Rapport de projet

Clinic Management System

Par: IBRAHIM Ahmad
Enseignant: M. Guévork Djaladian

1. Remerciement

Je tiens à exprimer mes sincères remerciements à Mr. DJALADIAN pour sa patience, sa guidance et son soutien tout au long de ce module de développement full stack. Grâce à ses précieux conseils et à son enseignement de qualité, j'ai pu acquérir les compétences nécessaires pour réaliser ce projet avec succès. Je suis profondément reconnaissant pour tout le temps qu'il a consacré, et je suis convaincu que ces leçons me seront très utiles dans ma carrière professionnelle future. Encore une fois, merci beaucoup pour tout ce que vous avez fait.

2. Résumé

J'ai conçu une application de gestion de clinique pour aider les professionnels de la santé à gérer efficacement leurs patients et leurs rendez-vous. Elle permet aux utilisateurs de créer et de gérer des fiches de patients, d'organiser des rendez-vous. En outre, l'application offre une vue d'ensemble des rendez-vous de la semaine. J'ai utilisé NestJs pour le développement backend, TypeORM Mysql pour stocker les données de l'application et j'ai développé une interface utilisateur intuitive avec React template Typescript pour faciliter l'utilisation de l'application par les professionnels de la santé. En tout, mon application de gestion de clinique vise à améliorer la gestion des patients et à rendre le processus de prise de rendez-vous plus efficace pour les professionnels de la santé.

3. Introduction

J'ai eu du mal à trouver une idée originale pour mon projet, car mon ego me poussait à vouloir faire quelque chose de plus complexe que de simples projets. Finalement, après beaucoup de temps, j'ai décidé de me lancer dans la création d'un "clinic management system", car je ne voulais pas simplement reproduire des projets déjà existants comme Instagram, Facebook ou Twitter, qui sont facilement accessibles sur GitHub. Je voulais plutôt me lancer dans quelque chose d'intéressant et qui nécessiterait l'utilisation de certaines technologies que l'on ne trouve pas facilement sur GitHub.
L'objectif de cette application est de simplifier le processus de prise de rendez-vous et de permettre aux utilisateurs de suivre facilement leur carnet de rendez-vous.
Le projet n’est pas 100% complet, il manque beaucoup de fonctionnalités et concernant le front-end les pages ne sont pas toutes faites, justement à cause du manque du temps, mais bien sûr ça ne m'empêche pas de les finir, enfin c’est un projet personnel.
J’ai essayé au début de faire un “clean code”, bien sur pour écrire un clean code il faut tout d'abord réfléchir avant de commencer à coder, mais bon j’avais pas de temps pour faire ca, mais je vais faire un “refactor” pour mon code pour le rendre clean après, mais ce qui est important et de prendre une note sur ce projet pour l’instant.
Ce rapport a pour objectif de présenter mon application de gestion de clinique et de décrire les étapes que j’ai suivies pour la

développer. Je présenterai également les fonctionnalités de l'application et leur utilité pour les professionnels de la santé. Enfin, je discuterai des perspectives d'avenir de mon application et des pistes d'amélioration que j'ai identifiées.

