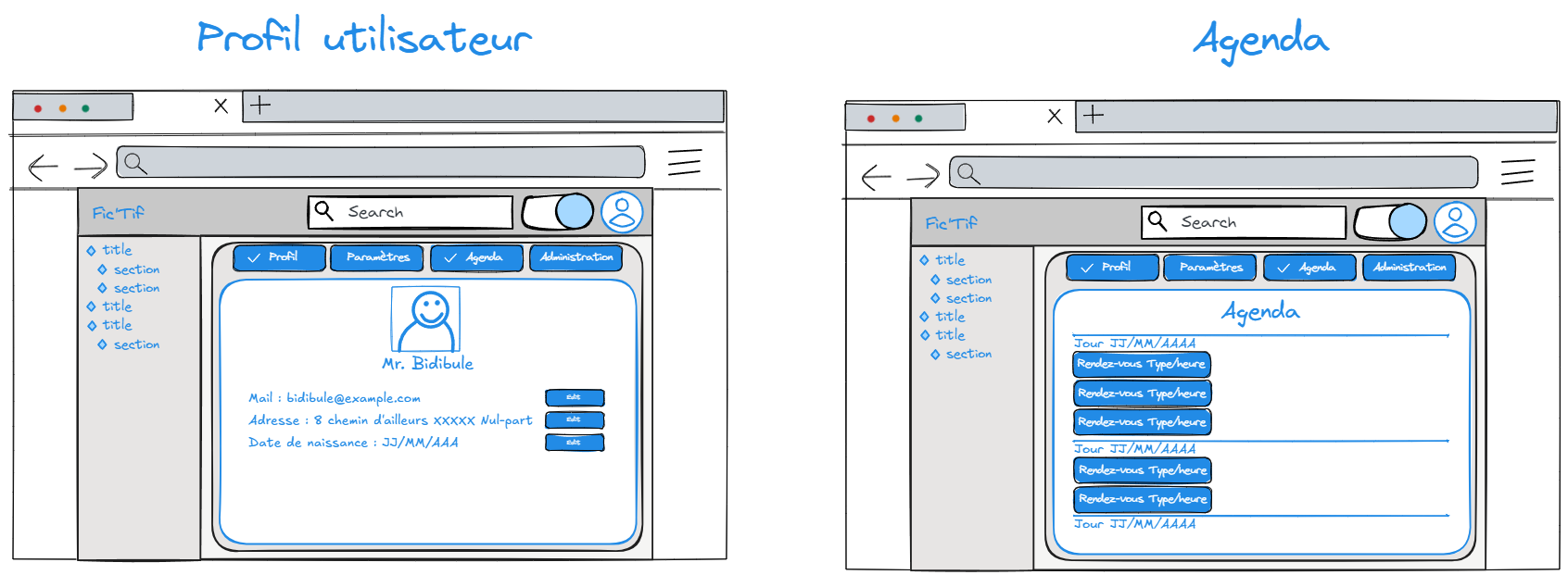
***Le bleu et l’orange****. En associant ces deux couleurs dans vos réalisations graphiques, vos peintures, vos photos ou encore votre logo, vous vous assurez d’avoir un rendu contrasté et intemporel.*



### Wireframes

Pour le design de mon application, je me suis contrains à éviter au maximum le scrolling vertical ou horizontal. J’ai donc pensé la vue mobile et desktop en fonction au sein d’un wireframe que j’ai réalisé avec [Excalidraw](https://excalidraw.com) *(*[*https://excalidraw.com*](https://excalidraw.com)*)*.





## Mise en place de l’environnement

### Les repos git

Dés le début du projet, j’ai imaginé le frontend séparé du backend. Cependant pour une raison de praticité j’ai travaillé avec un seul repo git qui contenait deux dossiers accueillant les deux parties de mon projet.

