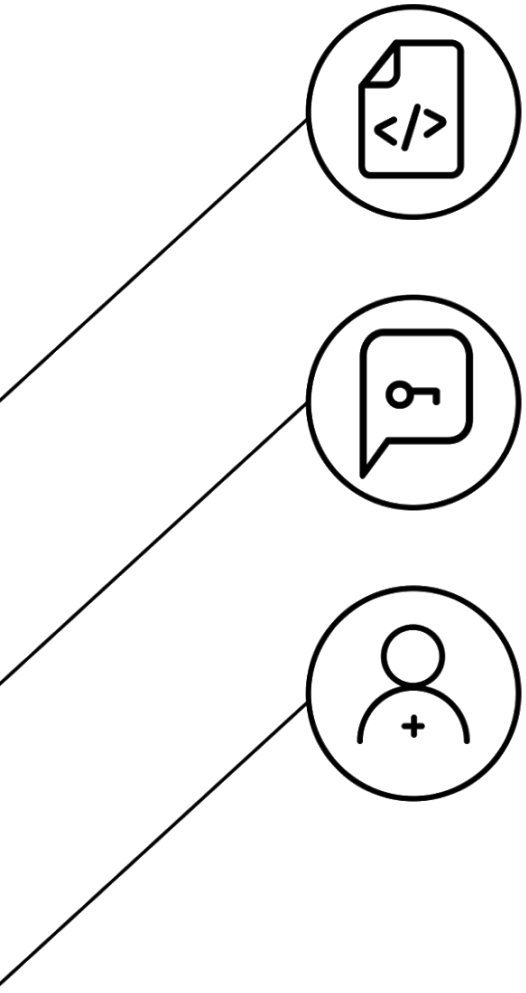

Integrale Eindopdracht

Hbo-bachelor ICT/Software Development

Leerlijn Backend

Opdrachtbeschrijving en deelopdrachten

Randvoorwaarden, structuur en beoordeling

NOVI HOGESCHOOL

Inhoud

Integrale eindopdracht Backend 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 te realiseren:

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. **Backend Documentatie**
   De student stelt, op basis van de Software Development Life Cycle, technische documentatie op voor de backend van een applicatie, gebruikmakend 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. **Spring Boot**
   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. 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.

Minimale randvoorwaarden aan het systeem:

- Juiste toepassing van autorisatie en authenticatie;
- Mogelijkheid voor communicatie met een relationele database middels CRUD-opdrachten;
- Jouw applicatie is een RESTful webservice die via endpoints data uit verschillende tabellen beheert en terugkoppelt.
- Er moeten bestanden geüpload en gedownload kunnen worden;
- Gebruikte code wordt getest met behulp van unit-testen en mocken;
- Mogelijkheid voor applicatie testing op basis van relevante testdata.

Om de leerlijn Backend met succes af te ronden, is een voldoende nodig voor de integrale eindopdracht. 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 uit te werken. 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 backend systeem 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, 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 worden 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 reparatiehandelingen (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 de database, data moet kunnen ophalen uit de database en dat er bestanden geüpload kunnen worden. De opdracht voldoet dus aan de eerdergenoemde 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 en 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. Technisch Ontwerp

Je gaat de backend van een webapplicatie bouwen aan de hand van bijgeleverde casus (de autogarage) of jouw eigen idee voor een webapplicatie. Voordat je aan de slag kunt met het ontwikkelen van de applicatie moet eerst nog het één en ander worden uitgezocht. De systeemeisen moeten worden vastgesteld aan de hand van de verschillende gebruikersrollen (user-rollen). Nadat duidelijk is wat de wensen zijn, ga je aan de slag met het bouwen van de backend van de webapplicatie.

Uit het technisch ontwerp is eenvoudig te lezen wat ontwikkeld en opgeleverd gaat worden. Je maakt tenminste twee sequentiediagrammen en een klassendiagram. Stel deze diagrammen volgens geldende richtlijnen op en maak correct gebruik van relevante technieken.

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

Op te leveren:

- Technisch ontwerp met daarin:
  - Korte beschrijving van deze opdracht met de functionaliteiten, verwoord in *user stories* of functionele eisen.
  - Klassendiagram van alle entiteiten.
  - Minimaal twee volledig uitgewerkte sequentiediagrammen.

Deelopdracht 2. Software schrijven

Het is nu tijd om de webapplicatie te bouwen! De webapplicatie bestaat alleen uit geschreven code (backend). Je maakt gebruik van Java en Spring Boot 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 technisch ontwerp zijn opgenomen, aan de hand hiervan bouw je backend applicatie. 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);
- Maakt gebruik van een build system (maven);
- Bevat unit-testen;
- Is beveiligd en bevat minstens twee user-rollen.

Op te leveren:

- Projectmap met broncode, zie randvoorwaarden voor gedetailleerde informatie.

