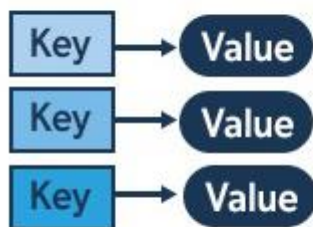
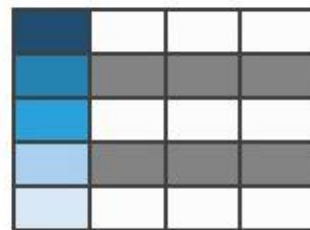


PROJET NO SQL

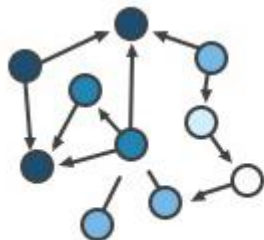
Key-Value



Column-Family



Graph



Document



Table des matières

| | |
|---|----|
| Modalité d'évaluation détaillées | 3 |
| Mise en place de la structure avec docker | 7 |
| a. Authentification et gestion des utilisateurs : | 8 |
| Permettre aux utilisateurs de s'inscrire et de se connecter | 8 |
| Développement du backend : | 8 |
| Développement du frontend | 13 |
| b. Gestion des projets : création, modification, suppression, et visualisation des projets. | 24 |
| Développement sur le backend | 24 |
| Développement sur le frontend | 25 |
| d. Gestion des membres de l'équipe : inviter des membres, définir des rôles, attribuer des tâches et suivre la progression..... | 27 |
| Développement sur le frontend | 27 |
| g. Synchronisation avec le calendrier : intégration avec Google Calendar API. | 30 |

LIEN GITHUB POUR LE DOCKER FILE :

https://github.com/salmanwk34/PROJET_NOSQL_ARTZ

(Branche master qui sert de dev)

LIEN GITHUB DU CONTENEUR BACKEND :

https://github.com/salmanwk34/NOSQL_BACKEND

LIEN GITHUB DU CONTENEUR FRONTEND :

https://github.com/salmanwk34/NOSQL_FRONTEND

Modalité d'évaluation détaillées

Présentation du contexte

Aujourd'hui, de nombreuses entreprises et particuliers sont confrontés à la gestion de projets et de tâches multiples.

Ils ont besoin d'un outil simple et efficace pour les aider à suivre leurs activités, à planifier leurs tâches et à gérer leur temps.

Dans ce contexte, l'application de gestion de tâches et de projets a pour objectif de répondre à ce besoin en offrant une plateforme accessible et facile à utiliser pour gérer et organiser les tâches et les projets.

Justification des méthodes et outils utilisés

Base de données NoSQL : MongoDB

Une base de données NoSQL est choisie pour sa flexibilité et sa facilité de mise à l'échelle.

Elle permet de stocker les données des utilisateurs, des projets, des tâches et des relations entre eux sous forme de documents.

Cela facilite les modifications de la structure de données sans perturber l'ensemble de l'application.

Framework Web et mobile (ex. : Angular ; React Native, Flutter pour mobile) :

Nous allons utiliser Angular car c'est une solution complète contrairement à REACT

Ces Framework sont utilisés pour créer une interface utilisateur intuitive et réactive pour l'application, accessible sur différents appareils

Ils permettent également de faciliter le développement en réutilisant des composants et en suivant des bonnes pratiques.

Backend (ex. : Node.js avec Express) :

Un serveur backend est nécessaire pour gérer la logique métier, l'authentification des utilisateurs et la communication entre la base de données et les clients (applications Web et mobile).

Node.js est choisi pour sa performance et sa facilité d'intégration avec les bases de données NoSQL et les frameworks Web et mobile.

API de calendrier (ex. : Google Calendar API) :

L'API de calendrier sera utilisée pour synchroniser les tâches de l'application avec le calendrier personnel de l'utilisateur. (ECHEC DE LA MISE EN PLACE) mais mon app.js utilise Express en tant qu'API

Cela permet aux utilisateurs de suivre leurs tâches à partir d'un seul endroit et d'éviter les conflits entre les différentes plateformes.

Gestion de version du code (ex. : Git) :

Git est utilisé pour versionner le code source du projet, permettant ainsi de suivre les modifications, de collaborer avec d'autres développeurs et de revenir facilement à une version antérieure en cas de problème.

Les connaissances mobilisées

Le projet nécessite des compétences en développement web et mobile, gestion de bases de données NoSQL, intégration d'API et développement d'algorithmes.

Le choix des outils et des technologies appropriés démontre la maîtrise des connaissances requises.

Démontrer les fonctionnalités mises en place

Les fonctionnalités principales incluent la gestion des projets et des tâches, la synchronisation avec un calendrier externe, la priorisation et l'ordonnancement des tâches.

Ces fonctionnalités seront démontrées à travers des captures d'écran, des schémas et des démonstrations en direct de l'application.

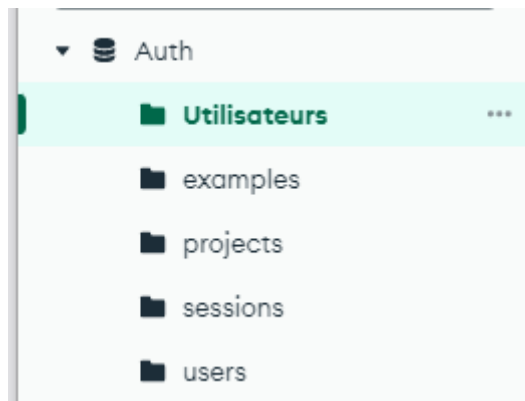
Décrire l'architecture du site

L'architecture du projet peut être décrite en termes de frontend (web et mobile), backend (serveur) et base de données.

| | | | | | | | |
|---------------|---------------|--|---------------|----------------|----------------|---|---|
| nosql_project | | - | Running (3/3) | 46 minutes ago | ⌵ | ⋮ | 🗑 |
| backend-1 | e1547daa412b | nosql_project-backend | Running | 3000.3000 | 46 minutes ago | ⌵ | 🗑 |
| frontend-1 | d7b2a5ac3683 | nosql_project-frontend | Running | 4200.4200 | 7 hours ago | ⌵ | 🗑 |
| mongodb-1 | bba1f693b4573 | mongo:latest | Running | 27017.27017 | 7 hours ago | ⌵ | 🗑 |

Décrire la structure de la base de données :

La structure de la base de données NoSQL sera présentée en détaillant les collections (ou tables) et les documents (ou enregistrements) utilisés pour stocker les données des utilisateurs, des projets et des tâches



Créer ou intégrer une API

L'API de calendrier (par exemple, Google Calendar API) sera intégrée pour permettre la synchronisation des tâches avec le calendrier personnel de l'utilisateur.

Pour combler l'échec, j'utilise déjà mon backend avec Express en tant qu'API

Utiliser une base de données NoSQL

Une base de données NoSQL (comme MongoDB) sera utilisée pour stocker les données du projet.

Mettre en place des fonctionnalités mettant en avant vos compétences en algorithmie

Algorithme sur la gestion des erreurs sur angular démontre cette compétence

Mettre en place son environnement de développement et de production

L'environnement de développement et de production sera configuré en utilisant des outils appropriés pour la gestion des dépendances, le déploiement et la surveillance de l'application.

