Une image contenant texte, ciel nocturne

Description générée automatiquement

Framework mongoose

Afin de faciliter l’accès et les différentes opérations liées à la base MongoDB, nous allons utiliser un framework nommé [Mongoose](http://mongoosejs.com/" \t "_blank). Pour installer ce framework, rien de plus simple, il suffit d’entrer la commande dans votre console :

**npm install mongoose –save**

Nous utiliserons dans projetGo une base MongoDB Atlas et cela nous évite d’installer MongoDB sur notre poste.

Diagram

Description automatically generated

**Utilisation de Mongoose dans notre API**

Nous allons y ajouter quelques lignes de codes pour indiquer que nous allons utiliser le framework mongoose et ensuite nous établissons la connexion avec la base dans la page <dbConnect.js>

Text

Description automatically generated

Nous devons définir un schéma pour nos données. Mongoose utilise des schémas pour modéliser les données, on a créé la page <Membre.js> dans le dossier <Models> pour créer le modèle.

const MembreSchema = new mongoose.Schema

Text

Description automatically generated

On définit aussi dans la page <Membre.js> le nom de la Collection de notre base de données MongoDb Atlas.

*\*par default Mongoose ajoute une <s> a la fin de nom (Membre => Membres).*

Text

Description automatically generated

Maintenant que nous nous sommes connectés à la base, nous allons implémenter chacune des méthodes (GET, POST, PUT, DELETE) de notre API dans le dossier <api/membres> par son [ID].

Text

Description automatically generatedText

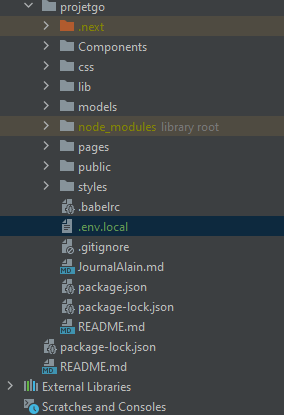
Description automatically generated

Qui sont appelées par notre formulaire <Form.js> dans le dossier <Components>

Text

Description automatically generated

*Pour référence : https://mongoosejs.com/docs/guide.html*

**Introduction a l'explorateur de fichier**

Le projet est divisé en plusieurs sections.

Le dossier racine de l'application est : projetgo

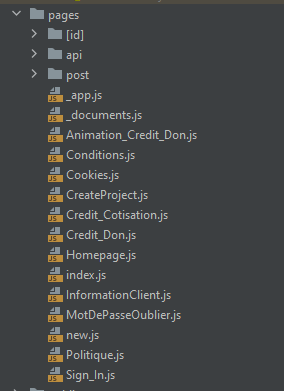
La barre de navigation et le form pour créer un nouveau membre sont dans le dossier Components.

Leurs CSS respectif sont dans le fichier css.  
  
Le dossier pages contient 2 sous fichier en plus d'une multitude de pages. Nous les verrons en détails plus tard.

Le dossier public comprend les images de toutes les pages internet.

Le Dossier Styles comprend le css de toutes ces pages.  
  
Le fichier .babelrc est relier au plugin Babel utiliser dans la barre de navigation.  
  
Le fichier .env.local permet de communiquer avec la base de donner.

**LE DOSSIER PAGES**

Il contient deux sous dossier  
  
le dossier api

et le dossier post.

Nous leur reviendrons plustard.

Le fichier Hompage.js est notre page mere ateignable apres la connexion qui redirige le client vers toutes les autres pages ou presque.

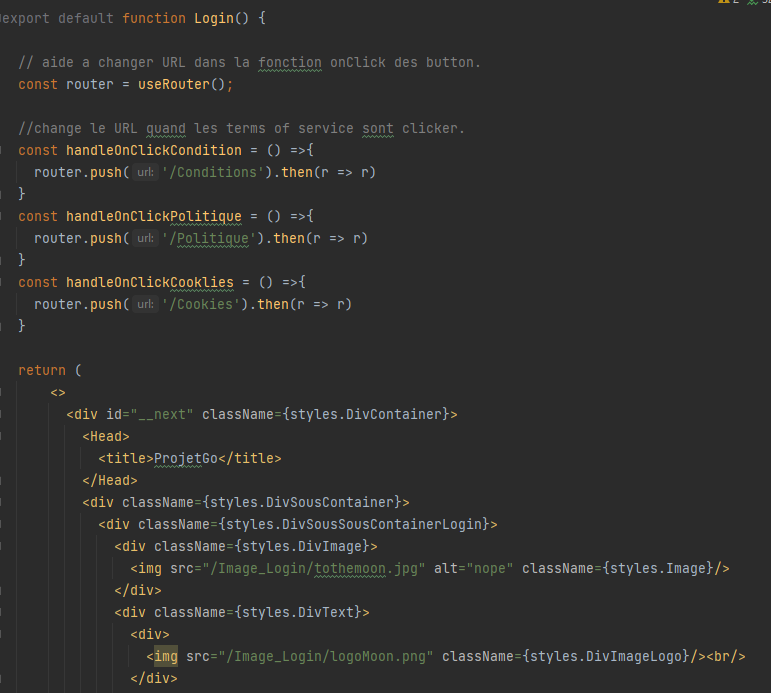
Le fichier index.js est la page par defaut.  
  
