

REST-API?

Gepresenteerd door Lukas Olivier

Goedenavond, dames en heren. Stelt u zich eens voor: u bent een bedrijf dat waardevolle informatie bezit, informatie die de wereld kan verbeteren. U wilt deze informatie delen op een efficiënte en veilige manier. Maar hoe bereikt u dat doel?

Het antwoord ligt in de wereld van de technologie, en specifieker nog, in het concept van een

REST-API, oftewel Representational State Transfer Application Programming Interface.

Een REST-API fungeert als een digitale brug die uw informatie toegankelijk maakt voor andere

toepassingen en websites. Het opent de deur naar een wereld van mogelijkheden, als een sleutel

die toegang geeft tot een schat aan kennis en innovatie.

Maar te midden van de vele keuzes, hoe selecteert u de juiste API technologie die perfect aansluit

bij uw behoeften en doelgroep? Dat is de vraag die velen van ons bezighoudt.

In deze TED Talk nemen we u mee op een reis door de fascinerende wereld van API's. We ontrafelen de hoe het werkt, bespreken de verschillende mogelijkheden en benadrukken de

voordelen van RESTful-API's - die uitblinken in gebruiksgemak, flexibiliteit en schaalbaarheid

AP- wat?

Een REST-API fungeert als een digitale brug die uw informatie toegankelijk maakt voor andere toepassingen en websites. Het opent de deur naar een wereld van mogelijkheden, als een sleutel die toegang geeft tot een schat aan kennis en innovatie.



API's zijn cruciale instrumenten voor moderne softwareontwikkeling, waarbij ze fungeren als de bruggen die de communicatie tussen verschillende softwaretoepassingen mogelijk maken. Het concept van een API, gaat veel dieper dan alleen maar het definiëren van regels en specificaties. Het is in feite de taal die applicaties spreken om met elkaar te interageren, wat resulteert in naadloze integratie tussen diverse systemen. Deze integratie is van onschabare waarde voor het ecosysteem van apps en websites, waardoor gebruikers een meer samenhangende en rijkere ervaring kunnen hebben.

Stap voor stap

- | | | | | | |
|---------------------------|------------------------|------------------------------|--------------------------------|----------------------------|-------------------------------------|
| ◆
01
Client Request | ◆
02
API-aanroep | ◆
03
Server verwerking | ◆
04
Database-interactie | ◆
05
Server response | ◆
06
Client Response Handling |
|---------------------------|------------------------|------------------------------|--------------------------------|----------------------------|-------------------------------------|

Laten we eens kijken naar een stapsgewijze uitleg van hoe een API werkt aan de hand van een voorbeeld waarbij een student zich inschrijft bij een school:

Stap voor stap

◆
01

Client Request

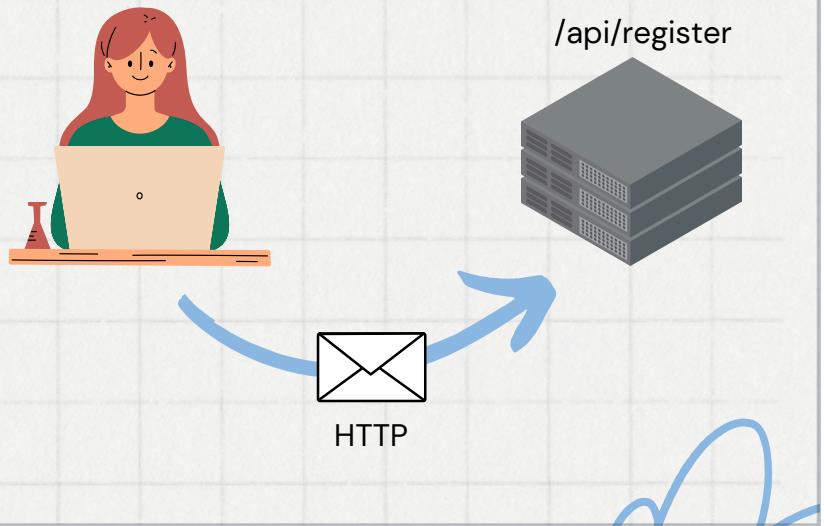


Het proces begint wanneer een student via een online formulier op de website van de school informatie invult om zich in te schrijven. Deze informatie kan onder meer de naam van de student, adres, geboortedatum en andere relevante gegevens omvatten.

Stap voor stap

❖
02

API aanroep



Zodra de student op de knop "Verzenden" klikt, wordt er een HTTP-verzoek naar de server gestuurd waarin de ingevulde informatie wordt opgenomen. Dit verzoek wordt verzonden naar een API-endpoint dat specifiek is ontworpen om inschrijvingsverzoeken te verwerken.

