



# IT@AP

## Toegepaste informatica

### Technische documentatie GATAM- platform

Onderdeel van de stage  
ondersteund door de

## AP Hogeschool

en uitgevoerd op en begeleid door het bedrijf

## GATAM

Cuypers Jan  
Henri Yelko  
Heylen Roan  
Vervoort Kobe

Specialisatie IT & Software

# Inhoudstafel

<b>VERSIEBEHEER .....</b>	<b>3</b>
<b>TERMEN EN AFKORTINGEN .....</b>	<b>3</b>
<b>1. SAMENVATTING VAN DE OPDRACHT.....</b>	<b>4</b>
<b>2. IMPACT OP DE INFRASTRUCTUUR .....</b>	<b>4</b>
<b>4. RELEASE PLAN .....</b>	<b>5</b>
A. STAPPENPLAN INSTALLATIE .....	5
B. CONFIGURATIE VOOR PRODUCTIEOMGEVING .....	9
C. GITLAB CI/CD .....	10
<b>5. TECHNISCH DESIGN.....</b>	<b>12</b>
A. GEBRUIKTE TECHNOLOGIEËN .....	12
<b>6. EXTERNE SYSTEEMINTERFACES.....</b>	<b>21</b>
<b>7. DATAMIGRATIE.....</b>	<b>22</b>
<b>8. SECURITY EN AUTORISATIEROLLEN .....</b>	<b>22</b>
<b>9. UNIEKE STRUCTUREN.....</b>	<b>25</b>
<b>10. DOCUMENTATIE .....</b>	<b>29</b>
<b>11. BRONVERMELDING.....</b>	<b>30</b>

## Versiebeheer

Nr.	Datum	Verspreiding	Status	Wijziging
0.01	2024-12-19	Jan Cuypers	Eerste draft	Titelpagina, Samenvatting van de opdracht, Architectuur
0.02	2024-12-20	Roan Heylen	Tweede draft	Release plan, technologieën, configuratie, API-endpoints
0.03	2024-12-21	Kobe Vervoort	Derde draft	Security en autorisatie rollen, externe systeeminterfaces & klassediagram
1.00	2024-12-22	Kobe Vervoort Jan Cuypers Roan Heylen Yelko Henri	Finale versie	

---

## Termen en Afkortingen

Term	Omschrijving
CI/CD	Continuous Integration / Continuous deployment

## 1. Samenvatting van de opdracht

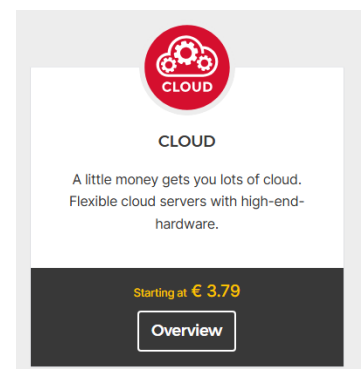
GATAM is een organisatie die zich inzet voor mensen die het label een grote afstand tot de arbeidsmarkt toegekend krijgen. Ze helpen deze mensen bij contactmomenten, waarbij er opgemerkt werd dat GATAM liever ook wat interactie buiten deze momenten wou. Dit wegens er anders een tijd altijd tussen zat dat ze actief konden vooruitgaan.

GATAM kwam op met het idee voor een applicatie waarbij trajectvolgers modules kunnen oplossen met vragen erin. Zo kunnen trajectvolgers buiten de contacturen nog altijd actief geëngageerd worden. Hierbij werden de studenten van AP Hogeschool ingeschakeld om een applicatie te ontwikkelen die het volgende mogelijk maakt: modules met vragen erin aanmaken, de vooruitgang en opvolging van deze modules bekijken, en een infrastructuur bieden om deze modules te creëren en gebruikers te beheren.

## 2. Impact op de infrastructuur

Momenteel heeft GATAM nog geen bestaande infrastructuur, wat betekent dat de infrastructuur nog bepaald moet worden. Hiervoor werd dus gekozen voor Hetzner. Hetzner goedkoper tegenover andere cloud providers, en daardoor is er dus voor Hetzner Cloud gekozen.

Er zijn verschillende plannen beschikbaar: Shared vCPU en Dedicated vCPU. Er wordt gekozen voor een Shared vCPU, omdat deze voldoet aan de behoeften van de applicatie. Bij nader onderzoek blijkt dat de goedkoopste versie voldoende is voor het systeem. Een ander groot voordeel is dat Hetzner al zijn servers in Europa heeft. Er is keuze uit servers uit Duitsland en Finland de logische keuze hier zou Duitsland zijn.



	VCPU	RAM	NVME SSD	TRAFFIC INCL. ⓘ	IPV4	HOURLY ⓘ	MONTHLY
CX22	2	4 GB	40 GB	20 TB	✓	€ 0.006	€ 3.79 <sub>max.</sub>

## 4. Release plan

### A. *Stappenplan installatie*

De applicatie kan op 3 manieren opgezet worden:

- Manueel
- Docker
- GitLab CI/CD

Bij het opzetten van het project moeten er enkele environment variabelen worden geconfigureerd:

#### **Backend**

<i>Key</i>	<i>Uitleg</i>
<i>MSSQLSRV_DB_URL</i>	De database connection string voor het connecteren met de database
<i>MSSQLSRV_PASSWORD</i>	Het password dat zal gebruikt worden voor het administrator account “sa” in de MSSQL-database.
<i>JwtIssuerSigningKey</i>	De signing key gebruikt voor de JWT-tokens.
<i>DefaultAdministratorEmail</i>	De standaard naam voor het administrator email.  Bv: “administrator”
<i>DefaultEmail</i>	Het prefix voor het standaard email dat gebruikt wordt op mails te generen. Het prefix wordt ook gebruikt voor het administrator account.  Bv: “gatam.com”

<i>Key</i>	<i>Uitleg</i>
<i>Auth0Domain</i>	De Auth0 API URL.
<i>Auth0Audience</i>	De audience token voor toegang met de API.
<i>Auth0ClientId</i>	De token voor de Auth0 client.
<i>Auth0ClientSecret</i>	De secret token voor validatie van de Auth0 client.

Voor meer informatie bij het vinden van Auth0 credentials:

<https://auth0.com/docs/quickstart/webapp/aspnet-core-blazor-server/interactive>

## Frontend

In de volgende locaties bevindt zich de configuratie voor de variabelen in de frontends:

- */AP.GATAM.FrontendUsers/wwwroot/appsettings.json*
- */AP.GATAM.FrontendGatam/wwwroot/appsettings.json*

Hierbij vinden we enkele variabelen in de vorm van JSON terug.

```
{
  "Auth0": {
    "Authority": "AUTHORITY",
    "ClientId": "CLIENTID",
    "Audience": "AUDIENCE"
  },
  "ASPNETCORE_ENVIRONMENT": "Development"
}
```

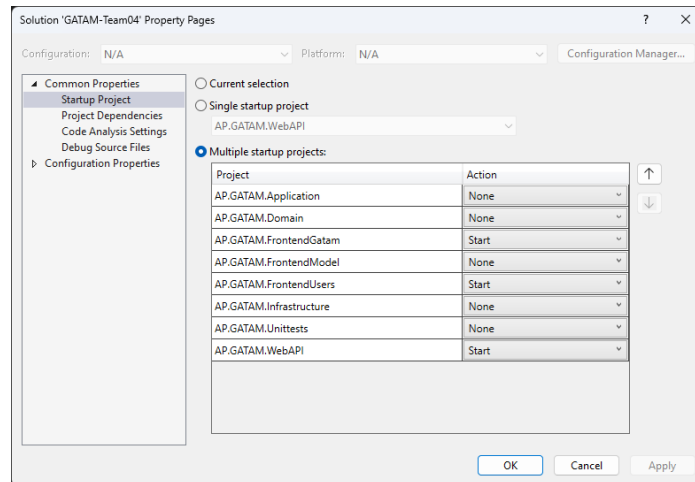
Net zoals in de backend zullen de Auth0 credentials moeten geconfigureerd worden.

