|  |  |
| --- | --- |
|  | Graduaatsproef |
|  |  |
|  | Zeno Boumerkhoufa  ProjectWerk  2022-2023 |

Inhoud

[Samenvatting 2](#_Toc136800724)

[Summary 3](#_Toc136800725)

[Inleiding 4](#_Toc136800726)

[Het groepswerk 5](#_Toc136800727)

[Het individueel deel 6](#_Toc136800728)

[De vergelijking 8](#_Toc136800729)

[Wat heb ik bijgeleerd 10](#_Toc136800730)

[Bronnenlijst 11](#_Toc136800731)

# Samenvatting

Hier heeft u het verslag van mijn graduaatsproef waarin 2 programmeertalen worden vergeleken met elkaar. Het gaat om de backend van een fleet-management applicatie die als groepsopdracht is gegeven door AllPhi. De volledige applicatie is een groepswerk dat is ontwikkeld met C# als backend en React als frontend. Voor mijn graduaatsproef heb ik de oorspronkelijke C# backend uit het groepswerk herschreven naar Python met PyCharm als ontwikkelingsomgeving. Door alleen het gedeelte dat betrekking heeft op voertuigen te herschrijven, kon ik de verschillen en overeenkomsten tussen de twee programmeertalen ervaren, evenals de voor- en nadelen van elk. Voor mijn graduaatsproef heb ik een andere database gebruikt dan die van het groepswerk, maar met dezelfde configuratie, zodat alles op dezelfde manier kan worden opgeslagen en goed kan worden vergeleken. Dit zorgt ervoor dat het resultaat niet kan worden gemanipuleerd door gegevens op een gemakkelijkere of andere manier in de database op te slaan, alles wordt op een gelijke manier verwerkt.

# Summary

Here is the report of my graduate’s thesis in which two programming languages are compared. It concerns the backend of a fleet management application that was assigned by AllPhi. The complete application is a group project that was developed with C# as the backend and React as the frontend. For my graduate’s thesis, I reworked the backend originally created in C# in the group project using Python with PyCharm as the development environment. By reworking (only) the part related to vehicles, I was able to experience the differences and similarities between the two programming languages and understand the advantages and disadvantages of each. For my graduate’s thesis, I used a different database than the one used in the group project, but with the same setup so that everything could be stored in the same way and easily compared, ensuring that the results cannot be manipulated by storing data in the database in an easier or different manner. All data is processed in a consistent manner.

# Inleiding

Als graduaatsproef voor het vak ProjectWerk moesten we een deel van ons groepswerk opnieuw maken in een programmeertaal die we niet in de lessen leren, maar waar we wel in geïnteresseerd zijn. Ik had origineel gekozen om enkel de API van de Vehicles klasse van het groepswerk te herwerken in Python maar heb dan later besloten om de volledige backend van de Vehicle klasse te herwerken om niet te veel problemen te hebben met de link tussen Python en C# bestanden.

Ik heb hiervoor gekozen omdat ik vorig academiejaar in mijn 2de semester stage heb gelopen voor 4 maanden bij Crosslang waar ik in Python heb leren werken met PyCharm als ontwikkelingsomgeving. Dat was ook een API dat ik heb opgezet maar i.p.v. met een modellaag was dat met 3 microservices die in een docker(-compose) container waren opgezet.

Als eerste heb ik een tutorial/user-guide gevolgd op <https://fastapi.tiangolo.com/tutorial/> om alles nog eens op te frissen van API en FastAPI in Python. Verder heb ik dan het groepswerk erbij genomen en bekeken welke bestanden ik in Python ook nodig heb om het te kunnen herwerken met dezelfde functionaliteiten. Eenmaal begonnen met de API om te zetten naar een FastAPI in Python merkte ik al snel dat gegevens van C# bestanden van de modellaag door geven naar de FastAPI in Python niet zo gemakkelijk ging en heb dan besloten om de volledige backend voor de Vehicle klasse in Python te herwerken. Eenmaal ik dit besloten had, ben ik voor alle C# bestanden die van belang zijn om Vehicles te kunnen beheren in de fleet-management applicatie begonnen met de modellen te vertalen en te herwerken en heb ik op dezelfde wijze als men een backend in C# maakt de backend herwerkt in Python.

# Het groepswerk

Het groepswerk was een fleet-management applicatie die als opdracht was gegeven door AllPhi. Dit groepswerk heb ik gemaakt met 4 andere medestudenten. We hebben van in het begin afgesproken wie zich op welke delen ging focussen om alles zo vlotjes mogelijk te laten verlopen. Om alles goed te laten verlopen hebben we naast de ene dag dat we samen kwamen op campus ook veel contact gehad via andere kanalen zoals WhatsApp.