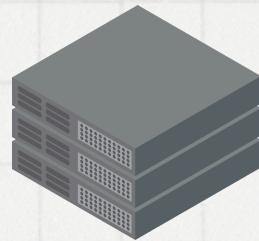
Stap voor stap

◆
O3

Server
verwerking

Leeftijd > 0?

Naam ingevuld?

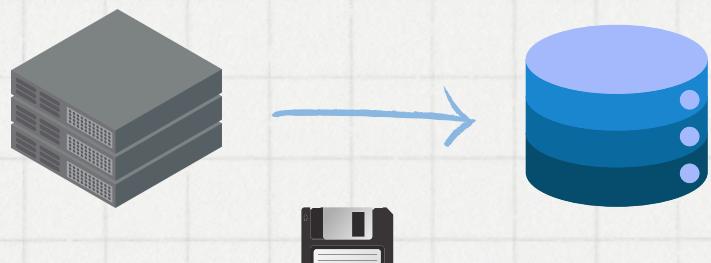


De server ontvangt het verzoek en begint de ingediende informatie te verwerken. Dit omvat het controleren van de geldigheid van de verstrekte gegevens, zoals het controleren of de geboortedatum in een geldig formaat is en het controleren of alle vereiste velden zijn ingevuld.

Stap voor stap

◆
04

Server
verwerking



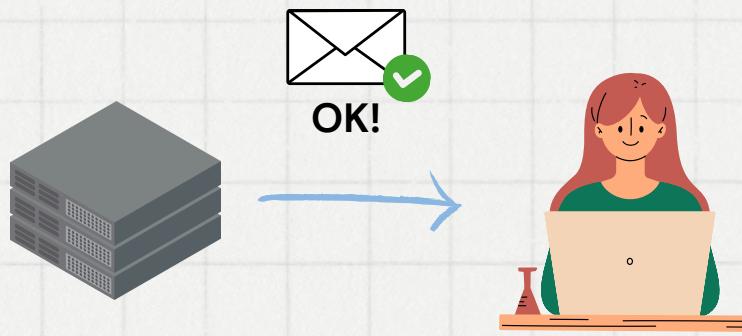
SAVING...

Nadat de server de ingediende informatie heeft gevalideerd,
communiceert deze met de database om de studentgegevens op te slaan.

Stap voor stap

◆
05

Server Response



Na succesvolle verwerking van het inschrijvingsverzoek stuurt de server een HTTP-respons terug naar de client. Deze respons bevat meestal een bevestiging van de inschrijving, samen met eventuele aanvullende informatie, zoals een inschrijvingsnummer of instructies voor vervolgstappen.

