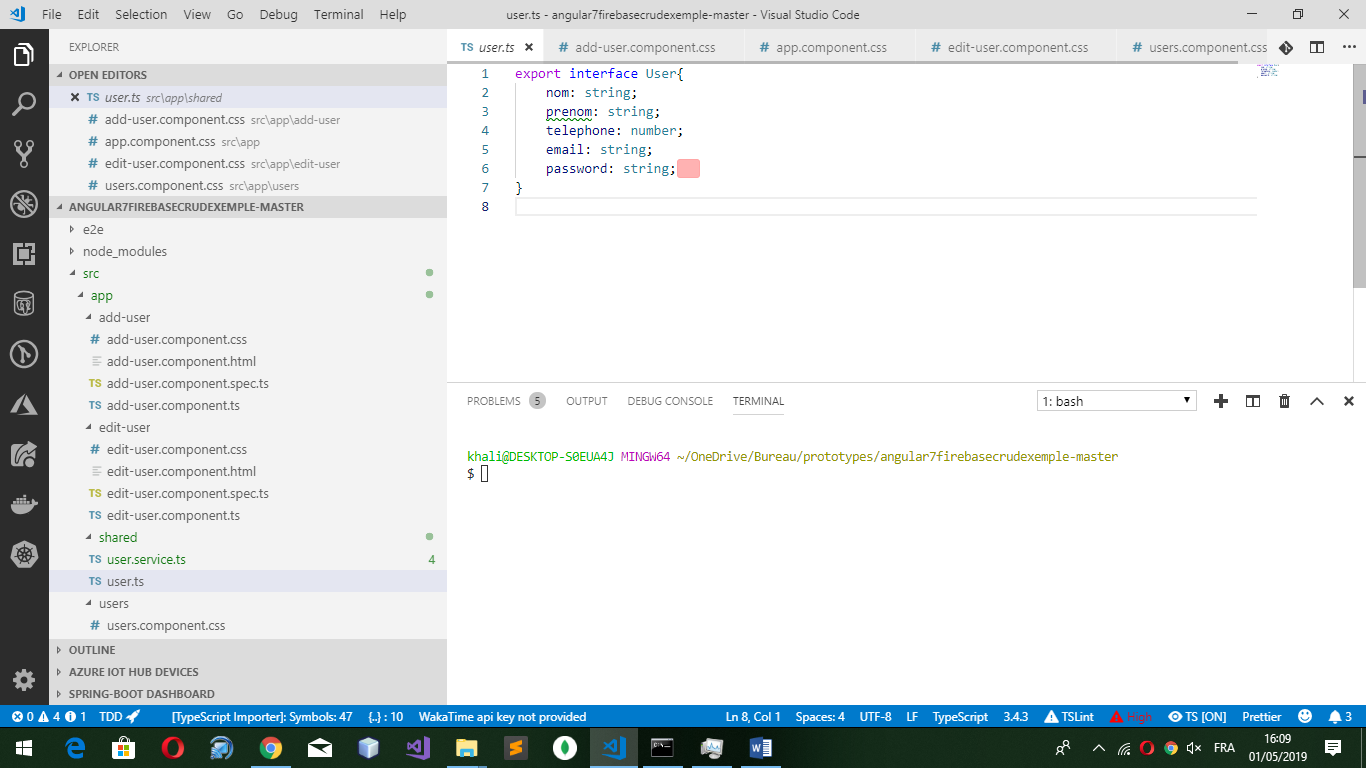
Les formulaire et real time databases avec angular 8 :

# L’instanciation (création de l’instance et l’importation) :

## La création de la couche modèle :

* Sous src/app de votre projet angular créez la classe user.ts via la commande suivante :

Ng g class **user**

* Dans la classe user ajouter le code suivant :
* Remarque : la classe typescript ne nécessite pas des méthodes get set ou des constructeurs

