1. Overview General

Funcționalitatea de **login** permite utilizatorilor să se autentifice în aplicație folosind numele de utilizator și parola. Acest proces implică verificarea credențialelor introduse de utilizator și, dacă acestea sunt corecte, redirecționarea utilizatorului către pagina principală a aplicației.

2. Flow-ul Codului pentru Login

Vom trece prin fiecare parte a codului care este responsabilă pentru funcționalitatea de login.

Frontend (Angular)

Frontend-ul colectează informațiile de la utilizator (nume de utilizator și parolă) și le trimite la server pentru a fi verificate.

- **login.component.html**: Acesta este fișierul HTML care definește structura vizuală a formularului de login.

html

Copy code

```
<form [formGroup]="loginInputForm">
 <div class="login-form-flex">
  <mat-card>
   <mat-card-title class="title mb-3">Login</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-button color="accent" (click)="forgotPassword()">Forgot Password?</button>
     <button mat-raised-button color="primary" (click)="loginInput()">Login</button>
```

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

- **loginInputForm**: Este un FormGroup care conține câmpurile userName și passWord. Această structură este utilizată pentru a lega câmpurile de intrare din HTML la logica din TypeScript.
- **Butonul de login**: Când utilizatorul apasă pe butonul "Login", metoda loginInput() este apelată pentru a procesa autentificarea.

- **login.component.ts**: Acesta este fișierul TypeScript care gestionează logica pentru componenta de login.

typescript

Copy code

```
import { Component, OnInit } from '@angular/core';
import { MatSnackBar } from '@angular/material/snack-bar';
import { Router } from '@angular/router';
import { AuthenticationService } from '../../services/authentication.service';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { Credentials } from '../../models/credentials.model';
import { CookieService } from 'ngx-cookie-service';
import { AccountService } from '../../services/account.service';


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

```
constructor(
  private authenticationService: AuthenticationService,
  private snackBar: MatSnackBar,
  private router: Router,
  private cookieService: CookieService,
  private accService: AccountService
) {}

ngOnInit(): void {
  this.checkLoggedUser(); // Verifică dacă utilizatorul este deja logat
}

protected loginInputForm: FormGroup = new FormGroup({
  userName: new FormControl('', Validators.required), // Validează câmpul userName
  passWord: new FormControl('', Validators.required) // Validează câmpul passWord
});

public loginInput() {
  let userName = this.loginInputForm.value.userName;
  let passWord = this.loginInputForm.value.passWord;

  let credentialsModel: Credentials = {
    username: userName,
    password: passWord
  };

  this.authenticationService.login(credentialsModel).subscribe({
    next: (res: any) => {
      this.router.navigate(['/products-list']);
```

```
      this.cookieService.set('cookieName', 'cookieValue');
      this.accService.setLoggedInState();
    },
    error: (err: any) => {
      console.log(err);
      this.openFailureSnackBar(err.error);
    }
  });
 }

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

 private checkLoggedUser() {
  if (this.cookieService.check('cookieName'))
    this.router.navigate(['/products-list']);
 }

 public forgotPassword() {
  this.router.navigate(['/forgot-password']);
 }
}
```

- **loginInputForm**: Este formularul de login care colectează numele de utilizator și parola.

- **loginInput()**: Metoda care este apelată atunci când utilizatorul încearcă să se logheze. Aceasta trimite datele de login la server prin serviciul AuthenticationService.
- **checkLoggedUser()**: Verifică dacă utilizatorul este deja autentificat (în baza unui cookie) și, dacă da, îl redirecționează automat la lista de produse.
- **forgotPassword()**: Redirecționează utilizatorul către pagina de resetare a parolei.

Backend (ASP.NET Core)

Partea de backend primește cererea de login și verifică dacă credențialele introduse de utilizator sunt corecte.

- **AuthenticationController.cs**: Acesta este controller-ul responsabil pentru autentificare.

csharp

Copy code

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

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

  [HttpPost(Name = "Login")]
  public ActionResult Login(UpdateCredentialsModel credentialsModel)
  {
    try
    {
      User user = _appContext.Users.FirstOrDefault(x => x.Username ==
credentialsModel.Username);
```

```
        if (user == null || user.Password != credentialsModel.Password)
        {
            throw new ArgumentException("You have entered an invalid username or password.");
        }
    }
    catch (Exception e)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, e.Message);
    }

    return Ok();
  }
}
```

- **Login()**: Metoda care primește cererea de login. Aceasta caută utilizatorul în baza de date și verifică dacă parola introdusă corespunde cu cea stocată. Dacă datele sunt corecte, returnează un răspuns de succes.

- **User.cs**: Modelul de date care reprezintă un utilizator în baza de date.

csharp

Copy code

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

- **UpdateCredentialsModel.cs**: Acesta este modelul utilizat pentru a trimite credențialele de autentificare de la frontend la backend.

csharp

Copy code

```
public class UpdateCredentialsModel : CredentialsModel
{
    public string? NewPassword { get; set; }
}
```

Rezumat

1. **Frontend**:
    - Utilizatorul introduce numele de utilizator și parola.
    - Apasă pe butonul "Login", iar aplicația trimite datele la server printr-o cerere HTTP POST.
    - Dacă autentificarea reușește, utilizatorul este redirecționat către lista de produse.
2. **Backend**:
    - Serverul primește cererea de autentificare și verifică dacă utilizatorul există și dacă parola este corectă.
    - Dacă totul este în regulă, serverul returnează un răspuns de succes, iar utilizatorul este considerat autentificat.