Pour explication j'ai utilisé un système très spécial, j'ai installé un conteneur avec le mode développement de angular, puis avec les commandes suivantes je pouvais construire mes configurations et mettre en prod

```
http-server /usr/src/app/dist/frontend -p 4200
```

ng build --configuration=production

Versionner son code

Utilisation de Git pour le docker-compose

Respecter la commande

Terminé plusieurs mois avant le deadline

Respecter les dates de rendu

Terminé plusieurs mois avant le deadline

Définir et appliquer une méthodologie de travail adaptée à la commande

Méthode de développement Agile

Adopter une posture professionnelle dans le cadre d'une production

Utilisation de docker-compose / virtualisation comme dans une réelle production

Respecter les règles orthographiques et de grammaire

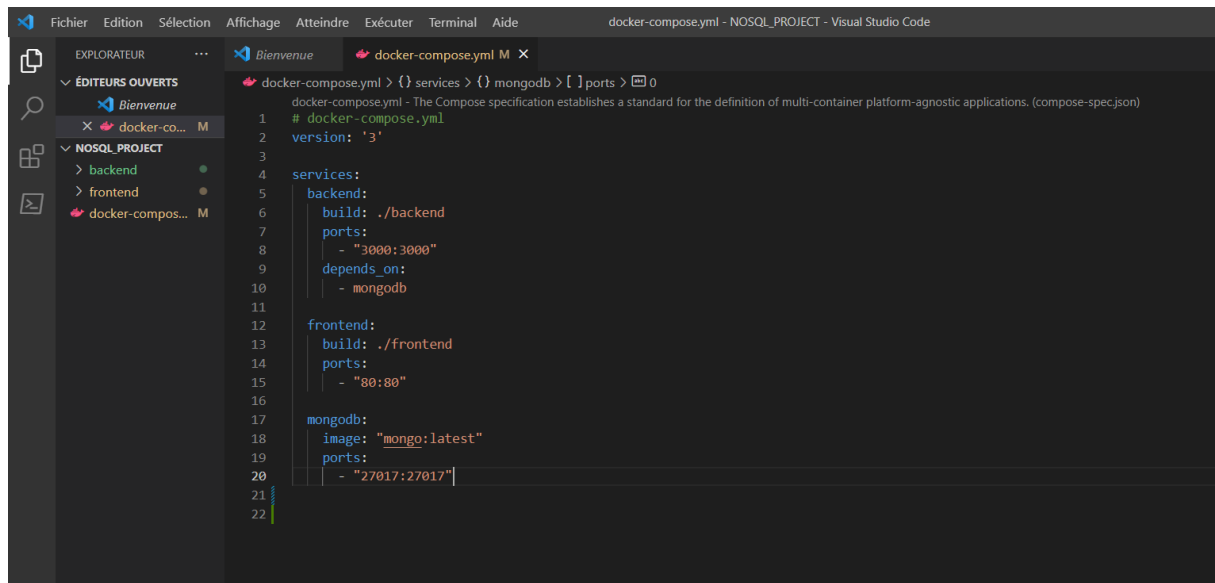
Ok

Commentez votre code et votre projet

Mon code n'est pas entièrement commenté car je développe directement depuis le conteneur

















Mise en place de la structure avec docker

On utilisera docker-compose pour gérer notre application multi-conteneurs



```
1 docker-compose.yml > {} services > {} mongodb > [ ] ports > 0
2 docker-compose.yml - The Compose specification establishes a standard for the definition of multi-container platform-agnostic applications. (compose-spec.json)
3 # docker-compose.yml
4 version: '3'
5
6 services:
7   backend:
8     build: ./backend
9     ports:
10      - "3000:3000"
11     depends_on:
12      - mongodb
13
14   frontend:
15     build: ./frontend
16     ports:
17      - "80:80"
18
19   mongodb:
20     image: "mongo:latest"
21     ports:
22      - "27017:27017"
```

Voici l'infra sous docker

| | | | | | |
|--------------------------|---|--|---------------|--|---|
| <input type="checkbox"/> |  nosql_project | - | Running (3/3) | 1 second ago |    |
| <input type="checkbox"/> |  frontend-1 4b2d861e309a | nosql_project-frontend | Running | 80:80 1 second ago |    |
| <input type="checkbox"/> |  mongodb-1 bbaf693b4573 | mongo:latest | Running | 27017:27017 1 second ago |    |
| <input type="checkbox"/> |  backend-1 f3befb746574 | nosql_project-backend | Running | 3000:3000 1 second ago |    |

Voici le repository sous une branche master qui nous servira de dev

salmanwk34 / PROJET_NOSQL_ARTZ Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master had recent pushes 19 minutes ago [Compare & pull request](#)

master 2 branches 0 tags [Go to file](#) [Add file](#) [Code](#)

This branch is 1 commit ahead, 1 commit behind main. [Contribute](#)

| salmanwk34 V1 | | |
|--------------------|----|----------------|
| backend | V1 | 19 minutes ago |
| frontend | V1 | 19 minutes ago |
| docker-compose.yml | V1 | 19 minutes ago |

Help people interested in this repository understand your project by adding a README. [Add a README](#)

About
No de

Relea
No rele
[Create](#)

Pack:
No pac
[Publish](#)

a. Authentification et gestion des utilisateurs :

Permettre aux utilisateurs de s'inscrire et de se connecter

Développement du backend :

Mise en place l'authentification des utilisateurs en utilisant le package Passport.js.

On installe d'abord les paquets

```
# npm install passport

added 3 packages, and audited 61 packages in 2s

8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm notice
npm notice New patch version of npm available! 9.6.4 -> 9.6.5
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.6.5
npm notice Run npm install -g npm@9.6.5 to update!
npm notice
```



```
# npm install passport-local

added 1 package, and audited 62 packages in 850ms

8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
■
```

Pour le hachage de mot de passe

```
# npm install bcryptjs

added 1 package, and audited 63 packages in 725ms

8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
■
```

Maintient de la connexion entre le client / serveur

```
# npm install express-session

added 5 packages, and audited 68 packages in 3s

8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Stockages donnés sur dans notre BDD

```
# npm install connect-mongo

added 23 packages, and audited 91 packages in 8s

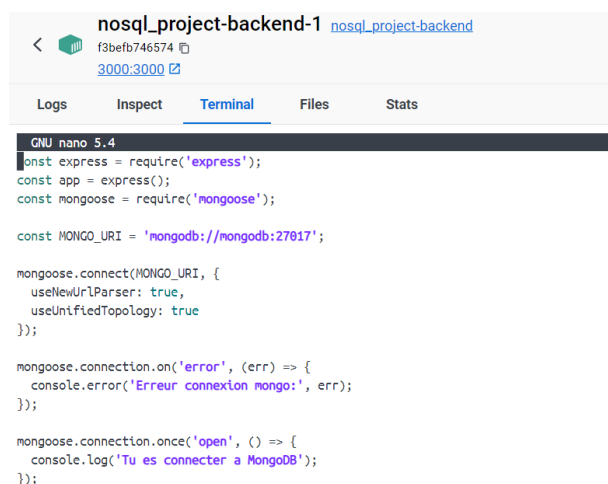
8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
.. ■
```

On hash le mot de passe que l'on souhaite

```
found 0 vulnerabilities
# # npm fund
/bin/sh: 23: npm: not found
# node hash_password.js
Le mot de passe hache est : $2b$10$WpaN8AnHHPD3778wP/yY1.H1uiOCmABVWmqserYwxGOQ3dqKy06Cq
# ■
```

On se connecte à la BDD pour l'instant sans authentification



The screenshot shows a web application interface for a project named "nosql_project-backend-1". The interface includes a header with a back arrow, a logo, the project name, and a link to "nosql_project-backend". Below the header, there are tabs for "Logs", "Inspect", "Terminal", "Files", and "Stats". The "Terminal" tab is active, displaying a terminal window with the following code:

