DOCUMENTATIERAPPORT
onderdeel van de BACHELORPROEF

Analyse van REST-API technologieën

Voor efficiënte en veilige toegang tot klantgegevens in externe applicaties

Bachelor	Toegepaste Informatica
Keuzetraject Afstudeerrichting	Software Engineer
Academiejaar	2023 - 2024
Student	Lukas Olivier
Interne begeleider	Dieter Mourisse (Howest)
Externe promotor	Stefaan Missiean

howest.be

Toelating tot bruikleen

De auteur(s) geeft (geven) de toelating dit documentatierapport (onderdeel van de bachelor proef) 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.

3/06/2024

Woord vooraf

Met genoegen presenteer ik dit documentatierapport als onderdeel van mijn bachelorproef binnen het keuzetraject Software Engineer van de opleiding Toegepaste Informatica.

Ik richtte me op het analyseren van REST-API-technologieën, met als specifiek doel een optimale oplossing te vinden voor efficiënte en veilige toegang tot klantgegevens in externe applicaties.

Ik wil mijn oprechte dank betuigen aan alle mensen die hebben bijgedragen aan dit onderzoek, met in het bijzonder: dhr. Stefaan Missiaen, algemeen directeur van het bedrijf Integreat. Hij heeft mij niet alleen voorzien van de idee/onderzoeksvraag voor deze bachelorproef, maar heeft mij ook geholpen om behoeften van Integreat goed te begrijpen, wat het mogelijk maakte om mijn experiment duidelijk te definiëren. Daarnaast dhr. Dieter Mourisse, mijn interne begeleider van Howest, bedanken voor de begeleiding tijdens het uitwerken van deze bachelorproef. Dankzij hun begeleiding en expertise is dit onderzoek mogelijk gemaakt.

Dit onderzoek vormt een brug tussen theoretische inzichten en praktische toepassingen in de wereld van software engineering. Ik hoop dat de bevindingen en aanbevelingen in dit rapport zullen bijdragen aan een beter begrip en effectief gebruik van REST-API-technologieën in uiteenlopende softwareontwikkelingsprojecten.

Lukas Olivier

Izegem – 19/04/2024

Inhoudsopgave

1 Inleiding

1.1 Algemeen

Deze bachelorproef, getiteld “Een grondige analyse van REST-API-technologieën voor efficiente en veilige toegang tot klantgegevens in externe applicaties,” is ontstaan tijdens mijn stage bij Integreat, een ERP software bedrijf uit Waregem met primaire focus op de educatieve sector. Het onderzoek richt zich op het selecteren van een optimale REST-API-technologie voor klantgegevens integratie, een essentieel aspect in de huidige digitale infrastructuur. Het primaire doelpubliek van deze proef zijn professionals en organisaties die streven naar een efficiënte en veilige manier om klantgegevens te beheren en te integreren in externe applicaties. De stagecontext bood waardevolle praktijkervaring. De scope van deze proef gaat evenwel ruimer en focust op de algemene toepasbaarheid van de bevindingen voor het brede domein van software engineering.

1.2 Probleemstelling

Dit onderzoek richt zich op het identificeren van de meest passende REST-API voor Integreat. Tot op heden kunnen klanten deze informatie enkel bekijken via klassieke import- en exportfunctionaliteiten.

Dit probleem werd voorgelegd door bedrijf Integreat, vertegenwoordigd door de algemeen directeur, dhr. Stefaan Missiaen. De aanleiding voor dit onderzoek is de nood aan een efficiente- en veilige manier voor klanten om gegevens op te halen.

Er zijn verschillende REST-API-opties beschikbaar op de markt, maar geen enkele is tot nu toe geïmplementeerd. Dit probleem moet aangepakt worden om te voorkomen dat klanten problemen ondervinden bij het verkrijgen van toegang tot hun gegevens, wat kan leiden tot vertragingen en inefficiënties in hun softwareoplossingen.

Het finale doel is om een efficiënte en veilige methode te vinden waarmee klanten vlot gegevens kunnen verkrijgen.

Het is daarom van essentieel belang om een geschikte REST-API technologie te vinden die aan al deze eisen voldoet en waarmee een vlotte overgang van import-export functionaliteiten naar een REST-API mogelijk is.

1.3 Onderzoeksvraag

De hoofdvraag die deze bachelorproef wil beantwoorden is: “Welke REST-API is het meest geschikt voor de toepassing van Integreat om gegevens efficiënt en veilig op te vragen?”

