**PLAN VAN AANPAK**

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

|  |
| --- |
| **Het Selecteren van een Optimale API voor Klantgegevens** |
| 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 |

**Inhoudsopgave**

[1. Onderzoeksfase 3](#_Toc158986034)

[1.1. C# - ASP.NET Core 3](#_Toc158986035)

[1.2. Python – Django 3](#_Toc158986036)

[1.3. Java - Spring Boot 3](#_Toc158986037)

[1.4. Node.js - Apollo Server (GraphQL) 3](#_Toc158986038)

[1.5. Deno – Oak 4](#_Toc158986039)

[2. Prompts: 4](#_Toc158986040)

[1.6. Google Gemini 4](#_Toc158986041)

# 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.

## C# - ASP.NET Core

### Inleiding

Voor mijn project koos ik voor C# in combinatie met ASP.NET Core vanwege de krachtige en objectgeoriënteerde aard van C#. Het biedt sterke typen, wat bijdraagt aan een robuuste codebase en gemakkelijke foutdetectie. ASP.NET Core biedt een uitgebreid framework voor webontwikkeling, met ingebouwde beveiligingsfuncties en goede prestaties. Bovendien integreert het naadloos met Visual Studio, waardoor het ontwikkelingsproces efficiënter wordt en het gemakkelijk is om te werken met andere Microsoft-producten.

## Python – Django

Voor mijn webapplicatie koos ik voor Python met het Django-framework vanwege de snelle ontwikkeltijd en de eenvoudige syntax van Python. Django, als een high-level webframework, biedt ingebouwde beveiliging, administratieve tools en een krachtig ORM-systeem. Het volgt het "Don't Repeat Yourself" (DRY) principe, wat resulteert in compacte en onderhoudsvriendelijke code. De grote Django-community en uitgebreide documentatie maken het eenvoudig om problemen op te lossen en nieuwe functies toe te voegen.

## Java - Spring Boot

Voor mijn backend-ontwikkeling koos ik voor Java met het Spring Boot-framework vanwege de uitgebreide mogelijkheden voor het bouwen van robuuste en schaalbare applicaties. Java biedt platformonafhankelijkheid en uitstekende prestaties, terwijl Spring Boot het ontwikkelingsproces versnelt door conventies en ingebouwde functionaliteiten. Het ondersteunt ook microservices-architecturen, waardoor het gemakkelijk is om componenten schaalbaar en onderhoudbaar te maken.

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

Voor mijn realtime webapplicatie koos ik voor Node.js vanwege de asynchrone en niet-blokkerende aard, wat resulteert in snelle en schaalbare applicaties. In combinatie met Apollo Server, dat GraphQL ondersteunt, kan ik efficiënter gegevens opvragen en de frontend-naar-backend communicatie optimaliseren. Node.js heeft ook een grote en actieve gemeenschap, waardoor het gemakkelijk is om hulp te vinden en bibliotheken te integreren.

## Deno – Oak

Voor mijn nieuwe project koos ik voor Deno als runtime-omgeving, vanwege de verbeteringen ten opzichte van Node.js, zoals ingebouwde TypeScript-ondersteuning en een verbeterde beveiligingsarchitectuur. Oak, als een lichtgewicht webframework voor Deno, biedt een eenvoudige en modulaire manier om webapplicaties te bouwen. Deno's module-systeem en het vermijden van npm in favor van URL's maken het gemakkelijk om afhankelijkheden te beheren en het ontwikkelproces te stroomlijnen.

Bovenkant formulier

# Prompts:

## Google Gemini

Help me van start door enkele technologieën voor te stellen waarmee ik de API kan bouwen. Hun data staat op een SQL server database. <omschrijving van de onderzoeksvraag>