# Emerging trends research



Naam: Rienk Engbrenghof

Email: [429059@student.fontys.nl](mailto:429059@student.fontys.nl)

Klas: S6-RB03T

# Introductie

Binnen dit semester is het van belang om te laten zien dat we kritisch kunnen denken en dat we ons werk kunnen verdelen in vragen waar we onderzoek naar kunnen doen. Een van deze onderzoeken is het emerging trends onderzoek. Hierbij heb ik gekozen voor het onderwerp: programming paradigms.

Door middel van subvragen zal binnen dit onderzoek de hoofdvraag worden opgelost. Deze vragen worden door het DOT-framework ondersteund. In mijn geval zal het onderzoek kunnen worden toegepast in mijn individuele project.

# Logboek

|  |  |
| --- | --- |
| 4/4/2022 | Feedback op onderzoeksvragen verwerkt. |
| 13/4/2022 | Begonnen met emerging trends research. |
| 16/4/2022 | Verder uitwerken onderzoeksmethoden subvraag 1. |
| 20/4/2022 | Verder uitwerken onderzoeksmethoden subvraag 2 en hoofdvraag. |

# Inhoudsopgave

[1. Methodiek 4](#_Toc101365472)

[2. Subvragen 5](#_Toc101365473)

[2.1 Welke programming paradigms zijn er allemaal? 5](#_Toc101365474)

[2.1.1 Literature search 5](#_Toc101365475)

[2.1.2 Available product analysis 7](#_Toc101365476)

[2.1.3 Conclusie subvraag 1 7](#_Toc101365477)

[2.2 Waarom kies ik voor C#? 8](#_Toc101365478)

[2.2.1 Available product analysis 8](#_Toc101365479)

[2.2.2 Gap analysis 9](#_Toc101365480)

[2.2.3 Conclusie subvraag 2 9](#_Toc101365481)

[3. Hoofdvraag 10](#_Toc101365482)

[3.1 Literature search 10](#_Toc101365483)

[3.2 Subvraag onderbouwing 10](#_Toc101365484)

[4. Conclusie 11](#_Toc101365485)

[5. Bronvermelding 12](#_Toc101365486)

# Methodiek

De hoofdvraag die in dit onderzoek zal worden beantwoord is: Hoe kan ik een programming paradigm in mijn twitter applicatie, die geprogrammeerd is in C#, toepassen?

Deze hoofdvraag zal door middel van de volgende twee subvragen beantwoord worden:

1. Welke programming paradigms zijn er allemaal?

**Methoden**

* Literature search, door middel van deze methode zal het onderzoek de verschillende paradigms bevatten.
* Available product analysis, met deze methode kan worden besloten welk paradigms het beste werkt voor mijn project.

1. Waarom kies ik voor C#?

**Methoden**

* Available product analysis, door middel van deze methode zal worden vastgesteld welke alternatieven er allemaal zijn.
* Gap analysis, door middel van deze methode zal worden vastgesteld wat de ideale situatie is en hoe C# mij hier gaat brengen.

De hoofdvraag zal vervolgens naast de twee subvragen nog een literature search bevatten. Zodat deze vraag volledig beantwoord wordt.

# Subvragen

## 2.1 Welke programming paradigms zijn er allemaal?

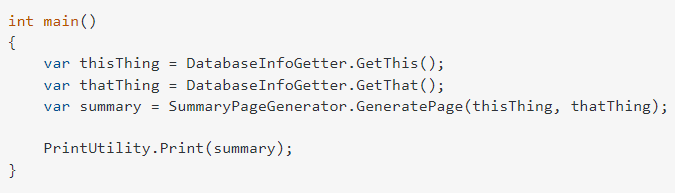
### 2.1.1 Literature search

De filosofie achter de programming paradigms varieert van bron tot bron. Sommige bronnen zegt dat er twee stromingen zijn binnen de paradigms (GeeksForGeeks bron 2), terwijl andere bronnen spreken over 4 paradigms (hackr.io bron 3).

Na uitgebreid zoeken, blijkt dat de meeste bronnen het principe van de 4 paradigms aanhouden. Deze 4 paradigms worden dan ook verder omschreven binnen dit onderzoek.