Stap voor stap

◆
06

Client Response
Handling

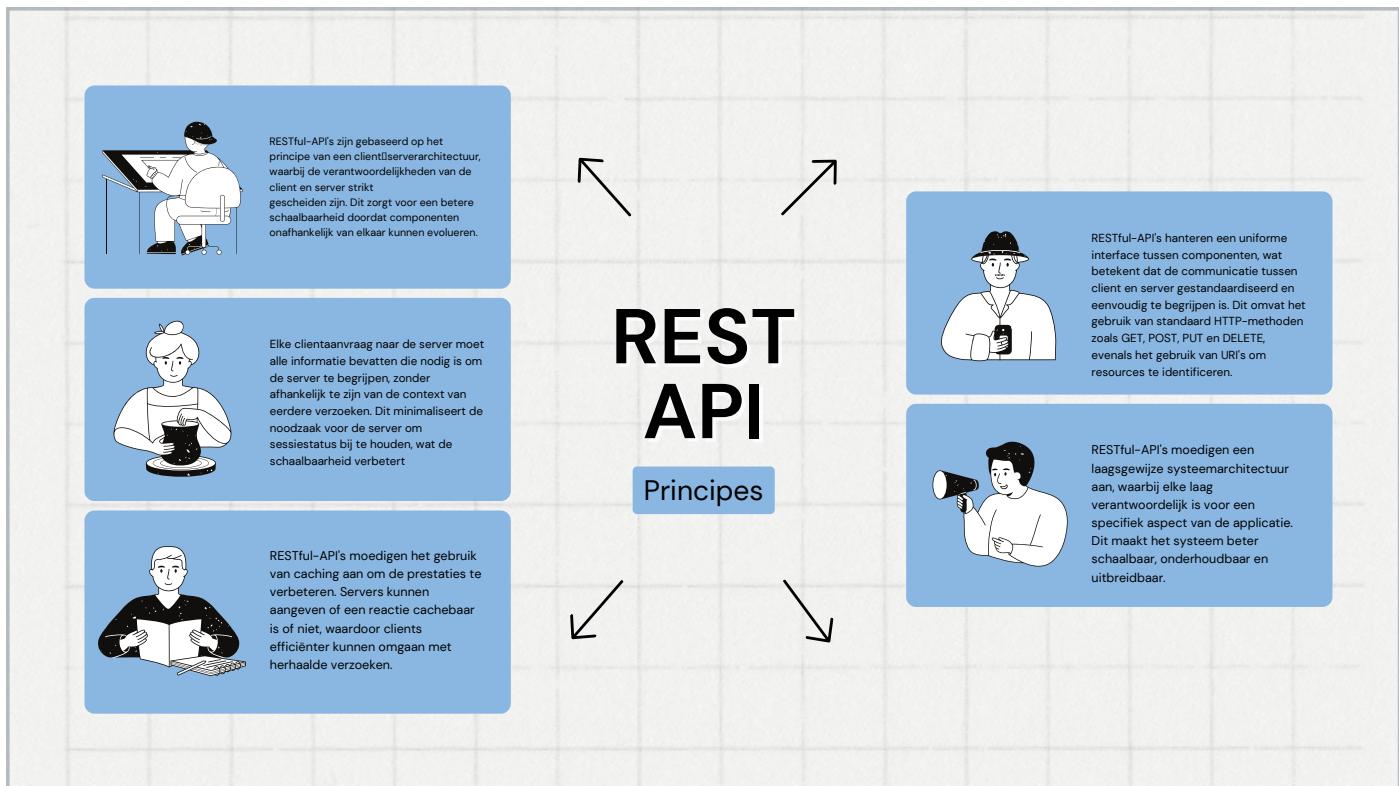


Tot slot ontvangt de student de respons van de server en wordt deze weergegeven op de website. De student ziet mogelijk een bericht dat de inschrijving succesvol is voltooid, waardoor ze verder kunnen gaan met het volgende deel van het inschrijvingsproces of andere acties kunnen ondernemen.

Stap voor stap

-
- | | | | | | |
|---------------------|------------------|------------------------|--------------------------|----------------------|-------------------------------|
| ◆ 01 Client Request | ◆ 02 API-aanroep | ◆ 03 Server verwerking | ◆ 04 Database-interactie | ◆ 05 Server response | ◆ 06 Client Response Handling |
|---------------------|------------------|------------------------|--------------------------|----------------------|-------------------------------|

Door dit proces van client-servercommunicatie te doorlopen, maakt de API een gestroomlijnde interactie mogelijk tussen de student en de schoolapplicatie, waardoor een efficiënt en geautomatiseerd inschrijvingsproces wordt geboden. Dit draagt bij aan een verbeterde gebruikerservaring en operationele efficiëntie voor zowel de studenten als de schooladministratie.

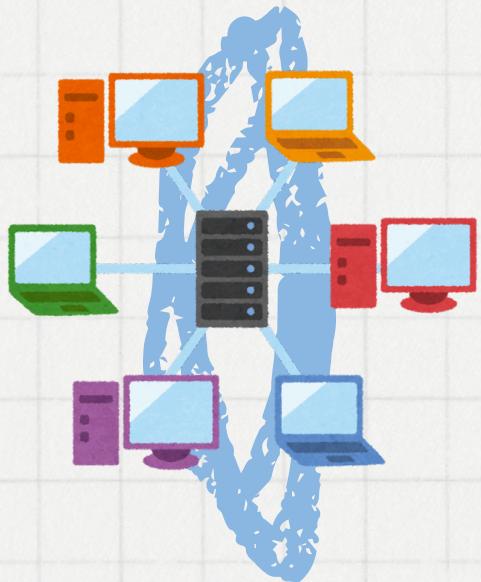


In de wereld van API's zijn er verschillende soorten, elk met hun eigen unieke kenmerken en

toepassingen. Een cruciale categorie die de aandacht verdient, zijn de Web-API's. Deze API's zijn

ontworpen om toegankelijk te zijn via het internet, waardoor ze een breed scala aan mogelijkheden bieden voor gegevensuitwisseling tussen verschillende platforms. Een specifiek