Om antwoord te kunnen geven op de hoofdvraag, zijn de volgende deelvragen opgesteld:

1. Wat is een API?
2. Welke REST-API-technologieën zijn momenteel populair op de markt?
3. Hoe presteren deze REST-API-opties op het gebied van prestaties, productiviteit, veiligheid en ondersteuning?
4. Welke REST-API-optie biedt de beste balans tussen deze factoren en is het meest geschikt voor Integreat?

Deze deelvragen zullen helpen bij het grondig onderzoeken van het probleem en het vinden van de meest geschikte REST-API. Ze zullen ook zorgen voor een systematische aanpak van het onderzoek, wat zal bijdragen aan de kwaliteit en relevantie van de resultaten.

1.4 Wat is een API?

Voor aanvang van het onderzoek is het essentieel om te begrijpen wat een API is en doet.

1.4.1 API

API staat voor Application Programming Interface. Het is een set van definities en protocollen die de communicatie tussen twee softwareapplicaties mogelijk maakt. API's fungeren als tussenpersonen, waardoor applicaties gegevens kunnen uitwisselen en taken kunnen uitvoeren zonder de interne details van elkaar te kennen.

1.4.2 Web-API

Web-API's zijn een specifiek type API dat is ontworpen voor gebruik op het web. Ze gebruiken vaak HTTP als protocol voor het verzenden en ontvangen van gegevens en zijn toegankelijk via URL's. Web-API's stellen ontwikkelaars in staat om webapplicaties te bouwen die met elkaar kunnen communiceren en gegevens kunnen delen met externe bronnen.

1.4.3 RESTful-API

RESTful-API's (Representational State Transfer) vormen een subset van web-API's die een reeks ontwerpprincipes volgen om een consistente en efficiënte communicatie te waarborgen. Ze maken gebruik van standaard HTTP-methoden (GET, POST, PUT, DELETE) en -statuscodes om acties op resources te definiëren. Daarnaast zijn er enkele belangrijke ontwerpprincipes die de basis vormen van RESTful-API's:

1. **Statelessness**: RESTful-API's zijn stateless, wat betekent dat de server geen informatie over de clients bijhoudt. Elke aanvraag van een client bevat alle benodigde informatie om begrepen en verwerkt te worden.

2. **Uniform Interface**: De interface tussen client en server moet uniform zijn, wat betekent dat de communicatie tussen verschillende componenten van het systeem zo eenvoudig en consistent mogelijk moet zijn. Dit omvat het gebruik van goed gedefinieerde resources, URI's voor resource-identificatie, gestandaardiseerde HTTP-methoden en het gebruik van media types voor representatie

3. **Cacheable**: RESTful-API's ondersteunen caching om de prestaties en schaalbaarheid te verbeteren. Servers moeten aangeven of een reactie cachebaar is of niet, en clients moeten in staat zijn om opgeslagen reacties te gebruiken om toekomstige aanvragen te vervullen, wat de efficiëntie van het systeem verbeterd.

4. **Layered System**: Een RESTful-systeem moet een hiërarchie van lagen hebben, waarbij elke laag een bepaalde functionaliteit biedt en beperkingen oplegt aan de componenten die erbovenop zijn gebouwd. Dit helpt bij het bevorderen van schaalbaarheid door componenten te isoleren en complexiteit te verminderen.

1.4.4 Belangrijke onderscheidingen

- **Niet elke API is een REST-API:** Er bestaan API's die lokaal communiceren tussen verschillende onderdelen van dezelfde applicatie of die gebruikmaken van andere protocollen dan HTTP.

- **Niet elke REST-API is een RESTful-API:** Sommige REST-API's gebruiken andere architectonische stijlen of bieden minder strikte richtlijnen voor communicatie.

1.4.5 De voordelen van RESTful-API's

- **Eenvoudig te gebruiken:** De standaard HTTP-methoden en -codes zijn universeel gekend en eenvoudig te begrijpen.

- **Lichtgewicht:** RESTful-API's verbruiken minder bandbreedte en resources dan andere API-stijlen.

- **Schaalbaar:** RESTful-API's zijn goed uit te breiden en kunnen eenvoudig worden aangepast aan nieuwe eisen.

- **Toekomstbestendig:** De REST-architectuur is al decennia lang in gebruik en heeft bewezen robuust en flexibel te zijn.