```
GNU nano 5.4
const express = require('express');
const app = express();
const mongoose = require('mongoose');

const MONGO_URI = 'mongodb://mongodb:27017';

mongoose.connect(MONGO_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true
});

mongoose.connection.on('error', (err) => {
  console.error('Erreur connexion mongo:', err);
});

mongoose.connection.once('open', () => {
  console.log('Tu es connecter a MongoDB');
});
```

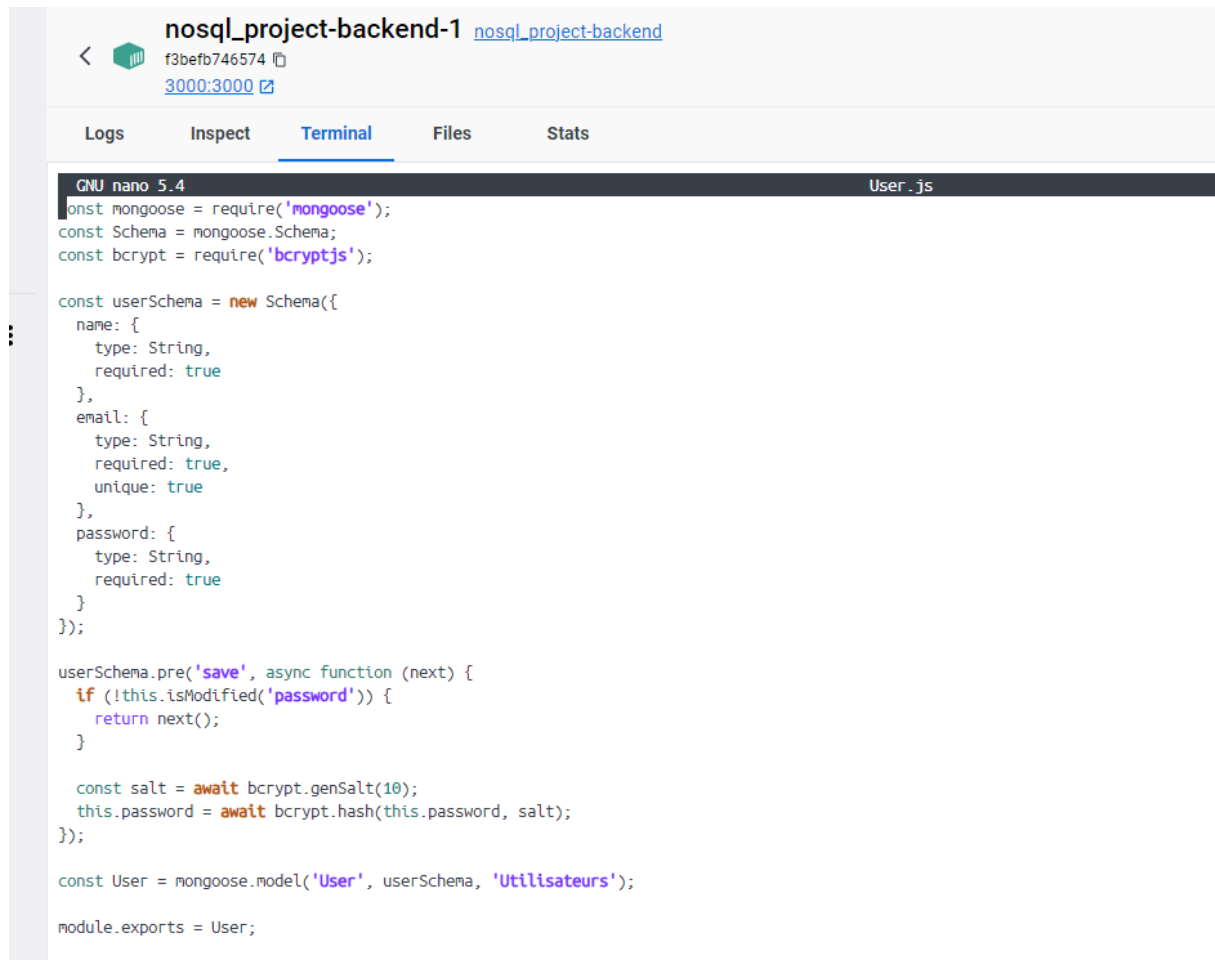
Pour savoir si mon backend se connecte bien au serveur mongoDB, il retournera « Tu es connecter a MongoDB »

Voici le message qui le confirme dans les logs

```
2023-04-21 16:40:01 App listening at http://localhost:3000
2023-04-21 16:42:47 Server started on port 5000
2023-04-26 13:40:20 Server started on port 5000
2023-04-26 13:44:22 Server started on port 5000
2023-04-26 13:57:47 Server started on port 5000
2023-04-26 14:04:40 Server is running on port 3000
2023-04-26 14:04:40 Tu es connecter a MongoDB )
```

Nous allons maintenant crée un utilisateur dans notre base de données NoSQL.

Nous avons ci-dessous User.js qui définit un schéma mongoose pour la collection Utilisateurs avec pour champ name, email et password



```
nosql_project-backend-1 nosql_project-backend
f3befb746574 3000:3000

Logs Inspect Terminal Files Stats

GNU nano 5.4 User.js
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const bcrypt = require('bcryptjs');

const userSchema = new Schema({
  name: {
    type: String,
    required: true
  },
  email: {
    type: String,
    required: true,
    unique: true
  },
  password: {
    type: String,
    required: true
  }
});

userSchema.pre('save', async function (next) {
  if (!this.isModified('password')) {
    return next();
  }

  const salt = await bcrypt.genSalt(10);
  this.password = await bcrypt.hash(this.password, salt);
});

const User = mongoose.model('User', userSchema, 'Utilisateurs');
module.exports = User;
```

Puis nous avons createUser.js qui utilise le modèle du fichier User.js et qui permet de crée un utilisateur.

```
nosql_project-backend-1 nosql_project-backend
f3befb746574 3000:3000

Logs Inspect Terminal Files Stats

GNU nano 5.4 createUser.js
const mongoose = require('mongoose');
const User = require('./User');
const bcrypt = require('bcryptjs');

const MONGO_URI = 'mongodb://mongodb:27017/Auth';

mongoose.connect(MONGO_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

mongoose.connection.on('error', (err) => {
  console.error('Erreur connexion mongo:', err);
});

mongoose.connection.once('open', async () => {
  console.log('Tu es connecter a MongoDB');

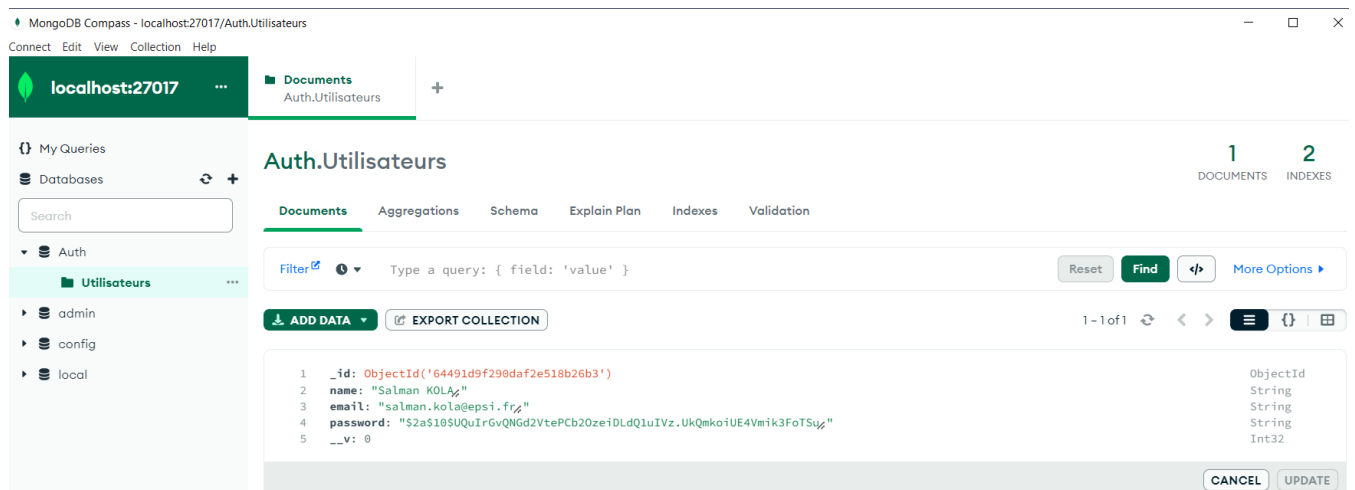
  const newUser = new User({
    name: 'Salman KOLA',
    email: 'salman.kola@epsi.fr',
    password: 'bachelor'
  });

  try {
    const salt = await bcrypt.genSalt(10);
    newUser.password = await bcrypt.hash(newUser.password, salt);
    await newUser.save();
    console.log('Utilisateur cree avec succes!');
  } catch (err) {
    console.error('Erreur lors de la creation de lutilisateur:', err);
  } finally {
    mongoose.connection.close();
  }
});
```

Il se connecte à MongoDB puis crée l'utilisateur.

