Sprint 2 Введение в Angluar

Создание проекта Angular

- установить инструмент командной строки Angular `npm install -g @angular/cli@latest`
- создание проекта `ng new SportStore.Angular`

Примечание: при создании проекта настройки: препроцессор scss, ssr off

- внести в gitignore папку `node_modules` и файл `.angular`

Обзор архитектуры приложения Angular

Проверка работоспособности `ng serve`

при запуске не отправлять данные по статистике

Создание компонента

Создайте новый компонент `home`. Для этого создайте в папке `src` новую папку `components` и перейдите в нее.

```
ng g c home --skip-tests
```

Рендеринг компонента в главном компоненте

В шаблоне компонента `app` подключите новый компонент `home`.

app.component.html

```
<app-home/>
```

В компоненте `app` импортируйте компонент `home`

```
import { Component } from '@angular/core';
import { HomeComponent } from "../components/home/home.component";

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [HomeComponent],
  templateUrl: './app.component.html',
```

```
  styleUrl: './app.component.scss'
})
export class AppComponent {
  title = 'SportStore.Angular';
}
```

Настройка HttpClient

Найдите в проекте файл конфигурации главного компонента `app.config.ts` и замените в нем код. Здесь определяется провайтер для работы с http.

app.config.ts

```
import { provideHttpClient, withInterceptorsFromDi } from '@angular/common/http';

export const appConfig: ApplicationConfig = {
  providers: [

            ...
            provideHttpClient(withInterceptorsFromDi())
            ...

            ]
};
```

Observable. Подписка на запрос

- создайте папку `models`, в котороой создайте файл `user.ts`

```
export default interface User {
  id: string;
  name: string;
}
```

- в папке `src` создайте папку `services` и выполните команду:

```
ng g s userslocal --skip-tests
```

В результате сгенерируется файл сервиса.

```
import { Injectable } from '@angular/core';
import User from '../models/user'

@Injectable({
```

```
  providedIn: 'root'
})
export class UsersLocalService {

  getLocalUsers(): User[] {
    var users = [{ "id": "1", "name": "user1" },
                 { "id": "2", "name": "user2" },
                 { "id": "3", "name": "user3" }]
    return users;

  }

}
```

Сделайте запрос к локальной коллекции пользователей в компоненте home;

home.component.ts

```
import { CommonModule } from '@angular/common';
import { HttpClient } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import  User from '../../models/user'
import { UsersLocalService } from '../../services/userslocal.service';

@Component({
  selector: 'app-home',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './home.component.html',
  styleUrl: './home.component.scss'
})


export class HomeComponent implements OnInit {

  users: User[] = []
  title: string = "Home"

  constructor(private usersLocalService:UsersLocalService) { }

  ngOnInit(): void {
    this.users = this.usersLocalService.getLocalUsers();
  }
}
```

Управляющие конструкции в шаблоне компонента

- в шаблоне компонента home поместите следующий код:

```
<h3> Пользователи </h3>

<ol *ngFor="let user of users">
    <li> {{user.name}}</li>
</ol>
```

или вы можете в шаблоне напрямую обращаться к сервису, если в конструкторе сделаете его `public`.

```
<ul *ngFor="let user of usersLocalService.getLocalUsers()">
    <li> {{user.name}}</li>
</ul>
```

`*ngFor` - это директива, которая является простым циклом for. Для включения этой функциональности надо импортировать в компонент `home` модуль `CommonModule`.

Получение данных API

- теперь измените компонент `home` для получение данных о пользователях через API.

home.component.ts

```
import { CommonModule } from '@angular/common';
import { HttpClient } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import  User from '../../models/user'
import getLocalUsers from '../../services/users.service'

@Component({
  selector: 'app-home',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './home.component.html',
  styleUrl: './home.component.scss'
})


export class HomeComponent implements OnInit {

  users: User[] = []
  title: string = "Home"

  constructor(private http:HttpClient) { }

  ngOnInit(): void {
    //this.users = this.usersLocalService.getLocalUsers();
    this.getUsers();
  }
```

```
  getUsers() {
    this.http.get<User[]>('http://localhost:5290/User').subscribe({
      next: response => this.users = response,
      error: error => console.log(error)
    })
  }

}
```

Cors. Подключение и конфигурация

По умолчанию политика браузера такова, что он не разрешает совершать запросы к ресурсу, который находится в другом домене, если принимающая сторона явно не разрешает это. То есть наше приложение Angular не сможет получить ответ от API, пока API не разрешит это. Эта политика называетяс CORS (Cross-Origin Resource Sharing).

Теперь вернитесь в проект API и разрешите обращение к нему от всех источников и заголовков.

Program.cs

```
builder.Services.AddCors();
...
app.UseCors(o => o.AllowAnyOrigin().AllowAnyHeader());
```

Задание: создайте `usersService` получите данные из API в шаблон компонента `home`.

Установка пакета Angular Material

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

При установке вас попросят выбрать тему, типографию и анимацию.