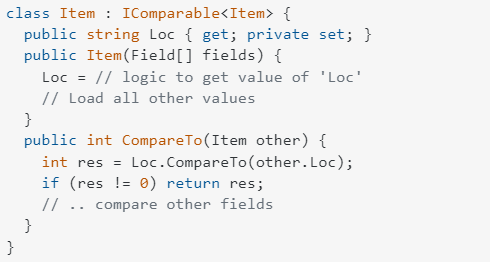
Het eerste paradigm waarover gesproken wordt is: **procedural programming**. Dit paradigm staat ook wel bekend als imperative programming. Binnen dit paradigm zijn statements gestructureerd tot procedures. Dit houdt in dat dit paradigm gebruik maakt van een volgorde van procedures. Hierin wordt de computer stap voor stap verteld wat hij moet doen.

Een voorbeeld van een stuk code dat volgens het procedural programming paradigm gestructureerd is staat hieronder. Hierin is het duidelijk dat het van boven naar beneden werkt.



Een ander paradigm is **logical programming**, hierbij vertel je het systeem wat je nodig hebt i.p.v. dat je het systeem een stapsgewijze oplossing geeft (procedural programming). Logical programming heeft statements nodig die waar zijn (logisch) en wanneer er dan door de gebruiker wordt ingegeven welke beslissingen moeten worden gemaakt, dan kan het systeem dit zelf doen.

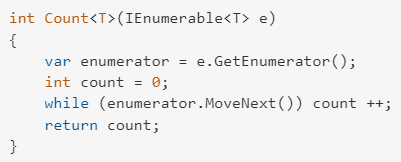
Een voorbeeld van logical programming staat hieronder, dit is een stuk code (dat nog verder ingevuld dient te worden), die items vergelijkt onder bepaalde voorwaarden. Deze voorwaarden kunnen worden vastgesteld door de gebruiker, het systeem maakt vervolgens zelf de logische keuzes.



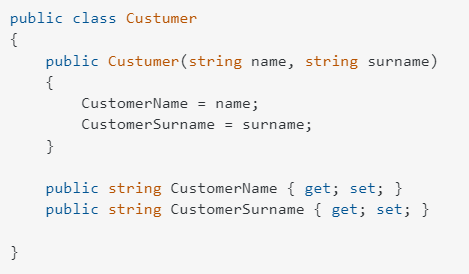
Het derde programming paradigm is **functional programming**, een van de kenmerken van functional programming is dat het gebruik maakt van pure functies. Hierbij krijgt een functie een input (bijvoorbeeld een lijst), de functie kan deze lijst modificeren en vervolgens retourneren. Hierbij wordt geen gebruik gemaakt van globale variabelen.

Functional programming kent nog andere kenmerken zoals: recursion (een functie kan zichzelf aanroepen binnen de functie), referential transparency (een functie retourneert hetzelfde onder elke omstandigheid) en immutable variables (variabele kan niet worden aangepast na het initialiseren).

Onderstaande afbeelding is een voorbeeld van een pure functie. Hierin wordt een input e gegeven, hier wordt een count over uitgevoerd en vervolgens wordt deze count geretourneerd.



Het vierde paradigm dat besproken wordt in het artikel is **object-oriented programming**, dit paradigm kent verschillende kenmerken. Deze kenmerken zijn: encapsulation (onnodige details worden in klasses gestopt, data wordt gebundeld), inheritance (een klasse kan overerven van een andere klasse), data abstraction (een versimpeld beeld van het hele plaatje) en polymorphism (een variabele, functie of object kan meerdere vormen aannemen).

Het komt er dus op neer dat er met objecten wordt gewerkt en dat deze objecten kenmerken en functies hebben. Met deze objecten kan vervolgens worden gewerkt.

De afbeelding hiernaast laat een voorbeeld zien van een customer object. Dit customer object heeft de variabelen name en surname. Met dit object kan vervolgens makkelijk geprogrammeerd worden omdat het een duidelijke structuur aanbrengt.

Over het algemeen worden procedural en object-oriented programming samengevat onder declaritive programming. Functional en logical programming worden veelal samengevat onder imperative programming.

### 2.1.2 Available product analysis

Om te bekijken welk paradigm het beste aansluit bij mijn project, wordt er gebruik gemaakt van een available product analysis.

Na de literature search is naar voren gekomen dat alle besproken paradigms beschikbaar zijn in C# (de taal waar mijn applicatie in wordt geschreven, waarom dit zo is wordt besproken in [hoofdstuk 2.2](#_2.2_Waarom_kies)).

Tijdens onze opleiding is ons aangeleerd om object georiënteerd te werk te gaan. Maar na de literature search ([hoofdstuk 2.1.1](#_2.1.1_Literature_search)) blijkt dat wij alle paradigms wel ooit toegepast hebben.

Echter moet de keuze worden gemaakt voor 1 voornamelijk paradigm. En op basis van ervaring en het juist aanbrengen van structuur zal het object-oriented programming de voorkeur hebben. Alleen worden de andere paradigms (onbewust) ook toegepast.

