**DOCUMENTATIERAPPORT**

**ONDERDEEL VAN DE BACHELORPROEF**

\_\_\_\_\_\_\_\_\_\_\_\_\_

Een grondige analyse van API-opties voor efficiënte en

veilige toegang tot klantgegevens in externe applicaties

|  |  |
| --- | --- |
| Bachelor | Toegepaste Informatica |
| Keuzetraject | Software Engineer |
| Academiejaar | 2023 - 2024 |
| Student | Lukas Olivier |

# Melding van vertrouwelijkheid (enkel indien van toepassing)

*Indien dit documentatierapport vertrouwelijke informatie bevat, wordt de volgende tekst in verband met vertrouwelijkheid opgenomen. Indien niet, dan verwijder je deze pagina.*

Vertrouwelijk tot en met dd/mm/20yy

Belangrijk

Dit documentatierapport (dat onderdeel uitmaakt van de bachelorproef) bevat vertrouwelijke informatie en/of vertrouwelijke onderzoeksresultaten die toebehoren aan de Howest of aan derden. Dit rapport of enig onderdeel ervan mag op geen enkele wijze publiek gemaakt worden zonder de uitdrukkelijke schriftelijke voorafgaande toestemming vanwege Howest. Zo mag dit rapport onder geen voorwaarde door derden worden ingekeken of aan derden worden meegedeeld. Het is verboden om dit rapport te kopiëren of op eender welke manier te dupliceren. Indien de vertrouwelijke aard van dit rapport niet wordt gerespecteerd, kan dit onherstelbare schade veroorzaken aan Howest. Bovenstaande bepalingen zijn van kracht tot en met de embargodatum.

# Toelating tot bruikleen

De auteur(s) geeft (geven) de toelating dit documentatierapport (onderdeel van de bachelorproef )voor consultatie beschikbaar te stellen en delen van dit rapport te kopiëren voor persoonlijk gebruik. Elk ander gebruik valt onder de bepalingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting de bron uitdrukkelijk te vermelden bij het aanhalen van resultaten uit dit rapport.

8/03/2024

# Woord vooraf

In het woord vooraf staan de gebruikelijke dankbetuigingen. Alle personen worden bedankt die meegeholpen hebben met het eindwerk.

De personen die de belangrijkste bijdrage hebben geleverd, worden als eerste bedankt. Schrijf de naam, functie en titel van personen correct.

Vermeld onderaan je naam, plaats en datum (facultatief). Een handtekening past hier niet.

Doordat het woord vooraf sterk persoonlijk is, wordt het vaak in de ik-vorm geschreven.

# Inhoudsopgave

#### Woord vooraf

[Onderzoeksfase 6](#_Toc160780421)

[Demo Project: Student Management API 6](#_Toc160780422)

[Node.js - Express 7](#_Toc160780423)

[Inleiding 7](#_Toc160780424)

[C# - ASP.NET Core 8](#_Toc160780425)

[Inleiding 8](#_Toc160780426)

[Python – Django 9](#_Toc160780427)

[Inleiding 9](#_Toc160780428)

[Java - Spring Boot 10](#_Toc160780429)

[Inleiding 10](#_Toc160780430)

[Node.js - Apollo Server (GraphQL) 11](#_Toc160780431)

[Inleiding 11](#_Toc160780432)

# Inleiding

## Algemeen

Algemeen over het domein, de algemene scope. Ruime probleemstelling. Beschrijving van het primaire doelpubliek van deze bachelorproef.

Een beknopte alinea waarin je zegt dat deze bachelorproef tot stand gekomen is tijdens het stagelopen in bedrijf/instelling X. Zodat de lezer een beetje de context kent waar je dit onderzoek hebt uitgevoerd. De focus mag niet op het aparte stagewerk liggen, enkel ter verduidelijking van het onderzoek voor deze bachelorproef.

## Probleemstelling

Specifieke limiterende scope. De specifieke probleemstelling die je behandelt.

## Onderzoeksvraag

Op basis van bovenstaande probleemstelling, de concrete onderzoeksvraag letterlijk formuleren. Verduidelijken dat je hierop een antwoord zult zoeken doorheen de bachelorproef.

## Experiment

Welke methoden gebruik je om antwoord te geven op je onderzoeksvraag? Bespreek hier **kort** je onderzoeksopzet (soort onderzoek, dataverzameling, dataomschrijving en analysemethode). Dit deel wordt in detail in het volgende hoofdstuk besproken.

# Onderzoeksfase

Om een grondige vergelijking tussen verschillende API's mogelijk te maken, is het essentieel om een diepgaand onderzoek uit te voeren naar de diverse mogelijkheden. Met als doel een breed scala aan perspectieven te verkrijgen, streef ik ernaar een vergelijkbare demo-API te repliceren in verschillende programmeertalen. Voor elke programmeertaal zal ik op zoek gaan naar het meest populaire en/of effectieve framework om een API te ontwikkelen.