4. Gestion du projet

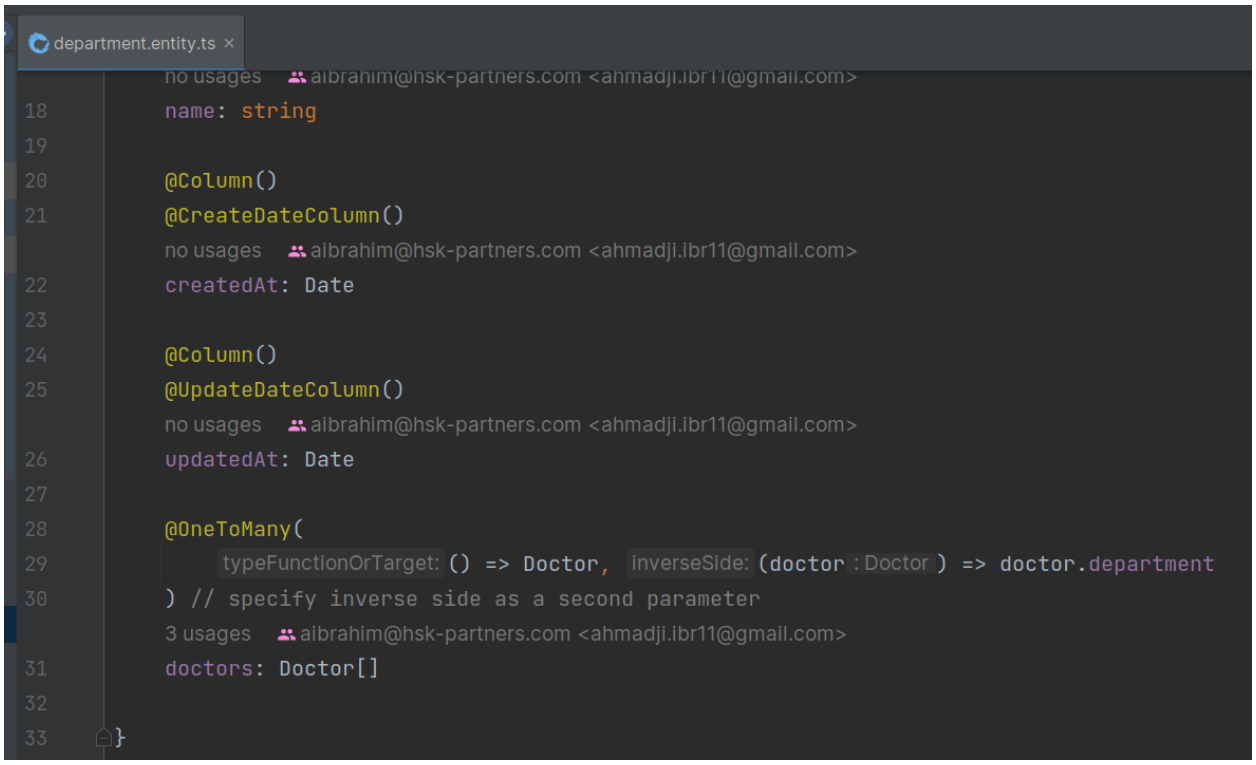
Au départ, j'ai consacré 4 jours à réfléchir à la création de la base de données de mon application. J'ai fait des recherches pour trouver une idée originale et j'ai élaboré un plan détaillé des fonctionnalités que je voulais inclure dans mon projet, comme le mode de paiement et la gestion des membres du personnel. J'ai ensuite créé ma base de données en utilisant des fichiers .entity qui définissent les colonnes et les propriétés de chaque table, ainsi que les relations entre elles. La création de la base de données s'est déroulée sans difficulté, car j'ai l'habitude de travailler avec ce genre de technologies.

Une fois la base de données créée, j'ai entamé le développement du backend de mon application, qui consiste à implémenter les fonctionnalités de l'application et à les connecter à la base de données. J'ai commencé par mettre en place les opérations CRUD (create, read, update, delete) pour chaque table de la base de données, avant de m'occuper de l'authentification des utilisateurs. J'ai choisi de mettre en place l'authentification en dernier, car c'est une étape qui nécessite souvent un peu plus de temps et de réflexion. Pendant le développement, j'ai organisé mon code de manière à ce que les logiques et les configurations soient regroupées dans des fichiers .service dans chaque dossier. Je n'ai pas rencontré de difficultés particulières pendant le développement du backend, même si mon code n'est pas encore parfaitement "clean" selon les standards de l'industrie.