Вывод пользователей в виде таблицы

В шаблоне компонента `home` применяются элементы из библиотеки пользовательского интерфейса `Angular Material`. Для отображения таблицы нужно импортировать модель `MatTableModule`,

```
import {MatTableModule} from '@angular/material/table';
```

а также объявить новое свойство `displayedColumns` в компоненте `home`, которое нужно для отображение таблицы.

```
displayedColumns: string[] = ['id', 'name'];
```

home.component.html

```
<h1> {{title}}</h1>

<table mat-table [dataSource]="users" class="mat-elevation-z8">
  <ng-container matColumnDef="id">
    <th mat-header-cell *matHeaderCellDef> Guid </th>
    <td mat-cell *matCellDef="let element"> {{element.id}} </td>
  </ng-container>

  <ng-container matColumnDef="name">
    <th mat-header-cell *matHeaderCellDef> Name </th>
    <td mat-cell *matCellDef="let element"> {{element.name}} </td>
  </ng-container>

  <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
  <tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>
</table>
```

Настройка https сертификата для localhost (опция)

Создание центра сертификации

- создайте файл `openssl.conf`

```
[req]
distinguished_name = req_distinguished_name
prompt = no
default_bits = 2048
default_md = sha256
[req_distinguished_name]
CN = localhost
emailAddress = test@git.scc
```

- выполните команду только в `Git Bash` в директории, где находится файл `openssl.conf`:

```
openssl req -x509 -newkey rsa:4096 -days 365 -nodes -keyout root_ca.key -out
root_ca.crt -config openssl.conf
```

На выходе генерируются два файла: `root_ca.key` и `root_ca.crt`

Генерация приватного ключа

```
openssl genrsa -out localhost.key 2048
```

На выходе генерируются `localhost.key`

Создание запрос на подписывание ключа к центру сертификации

```
openssl req -new -key localhost.key -out localhost.csr -config openssl.conf
```

На выходе генерируются `localhost.csr`

Подпись запроса нашим центром сертификации

- создайте файл `localhost.ext`

```
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
subjectAltName=@alt_names
[alt_names]
DNS.1=localhost
DNS.2=backend
IP.1=127.0.0.1
IP.2= {id}
```

Замечание: вместо {ip} подставьте ip вашего компьютера

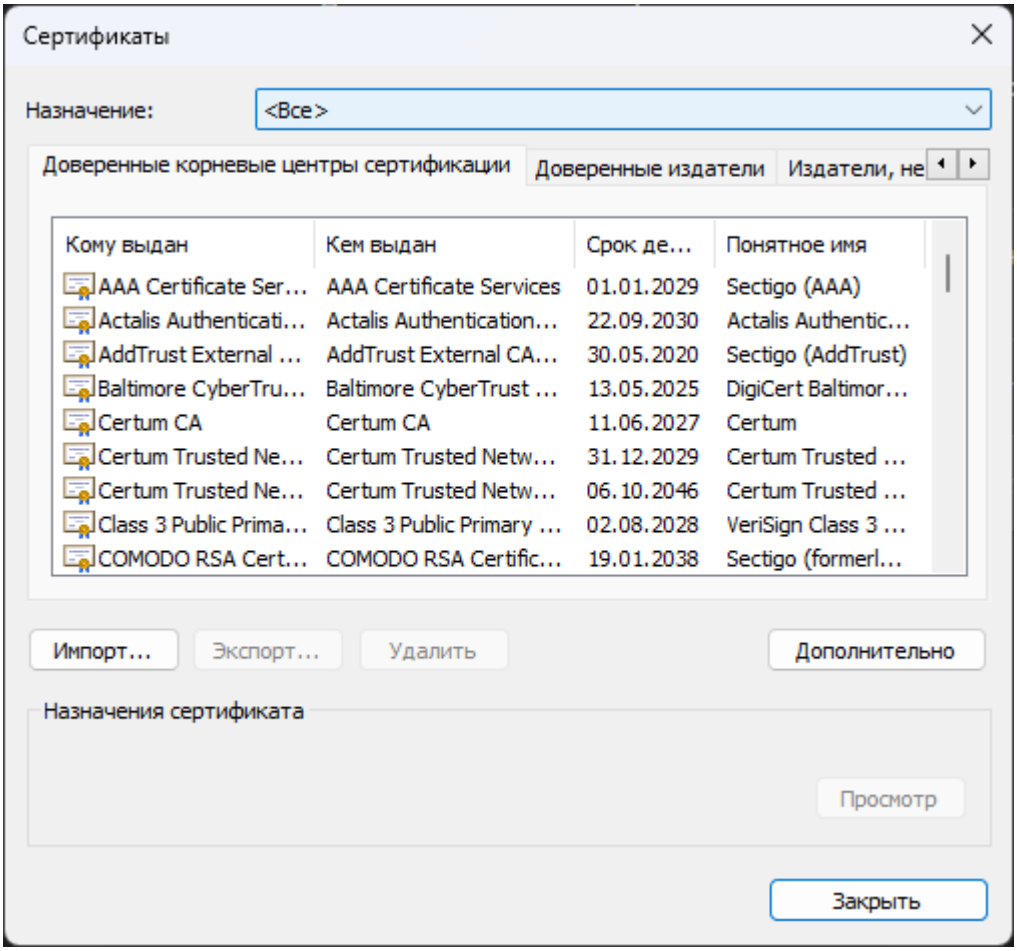
- выполните команду

openssl x509 -req
-CA root_ca.crt
-CAkey root_ca.key
-in localhost.csr
-out localhost.crt
-days 365
-CAcreateserial
-extfile localhost.ext

На выходе генерируются `localhost.crt`

Установить центр сертификации в Chrome

- Настройки -> Конфиденциальность и безопасность -> Безопасность -> Настроить сертификаты
- найдите доверенные центры сертификации и импортируйте файл `localhost.crt`



Настройка ssl для localhost в приложении Angular:

- в файле `angular.json` вставьте в `options` в секцию `serve` с указанием пути к ssl сертификату и ключу.

angular.json

```
    "options": {
            "sslCert": "./src/crt/localhost.crt",
            "sslKey": "./src/crt/localhost.key",
            "ssl": true
```

- запустите приложение `npm run start` или `ng serve`. Приложение должно работать на https

Примечание: В этой статье описывается подробно процесс создания самоподписанного сертфиката для localhost. https://dzen.ru/a/ZQ4nAsQZ6GkuFw7_