## Manueel (lokaal)

Vereisten:

- Visual Studio, JetBrains Rider of IDE naar keuze
- MSSQL Server

Bij het opstarten, zullen er meerdere projecten geselecteerd moeten worden als opstartproject.



Deze projecten zullen lokaal opstarten en beschikbaar zijn op de volgende poorten:

- AP.GATAM.FrontendGatam
  - Poort 443
- AP.GATAM.FrontendUsers
  - Poort 4443
- AP.GATAM.WebAPI
  - Poort 8080

## Docker

*Voor Docker, zie eerst B. Configuratie voor productieomgeving.*

Vereisten:

- Docker compose
- (Voor Windows) WSL

Het docker compose script bevindt zich in de folder: `/docker/docker-compose.yml`. En kan uitgevoerd worden door het volgende commando:

```
docker-compose up
```

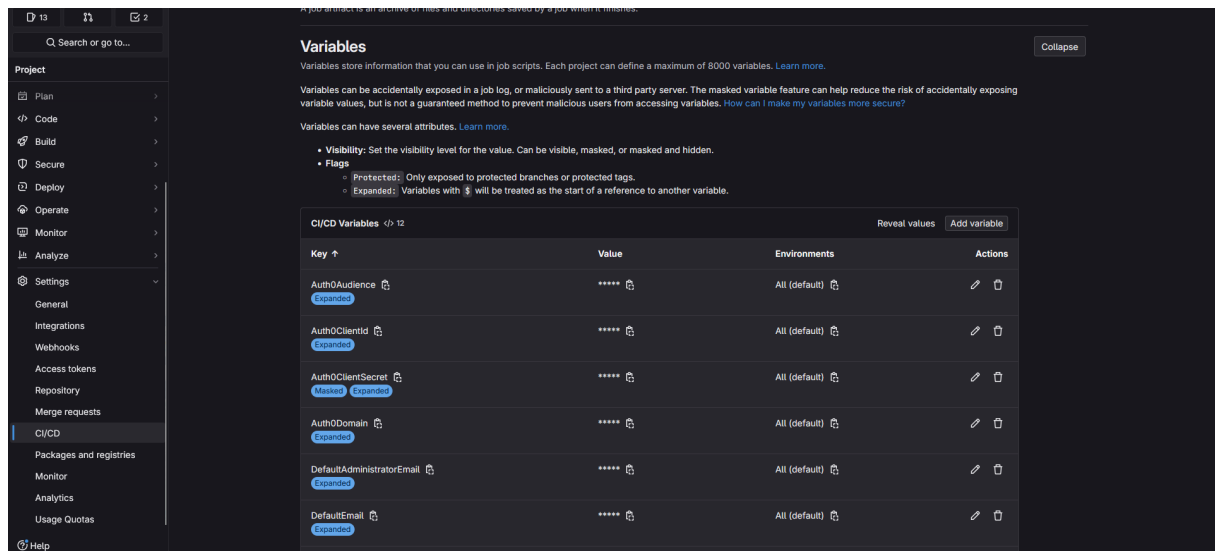
## GitLab CI/CD

Voor GitLab, zie eerst *B. Configuratie voor productieomgeving*

Het GitLab CI/CD script bevindt zich onder `.gitlab-ci.yml`.

Bij het opstellen van een GitLab project zal dit automatisch gedetecteerd worden, builden en deployen bij elke merge request, of commit op main.

Environment variables worden automatisch doorgegeven via GitLab, onder *Settings > CI/CD > Variables*.



**Variables**

Variables store information that you can use in job scripts. Each project can define a maximum of 8000 variables. [Learn more.](#)

Variables can be accidentally exposed in a job log, or maliciously sent to a third party server. The masked variable feature can help reduce the risk of accidentally exposing variable values, but is not a guaranteed method to prevent malicious users from accessing variables. [How can I make my variables more secure?](#)

Variables can have several attributes. [Learn more.](#)

- **Visibility:** Set the visibility level for the value. Can be visible, masked, or masked and hidden.
- **Flags**
  - **Protected:** Only exposed to protected branches or protected tags.
  - **Expanded:** Variables with `$` will be treated as the start of a reference to another variable.

Key	Value	Environments	Actions
AuthOAudience	*****	All (default)	<a href="#">Expanded</a> <a href="#">Edit</a> <a href="#">Delete</a>
AuthOClientid	*****	All (default)	<a href="#">Expanded</a> <a href="#">Edit</a> <a href="#">Delete</a>
AuthOClientSecret	*****	All (default)	<a href="#">Masked</a> <a href="#">Expanded</a> <a href="#">Edit</a> <a href="#">Delete</a>
AuthODomain	*****	All (default)	<a href="#">Expanded</a> <a href="#">Edit</a> <a href="#">Delete</a>
DefaultAdministratorEmail	*****	All (default)	<a href="#">Expanded</a> <a href="#">Edit</a> <a href="#">Delete</a>
DefaultEmail	*****	All (default)	<a href="#">Expanded</a> <a href="#">Edit</a> <a href="#">Delete</a>



## B. Configuratie voor productieomgeving

Er zijn extra environment variables die geconfigureerd moeten zijn voor productiegebruik.

### Backend

Key	Uitleg
<i>PROJECT_HOST</i>	Het domein waarop de frontend wordt gehost voor de administratieve GATAM gebruikers.
<i>PROJECT_USERS_HOST</i>	Het domein waarop de frontend wordt gehost voor de niet-administratieve GATAM gebruikers.

### Frontend

In de volgende locaties bevindt zich de configuratie voor de variabelen in de frontends:

- */AP.GATAM.FrontendUsers/wwwroot/appsettings.json*
- */AP.GATAM.FrontendGatam/wwwroot/appsettings.json*

```
{
  "FRONTEND_GATAM": "asp-gatam-04.us.to",
  "FRONTEND_USERS": "asp-gatam-04-users.us.to",
  "ASPNETCORE_ENVIRONMENT": "Production"
}
```

Deze variabelen worden hier gedefinieerd voor het te kunnen redirecten naar een andere frontend. Maar ook, voor het te kunnen vinden van de API host.