Het procedural programming zal worden toegepast op een plek waar het nodig is om stapsgewijs te werken. Het logical programming zal worden toegepast op plekken waar algoritmes en vergelijkingen nodig zijn. Dit is meestal het geval in applicaties. Ook functional programming zal worden toegepast binnen de applicatie. Alleen is de kans groot dat de functies niet puur zijn omdat er gebruik wordt gemaakt van globale variabelen.

### 2.1.3 Conclusie subvraag 1

Al bij al zal de applicatie dus voornamelijk gebruik maken van object-oriented programming, maar ook de andere paradigms (besproken in [hoofdstuk 2.1.2](#_2.1.2_Available_product)) zullen aanwezig zijn binnen de applicatie.

## 2.2 Waarom kies ik voor C#?

### 2.2.1 Available product analysis

In dit hoofdstuk wordt begonnen met het opnoemen van verschillende alternatieven voor de Backend taal. Binnen dit onderzoek is het van belang om de keuze voor C# te ondersteunen. Deze alternatieven worden vervolgens in [hoofdstuk 2.2.2](#_2.2.2_Gap_analysis) met elkaar vergeleken.

De alternatieven moeten voldoen aan de volgende eisen: geschikt voor backend, een goede performance, integreerbaar met het .NET framework , makkelijk te gebruiken en welbekend (voor een goede overdraagbaarheid).

**Talen en kenmerken**

**- C#,** deze taal is vrij bekend onder programmeurs. Een voordeel ervan is dat het een snellere en veiligere oplossing is voor C. Het is volledig geïntegreerd met het .NET framework. Een nadeel is dat het een steile leercurve heeft en dat het minder flexibel is dan talen zoals C++.

**- Python,** python is een relatief simpele taal die zeer welbekend is. Veel grote applicaties zijn al gebouwd met python. Doordat het zo bekend is, bestaan er veel toolkits waarvan gebruik kan worden gemaakt. Een nadeel van python is dat het packages nodig heeft om gebruik te maken van het .NET framework.

**- Java,** deze taal is ook zeer bekend onder programmeurs. Java staat bekend als een taal die overal gerund kan worden en zeer toegankelijk is. Ook is het goed geschikt voor mobiele applicaties. Een nadeel van Java is zijn integratie met de cloud.

**- Ruby,** deze taal is binnen onze opleiding vrij anoniem gebleven. Echter is een grote community en aanhang voor deze taal. Het is geschikt als backend taal. Ruby kent veel verschillende tools waardoor het een tijd efficiënte taal is. Een nadeel van Ruby is dat het niet heel flexibel is, een hoop objecten zijn al voor geprogrammeerd. Ook de performance van de applicatie is niet optimaal.

**- PHP,** nog een geschikte backend taal is PHP. Deze taal wordt ook veel gebruikt en kan gerund worden op diverse operating systems. Een nadeel van PHP is dat het niet geschikt is voor grote web-based applicaties.

**- JavaScript,** deze taal wordt voor zowel frontend als backend development gebruikt. Het heeft een grote community en wordt beschouwd als een van de top programmeertalen. Het maakt gebruik van light-weight scripts waardoor de performance goed is.

### 2.2.2 Gap analysis

Om vast te stellen welke backend taal het beste aansluit bij de eisen voor het project, is het belangrijk om eerst (nogmaals) de eisen te citeren. Deze zijn: mijn ervaring met de taal, een goede performance, integreerbaar met het .NET framework , makkelijk te gebruiken en welbekend (voor een goede overdraagbaarheid).

Binnen de talen uit [hoofdstuk 2.2.1](#_2.2.1_Available_product) vergeleken (op een schaal van 1 tot 5) met deze eisen.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Ervaring** | **Performance** | **.NET frame.** | **Makkelijk te gebruiken** | **Welbekend** | **Totaal** |
| **C#** | 5 | 4 | 5 | 4 | 5 | 23 |
| **Python** | 3 | 4 | 3 | 4 | 5 | 19 |
| **Java** | 5 | 4 | 2 | 3 | 5 | 19 |
| **Ruby** | 1 | 4 | 1 | 5 | 4 | 15 |
| **PHP** | 1 | 4 | 3 | 3 | 5 | 16 |
| **JavaScript** | 3 | 5 | 3 | 4 | 5 | 20 |

### 2.2.3 Conclusie subvraag 2

Bij het vergelijken van de verschillende eisen en talen, komt naar voren dat C# het beste aansluit bij de eisen voor een backend taal. Dit komt voornamelijk door de goede intergratie met het .NET framework.