Dans un premier temps, j’ai effectué des opérations simples directement sur la branche main. Mais comme le projet demandé un développement plus ambitieux que les projets effectués dans le cadre de l’apprentissage au sein de la formation, il me fallait adopter une autre méthode de travail. J’ai alors commencé à utiliser des branches différentes pour l’implémentation de features, assurant une version stable sur la branche main et me servant des autres branches pour travailler. De cette façon j’ai appris l’usage des rebases et la gestion des conflits.

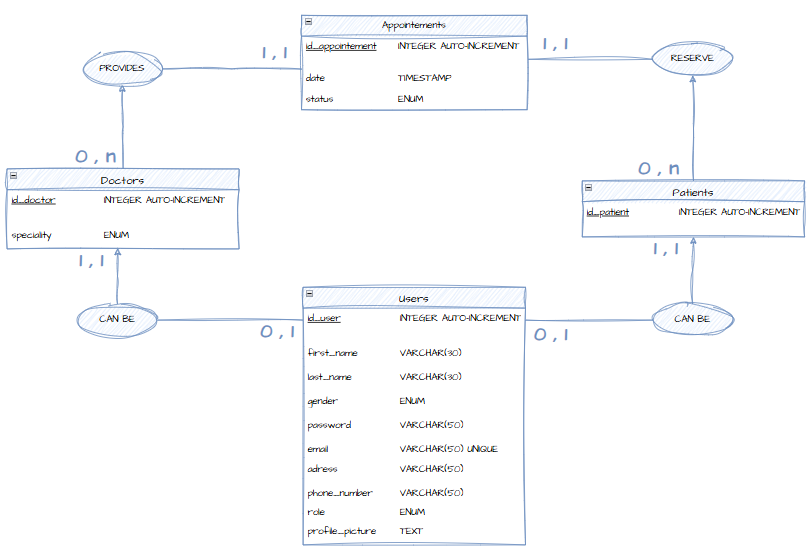
J’ai également adopté une convention de nommage pour mieux structurer le versionnage sur le repo distant. (Exemples de commit : ‘ git commit -m « fix: feature login »’ ‘git commit -m « feature: login »)

Le projet m’a permis de mettre en relief l’importance de maitriser git et la rigueur qui est nécessaire pour obtenir un dossier de projet propre.

## Mise en place de la base de données

La création d’une base de données est une étape importante. Cette dernière influencera directement la façon dont l’application s’organisera pour mener à bien les actions attendues, de ce fait j’ai créer un MCD pour mieux organiser mes idées et concevoir la logique autour de mes données et leurs interactions, puis j’ai établi seulement après la base de données.

### MCD



J’ai opté pour une organisation en losange de mes données entre les utilisateurs (Users) et les rendez-vous (Appointments) qui passerait alors par les docteurs (Doctors) et par les patients (Patients). En effet, j’ai considéré qu’un rendez-vous a lieu entre un patient et un docteur et que chacun d’entre eux partage le fait d’être un utilisateur du site, ayant des données communes comme un nom, prénom, genre, email, mot de passe, etc…

Avec cette organisation, j’évite d’avoir deux collections de données qui sont jumelles à la différence des docteurs qui aurait une spécialité (speciality) en plus par rapport aux patients.

De son côté, la collection rendez-vous n’ajoute qu’un horaire de début de consultation et un status. J’ai en effet opté pour des consultations à durée unique de 30 minutes, de ce fait je n’ai besoin que de l’horaire de début. Le status permet de notifier si le rendez-vous est en attente, annulé, ou effectué (« pending », « canceled » ou « done »).

Remarque : A cet instant, j’avais déjà notifié un problème dans mon application. Si un utilisateur peut être un docteur ou un patient, lors de l’enregistrement d’un utilisateur il me faudrait alors pouvoir faire la différence. Cependant, une case a cocher dans le formulaire d’inscription pour s’inscrire en tant que docteur est hors de propos, établir une page non accessible par la navigation par l’UI non plus (Il suffirait d’entrer le chemin dans l’URL pour obtenir le formulaire d’inscription pour les docteurs pour que quiconque puisse s’inscrire comme docteur) …

Pour répondre à cette problématique, j’ai alors imaginé un second portail web qui accueillerait le formulaire d’inscription des docteurs avec des vérifications supplémentaires (Code reçu par mail par exemple) et la partie administration de Fic’Tif. Cependant, faute de temps, ce second portail n’est pas présent dans ce projet pour le moment.

