**Important notes (The Juice)**

App.module.ts = In this file we can control the routing

Create new components = cd ClientApp, ng g c <component\_name> --module app

Angular for loop in the .html = <div \*ngFor=”let flight of searchResult”> {{flight}} </div>

In our Controller we are creating the endpoint with the needed parameter ( [HttpGet] )

In order to see the port that .Net uses you have to go to the file Properties > launchSettings.json

Connectivity between .Net and Angular (REST API of .Net -> Angular downloads the json that the REST API provides and uses it)

In order to install openapi = npm install -d ng-openapi-gen

In order to create automatically the files for the client API we should use the command npx ng-openapi-gen

In file services.ts we can find the name of the service that we can use it to communicate with the endpoint

You can insert an event listener in the html code like = <button (click)=”search()” … >

In order to insert a link to an another internal page we have to do the following = [routerLink] = “[‘/book-flight’]”

SPA (Single Page Application) -> They are complete applications that run in the user’s browser. The main job of the SPA is to try handling anything locally rather than sending requests to the server. Less communication with the server.

There are two parameter for the classes: 1) [ApiController]=is for external usage so that an another like Angular can connect to this controller, 2) [Controller] = with this it won’t be available to connect from another projects. ( <https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-7.0> )

Angular uses TypeScript by default. Components, Selectors, Decorators



**What is the component decorator in Angular?**

One-way Binding, Event Binding ((click) = incrementNow())

**Basic Routing**

When I click a specific url then a specific component is loaded and the rest page remains unchanged. The unchanged part is the AppComponent. The containers which is changed is called router-outlet.

Path + component is called Route

[routerLink] -> we use it as an attribute in html in order to specify the route that we are going to use when the element is going to be clicked

**Creating and Routing**

ClientApp -> open in terminal -> ng g c search-flights --module app (ng for angular, g for generate, c for component, module app in order to attach it to our app module)

**Visualizing a List in a Component**

Npm install @fontawesome/fontawesome-free

In angular.json file we have to insert the library that we just downloaded in order to apply it to our angular project (architect -> styles -> "node\_modules/@fortawesome/fontawesome-free/css/all.css")

With the use of angular.json we can configure our builds

Went to search-flights.html in order to change the html code (used some Bootstrap)

Interfaces -> they are used as abstractions. Then we created in the TypeScript class some interfaces and then we extended the implementation and changed the styling of the .html also

**Getting Lists from Web API**

The Server Side Application must provide a way to communicate with the Client Side Application which is our Angular app in that case. The Server Side provides are called web services.

The Proxy is the program that redirects as from the C# .NET to the Angular program. We have to ports for each of these elements. We can find these ports in the file Properties/launchSettings.json

The name of the endpoint is Search (

[HttpGet]

Public IEnumerable <string> Search() => new string[] {…};

)

Records are immutable data structures. -> a Record = public record TimePlaceRm(string Place, DateTime Time);

An instantiation of a Record = var original new TimePlaceRm (Place: “Los Angeles”, Time: DateTime.Toay);

I cannot change the fields of a record after their initialization but I can copy the original and change it = var copy = original with {Place = “Berlin”};

Setting up Swagger and Open API = Tools->NuGet Package Manager, Manage NuGet Packages for Solution..,Browse,Search for Swashbuckle.AspNetCore,download the latest version. I’m using the port of the C# and I have to use the following route https://localhost:<port>/swagger/index.html

From the Angular side we need some code which will connect to our API to get the search results from our database. We can use a plugin that is going to read the result of swagger in order to get the data. So this is the Open API and the Swagger is using it’s output in order to show the results of it in a UI environment.

**Downloading Data from Web API using Angular**

We will install an npm package which is called ng open api generation which automatically creates the client side code.

If I want to install a package only for the development environment I have to use the following command = npm install -d <package name>

Swashbuckle Library (the library is used to generate documentation for our API) -> Swagger.json (the documentation for the API) -> ng-openapi-gen (takes the swagger.json file as input) -> Rest Client API (generates the code in order to connect with the back-end in order to get the data)

Cross-Origin Resource Sharing (CORS) = Program.cs, which domains are allowed to connect to our application this is what we are configuring with the following parameter app.UseCors(builder => builder.WithOrigins(“\*”));

Npm install -d ng-openapi-gen, Under ClientApp create a file named ng-openapi-gen.json, We don’t use https for the openapi-gen so that’s why we have to go to launchSettings.json and get the port of the http url that is there

Npx ng-openapi-gen = with npx we can execute packages, with this command we are generating the client API, we have to run this command while the app is running so that we can generate the files.

Services are injectable (dependency injection). Get a service into my class = <https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection>

**Parameterizing Angular Routes**

Npx ng g c book-flight --module app = create a new component and then add routes in the file app.module.ts

In order to add a rout in a button you have to use the following attribute = [routerLink] = “[’/book-flight’]”

Sending flightId parameter = we have to change the book-flight route to book-flight/:flightId and then I have to change the routerLink [‘/book-flight’, flight.id]. Then we have to change the file book-flight.component.ts (import ActivatedRoute, counstruvtor(private route: ActivatedRoute) { }, and in ngOnInit() = this.route.paramMap.subscribe(p => this.findFlight(p.get(“flightId”)), we have also to declare in the class the parameter flightId: string = ‘not loaded’)

Create a new endpoint in the FlightController in order to check if we have the specific id into our collection.

**Documenting HTTP Response Codes**

In an action if we use the ActionResult class we can return also a status id which is important in order to handle the situation.

Above the controller we can add the [ProducesResponseType(400)], [ProducesResponseType(500)], [ProducesResponseType(typeof(FlightRm), 200)] in order to document them in Swagger UI

Parameterizing routes

**Posting Data**

Create a new component = npx ng g c register-passenger --module app, copy the html from the repository