**Mijn Reis Van C# Naar Java**

Graduaatsproef

Indy Baeté

A blue and black symbol

Description automatically generatedA logo of a company

Description automatically generatedA logo of a cup with a flame

Description automatically generated

# Inhoud

Contents

Voorpagina 1

[Inhoud 2](#_Toc153853213)

[Uitleg over de gekozen nieuwe technologie 3](#_Toc153853214)

[Beschrijving van mijn project / opdracht 4](#_Toc153853215)

[Hoe zit mijn project in elkaar / hoe ben ik tewerk gegaan 6](#_Toc153853216)

[Backend (Java): 6](#_Toc153853217)

[Frontend (React): 10](#_Toc153853218)

# Uitleg over de gekozen nieuwe technologie

Ik heb ervoor gekozen om Java te leren als nieuwe technologie. Java is een zeer bekende programmeertaal waarin ik altijd al geïnteresseerd was. Echter, mijn keuze voor Java komt niet alleen uit mijn interesse in de taal. De voornaamste reden voor deze keuze is mijn ambitie om verder te studeren voor mijn bachelor toegepaste informatica. In deze opleiding is Java de meest gebruikte programmeertaal.

Aangezien ik van plan ben om de bachelor te volgen via een versneld traject, dat is mogelijk dankzij mijn diploma graduaat programmeren. Zal ik vrijstellingen krijgen voor veel vakken waarin Java wordt toegepast. Het wordt echter van mij wel verwacht dat ik weet hoe ik met Java moet werken. Daarom ging ik normaal gezien deze zomer de taal Java toch al moeten leren. De opdracht van mijn graduaatsproef, waarin we een nieuwe taal/technologie moesten leren, bleek dus bijzonder goed van pas te komen. Dan moet ik deze zomer Java niet meer leren en kan ik mij focussen op een vakantiejob.

Dus ik heb Java geleerd waarbij ik heb gewerkt met Maven en het Spring Boot-framework. Daarnaast heb ik het Spring Data JPA-framework gebruikt voor het beheren van mijn database. Ik heb dus met meerder nieuwe technologieën gewerkt.

# Beschrijving van mijn project / opdracht

Ik heb een klanten en leden beheersysteem ontwikkeld voor een hotel. Hoewel de opdracht specifiek gericht was op het beheren van klanten en leden voor een hotel, is het mogelijk om dit beheer systeem licht aan te passen en te gebruiken voor andere applicaties waarbij je klanten kunt beheren. Mijn applicatie is ook volledig in het Engels geschreven.

De inspiratie voor deze opdracht kwam van een eerdere opdracht waarbij we een vergelijkbaar systeem moesten ontwikkelen, maar dan in C# met WPF als frontend. Mijn opdracht is niet exact hetzelfde. Maar het is wel heel erg gelijk.

Het klanten en leden beheersysteem is opgebouwd rondom de volgende functionaliteiten. Een lijst van klanten waarin je klanten kunt toevoegen, bijwerken, en verwijderen. Klanten hebben ook een lijst met leden onder zich waarin je leden kunt toevoegen, bijwerken, en verwijderen.

De gegevens en validatie van klanten en leden omvatten:

Een klant heeft een identificator, naam, e-mailadres, telefoonnummer en postadres (gemeente, postcode, straatnaam, huisnummer). Leden van een klant hebben een identificator, naam en geboortedatum.

Voor validatie heb ik het niet al te ingewikkeld gemaakt. Alle velden mogen niet leeg zijn, er moet een '@' in een e-mailadres zitten en een geboortedatum mag niet in de toekomst liggen. Degene die de klanten beheert, heeft dus redelijk veel vrijheid in welke data er ingevoerd wordt. Mocht ik extra tijd hebben gehad, dan had ik deze validatie strenger gemaakt, maar dit vond ik geen prioriteit.

Voor de implementatie van het project heb ik gebruik gemaakt van verschillende technologieën. Waaronder een backend geschreven in Java met Spring Boot en Maven, MySQL als database, en het Spring Data JPA-framework voor het beheren en communiceren tussen de database en de backend. Ik heb gewerkt met een 3-lagen model voor mijn backend, dit heb ik gedaan door alle lagen op te splitsen in verschillende packages. De frontend is ontwikkeld met React, waarbij ik Yarn heb gebruikt als package manager.

In tegenstelling tot de oorspronkelijke opdracht in C#, waar de backend rechtstreeks met de WPF-frontend communiceert, heb ik in mijn implementatie natuurlijk gebruik gemaakt van een REST API voor de communicatie tussen de backend en frontend. Ik vond dit zelf zeer interessant omdat ik graag wilde weten hoe je een REST API in Java kon ontwikkelen.

