-----\*\*\*\*\* PROJET REALISE AVEC VISUAL STUDIO CODE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*------------------

launchsetting.json --> http seulement

-Lancer **dotnetcore: dotnet watch run** --- watch en option

-creer .net visualstudio code :

en ligne de commande : **dotnet new 'nomprojet'**

package pour entity DbContext ----> voir DattingApp.api.csproj

<PackageReference Include="Microsoft.EntityFrameworkCore" Version="3.1.3"/>

<PackageReference Include="Microsoft.EntityFrameworkCore.Sqlite" Version="3.1.3"/>

---> créer

-les classes pour les tables

-le dataContext pour communiquer entre projet et base via entity

-option dans dataContext la **connectionString**

Dans **appsettings.json** "ConnectionStrings":{

"DefaultConnection":"Data Source=datingapp.db"

Dans Startup ---> injection public void **ConfigureServices**(IServiceCollection services)

{

services.AddDbContext<DataContext>(X=>X.UseSqlite(Configuration.GetConnectionString("DefaultConnection")));

services.AddControllers();

}

entity code first avec **migration**

I) installer Microsoft.EntityFrameworkCore.Design nuget---

en ligne de commande : **dotntet tool install --global** si non installé

si installé ---->

**dotnet ef -h** ---> pour trouver l'option à entrer pour creer la bdd depuis le code

puis dans le rep du projet :--

Création de la base :

-**dotnet ef migrations** add InitialCreate

**Dans une api les controleurs dérivent de la classe ControllerBase -🡪 pas de vue associée au projet**

**Dans une appli mvc les controleurs dérivent de la classe Controller 🡪 vues**

Installation de Angular client:

dans le repertoire du projet

**npm install -g @angular/cli**

dans le repertoire parent du projet

créer le projet angular :

**ng new DattingApp-SPA**

Outils à installer dans visual :

- Angular snippets --> ajoute des extraits de code

- Angular files ---> ajout de fonctionnalités éditeur

- Angular language ---->

* - Rename tag ---> permet de modifier les balises html plus facilement
* - Braket pair colorised2 ---> config des couleurs de visual
* - Debugger for chrome --
* - Material icon theme --> change les icones de fichier dans VSC
* - Pretier code formatter --> améliore l'apparence du code
* - TS lint ??
* - Angular 2 switched --> raccourci clavier pour bascule html css

**Exemple page simple développée sur visual studio code**

Affichage : Données de la table value (id,name) avec **DattingApp-SPA** Angular

Données : Table value créée dans l’api **DattingApp.api** utilisant Entity code first

Ajout d’une page

Angular-->nlle page : bn dr--> dans app--> generate componenent (nom minuscule)

* création d'un rep avec .css .html .ts

Pour fonctionnement en client

**Dans app.modules.ts**

* En tête 🡪 référencer le chemin

import { HttpClientModule } from '@angular/common/Http';

* Ajouter HttpClientModule

Imports [

HttpClientModule

]

Comme c’est un projet page unique lancement auto au démarrage, **dans value.component.ts**, ajouter une requête dans le constructeur :

1. Importer HttpClient et l’ ajouter comme instance du constructeur
2. import {HttpClient} from '@angular/common/http';

export class ValueComponent implements OnInit {

  values : any;

  constructor(private http:HttpClient) { }

  ngOnInit() {

    this.getValues();

  }

  getValues(){

    this.http.get('http://localhost:5000/api/values').subscribe(response=>{

      this.values = response;

    },error => {console.log(error);});

  }

}

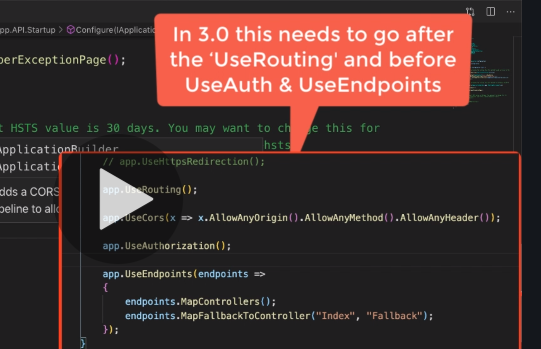
Il faut enfin ajouter un service pour pouvoir communiquer du port 5000 à 4200 malgré les règles de sécurité dans l’app serveur Api: **startup.cs**

Dans **ConfigureServices** :

  services.AddCors();//ajout service pour le passage d'un port à un autre

            // ce service est configuré dans void Configure

2)dans Configure :



1. Afficher dans valueComponent.html

<p \*ngFor="let value of values">

  {{value.id}},{{value.name}}

</p>

1. Mise en page:

**Install bootstrap font-awesome** en ligne de commande

Importer bootstrap dans le fichier CSS

@import '../node\_modules/bootstrap/dist/css/bootstrap.min';

@import '../node\_modules/font-awesome/css/font-awesome.min.css';

copie du projet Angular pour transmettre --> ne pas inclure node\_module ====> les fichiers sont rechargés avec le package.json

Token authentification :

Token Json 🡪 Standards RFC7519



Informations publiques

Algo codage

Type de codage

Zone cryptée

Cryptage en tête

Crypotage Public

Permet au programme de vérifier l’authentification d’un utilisateur sans interroger le serveur

Token :

En tête

Coprs

Signature : Infos de login à contrôler

* A la première connexion envoi mdp et nom au serveur
* Le serveur hache et contrôle la concordance entre mdp envoyé et contenu de la base
* Le serveur envoi un jeton au client
* Le client sauvegarde le jeton en local
* Ensuite les requêtes se font ensuite en envoyant le jeton

Autentification

Modules à ajouter pour l’autentification :

* Microsoft.IdentityModel.Tokens
* System.IdentityModel.Tokens.Jwt

Autentification par logiciel :

Module à ajouter :

* Microsoft.AspNetCore.Authentication.JwtBearer

Dans startup.cs

* Configurer le service
* services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme).AddJwtBearer(options=>{
* options.TokenValidationParameters = new TokenValidationParameters
* {
* ValidateIssuerSigningKey = true,
* IssuerSigningKey = new SymmetricSecurityKey(Encoding.ASCII.GetBytes(Configuration
* .GetSection("AppSettings:Token").Value)),
* ValidateIssuer = false,
* ValidateAudience = false
* };
* });
* Configurer
* app.UseRouting();
* app.UseAuthentication();
* app.UseAuthorization();