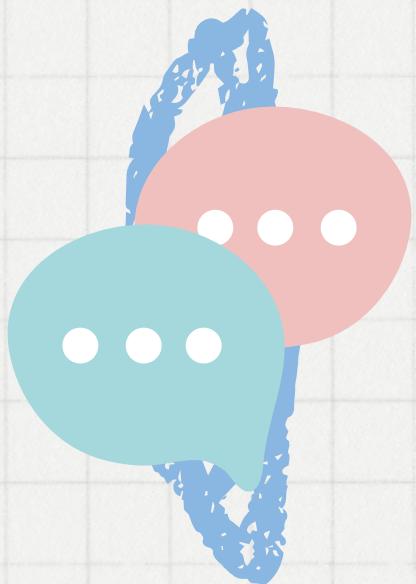
type web-API dat opvalt, zijn de RESTful-API's. Deze API's volgen een gestandaardiseerde aanpak voor het efficiënt en flexibel uitwisselen van data, wat essentieel is voor moderne webapplicaties en services. De belangrijkste eigenschappen zijn:



Client-Server Architectuur

RESTful-API's zijn gebaseerd op het principe van een client-serverarchitectuur, waarbij de verantwoordelijkheden van de client en server strikt gescheiden zijn. Dit zorgt voor een betere schaalbaarheid doordat componenten onafhankelijk van elkaar kunnen evolueren.

RESTful-API's zijn gebaseerd op het principe van een client-serverarchitectuur, waarbij de verantwoordelijkheden van de client en server strikt gescheiden zijn. Dit zorgt voor een betere schaalbaarheid doordat componenten onafhankelijk van elkaar kunnen evolueren.



Stateless Communicatie

Elke clientaanvraag naar de server moet alle informatie bevatten die nodig is om de server te begrijpen, zonder afhankelijk te zijn van de context van eerdere verzoeken. Dit minimaliseert de noodzaak voor de server om sessiestatus bij te houden, wat de schaalbaarheid verbetert.

Elke clientaanvraag naar de server moet alle informatie bevatten die nodig is om de server te begrijpen, zonder afhankelijk te zijn van de context van eerdere verzoeken. Dit minimaliseert de noodzaak voor de server om sessiestatus bij te houden, wat de schaalbaarheid verbetert.



Cachebaarheid

RESTful-API's moedigen het gebruik van caching aan om de prestaties te verbeteren. Servers kunnen aangeven of een reactie cachebaar is of niet, waardoor clients efficiënter kunnen omgaan met herhaalde verzoeken.

RESTful-API's moedigen het gebruik van caching aan om de prestaties te verbeteren. Servers kunnen aangeven of een reactie cachebaar is of niet, waardoor clients efficiënter kunnen omgaan met herhaalde verzoeken.



Uniforme Interface

RESTful-API's hanteren een uniforme interface tussen componenten, wat betekent dat de communicatie tussen client en server gestandaardiseerd en eenvoudig te begrijpen is. Dit omvat het gebruik van standaard HTTP-methoden zoals GET, POST, PUT en DELETE, evenals het gebruik van URI's om resources te identificeren.

RESTful-API's hanteren een uniforme interface tussen componenten, wat betekent dat de communicatie tussen client en server gestandaardiseerd en eenvoudig te begrijpen is. Dit omvat het gebruik van standaard HTTP-methoden zoals GET, POST, PUT en DELETE, evenals het gebruik van URI's om resources te identificeren.



Laagsgewijze Systeemarchitectuur

RESTful-API's moedigen een laagsgewijze systeemarchitectuur aan, waarbij elke laag verantwoordelijk is voor een specifiek aspect van de applicatie. Dit maakt het systeem beter schaalbaar, onderhoudbaar en uitbreidbaar.

RESTful-API's moedigen een laagsgewijze systeemarchitectuur aan, waarbij elke laag verantwoordelijk is voor een specifiek aspect van de applicatie. Dit maakt het systeem beter schaalbaar, onderhoudbaar en uitbreidbaar.

Best practices API

Laten we het hebben over best practices bij het ontwikkelen van API's. Deze best practices zijn niet alleen essentieel voor het waarborgen van de veiligheid en betrouwbaarheid van systemen, maar ze vormen ook de ruggengraat van een goed ontwikkelde API.



Laten we het hebben over best practices bij het ontwikkelen van API's. Deze best practices zijn niet alleen essentieel voor het waarborgen van de veiligheid en betrouwbaarheid van systemen, maar ze vormen ook de ruggengraat van een goed ontwikkelde API.



Hashing

Laten we beginnen met het belang van het hashen van wachtwoorden. Het hashen van wachtwoorden is een cruciale stap in het beveiligen van gebruikersgegevens. In plaats van wachtwoorden op te slaan als platte tekst, worden ze gehasht, wat betekent dat ze worden omgezet in een unieke reeks karakters. Deze gehashte wachtwoorden zijn moeilijk terug te zetten naar hun oorspronkelijke vorm, waardoor ze veiliger zijn in het geval van een inbreuk op de database.

