Sprint 5 Register and Authentication in Angular

Регистрация пользователя

- в проекте Angular в папке `components` создайте компоненты `header` и `users`. Для этого перейдите в `components` и выполните команды:

```
ng g c header --skip-tests
```

- выполните также для `users`

Провайдер роутера

- проверьте конфигурацию для роутера

app.config.ts

```
import { provideRouter } from '@angular/router';
import { routes } from './app.routes';
```

- в коллекцию `providers` внесите новый элемент `provideRouter(routes)`

Роутинг

app.routes.ts

```
import { Routes } from '@angular/router';
import { HeaderComponent } from '../components/header/header.component';
import { HomeComponent } from '../components/home/home.component';
import { UsersComponent } from '../components/users/users.component';

export const routes: Routes = [
    { path: 'header', component: HeaderComponent },
    { path: 'users', component: UsersComponent },
    { path: 'home', component: HomeComponent },
    { path: '', component: HomeComponent },
];
```

Компонент router-outlet

- в шаблоне `app` внесите изменения. Также импортируйте компонент `header` в `app` компонент.

```
<app-header/>
<router-outlet/>
```

Angular Material

- установите пакет

```
ng add @angular/material
```

в `angular.json` в разделе `styles` подключите предустановленную тему:

```
"styles": [
        "@angular/material/prebuilt-themes/indigo-pink.css",
        "src/styles.scss"
    ],
```

Material Icons

- установка пакета для локальной разработки
- `npm install material-design-icons --save`
- `npm install material-design-icons-iconfont --save`
- подключение пакета в `styles.scss`.

```
@import '../node_modules/material-design-icons-iconfont/dist/material-design-icons.css'
```

Создание компонента header

- подключите в компоненте `header` модули:

```
imports: [CommonModule, MatIconModule, MatToolbarModule, MatButtonModule],
```

- в шаблоне компонента

```
<mat-toolbar color="primary">

  <button mat-button >
```

```
    <mat-icon>
      home
    </mat-icon>
    Home
  </button>

</mat-toolbar>
```

Список всех пользователей

- добавьте новую кнопку в `mat-toolbar` в `header` для отображения пользователей, а также примените маршрутизацию к этой кнопке к компоненту `users` по нажатию на кнопку. Для этого импортируйте модуль `RouterLink` в компонент `header`.

```
  <button mat-button [routerLink] = "['/users']">
      <mat-icon>
          people
      </mat-icon>
      Пользователи
  </button>
```

- сделайте ссылку на корневую страницу для кнопки с домом.

Задание: перенесите функционал вывода списка пользователей из компонента `home` в компонент `users`. В компоненете `home` в шаблоне оставьте только приветствие пользователя.

Создание компонента auth

- в шаблоне компонента `auth` создайте форму:

```
<div class="auth">
    <h1> Авторизация </h1>
    <form (ngSubmit)= "login()">
        <p>
            <input placeholder="Login" name="login" [(ngModel)]="model.login"/>
        </p>
        <p>
            <input placeholder="Password" name="password"
[(ngModel)]="model.password"/>
        </p>
       <button type="submit">Отправить</button>
    </form>
</div>
```

в компоненте `auth` создайте свойство `model:any = {}` и метод:

```
  login(){
    console.log(this.model)
  }
```

Также надо импортировать модуль `FormModule` для работы директивы `[(ngModel)]`.

Проверьте работу формы в консоли браузера

Подключение компонентов Angular Material

- замените тег `input` для логина и пароля:

```
  <mat-form-field>
      <mat-label>Login</mat-label>
      <input matInput name="login" type="text" [(ngModel)] = "model.login" />
  </mat-form-field>
```

```
    <mat-form-field>
        <mat-label>Password</mat-label>
        <input matInput name="password" type="password" [(ngModel)]
="model.password" />
    </mat-form-field>
```

- замените отображение кнопки

```
    <button mat-button color="primary" type="submit">
        <mat-icon>
          login
        </mat-icon>
        Войти
    </button>
```

- проверьте импорт в компоненте `auth`: должны быть `FormsModule, MatInputModule, MatFormField, MatLabel, MatIcon`

Замечание: в консоли браузера может быть предупреждение про autocompele для полей ввода. Поставьте атрибут `autocomplete="off"` для полей ввода.

Создание метода авторизации в API

- создайте метод в `UsersController`. Чтобы его создать надо добавить сингнатуру в интерфейс и метод в `UserRepository`.

```
    [HttpPost("Login")]
    public ActionResult Login(UserDto userDto){
        var user = _repo.FindUser(userDto.Login);
        return CheckPasswordHash(userDto, user);
    }
```

- проверку логики пароля вынесите в отдельный приватный метод

```
private ActionResult CheckPasswordHash(UserDto userDto, User user)
    {

        using var hmac = new HMACSHA256(user.PasswordSalt);
        var computedHash =
hmac.ComputeHash(Encoding.UTF8.GetBytes(userDto.Password));

        for (int i = 0; i < computedHash.Length; i++)
        {
            if (computedHash[i] != user.PasswordHash[i])
            {
                return Unauthorized($"Неправильный пароль");
            }
        }

        return Ok(user);
    }
```

