Projectbegin - Notes

Architecture:

- angular for Front-End
- .Net for Back-End
- Google Auth for Accounts (which I can implement using firebase afaik)
- Firebase for Hosting
- Firestore (or MongoDB + MongoDBAtlas)
    - Pricing
    - Firebase Pricing

The Application uses german as Domain language.

Requirements:

Rezept:

- Rezeptname
- Zutaten können hingefügt werden
    - Name der Zutat
    - Mengenangabe
    - Einheit (ml/gramm/…)
- [Optional] Beschreibung
- [Optional] Zubereitungsdauer/n (plural? 10min schneiden, 20min Ofen)
- [Optional] Angabe von resultierenden Portionen
- [Optional] Eine oder mehrere Kategorieren, die man wie “Tags” hinzufügen kann wie z.B. “Auflauf”, “Pfannengericht“, “Kaltspeise“, “Nachtisch”, “Getränk”, …
- Man sollte zusätzliche (wiederverwendbare und filterbare?) Notizen hinfügen können wie z.B. “Kann auch im Instantpot erledigt werden“

Technical data:

- Erstelldatum
- Ersteller

NOT MVP:

In future:

- Calculation Function to automatically adjust the Zutaten-Mengenangabe based on selected “gewünschte Menge der resultierenden Portionen”
    - To be discussed: if no portion amount is given, automatically assume two? (so you can still use the Caluclation Function)
- Use the accounts
- Take into consideration, one could supply a new Zutat (should be automated) which already existed, but spelled differently (or even translated). So the app should contain a merge UI for merging two zutaten to one, replacing all references of ZutatA with new references to ZutatB, so ZutatA can be deleted. To avoid falsely created Zutaten the input box for Zutat should be a typeable, but also have a auto-completion “dropdown”.

The Abrreviation for the project shall be GKB.

does not need accessability or localization

Rezepte are often modified by different people, so there should be a edit button to edit directly, and a “copy” button which copies all data, and keeps a reference to the recept it was copied on (or a copy of it at the time it was refeences)

take care of CORS

.NET CORS Guide: What It Is and How to Enable It

Know-How

CRUD using C# REST

Angular CRUD Using .Net Core Web API

ENUM

Angular Basics: Working With Enums in Angular

In-Memory REST Interceptor

`TUTORIAL:`

Angular In-memory Web API tutorial: Mocking CRUD APIs in Angular - LogRocket Blog

TROUBLESHOOTING:

1. app.module.ts:

InMemoryModule must come AFTER HttpClientModule:

```
imports: [

…

 HttpClientModule,

 HttpClientInMemoryWebApiModule.forRoot(DataService, { dataEncapsulation: false })

],
```

2.

404 using Angular inMemoryWebApi

The problem was that I was returning the array and not an object from the data.service object.

Another easy to overlook piece of the InMemoryWebAPI is that the url you use has to correspond to the name of the array

`data.service.ts,`

`return {links: links};`

`links.service.ts:`

`private restUrl = "http://localhost:8080/api/links";`

Migrate to Standalone

Angular

C#

- Best Practises for XML Documentations by UNIT (this must be good) Best Practices for XML Documentation | NUnit Docs

Unit Testing

Testing in .NET - .NET

How to get your appsettings.json values into DI in asp.net core
Create a web API with ASP.NET Core and MongoDB

Host web api in Docker Container
Host ASP.NET Core in Docker containers

CORS & SWAGGER

LINK: .NET CORS Guide: What It Is and How to Enable It
Enable Cross-Origin Requests (CORS) in ASP.NET Core

