Applicatie Onderzoek

MIJN JACHTVELD

De kracht van Mijn Jachtveld

Datum: 02-04-2024

School: Fontys University of Applied Sciences

Naam: Cédric Berden

Versie: 1.2

1e Assessor: Thijs Naus

2e Assessor: Jan Oonk

Stakeholder: Jaap Verhofstad

Inhoudsopgave

Introductie

In dit onderzoek gaat er gekeken worden hoe de applicatie in elkaar zit en hoe het beste een API-laag toegevoegd kan worden. Daarnaast wordt er ook gekeken of het een toevoeging is om OOP te gebruiken voor dit project en wat de voor- en nadelen daarvan zijn. Uiteindelijk laten we zien hoe ik het heb geïmplementeerd in het project.

Bestaande applicatie

Als ik naar de bestaande applicatie kijk van Mijn Jachtveld is het niet gemaakt met de OOP-standaarden zoals het bij ons op school geleerd wordt.

Hoewel het niet een ontwerppatroon volgt zoals MVC (Model-View-Controller) of een andere gangbare architectuur, lijkt het te zijn geschreven met een focus op het snel en makkelijk te maken. Er wordt gebruik gemaakt van SQL-query's om gegevens uit een database op te halen en HTML wordt gebruikt om de resulterende gegevens weer te geven.

Als je kijkt naar de programmeerstijl lijkt het te neigen naar een imperatieve stijl, waarbij de code stap voor stap wordt uitgevoerd en instructies worden gegeven over hoe de computer bepaalde taken moet uitvoeren. De code is gestructureerd rond functies die specifieke taken uitvoeren.

De applicatie werkt hetzelfde als een andere designmethode alleen om voor andere die verder werken in de applicatie kan het onduidelijk overkomen. Door te werken met OOP wordt er gebruik gemaakt van een vaste structuur. Met OOP maak je gebruik van classes en objecten waardoor de software beter gestructureerd wordt. Hierdoor is het voor iemand die nog niet met het project gewerkt heeft makkelijker om zich te kunnen implementeren in het project.

Bronnen

1. What is object-oriented programming? OOP explained in depth (24-01-2024) Erin Doherty. https://www.educative.io/blog/object-oriented-programming

Beste manier om een API-laag toe te voegen

Omdat de rest van de applicatie niet gemaakt is met OOP kunnen we niet makkelijk functies aanroepen die al gemaakt zijn. Hierdoor is het handiger om een aparte folder/laag aan te maken die los staat van de rest van de applicatie.

Het worden API's die Mijn Jachtveld uitbreiden voor derde en daarom wordt er gebruik gemaakt van de bestaande database.

Inrichten van de API-laag

De API-laag gaat op worden gesteld door gebruik te maken van OOP en een MVC-structuur aan te houden. Dit doe ik zodat het bedrijf kan zien hoe er op een andere manier een applicatie gemaakt kan worden.

Hier gaan we beschrijven wat OOP en MVC is en wat de voor- en nadelen ervan. Het is een veelgebruikte manier van programmeren.

OOP

OOP is een programmeerparadigma in de computerwetenschappen dat afhankelijk is van het concept van klassen en objecten.

OOP staat voor Object-Oriented Programming (Objectgeoriënteerd programmeren) en is een programmeerparadigma dat draait om het creëren van objecten die gegevens bevatten in de vorm van velden (ook wel eigenschappen genoemd) en methoden (functies die op die gegevens kunnen worden toegepast). Laten we eens kijken naar de belangrijkste concepten van OOP:

Klassen: Een klasse is een blauwdruk voor het creëren van objecten. Het definieert de eigenschappen en methoden die elk object van die klasse zal hebben. Bijvoorbeeld, als we een klasse "Auto" hebben, kunnen de eigenschappen zijn: merk, model, kleur, en de methoden kunnen zijn: starten, stoppen, rijden, enzovoort.

Objecten: Een object is een instantie van een klasse. Het is een entiteit die de eigenschappen en methoden van de klasse bezit. Als we de klasse "Auto" hebben, kan een object van die klasse bijvoorbeeld een Toyota Corolla zijn met een rode kleur.