# La création du component add-user (couche vue) :

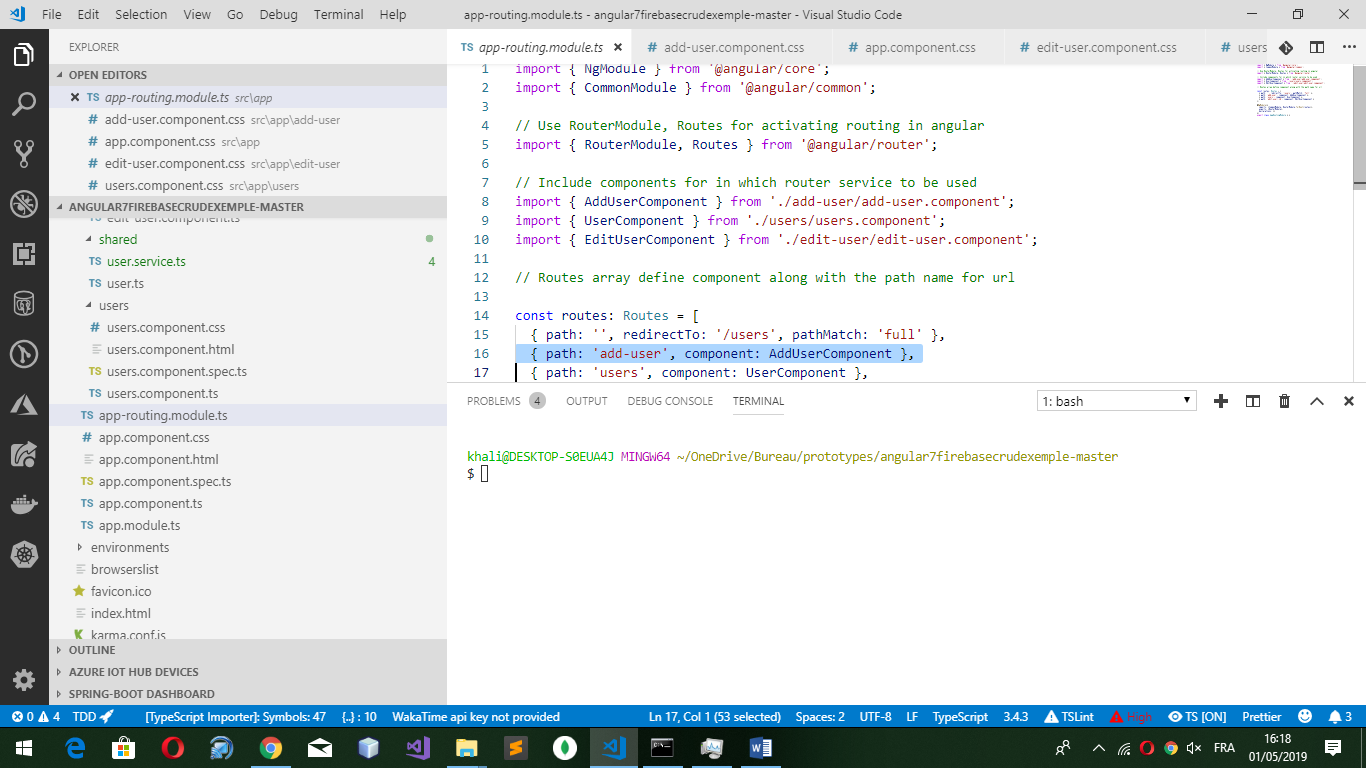
* Taper dans votre terminale la commande ng g c {{nom de component}} ; dans cet exemple : ng g c add-user
* Ajouter le routage de votre component comme suit:

Dans app-routing-module.ts importer le component en ajoutant le suivant :

import { AddUserComponent } from './add-user/add-user.component';

* Ajouter le chemin d’accès au component via la commande suivante :

{ path: 'add-user', component: AddUserComponent }



* Ajouter sous le dossier de votre component dans le fichier HTML (add-user.component.html) le code suivant :