- FRONTEND\_GATAM
  - Het domein voor de frontend van de administratieve GATAM gebruikers.  
Hieronder wordt ook de API gehost.
- FRONTEND\_USERS
  - Het domein voor de frontend van de niet-administratieve GATAM gebruikers.
- ASPNETCORE\_ENVIRONMENT
  - Hierdoor verandert het environment naar productie, en zal het de bovenste twee variabelen gebruiken voor redirects.

## C. GitLab CI/CD

De CI/CD werkt als volgt.

### De build fase

1. De CI/CD bouwt automatisch bij de volgende criteria:
  - a. Bij elk commit op de main branch
  - b. Voor elk merge request

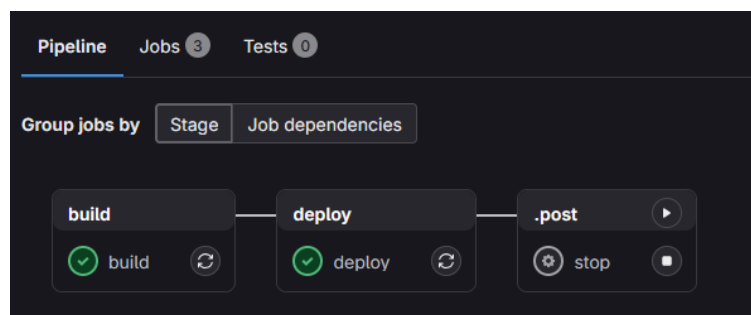
### De deploy fase

2. De pipeline verwijderd alle oude images door *docker system prune -f* uit te voeren.
3. De pipeline runt een *docker compose up* commando in de */docker/* folder.

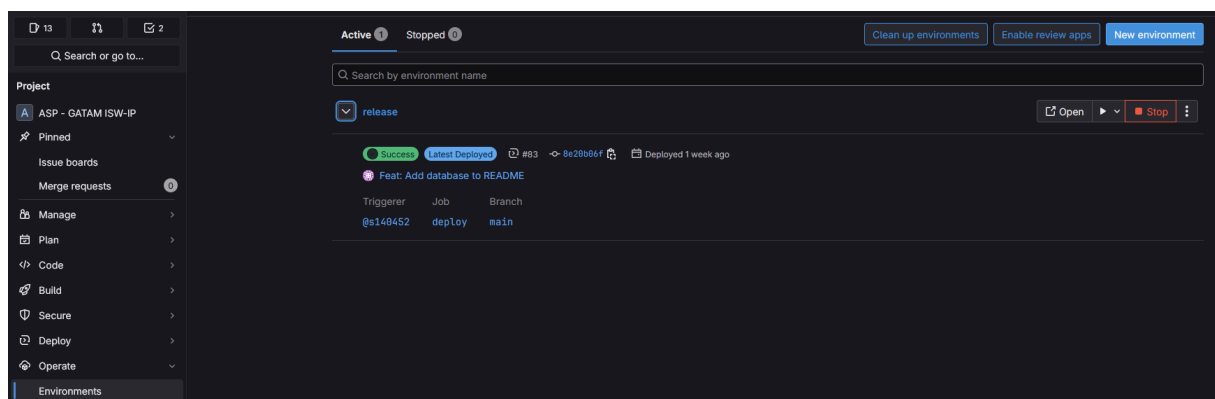
De pipeline laat dan al het buildwerk over aan het *docker-compose.yml* script.  
Dit script maakt 4 verschillende containers:

- Migration
  - Het migration script zal bij elke build de database leegmaken en hierbij een migration op toepassen. Dit voorkomt dat migrations niet lukken als er al data aanwezig is. Het script bevindt zich in */docker/migrations/Dockerfile*
- Backend
- Frontend Gatam
- Frontend Users

Hier beneden zien we een overzicht van de verschillende stages dat het GitLab CI/CD script doorgaat:



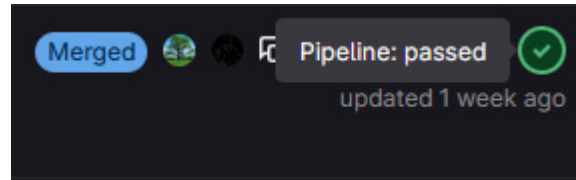
Bij het uitvoeren van een pipeline zal er zich een environment bevinden onder *Operate > Environments*.



## Merge requests

Bij de merge requests wordt er automatisch een “dummy” project opgebouwd, en worden alle tests automatisch uitgevoerd.






Dit zorgt ervoor dat bij elke merge request de pipeline succesvol zal moeten uitgevoerd worden, voordat men kan mergen.



## 5. Technisch design

### A. Gebruikte Technologieën

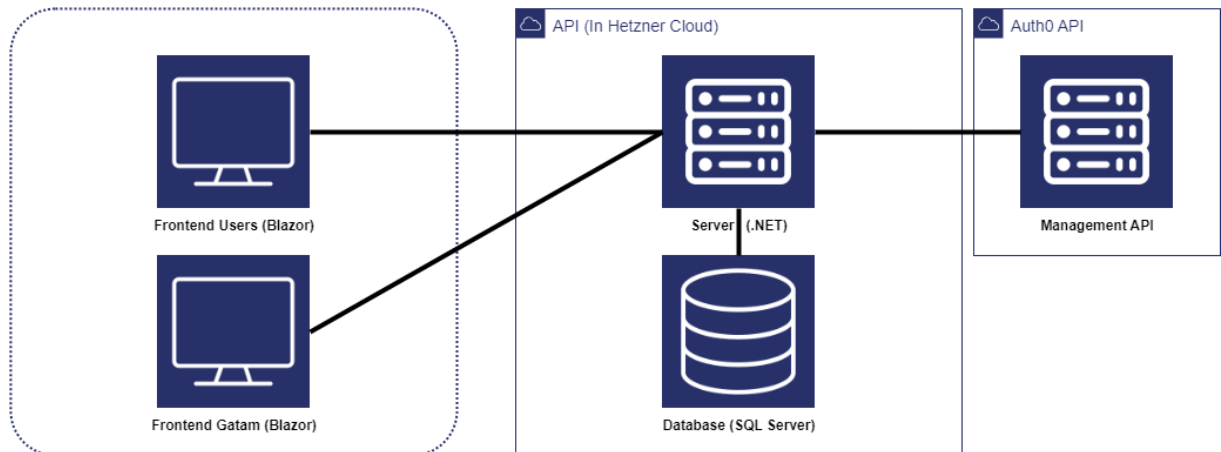
Onderstaande tabel toont de gebruikte technologieën met hun functie in het project:

Component	Logo	Naam	Gebruik	Licentie	Open-Source?
Frontend		Blazor	Blazor wordt gebruikt voor de frontend. Er zijn twee frontends voorzien. Één voor GATAM medewerkers en een andere voor trajectvolgers.	MIT	✓ Ja
Backend		ASP.NET	ASP.NET wordt gebruikt voor de backend. Waarbij het gaat over een API waarnaar de frontend benodigde data kan ophalen.  De backend is opgesteld volgens de principes van CLEAN Architecture.	Apache License 2.0	✓ Ja
Database		SQL Server	Microsoft SQL Server is gebruikt als database voor het bijhouden van de relationele data.	Proprietary License	✗ Nee
Authenticatie		Auth0	Auth0 wordt gebruikt voor de authenticatie van een gebruiker. Het beheert enkel en alleen authenticatiestatus (logged in, logged out) van een gebruiker.	Proprietary License	✗ Nee
Reverse Proxy		Traefik	Traefik is een reverse proxy, het is de middenman tussen de Docker containers en de buitenwereld.  In het geval van onze applicatie, wordt dit gebruikt om Docker containers onder verschillende domeinen beschikbaar te stellen.	Apache License 2.0	✓ Ja

