1. Overview General

Într-o aplicație web, procesul de **registrare** permite utilizatorilor să creeze un cont nou. Acest proces implică colectarea informațiilor de la utilizator, cum ar fi un nume de utilizator și o parolă, validarea acestor informații, și apoi stocarea lor într-o bază de date. În proiectul tău, acest flux implică atât partea de frontend (ce vede utilizatorul) cât și backend (logica din spatele aplicației).

2. Flow-ul Codului pentru Registrare

Să vedem cum funcționează fiecare parte din cod:

Frontend (Angular)

Frontend-ul este responsabil de colectarea datelor de la utilizator și de trimiterea lor la server.

- **register.component.html**: Acesta este fișierul care definește cum arată formularul de înregistrare. Conține un formular cu câmpuri pentru nume de utilizator și parolă.

html

Copy code

```
<form [formGroup]="registerInputForm">
 <div class="register-form-flex">
  <mat-card>
   <mat-card-title class="title mb-3">Register</mat-card-title>
   <mat-card-content>
    <mat-form-field>
     <input type="text" matInput placeholder="User Name" formControlName="userName">
    </mat-form-field>
    <br/>
    <mat-form-field>
     <input type="password" matInput placeholder="Password" formControlName="passWord"/>
    </mat-form-field>
   </mat-card-content>
   <mat-card-actions>
    <div class="button-flex-container">
     <button mat-raised-button color="primary" (click)="registerInput()">Register</button>
    </div>
```

```
    </mat-card-actions>
   </mat-card>
  </div>
</form>
```

- **register.component.ts**: Acesta este fișierul TypeScript asociat cu componenta de înregistrare. Conține logica necesară pentru a colecta datele din formular și a le trimite la server.
    - registerInputForm: Este un FormGroup, un model reactiv care leagă formularul de HTML și verifică validitatea acestuia.
    - registerInput(): Aceasta este metoda care se apelează atunci când utilizatorul apasă pe butonul "Register". Colectează datele din formular și le trimite la server folosind authenticationService.register().

typescript

Copy code

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { AuthenticationService } from '../../services/authentication.service';
import { MatSnackBar } from '@angular/material/snack-bar';
import { Router } from '@angular/router';

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

  protected registerInputForm!: FormGroup;

  constructor(private authenticationService: AuthenticationService, private snackBar: MatSnackBar,
private router: Router) {}
```

```
ngOnInit(): void {
  this.initForm();
}

private initForm() {
  this.registerInputForm = new FormGroup({
    userName: new FormControl('', Validators.required),
    passWord: new FormControl('', Validators.required),
  });
}

public registerInput() {
  const userName = this.registerInputForm.value.userName;
  const passWord = this.registerInputForm.value.passWord;

  const credentialsModel = { username: userName, password: passWord };

  this.authenticationService.register(credentialsModel).subscribe({
    next: (res: any) => {
      this.router.navigate(['/login']);
      this.openSuccessSnackBar(res.message);
    },
    error: (err: any) => {
      console.log(err);
      this.openFailureSnackBar(err.error);
    }
  });
}
```

```
  public openSuccessSnackBar(message: string) {
    this.snackBar.open(message, "OK", {
      duration: 3000,
      panelClass: ['green-snackbar', 'register-snackbar'],
    });
  }

  public openFailureSnackBar(message: string) {
    this.snackBar.open(message, "Try again!", {
      duration: 3000,
      panelClass: ['red-snackbar', 'register-snackbar'],
    });
  }
}
```

Backend (ASP.NET Core)

Backend-ul gestionează logica de afaceri și interacțiunea cu baza de date.

- **AuthenticationController.cs**: Acesta este controller-ul care primește cererea de înregistrare din frontend. Metoda Register() este responsabilă de crearea unui nou utilizator.
    - Începe prin a verifica dacă utilizatorul există deja.
    - Dacă nu există, creează un nou User și îl adaugă în baza de date.
    - Dacă totul este în regulă, returnează un mesaj de succes către frontend.

```
csharp
Copy code
[Route("api/[controller]/[action]")]
[ApiController]
public class AuthenticationController : ControllerBase
{
    private AppContext _appContext;
```

```
public AuthenticationController(AppContext appContext)
{
   _appContext = appContext;
}

[HttpPost(Name = "Register")]
public ActionResult Register(CredentialsModel credentials)
{
   try
   {
      User user = new User();
      user.Username = credentials.Username;
      user.Password = credentials.Password;

      User existingUser = _appContext.Users.FirstOrDefault(x => x.Username == credentials.Username);

      if (existingUser != null)
      {
         throw new ArgumentException("Username already exists!");
      }

      _appContext.Users.Add(user);
      _appContext.SaveChanges();
   }
   catch (Exception e)
   {
      return StatusCode(StatusCodes.Status500InternalServerError, e.Message);
   }
```

```
        return Ok(new
        {
            Message = "Register Successful. You can login using the credentials."
        });
    }
}
```

- **AppContext.cs**: Este contextul bazei de date, unde sunt definite tabelele. DbSet<User> definește tabela Users în baza de date, în care sunt stocate informațiile despre utilizatori.

csharp

Copy code

```
public class AppContext : DbContext
{
    public AppContext(DbContextOptions<AppContext> options) : base(options) { }

    public DbSet<User> Users { get; set; }
}
```

- **User.cs**: Este modelul de date care reprezintă un utilizator în baza de date. Are câmpuri precum Id, Username, și Password.

csharp

Copy code

```
public class User
{
    public int Id { get; set; }
    public string Username { get; set; }
    public string Password { get; set; }
}
```

Rezumat

1. **Frontend**:
    - Utilizatorul completează un formular cu numele de utilizator și parola.
    - Când formularul este trimis, datele sunt colectate și trimise către server printr-o cerere HTTP POST.
2. **Backend**:
    - Serverul primește cererea de înregistrare.
    - Verifică dacă utilizatorul există deja.
    - Dacă utilizatorul nu există, adaugă utilizatorul în baza de date.
    - Returnează un răspuns către frontend pentru a informa utilizatorul dacă înregistrarea a avut succes sau nu.
3. **Feedback pentru Utilizator**:
    - În funcție de răspunsul serverului, utilizatorul primește un mesaj de succes sau de eroare printr-o snackbar în frontend.