<div class="pt-3 pb-2 mb-3 border-bottom">

<h1 class="h2">Ajouter User</h1></div><form >

<div class="row">

<div class="col-lg-5 col-md-12 col-sm-12">

<div class="row">

<div class="col-md-12 mb-3">

<label>nom</label>

<input type="text" class="form-control" required> </div>

<div class="col-md-12 mb-3">

<label>prenom</label>

<input type="text" class="form-control">

</div> </div>

<div class="row">

<div class="col-md-12 mb-3">

<label>email</label>

<input type="email" class="form-control" required> </div>

<div class="col-md-12 mb-3">

<label>Password</label>

<input type="password" class="form-control" required> </div>

<div class="col-md-12 mb-3">

<label>Telephone</label>

<input type="text" class="form-control" required>

</div>

</div>

<div class="form-group text-right">

<button type="button" class="btn btn-secondary gap-right">Annuler</button>

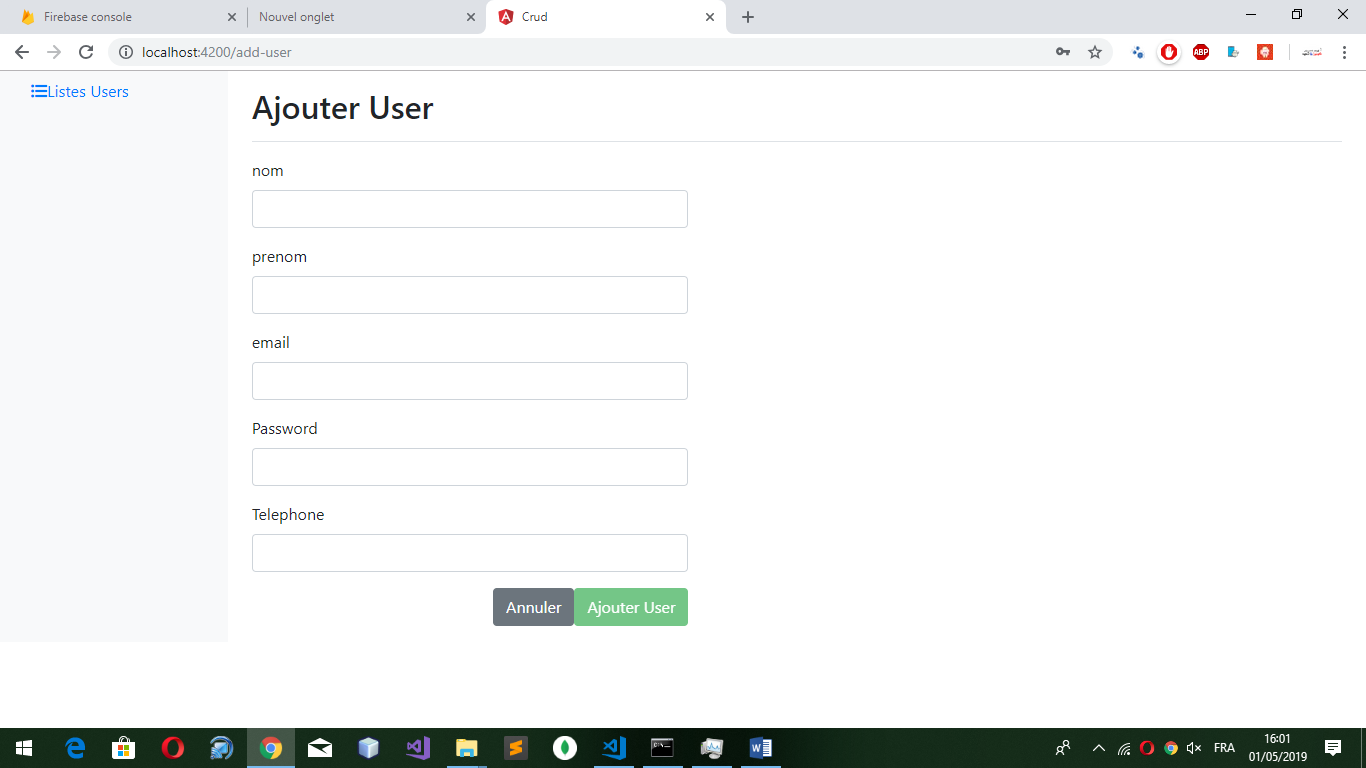
<button type="submit" class="btn btn-success" >Ajouter User</button>