Laten we beginnen met het belang van het hashen van wachtwoorden. Het hashen van wachtwoorden is een cruciale stap in het beveiligen van gebruikersgegevens. In plaats van wachtwoorden op te slaan als platte tekst, worden ze gehasht, wat betekent dat ze worden omgezet in een unieke reeks karakters. Deze gehashte wachtwoorden zijn moeilijk terug te zetten naar hun oorspronkelijke vorm, waardoor ze veiliger zijn in het geval van een inbreuk op de database.



Authorisatie

Het opzetten van een goed doordacht autorisatiesysteem stelt ons in staat om te bepalen welke gebruikers toegang hebben tot specifieke delen van onze API, zoals het opvragen van informatie over leerlingen van andere scholen. Dit kan worden gerealiseerd door middel van tokens, sessies of andere vormen van authenticatie, afhankelijk van de specifieke behoeften van het systeem.

Authorisatie vormt een essentieel onderdeel van de beveiliging van een API. Het opzetten van een goed doordacht autorisatiesysteem stelt ons in staat om te bepalen welke gebruikers toegang hebben tot specifieke delen van onze API, zoals het opvragen van informatie over leerlingen van andere scholen. Dit kan worden gerealiseerd door middel van tokens, sessies of andere vormen van authenticatie, afhankelijk van de specifieke behoeften van het systeem. In mijn geval heb ik gekozen voor JWT tokens. Deze tokens worden veel gebruikt vanwege hun eenvoudige implementatie, schaalbaarheid en hun vermogen om authenticatie en autorisatie in gedistribueerde systemen te ondersteunen.

2.0

Versioning

Door onze API's te versioneren, kunnen we ervoor zorgen dat oudere clients niet worden verstoord door wijzigingen in de API-structuur. Dit stelt ons in staat om nieuwe functies toe te voegen en fouten te corrigeren zonder bestaande integraties te verbreken.

Versioning van API's is ook een best practice die vaak over het hoofd wordt gezien, maar toch van groot belang is. Door onze API's te versioneren, kunnen we ervoor zorgen dat oudere clients niet worden verstoord door wijzigingen in de API-structuur. Dit stelt ons in staat om nieuwe functies toe te voegen en fouten te corrigeren zonder bestaande integraties te verbreken.



Validatie

Zonder validering zou deze foutieve invoer de database kunnen versturen en de integriteit van de gegevens in gevaar brengen. Met grondige validatie kunnen we dergelijke problemen voorkomen en de algehele stabiliteit en veiligheid van onze API verbeteren.

Validatie is een essentiële stap in API-ontwikkeling. Stel je bijvoorbeeld een nieuwe student voor die per ongeluk onjuiste informatie invoert, zoals een ongeldige e-mailadresformaat of een leeftijd buiten het toegestane bereik. Zonder validering zou deze foutieve invoer de database kunnen versturen en de integriteit van de gegevens in gevaar brengen. Met grondige validatie kunnen we dergelijke problemen voorkomen en de algehele stabiliteit en veiligheid van onze API verbeteren.



.env

Anders dan bij andere bestanden, worden .env-bestanden niet opgenomen in de broncode en blijven ze buiten de versiebeheersystemen zoals GitHub. Dit minimaliseert het risico op onbedoelde openbaarmaking van gevoelige informatie tijdens het delen van code of het pushen naar een gedeelde repository.

Het is belangrijk om geheime informatie, zoals API-sleutels en wachtwoorden, in aparte .env-bestanden te bewaren vanwege hun veiligheidsvoordelen tijdens runtime. Anders dan bij andere bestanden, worden .env-bestanden niet opgenomen in de broncode en blijven ze buiten de versiebeheersystemen zoals GitHub. Dit minimaliseert het risico op onbedoelde openbaarmaking van gevoelige informatie tijdens het delen van code of het pushen naar een gedeelde repository. Bovendien biedt het gebruik van .env-bestanden een handige en gestructureerde manier om gevoelige informatie te beheren, waardoor de algehele beveiliging van ons systeem wordt vergroot.

Best practices API

Kortom, het implementeren van deze best practices – van het hashen van wachtwoorden tot het versioneren van API's en het beheren van geheime informatie – is essentieel voor het bouwen van veilige, betrouwbare en goed onderhouden API's. Door deze richtlijnen te volgen, kunnen we de integriteit van onze systemen waarborgen en een solide basis leggen voor toekomstige groei en ontwikkeling.



Kortom, het implementeren van deze best practices - van het hashen van wachtwoorden tot het versioneren van API's en het beheren van geheime informatie - is essentieel voor het bouwen van veilige, betrouwbare en goed onderhouden API's. Door deze richtlijnen te volgen, kunnen we de integriteit van onze systemen waarborgen en een solide basis leggen voor toekomstige groei en ontwikkeling.