Deelopdracht 3. Verantwoordingsdocument en installatiehandleiding

Zodra je begint met programmeren, ga je ook beginnen aan het verantwoordingsdocument. Hierin leg je vast welke technische ontwerpbeslissingen je maakt en *waarom* je deze keuzes gemaakt hebt. Dit werkt het beste als je dit al tijdens het ontwikkelen van jouw product begint te maken. Waarom heb je deze specifieke Java-library gebruikt en niet een andere? Waarom ben je van conventies of architecturale richtlijnen afgeweken? Welke doorontwikkelingen zijn er mogelijk of misschien zelfs wenselijk, en waarom heb je deze zelf niet door kunnen voeren? Heb je bijvoorbeeld iets achterwege gelaten in verband met een tekort aan tijd? Leg dan uit wat je liever had willen doen als je meer tijd had gehad. Reflecteer hierbij ook op je eigen leerproces: wat ging goed en wat kan de volgende keer beter? Let op: technieken die als randvoorwaarden gesteld zijn in de eindopdracht tellen niet mee.

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 user-rollen, een lijst van REST-endpoints (inclusief voorbeelden van de JSON-response). Deze voorbeelden moeten uitgeschreven zijn zoals in Postman zodat je ze kunt selecteren, kopiëren en plakken. Daarnaast moeten de endpoints beveiligd zijn. Leg ook uit op welke manier ze beveiligd zijn. Ook voeg je een stappenplan met installatie instructies toe. Dit stappenplan moet te volgen zijn voor een collega-developer die nog nooit met Java of SpringBoot heeft gewerkt.

Op te leveren:

- Verantwoordingsdocument in PDF (.pdf) of mark down (.md) met daarin:
    - Minimaal 10 beargumenteerde technische keuzes.
    - Een beschrijving van limitaties van de applicatie en beargumentatie van mogelijke doorontwikkelingen.
- Installatiehandleiding in PDF (.pdf) of mark down (.md).

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 10 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 met Long term support, bijvoorbeeld Java 11 of Java 17. (Alleen Java wordt geaccepteerd als programmeertaal, tenzij je schriftelijk akkoord hebt van de betreffende docent voor een andere programmeertaal.)
- het Spring Boot framework.
- Maven, als dependency manager.

Jouw applicatie voldoet aan de volgende eisen:

- De applicatie crasht niet en kan voldoende omgaan met exceptions.
- Jouw project is geüpload naar GitHub, deze is publiekelijk toegankelijk en de repository bevat geen .iml bestand, .idea- out- of targetmappen.

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 dat wanneer het document geprint wordt, het 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 bepaald in hoeverre je de betreffende leeruitkomst aantoonbaar hebt gerealiseerd.

# 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 Backend Documentatie (LU2) en Spring Boot (LU4).**	**25%**		
Criterium 1.1	De student ontleedt de juiste systeemeisen uit een (eigen) opdrachtbeschrijving of bestaande online applicatie. Hij beschrijft kort wat de applicatie inhoudt en formuleert dit in user stories. LU2)	5%		
Criterium 1.2	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. Het klassendiagram is taal- en platformonafhankelijk en hoeft geen methodes te bevatten. (LU2 en LU4)	10 %		
Criterium 1.3	De student legt in het technisch ontwerp op logische en correcte wijze alle verschillende architecturale lagen op juiste wijze vast in de twee sequentiediagrammen en levert deze aan volgens de principes van de Software Development Life Cycle. Deze sequentiediagrammen bevatten de juiste klasse- en methode namen. (LU2 en LU4)	10%		
2. Software schrijven	**Betreft aspecten van de leeruitkomsten van alle cursussen van deze leerlijn.**	**65%**		
Criterium 2.1	De student levert een project op dat voldoet aan de eisen die omschreven zijn onder 'Algemene opdrachtomschrijving.'	10%		
Criterium 2.2	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 attributen, methoden, overerving, interfaces en abstracte klassen. (LU1)	10%		
Criterium 2.3	De student voert integratie-tests uit door de application context van zijn applicatie te testen.	10%		

	Hiervoor gebruikt hij Spring Boot test en WebMvc. Tevens voert de student unittests uit die gebruikmaken van de drie A's, waarbij de test coverage minimaal 50% is exclusief getters en setters). (LU1 en LU4)			
Criterium 2.4	De student past de principes Maven build lifecycle toe bij het beheren van externe code en zijn libraries. (LU1)	5%		
Criterium 2.5	De student beheert zijn code met gebruik van Git om met versiebeheer de voortgang van het project vast te leggen. De student heeft kleine commits, maakt pull requests en mergt regelmatig. (LU2)	5%		
Criterium 2.6	De student leest en bewerkt data met behulp van SQL, JPA en Hibernate, maakt gebruik van Spring Boot ORM om gegevens uit een database te halen en draagt zorg voor de autorisatie en authenticatie in de database. (LU3 en LU4)	5%		
Criterium 2.7	De student past autorisatie en authenticatie toe met behulp van Spring Security. (LU3)	5%		
Criterium 2.8	De student implementeert een webservice volgens de RESTful richtlijnen, waarbij hij HTTP-methodes gebruikt om de vertaalslag te maken naar acties met de data. (LU3)	5%		
Criterium 2.9	De student schrijft Java-code op basis van Clean Code, Design Patterns en SOLID. (LU5)	10%		
3. Verantwoordingsdocument en Installatiehandleiding	**Betreft aspecten van de leeruitkomst van de cursus Backend Documentatie. (LU2)**	**10%**		
Criterium 3.1	De student schrijft een duidelijke 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 verantwoordingsdocument waarin hij een beargumenteerd overzicht geeft van de toegepaste technieken. (LU2)	5%		
		Totaal 100%		