</div>

</div>

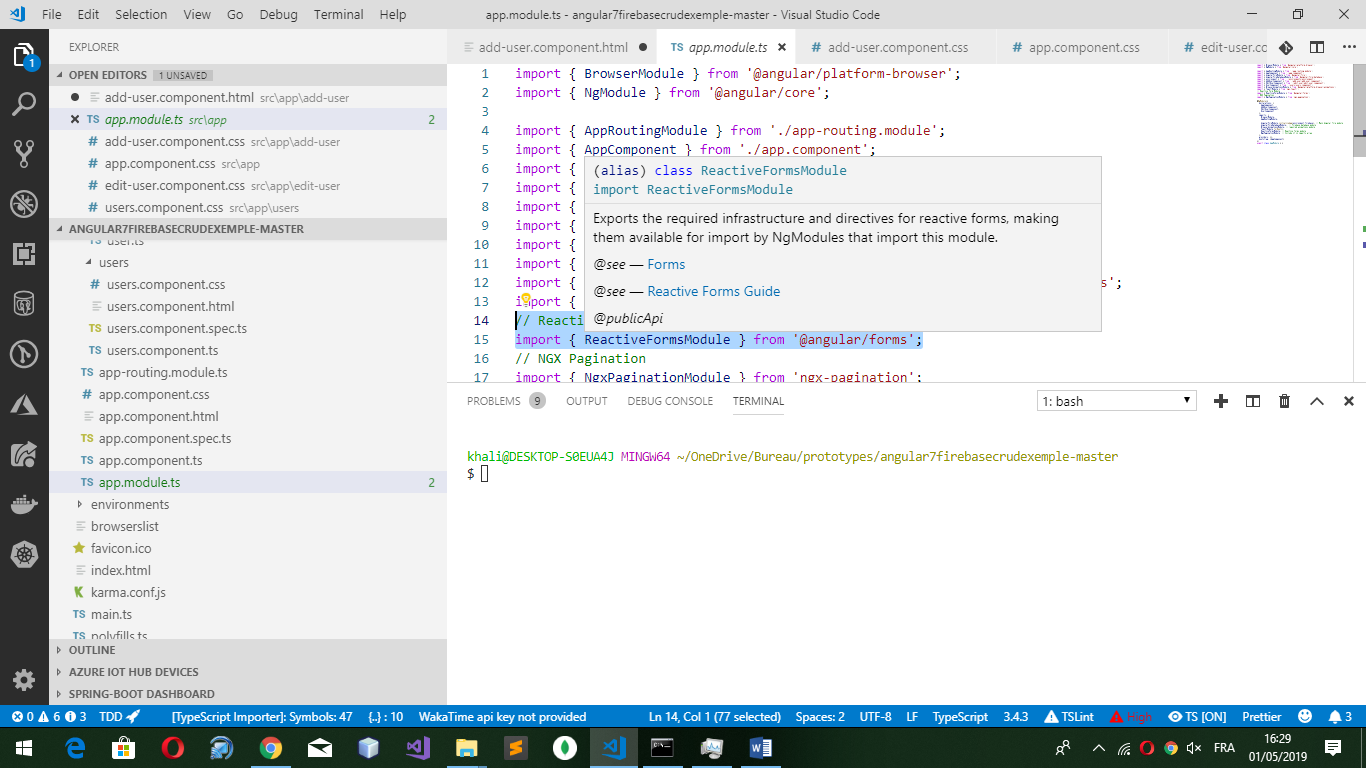
</div>

</form>



# Contrôle de saisit et liaison des couches (modèle et vue) :

* Installez reactive form et importez le dans app.module.ts :



En tappant :

// Reactive Form Module

import { ReactiveFormsModule } from '@angular/forms';

* Déclarez la classe user dans votre component :
* dans add-user.ts ajouter le suivant : import { User } from '../user;

intégrer les bibliothéques de controle de saisit et formulaire comme suit : import { FormBuilder, FormGroup, FormControl, Validators } from '@angular/forms';

* remplacez le constructeur vide de add-user.ts par le code suivant :

public userForm: FormGroup;

constructor(

public fb: FormBuilder, // Form Builder service for Reactive forms

) { }

* Ajouter des contrôles de saisit à votre code en tapant le suivant :

// Reactive user form

userForm() {

this.userForm = this.fb.group({

nom: ['', [Validators.required, Validators.minLength(6)]],

prenom: ['', [Validators.required, Validators.minLength(6)]],

email: ['', [Validators.required, Validators.pattern('^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+.[a-zA-Z0-9-.]+$')]],

password: ['', [Validators.required, Validators.minLength(6)]],

telephone: ['', [Validators.required, Validators.pattern('^[0-9]+$')]]

})

}

* Modifier votre formulaire comme suit :

