ONtwerpdocument

Mitch Kuijpers s27

Inhoudsopgave

**Inleidng1**

**Databaseontwerp2**

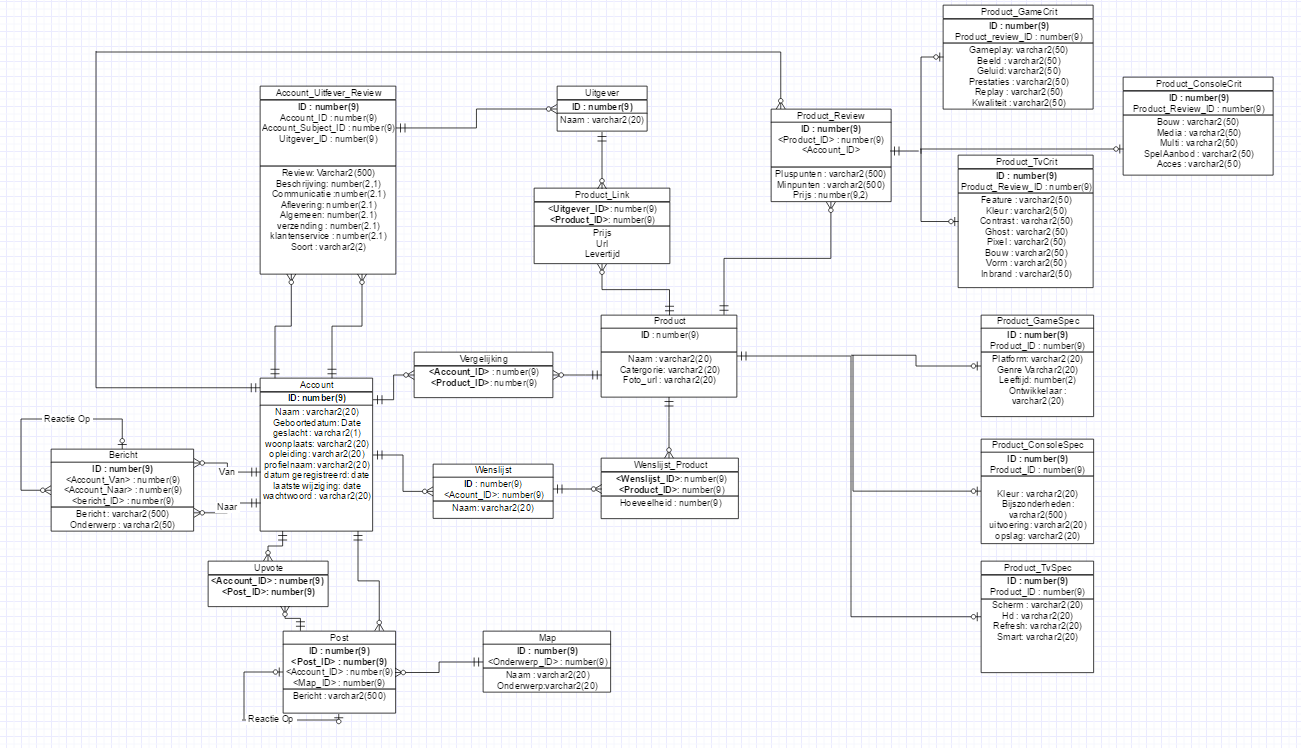
**Klassediagram3**

**Conclusie4**

Inleiding

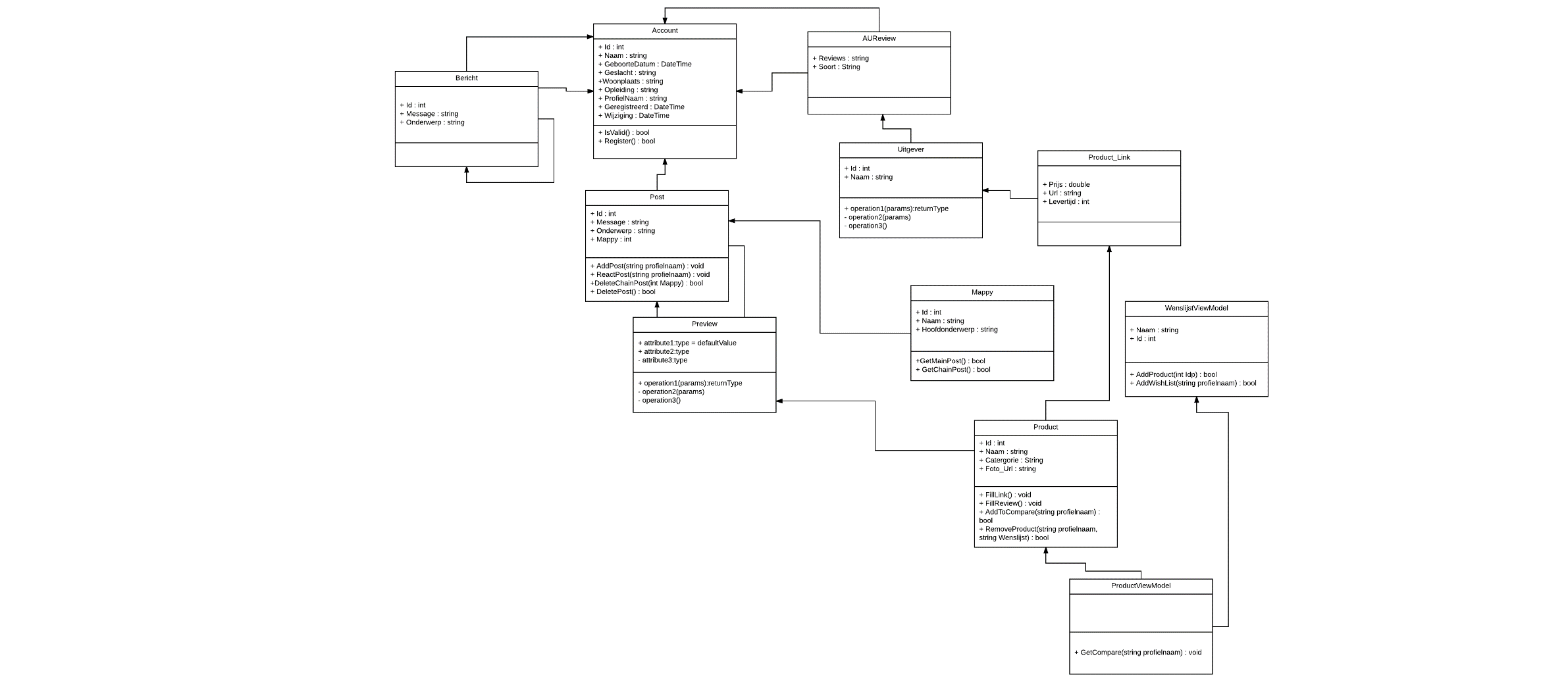
In dit ontwerpdocument staat het ontwerp voor mijn MVC opdracht. Ik gebruik de database die ik van de eerste opdracht van database heb moeten maken. Deze database recreëert de informatie in opgeslagen zou kunnen worden op de website “Tweakers.net”. Voor deze applicatie ga ik dus hun website proberen na te maken. Dit moet worden gedaan in MVC en is dus een totaal andere opzet dan een normale applicatie. Dit betekent dus dat ik mijn klasse diagram daar heel erg op moet aanpassen. Helaas weet ik voordat ik begin aan deze opdracht nog niet hoe het precies in elkaar zit, dus zal ik wat aanpassingen moeten doen onderweg.