```
# node createUser.js
Tu es connecter a MongoDB
Utilisateur cree avec succes!
# ls
User.js  createUser.js
```

On se connecte à l'aide de Compass pour vérifier, on voit notre utilisateur dans la collection Utilisateurs avec un mot de passe chiffrer



Développement du frontend

CETTE METHODE A ETE CHANGER

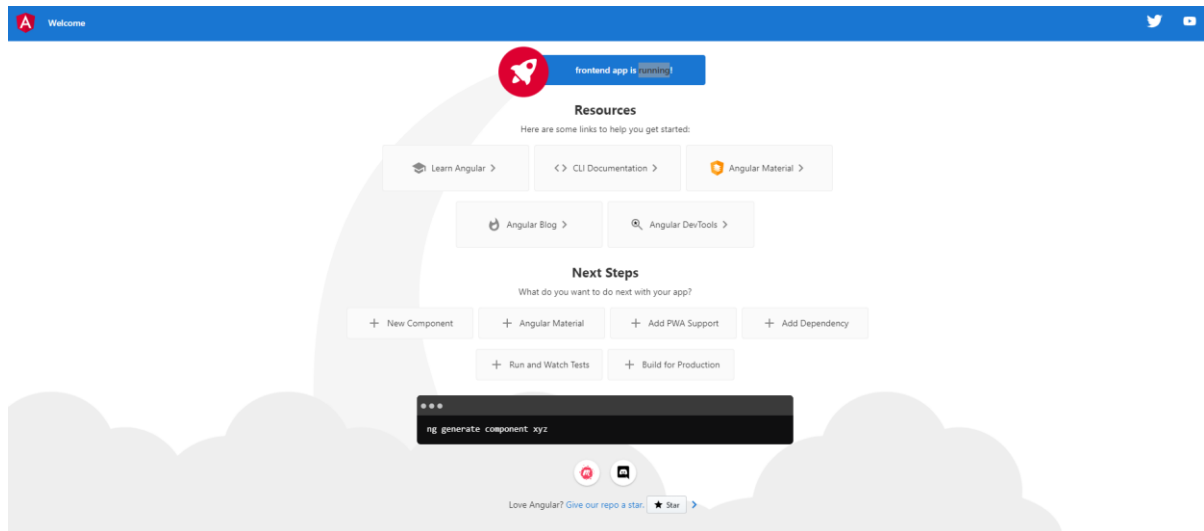
On va maintenant mettre le frontend, on commence par le dockerfile :

```

frontend > dockerfile > ...
1 FROM node:latest as build
2
3 WORKDIR /usr/src/app
4
5 COPY package*.json ./
6
7 RUN npm install
8
9 COPY . .
10
11 RUN npm run build
12
13 # Étape de production
14 FROM nginx:1.17.1-alpine
15
16 COPY --from=build /usr/src/app/dist/frontend /usr/share/nginx/html
17
18 EXPOSE 80
19
20 CMD ["nginx", "-g", "daemon off;"]
21

```

Angular se servira de NGINX pour se déployer, ce qui donne cela :



J'ai finalement utilisé ce dockerfile, j'ai appris sur ce projet comment me servir de angular.

J'ai tout simplement mis en place angular CLI afin d'avoir mon environnement de test.

```
frontend > dockerfile > ...
1 FROM node:latest
2
3 WORKDIR /usr/src/app
4
5 COPY package*.json ./
6
7 RUN npm install -g @angular/cli@15.2.7
8 RUN npm install
9
10 COPY . .
11
12 RUN ng g c signup
13 RUN ng g c dashboard
14 RUN ng g c home
15 RUN ng generate service api
16 RUN ng generate service login
17
18 EXPOSE 4200
19
20 CMD ["npm", "start"]
```

Une fois que je modifie les composants en faisant mes propres configurations avec nano

```
GNU nano 5.4 login.component.ts
import { Component, OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { AuthService } from '../auth.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html'
})
```

Je vérifie que les fichiers sont corrects dans les logs

```
PROBLÈMES  SORTIE  CONSOLE DE DÉBOGAGE  TERMINAL
nosql_project-frontend-1 | polyfills.js | polyfills | 314.27 kB |
nosql_project-frontend-1 | styles.css, styles.js | styles | 210.27 kB |
nosql_project-frontend-1 | main.js | main | 30.91 kB |
nosql_project-frontend-1 | runtime.js | runtime | 6.51 kB |
nosql_project-frontend-1 | | Initial Total | 2.96 MB
nosql_project-frontend-1 | Build at: 2023-05-11T14:48:52.103Z - Hash: e79ac11373fbfff6 - Time: 8199ms
nosql_project-frontend-1 | ** Angular Live Development Server is listening on 0.0.0.0:80, open your browser on http://localhost:80/ **
nosql_project-frontend-1 | ✓ Compiled successfully.
```

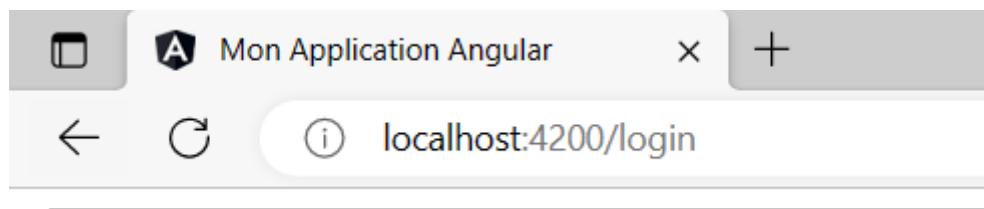
Je compile ensuite à l'aide de « ng build --configuration=production »

```
# ng build --configuration=production
✓ Browser application bundle generation complete.
✓ Copying assets complete.
✓ Index html generation complete.
```

| Initial Chunk Files | Names | Raw Size | Estimated Transfer Size |
|-------------------------------|-----------|-----------|-------------------------|
| main.d12fe441b7e6038b.js | main | 237.34 kB | 61.79 kB |
| polyfills.794d7387aea30963.js | polyfills | 33.09 kB | 10.63 kB |
| runtime.89bfad0fe920d2c9.js | runtime | 894 bytes | 518 bytes |
| styles.9f9538d0369d9e75.css | styles | 256 bytes | 131 bytes |
| Initial Total | | 271.55 kB | 73.05 kB |

Puis je démarre mon serveur http local en utilisant http-server /usr/src/app/dist/frontend -p 4200

Ce qui me permet de le voir en direct



On va maintenant pouvoir s'occuper du frontend

On a notre index.html

```
GNU nano 5.4 index.html *
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Gestion de projet</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Un fichier .css qui est rediriger ailleurs

```
GNU nano 5.4 styles.css
@import './app/app.component.css';
```

Nos composants

```
# ls
api.service.spec.ts  app-routing.module.ts  app.component.html  app.component.ts  auth.service.spec.ts  dashboard  login  login.service.ts  signup
api.service.ts       app.component.css      app.component.spec.ts  app.module.ts      auth.service.ts       home      login.service.spec.ts  project  task
```