Encapsulation (Inkapseling): Dit is het principe waarbij de interne werking van een object verborgen is voor de buitenwereld, en alleen de relevante informatie wordt blootgesteld via methoden. Dit helpt om de complexiteit te verminderen en de veiligheid van de gegevens te waarborgen.

Inheritance (Overerving): Dit is een mechanisme waarbij een nieuwe klasse kan worden gemaakt, gebaseerd op een bestaande klasse, en de eigenschappen en methoden van die bestaande klasse kan overnemen en uitbreiden. Bijvoorbeeld, als we een klasse "Vrachtwagen" hebben, kan deze erven van de klasse "Auto", maar extra eigenschappen en methoden hebben die specifiek zijn voor vrachtwagens.

Polymorfisme: Dit is het principe waarbij objecten van verschillende klassen zich op een uniforme manier kunnen gedragen. Het stelt ons in staat om methoden te definiëren in de basisklasse die kunnen worden overschreven in afgeleide klassen, waardoor verschillende klassen verschillende implementaties van dezelfde methode kunnen hebben.

OOP bevordert herbruikbaarheid, modulariteit en onderhoud baarheid van code, waardoor complexe systemen kunnen worden ontwikkeld en onderhouden met een hoger niveau van organisatie en structuur.

Voordelen van OOP:

Modulaire code: OOP stelt ontwikkelaars in staat om code te organiseren in modules (klassen en objecten), waardoor het gemakkelijker wordt om code te begrijpen en te onderhouden.

Herbruikbaarheid: Objectgeoriënteerde programma's bevorderen herbruikbaarheid van code door middel van concepten zoals overerving en polymorfisme.

Flexibiliteit: OOP maakt het mogelijk om code gemakkelijk aan te passen en uit te breiden door middel van overerving en polymorfisme.

Nadelen van OOP:

Steile leercurve: Voor ontwikkelaars die niet bekend zijn met OOP-concepten kan het leren en begrijpen ervan een uitdaging zijn.

Overhead: OOP kan leiden tot overhead in termen van geheugen- en prestatiegebruik vanwege de extra abstractielagen.

MVC:

MVC staat voor Model-View-Controller en is een ontwerppatroon dat vaak wordt gebruikt in softwareontwikkeling om de structuur van een applicatie te organiseren. Laten we eens kijken naar wat elke laag inhoudt:

Model: Dit is verantwoordelijk voor het beheren van de gegevens van de applicatie en de logica die ermee samenhangt. Het kan bijvoorbeeld queries bevatten om gegevens op te halen, validatie van invoer en andere bedrijfslogica. Het Model representeren de 'kennis' van de applicatie.

View: De View is verantwoordelijk voor het presenteren van de gegevens aan de gebruiker en het reageren op gebruikersinteracties. Dit kan een gebruikersinterface zijn, zoals een webpagina, een mobiele app-interface, of API’s. De View neemt gegevens van het Model en presenteert deze op een manier die begrijpelijk is voor de gebruiker.

Controller: De Controller is een tussenpersoon tussen Model en View. Het ontvangt gebruikersinvoer van de View, verwerkt deze en stuurt vervolgens instructies naar het Model om de nodige acties uit te voeren (bijv. gegevens bijwerken, ophalen, verwijderen) en bepaalt welke View moet worden weergegeven als reactie op de gebruikersinvoer. De Controller vertegenwoordigt de 'stroom' van de applicatie.

Door gebruik te maken van MVC-patroon worden de verschillende aspecten van een applicatie gescheiden, wat de code onder houdbaarheid en -schaalbaarheid verbetert. Het maakt het gemakkelijker om wijzigingen aan te brengen in één deel van de applicatie zonder dat dit invloed heeft op andere delen.

Voordelen van MVC:

Scheiding van zorgen: MVC scheidt de presentatie, de logica en de gegevens van een applicatie, waardoor de code gemakkelijker te begrijpen, te onderhouden en te testen is.