Exemple des entités:

department.entity.ts

```typescript
import {   Aibrahim, 12/13/22, 1:53 AM • create the whole databa
    BaseEntity,
    Column,
    CreateDateColumn,
    Entity, OneToMany,
    PrimaryGeneratedColumn,
    UpdateDateColumn
} from "typeorm";
import {Doctor} from "../../doctors/entities/doctor.entity";

5+ usages   aibrahim@hsk-partners.com <ahmadji.ibr11@gmail.com>
@Entity( name: 'departments')
export class Department extends BaseEntity {

    @PrimaryGeneratedColumn()
    3 usages   aibrahim@hsk-partners.com <ahmadji.ibr11@gmail.com>
    id: number

    @Column()
    no usages   aibrahim@hsk-partners.com <ahmadji.ibr11@gmail.com>
    name: string

    @Column()
```

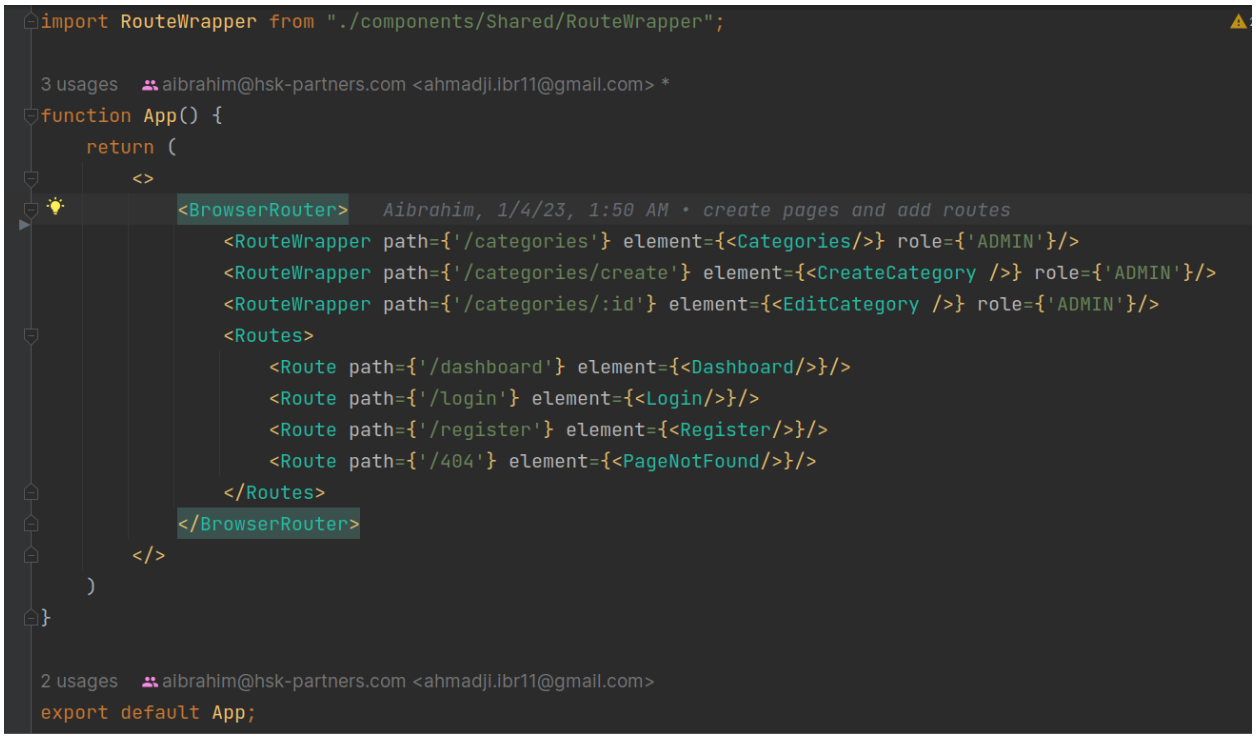


Exemple de service:

Enfait j’ai pas parlé des technos utilisées et de la structure de projet parce que c'était pas de notre choix, c’est ce qu’on a prix dans le module.

