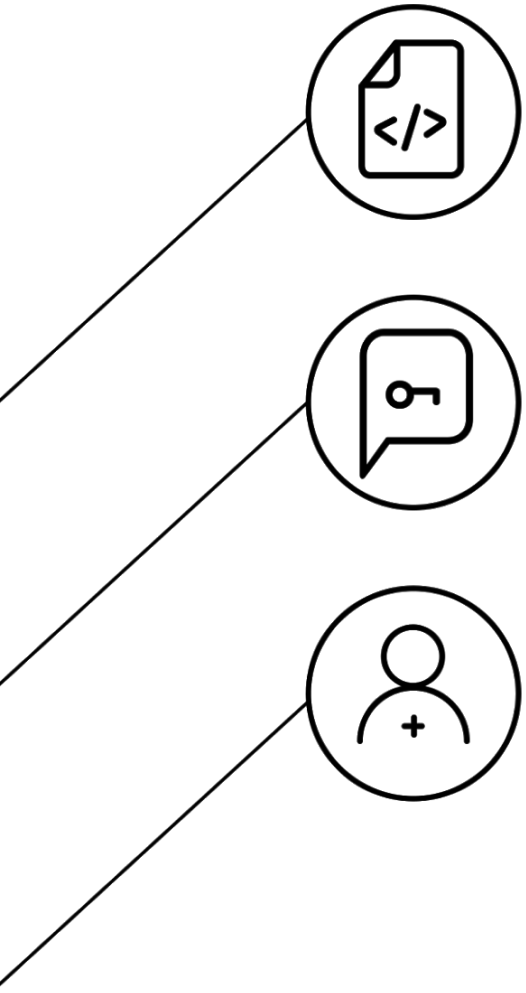

Integrale Eindopdracht

Hbo-bachelor ICT/Software Development

Leerlijn Back End Programming

Opdrachtbeschrijving en deelopdrachten

Randvoorwaarden, structuur en beoordeling

NOVI HOGESCHOOL

Inhoud

Integrale eindopdracht Back End Programming (30 EC) 3

Algemene opdrachtbeschrijving 3

Deelopdrachten 4

Voorbeeldcasus 4

Toelichting voorbeeldcasus en eisen aan de opdracht 5

Deelopdracht 1. Functioneel en Technisch Ontwerp 6

Deelopdracht 2. Software schrijven 6

Deelopdracht 3. Verantwoordingsdocument en installatiehandleiding 7

Randvoorwaarden 8

Structuur 8

Beoordelingscriteria 9

Integrale eindopdracht Back End Programming (30 EC)

In deze leerlijn leer je de backend van een applicatie uit te denken, te beschrijven en te programmeren in Java met behulp van Spring boot. Om deze leerlijn af te ronden dien je de volgende leeruitkomsten aan te tonen:

1. **Java Programmeren**
   De student programmeert in Java, waarbij hij OOP-structuren toepast. Hierbij past de student automatisch testen toe en beheert hij externe code met behulp van Maven, waardoor men in een team aan Java-projecten kan werken. (LU1)
2. **Software Development Process**
   De student stelt, op basis van de Software Development Life Cycle, technische documentatie op voor de backend van een applicatie, gebruik makend van UML-diagrammen. (LU2)
3. **Database Development**
   De student ontwerpt een relationele database, waarin data met onderlinge relaties veilig opgeslagen en uitgelezen kan worden, aan de hand van een technisch ontwerp document. Tevens beheert de student de data en rechten van databasegebruikers en voert hij CRUD-opdrachten uit op de database. (LU3)
4. **Java & Spring**
   De student zet een backend applicatie op met behulp van het Spring-boot framework en maakt gebruik van verschillende architecturale lagen binnen Spring. De student test zijn applicatie met unit-testen en het mocken van klassen en tevens communiceert de applicatie met een database. (LU4)
5. **Clean Code & Design Patterns**
   De student schrijft zijn code volgens de afgesproken conventies van Clean Code en ontwikkelt highly cohesive en loose coupled code, door de toepassing van Design Patterns en SOLID. (LU5)

Algemene opdrachtbeschrijving

Backend ontwikkelaars brengen gebruikers en systemen op steeds meer manieren met elkaar in verbinding. Gebruikers kunnen, chatten, producten verkopen, samen documenten schrijven, code delen en beheren en nog veel meer.