OR: IServiceCollection.AddCors C# (CSharp) Code Examples - HotExamples

```
public class Startup {
    public Startup(IConfiguration configuration) {
        Configuration = configuration;
    }

    public IConfiguration Configuration {
        get;
    }

    // This method gets called by the runtime.
    // Use this method to add services to the container.

    public void ConfigureServices(IServiceCollection services) {

        services.AddScoped < IDepartmentRepository, DepartmentRepository > ();
        services.AddScoped < IEmployeeRepository, EmployeeRepository > ();
        services.AddDbContext < APIDbContext > (options => options.UseSqlServer(Configuration.GetConnectionStrin
        services.AddSwaggerGen(options => {
            options.SwaggerDoc("v1", new OpenApiInfo {
                Title = "WEB API",
                    Version = "v1"
            });
        });



        //Enable CORS
        services.AddCors(c => {
            c.AddPolicy("AllowOrigin", options => options.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());
        });


        //JSON Serializer
        services.AddControllersWithViews().AddNewtonsoftJson(options => options.SerializerSettings.ReferenceLoop
        services.AddControllers();
    }




    // This method gets called by the runtime.
    // Use this method to configure the HTTP request pipeline.

    public void Configure(IApplicationBuilder app, IWebHostEnvironment env) {

        if (env.IsDevelopment()) {
            app.UseDeveloperExceptionPage();
        }

        app.UseCors(options => options.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());
        app.UseHttpsRedirection();
        app.UseRouting();
        app.UseAuthorization();
        app.UseEndpoints(endpoints => {
            endpoints.MapControllers();
        });


        app.UseSwagger();
        app.UseSwaggerUI(c => {
            c.SwaggerEndpoint("/swagger/v1/swagger.json", "WEB API");
            c.DocumentTitle = "WEB API";
            c.DocExpansion(DocExpansion.List);
        });



        app.UseStaticFiles(new StaticFileOptions {
            FileProvider = new PhysicalFileProvider(Path.Combine(Directory.GetCurrentDirectory(), "Photos")),
                RequestPath = "/Photos"
        });
    }
}

```

Compiler Warnings

To avoid the CS8618 warning replace the following:

`` `public string Name { get; set; }` ``

with this:

`` `public required string Name { get; set; }` ``

I dont know what life has become, but its EF Core recommendation, so we will follow: Working with nullable reference types - EF Core

Another approach would be to assign it with a default value. An empty string or a null forgiving operating null:

`` `public string Name { get; set; } `` = null!\`

or

`` `public string Name { get; set; } `` = string.Empty\`

MongoDB

- SetUp User/Pw: MongoDB what are the default user and password?
- The mongosh (shell) uses javascript Database Methods
- C# Quick Start
- C# Connection Guide
- Example Video (offcially by MongoDB) Build an ASP.NET app with the Mongo C# Driver
- MongoDB QuickTutorial for basic CRUD (C# Console) App MongoDB & C Sharp: CRUD Operations Tutorial | MongoDB
- MongoDB’s C# Driver GitHub - mongodb/mongo-csharp-driver: The Official C# .NET Driver for MongoDB install `dotnet add package MongoDB.Driver` (ofc within the api folder, and not the root folder which is only for git and a shared workspace in vscode)
- Microsofts Create a web API with ASP.NET Core and MongoDB
- Mongo/C# QuickReference Quick Reference
- Mongo’s C# Reference Namespaces

vscode workspace set up

this blog article helped, in figuring out, how to get the launch and tasks file for the dotnet webapi project, which were missing.

VS Code + .NET - Debug a .NET Web App in Visual Studio Code | Jason Watmore's Blog

Using Docker + MongoDB + ASP.NET Core for deployment

Docker + MongoDB + .NET Core = a good time

Docker / Linux

Use windows terminal to connec to docker container’s bash:

```
docker exec -it container-name bash
```

ping another container from the bash:

```
ping another-container
```

to use ping install iputils first:

1. apt update
2. apt install iputils-ping

use pathing to run a specifc docker-compose file

```
docker-compose -f path/docker-compose.yml -d
```

-d is for dettached mode (to get your terminal back)

Delete all docker images at once (windows)

```
docker rmi $(docker images -q)
```

the -q stands for quietly

Delete all docker containers at once (windows):

```
docker rm --force $(docker ps -a -q)
```

-a stands for list all containers regardless if they are running or not.