De fleet-management applicatie had 3 grote onderdelen: het personeel/bestuurders, de tankkaarten en de voertuigen. Van alle 3 konden er nieuwe toegevoegd worden, bestaande aangepast worden en verwijderd worden. Het verwijderen werd enkel als soft-deletes gedaan zoals gevraagd door AllPhi. Verder moest er bij het verwijderen van een persoon, tankkaart of voertuig ook altijd rekening gehouden worden met de gekoppelde gegevens zodat een persoon niet gekoppeld kon zijn aan een voertuig dat verwijderd werd. Er zijn dus veel controles toegevoegd naast het controleren van de ingevoerde gegevens om te controleren op gekoppelde gegevens die verwijderd waren en of bij het verwijderen de gekoppelde gegevens werden aangepast.

# Het individueel deel

Voor het individueel deel van de graduaatsproef was de opdracht om van jouw eigen groepswerk een deel te herwerken in een andere programmeertaal in een andere ontwikkelingsomgeving die we niet expliciet leren maar misschien wel eens hebben bekeken in de lessen. Ik heb als opdracht gekozen om de API van het deel van de voertuigen te herwerken in Python, waarmee ik tijdens mijn stage had leren werken in de ontwikkelingsomgeving PyCharm. Aangezien ik op mijn stage ook een API heb opgezet in Python ging de tutorial/user-guide op <https://fastapi.tiangolo.com/tutorial/> redelijk vlotjes.  
Als eerste bij het opzetten van een project vraagt PyCharm een “virtual environment”. Hier heb ik wat problemen gehad doordat de Python versie die ik geïnstalleerd had staan niet volledig up to date was en niet volledig correct was geïnstalleerd. Nadat ik Python verwijderd en opnieuw geïnstalleerd had, is dit probleem snel opgelost geraakt en kon ik beginnen met code te schrijven.   
Na een simpele GET-request “Hello World” en een request die met een meegegeven naam “Hello {naam}” teruggaf ben ik begonnen aan de herwerking van de GET om alle Vehicles uit de database op te vragen. Hierbij begon ik al duidelijk de verschillen en gelijkenissen zien tussen de programmeertalen.  
A screen shot of a computer code

Description automatically generated with low confidence  
A screen shot of a computer program

Description automatically generated with low confidence

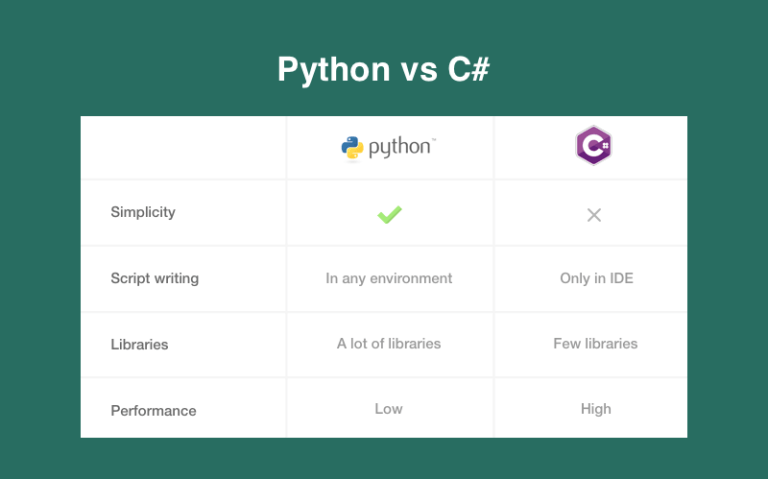
Zoals hierboven op de screenshots van de GET-request voor alle Vehicles te zien is, is er wel een duidelijk verschil maar zijn er ook een paar duidelijke gelijkenissen. Eén van de duidelijke verschillen is de manier waarop alles wat binnen de ‘methode of loops’ behoort, wordt afgezonderd van de andere code. C# werkt met de accolades en Python werkt met een dubbele punt en een tab-spatie. De manier waarop de code wordt uitgevoerd (try) en wordt opgevangen in de catch (C#) en in de except (Python) zijn wel bijna identiek op het verschil van de return en raise methodes na. Deze hebben wel hetzelfde doel maar in C# kan een return voor meer gebruikt worden en een raise in Python kan alleen maar voor alle soorten exceptions worden gebruikt.

Nadat dit werkte, heb ik nog enkele methodes toegevoegd aan de API: een GET-request waarbij een vehicle wordt opgevraagd met een specifiek chassisnummer, een POST-request om een vehicle te kunnen toevoegen, een PUT-request om een vehicle te kunnen updaten en een DELETE-request met een soft-delete. Bij het GET-request om te zoeken op chassisnummer, het PUT-request om een vehicle aan te passen en de DELETE-request werd het chassisnummer meegegeven om het juiste vehicle uit de database op te kunnen halen en om die dan ook te kunnen aanpassen voor een update of soft-delete. Bij alle andere GET-requests wordt er niets meegegeven. Bij de POST-request om een vehicle toe te voegen is het wel nodig een volledig vehicle object mee te geven.

# De vergelijking

Als we enkel de programmeertalen gaan vergelijken dan is het verschil niet zo groot aangezien ze beide voor software en web development worden gebruikt, het grootste verschil ligt hem dan in de syntax en de moeilijkheidsgraad om de taal te leren. Python zou door de samengestelde syntax van al de verschillende talen waaruit de inspiratie komt gemakkelijk te leren zijn. C# is ook één van de gemakkelijkere programmeertalen om te leren maar is toch een beetje moeilijker aan te leren dan Python.