<div class="pt-3 pb-2 mb-3 border-bottom">

<h1 class="h2">Ajouter User</h1>

</div>

<!-- Student form -->

<form [formGroup]="userForm" novalidate>

<div class="row">

<div class="col-lg-5 col-md-12 col-sm-12">

<div class="row">

<div class="col-md-12 mb-3">

<label>nom</label>

<input type="text" formControlName="nom" class="form-control" required>

</div>

<div class="col-md-12 mb-3">

<label>prenom</label>

<input type="text" formControlName="prenom" class="form-control">

</div>

</div>

<div class="row">

<div class="col-md-12 mb-3">

<label>email</label>

<input type="email" formControlName="email" class="form-control" required>

<!-- Showing errors using getter method -->

</div>

<div class="col-md-12 mb-3">

<label>Password</label>

<input type="password" formControlName="password" class="form-control" required>

<!-- Showing errors using getter method -->

</div>

<div class="col-md-12 mb-3">

<label>Telephone</label>

<input type="text" formControlName="telephone" class="form-control" required>

<!-- Showing errors using getter method -->

</div>

</div>

<div class="form-group text-right">

<button type="button" class="btn btn-secondary gap-right" (click)="ResetForm()">Annuler</button>

<button type="submit" class="btn btn-success" [disabled]="!userForm.valid">Ajouter User</button>

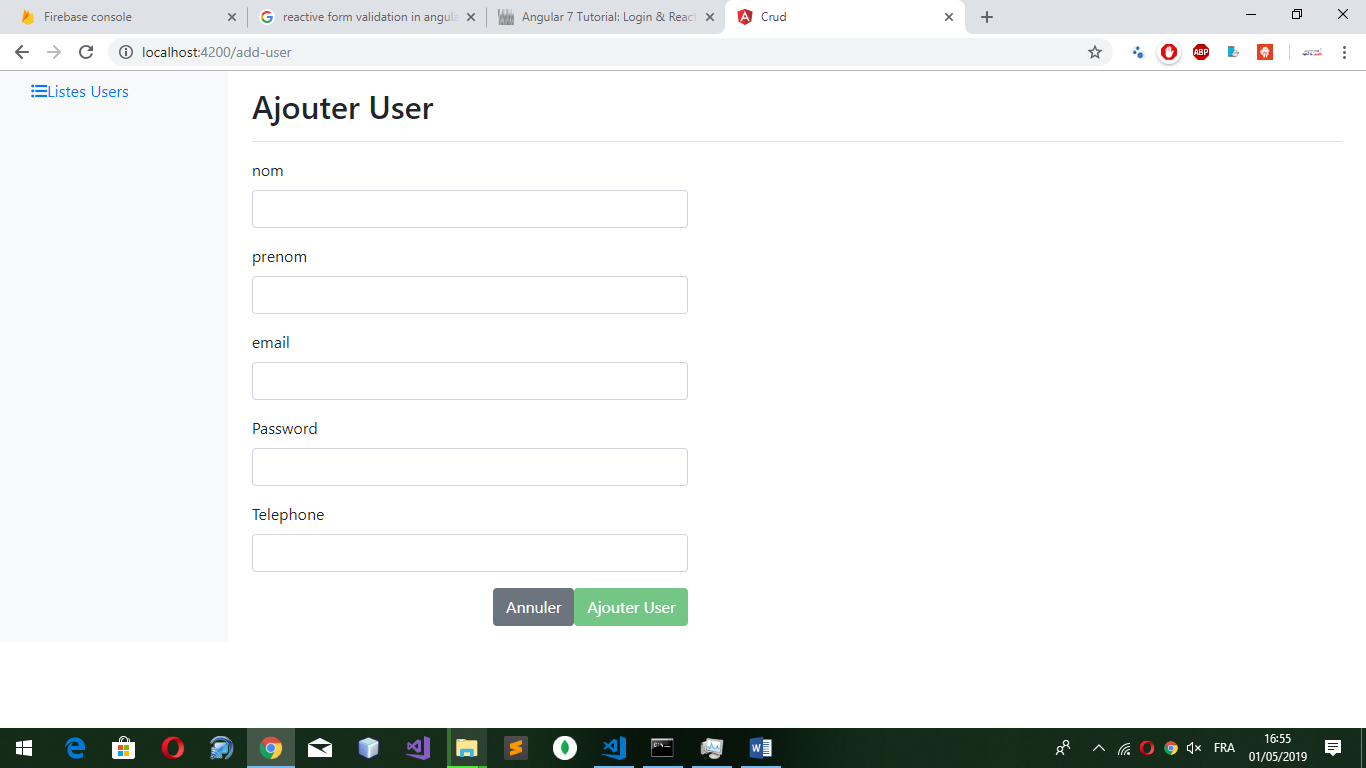
</div>

</div>

</div>

</form>

Le formulaire sera comme suit :