## Les dépendances indispensables

Pour réaliser correctement le projet, je connaissais déjà les problématiques techniques grâce au travail sur le concept du projet. De ce fait, souhaitant utiliser React et Node.js pour le frontend et Express et Node.js pour le backend, j’avais besoin d’une liste de module qui se montreraient indispensable pour mon projet, couvrant des aspects de sécurité ou répondant aux problématiques techniques propres au projet.

#### Front-end

* [React Router](https://www.npmjs.com/package/react-router-dom) - Permet de gérer la navigation entre les différentes vues/pages de mon application. (*https://www.npmjs.com/package/react-router-dom*)
* [React big calendar](https://www.npmjs.com/package/@types/react-big-calendar) - Permet d'ajouter un calendrier interactif et personnalisable dans mon application. (*https://www.npmjs.com/package/@types/react-big-calendar*)
* [React Toastify](https://www.npmjs.com/package/react-toastify) - Permet d'afficher des notifications toast (notifications temporaires) de manière simple et élégante tout en gardant le comportement de la page souhaité (Pas de scrolling) ([*https://www.npmjs.com/package/react-toastify*](https://www.npmjs.com/package/react-toastify))

Ici, les modules convoités par défaut se devaient de répondre à des problématiques techniques. J’ai alors commencé à parcourir leurs différentes documentations pour veillez à ce que leur emploi soit justifié et corresponde aux besoins de mon projet.

#### Back-end

* [Bcryptjs](https://www.npmjs.com/package/bcryptjs) - Permet de hacher les mots de passe de manière sécurisée. (*https://www.npmjs.com/package/bcryptjs*)
* cookie parser – Permet de parser les cookies envoyés dans les requêtes HTTP. (*https://www.npmjs.com/package/cookie-parser*)
* Cors - Permet de gérer le **Cross-Origin Resource Sharing (CORS)** pour autoriser ou restreindre l'accès à l'API depuis différents domaines. (*https://www.npmjs.com/package/cors)*
* Express async errors - Permet de gérer les erreurs asynchrones dans Express. (*https://www.npmjs.com/package/express-async-errors*)
* Express mongo sanitize - Protection contre les attaques d'injection MongoDB en nettoyant les requêtes entrantes. (*https://www.npmjs.com/package/express-mongo-sanitize*)
* Express rate limit - Limitation du nombre de requêtes d'un utilisateur pour prévenir les attaques par déni de service (DDoS) ou le brute-force. (*https://www.npmjs.com/package/express-rate-limit*)
* Helmet - Middleware de sécurité pour Express qui aide à protéger l'application contre des vulnérabilités web courantes. (*https://www.npmjs.com/package/helmet*)
* http status codes - Fournit des constantes pour les codes de statut HTTP afin de rendre le code plus lisible et éviter les erreurs de typographie. (*https://www.npmjs.com/package/http-status-codes*)
* Jsonwebtoken - Permet de créer et vérifier des JSON Web Tokens (JWT) pour l’authentification. (*https://www.npmjs.com/package/jsonwebtoken*)
* Mongoose - ODM (Object Data Modeling) pour MongoDB, facilite l'interaction avec la base de données en utilisant des modèles. (*https://www.npmjs.com/package/mongoose*)
* Zod - Bibliothèque de validation de schémas JavaScript pour valider les entrées. (*https://www.npmjs.com/package/zod*)

Avec l’usages de ses modules je m’assure d’obtenir un backend sécurisé, capable de gérer les sessions des utilisateurs et les erreurs éventuelles ainsi que la communication avec une base de donnée MongoDB.

Remarque : Ce projet m’a permis d’apprendre comment se comporte un backend et comment développer une API (Gestion des erreurs, communication entre le front et le back etc etc). J’ai donc suivi des recommandations pour m’assurer d’aborder chaque point de vigilance que l’on se doit de connaitre en tant que développeur.