1.4.6 Conclusie

API's zijn essentiële tools voor het bouwen van moderne softwareapplicaties. REST-API's en RESTful-API's vormen een krachtige combinatie die efficiënte communicatie tussen applicaties op het web mogelijk maakt. Inzicht in de werking en voordelen van deze technologieën is van cruciaal belang voor ontwikkelaars die webapplicaties willen bouwen die naadloos met elkaar kunnen communiceren.

In dit document zal gefocust worden op RESTful Web-API's, die voor de beknoptheid verder zal afkort worden naar REST-API's. De werking van REST-API’s wordt uitgelegd, de voordelen die ze bieden en de verschillende frameworks die beschikbaar zijn voor hun ontwikkeling.

1.4.7 Experiment

Dit onderzoek combineert kwalitatief en kwantitatief onderzoek:

Kwalitatief onderzoek

Het kwalitatieve deel richt zich op de behoeften van scholen om gegevens uit te wisselen tussen verschillende systemen via een REST-API.

Kwantitatief onderzoek

Het kwantitatieve deel evalueert en vergelijkt REST-API technologieën op criteria zoals veiligheid en efficiëntie. Data worden verzameld via literatuur- en online onderzoek naar beschikbare REST-API technologieën, bestaande uit technische specificaties en prestatiegegevens. Analyse en vergelijking van de data zullen de meest geschikte REST-API identificeren, waarbij elke optie wordt geëvalueerd en vergeleken op basis van criteria.

2 Experiment

Om een grondige vergelijking tussen verschillende API's mogelijk te maken, is het essentieel om uitgebreid 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 het meest populaire en/of effectieve framework gebruiken om een API te ontwikkelen. Daarna documenteer ik telkens mijn bevindingen. U kan de volledige uitgewerkte projecten vinden in de bijlagen.

Aangezien scholen een aanzienlijke klantengroep zijn voor Integreat, zal ik als voorbeeld een Student Management API ontwikkelen.

2.1 Demo Project - Student Management API

De functionaliteiten van de API zijn vastgelegd in de Swagger OpenAPI-specificatie. Uitgebreide details over de beschikbare handelingen, inclusief authenticatie met JSON Web Tokens (JWT), zijn te vinden in de bijlagen. Hieronder volgt een samenvatting van enkele belangrijke handelingen.

2.1.1 Authenticatie

De REST-API maakt gebruik van bearer tokens voor authenticatie. Op basis van deze tokens wordt alleen informatie teruggegeven waar de gebruiker toegang toe heeft. Bijvoorbeeld, het opvragen van een lijst met studenten zal alleen studenten opleveren die aan de school van de gebruiker zijn gekoppeld.

2.1.2 Functionaliteiten

Hieronder volgt een lijst van endpoints die elke REST-API zal bevatten:

- **Lijst van scholen:** Hiermee kan de gebruiker de lijst van scholen opvragen die hij of zij bezit.

- **Lijst van studenten per school**: Hiermee kan de gebruiker een lijst van studenten opvragen die zijn ingeschreven bij een specifieke school**.**

- **Lijst van niet-ingeschreven studenten**: Dit endpoint maakt het mogelijk om een lijst op te vragen van studenten die niet zijn ingeschreven bij een school.

- **Inschrijven bij een school:** Hiermee kunnen studenten worden ingeschreven bij een specifieke school waarvan de gebruiker eigenaar is.

- **Uitschrijven bij een school:** Dit endpoint maakt het mogelijk voor de gebruiker om studenten uit te schrijven die zijn ingeschreven bij een school die aan hem/haar is gekoppeld.

- **Toevoegen van nieuwe studenten:** Gebruikers kunnen nieuwe studenten toevoegen met alle benodigde informatie.

- **Verwijderen van studenten:** Dit endpoint biedt de mogelijkheid om studenten te verwijderen uit het systeem.

Voor een gedetailleerde beschrijving van de API-handelingen wordt verwezen naar de Swagger OpenAPI-specificatie in de bijlagen.

2.2 Express.js

2.2.1 Waarom

Express.js is een robuust en flexibel framework voor het ontwikkelen van webapplicaties en API's binnen Node.js. Met een minimalistische aanpak biedt het een intuïtieve manier om routes te definiëren, middleware toe te passen en HTTP-verzoeken te verwerken.