Voici notre navbar pour l'instant

```
GNU nano 5.4 app.component.html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Mon Application Angular</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <nav class="navbar">
    <ul>
      <li><a routerLink="/">Accueil</a></li>
      <li><a routerLink="/login">Se connecter</a></li>
      <li><a routerLink="/signup">S'inscrire</a></li>
      <li><a routerLink="/dashboard">Tableau de bord</a></li>
    </ul>
  </nav>

  <main>
    <router-outlet></router-outlet>
  </main>
</body>
</html>
```

Nos paramètres de routes sont gérés ici

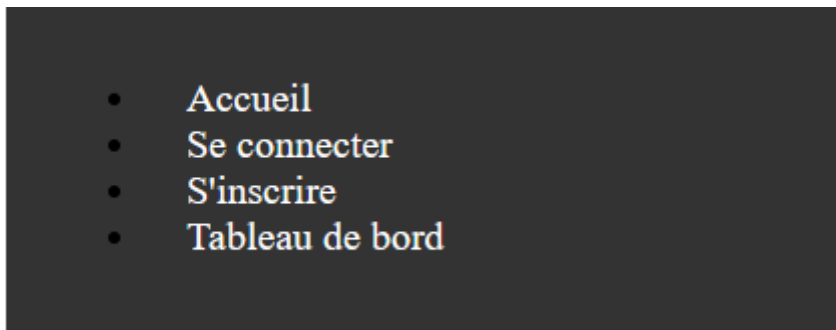

```
GNU nano 5.4 app-routing.module.ts *
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { HomeComponent } from './home/home.component';
import { SignupComponent } from './signup/signup.component';
import { LoginComponent } from './login/login.component';
import { DashboardComponent } from './dashboard/dashboard.component';
import { ProjectComponent } from './project/project.component';
import { TaskComponent } from './task/task.component';

const routes: Routes = [
  { path: 'home', component: HomeComponent },
  { path: 'signup', component: SignupComponent },
  { path: 'login', component: LoginComponent },
  { path: 'dashboard', component: DashboardComponent },
  { path: 'project', component: ProjectComponent },
  { path: 'task', component: TaskComponent },
  { path: '', redirectTo: '/login', pathMatch: 'full' }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Ce qui donne



Exemple pour login :

Composé de 3 fichiers 1 en html

C'est le fichier qui gère l'affichage de ma rubrique login

```
GNU nano 5.4 login.component.html
<form (ngSubmit)="login()">
  <div class="form-group">
    <label for="email">Email</label>
    <input type="email" id="email" [(ngModel)]="model.email" name="email" required>
  </div>
  <div class="form-group">
    <label for="password">Mot de passe</label>
    <input type="password" id="password" [(ngModel)]="model.password" name="password" required>
  </div>
  <button type="submit">Se connecter</button>
</form>
<p>{{ message }}</p>
```

1 fichier ccs qui gère le style de ma page

```
GNU nano 5.4 login.component.css
body {
  background-color: #f9f9f9;
}

.login-container {
  width: 300px;
  padding: 16px;
  background-color: white;
  margin: 0 auto;
  margin-top: 50px;
  border-radius: 8px;
  box-shadow: 0px 0px 10px 0px #0000001a;
}

.login-container h1 {
  text-align: center;
  color: #333333;
}

.login-container input[type="email"],
.login-container input[type="password"] {
  width: 100%;
  padding: 10px;
  margin: 10px 0;
  border-radius: 5px;
  border: 1px solid #ddd;
  box-sizing: border-box;
}
```

Puis le dernier fichier qui gère les configurations de la page qui se connecte a l'api express du backend

```
GNU nano 5.4 login.component.ts
selector: 'app-login',
templateUrl: './login.component.html',
styleUrls: ['./login.component.css']
})
export class LoginComponent {

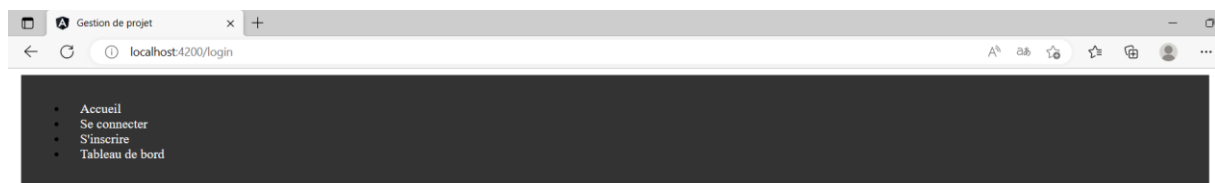
  model: any = {};
  message: string = '';

  constructor(private http: HttpClient) { }

  ngOnInit(): void {
  }

  login(): void {
    this.http.post('http://localhost:3000/login', this.model).subscribe(
      (response) => {
        console.log('Connexion reussie', response);
        this.message = 'Connexion reussie';
        this.model = {};
        // Vous pouvez naviguer vers une autre page ici si vous le souhaitez, par exemple:
        // this.router.navigate(['/dashboard']);
      },
      (error) => {
        console.error('Erreur lors de la connexion', error);
        this.message = 'Erreur lors de la connexion';
      }
    );
  }
}
```

Ce qui donne la page ci-dessous



Connexion

Email

Mot de passe

Se connecter

Voici pour signup

Search for local and remote images, containers, and more...

Ctrl+K

nosql_project-frontend-1

nosql_project-frontend

d7b2a5ac3683

4200:4200

Logs

Inspect

Terminal

Files

Stats

GNU nano 5.4

signup.component.css *

```
signup-form {
  max-width: 500px;
  margin: 0 auto;
  padding: 20px;
  border: 1px solid #ddd;
  border-radius: 5px;
  background-color: #fff;
  box-shadow: 0px 0px 10px 0px rgba(0,0,0,0.1);
}

.signup-form h1 {
  font-size: 24px;
  text-align: center;
  color: #333;
  margin-bottom: 20px;
}

.signup-form .form-group {
  margin-bottom: 15px;
}

.signup-form .form-group label {
  font-size: 18px;
  color: #333;
}

.signup-form .form-group input {
  width: 100%;
```

Le fichier html

nosql_project-frontend-1

nosql_project-frontend

d7b2a5ac3683

4200:4200

STATUS
Running

Logs

Inspect

Terminal

Files

Stats

GNU nano 5.4

signup.component.html *

```
<div class="signup-form">
  <h1>Inscription</h1>
  <form (ngSubmit)="register()" #registerForm="ngForm">
    <div class="form-group">
      <label for="username">Nom:</label>
      <input type="text" class="form-control" id="username" [(ngModel)]="model.username" name="username" required>
    </div>
    <div class="form-group">
      <label for="email">Email:</label>
      <input type="email" class="form-control" id="email" [(ngModel)]="model.email" name="email" required>
    </div>
    <div class="form-group">
      <label for="password">Mot de passe:</label>
      <input type="password" class="form-control" id="password" [(ngModel)]="model.password" name="password" required>
    </div>
    <button type="submit" class="btn btn-primary">S'inscrire</button>
  </form>
  <p class="message" *ngIf="message">{{ message }}</p>
</div>
```

Et le fichier de conf de l'onglet.