Cotés Front-End:

Peut être c’est la partie la plus dur, ou j’ai un peu pris de temps dedans, comme que j’aime pas faire des truc basic du coup je peux pas faire que des fonctionnalité seulement alors que c’est du UI (user interface) et donc il faut faire une “dashboard” très attractive, comme cette dashboard par exemple :
Material Tailwind
Malheureusement, le manque de temps a été un frein pour la réalisation de toutes les fonctionnalités que j'avais prévues. J'ai donc décidé de me concentrer sur les pages de login et de registre pour les patients, qui sont essentielles pour l'utilisation de mon application, et d'implémenter les trucs principaux qu’on pris dans le cours pour prendre un peu des points 😕.
J’ai créé un composant RouteWrapper pour la restriction des accès sur les pages en fonction des rôles.



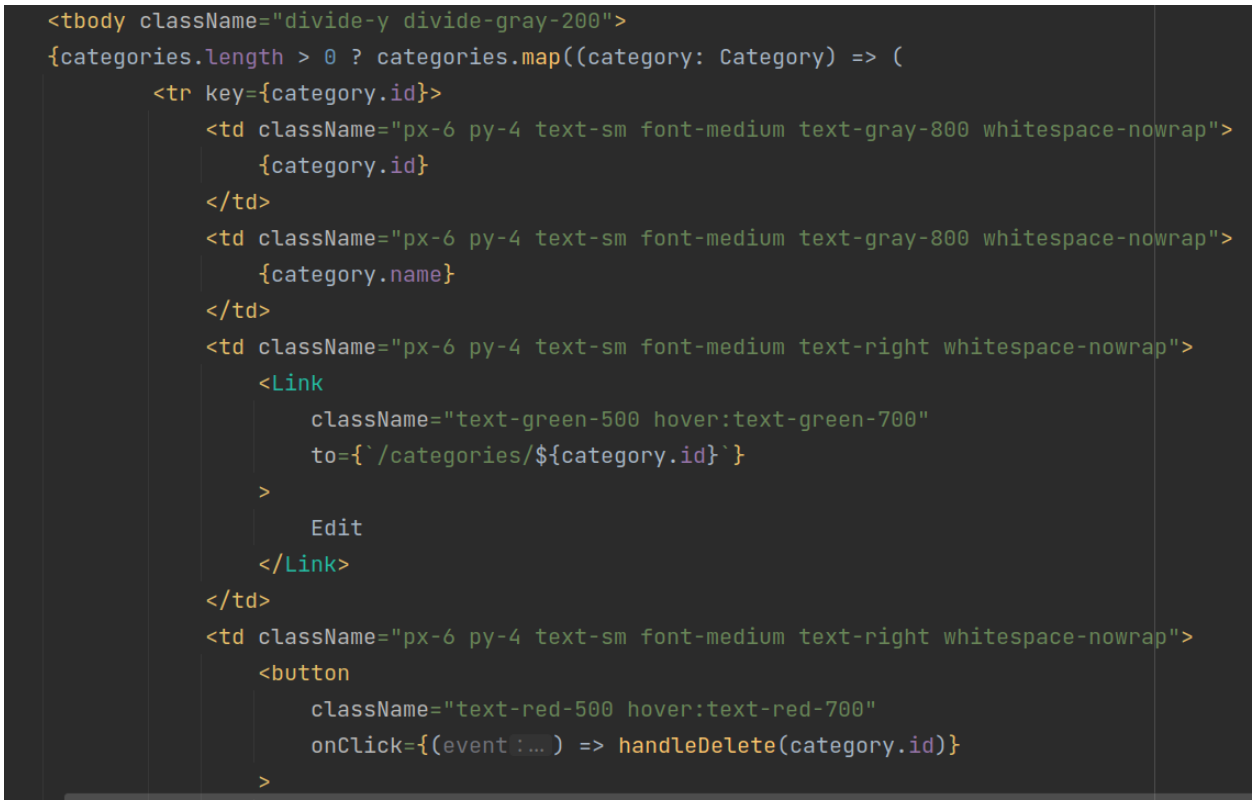
```
import RouteWrapper from "./components/Shared/RouteWrapper";

3 usages   aibrahim@hsk-partners.com <ahmadji.ibr11@gmail.com> *
function App() {
    return (
        <>
            <BrowserRouter>   Aibrahim, 1/4/23, 1:50 AM • create pages and add routes
                <RouteWrapper path={'/categories'} element={<Categories/>} role={'ADMIN'}/>
                <RouteWrapper path={'/categories/create'} element={<CreateCategory />} role={'ADMIN'}/>
                <RouteWrapper path={'/categories/:id'} element={<EditCategory />} role={'ADMIN'}/>
                <Routes>
                    <Route path={'/dashboard'} element={<Dashboard/>}/>
                    <Route path={'/login'} element={<Login/>}/>
                    <Route path={'/register'} element={<Register/>}/>
                    <Route path={'/404'} element={<PageNotFound/>}/>
                </Routes>
            </BrowserRouter>
        </>
    )
}

2 usages   aibrahim@hsk-partners.com <ahmadji.ibr11@gmail.com>
export default App;
```