Versiebeheer		GitLab	<p>GitLab wordt gebruikt voor versiebeheer.</p> <p>We gebruiken GitLab voor het deployen van de applicatie met een CI/CD. Waarvan merge requests hier ook gebruik maken in de vorm van een pipeline.</p>	MIT	✓ Ja
Schaling		Docker	<p>Docker is een container-based applicatie. Het zorgt ervoor dat elk component van de applicatie in een container runt.</p> <p>Het zorgt ervoor dat we elke component van de applicatie gemakkelijk kunnen schalen, beheren en opstarten.</p>	Apache 2.0	✓ Ja

## B. Architectuur

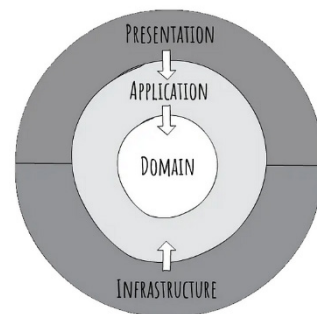
Onderstaande tekening toont een overzicht van de architectuur van de applicatie gebruikt tijdens development.



Tijdens development werd er ook gehandeld volgens de principes van CLEAN Architecture in de backend. We geven hier een zeer kort overzicht over:

In onze applicatie volgens CLEAN Architecture zijn er volgende layers:

- Application Core Layer (Application+ Domain) :
  - Business logic
  - Validation logic
  - Domain Model (Entities)
- Infrastructure:
  - Gateway to the outside world
  - Repository, Unit of Work, ...
  - External API's
- Presentation:
  - WebAPI
  - Middleware handling



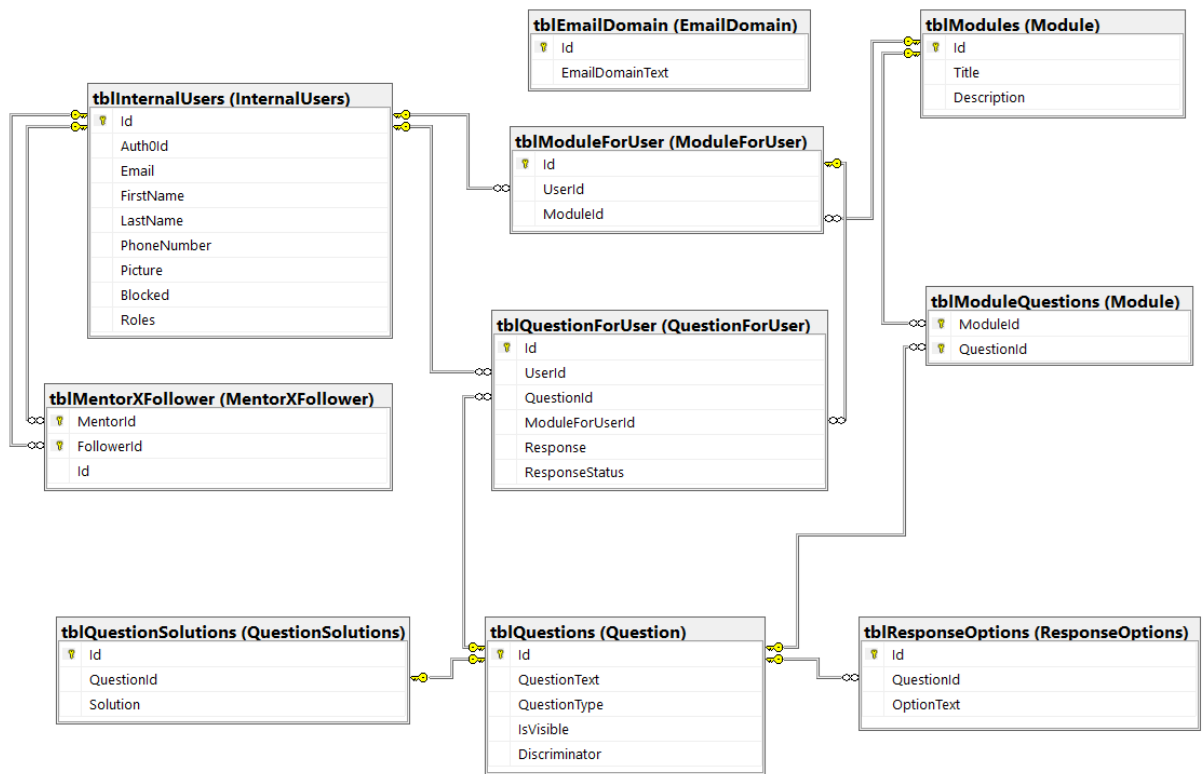
In de frontend is hier niet voor geopteerd, maar is wel een laag voor model klassen bij te houden gemaakt. Dit uit inspiratie aan de domain laag van CLEAN Architecture.

Op onderstaand klassediagram zien we alle relevante klassen die onze applicatie nodig heeft om te werken. Merk op dat er twee enumeratie types worden gebruikt.