Récupération des données :

Ajouter le code suivant afin de lire les valeurs de vos champs :

get nom() {

return this.userForm.get('nom');

}

get prenom() {

return this.userForm.get('prenom');

}

get email() {

return this.userForm.get('email');

}

get password() {

return this.userForm.get('password');

}

get telephone() {

return this.userForm.get('telephone');

}

# Integration de firebase :

* Créer un compte gmail et accéder à votre firebase console en tappant dans la barre de navigation l’url suivant :
* <https://console.firebase.google.com/u/0/>

créer un projet nommé gestion\_users sous votre console :

* appuyez sur l’icone du web et copier le code dans var config
* installer angular firemodule : en tappant : npm i @angular/fire
* intégerez angularfire module dans votre application en ajoutant dans app.module.ts le suivant :

import { AngularFireModule } from '@angular/fire';

import { AngularFireDatabaseModule } from '@angular/fire/database';

import { environment } from '../environments/environment';

* modifier la section imports en ajoutant le suivant :

AngularFireModule.initializeApp(environment.firebase), // Main Angular fire module

AngularFireDatabaseModule, // Firebase database module

* dans le fichier environnement.ts modifier le code en ajoutant le token de votre console firebase comme suit :

export const environment = {

production: false,

firebase:{

apiKey: "AIzaSyDWA7cOyTMflmdA7zw08S\_DxZJ5zM-1jjw",

authDomain: "xxx-1987c.firebaseapp.com",

databaseURL: "https://xxx-1987c.firebaseio.com",

projectId: "xxx-1987c",

storageBucket: "xxx-1987c.appspot.com",

messagingSenderId: "1046644491353"

}

};

# integration de la couche service :

* tapez : ng g service user pour générer le fichier de service pour user
* créez les méthodes suivante pour générer le crud service de la classe user (précédente)

import { Injectable } from '@angular/core';

import { User } from '.. /user'; // User data telephone interface class

import { AngularFireDatabase, AngularFireList, AngularFireObject } from '@angular/fire/database';

// Firebase modules for Database, Data list and Single object

@Injectable({

providedIn: 'root'

})

export class UserService {

UsersRef: AngularFireList<any>; // Reference de user dans firebase

UserRef: AngularFireObject<any>; // Reference

// Inject AngularFireDatabase Dependency in Constructor

constructor(private db: AngularFireDatabase) { }

// Create User

AjouterUser(User: User) {

this.UsersRef.push({

nom: User.nom,

prenom: User.prenom,

password: User.password,

email: User.email,

telephone: User.telephone,

})

}

// read Single User Object

GetUser(id: string) {

this.UserRef = this.db.object('users/' + id);

return this.UserRef;

}

// Fetch Listes Users

GetUsersList() {

this.UsersRef = this.db.list('users');

return this.UsersRef;

}

// Update User Object

ModifierUser(User: User) {

this.UserRef.update({

nom: User.nom,

prenom: User.prenom,

password: User.password,

email: User.email,

telephone: User.telephone,

})

}

// Delete User Object

SupprimerUser(id: string) {

this.UserRef = this.db.object('users/' + id);

this.UserRef.remove();

}

}

Dans add-user.component.ts de votre component ajouter le suivant :

submitUserData() {

this.crudApi.AjouterUser(this.userForm.value); // Submit user data using CRUD API

this.toastr.success(this.userForm.controls['prenom'].value + ' successfully added!'); // Show success message when data is successfully submited