# Hoofdvraag

## 3.1 Literature search

Om de hoofdvraag de ondersteunen. Zal binnen dit hoofdstuk een literature search gedaan worden naar de integratie van het object-oriented programming paradigm in combinatie met de taal C#.

Volgens verschillende bronnen is het goed mogelijk om in C# volgens het object-oriented programming paradigm te programmeren. Dit standpunt is goed te ondersteunen wanneer je de karakter eigenschappen van object-oriented programming bekijkt en deze vergelijkt met de mogelijkheden in C#:

**Abstraction modeling:** in C# is het mogelijk om een klasse aan te maken. Waarin de benodigde methoden en properties staan. Hierin kunnen alleen de benodigde methoden en properties getoond en gebruikt worden als dat nodig is.

**Encapsulation:** dit kenmerk lijkt erg op het eerste kenmerk. Echter heeft het een iets andere betekenis. Namelijk dat je bepaalde dingen alleen kan aanroepen door publieke functies. Ook dit is mogelijk in C#. Hierbij kunnen public methoden de private velden in een klas aanpassen.

**Inheritance:** dit kenmerk gaat over overerving, in C# kan je een nieuwe klasse creëren op basis van een bestaande klasse. Hiervoor wordt het symbool : gebruikt.

**Polymorphism:** ook dit kenmerk lijkt op de bovenstaande. Hier gaat het erom dat er inheritance plaatsvind maar dat deze klasse verschillende vormen aannemen. Ook dit is mogelijk met C#.

## 3.2 Subvraag onderbouwing

In [hoofdstuk 2.1.3](#_2.1.3_Conclusie_subvraag) staat waarom er gekozen is voor het programming paradigm.

In [hoofdstuk 2.2.3](#_2.2.3_Conclusie_subvraag) staat waarom er gekozen is voor de specifieke taal.

# Conclusie

De hoofdvraag wordt allereerst ondersteund door de twee subvragen. Zoals besproken ondersteund de eerste subvraag de keuze voor het programming paradigm ([hoofdstuk 2.1.3](#_2.1.3_Conclusie_subvraag)). Hier is gekozen voor een combinatie van de paradigms maar met een overheersende rol voor het object-oriented programming paradigm.

Hierna wordt de hoofdvraag ondersteund door de tweede subvraag over de keuze voor de taal ([hoofdstuk 2.2.3](#_2.2.3_Conclusie_subvraag)). Hierin komt duidelijk naar voren waarom er gekozen is voor C#.

Vervolgens wordt de hoofdvraag ondersteund door een literature search, die aantoont dat C# geschikt is om object-oriented te programmeren.

Door de documentatie omtrent C# en object-oriented programming te bestuderen (deze staat ook in de bronnen) zal ik deze kunnen toepassen in mijn applicatie voor dit semester.

# Bronvermelding

*Methods - ICT research methods*. (z.d.). HBO ICT. Geraadpleegd op 14 april 2022, van <https://ictresearchmethods.nl/Methods>

*Introduction of Programming Paradigms*. (2022, 24 maart). GeeksForGeeks. Geraadpleegd op 14 april 2022, van

<https://www.geeksforgeeks.org/introduction-of-programming-paradigms/>

Hari, S., & Kette, R. C. (z.d.). *Programming Paradigms: A must know for all Programmers*. Hackr.Io. Geraadpleegd op 14 april 2022, van

<https://hackr.io/blog/programming-paradigms>

Eastwood, B. (2022, 5 januari). *The 10 Most Popular Programming Languages to Learn in 2022*. Northeastern University Graduate Programs. Geraadpleegd op 14 april 2022, van

<https://www.northeastern.edu/graduate/blog/most-popular-programming-languages/>

Gallinelli, N. (2022, 9 februari). *Front End vs. Back End Development*. Flatiron School. Geraadpleegd op 20 april 2022, van

<https://flatironschool.com/blog/front-end-vs-back-end-development/#:%7E:text=Ruby%2C%20Python%2C%20and%20PHP%20are,nature%2C%20that’s%20not%20the%20case>.

GeeksforGeeks. (2021b, september 1). *Advantages and Disadvantages of PHP*. Geraadpleegd op 20 april 2022, van

<https://www.geeksforgeeks.org/advantages-and-disadvantages-of-php/>

B. (2022, 11 maart). *Object-Oriented Programming (C#)*. Microsoft Docs. Geraadpleegd op 20 april 2022, van

<https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/tutorials/oop>