import ...
type RouteWrapperProps = {
4 usages aibrahim@hsk-partners.com <ahmadji.ibr11@gmail.com>
element: React.ReactNode;
4 usages aibrahim@hsk-partners.com <ahmadji.ibr11@gmail.com>
path: string;
5+ usages aibrahim@hsk-partners.com <ahmadji.ibr11@gmail.com>
role?: string;
}
5+ usages aibrahim@hsk-partners.com <ahmadji.ibr11@gmail.com>
const RouteWrapper: React.FC<RouteWrapperProps> = (props : RouteWrapperProps) => {
const navigate = useNavigate();
console.log(localStorage.getItem(key: 'role'));
useEffect(effect: () => {
(
() => {
if (props.role && localStorage.getItem(key: 'role') !== props.role) {
console.log('redirect');
navigate('/404');
}

Pour l'instant ce qui fonctionne dans mon front-end: Les CRUD des catégories, Login et register (création du profil de patient).
Je compte continuer à travailler sur le front-end de mon application dans les prochaines semaines pour en améliorer l'ergonomie et l'esthétique, et finir ce que j'ai commencé et arriver a mon but.

Exemple du composant front-end avec l’appelle de l’api:

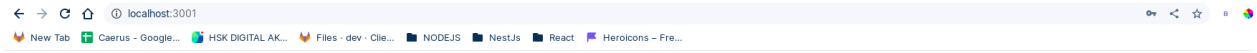
```
import {Category} from "../../models/Category";
import React, {useEffect, useState} from "react";
import Wrapper from "../Shared/Wrapper";
import {Link} from "react-router-dom";    Aibrahim, 1/6/23, 4:28 AM • Category CRUD fini

3 usages    aibrahim@hsk-partners.com <ahmadji.ibr11@gmail.com>
const Categories = () => {

    const [categories, setCategories] = useState( initialState: []);

    1 usage    aibrahim@hsk-partners.com <ahmadji.ibr11@gmail.com>
    const getCategories = async () => {
        const response = await fetch( input: 'http://localhost:3000/categories', init: {
            method: 'GET',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': 'Bearer ' + localStorage.getItem( key: 'token')
            },
        });
        return await response.json();
    }

    useEffect( effect: () => {
```

