OPENAPI

Een open API helpt een developer om te begrijpen wat exact een REST API doet zonder elke lijn code te moeten lezen. Het beschrijft de api zijn interface op een simpele manier zodat het snel te begrijpen is. Een open API zal een UI genereren om het testen van controllers sneller te maken. Er zijn een paar hoofdkenmerken voor open API's. Ze zijn beschikbaar voor de developers met weinig beperkingen met eventueel uitzondering op een registratie systeem. Ze zijn volledig vrij om te gebruiken zonder copyright restricties of andere soort zaken. De persoon die de API published kan zelf kiezen wat exact hij wilt op de open API. Om openAPI te implementeren heb ik gebruik gemaakt van Swagger.

1. Ga naar manage Nuget packages.
2. Installeer de Nuget package: Swashbuckle.AspNetCore
3. Voeg het toe bij de API
4. In de Startup klasse vermelden we bij ConfigureServices dat we Swagger gebruiken door services.AddSwaggerGen(); lijn toe te voegen.
5. We kunnen extra toevoegen om meer te vertellen over de API.

```
//OPENAPI
services.AddSwaggerGen(x =>
{
    x.SwaggerDoc("v1", new OpenApiInfo { Title = "Cloud API Project Kasper", Version = "v1" });
});
```

6. Bij de configure method schrijven we: app.UseSwagger();
7. Verder enabled we ook de UI. We geven hier het PATH naar de swagger json die alle endpoints zal lezen van de API. Dit zal de client generen.

```
//OPENAPI
app.UseSwagger();
app.UseSwaggerUI(c =>
{
    c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API");
});
```

Champion

POST /api/v1/champions

PUT /api/v1/champions

GET /api/v1/champions

DELETE /api/v1/champions/{id}

GET /api/v1/champions/{id}

GET /api/v1/champions/faction

Faction

GET /api/v1/factions

POST /api/v1/factions

GET /api/v1/factions/{id}

PUT /api/v1/factions/{id}

DELETE /api/v1/factions/{id}

Item

POST /api/v1/items

GET /api/v1/items

DELETE /api/v1/items/{id}

Responses

Code	Description	Links
200	Success Media type: text/plain Controls Accept header. Example Value | Schema	*No links*


```
[
  {
    "id": 0,
    "championName": "string",
    "championRole": "string",
    "releaseDate": "2021-05-31T17:57:47.824Z",
    "factionId": 0,
    "faction": {
      "id": 0,
      "factionName": "string",
      "factionDesciption": "string"
    },
    "championItem": [
      {
        "championID": 0,
        "champion": "string",
        "itemID": 0,
        "item": {
          "id": 0,
          "itemName": "string",
          "championItem": [
            "string"
          ]
        }
      }
    ]
  }
]
```