this.ResetForm(); // Reset form when clicked on reset button

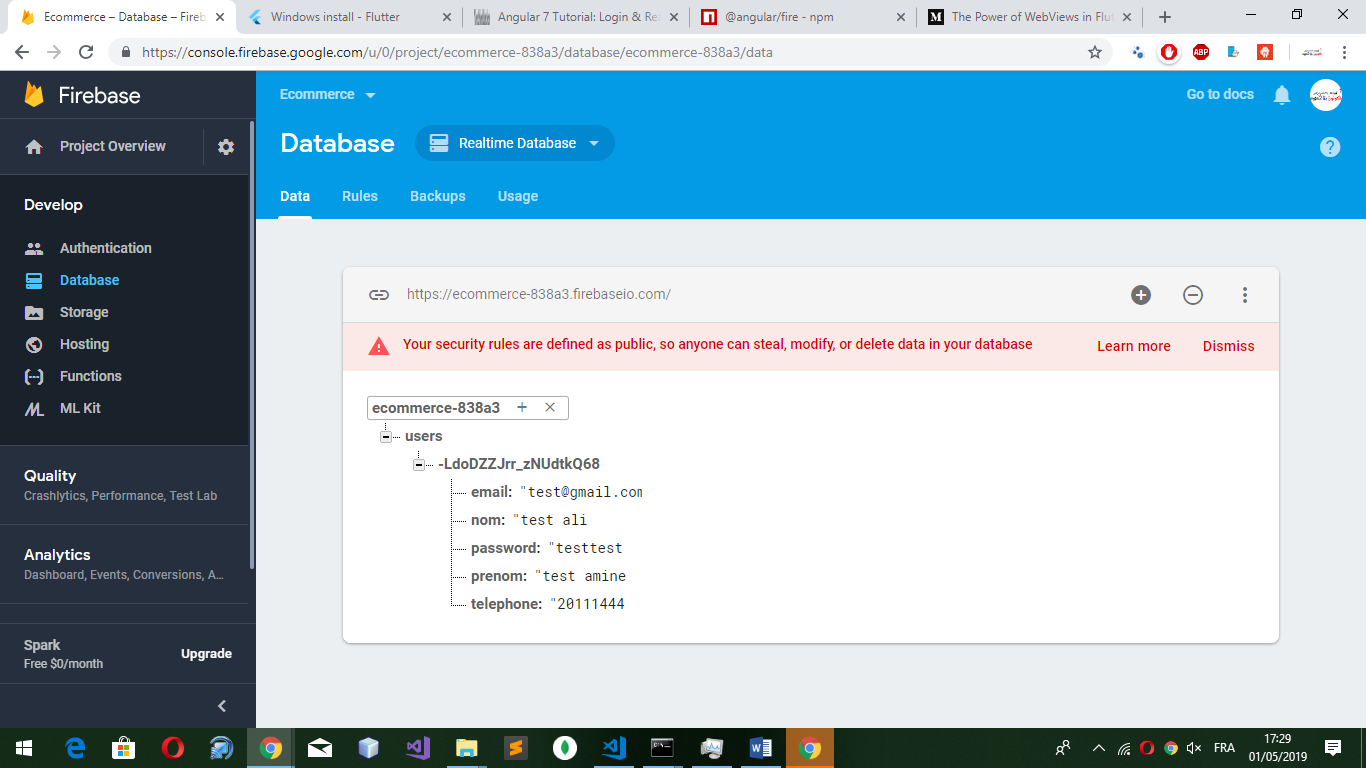
this.router.navigate(['users']);

};

Modifiez la balise form comme suit : <form [formGroup]="userForm" (ngSubmit)="submitUserData()" novalidate>

Et tester le code en vérifiant l’ajout dans votre firebase project database console ;

Vous aurez le suivant :



Affichage des données depuis la BD :

* taper ng g c List-users pour créer le component d’affichage de liste des utilisateurs

dans user.component.ts ajouter :

<div class="d-flex justify-content-between flex-wrap flex-md-nowrap align-items-center pt-3 pb-2 mb-3 border-bottom">

<h1 class="h2">Listes users</h1>

<!-- It won't show if there is no user data -->

<a routerLink="/add-user" class="btn btn-success" \*ngIf="hideWhenNoStudent">

<i class="fas fa-plus custom-fa-plus"></i>

Ajouter user

</a>

</div>

<div class="pricing-header mx-auto">

<!-- Preloader shows before the data loads-->

<div class="no-data text-center" \*ngIf="preLoader">

<img src="assets/preloader.gif" class="preloader-icon" alt="No user">

</div>

<!-- No data shows when their is no user data available -->

<div class="no-data text-center" \*ngIf="noData">

<img src="assets/no-user.svg" class="nodata-msg" alt="No user">

<p class="nodata-msg">No user added yet!</p>

<a routerLink="/add-user" class="btn btn-success">

<i class="fas fa-plus custom-fa-plus"></i>

Ajouter user

</a>

</div>

<!-- Showing users data -->

<div class="table-responsive" \*ngIf="hideWhenNoStudent">

<table class="table table-bordered table-responsive-sm table-responsive-md table-responsive-lg">