```
GNU nano 5.4                                signup.component.ts
import { Component, OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-signup',
  templateUrl: './signup.component.html',
  styleUrls: ['./signup.component.css']
})
export class SignupComponent implements OnInit {

  model: any = {};
  message: string = '';

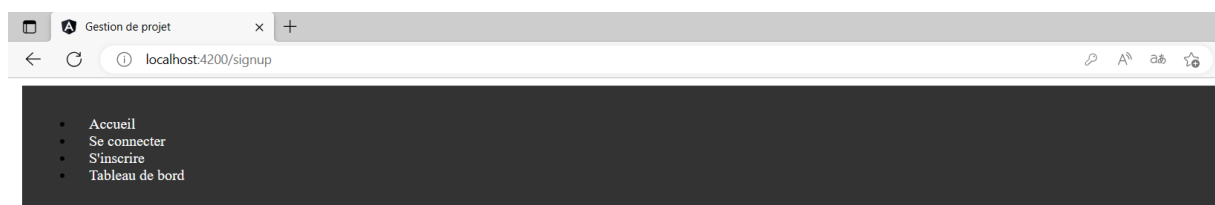
  constructor(private http: HttpClient) { }

  ngOnInit(): void {
  }

  register(): void {
    this.http.post('http://localhost:3000/register', this.model).subscribe(
      (response) => {
        console.log('Inscription reussie', response);
        this.message = 'Inscription reussie !';
        this.model = {};
      },
      (error) => {
        console.error('Erreur lors de l\'inscription', error);
        this.message = 'Erreur lors de l\'inscription';
      }
    );
  }
}
```

[Read 33 lines]

Ce qui donne :



Inscription

Nom:

Email:

Mot de passe:

S'inscrire

On va maintenant conditionner l'affichage en fonction des états d'authentification, on va donc effectuer quelques modifications sur de fichier existants, lorsque je me connecte avec mon utilisateur.

Sur le login je veux qu'il me redirige seulement sur tableau de bord pour l'instant donc on met une roue

```
GNU nano 5.4 login.component.ts *
})
export class LoginComponent {

  model: any = {};
  message: string = '';

  constructor(private http: HttpClient, private authService: AuthService, private router: Router) { }

  ngOnInit(): void {
  }

  login(): void {
    this.http.post('http://localhost:3000/login', this.model).subscribe(
      (response: any) => {
        console.log('Connexion reussie', response);
        this.message = 'Connexion reussie';
        this.authService.setUserInfo(response); // Ajout de cette ligne
        this.model = {};
        this.router.navigate(['/dashboard']); // Naviguer vers le tableau de bord apres connexion r        ussie
      },
      (error) => {
        console.error('Erreur lors de la connexion', error);
        this.message = 'Erreur lors de la connexion';
      }
    );
  }
}
```

On va changer le path par défaut donc sur login

```
{ path: '', redirectTo: '/login', pathMatch: 'full' }
```

Sur le fichier de conf on vise bien notre fonction + on rajoute un bouton se déconnecter

```
<li><a routerLink="/dashboard" *ngIf="authService.isLoggedIn()">Tableau de bord</a></li>
</ul>
<button *ngIf="authService.isLoggedIn()" (click)="logout()">Se deconnecter</button>
</div>
```

En parlant de fonction voici celle qui nous permet de stocker les utilisateurs (équivalent de cookies)
C'est pour savoir quel utilisateur est connecté par exemple.

Voici le code :

```
GNU nano 5.4 auth.service.ts
providedIn: 'root'
})
export class AuthService {

  private loginUrl = 'http://localhost:3000/login';

  constructor(private http: HttpClient) { }

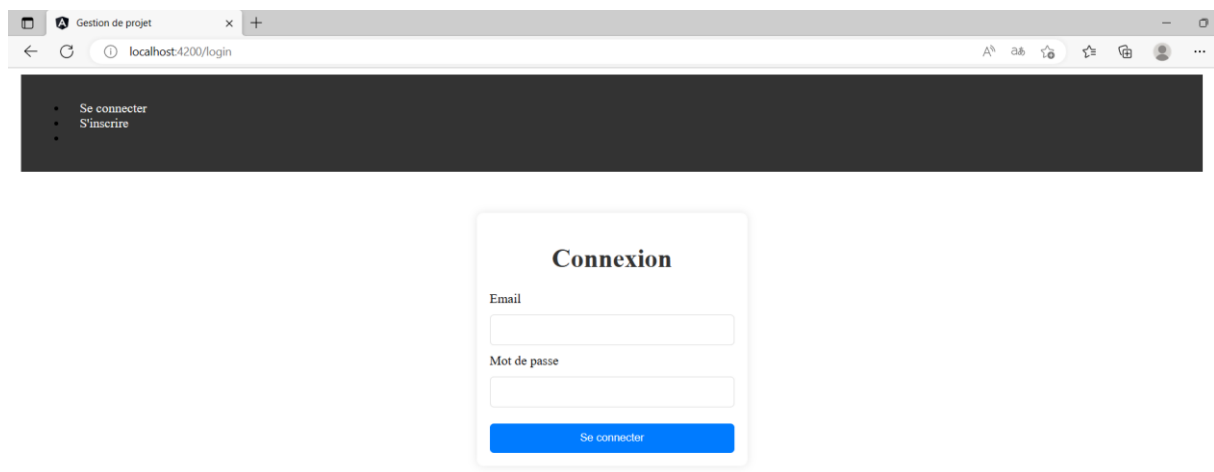
  onLogin(user: any) {
    return this.http.post<any>(this.loginUrl, user);
  }

  // Stocker l'information de l'utilisateur connect dans le localStorage
  setUserInfo(user: any) {
    localStorage.setItem('user', JSON.stringify(user));
  }

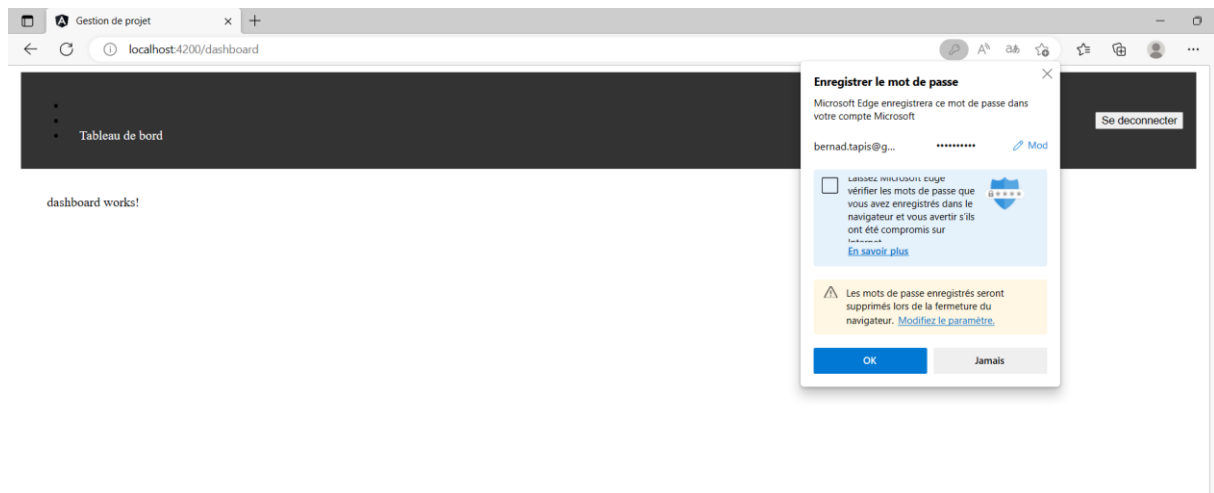
  // Verif si l'utilisateur est connect
  isLoggedIn() {
    return localStorage.getItem('user') !== null;
  }

  // Supprimer l'information de l'utilisateur connect localStorage
  logout() {
    localStorage.removeItem('user');
  }
}
```

Donc là on a seulement la possibilité de se connecter et de s'inscrire.



Puis une fois connecté, on a seulement le bouton « Se déconnecter » et les menus.



b. Gestion des projets : création, modification, suppression, et visualisation des projets.

Développement sur le backend

On commence par créer le modèle pour notre nouvelle collection dans la BDD

```
nosql_project-backend-1 nosql_project-backend
e1547daa412b
3000:3000

Logs Inspect Terminal Files Stats

GNU nano 5.4 project.model.js *
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

let ProjectSchema = new Schema({
  name: {type: String, required: true, max: 100},
  description: {type: String, required: true},
});

module.exports = mongoose.model('Project', ProjectSchema);
```

On crée ensuite des routes

```
GNU nano 5.4
const express = require('express');
const router = express.Router();

const project_controller = require('../controllers/project.controller');

router.post('/create', project_controller.project_create);
router.get('/:id', project_controller.project_details);
router.put('/:id/update', project_controller.project_update);
router.delete('/:id/delete', project_controller.project_delete);

module.exports = router;
```


On passe maintenant sur le Controller donc voici son fichier de conf