Herbruikbaarheid: Door de modulaire opzet van MVC kunnen componenten gemakkelijk worden hergebruikt in verschillende delen van een applicatie.

Gemakkelijk onderhoud: De scheiding van verantwoordelijkheden maakt het gemakkelijker om wijzigingen aan te brengen in één deel van de applicatie zonder de rest te beïnvloeden.

Nadelen van MVC:

Complexiteit: MVC kan in sommige gevallen complexiteit toevoegen aan een applicatie, vooral voor kleinere projecten waarvoor de structuur mogelijk overdreven is.

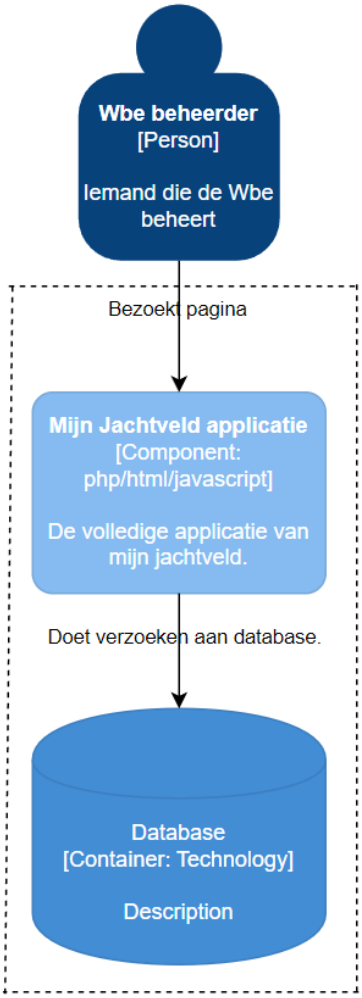
Leercurve: Net als bij OOP kan het begrijpen en toepassen van het MVC-patroon een uitdaging zijn voor ontwikkelaars die er niet bekend mee zijn.

Bronnen

1. What is object-oriented programming? OOP explained in depth (24-01-2024) Erin Doherty. https://www.educative.io/blog/object-oriented-programming
2. What are the advantages and disadvantages of object-oriented programming? (28-02-2024) https://www.linkedin.com/advice/1/what-advantages-disadvantages-object-oriented-k0nlf#:~:text=OOP%20allows%20programmers%20to%20create,memory%20overhead%2C%20and%20performance%20issues.
3. Model-view-controller (MVC) (01-09-2023) Robert Sheldon https://www.techtarget.com/whatis/definition/model-view-controller-MVC
4. What Is OOP (Object Oriented Programming)? Meaning, Concepts, and Benefits (04-09-2022) Chiradeep BasuMallick https://www.spiceworks.com/tech/devops/articles/object-oriented-programming/
5. The definitive guide to object-oriented programming in PHP (04-09-2023) Adebayo Adams https://www.honeybadger.io/blog/in-depth-guide-to-object-oriented-programming-in-php/

Architectuur applicatie

Zoals Mijn Jachtveld nu is gemaakt zit alles in een laag, dit willen we veranderen wanneer er gewerkt wordt met API's. Dit is dan ook het gedeelte waarover we hebben gesproken met MVC want zo gaan we de API en de backend ook inrichten. Hierdoor is de applicatie niet afhankelijk van elk gedeelte van het project. Om een voorbeeld te geven als de huidige applicatie gemaakt zou zijn op de MVC manier was het een stuk makkelijker geweest om de API's in te voegen in de applicatie. Dit is omdat de backend los staan van de huidige applicatie waardoor de backend hergebruikt kan worden en zo hoeft de hele applicatie niet herbouwd worden. Hieronder zie je een afbeelding van hoe de applicatie van Mijn Jachtveld is gemaakt.



Zo ziet Mijn Jachtveld er op het moment uit en nu laat ik zien hoe ik van plan hoe ik de API's wil gaan maken. Ik heb niet van elke class een aparte container gemaakt maar het in een container gedaan om de c4 diagram overzichtelijk te houden.