- проверьте работу метода в API

AuthService

- создайте сервис `auth.service.ts`:

```
ng g s auth --skip-tests
```

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { map } from 'rxjs/operators';
import User from '../models/user';
import { ReplaySubject } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuthService {
```

```
  baseUrl:String = "http://localhost:[port]/api/";

  private currentUserSource = new ReplaySubject<User>(1);
  currentUser$ = this.currentUserSource.asObservable();

  constructor(private http:HttpClient) { }

  login(model:any){

    return this.http.post<User>(this.baseUrl + "Users/Login", model).pipe(
      map((response: User) => {
        const user = response;
        if(user){
          localStorage.setItem("user",JSON.stringify(user))
          this.currentUserSource.next(user);
          console.log(user)
        }
        else{
          console.log(response)
        }
      })
    )
  }

}
```

- в методе компонента `auth` вызовите метод сервиса `authService`, предварительно запросив его в конструкторе компонента.

```
  login(){
    this.authService.login(this.model).subscribe({next: r => console.log(r),
error: e => console.log(e)})
  }
```

Замечание: Observable-объекты - это ленивые объекты, на которые надо подписаться, чтобы получить результат. После успешного ответа от API, мы сохраняем пользователя в `localStorage` браузера, сериализуя его в json.

- после успешной авторизации надо выполнить переход на компонент `home`. Для этого создайте объект роутера:

```
router:Router = new Router()
```

- добавьте логику перехода в методе `login` компонента `auth`:

```
  login(){
    this.authService.login(this.model).subscribe({next: r =>
{this.router.navigate(["home"])}, error: e => console.log(e)})
  }
```

- проверьте `localStorage` на предмет объекта по ключу `user`

Регистрация

- создайте новый компонент `sign`
- добавьте в `authService` метод `register`:

```
  register(model:any){
    return this.http.post(this.baseUrl +"Users/", model)
  }
```

- теперь в компоненте `sign` создайте метод `sign`, который вызывает метод сервиса:

```
  sign(){
    this.authService.register(this.model).subscribe({next: r => console.log(r),
error: e => console.log(e.error)})
  }
```

- в шаблоне компонента `sign` в форме примените метод `sign` одноименного компонента:

```
<div class="sign">
    <h1> Регистрация </h1>
    <form (ngSubmit)= "sign()" autocomplete="off">
        <p>
            <mat-form-field>
                <mat-label>Login</mat-label>
                <input matInput name="login" type="text" [(ngModel)] =
"model.login" autocomplete="off" />
            </mat-form-field>
        </p>
        <p>
            <mat-form-field>
                <mat-label>Password</mat-label>
                <input matInput name="password" type="password" [(ngModel)]
="model.password" autocomplete="off" />
            </mat-form-field>
        </p>
        <button color="primary" mat-flat-button type="submit">
```

```
            Создать
        </button>
    </form>
</div>
```

- проверьте импорт модулей в компоненте `sign`

```
imports: [MatButton, MatFormField, CommonModule, MatLabel, FormsModule, MatIcon,
MatInputModule]
```

Задание: при успешной регистрации перейдите на компонент авторизации.

Условный рендеринг и работа с состоянием

- создайте кнопку выхода в шаблоне компонента `header`

```
    <button mat-button type="button">
        <mat-icon>
          logout
        </mat-icon>
        Выход
    </button>
```

- если пользователь авторизован, то надо показывать кнопку выхода и кнопку списка всех пользоватлей,а если нет, то не показывать кнопки. Примените директиву для условного рендеринга к атрибуту `button`:

```
*ngIf="authService.currentUser$ | async"
```

- в сервисе `authService` напишите логику метода для выхода:

```
  logout(){
    localStorage.removeItem("user")
    this.currentUserSource.next(null!);
  }
```

- далее метод будет выполняться на событии `click`:

```
(click)="authService.logout()"
```

Замечание: объект `authService` в конструкторе должен быть `public`, чтобы можно было применять методы сервиса в шаблоне компонента.

Задание: после выполения метода `logout()` надо перейти на компонент авторизации с помощью роутера. Настройте логику маршрутизации из сервиса `AuthService`.

Задание: добавьте кнопку "Регистрация" в компонент `header`.

Валидация в формах Angular

Входные и выходные параметры

Interceptors

Маршруты по сегментам

Spinner loading

Пакет ngx-toastr

Guards

```
ng g guard auth --skip-tests
```

HttpParams

HttpHeaders

```
    postData(user: User){
        const myHeaders = new HttpHeaders().set("Accept", "application/json");
        return this.http.post("http://localhost:3000/postuser", user,
{headers:myHeaders});
    }
```

query params

```
  <a routerLink="/item/5" [queryParams]="{>product':'phone', 'price': 200}">Item
5</a>
```

CanDeactivate на форме регистрации

`canActivate` проверяет возможность перехода на определенный компонент, а `canDeactivate` проверяет возможность ухода с определенного компонента

Bootwatch

- https://bootswatch.com/
- npm install bootwatch

Shared Module

`ng g m shared --flat`

Задание

- обработать исключение `Npgsql.PostgresException`, которое вылетает когда база данных не создана.
- обработать исключение соединения с сервером