2.2.2 Ondersteuning en Documentatie

Express.js wordt ondersteund door uitgebreide en duidelijke documentatie, inclusief tutorials, API-referenties en voorbeeldcode. Deze bronnen zijn waardevol voor ontwikkelaars om snel aan de slag te gaan en complexe problemen op te lossen. Bovendien is er een actieve community beschikbaar via forums en platforms zoals Stack Overflow, waardoor ontwikkelaars gemakkelijk hulp en ondersteuning kunnen vinden.

2.2.3 Veiligheid

Express.js biedt verschillende libraries om de beveiliging te waarborgen, zoals het hashen van wachtwoorden met Bcrypt, autorisatie met JSON Web Tokens (JWT) en validatie met JOI. Gevoelige informatie wordt apart gehouden in een .env-bestand, terwijl extra bescherming tegen aanvallen wordt geboden door middel van de Helmet library.

2.2.4 Prestatie

Express.js staat bekend om zijn hoge prestaties, zelfs onder zware belasting. Het framework is geoptimaliseerd om efficiënt te werken en kan snel reageren op HTTP-verzoeken, waardoor het ideaal is voor het bouwen van schaalbare API's die grote hoeveelheden verkeer kunnen verwerken.

2.2.5 Ontwikkelaarsproductiviteit

Dankzij de eenvoudige syntax en de volwassenheid van het framework kunnen ontwikkelaars snel en efficiënt werken. Ik kon het volledige REST-API demoproject een binnen één dag opzetten, inclusief de benodigde beveiligingsmechanismen, CRUD operaties en user management. Bovendien is er een overvloed aan informatie en voorbeeldcode beschikbaar op het internet, waardoor het relatief eenvoudig is om oplossingen te vinden voor specifieke problemen.

2.2.6 Conclusie

Express.js een uitstekende keuze voor het ontwikkelen van REST-API's. Het framework biedt een balans tussen snelheid, flexibiliteit, veiligheid en ondersteuning, waardoor het geschikt is voor een breed scala aan projecten. Met zijn bewezen prestaties en uitgebreide gemeenschap is Express.js een betrouwbare keuze voor zowel kleine als grote projecten.

2.3 Django

2.3.1 Waarom

Django, gebouwd op Python, is een krachtig framework dat een breed scala aan mogelijkheden biedt voor het ontwikkelen van webapplicaties en REST-API's. Hoewel het framework meer dan alleen een API-ondersteuning biedt en ook geschikt is voor front-end ontwikkeling, is het nog steeds een waardevolle optie - zelfs voor demo-doeleinden – dankzij zijn uitgebreide functionaliteit en solide documentatie.

2.3.2 Ondersteuning / documentatie

De documentatie van Django is modern en van hoge kwaliteit, waardoor ontwikkelaars snel aan de slag kunnen en complexe problemen kunnen oplossen. Bovendien is er een bloeiende community beschikbaar voor ondersteuning via forums. Dit maakt het gemakkelijk om hulp te vinden bij eventuele obstakels tijdens de ontwikkeling.

2.3.3 Ontwikkelaarsproductiviteit

Het opzetten en uitbreiden van een API met Django verloopt over het algemeen vlot en moeiteloos. De framework biedt een krachtige set tools en functionaliteiten die ontwikkelaars in staat stellen om snel aan de slag te gaan. Echter, ik ben minder enthousiast over de mappenstructuur van Django, gezien er redelijk veel functionaliteit geconcentreerd is in dezelfde bestanden, wat tot enige onduidelijkheid kan leiden tijdens het ontwikkelproces.

Daarnaast vereist het integreren van een bestaande database en tabellen wat extra configuratie en aanpassingen, wat mogelijk de ontwikkeltijd kan verlengen. Echter, wanneer je te maken hebt met een project waarvoor een nieuwe database en tabellen nodig zijn, verloopt dit proces zeer vlot met behulp van migrations.

2.3.4 Prestatie

Na mijn tests heb ik opgemerkt dat Django over het algemeen solide presteert, hoewel het niet zo schaalbaar lijkt te zijn als sommige van de andere opties die ik heb onderzocht. Django kan weliswaar snel en efficiënt reageren op HTTP-verzoeken, maar het lijkt minder geschikt te zijn voor het bouwen van REST-API's die grote hoeveelheden verkeer moeten verwerken.

2.3.5 Veiligheid

Django biedt ingebouwde beveiligingsmechanismen voor authenticatie, autorisatie en gegevensvalidatie. Dit omvat bescherming tegen veelvoorkomende aanvallen zoals XSS, CSRF en SQL-injectie. Echter, vanwege het gebrek aan veel third-party libraries voor bijvoorbeeld validatie, kan het zijn dat ontwikkelaars meer handmatig werk moeten verrichten om aanvullende beveiligingsmaatregelen te implementeren.