Voor deze eindopdracht ga jij een applicatie bedenken waarvan je alleen de backend gaat programmeren. Je bedenkt wat de gebruiker zou moeten kunnen met jouw applicatie, documenteert dit met behulp van UML in een technisch ontwerp en gaat hier vervolgens de backend code voor schrijven.

Jouw systeem heeft minimaal de volgende eigenschappen:

- Autorisatie en authenticatie;
- Communicatie met een relationele database middels CRUD-opdrachten;
- Een rest endpoint die data uit verschillende tabellen combineert en terugkoppelt;
- RESTful webservice;
- Er moeten bestanden geüpload en gedownload kunnen worden;
- Jouw code wordt getest met behulp van unit-testen en mocken;
- Jouw applicatie vult een database met relevante testdata, zodat de applicatie getest kan worden.

Om de leerlijn Back End Programming met succes af te ronden is een voldoende nodig voor de integrale opdracht; met de deelopdrachten kunnen geen losse cursussen worden afgerond.

Op te leveren producten:

- De documentatie en applicatie lever je in een zip-bestand in van maximaal 50mb. Meer informatie hierover vind je in het hoofdstuk Randvoorwaarden.

Deelopdrachten

Voor de uitvoering van de eindopdracht mag je zelf een casus bedenken, dit is echter geen vereiste. Hieronder bieden we je een voorbeeldcasus aan, je kunt ervoor kiezen om alle opdrachten aan de hand van dit voorbeeld uitwerken. Ook wanneer je een eigen casus gebruikt kun je je voordeel doen met dit voorbeeld; het geeft namelijk duidelijk weer hoe je alle elementen terug kunt laten komen zodat jouw applicatie voldoet aan alle voorwaarden.

Voorbeeldcasus

Voor mijn eindopdracht ga ik het backendsysteem van een autogarage programmeren. In deze garage komen klanten hun auto afleveren voor een reparatie. Een administratief medewerker voegt de klant en de auto toe aan het systeem, wanneer de klant en of de auto voor het eerst bij de garage komen. De medewerker plant vervolgens een moment in om de auto te keuren. Tijdens deze registratie kunnen de autopapieren in pdf-formaat toegevoegd worden.

Een monteur keurt vervolgens de auto en voegt de gevonden tekortkomingen toe aan de auto in het systeem. Nadat de auto gekeurd is, neemt de monteur contact op met de klant. Gaat de klant akkoord met de reparatie, dan maakt de monteur een afspraak om de auto te repareren.

Gaat de klant niet akkoord met de reparatie dan zet de monteur dat in het systeem, maakt hij de bon op voor de keuring, à 45 euro, en kan de klant de auto komen ophalen en wordt de reparatie op 'niet uitvoeren' gezet.

Wanneer de klant akkoord gaat, voegt de monteur toe wat afgesproken is en gaat hij de auto repareren. Elk onderdeel dat gebruikt wordt, wordt toegevoegd aan de reparatie. Ook wordt elke handeling gedocumenteerd. Vervangt de monteur bijvoorbeeld de remschijf dan wordt het onderdeel remschijf aan de reparatie toegevoegd en wordt de handeling remschijf vervangen aan de reparatie toegevoegd.

Al deze onderdelen en handelingen staan al, inclusief prijs, in het systeem. De monteur kan deze opgeslagen handelingen en onderdelen selecteren. Omdat een monteur soms iets specifieks moet doen, kan de monteur ook een 'overige' handeling en prijs toevoegen.