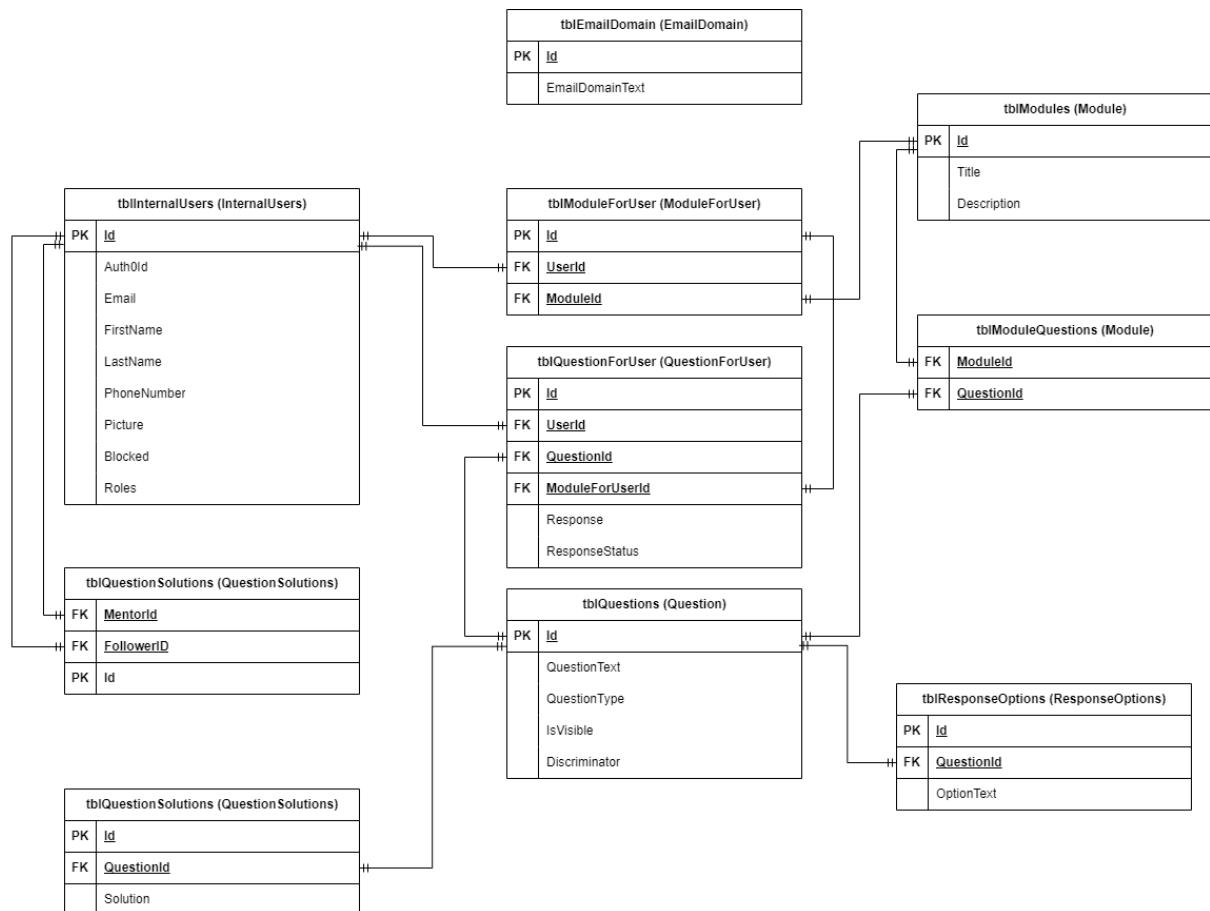
#### D. Conceptuele structuur databank

De onderstaande afbeelding bevat het Entity Relationship Diagram van de database:





Merk op dat een InternalUser **wel** nul of meerdere modules toegewezen kunnen hebben en dat dit niet gelimiteerd is op één. Een InternalUser kan om diezelfde reden ook nul of meerdere QuestionForUser's toegewezen hebben. Verder is het vanzelfsprekend dat een Module ook nul of meerdere Questions kan bevatten.



## E. API-Endpoints

In deze tabel vinden we de API-endpoints terug van de applicatie. Voor meer details is er in de bijlage een PDF toegevoegd met de API details vanuit OpenAPI.

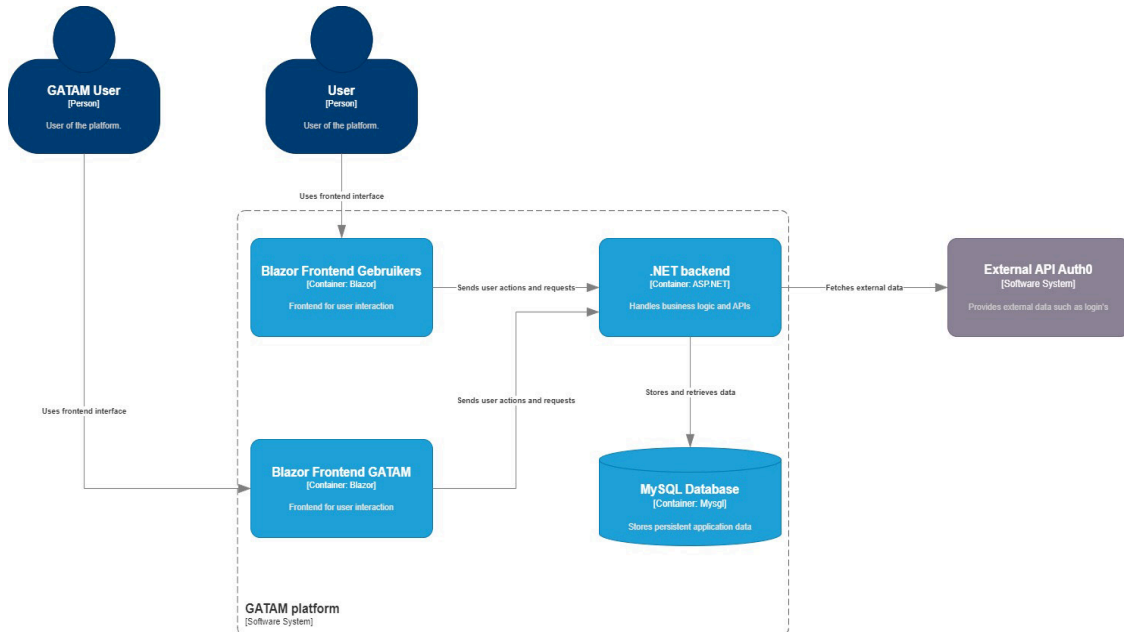
Methode	Parameters of body	Route (/api/v1/)	Return Waarde (indien van toepassing)	Statuscode
GET		/EmailDomain	Het default domein voor het genereren van emails	OK (200)
PUT	string emailDomainText	/EmailDomain	Het <i>nieuwe</i> default domein voor het genereren van emails	OK (200)
GET		/MentorXFollower/followers/unassigned	De trajectvolgers die nog geen trajectbegeleider hebben	OK (200)
GET	int Id	/MentorXFollower/{MentorId}	Trajectvolgers van de mentor	OK (200)
GET	int followerId	/MentorXFollower/mentors/{followerId}	Trajectvolgers van een trajectvolger	OK (200)
POST	int mentorId int followerId	/MentorXFollower	De MentorXFollower link.	Created (201)
DELETE	int mentorId int followerId	/MentorXFollower/{mentorId}/{followerId}		No Content (204)
GET		/Module	Alle modules in de applicatie.	OK (200)
POST	CreateModuleDto dto	/Module	Een nieuwe module.	Created (201)
PUT	UpdateModuleDto dto	/Module	De geüpdate module.	Created (201)
PATCH	SolveModuleDto dto	/Module	De opgeloste module	OK (200)
GET	int moduleId	/Module/course-follower/id/{moduleId}	De module met de id voor een bepaalde trajectvolger	OK (200)

GET	int moduleId	/Module/edit/id/{moduleId}	De module met de gegeven id voor te bewerken	OK (200)
GET	int userId	/Module/course-follower/user/{userId}	De modules voor een trajectvolger met de gegeven id	OK (200)
GET	int userId	/Module/user/{userId}	De modules voor een trajectvolger met de gegeven id met detailinformatie	OK (200)
POST	AssignAndDeassign-ModuleDto dto	/Module/assign	De gelinkte module	Created (201)
DELETE	int moduleId	/Module/{moduleId}		No Content (204)
DELETE	int mentorId int followerId int moduleId	/Module/de-assign/{mentorId}/{followerId}/{moduleId}		No Content (204)
GET	int id	/Module/user-progress/{id}	De niet voltooide, en voltooide modules met hun antwoorden	OK (200)
GET		/Module/my-progress	De niet voltooide, en voltooide modules met de gebruiker zijn antwoorden	OK (200)
PATCH	PatchQuestionDto dto	/Questions	De aangepaste vraag	Created (201)
GET	string auth0Id	/User/login/{auth0Id}	De JWT token voor het gebruik op de applicatie	OK (200)
POST	CreateUserWith-AdminDto user	/User/register		OK (200)
GET		/User	Alle gebruikers in de applicatie	OK (200)
POST	CreateUserWith-AdminDto user	/User	De aangemaakte gebruiker	Created (201)