Experiment

- Express.js, gebaseerd op Node.js
- Django, ontwikkeld met Python
- ASP.NET met behulp van C#, ontwikkeld door Microsoft
- Bun, een relatief nieuw framework, is ontworpen met een focus op eenvoud, snelheid en modulariteit.



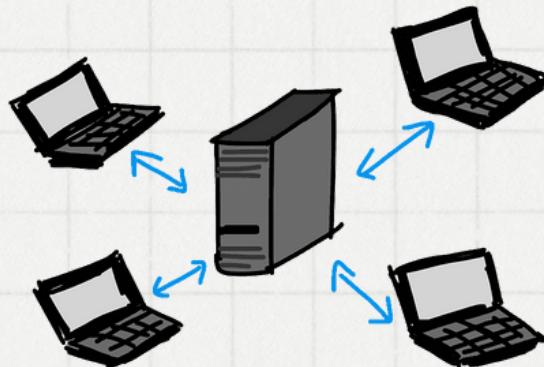
In mijn onderzoek heb ik de prestaties van vier prominente REST-API frameworks onder de loep

genomen:

- Express.js, gebaseerd op Node.js
- Django, ontwikkeld met Python
- ASP.NET met behulp van C#, ontwikkeld door Microsoft
- Bun, een relatief nieuw framework, is ontworpen met een focus op eenvoud, snelheid en modulariteit. Elk van deze frameworks heeft zijn eigen unieke kenmerken en specialisaties.

Postman

- 100 virtuele gebruikers
- 5 minuten
- ophalen van 50 studenten



Met behulp van Postman (een krachtige tool voor het testen en analyseren van API's) heb ik een simulatie opgezet met 100 virtuele gebruikers en de endpoints gedurende 5 minuten getest. Het doel? Het meten van de prestaties bij het ophalen van gegevens van 50 studenten.

Resultaten

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%

Wat hebben we ontdekt? Dat deze frameworks sterk presteren, met minimale fouten en een acceptabele responstijd. Laten we de cijfers eens nader bekijken. Het aantal verzoeken varieerde in de 26.000, met doorvoersnelheden van rond de 87 verzoeken per seconde. Opmerkelijk is dat de gemiddelde responstijd aanzienlijk versilde tussen de frameworks, variërend van 7ms tot 30ms. En hier komen de interessante nuances naar voren.

Resultaten

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	<u>10ms</u>	30ms	<u>7ms</u>	9ms
Error rate	0.00%	0.00%	0.00%	0.00%

Express.js en ASP.NET schitterden met de laagste gemiddelde responstijden, wat wijst op hun efficiënte verwerking van verzoeken.

Resultaten

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%

Django daarentegen liet langere responstijden zien, wat suggereert dat het beter geschikt zou kunnen zijn voor situaties met minder verkeer of kleinere datasets.