<thead>

<tr>

<th scope="col">nom</th>

<th scope="col">prenom</th>

<th scope="col">email</th>

<th scope="col">password</th>

<th scope="col">Telephone</th>

</tr>

</thead>

<tbody>

<!-- \*ngFor loop iterates over Student array and fetch the user's data -->

<!-- paginate pipe will add pagination in user's list, it won't show if items are less then 7 -->

<tr \*ngFor="let user of User | paginate: { itemsPerPage: 7, currentPage: p }; let i = index;">

<td>{{user.nom}} </td>

<td>{{user.prenom}} </td>

<td>{{user.email}} </td>

<td>{{user.password}}</td>

<td>{{user.telephone}} </td>

</tr>

</tbody>

</table>

</div>

<!-- Pagination -->

<pagination-controls (pageChange)="p = $event" autoHide="true" responsive="true"></pagination-controls>

</div>

* ajouter au code de List-user.component.ts le suivant :

//importation des services et modèle de base de données

import { User } from '.. /user'; // user interface class for Data types.

import { UserService } from '.. /user.service'; // CRUD API service class

dans la classe ajouter le suivant pour la pagination et l’initialisation des variables :

p: number = 1; // Settup up pagination variable

User: User[]; // Save users data in user's array.

hideWhenNoUsers: boolean = false; // Hide users data table when no user.

noData: boolean = false; // Showing No user Message, when no user in database.

preLoader: boolean = true; // Showing Preloader to show user data is coming for you from thre server(A tiny UX Shit)

constructor(

public crudApi: UserService, // Inject user CRUD services in constructor.

) { }

***Pour synchroniser la lecture des données avec firebase  ajouter le suivant :***

dataState() {

this.crudApi.GetUsersList().valueChanges().subscribe(data => {

this.preLoader = false;

if (data.length <= 0) {

this.hideWhenNoStudent = false;

this.noData = true;

} else {

this.hideWhenNoStudent = true;

this.noData = false;

}

})

}

***Pour initialiser la page mettez le code suivant dans votre méthode OnInit()***

ngOnInit() {

this.dataState(); // Initialize user's list, when component is ready

let s = this.crudApi.GetUsersList();

console.log(s);

s.snapshotChanges().subscribe(data => { // Using snapshotChanges() method to retrieve list of data along with metadata($key)

this.User = [];

data.forEach(item => {

let a = item.payload.toJSON();

//console.log(a);

a['$key'] = item.key;

this.User.push(a as User);

})

})

}

Le code doit être comme suit :

//List-user.component.ts

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

import { User } from '../ user'; // user interface class for Data types.

import { UserService } from '.. /user.service'; // CRUD API service class

@Component({

selector: 'app-users',

templateUrl: './users.component.html',

styleUrls: ['./users.component.css']

})

export class UserComponent implements OnInit {

p: number = 1; // Settup up pagination variable

User: User[]; // Save users data in user's array.

hideWhenNoStudent: boolean = false; // Hide users data table when no user.

noData: boolean = false; // Showing No user Message, when no user in database.

preLoader: boolean = true; // Showing Preloader to show user data is coming for you from thre server(A tiny UX Shit)

constructor(

public crudApi: UserService, // Inject user CRUD services in constructor.

public toastr: ToastrService // Toastr service for alert message

) { }

ngOnInit() {

this.dataState(); // Initialize user's list, when component is ready

let s = this.crudApi.GetUsersList();

console.log(s);

s.snapshotChanges().subscribe(data => { // Using snapshotChanges() method to retrieve list of data along with metadata($key)

this.User = [];

data.forEach(item => {

let a = item.payload.toJSON();

//console.log(a);

a['$key'] = item.key;

this.User.push(a as User);

})

})

}

// Using valueChanges() method to fetch simple list of users data. It updates the state of hideWhenNoStudent, noData & preLoader variables when any changes occurs in user data list in real-time.

dataState() {

this.crudApi.GetUsersList().valueChanges().subscribe(data => {

this.preLoader = false;

if (data.length <= 0) {

this.hideWhenNoStudent = false;

this.noData = true;

} else {

this.hideWhenNoStudent = true;

this.noData = false;

}

})

}

// Method to delete user object

}

# Routing et mise en place :

import { UserComponent } from './users/users.component';

Dans routes de votre projet :

{ path: 'users', component: UserComponent }