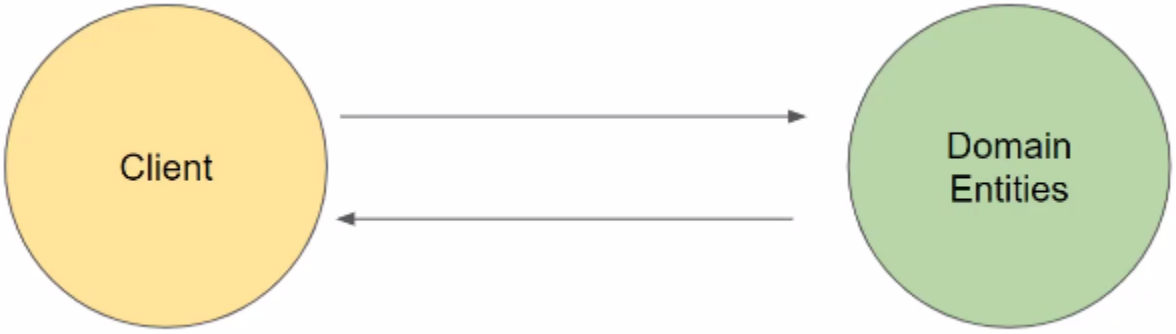

DELETE /api/v1/champions/{id}

GET /api/v1/champions/{id}

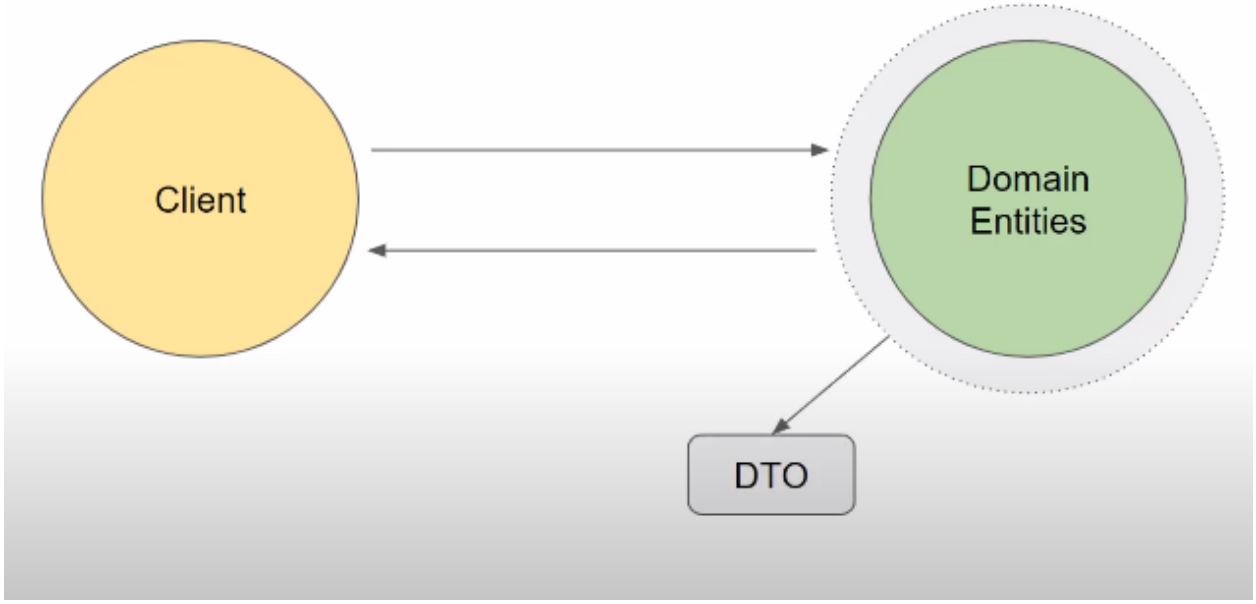
https://docs.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-5.0
https://docs.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-swashbuckle?view=aspnetcore-5.0&tabs=visual-studio

DTO

Kort voor Data Transfer Objects. Het is een object data informatie doorstuurt tussen 2 processen.



Normaal gezien ziet communicatie tussen een client en je data tabellen.



Een DTO is een soort van bescherming. Er wordt zo geen rechtstreekse manipulatie van de tabellen gedaan, maar in de plaats worden de DTO's eerst veranderd. In code gebeurd dit door gebruik te maken van mapping. Je maakt gebruik van de DTO klassen en verwijst die naar de echte klassen door maps te gebruiken.

8. Ga naar manage Nuget packages.
9. Installeer de Nuget package: Automapper.Extensions.Extensions.Microsoft.DependencyInjection
10. Voeg het toe bij de API
11. In de Startup klasse vermelden we bij ConfigureServices

```
//DTO
services.AddAutoMapper(AppDomain.CurrentDomain.GetAssemblies());
```

12. We maken een DTO variant van onze modellen. Dit wil dus zeggen dat voor Champion je een ChampionDTO hebt. Faction een FactionDTO enz.
13. We voegen een klasse toe Mapping Profile.

```
1 reference | KasperRuys, 5 hours ago | 1 author, 1 change
public class MappingProfile: Profile
{
    0 references | KasperRuys, 5 hours ago | 1 author, 1 change
    public MappingProfile()
    {
        CreateMap<FactionDTO, Faction>();

    }
}
```

Hier zullen we de mapping aanmaken tussen de 2 models.

14. Dan in onze controllers in plaats van rechtstreek bijvoorbeeld Faction te manipuleren moeten we nu de DTO manipuleren en verwijzen naar de echte Faction met mapping.

```
[HttpPost]
0 references | 0 changes | 0 authors, 0 changes
public IActionResult AddFaction([FromBody] FactionDTO factiondto)
{

    // DTO POST
    var faction = _mapper.Map<Faction>(factiondto);
    return Ok(faction);
```

CQRS