GET		/User/admin/role	Alle rollen in de applicatie	OK (200)
POST	CreateUserFor-MentorDto user	/User/mentor	De nieuwe gebruiker dat gelinkt is aan de mentor	Created (201)
PUT	UpdateUserWith-AdminDto dto	/User/admin/account	De geüpdate gebruiker	OK (200)
PUT	UpdateUserWith-CourseMentorDto dto	/User/mentor/account	De geüpdate gebruiker	OK (200)
PATCH	PatchUserByIdDto dto	/User/account/partial	De geüpdate gebruiker	OK (200)
PATCH	int id	/User/block/{id}	De geblokkeerde gebruiker	OK (200)
DELETE	int id	/User/{id}		No Content (204)

## 6. Externe systeeminterfaces

Hieronder vinden we een C4-model terug dat de communicatie uitlegt binnen ons systeem. Zowel de communicatie van ons eigen systeem is hierin uitgelegd alsook de communicatie met de Externe API Auth0.



Zoals we op het bovenstaand C4-model kunnen zien bestaan de externe systemen voor dit project enkel uit de Externe Auth0 API. We maken gebruik van de **Auth0 Management API** door middel van een package, genaamd Auth0.ManagementApi. Daarmee kunnen we verschillende functies uitvoeren op ons Auth0 project. *Under the hood* gebruikt dit package wel de Auth0 Management API en **HTTPS** calls om data op te halen vanuit Auth0 zelf.

Om gemakkelijk data op te halen met deze package maakten we enkele util klassen aan, waarmee we access tokens kunnen genereren, het laatste gegenereerde access token en de ManagementApiClient kunnen opvragen. Dit doen we zodat we niet steeds een nieuwe client moeten aanmaken om dan weer aan te roepen; we behandelen de ManagementApiClient dus als een Singleton. De access tokens genereren we aan de hand van onze Auth0 project secret. Bij het maken van een request via de Auth0 ManagementApiClient zal het gegenereerde access token meegegeven moeten worden en krijgen we een **TokenResponse** terug.

Deze TokenResponse wordt via de JsonSerializer gemapt naar volgende klasse:

```
public class TokenResponse
{
    [JsonPropertyName("access_token")] public string? AccessToken { get; set; }

    [JsonPropertyName("expires_in")] public int? ExpiresIn { get; set; }

    [JsonPropertyName("token_type")] public string? TokenType { get; set; }
}
```

Dit doen we zodat we de TokenResponse gemakkelijk kunnen uitlezen.

Het laatste gegenereerde access token zal altijd opgeslagen worden als een property van de Auth0TokenUtil klasse (in memory) en kan zo overal waar nodig opgevraagd worden (public). Wanneer het huidige access token verlopen is, zal er zo automatisch een nieuw access token aangemaakt kunnen worden.

## 7. Datamigratie

Er zal geen datamigratie plaatsvinden, omdat we een volledig nieuw systeem bouwen voor GATAM.

Dit betekent dat we geen bestaande data hoeven over te zetten naar het nieuwe systeem. Het nieuwe systeem wordt vanaf de basis opgebouwd, zonder afhankelijkheid van oude gegevens. Daardoor is er geen migratieproces.

## 8. Security en autorisatie rollen

Om ervoor te zorgen dat onze applicatie zo veilig mogelijk is, implementeerden we volgende beveiligingsmaatregelen op verschillende niveaus van onze applicatie.

Allereerst maken we gebruik van **environment variabelen** om onze geheime keys en configuratiestings in op te slaan. Deze worden in een .env bestand gezet, dat in de .gitignore staat, zodat het niet gelekt wordt naar GitLab toe. Wanneer onze applicatie aan staat, kan het die environment variabelen ophalen om dan te gebruiken in de code die uitgevoerd moet worden.

Daarnaast gebruiken we **Auth0** om in te loggen op onze applicatie (authenticatie). De implementatie van Auth0 in onze applicatie zorgt voor een beveiligde authenticatie-flow. Auth0 encrypteert de opgeslagen wachtwoorden met het gekende encryptiesysteem **bcrypt**. Ook stelden we een minimale wachtwoordsterkte in: wachtwoorden moeten namelijk bestaan uit minimaal 8 karakters, moeten minstens één hoofdletter, één kleine letter en één nummer bevatten, en moeten een speciaal karakter hebben.

Om gebruik te maken van Auth0 is het natuurlijk ook noodzakelijk dat onze applicatie door middel van een access token toegang krijgt tot ons Auth0 project door middel van de Auth0 Management API Client (Nuget package). We genereren een **access token** om dit te kunnen realiseren (en verversen dit access token telkens dit verlopen is). Deze access tokens worden niet opgeslagen, maar leven in memory.

Om ervoor te zorgen dat bepaalde gebruikers enkel toegang hebben tot de relevante pagina's en functionaliteiten maken we gebruik van **rolgebaseerde autorisatie**. Na het succesvol inloggen via Auth0 zal er een **JWT** opgeslagen worden in de localstorage van de ingelogde gebruiker, zodat men deze JWT niet rechtstreeks aanpasbaar is, maar door onze applicatie opgeslagen moet worden. In deze JWT vinden we zaken terug zoals rollen, gebruikers ID en e-mail.

In onze frontend maken we gebruik van deze rolgebaseerde autorisatie om zo toegang te verlenen tot bepaalde pagina's en componenten die relevant zijn voor de rollen van de gebruiker. We stelden hier een **SecurityConfig voor in met Access Policies**, die we via een attribute (Authorize) in de pagina zelf kunnen aanroepen om zo per pagina de juiste rollen toegang te geven tot de pagina.

Deze beveiligingsstrategie pasten we ook toe in de backend. We stelden weer een **SecurityConfig in met Access Policies** voor rolgebaseerde toegang tot de API. In de frontend implementeerde we een **JwtAuthenticationHandler** die ervoor zorgt dat wanneer de API wordt aangeroepen vanuit de frontend, de JWT van de ingelogde gebruiker in de Authorization header als een Bearer token wordt meegestuurd. In de backend wordt deze request dan opgevangen door de **CustomJwtMiddleware**, waar deze JWT uit de request wordt gehaald om te checken welke gebruiker de request heeft gestuurd en welke rollen deze juist heeft. Op deze manier kunnen we deze rol dan vergelijken met de rol die toegang heeft voor het aangesproken API endpoint door te kijken of deze rol in de Access Policy zit van dit endpoint. Als dit het geval is, zal de API een goede

response (200 Ok) terugsturen, indien dit niet het geval is, wordt er een reponsestatus 403 Forbidden (of 401 Unauthorized, indien er geen JWT aanwezig was in de request) teruggestuurd.