Databaseontwerp



Het databaseontwerp spreekt grotendeels voor zichzelf. Het enigste wat wel op te merken is de verdeling van de reviews en de categorieën in de reviews en producten. Het is namelijk zo, dat ik voor alle categorieën een aparte tabel heb, deze tabel dient voor extra informatie zodat de hoofd tabellen niet overvol raken. Ik heb de categorieën dus onderverdeeld in 3 verschillende, Dit zijn: Games, Consoles en Tv’s. Verder heb ik de reviews voor producten gesplitst met die van een account of uitgever, omdat deze heel verschillend zijn. Ik denk er wel over na om de waardes uit de categorieën in een 2d string array te zetten, zodat dit veel makkelijk is om in te laden en te verwerken.

ClasseDiagram



In mijn klasse diagram heb ik voornamelijk ongeveer de relaties gebruikt uit mijn database. Aangezien we met erg directe models werken heb ik ook een viewmodel moeten toevoegen. Deze viewmodel is de ProductViewModel, Deze is voor het inladen van producten samen met de wenslijsten, aangezien ik de wenslijst nodig heb bij bijvoorbeeld het toevoegen van een product aan een wenslijst. Deze kan ik ook hergebruiken voor de connectie van 1 product aan een wenslijst, aangezien ik dan gewoon de eerst record van de lijst neem. Verder heb ik de koppel tabel gemaakt als een klas genaamd Product\_Link. Met deze klas geef ik de relatie tussen de meerdere uitgevers vaar een enkele product weer en word hier de specifieke informatie in opgeslagen die bij deze koppeling hoort. Verder zijn er weinig methodes, maar dit komt doordat er heel veel inladen en plaatsen zal plaatsvinden.

Ik heb het programma verdeeld in lagen. Wat je hier ziet zijn de models/de logica laag. Verder heb ik nog een interactie laag, dit is de layout die geschreven is in html and Css. Als derde laag heb ik een database klasse, hier zijn alle querries in opgeslagen. Deze klasse is verdeeld in regio’s voor elke controller. De reviews heb ik wel geïmplementeerd in het model, maar de functionaliteit van deze models/controller zal niet worden toegepast vanwege tijdsgebrek.

Conclusie

Na het implementeren van mijn ontwerp ben ik tegen heel wat dingen aangelopen. Het was namelijk veel te vaak het geval dat objecten in een object leeg bleven, het was af en toe handig als ik daadwerkelijk alles had moeten vullen. De volgende keer wanneer ik een MVC applicatie zou maken, dan zou ik veel vaker alleen het ID van het object waar een bepaalde model naar toe verwijst. En dan zou ik meerdere viewmodels kunnen gebruiken om de samenhang tussen de klassen te weergeven. Op deze manier zou ik elke keer het juiste object uit de database halen i.p.v. onhandig de variabelen terug posten die je nodig hebt van het object. Dit zorgde namelijk voor heel veel overbodige Code. Verder zijn er een paar onhandige dingen in mijn Database, zoals de categorieën. Dit had ik kunnen oplossen door een tabel te maken met heel veel entries. Het zou trager geweest zijn, maar wel veel meer tijd efficiënt. Ook heb ik gemerkt dat de database klasse heel erg vol raakt, de volgende keer ga ik deze misschien opsplitsen in verschillende klasse i.p.v. één enkele met verschillende regio’s. Uiteindelijk ben ik wel tevreden met het resultaat, ondanks dat mijn code niet heel robuust is.