## Création du back-end

### Technologies employées

J’ai utilisé Express et Node.js pour le backend, j’avais besoin d’une liste de module qui se montreraient indispensable pour mon projet, couvrant des aspects de sécurité ou répondant aux problématiques techniques propres au projet.

### Création de la base MongoDB

J’ai alors créé ma base MongoDB et je l’ai associé à mon backend naissant. J’ai pour cela eu recours a un fichier .env qui me permet de stocker les variables d’environnement nécessaires au bon fonctionnement de mon application. La première de ses variables fut donc celle requise pour me connecter à ma base de données MongoDB.

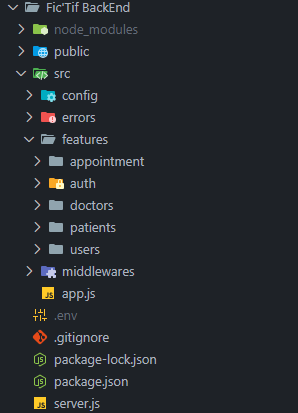
Par la suite, ce fichier se verra également attribuer des variables qui serviront à paramétrer mes Tokens, mes cookies et le port de déploiement lors de la phase de développement.

### Modules et dépendances

* [Bcryptjs](https://www.npmjs.com/package/bcryptjs) - Permet de hacher les mots de passe de manière sécurisée. (*https://www.npmjs.com/package/bcryptjs*)
* cookie parser – Permet de parser les cookies envoyés dans les requêtes HTTP. (*https://www.npmjs.com/package/cookie-parser*)
* Cors - Permet de gérer le **Cross-Origin Resource Sharing (CORS)** pour autoriser ou restreindre l'accès à l'API depuis différents domaines. (*https://www.npmjs.com/package/cors)*
* Express async errors - Permet de gérer les erreurs asynchrones dans Express. (*https://www.npmjs.com/package/express-async-errors*)
* Express mongo sanitize - Protection contre les attaques d'injection MongoDB en nettoyant les requêtes entrantes. (*https://www.npmjs.com/package/express-mongo-sanitize*)
* Express rate limit - Limitation du nombre de requêtes d'un utilisateur pour prévenir les attaques par déni de service (DDoS) ou le brute-force. (*https://www.npmjs.com/package/express-rate-limit*)
* Helmet - Middleware de sécurité pour Express qui aide à protéger l'application contre des vulnérabilités web courantes. (*https://www.npmjs.com/package/helmet*)
* http status codes - Fournit des constantes pour les codes de statut HTTP afin de rendre le code plus lisible et éviter les erreurs de typographie. (*https://www.npmjs.com/package/http-status-codes*)
* Jsonwebtoken - Permet de créer et vérifier des JSON Web Tokens (JWT) pour l’authentification. (*https://www.npmjs.com/package/jsonwebtoken*)
* Mongoose - ODM (Object Data Modeling) pour MongoDB, facilite l'interaction avec la base de données en utilisant des modèles. (*https://www.npmjs.com/package/mongoose*)
* Zod - Bibliothèque de validation de schémas JavaScript pour valider les entrées. (*https://www.npmjs.com/package/zod*)

Avec l’usages de ses modules je m’assure d’obtenir un backend sécurisé, capable de gérer les sessions des utilisateurs et les erreurs éventuelles ainsi que la communication avec une base de données MongoDB.

### Structure

La structure de mon backend veille à regrouper chaque fonctionnalité par types, permettant une navigation fluide et un suivi efficace de ces mêmes fonctionnalités.

Les noms de dossiers config et errors sont relativement pertinents. Ils me permettent de d’accueillir le fichier de configuration de la connexion vers la base de données et mes gabarits d’erreur comme pour l’erreur d’authentification.

Le dossier features contient les controllers, services et potentiel shémas/models nécessaire pour dispenser les services dont mon application a besoin.

auth par exemple, gère tout ce qui concerne la connexion des utilisateurs, de la création de compte à la vérification de validité de session.

