Logo

Description automatically generated

**Gestion des médecins**

**Alexis DUCHEMANN**

**SIO2B**

Sommaire

[Introduction 3](#_Toc99726958)

[Contexte 3](#_Toc99726959)

[Arborescence 3](#_Toc99726960)

[Structure demandée 3](#_Toc99726961)

[Technologies utilisées 4](#_Toc99726962)

[Modèle Conceptuel de Données 4](#_Toc99726963)

[Application 5](#_Toc99726964)

[Création d’une API / Récupération des médecins 5](#_Toc99726965)

[Connexion 5](#_Toc99726966)

[Navigation à travers le site 5](#_Toc99726967)

[Consultation des médecins, départements, pays 6](#_Toc99726968)

[Création des médecins / Modifications des médecins 6](#_Toc99726969)

# Introduction

## Contexte

Le laboratoire Galaxy Swiss Bourdin (GSB) est amené dans ses différentes activités à contacter les médecins installés, par exemple les visiteurs médicaux du laboratoire se déplacent afin de présenter les nouveaux produits pharmaceutiques. Une base de données de médecins est utilisée dans ces différents processus. Cette base de données évolue fréquemment, ceci à cause de départ à la retraite de médecins ou d’installation de nouveaux praticiens. Les applications doivent intégrer ces évolutions des données. Il a été décidé de confier à une société de services informatiques la responsabilité de développer deux applications donnant accès aux informations de la base de données.

Le projet a été réalisé en MVVM (moedel-view view-model). Pour ce faire j’ai utilisé React pour le front end et NodeJS pour le Back-end.

## Arborescence

Création d’un médecin

1



Page de connexion

Liste des pays

Liste des départements

Liste des médecins

Département par pays

Médecin par département

Table

Description automatically generated

Information d’un médecin

+ modification

## Structure demandée

## Technologies utilisées

## Icon Description automatically generated

app.listen("3001", () => {  
 ***console***.log("running on port 3001")  
})

app.listen("3001", () => {  
 ***console***.log("running on port 3001")  
})

Icon

Description automatically generated

`

Modèle Conceptuel de DonnéesDiagram

Description automatically generated

# Application

## Création d’une API / Récupération des médecins, départements etc…

Afin de récupérer les données de la base de données nous allons créer des requêtes permettant de les convertir et les afficher sur un site sur lequel nous récupérerons les données depuis notre application. Les données sont transformées en objet JSON.

Nous afficherons nos objets en localhost:3001 et pour ce faire nous devrons indiquer sur quel port s’afficheront les données.

app.listen("3001", () => {  
 ***console***.log("running on port 3001")  
})

Récupération des médecins de la base de données :

app.get('/medecins', async (req, res) => {  
 try{  
 const db = await getConnection();  
 const medecins = await db.query("SELECT medecin.id, nom, prenom, adresse, tel, specialiteComplementaire, libelle FROM medecin, departement WHERE medecin.departement = departement.id order by medecin.id; ");  
 res.send(medecins);  
 }catch(error){  
 ***console***.log("erreur")  
 };  
});

Affichage des données : (url = localhost:3001/medecins)

Text

Description automatically generated

Le fonctionnement restera le même pour les autres tables de la base de donnée

## Navigation à travers le site

Pour naviguer à travers l’application j’ai créé différentes routes amenant à différentes pages. Dans ReactJS les routes sont créés grâce à la dépendance BrowserRouter de react-router-dom.

function App(){  
  
 return (  
 <div className="App">  
  
 <Router>  
 <Navbar/>  
 <Routes>  
 <Route exact path="/medecins" element={<Medecins/>} />   
 <Route exact path="/medecininfo/:idMedecin" element={<MedecinInfo/>} />   
 <Route exact path="/departements" element={<Departements/>} />  
 <Route exact path="/pays" element={<Pays/>} />  
 <Route exact path="/login" element={<LoginPage/>} />  
 <Route exact path="/createmedecins" element={<CreateMedecins/>} />  
 <Route exact path="/modiftmededcins" element={<ModifyMedecins/>} />  
 </Routes>  
 </Router>  
 </div>  
 );  
}

Souhaitant garder la barre de navigation disponible tout le temps nous ne la mettons pas dans les routes mais dans le router.

## Consultation des médecins, départements, pays

* Affichage des donneés en tableau :

const columns = [

{ field: 'id', headerName: 'Identifiant', flex: 0.3 },

{ field: 'nom', headerName: 'Nom', flex: 0.5 },

{ field: 'prenom', headerName: 'Prénom', flex: 0.5 },

{ field: 'adresse', headerName: 'Adresse', flex: 1},

{ field: 'tel', headerName: 'Téléphone', flex: 0.5 },

{ field: 'specialiteComplementaire', headerName: 'Specialité Complémentaire', flex: 0.6 },

{ field: 'libelle', headerName: 'Département', flex: 0.5 },

{ field: "Modify",

renderCell: (cellValues) => {

const idMedecin = cellValues.row.id;

*return* (<Link className="modify" to={`/medecininfo/${idMedecin}`}>Modifier</Link>)}

},

{ field: "Delete",

renderCell: (cellValues) => {

const idMedecin = cellValues.row.id;

*return* (<Link className="delete" to={`/medecininfo/${idMedecin}`}>Supprimer</Link>)}

}

];

## Création des médecins / Modifications des médecins