Wanneer de reparatie voltooid is, wordt de reparatie op voltooid gezet en kan de klant opgebeld worden. De klant wordt opgebeld door een administratief medewerker. Deze medewerker kan een lijst opvragen met te bellen klanten waarvan de reparatie voltooid is of de status 'niet uitvoeren' is.

Wanneer de klant de auto komt ophalen zal een kassamedewerker de bon laten genereren door het systeem. De bon bevat de keuring + bedrag, de handelingen + bedrag en de onderdelen + bedrag. Bij alle bedragen moet het BTW tarief nog berekend worden voordat de bedragen op de bon getoond worden. Wanneer de klant betaald heeft, wordt de status op betaald gezet.

Daarnaast is er een backoffice medewerker die onderdelen (naam, prijs, voorraad) kan toevoegen aan het systeem, voorraden kan aanpassen en handelingen (naam, prijs) kan toevoegen aan het systeem. Alle prijzen in het systeem zijn exclusief BTW.

Toelichting voorbeeldcasus en eisen aan de opdracht

In bovenstaande casus heeft de student verschillende user-rollen genoemd. Hij toont aan dat de applicatie data moet kunnen opslaan in & kunnen ophalen uit de database en dat er bestanden geüpload kunnen worden. De opdracht voldoet dus aan de eerder genoemde minimale eisen die aan de applicatie gesteld worden.

De verschillende user rollen die genoemd zijn, zijn administratief medewerker, monteur, kassamedewerker en backoffice medewerker. Deze rollen hebben allemaal taken die alleen zij kunnen uitvoeren. In andere woorden, deze rollen geven de gebruiker autorisatie om bepaalde handelingen met het systeem uit te voeren.

Het toevoegen van klanten en auto's, het toevoegen van onderdelen & handelingen aan reparaties en het aanpassen van de reparatiestatus zijn allemaal voorbeelden van CRUD-operaties. Het opmaken van de bon is een voorbeeld van het combineren van data uit verschillende tabellen.

De student heeft dus een casus beschreven die voldoet aan de gestelde functionele eisen.

Deelopdracht 1. Functioneel en Technisch Ontwerp

Je gaat een webapplicatie bouwen aan de hand van bijgeleverde casus (de autogarage) of omdat je zelf een goed idee hebt voor een webapplicatie. Voordat je aan de slag kunt met het ontwikkelen van de applicatie moet eerst nog het één en ander worden uitgezocht. In deze deelopdracht ga je informatie verzamelen en achterhalen wat de behoefte is voor een nieuwe applicatie. De requirements moeten worden vastgesteld. Nadat duidelijk is wat de wensen zijn ga je aan de slag met het ontwerpen van de te bouwen webapplicatie. Je ontwikkelt een functioneel ontwerp voor de webapplicatie. Het functioneel ontwerp moet te begrijpen zijn door iemand zonder IT-achtergrond. Met het functioneel ontwerp laat je zien dat de wensen van de opdrachtgever verwerkt zijn in het ontwerp.

Op basis van het functioneel ontwerp maak je een vertaling naar een technisch ontwerp. Het zwaartepunt van deze deelopdracht ligt bij het technisch ontwerp. Uit het technisch ontwerp is eenvoudig te lezen wat ontwikkeld en opgeleverd gaat worden. Je maakt tenminste een use-case diagram en een klassendiagram. Stel onderstaande diagrammen volgens geldende richtlijnen op en maak correct gebruik van relevante technieken.

Tip: Het is verstandig om al tijdens het maken van de eerste deelopdracht notities te maken voor het verantwoordingsdocument dat je voor de derde deelopdracht gaat maken.