Verder maken we ook gebruik van **Docker** voor de containerisatie van onze applicatie. We gebruiken waar mogelijk enkel officiële images (zoals voor het opzetten van onze MS SQL Server database). Docker Compose, onze orchestrator, zorgt voor netwerkisolatie tussen containers en vereist zorgvuldig beheer om onbedoelde beveiligingslekken te voorkomen.

Hoewel geen enkel systeem absoluut onkwetsbaar is, verminderen deze maatregelen het risico op inbreuken en ongeoorloofde systeemaanvallen aanzienlijk.

Vervolgens hebben we ook verschillende rollen in onze applicatie:

- Trajectvolger: Dit omvat de basisgebruiker onze applicatie en heeft de minste toegang.
- Trajectbegeleider: Een speciale rol voor een gebruiker die trajectvolgers gekoppeld kan hebben aan zichzelf, die deze kan opvolgen en deze modules kan opgeven.
- Content Maker: Gebruikers die deze rol hebben, staan in voor het aanmaken en bewerken van alle modules en vragen.
- Administrator: Een administrator kan alles in de applicatie en heeft toegang tot elke pagina / component daarvan.

De matrix hieronder legt uit welke rol permissies heeft tot bepaalde componenten in de applicatie.

Rol	Trajectvolger	Trajectbegeleider	Content Maker	Administrator
Vragen oplossen	✓	✓	✓	✓
Vragen aanmaken	✗	✗	✓	✓
Vragen beheren	✗	✗	✓	✓
Modules aanmaken	✗	✗	✓	✓
Modules bewerken/verwijderen	✗	✗	✓	✓
Module bekijken	✓	✗	✓	✓
Modules toewijzen	✗	✗	✓	✓
Trajectvolgers (ont)koppelen	✗	✓	✗	✓
Profiel bekijken	✓	✓	✓	✓
Profiel aanpassen	✗	✓	✗	✓
Gebruikerslijst bekijken	✗	✗	✗	✓
Gekoppelde trajectvolgerslijst bekijken	✗	✓	✗	✓
Gebruikers beheren	✗	✗	✗	✓
Gebruiker aanmaken	✗	✗	✗	✓
Basis e-mail domein aanpassen voor gebruikerscreatie	✗	✗	✗	✓



## 9. Unieke Structuren

In volgende sectie beschrijven we unieke structuren in het behandelen van problemen waar rekening mee gehouden moet worden.

### A. Exception Handling

De exception handling in de applicatie voor in de frontend te laten zien wat er lukt of verkeerd gaat wordt als volgt geregeld.

Backend:

- 1) Exception wordt gegoooid.

Vb:

```
throw new ModuleForUserNotFound(request.Dto.FollowerId, request.Dto.ModuleId);

1 usage  https://gitlab.apstudent.be/jan.cuyppers  More...
public class ModuleForUserNotFound(int userId, int moduleId)
    : NotFoundException($"Module Id: {moduleId} not found in user Id: {userId}.");
```

- 2) Exception wordt in de middleware onderschept.

```
https://gitlab.apstudent.be/jan.cuyppers +1
public async Task Invoke(HttpContext context)
{
    try
    {
        await next(context);
    }
    catch (BaseException ex)
    {
        var response = new ErrorResponseInfo
        {
            StatusCode = (int)ex.StatusCode,
            // Get message from Messages.json file with all the Dutch definitions
            Message = ErrorMessages.GetMessage(ex.GetType()),
            ErrorType = GetErrorTypeForException(ex)
        };

        context.Response.StatusCode = response.StatusCode;
        context.Response.ContentType = "application/json";
        await context.Response.WriteAsync(JsonSerializer.Serialize(response));
    }
    catch (Exception ex)
    {
        var response = new ErrorResponseInfo
        {
            Message = ex.Message,
        };
    }
}
```

**Defined exceptions**

**Unexpected exceptions**

- 3) Waarbij deze zal zien of de exception gekend is. Indien gekend haalt deze de correcte Nederlandse zin op uit een JSON-file, en stuurt deze uit naar de frontend. Intern wordt de exception message met handige debug informatie nog als normaal getoond.

Vb:

Ophalen messages:

```
// Class to fetch the ErrorMessage.json file for Dutch messages
1 usage 2 https://gitlab.apstudent.be/jan.cuyper
public static class ErrorMessage
{
    private static readonly Dictionary<string, string> Messages;

    2 https://gitlab.apstudent.be/jan.cuyper
    static ErrorMessage()
    {
        var json :string = File.ReadAllText(path: "../AP.GATAM.Application/Resources/ErrorMessage.json");
        Messages = JsonSerializer.Deserialize<Dictionary<string, string>>(json) ?? new Dictionary<string, string>();
    }

    1 usage 2 https://gitlab.apstudent.be/jan.cuyper
    public static string GetMessage(Type exceptionType)
    {
        return Messages.GetValueOrDefault(exceptionType.Name, "Er is een onverwachte fout opgetreden.");
    }
}
```

Vb message:

```
"ModuleForUserNotFound": "Het loskoppelen van de module is mislukt, de link van de module is niet gevonden.",
```

(Zie codeblock stap 2 voor werking)

Frontend:

- 1) In de frontend erven alle services over van een ServiceApiBlueprint Class. Deze klasse definieert GET, POST, PUT, PATCH en DELETE methoden. Waarbij hier de exception handling gebeurt.

Default Method:

```
5 usages https://gitlab.apstudent.be/jan.cuyper
private async Task<T> SendAsync<T>(Func<Task<HttpResponseMessage>> httpCall, Action? onSuccess = null)
{
    try
    {
        var response = await httpCall();

        if (!response.IsSuccessStatusCode)
        {
            // If response is unsuccessful, handle the error and throw a custom exception
            await HandleErrorResponse(response);
        }

        else
        {
            // Invoke needed to call a Toast message in service
            onSuccess?.Invoke();
        }

        // Json body empty -> otherwise error because trying to return T
        if (response.StatusCode == HttpStatusCode.NoContent)
        {
            return default;
        }

        // Return the successful response content
        return await response.Content.ReadFromJsonAsync<T>();
    }
}
```

**Makes Toastmessage with backend message**

**Service can define succes message**

Default GET, POST, PUT, PATCH en DELETE methoden:

```
10 usages https://gitlab.apstudent.be/jan.cuyper
protected Task<T> BlueprintGetAsync<T>(string url) =>
    SendAsync<T>(httpCall: () => Http.GetAsync(url));

4 usages https://gitlab.apstudent.be/jan.cuyper
protected Task BlueprintPostAsync<T>(string url, T content, Action? onSuccess = null) =>
    SendAsync<object?>(httpCall: () => Http.PostAsJsonAsync(url, content), onSuccess);

4 usages https://gitlab.apstudent.be/jan.cuyper
protected Task BlueprintPutAsync<T>(string url, T content, Action? onSuccess = null) =>
    SendAsync<object?>(httpCall: () => Http.PutAsJsonAsync(url, content), onSuccess);

3 usages https://gitlab.apstudent.be/jan.cuyper
protected Task BlueprintPatchAsync<T>(string url, T content, Action? onSuccess = null) =>
    SendAsync<object?>(httpCall: () => Http.PatchAsJsonAsync(url, content), onSuccess);

4 usages https://gitlab.apstudent.be/jan.cuyper
protected Task BlueprintDeleteAsync(string url, Action? onSuccess = null) =>
    SendAsync<object?>(httpCall: () => Http.DeleteAsync(url), onSuccess);
```

- 2) Bij een succesvolle operatie kan een onSuccess actie aangeroepen worden. Hierin kan een toastmessage worden invoked in de frontend zodat de gebruiker weet dat de operatie gelukt is.