2.3.6 Conclusie

Django is een krachtig framework voor het ontwikkelen van webapplicaties en REST-API's, met solide documentatie en een brede community voor ondersteuning. Hoewel het opzetten van een API over het algemeen vlot verloopt, kan de mappenstructuur enige onduidelijkheid veroorzaken. Integratie met bestaande databases kan extra configuratie vereisen, hoewel het migratieproces soepel verloopt voor nieuwe databases. De prestaties van Django zijn solide en het biedt ingebouwde beveiligingsmechanismen, maar het gebrek aan third-party libraries kan ontwikkelaars dwingen om meer handmatig werk te verrichten voor aanvullende beveiligingsmaatregelen. Over het algemeen blijft Django een waardevolle keuze voor het ontwikkelen van zowel kleine als grote projecten.

2.4 ASP.NET Core (.NET 8)

2.4.1 Waarom

ASP.NET Core, ontwikkeld door Microsoft, is een robuust framework voor het bouwen van webapplicaties en REST-API's. Met zijn brede scala aan functionaliteiten en diepe integratie met het ecosysteem van Microsoft, biedt het een uitstekende optie voor ontwikkelaars die bekend zijn met de Microsoft-stack of op zoek zijn naar een krachtig en betrouwbaar framework.

2.4.2 Ondersteuning / documentatie

De documentatie voor ASP.NET Core is uitgebreid en van hoogwaardige kwaliteit, waardoor ontwikkelaars snel aan de slag kunnen en complexe problemen kunnen oplossen. Microsoft heeft een sterke aanwezigheid in de ontwikkelaarsgemeenschap, wat resulteert in een overvloed aan online bronnen, forums en tutorials voor ondersteuning.

2.4.3 Ontwikkelaarsproductiviteit

Het opzetten en uitbreiden van een API met ASP.NET Core verloopt over het algemeen soepel en efficiënt. Dankzij de integratie met Visual Studio en andere Microsoft-tools kunnen ontwikkelaars snel prototypes maken en functionaliteit toevoegen. De gestructureerde mappen en bestandsindeling zorgen voor een georganiseerde ontwikkelomgeving en bevorderen de samenwerking binnen teams.

2.4.4 Prestatie

ASP.NET Core staat bekend om zijn uitstekende prestaties en schaalbaarheid. Het framework is geoptimaliseerd voor het afhandelen van grote hoeveelheden verkeer en biedt diverse optimalisatietechnieken, zoals caching en asynchronous programming, voor een snelle respons op HTTP-verzoeken.

2.4.5 Veiligheid

Met ingebouwde beveiligingsmechanismen zoals authenticatie en autorisatie biedt ASP.NET Core een solide basis voor het beveiligen van API's tegen veelvoorkomende aanvallen. Bovendien is de integratie met Microsoft Identity naadloos, waardoor het implementeren van geavanceerde beveiligingsfuncties eenvoudig is. Daarnaast maakt het gebruik van de Fluent Validation NuGet-package objectvalidatie zeer eenvoudig, waardoor ontwikkelaars snel en effectief kunnen zorgen voor de juiste validatie van invoergegevens binnen hun API's.

2.4.6 Conclusie

ASP.NET Core is een krachtig framework voor het ontwikkelen van webapplicaties en REST-API's, met uitgebreide documentatie en sterke ondersteuning vanuit de ontwikkelaarsgemeenschap. Het biedt een gestroomlijnde ervaring dankzij de integratie met Visual Studio en andere Microsoft-tools. De uitstekende prestaties en ingebouwde beveiligingsmechanismen maken het een aantrekkelijke keuze voor zowel kleine als grote projecten binnen de Microsoft-stack.

2.5 Bun met Prisma & Elysia

2.5.1 Waarom

Bun is een efficiënt framework dat sterk lijkt op Node.js, waardoor het aantrekkelijk is voor ontwikkelaars die bekend zijn met dit ecosysteem. Prisma biedt uitstekende integratie met databases, waardoor het gemakkelijk is om gegevens te beheren. Elysia biedt veel out-of-the-box functies voor het ontwikkelen van API's, waardoor het een aantrekkelijke keuze is voor projecten die snel functionaliteit nodig hebben.