Le dossier middlwares contient les différents middlewares comme ceux nécessaires à la validation de Schema à l’aide de zod, les middlewares de gestion d’erreur, ou encore d’erreurs d’authentification.

### Création des premières routes

J’ai alors mis en place mes premières routes pour tester le bon fonctionnement de mon backend.



Organisée de cette manière, l'application est donc structurée en différentes fonctionnalités (authentification, utilisateurs, rendez-vous, médecins) avec des points d'API distincts, ce qui facilite la gestion et la maintenance de chaque fonctionnalité.

Les middlewares notFound et errorHandler permettent de gérer les cas où les routes sont introuvables ou où des erreurs surviennent pendant le traitement de la requête.

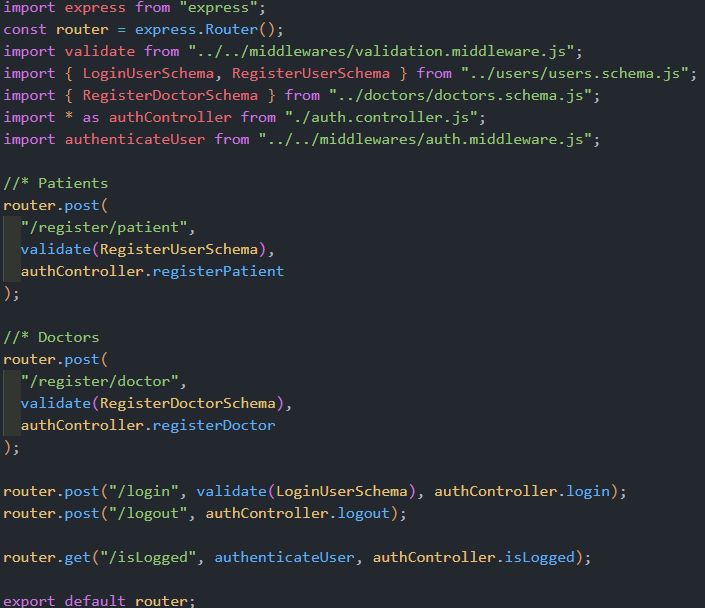
Cette séparation assure une clarté sur les différentes fonctionnalités de l’application, qui sont ensuite structurées de la façon suivante :

ROUTES -> MIDDLEWARES -> CONTROLLERS -> SERVICES

#### Routes

Chaque features a un fichier .route qui lui est associé, permettant de rediriger vers les différents controllers des services qui sont englobés dans la fonctionnalité.

Par exemple le système d’authentification :



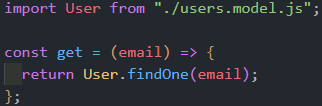
De cette manière j’obtiens un suivi clair de ce que chaque route effectue et intégrer un middleware comme ceux de validation de Schema ou de validation de session courante est facile sans perturber la clarté des routes ainsi créées.

#### Controllers



Ce controller à pour rôle de vérifier que la requête reçue est valide, sinon elle retourne une erreur d’authentification si la requête ne possède pas un email existant dans la base de données (Cette vérification s’effectue à l’aide d’un service dédié : usersService.get) ou si le mot de passe récupéré ne correspond pas à celui associé à l’adresse mail (Cette vérification s’effectue à l’aide du model de user qui a une méthode intégrée).