Ikzelf heb ondervonden dat Python inderdaad tot op een bepaald niveau gemakkelijker was doordat je elk deel van de code eigenlijk overal kan importeren en/of requiren wat in C# wat meer prutswerk is met de usings die bovenaan elk bestand gebruikt worden.



Een ander verschil dat Python wat gemakkelijker maakt voor mijzelf is dat je elk project op eenzelfde manier start en dat de soort applicatie of software die je maakt bepaald wordt door welke modules je importeert en gebruikt. Bij C# moet je al bij het aanmaken van je project kiezen wat voor project je wilt en voor elke soort heb je wel een paar keuzes. Als je dus niet goed bekend bent met de soorten projecten die er bestaan dan kan het zijn dat je de verkeerde keuze maakt of dat je een iets betere keuze had kunnen maken. Binnen een C# project kan je ook nog eens kiezen wat voor bestanden je gebruikt, bijvoorbeeld in een wpf-project kan je voor XAML kiezen, een window toevoegen of een page, deze lijken op elkaar maar hebben ook hun verschillen dus je moet wel weten welke het beste past bij jouw project.  
In Python maakt dit allemaal minder uit aangezien je de naam van het bestand kan kiezen en de extensie .py moet zijn en dan kan je aan de hand van de geïmporteerde modules kiezen wat je met dat bestand gaat doen in dat project.

Buiten de praktische verschillen met betrekking tot de bestandstypes die je gebruikt zit er bij elk natuurlijk ook een verschil in de syntax. Zoals eerder al besproken is het gebruik van accolades in C# en het gebruik van een dubbele punt gevolgd door een tab-spatie op de volgende lijn een verschillende manier die wordt gebruikt bij het afscheiden van delen code die tot een klasse of methode behoren.  
Een Ander verschil tussen de twee programmeertalen is dat C# vooral object-georiënteerd gebruikt wordt en dat je in Python alle manieren kan toepassen zonder extra modules of library’s te moeten installeren.

Het grootste verschil tussen te twee talen is dat C# is ontwikkeld met het Windows besturingssysteem in gedachten. Je kan het op IOS gebruiken maar dan zullen bepaalde functionaliteiten niet beschikbaar zijn door het besturingssysteem.  
Python is onafhankelijk van een besturingssysteem ontworpen en is aangepast zodat het gebruikt kan worden op alle besturingssystemen die er ter beschikking zijn zoals Windows, IOS en Linux.

Uiteindelijk kiest ieder bedrijf of persoon welke van deze twee en vele andere programmeertalen bij hen past. Een persoon of bedrijf maakt de keuze aan de hand van welke programmeertaal het meeste biedt wat je kan gebruiken. Als je software maakt voor Windows machines en je wilt dat het gemakkelijk compileert en te gebruiken is dan zou je eerder voor C# kiezen. Ben je van plan om software te schrijven die op alle verschillende besturingssystemen gemakkelijk herkend wordt dan zou je Python kunnen gebruiken. Ben je meer gericht op mobiele applicaties dan hebben beide de nodige library’s hiervoor maar dan ben je misschien toch beter met een andere taal zoals React of Kotlin.

Uiteindelijk kies je zelf wat je gebruikt en waarvoor je het gebruikt en als je niet kan kiezen dan is er genoeg te vinden online in de documentatie en onafhankelijke user-guides wat je kan gebruiken voor welke soort applicaties en/of software.

# Wat heb ik bijgeleerd

In het algemeen heb ik leren werken met Python in PyCharm, het was een leerervaring met ups en downs omdat sommige dingen gemakkelijker gingen dan andere zoals met de meeste dingen. Het ging ook redelijk gemakkelijk doordat ik ermee al in aanraking was gekomen tijdens mijn stage en doordat de manier van programmeren gelijkaardig is als bij C#.

Verder heb ik dan ook geleerd hoe je in Python een API opzet met de verschillende soorten requests, dit was zeer gelijkaardig aan de manier waarop dit wordt gedaan in C# waardoor het vlotjes ging.

De verbinding met een MySQL database heb ik voor eerst in Python gedaan voor dit project. Ik was hier niet bekend mee maar heb dit vlotjes kunnen doen aan de hand van voorbeelden en tutorials.

# Bronnenlijst

Tiangolo. FastAPI – tutorial – user guide. Geraadpleegd op <https://fastapi.tiangolo.com/tutorial/>

Groepswerk (Stolos). Alle broncode van de backend. Geraadpleegd op <https://lucvervoort0708@dev.azure.com/lucvervoort0708/Stolos/_git/Stolos>

Eigen Graduaatsproef. Broncode van mijn graduaatsproef in Python. Geraadpleegd op <https://github.com/ZenoBoumerkhoufa/individueel-project.git>

ThePythonGuru. Afbeelding met een paar hoofdverschillen tussen C# en Python. Geraadpleegd op <https://thepythonguru.com/reasons-to-use-python-over-other-programming-languages/>