Resultaten

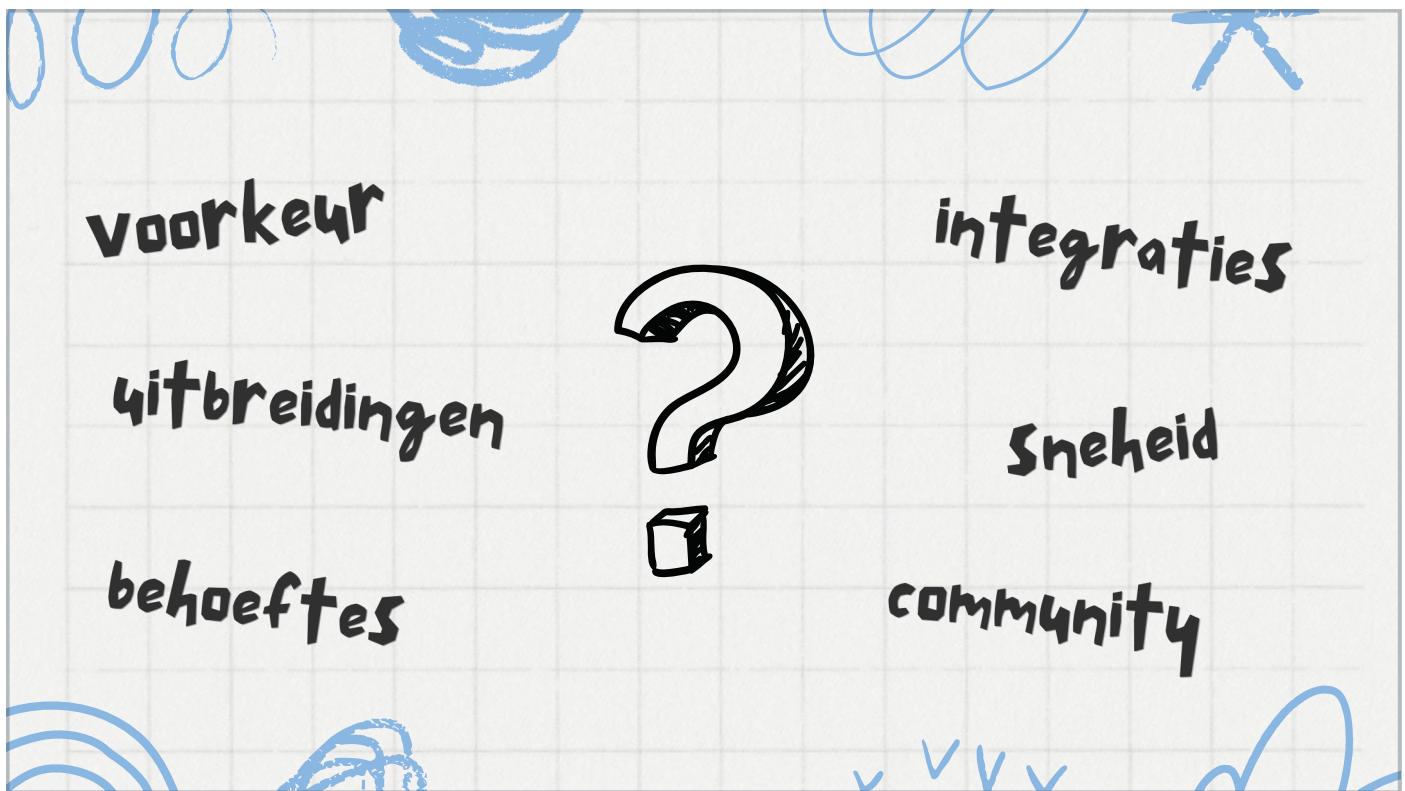
Name	Express.js	Django	Asp.net	Bun & Elysia
Amount of requests sent	26.616	26.113	26.657	26.754
Troughput requests/second	86.59	84.98	86.80	87.07
Avg. response time	10ms	30ms	7ms	9ms
Error rate	0.00%	0.00%	0.00%	0.00%

En laten we Bun niet vergeten, die verrassend genoeg presteerden met een gemiddelde responstijd vergelijkbaar met die van de topfavorieten.

Conclusie



Maar wat betekenen deze bevindingen? Dat alle onderzochte frameworks, ondanks kleine variaties, geschikt zijn voor het ontwikkelen van REST-API's met betrouwbare prestaties en minimale fouten.



Echter, bij het maken van een keuze moeten ontwikkelaars verschillende factoren overwegen. Misschien is Express.js de voorkeur vanwege zijn uitzonderlijk snelle verwerkingstijd. Of wellicht is ASP.NET Core aantrekkelijk vanwege zijn naadloze integratie met

SQL Server en uitgebreide extensiemogelijkheden via NuGet packages.

Maar uiteindelijk komt het neer op de specifieke behoeften van het bedrijf en de voorkeuren

binnen het team. Terwijl de prestatieresultaten een leidraad bieden, moeten de keuzes worden

afgewogen tegen de unieke vereisten van het project en de comfortniveaus van de ontwikkelaars

met de verschillende frameworks. Het vinden van de juiste balans tussen functionaliteit, prestaties

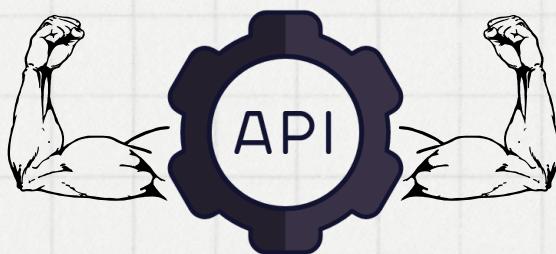
en ontwikkelaarservaring is cruciaal voor het maken van een weloverwogen beslissing