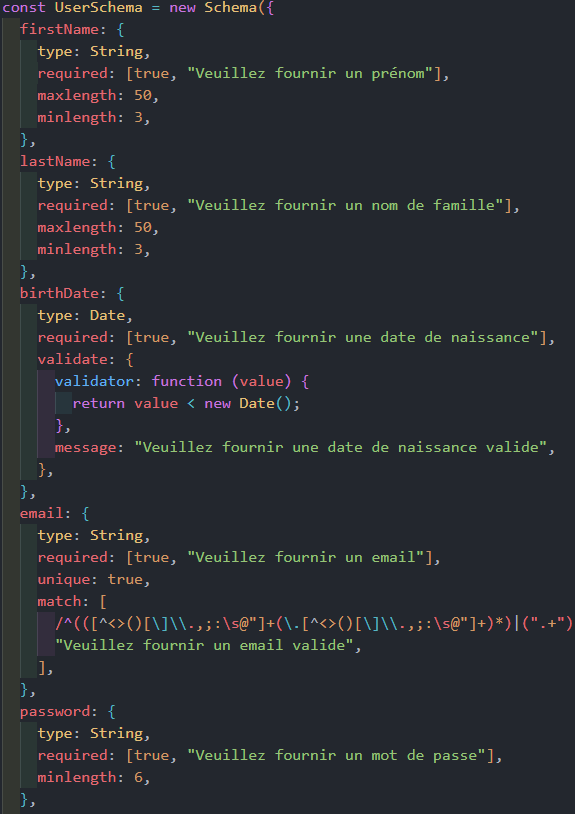
#### Services

Ce service est enfin appelé pour effectuer les vérifications nécessaires afin de logger un utilisateur.

Au préalable, l’on doit utiliser un ODM (Ici mongoose) pour définir des modèles d’objets afin de correspondre à la structure de nos documents et interagir avec eux à l’aide d’opérations CRUD (Create, Read, Update, Delete), dans cet exemple ce serait READ.

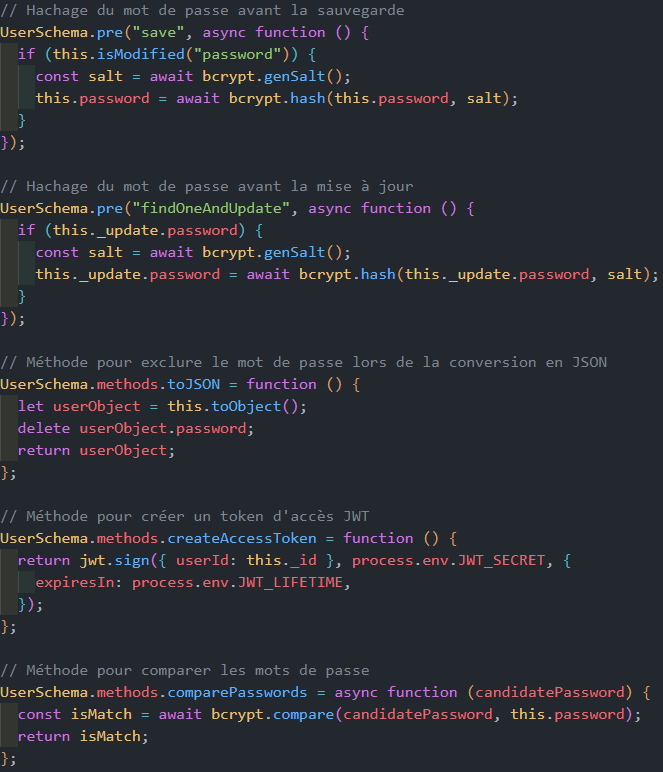
Il va rechercher dans la collection users un document avec comme valeur l’adresse email reçu dans la requête pour le champ email. (Les documents stockés dans la base MongoDB étant dans un format très similaire au JSON). Si un utilisateur possède cette adresse email, le service renvoi alors cet utilisateur au controller qui continue ses vérifications.

#### Models

Pour réaliser cette action, on a donc besoin du model d’objet user.

Ici, les propriétés du modèle sont définies afin qu’elles servent pour peupler les documents dans la base de données. Elles peuvent être définies avec des critères, par exemple, le nom et le prénom son ici requis et doivent être compris entre 3 et 50 caractères. D’autres propriétés sont présentes dans le modèle user, comme l’adresse, le numéro de téléphone et le rôle. Le rôle d’un user ne peut avoir que deux valeurs « patient » ou « doctor », c’est cette propriété qui permet de différencier un patient d’un médecin dans certains usages.

