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

# 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 bachelor'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 bachelor'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 gevolg 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 te 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 heb ik alle C# bestanden die betrekkingen hadden voor Vehicles te kunnen beheren in het fleet-management applicatie ben ik begonnen met de modellen te vertalen en te herwerken en in dezelfde wijze als men een backend in C# maakt de backend herwerkt in Python.

# Het groepswerk

Het groepswerk was een fleet-management applicatie dat 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 hadden we buiten de ene dag dat we samen kwamen op campus ook veel contact gehad via andere kanalen zoals WhatsApp.

Het fleet-management applicatie had 3 grote onderdelen, het personeel/bestuurders, 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 met het verwijderen van een persoon, tankkaart of voertuig moest er ook altijd rekening gehouden worden met de gekoppelde gegevens zodat een persoon niet gekoppeld kon zijn aan een voertuig dat verwijderd is geweest. Er zijn dus veel controles toegevoegd geweest buiten het controleren van de ingevoerde gegevens om te controleren op gekoppelde gegevens en bestaande gegevens.

# Individueel deel

Voor het individueel deel of het 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, vanuit mijn stage had ik hiermee leren werken in de ontwikkelingsomgeving PyCharm. Aangezien ik op mijn stage ook een API heb opgezet in Python ging de tutorial/user-guide op redelijk <https://fastapi.tiangolo.com/tutorial/> vlotjes.   
Nadat ik een simpele GET-request die “Hello World” en een request die met een meegegeven naam “Hello {naam}” teruggaf ben ik begonnen aan de herwerking van de GET van 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 ook een paar duidelijke gelijkenissen. Eén van de duidelijke verschillen is de manier waarop alles wat binnen de methode of loops behoort word afgezonderd van de andere code, C# werkt met de accolades en Python werkt met een : en een tab-spatie. De manier waarop de code word uitgevoerd (try) en word opgevangen in de catch (C#) en in de except (Python) zijn wel bijna identiek buiten het verschil van de return en raise methodes, deze hebben wel hetzelfde doel buiten dat in C# een return voor meer gebruikt kan worden en dat een raise in Python alleen maar voor alle soorten exceptions word gebruikt.

Nadat dit werkende was heb ik nog enkele methodes toegevoegd aan de API, de GET-request waarbij een vehicle word 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 voor de update te kunnen aanpassen voor een update of soft-delete, bij alle andere GET-requests word er niets meegegeven en bij de POST-request om een vehicle toe te voegen is er wel een volledig vehicle object nodig om mee te geven.