Impact

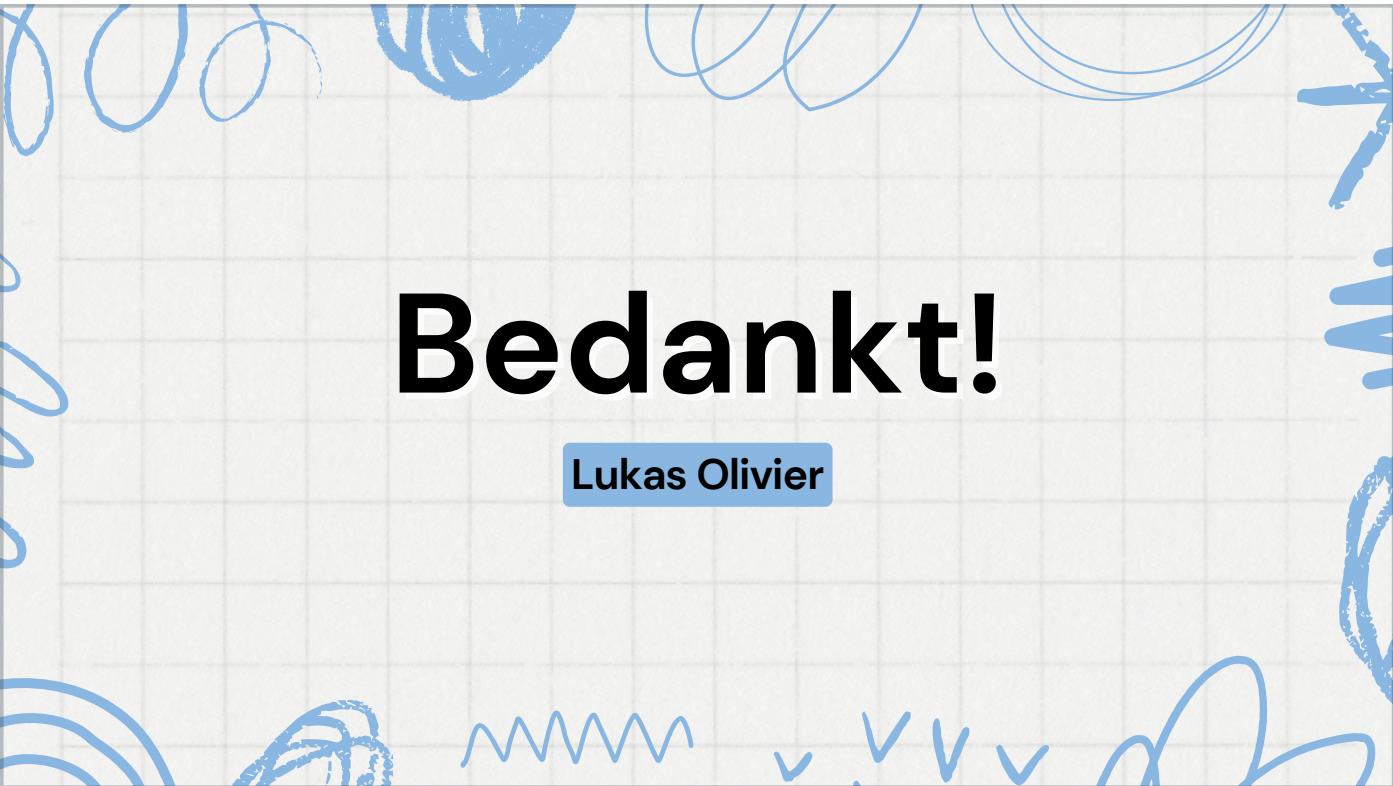


De keuze van de juiste API kan een significante impact hebben op de efficiëntie, veiligheid en gebruiksvriendelijkheid van uw applicatie. Wanneer we kijken naar de prestatieresultaten van verschillende REST-API technologieën, zoals Express.js, Django, ASP.NET en Bun, wordt duidelijk dat elk framework zijn eigen unieke kenmerken en voordelen heeft. Express.js blinkt uit in snelheid, terwijl Django een uitgebreid framework biedt met krachtige ingebouwde functies. ASP.NET core, daarentegen, biedt sterke integratie met SQL-server en uitgebreide extensiemogelijkheden met NuGet packages. En Bun, als nieuwkomer, biedt snelheid, eenvoud en modulariteit.

Omarm de kracht van API's!



Omarm de kracht van API's! Experimenteer met de verschillende frameworks en ontdek welke het beste past bij de behoeften en doelstellingen van uw project. Put kennis uit de open source community en deel uw ervaringen en creaties met de wereld. Door samen te werken en de kracht van API's te ontsluiten, bouwen we aan een betere, meer verbonden digitale toekomst. Laten we de mogelijkheden verkennen die API's bieden om innovatie te stimuleren, de efficiëntie te vergroten en nieuwe kansen te creëren voor groei en ontwikkeling. Met de juiste API aan onze zijde kunnen we onze ambities realiseren en een blijvende impact hebben in de wereld van technologie.



Bedankt!

Lukas Olivier