Ensuite, dans le cas du modèle users, on a également besoin de méthodes. Les méthodes sont des fonctions qui sont définies directement sur le modèle afin de manipuler des données ou d’effectuer des opérations spécifiques sur des documents de la base de données.



#### Schemas

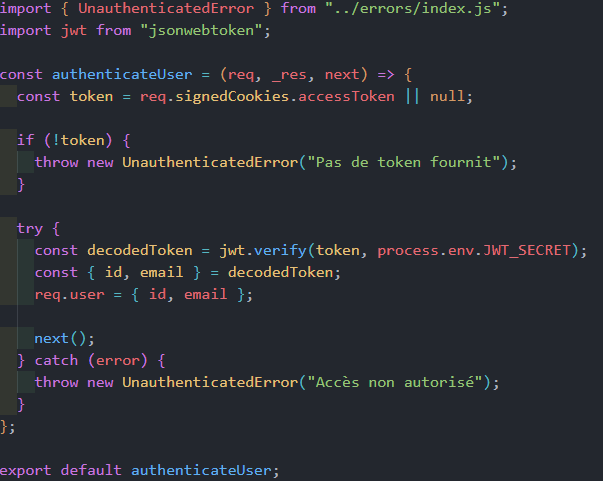
Les Schemas sont utiles pour vérifier les informations provenant du front et donc de l’utilisateur. C’est une bonne pratique d’effectuer cette action avant de contacter la base de données pour ne pas la compromettre en cas d’informations vérolées. Si nous reprenons l’exemple des routes précédemment exposées, la route /register/patient/ a un middleware de validation de Schema (Celui de « RegisterUserSchema ») qui compare les données reçues avec celles du Schema suivant :



Cela ressemble dans un certain sens au modèle, puisque le Schema vérifie que les informations reçues respectent bien des critères. Mais son objectif est bien différent ! Mongoose et ses modèles permettent d’effectuer des opérations en vue de communiquer avec la base de données. Zod et ses Schemas quant à eux, vérifient que les informations saisies correspondent au format attendu pour éviter de potentiels problèmes par la suite et pour renforcer la sécurité liée aux requêtes que l’API reçoit.

#### Middlewares

Les middlewares sont donc des fonctions qui vont effectués des vérifications ou des modifications sur la requête. La validation des schemas se comportent comme des middlewares et l’authenticateUser est également un middleware que j’appelle pour vérifier si le JWT token est toujours en cours de validité à chaque fois que la route /isLogged est appelée.



De cette façon j’obtiens une route globale qui gère mon authentification sur l’application /auth qui est ensuite divisée en plusieurs routes qui me garantissent la possibilité de réaliser les opérations nécessaires pour implémenter un gestionnaire de connexion tout en le rendant sécurisé à l’aide de middlewares et réalisant les tâches demandées à l’aide de l’ODM Mongoose. (Inscription, connexion, déconnexion, vérification de la validité de la session)

### Sécurité

## Simulation des routes backend

### Installation de PostMan

### Tests des routes backend

### Premiers peuplements de la base MongoDB

## Création du front-end

### Technologies employées

#### Daisy UI

#### Axios

#### React-router-dom

#### React-big-calendar

#### React-toastify

### Structure

### Création des routes de navigations

### Création d’un layout

## Fonctionnalités

### Theme toggler

### Prise de rendez-vous

### Login/Logout

#### Page de Login/Logout

#### Modale RGPD

#### Contexte de l’utilisateur

### Dashboard

#### Profil

#### Paramètres

#### Agenda

### Recherche de médecin

## Tests

## Documentation

## Déploiement

# La trêve de novembre

# Problèmes rencontrés

## MCD

## Conception du backend

## Login/Logout

# **Conclusion**

## Technologies balayées par le projet

### Frontend

### Backend

## Axes d’amélioration

## Et si c’était à refaire ?!

## Bilan

### Remerciements

### Le mot de la fin

# **Test**

## Titre secondaire

### Titre section

#### Titre sous-section