Op te leveren:

- Functioneel Ontwerp met:
    - Requirements.
- Technisch Ontwerp met de volgende UML diagrammen:
    - Klassendiagram van alle entiteiten
    - Minimaal twee volledig uitgewerkte sequentiediagrammen

Deelopdracht 2. Software schrijven

Je hebt zojuist de ontwerpfase afgerond, het is nu tijd om de webapplicatie te bouwen! Je kunt beginnen met het uitschrijven van de requirements en daarna doorgaan met het technische ontwerp. De webapplicatie bestaat alleen uit geschreven code (backend). Je maakt gebruik van Java voor het ontwikkelen van de backend van de webapplicatie. De wijze waarop de applicatie gebouwd wordt sluit aan bij de wensen en eisen die in het functioneel en technisch ontwerp zijn opgenomen. De input voor de te bouwen applicatie staat beschreven in je documentatie uit de ontwerpfase inclusief het prototype. De webapplicatie die je bouwt moet een multi-tier applicatie zijn. Je werkt individueel aan het bouwen van de webapplicatie.

De webapplicatie moet minimaal voldoen aan onderstaande eisen:

- De backend en de database zijn gescheiden en kunnen los van elkaar op verschillende systemen draaien.
- De backend is volgens OOP ontwikkeld met behulp van Java & Spring.
- De database is relationeel.
- Je hebt gebruik gemaakt van versiebeheer (GIT)
- Je applicatie maakt gebruik van een build system (maven en npm)
- Je applicatie bevat unit-testen
- Je applicatie is beveiligd en bevat meerdere userrollen

Op te leveren

- Broncode, zie randvoorwaarden voor gedetailleerde informatie

Deelopdracht 3. Verantwoordingsdocument en installatiehandleiding

In de opleveringsfase ga je aan de verder met je verantwoordingsdocument. Als programmeur ben je naast het schrijven van je code ook verantwoordelijk voor de overdracht naar functioneel beheerders en andere programmeurs. Je gaat dan ook een installatiehandleiding schrijven om de installatie van jouw applicatie soepel te laten verlopen en een verantwoordingsdocument waarin jij uitlegt welke programmeerkeuzes je hebt gemaakt.

Je schrijft een verantwoordingsdocument waarin je een overzicht maakt van alle toegepaste technieken. Je vermeldt bij iedere techniek op welk regelnummer van de code dit gevonden kan worden en je onderbouwt waarom deze techniek toegepast is.

Daarnaast schrijf je een installatiehandleiding die uitlegt hoe de applicatie geïnstalleerd kan worden en hoe deze gebruikt kan worden. Je neemt hierin een lijst op van benodigdheden om de applicatie te kunnen runnen, een lijst met (test)gebruikers en userrollen, een lijst van rest endpoints (inclusief JSON-voorbeelden) en op welke manier deze beveiligd zijn. Ook voeg je een stappenplan met installatie instructies toe.

Op te leveren

- Verantwoordingsdocument en installatiehandleiding

Randvoorwaarden

Hieronder vind je een aantal randvoorwaarden waaraan je eindproduct moet voldoen. Deze randvoorwaarden hebben een verplicht karakter.

Je eindproduct dient ingeleverd te worden in een zip-bestand van maximaal 50mb. Dit zip-bestand bevat de volgende elementen:

- Installatiehandleiding (PDF of markdown)
    - Op basis van de installatiehandleiding moet de applicatie binnen 30 minuten opgestart kunnen worden
- Technisch ontwerp (PDF of markdown)
- Broncode
    - Inclusief link naar Git-repository
- Verantwoordingsdocument (PDF of markdown)

Jouw applicatie is ontwikkeld met behulp van:

- Java (alleen Java wordt geaccepteerd als programmeertaal, tenzij je schriftelijk akkoord hebt van de betreffende docent)
- het Spring-boot framework
- Maven, als dependency manager

Structuur

De installatiehandleiding en het technische ontwerp zijn beide voorzien van een inhoudsopgave en inleiding. Ze zijn verzorgd en hebben een titelblad met datum en naam van de auteur. De documenten bevinden zich ook op een logische plaats in het inleverbestand.