Bv:

```
0+1 usages https://gitlab.apstudent.be/jan.cuypers
public async Task CreateModule(CreateModuleDto dto)
{
    await BlueprintPostAsync(BaseUrl, dto,
        onSuccess: () =>
            ToastService.ShowSuccess(
                message: "De Module werd succesvol aangemaakt. \nJe wordt doorgestuurd naar de modulelijst..."));
}
```

- 3) Bij een niet succesvolle operatie zal de service de Nederlandse bericht van de backend opvangen en tonen aan de gebruiker. Het bericht is zo opgesteld dat het niet sensitive data bevat, en het duidelijk blijft voor de gebruiker.

```
1 usage https://gitlab.apstudent.be/jan.cuypers
private async Task HandleErrorResponse(HttpResponseMessage response)
{
    var errorResponse = await response.Content.ReadFromJsonAsync<ErrorResponseInfo>();

    // If ErrorType is defined (Needed to display validation errors in frontend)
    if (errorResponse != null && errorResponse.ErrorType.HasValue)
    {
        throw new CustomApiException(
            errorResponse.StatusCode,
            errorResponse.ErrorType.Value,
            errorResponse.Message ?? "Er is een fout opgetreden."
        );
    }

    // ErrorType not defined -> Nothing special just show text
    throw new CustomApiException(
        (int)response.StatusCode,
        CallToApiResult.OtherError,
        $"Er is een fout opgetreden. Statuscode: {response.StatusCode}"
    );
}
```

## 10. Documentatie

In de tabel wordt uitgelegd welke documentatie er voor de software wordt gemaakt, en voor wie die bedoeld is:

- **Code documentatie:** Dit is voor programmeurs en ontwikkelaars. Het legt uit hoe de software werkt, zoals de functies en instellingen. Via Swagger kunnen ze de API testen.
- **Installatiehandleiding:** Dit is voor systeembeheerders en IT-professionals. Het bevat duidelijke stappen om de software te installeren en in te stellen, aangezien er geen automatische installatie is.
- **Gebruikershandleiding:** Dit is voor de eindgebruikers. Het geeft stapsgewijze uitleg over hoe ze de webapplicatie moeten gebruiken. Er komen vier verschillende handleidingen voor verschillende gebruikers: administrators, content makers, trajectbegeleiders en trajectvolgers.

Type Document	Beschrijving
Type: Code documentatie Doelpubliek: <ul style="list-style-type: none"><li>- Programmeurs</li><li>- Ontwikkelaars</li></ul>	Documentatie voor de API: <ul style="list-style-type: none"><li>- Swagger Doc zorgt voor een handig overzicht van alle API endpoints met hun parameters. Ook kunnen we via Swagger deze endpoints uittesten.</li></ul> <p>De code documentatie biedt een uitgebreid inzicht in de interne werking van de software. De voornaamste doelgroep bestaat uit programmeurs en ontwikkelaars die betrokken zijn bij onderhoud, uitbreiding, etc. Deze documentatie dient als een referentiebron voor het begrijpen van functies, klassen, methoden en parameters.</p>
Type: Installatiehandleiding Doelpubliek: <ul style="list-style-type: none"><li>- Systeembeheerders</li><li>- IT-professionals</li></ul>	Doordat we geen installatie aanbieden zal er een installatiehandleiding opgemaakt worden. Deze handleiding voorziet nauwkeurige instructies voor het succesvol installeren en configureren van de software.
Type: Gebruikershandleiding Doelpubliek: <ul style="list-style-type: none"><li>- Eindgebruikers</li></ul>	De voornaamste doelgroep van de gebruikershandleiding zijn de eindgebruikers. Deze handleiding is een stapsgewijze gids voor het efficiënt gebruik van de webapplicatie. Het legt de essentiële taken, functies en functionaliteiten op een begrijpelijke manier uit. <p>Er zullen vier gebruikershandleidingen worden opgemaakt voor de volgende doelgroepen:</p> <ul style="list-style-type: none"><li>• Administrators</li><li>• Content maker</li><li>• Trajectbegeleiders</li><li>• Trajectvolgers</li></ul>

## 11. Bronvermelding

- .NET. (2024, 03 20). Opgehaald van Wikipedia: <https://nl.wikipedia.org/wiki/.NET>
- auth0. (2024, 11 28). *NuGet Gallery | Auth0.ManagementApi 7.30.0*. Opgehaald van nuget: <https://www.nuget.org/packages/Auth0.ManagementApi>
- Batista, R. (2020, 02). *Create User - Encrypting or hashing the password*. Opgehaald van Auth0 Community: <https://community.auth0.com/t/create-user-encrypting-or-hashing-the-password/38629>
- Bradley-Schacht. (2011, 12 14). *Installing SharePoint 2010 and SQL Server 2012 RC0*. Opgehaald van SqlServerCentral: <https://www.sqlservercentral.com/blogs/installing-sharepoint-2010-and-sql-server-2012-rc0>
- GitLab. (2024). *GitLab*. Opgehaald van YouTube: <https://www.youtube.com/channel/UCnMGQ8QHMANVIsI3xJrihhg>
- Hasan, R. (2019, 08 09). *Fun with Docker - Part 1: An introduction...* Opgehaald van ccie.tv: <https://ccie.tv/fun-with-docker/>
- Mariën, S. (sd). presentatie Application Architecture. In *Application Architecture (niet publiek beschikbaar)*. cursus AP Hogeschool 24-25 Applied softwareproject.
- Okta, I. (2024). *Secure access for everyone. But not just anyone*. Opgehaald van Auth0: <https://auth0.com/>
- Schudde, D. (2020, 11 24). *Blazor, een inleiding*. Opgehaald van Vitas Blog: [https://blog.vitas.nl/blazor\\_een\\_inleiding.html](https://blog.vitas.nl/blazor_een_inleiding.html)
- Streng, S. (sd). *.Net C# — Clean Architecture & Dependency-Inversion-Principle*. Opgehaald van Medium: <https://blog.devgenius.io/net-c-clean-architecture-dependency-inversion-principle-d2d661c3f74d>
- Traefik. (2020, 07 15). Opgehaald van GitHub: <https://github.com/containous/traefik-library-image>
- What is ASP.NET? An Overview in 2025*. (2023, 06 07). Opgehaald van Flatirons: <https://flatirons.com/blog/what-is-aspnet-an-overview-2024/>