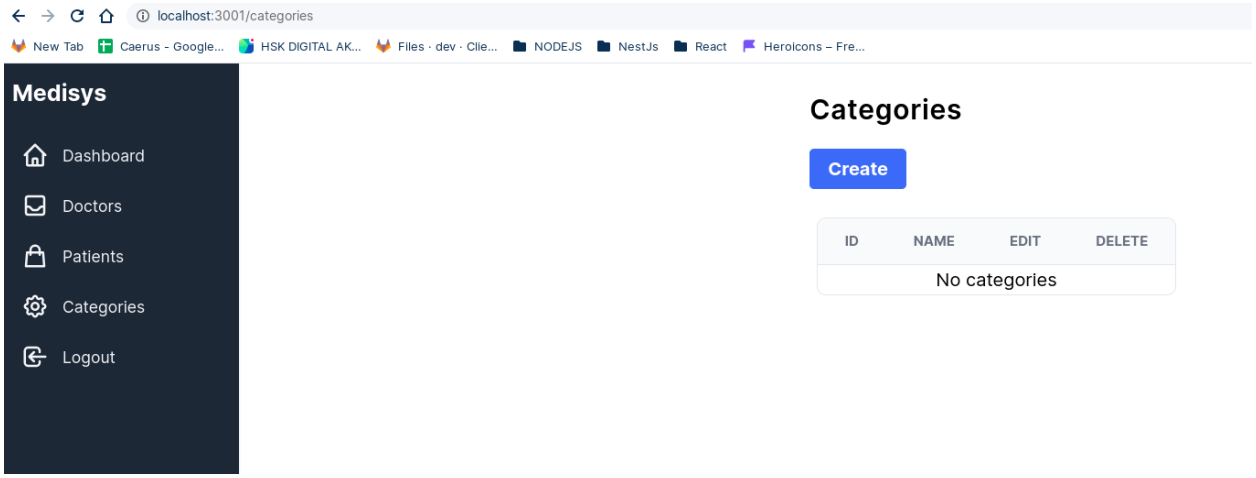


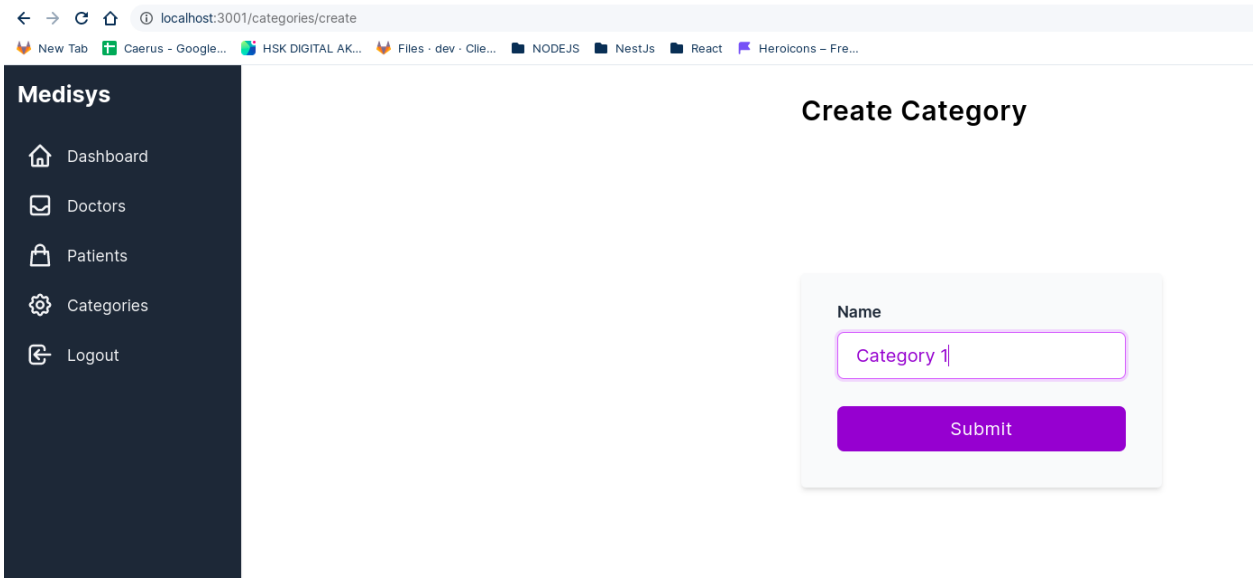
```
<tbody className="divide-y divide-gray-200">
{categories.length > 0 ? categories.map((category: Category) => (
        <tr key={category.id}>
            <td className="px-6 py-4 text-sm font-medium text-gray-800 whitespace-nowrap">
                {category.id}
            </td>
            <td className="px-6 py-4 text-sm font-medium text-gray-800 whitespace-nowrap">
                {category.name}
            </td>
            <td className="px-6 py-4 text-sm font-medium text-right whitespace-nowrap">
                <Link
                    className="text-green-500 hover:text-green-700"
                    to={`/categories/${category.id}`}
                >
                    Edit
                </Link>
            </td>
            <td className="px-6 py-4 text-sm font-medium text-right whitespace-nowrap">
                <button
                    className="text-red-500 hover:text-red-700"
                    onClick={(event:… ) => handleDelete(category.id)}
                >
```