```
GNU nano 5.4 project.controller.js *
const Project = require('../models/project.model');

exports.project_create = function (req, res) {
  let project = new Project({
    {
      name: req.body.name,
      description: req.body.description
    }
  });

  project.save(function (err) {
    if (err) {
      return next(err);
    }
    res.send('Project creer avec succes')
  })
};
```

Développement sur le frontend

On va créer un new composant sur le frontend

```
# ng g c list
ls
CREATE src/app/list/list.component.css (0 bytes)
CREATE src/app/list/list.component.html (19 bytes)
CREATE src/app/list/list.component.spec.ts (585 bytes)
CREATE src/app/list/list.component.ts (194 bytes)
UPDATE src/app/app.module.ts (1020 bytes)
```

Voici son fichier html

```
nosql_project-frontend-1 nosql_project-frontend
d7b2a5ac3683
4200:4200


Logs Inspect Terminal Files Stats
GNU nano 5.4 list.component.html


<h2>Liste des Projets</h2>
  <ul>
    <li *ngFor="let project of projects">
      <div class="project">
        <h3>{{ project.name }}</h3>
        <p>{{ project.description }}</p>
      </div>
      <button class="edit-button">Modifier</button>
      <button class="delete-button">Supprimer</button>
    </li>
  </ul>


```

Voici son fichier de conf

<



nosql_project-frontend-1

[nosql_project-frontend](#)

d7b2a5ac3683

4200:4200

Logs

Inspect

Terminal

Files

Stats

GNU nano 5.4

list.component.ts

```
import { Component, OnInit } from '@angular/core';
import { ProjectService } from '../project.service';

@Component({
  selector: 'app-list',
  templateUrl: './list.component.html',
  styleUrls: ['./list.component.css']
})
export class ListComponent implements OnInit {
  projects: any[] = [];

  constructor(private projectService: ProjectService) { }

  ngOnInit(): void {
    this.projectService.getProjects()
      .subscribe((res: any[]) => this.projects = res);
  }
}
```

Voici ce que le résultat donne

Liste des Projets

| |
|---|
| DAMN SALUT ça va <div>ModifierSupprimer</div> |
| Test faire 15 rapport <div>ModifierSupprimer</div> |
| Projet Samira Adc GAP <div>ModifierSupprimer</div> |
| salut dezedzedzed <div>ModifierSupprimer</div> |
| BIITCH DEJUZDJ <div>ModifierSupprimer</div> |

Voici les fonctions qui ont été rajouté pour les boutons modifier et supprimer

```
editProject(project: Project): void {
  project.editing = true;
}

confirmEdit(project: Project): void {
  this.projectService.updateProject(project._id, project)
    .subscribe((updatedProject: Project) => {
      const index = this.projects.findIndex(p => p._id === updatedProject._id);
      this.projects[index] = updatedProject;
      this.projects[index].editing = false;
    });
}

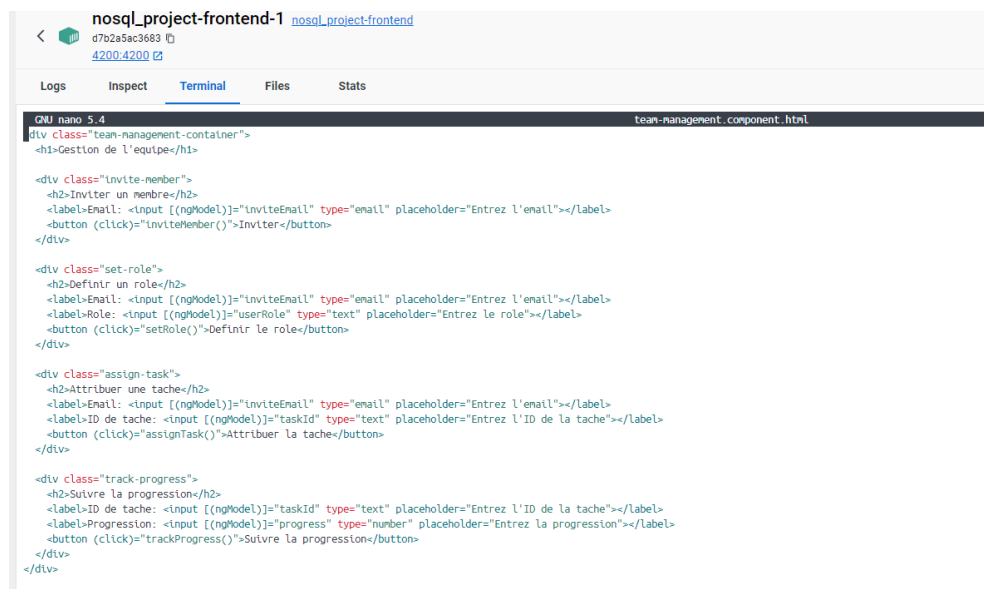
deleteProject(id: string): void {
  this.projectService.deleteProject(id)
    .subscribe(() => this.projects = this.projects.filter(project => project._id !== id));
}
```

d. Gestion des membres de l'équipe : inviter des membres, définir des rôles, attribuer des tâches et suivre la progression.

Développement sur le frontend

Dans l'Ideal voici une page qu'on alimentera plus tard ou pas forcément.

Voici le fichier html



The screenshot shows a web browser window with the address bar displaying 'nosql_project-frontend-1' and '4200.4200'. The browser's developer tools are open, showing the 'Terminal' tab. The terminal displays the output of a command, which is the HTML code for a team management component. The code is written in a dark theme and includes several sections for team management: 'Invite member', 'Set role', 'Assign task', and 'Track progress'. Each section contains HTML elements for labels, input fields, and buttons, along with AngularJS data-binding and event handling.

```
GNU nano 5.4 team-management.component.html
<div class="team-management-container">
  <h1>Gestion de l'équipe</h1>

  <div class="invite-member">
    <h2>Inviter un membre</h2>
    <label>Email: <input [(ngModel)]="inviteEmail" type="email" placeholder="Entrez l'email"></label>
    <button (click)="inviteMember()">Inviter</button>
  </div>

  <div class="set-role">
    <h2>Définir un rôle</h2>
    <label>Email: <input [(ngModel)]="inviteEmail" type="email" placeholder="Entrez l'email"></label>
    <label>Rôle: <input [(ngModel)]="userRole" type="text" placeholder="Entrez le rôle"></label>
    <button (click)="setRole()">Définir le rôle</button>
  </div>

  <div class="assign-task">
    <h2>Attribuer une tâche</h2>
    <label>Email: <input [(ngModel)]="inviteEmail" type="email" placeholder="Entrez l'email"></label>
    <label>ID de tâche: <input [(ngModel)]="taskId" type="text" placeholder="Entrez l'ID de la tâche"></label>
    <button (click)="assignTask()">Attribuer la tâche</button>
  </div>

  <div class="track-progress">
    <h2>Suivre la progression</h2>
    <label>ID de tâche: <input [(ngModel)]="taskId" type="text" placeholder="Entrez l'ID de la tâche"></label>
    <label>Progression: <input [(ngModel)]="progress" type="number" placeholder="Entrez la progression"></label>
    <button (click)="trackProgress()">Suivre la progression</button>
  </div>
</div>
```

Voici le fichier css

```
GNU nano 5.4 team-management.component.css
team-management-container {
display: flex;
flex-direction: column;
justify-content: space-around;
padding: 20px;
border: 1px solid #ccc;
border-radius: 5px;
margin: 10px;
max-width: 500px;
}

.team-management-container div {
margin-bottom: 20px;
}

.team-management-container h2 {
color: #333;
}

.team-management-container label {
display: block;
margin: 10px 0;
}

.team-management-container input {
width: 100%;
padding: 10px;
border-radius: 5px;
border: 1px solid #ccc;
}