In de documenten (installatiehandleiding en het technische ontwerp) zitten geen verwijzingen naar afbeeldingen, diagrammen of modellen buiten het document zelf. Ook zijn de diagrammen en afbeeldingen goed leesbaar en tekstueel uitgelegd of beschreven. Een goed uitgangspunt hier is wanneer het document geprint wordt, deze nog steeds volledig beoordeeld kan worden.

Beoordelingscriteria

De eindopdracht wordt beoordeeld op basis van de volgende beoordelingscriteria. Per criterium kent de beoordelaar een aantal punten toe. Hierbij wordt zowel gekeken naar de feitelijke realisatie van de leeruitkomst als naar het door de student getoonde inzicht in de bijbehorende theorie.

# Deelopdracht	Leeruitkomsten en beoordelingscriteria	Weging	Score in cijfers van 1 t/m 10	Weging maal score
1. Functioneel en Technisch Ontwerp	**Bevat aspecten van de leeruitkomsten van de cursussen Software Development Process (LU2), Database Development (LU3) en Java & Spring (LU4).**	**25%**		
Criterium 1.1	De student ontleedt de juiste systeemeisen uit een (eigen) opdrachtbeschrijving of bestaande online applicatie en verwerkt deze in een correct uitgewerkte requirements (LU2)	5%		
Criterium 1.2	De student legt in het technisch ontwerp op logische en correcte wijze alle verschillende architecturale lagen op juiste wijze vast in het sequentiediagram en levert deze aan volgens de principes van de Software Development Life Cycle (LU2 en LU4)	10%		
Criterium 1.3	De Student structureert de ontworpen functionaliteiten in een klassendiagram, waarbij hij rekening houdt met het vertalen van het klassendiagram naar een databasemodel (RRM) door Spring. (LU2 en LU3)	10 %		
Feedback door de docent				
2. Software schrijven	**Betreft aspecten van de leeruitkomsten van alle cursussen van deze leerlijn.**	**65%**		
Criterium 2.1	De student maakt Java applicaties waarbij hij op de juiste momenten, afhankelijk van de complexiteit van het op te lossen probleem, OOP-structuren gebruikt zoals overerving, interfaces en abstracte klassen. (LU1)	15%		
Criterium 2.2	De student test geschreven klassen individueel met unittests die gebruik maken van de drie A's, waarbij de test coverage minimaal 50% is exclusief getters en setters. (LU1)	10%		
Criterium 2.3	De student past de principes Maven build lifecycle toe bij het beheren van externe code en zijn libraries (LU1)	5%		
Criterium 2.4	De student beheert zijn code met gebruik van Git om met versiebeheer de voortgang van het project vast te leggen. (LU2)	5%		
Criterium 2.5	De student leest en bewerkt data met behulp van SQL en draagt zorg voor de autorisatie en authenticatie in de database. (LU3)	5%		

Criterium 2.6	De student past autorisatie en authenticatie toe met behulp van Spring Security (LU4)	5%		
Criterium 2.7	De student voert zijn integratie-testen uit met behulp van mocken. (LU4)	5%		
Criterium 2.8	De student gebruikt HTTP-methods om de vertaalslag te maken naar acties met de data. (LU4)	5%		
Criterium 2.9	De student schrijft Java-code op basis van Clean Code, Design Patterns en SOLID. (LU5)	10%		
Feedback door de docent				
3. Verantwoordingsdocument en Installatiehandleiding	**Betreft aspecten van de leeruitkomst van de cursus Software Development Process. (LU2)**	**10%**		
Criterium 3.1	De student schrijft een duidelijk geschreven installatiehandleiding waarmee de applicatie door derden in een andere omgeving kan worden geïnstalleerd, voorzien van stappenplan, lijst van benodigdheden, testgebruikers, userrollen en rest-endpoints. (LU2)	5%		
Criterium 3.2	De student schrijft een volledig, goed gestructureerd en duidelijk geschreven verantwoordingsdocument waarin hij een beargumenteerd overzicht geeft van de toegepaste technieken. (LU2)	5%		
Feedback door de docent		**Totaal 100%**		