2.5.2 Ondersteuning / documentatie

Hoewel deze frameworks relatief jong zijn, bieden ze allemaal overzichtelijke en moderne documentatie. Echter, vanwege hun recentere opkomst kan de beschikbaarheid van community ondersteuning voor het oplossen van problemen nog beperkter zijn in vergelijking met gevestigde frameworks zoals Node.js.

2.5.3 Ontwikkelaarsproductiviteit

Het opzetten en uitbreiden van de API's met Bun, Prisma en Elysia verloopt over het algemeen snel en efficiënt, zodra de juiste bronnen online zijn gevonden. De frameworks bieden tools en functionaliteiten die de ontwikkelaarsproductiviteit bevorderen, waardoor ze snel aan de slag kunnen met het bouwen van API's.

2.5.4 Veiligheid

Elysia biedt een breed scala aan ingebouwde functionaliteiten voor beveiliging, waaronder validatie, Cross-Origin Resource Sharing (CORS), error handling en authenticatie.

2.5.5 Prestatie

De prestaties van Bun in combinatie met Prisma en Elysia zijn over het algemeen indrukwekkend, gezien de snelheid en efficiëntie die worden geboden. Deze combinatie is geschikt voor het bouwen van schaalbare API's die kunnen omgaan met grote hoeveelheden verkeer.

2.5.6 Conclusie

Bun met Prisma en Elysia biedt een veelbelovende combinatie voor het ontwikkelen van snelle, efficiënte en goed beveiligde API's. Hoewel deze stack relatief nieuw is en mogelijk nog niet dezelfde mate van community ondersteuning heeft als gevestigde frameworks, compenseren de uitstekende documentatie en de ingebouwde beveiligingsfunctionaliteiten dit gedeeltelijk. Ontwikkelaars kunnen profiteren van de snelheid en productiviteit die worden geboden door deze combinatie van tools, wat het een aantrekkelijke optie maakt voor zowel kleine als grote projecten. Met de groeiende populariteit en verdere ontwikkelingen in deze technologieën, kan Bun met Prisma en Elysia zeker een waardevolle keuze zijn voor het ontwikkelen van moderne API's.

2.6 Samenvatting REST-API technologieën

2.6.1 Postman API prestatie resultaten

Gedetailleerde rapporten met betrekking tot de prestaties zijn beschikbaar in de bijlagen. De tests werden uitgevoerd met behulp van Postman, waarbij 100 virtuele gebruikers werden gesimuleerd op een vast profiel gedurende een periode van 5 minuten. Het doel van de test was om de prestaties van de endpoints te meten bij het ophalen van gegevens van 50 studenten.

Name	***Express.js***	***Django***	***Asp.net***	***Bun & Elysia***
Amount of requests sent	26.616	26.113	26.657	26.754
Troughput requests/second	86.59	84.98	86,80	87.07
Avg. response time	10ms	30ms	7ms	9ms
Error rate	0.00%	0.00%	0.00%	0.00%

Uit de prestatieresultaten van verschillende REST-API technologieën blijkt dat alle frameworks robuuste prestaties leveren met minimale fouten en een acceptabele responstijd. Ze vertonen vergelijkbare doorvoersnelheden, met slechts kleine variaties tussen hen.

Express.js en ASP.NET hebben de laagste gemiddelde responstijden, terwijl Django aanzienlijk langere responstijden laat zien. Bun & Elysia presteren ook goed met een gemiddelde responstijd vergelijkbaar met die van Express.js en ASP.NET.

Op basis van deze prestatieresultaten kunnen valt te concluderen dat alle geteste frameworks geschikt zijn voor het ontwikkelen van REST-API's met betrouwbare prestaties en weinig fouten. Django lijkt echter meer geschikt voor kleinere datasets of situaties met minder verkeer, vanwege zijn iets langzamere responstijden in vergelijking met andere frameworks. De keuze tussen deze frameworks kan ook afhangen van andere factoren, zoals ontwikkelaarsvoorkeur, beschikbare resources en integratiemogelijkheden met bestaande systemen.

2.6.2 Ontwikkelaarsproductiviteit

Bij het vergelijken van API-technologieën op het gebied van ontwikkelaarsproductiviteit is het belangrijk op te merken dat de voorkeur voor een bepaalde technologie subjectief kan zijn en afhankelijk is van persoonlijke ervaringen en voorkeuren. Hieronder worden mijn bevindingen en voorkeuren gedeeld.