# Hoe zit mijn project in elkaar / hoe ben ik tewerk gegaan

## Backend (Java):

Voordat ik met de backend begon, heb ik me eerst beziggehouden met Java te leren. Dit ben ik beginnen doen sinds het begin van het semester, met wekelijks een aantal uurtjes hieraan te besteden. Gedurende deze tijd heb ik veel geleerd over Java, zoals Spring Boot, kennis over Maven, het werken met het Spring Data JPA-framework, en nog veel meer. Ik heb dit gedaan door het volgen van vele cursussen en het bekijken van vele YouTube tutorials. Daarnaast heb ik talloze kleine oefenprojectjes uitgevoerd om mijn kennis te verdiepen. Dit heb ik niet alleen gedaan voor Java te leren voor mijn project, maar ik wilde echt begrijpen wat ik aan het doen was. Dit was vooral belangrijk omdat ik deze taal heel goed moet kunnen voor mijn bachelor. In ongeveer twee maanden tijd heb ik ongeveer 35 uur besteed aan het leren van Java en het maken van oefenprojecten voordat ik begon aan de backend van mijn hotelproject. Deze voorbereiding heeft me enorm geholpen, omdat ik al een duidelijk idee had van hoe ik alles zou aanpakken voordat ik aan mijn backend begon.

Vervolgens ben ik gestart met het opzetten van mijn backend. Nadat ik mijn spring boot project had aangemaakt, ben ik begonnen met het toevoegen van alle benodigde dependencies aan mijn pom.xml-bestand. Ik heb de dependency voor Spring Data JPA toegevoegd, evenals een dependency zodat ik kan verbinden met mijn MySQL-database. Daarnaast heb ik een dependency toegevoegd waarmee ik, net zoals bij ASP.NET Core, gebruik kan maken van Swagger UI om een goed overzicht te krijgen van al mijn requests.

A screenshot of a computer program

Description automatically generatedDaarna heb ik het 3-lagenmodel opgezet in verschillende packages: datalaag, businesslaag en restlaag, zoals je kunt zien in de bijvoegende afbeelding. De klasse HotelProjectApplication is de opstartklasse van mijn spring boot project, waarin ik niets hoefde te wijzigen. De klasse CorsConfig heb ik gemaakt om mijn applicatie te configureren, zodat mijn frontend toestemming had om verzoeken naar mijn backend te sturen.

Daarna heb ik eigenlijk tegelijkertijd gewerkt aan mijn datalaag, businesslaag en restlaag. Maar ik zal eerst uitleggen hoe de datalaag in elkaar zit, aangezien ik hier in het begin het meest aan heb gewerkt. Daarna zal ik uitleggen hoe de businesslaag in elkaar zit en dan hoe de restlaag in elkaar zit.

A screenshot of a computer

Description automatically generatedOmdat ik gebruik heb gemaakt van het Spring Data JPA-framework, was de datalaag niet bijzonder moeilijk. In het begin was het wel uitdagend om alles te begrijpen, maar omdat ik dit al eerder had geleerd viel het uiteindelijk mee. Het Spring Data JPA-framework is eigenlijk vrij vergelijkbaar met Entity Framework in .NET. Zoals je kunt zien op de bijgevoegde foto, had ik slechts 2 klassen en 2 interfaces nodig voor de datalaag. -

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

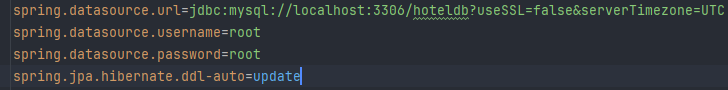
Description automatically generatedIn de CustomerEntity en MemberEntity klassen heb ik de klassen zelf en de vereiste properties gemarkeerd met annotaties. Op deze manier kan het Spring Data JPA-framework begrijpen hoe de database moet worden aangemaakt en hoe deze klassen moeten worden gekoppeld aan de database. Daarnaast heb ik ook de nodige constructors gemaakt en de nodige getters en setters in deze klassen geplaatst, zodat ik deze gegevens later kan mappen naar een domain model.

A screen shot of a computer program

Description automatically generatedDe JPA-repositories zijn interfaces die overerven van JpaRepository, zoals hieronder geïllustreerd. Bij het definiëren van de interface geef je de entity-klasse en het datatype van de primary key mee. Vervolgens kun je verschillende methoden uitvoeren op dit interface. In de businesslaag zul je later zien hoe dit wordt toegepast, omdat JPA al veel standaardmethoden biedt die je niet zelf hoeft te implementeren. Je hebt ook de mogelijkheid om aangepaste methoden te definiëren, zoals hieronder gedemonstreerd. Deze methoden worden gegenereerd volgens naming conventions. JPA begrijpt zelf wat er moet gebeuren dankzij deze naming conventions.

A screen shot of a computer program

Description automatically generated

Naast het definiëren van JPA-repositories, moest ik ook een database/schema aanmaken in MySQL en alle instellingen configureren in het application.properties bestand dat zich onder de resources map bevindt. Hierdoor kan er verbinding worden gemaakt met de aangemaakte database, en wanneer ik het Spring Boot-programma uitvoer, worden automatisch alle tabellen gegenereerd.

## Frontend (React):