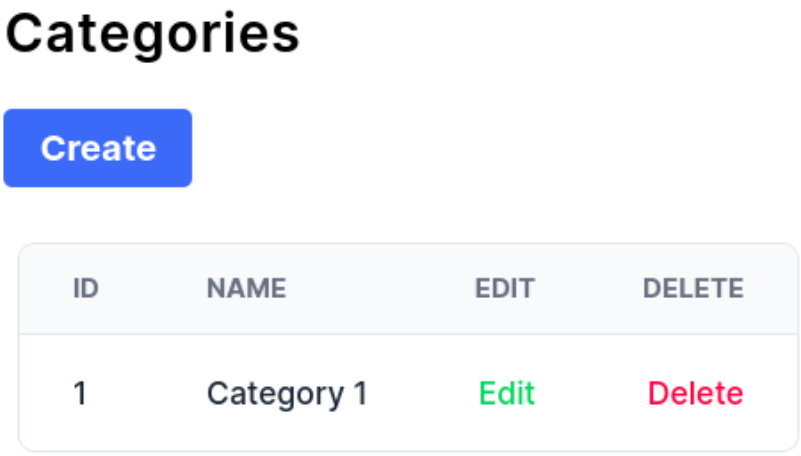
L'une des améliorations que je voudrai faire dans mon front-end est de créer un fichier .ts séparé qui s’appelle store, pour le traitement de tous les fonctions des api, comme ca dans mes composant j'appelle les fonctions, du coupe ca rend le code plus lisible.

Ecrans de mon app:






localhost:3001/categories/create
New Tab
Caerus - Google...
HSK DIGITAL AK...
Files · dev · Clie...
NODEJS
NestJs
React
Heroicons – Fre...
Medisys
Dashboard
Doctors
Patients
Categories
Logout
Create Category
Name
Category 1
Submit


Categories
Create
ID
NAME
EDIT
DELETE
1
Category 1
Edit
Delete

5. Conclusion

En conclusion, j'ai développé une application de gestion de clinique afin de simplifier le processus de prise de rendez-vous et de permettre aux utilisateurs de suivre facilement leur carnet de rendez-vous. J'ai utilisé une méthodologie de gestion de projet rigoureuse et des outils de gestion de projet performants pour mener à bien ce projet. J'ai identifié et géré les risques qui ont pu survenir au cours du développement de l'application. Le backend de l'application est maintenant opérationnel et offre toutes les fonctionnalités attendues, même si le code n'est pas encore parfaitement "clean". Le front-end de l'application est en cours de développement.

Il reste encore quelques fonctionnalités à mettre en place et de nombreux éléments à améliorer, mais je suis satisfait du travail accompli jusqu'à présent. J'espère que mon application sera utile aux professionnels de la santé et leur permettra de gagner du temps et de travailler de manière plus efficace. Je compte continuer à travailler sur mon application dans les prochaines semaines pour en améliorer les fonctionnalités et l'expérience utilisateur.