.team-management-container button {
display: inline-block;
background-color: #4CAF50;
border: none;
color: white;
padding: 10px 20px;
text-align: center;
text-decoration: none;
display: inline-block;
font-size: 16px;
margin: 10px 2px;
}
```

Puis voici le fichier de conf

```
CNU nano 5.4 team-management.component.ts
/ team-management.component.ts

import { Component, OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-team-management',
  templateUrl: './team-management.component.html',
  styleUrls: ['./team-management.component.css']
})
export class TeamManagementComponent implements OnInit {
  inviteEmail: string = '';
  userRole: string = '';
  taskId: string = '';
  progress: number = 0;

  constructor(private http: HttpClient) { }

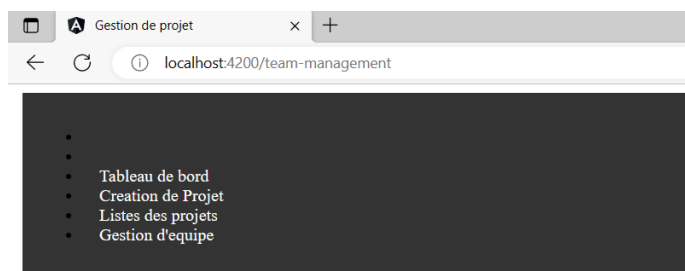
  ngOnInit(): void {
  }

  inviteMember(): void {
    this.http.post('http://localhost:3000/invite', { email: this.inviteEmail }).subscribe(
      response => console.log(response),
      error => console.error(error)
    );
  }

  setRole(): void {
    this.http.post('http://localhost:3000/setRole', { email: this.inviteEmail, role: this.userRole }).subscribe(
      response => console.log(response),
      error => console.error(error)
    );
  }

  assignTask(): void {
    this.http.post('http://localhost:3000/assignTask', { email: this.inviteEmail, taskId: this.taskId }).subscribe(
      response => console.log(response),
      error => console.error(error)
    );
  }
}
```

Ce qui donne



Gestion de l'equipe

Inviter un membre

Email:

Inviter

Definir un role

Email:

Role:

Definir

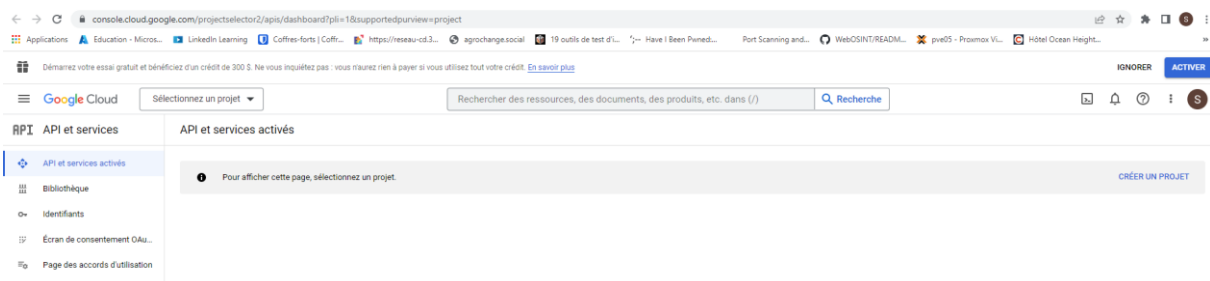
g. Synchronisation avec le calendrier : intégration avec Google Calendar API.

On génère notre nouveau composant

```
# ng generate component CalendarSync


CREATE src/app/calendar-sync/calendar-sync.component.css (0 bytes)
CREATE src/app/calendar-sync/calendar-sync.component.html (28 bytes)
CREATE src/app/calendar-sync/calendar-sync.component.spec.ts (642 bytes)
CREATE src/app/calendar-sync/calendar-sync.component.ts (229 bytes)
UPDATE src/app/app.module.ts (1544 bytes)
```

On se rend ensuite sur la console google cloud



On crée un projet

Nouveau projet



Il vous reste 11 projects dans votre quota. Demandez une augmentation ou supprimez des projets. [En savoir plus](#)

[MANAGE QUOTAS](#)

Nom du projet *

ID du projet : epsisalman. Vous ne pourrez pas le modifier par la suite. [MODIFIER](#)

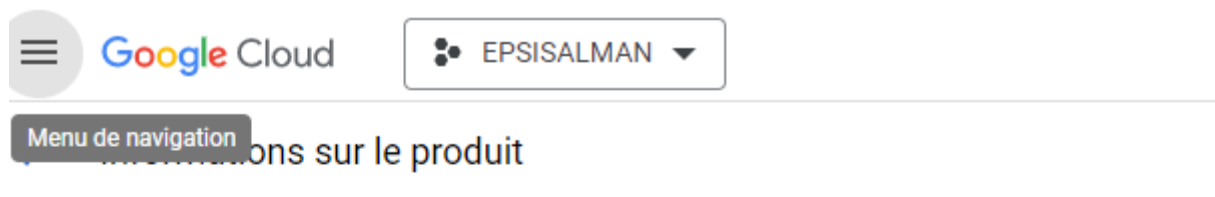
Zone *

[PARCOURIR](#)

Organisation ou dossier parent

[CRÉER](#) [ANNULER](#)

On active google calendar API



Google Calendar API

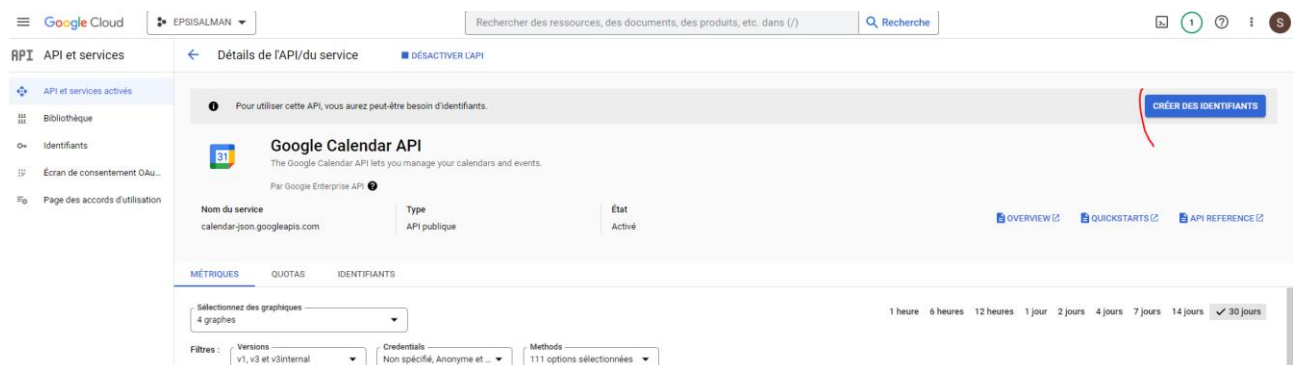
[Google Enterprise API](#)

Manage calendars and events in Google Calendar.

ACTIVER

ESSAYER CETTE API [↗](#)

On crée des identifiants











Codes secrets du client

Si vous modifiez les codes secrets d'un client, vous pouvez effectuer la rotation manuellement sans temps d'arrêt. [En savoir plus](#)



Le fait d'avoir plusieurs secrets augmente les risques de sécurité. Désactivez et supprimez l'ancien secret après avoir vérifié que votre application utilise bien le nouveau.

| | | |
|-----------------------|--|--|
| Code secret du client |  5YBjYth5rr |    |
| Date de création | 19 mai 2023 à 14:13:40 GMT+2 | |
| État | ✓ Activé | DÉSACTIVER |
| Code secret du client |  04SWsBc |    |
| Date de création | 19 mai 2023 à 14:14:13 GMT+2 | |
| État | ✓ Activé | DÉSACTIVER |

1. ADD SECRET

NON FONCTIONNELLE