Nous l'utilison comme interface de connexion.

**Les fichier dans le dossier Pages**

Les fichier sont ecrits en javascrip.

Dans chaque fichier il y a une fonction principale (Login dans limage ci bas)

Nous pouvons inclure dautre fonction pour ajouter des fonctionaliters la page internet. Dans l'exemple ci bas, nous pouvons voir des handleOnClick qui mene a une nouvelle page avec des router.push.



La fonction principale retourne une page en JSX(html).

**Les fonctions asynchrones**

Il peut aussi y avoir une fonction asynchrone qui nous permet de parler avec le serveur.  
Prenons la page Homepage.js en exemple.

Ici nous voyons la fonction asynchrone :getServerSideProps(context)

Elle attend la promesse du client (le serveur) avant de répondre.

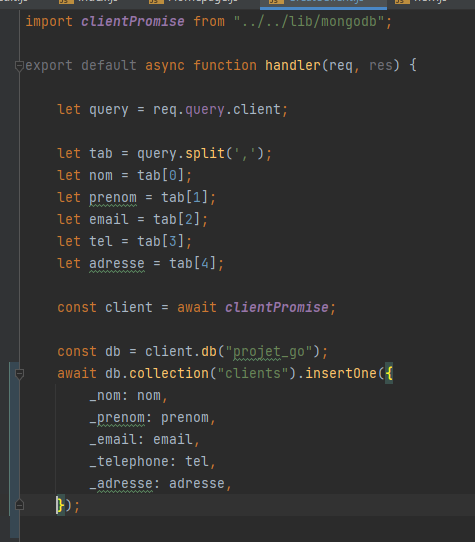
On appel ensuite la base de données avec cette promesse et on ajoute .bd("NomDeLaBaseDeDonner") a la variable qu'on a déclarer pour la promesse.

On envoi ensuite une requête, qui dans notre cas, prends toutes les instances de la collection "projets" de la base de donner et les mets dans un tableau dans notre programme.  
  
On doit ensuite mettre ce tableau en format JSON pour pouvoir le retourner comme argument a la fonction principale. Il faut faire ici abstraction de l'argument membre car il est tiré à partir de la page de connexion.



**Le dossier API**

Il contient tous les fichier javascript qui font que des requêtes au serveur.  
Regardons ici le fichier CreateClient.js



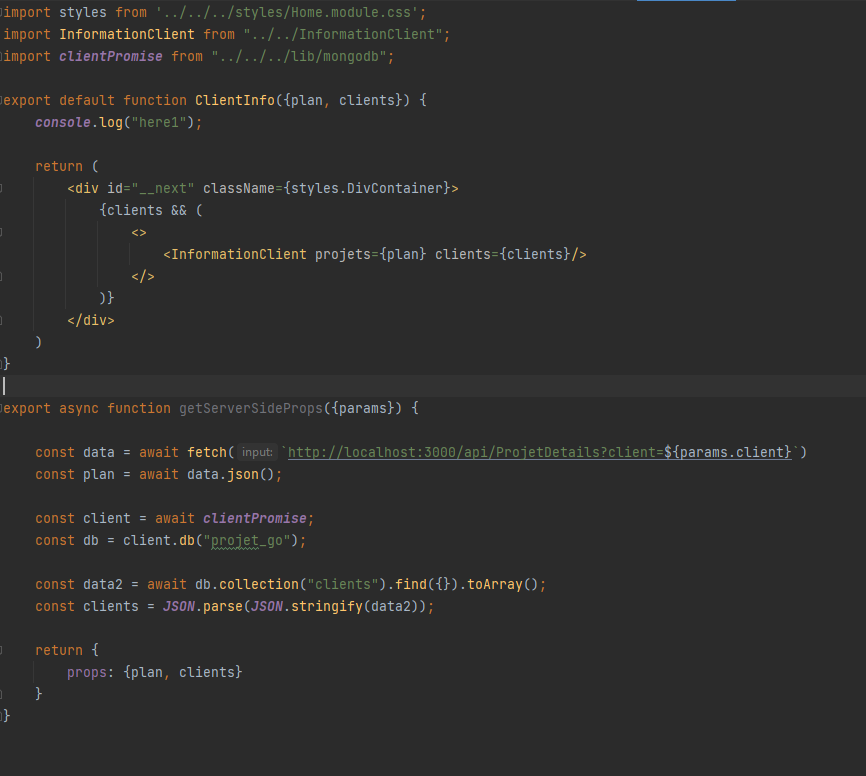
Il prend le fetch d'un autre fichier (voir dossier Post).

Le fetch est un long string qu'on doit séparer en plusieurs variables

Par la suite on fait une requête comme préciser dans la section asynchrone.

**Le dossier POST**

Chaque dossier à l’intérieur du dossier post utilise un chemin diffèrent. Il est très important de mettre chaque fichier dans un dossier diffèrent, car il pourrait y avoir des erreurs de chemin dans le cas contraire.



Il est l'intermédiaire entre le front end et le back end pour envoyer les objets d'une page l'autre.

Il passe des objets en argument afin d'appeler les pages visiter par l'utilisateur.

GLHF