## Demo Project: Student Management API

Het demo-project omvat het beheer van studenteninformatie met CRUD-operaties, waaronder het toevoegen van nieuwe studenten, het ophalen van de lijst met studenten, het bijwerken van studenteninformatie en het verwijderen van studenten die de school hebben verlaten. De API wordt ontwikkeld in verschillende programmeertalen om een vergelijkende evaluatie mogelijk te maken.

## Node.js - Express

### Inleiding

De keuze voor Node.js in combinatie met het Express-framework is gebaseerd op de brede acceptatie van Node.js en de bewezen effectiviteit en eenvoud van Express voor het ontwikkelen van webapplicaties en API's. Node.js biedt een asynchrone, niet-blokkerende runtime, terwijl Express een lichtgewicht en flexibel webframework is dat goed aansluit op de Node.js-omgeving.

De combinatie van Node.js en Express biedt een moderne en efficiënte benadering van API-ontwikkeling, met een eenvoudige syntax en goede prestaties. Deze keuze is relevant voor implementatie in verschillende programmeertalen binnen het kader van dit onderzoek, gezien de populariteit en flexibiliteit van Node.js met het Express-framework.

## C# - ASP.NET Core

### Inleiding

De keuze voor C# en ASP.NET Core wordt ingegeven door de gevestigde positie van C# als een veelgebruikte programmeertaal en ASP.NET Core als een krachtig en veelzijdig framework voor web-API's. C# biedt sterke typen, een geavanceerd ontwerppatroon, en uitgebreide ondersteuning in de ontwikkelgemeenschap. ASP.NET Core, met zijn modulaire en cross-platform karakter, is ideaal voor het bouwen van schaalbare en moderne API's.

Deze combinatie van C# en ASP.NET Core biedt een robuuste basis voor API-ontwikkeling, met uitgebreide functionaliteiten en brede acceptatie in de industrie. Dit maakt het relevant voor de implementatie in verschillende programmeertalen binnen het kader van dit onderzoek.

## Python – Django

### Inleiding

De keuze voor Python in combinatie met het Django-framework is gebaseerd op de wijdverspreide adoptie van Python als programmeertaal en de reputatie van Django als een krachtig webframework. Python biedt een heldere syntax, uitgebreide bibliotheken en een actieve gemeenschap, terwijl Django zich onderscheidt door zijn elegantie en gebruiksgemak bij het ontwikkelen van webtoepassingen.

Deze combinatie van Python en Django vormt een solide basis voor de ontwikkeling van API's, met aanzienlijke flexibiliteit en efficiëntie. De keuze voor dit duo is relevant voor de implementatie in verschillende programmeertalen binnen het kader van dit onderzoek.

## Java - Spring Boot

### Inleiding

De keuze voor Java in combinatie met Spring Boot is gestoeld op de brede acceptatie van Java in de softwareontwikkelingsindustrie en de robuuste mogelijkheden van het Spring Boot-framework. Java staat bekend om zijn platformonafhankelijkheid, sterke typen en uitgebreide ecosysteem. Spring Boot, als een opvatting van het Spring-framework, biedt een vereenvoudigde manier om snel en efficiënt op Java gebaseerde API's te ontwikkelen.

De combinatie van Java en Spring Boot levert een solide basis voor de ontwikkeling van schaalbare en betrouwbare API's. Deze keuze is relevant voor implementatie in verschillende programmeertalen binnen het kader van dit onderzoek, gezien de veelzijdigheid en populariteit van Java en de efficiëntie van het Spring Boot-framework.

## ****Node.js -**** Apollo Server ****(****GraphQL****)****

### Inleiding

De keuze voor Node.js in combinatie met Apollo Server, dat is geoptimaliseerd voor GraphQL, is gebaseerd op de wijdverspreide adoptie van Node.js en de effectiviteit van Apollo Server voor het implementeren van GraphQL-API's. Node.js biedt een asynchrone, niet-blokkerende runtime en heeft een grote ontwikkelaarsgemeenschap. Apollo Server is een gespecialiseerd framework dat gericht is op het bouwen van efficiënte en flexibele GraphQL-API's.

Deze combinatie van Node.js en Apollo Server (GraphQL) biedt een moderne en krachtige benadering van API-ontwikkeling. Het gebruik van GraphQL als querytaal biedt flexibiliteit voor client-applicaties om specifieke gegevens op te vragen, waardoor het relevant is voor implementatie in verschillende programmeertalen binnen het kader van dit onderzoek.