In termen van ontwikkelaarsproductiviteit gaat mijn voorkeur uit naar ASP.NET. De eenvoudige installatie van NuGet-packages voor functionaliteiten zoals validatie en autorisatie, gecombineerd met de naadloze integratie met SQL Server, maakt het ontwikkelen van API's met ASP.NET een vlotte ervaring. Bovendien verloopt het genereren van klassen en modellen op basis van bestaande tabellen in de database zeer efficiënt en van hoge kwaliteit. De overzichtelijke bestandsstructuur draagt bij aan de duidelijkheid van het project. Echter, het vinden van specifieke informatie voor een bepaalde .NET-versie kan soms een uitdaging zijn, aangezien er verschillende versies beschikbaar zijn.

Ook Express.js vind ik een aantrekkelijke optie . Het installeren van npm-packages verloopt soepel en de community en documentatie rond Express.js zijn uitstekend. Bovendien biedt Express.js een flexibele en intuïtieve ontwikkelomgeving, waardoor ontwikkelaars snel en efficiënt kunnen werken aan het bouwen van API's. De modulaire opzet van Express.js maakt het gemakkelijk om functionaliteiten toe te voegen en aan te passen aan specifieke projectbehoeften.

3 Conclusie

Voorafgaand aan de analyse van verschillende REST-API, is het belangrijk op te merken dat alle andere API-technologieën ook zeker goede keuzes zijn. Dit geldt met name voor individuen of bedrijven die liever met andere technologieën werken, zoals Python of JavaScript. Met alle frameworks die zijn overlopen, is het mogelijk om de best practices voor een REST-API te implementeren, waardoor het uiteindelijke oordeel afhankelijk is van individuele voorkeuren en bestaande technologiestacks.

Na uitgebreide analyse van verschillende REST-API technologieën voor Integreat, raad ik het gebruik van ASP.NET Core aan. Gezien het feit dat Integreat al werkt met .NET, zal de integratie met ASP.NET Core naadloos verlopen en gemakkelijk kunnen worden geïntegreerd in de bestaande applicaties en systemen. De soepele integratie met SQL Server, in combinatie met de degelijke documentatie én de mogelijkheid om essentiële functionaliteiten toe te voegen met behulp van NuGet-packages, maken het een ideale keuze. Met ASP.NET Core kan Integreat alle best practices voor een REST-API volgen en tegelijk efficiënt en veilig toegang bieden tot klantgegevens.

4 AI Engineering Prompts

Tijdens het schrijven van dit document heb ik gebruik gemaakt van de volgende prompts:

“Kunt u deze tekst verbeteren en aantrekkelijker maken voor de lezers in ongeveer X woorden?”

Deze prompt is bedoeld om de AI te vragen om een gegeven tekst te herformuleren, te verbeteren en aantrekkelijker te maken binnen een bepaald woordlimiet. Het is belangrijk op te merken dat de exacte woordlimiet (X) kan variëren afhankelijk van de specifieke tekst en de context.

“Herschrijf deze bronvermeldingen in IEEE standaard formaat, <voorbeeld van https://bib.kuleuven.be/> ”

Het doel van deze prompt is om de AI te instrueren de gegeven bronvermeldingen om te zetten naar het standaard formaat dat wordt gebruikt in de IEEE-richtlijnen voor bronvermelding. The AI wordt gevraagd dit te doen met als referentie een specifiek voorbeeld, in dit geval van https://bib.kuleuven.be/.

“Als u een lector was van toegepaste informatica, en mijn verslag zou verbeteren. Wat welke opmerkingen zou u dan geven?”

Deze prompt vraagt de AI om de rol van een lector toegepaste informatica aan te nemen en het verslag te beoordelen. De AI dient daarbij constructieve feedback te geven op de inhoud, structuur, duidelijkheid en technische juistheid van het verslag. Dit helpt om inzicht te krijgen in mogelijke verbeterpunten vanuit een educatief en vakinhoudelijk perspectief.

“Controleer dit verslag op grammaticale- en spellingsfouten”

Deze prompt instrueert de AI om een gegeven verslag te controleren op grammaticale fouten en spelvouten. Het doel is om ervoor te zorgen dat de tekst grammaticaal correct en vrij van typefouten is, wat de leesbaarheid en professionaliteit van het document ten goede komt.

5 Verwijzingen

Integreat.
https://www.integreat.be/
[Geraadpleegd 16 februari 2024].

"Application Programming Interface." Wikipedia. https://nl.wikipedia.org/wiki/Application_programming_interface
[‘Geraadpleegd op 15 maart 2024].

"Representational State Transfer." Wikipedia.
https://en.wikipedia.org/wiki/REST
[Geraadpleegd op 15 maart 2024].

"What is a REST API?" Red Hat.
https://www.redhat.com/en/topics/api/what-is-a-rest-api
[Geraadpleegd op 15 maart 2024].

"Introduction to Web APIs." Mozilla Developer Network.
https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction
[Geraadpleegd op 15 maart 2024].

"JavaScript APIs." W3Schools.
https://www.w3schools.com/js/js_api_intro.asp
[Geraadpleegd op 15 maart 2024].

Express.js
Kassem, Mahmoud. "How to Build an API with Node.js, Express, and TypeScript - 2024." Medium.
https://mahmoud-kassem.medium.com/how-to-build-an-api-with-node-js-express-and-type-script-2024-extended-part-1-6-f65df183dbc5
[Geraadpleegd op 15 maart 2024].

"Express.js Setup / Best Practices." YouTube. https://www.youtube.com/watch?v=Tw5Lu-pcpKS4
‘[‘Geraadpleegd op 15 maart 2024].

"Express.js Security." YouTube.
https://www.youtube.com/watch?v=RtLAwnYJOyQ
[Geraadpleegd op 15 maart 2024].

Django
Django REST framework
https://www.django-rest-framework.org/
[Geraadpleegd op 22 maart 2024].

"Django." YouTube.
https://www.youtube.com/watch?v=t-uAgl-AUxc
[Geraadpleegd op 22 maart 2024].

"Django Tutorial for Beginners." YouTube
https://youtu.be/llrlu4Qsl7c?si=CenP_g3JVmEq2jq7
[Geraadpleegd op 22 maart 2024].

ASP.NET

Microsoft. "Introduction to ASP.NET Core." [Online].
Beschikbaar: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0
[Geraadpleegd op 22 maart 2024].

C. Singh, "Build a RESTful Web API with .NET 8," Medium. [Online].
Beschikbaar: https://medium.com/@chandrashekharsingh25/build-a-restful-web-api-with-net-8-44fc93b36618.
[Geraadpleegd op 22 maart 2024].

"How to Create RESTful Web API using ASP.NET Core | .NET Core Tutorial," YouTube. [Online].
Beschikbaar: https://www.youtube.com/watch?v=sZnu-TyaGNk&t=442s.
[Geraadpleegd op 22 maart 2024].

"ASP.NET Core Web API Tutorial for Beginners - Building RESTful Service with C# and .NET Core," YouTube. [Online].
Beschikbaar: *https://www.youtube.com/watch?v=b8fFRX0T38M&t=1864s*
[Geraadpleegd op 22 maart 2024].

Bun, Elysia & Prisma

Prisma Documentation. [Online].
Beschikbaar: https://www.prisma.io/docs
[Geraadpleegd op 22 maart 2024].
Elysia Quick Start Guide. [Online].
Beschikbaar: https://elysiajs.com/quick-start.html
[Geraadpleegd op 22 maart 2024].

Bun HTTP API Documentation. [Online].
Beschikbaar: https://bun.sh/docs/api/http
[Geraadpleegd op 22 maart 2024].

"Introduction to Bun, Elysia & Prisma." YouTube. [Online].
Beschikbaar: https://www.youtube.com/watch?v=oBQ-rhbzQJc
[Geraadpleegd op 22 maart 2024].

R. Singh, "Creating REST API on Bun with ElysiaJS," Medium. [Online].
Beschikbaar: https://medium.com/@rajreetesh7/creating-rest-api-on-bun-with-elysiajs-bfa0627051c6
[Geraadpleegd op 22 maart 2024].

6 Bijlagen

6.1 OpenAPI Specificatie van Student Management API

https://github.com/LukasOlivier/bap/blob/95bf4f0eeca57d33cc053f16dd2d70eabcb1a2b3/documentation/openapi.yaml

6.2 Code van de uitgewerkte REST-API's

https://github.com/LukasOlivier/bap-rest-api/tree/97ecf1d7221fcf872d776d1d53c6c763d3388c01/Code

6.3 Postman API Performance testing reports

https://github.com/LukasOlivier/bap/tree/95bf4f0eeca57d33cc053f16dd2d70eabcb1a2b3/documentation/api-reports