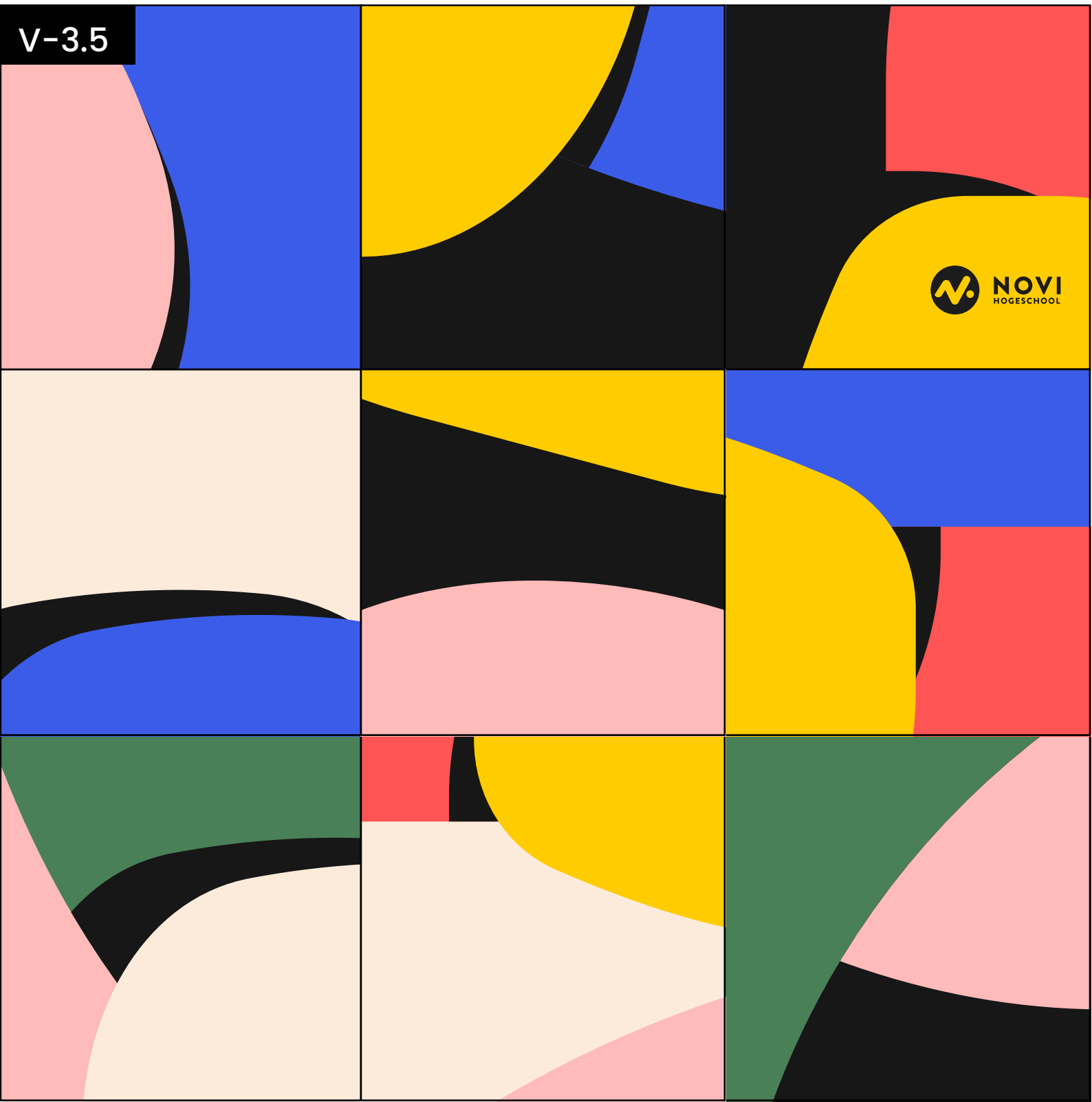
Eindopdracht

Leerlijn Backend (30 EC)



Inhoudsopgave

Eindopdracht Backend

Integrale eindopdracht

De leerlijn Backend bevat de cursussen programmeren met Java, Software design & documentatie en Spring Boot.

Om deze leerlijn af te ronden dien je de volgende leeruitkomsten aan te tonen:

1. **Programmeren met Java**
   De student ontwikkelt een functionele, gestructureerde en uitbreidbare Java web-API, waarbij test taken worden geautomatiseerd en het project wordt beheert door middel van version control.
2. **Software design & Documentatie**
   De student richt de relationele database in op basis van de user stories, beheert de data met SQL manipulatie en stelt technische documentatie op voor de backend en de database van de web-API, ondersteund door UML-diagrammen.
3. **Spring Boot**
   De student bouwt een beveiligde REST web-API met behulp van het Spring Boot Framework en maakt gebruik van automated testing.

Algemene opdrachtbeschrijving

Dagelijks gebruiken we online applicaties zoals Microsoft Teams, Google Drive en YouTube. Al deze applicaties zijn voorzien van zowel een frontend als een backend. Bij backend development ligt de focus op dataopslag, server-side logica en beveiliging. Je ontwikkelt het hart van de applicatie en bent daarmee verantwoordelijk voor de verwerking, bewerking en opslag van data in de applicatie.

Opdracht: je gaat een web-API bedenken en bouwen waarvan je alleen de backend gaat programmeren. Je bedenkt welke behoefte jouw product gaat vervullen en aan welke eisen de web-API moet voldoen. Hiervoor ga je eerst een plan schrijven en teken je de architectuur uit met behulp van UML. Daarna ga je de backend code schrijven.

Door deze opdracht uit te voeren, toon je aan een volwaardige web-API te kunnen bouwen in de backend. In het volgende hoofdstuk vind je meer informatie over de deelopdrachten.

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

- Technisch ontwerp met daarin o.a. de probleembeschrijving, de user stories, functionele- en niet-functionele-eisen, klassendiagram en sequentiediagrammen;
- Verantwoordingsdocument;
- De broncode van een Spring Boot web-API;
- Zelfgeschreven installatiehandleiding;

Deze producten worden ingeleverd als één ZIP-bestand.

Deelopdrachten

We gaan jouw vaardigheden als backend ontwikkelaar toetsen door je een beveiligde REST web-API te laten bouwen. Door verschillende CRUD-opdrachten in te richten, communiceert de web-API met de relationele database om data zowel op te slaan als op te halen. Omdat jouw web-API gebruikersaccounts ondersteunt met verschillende rollen en rechten, zorg jij ook voor authenticatie en autorisatie door jouw endpoints goed te beveiligen.

Voordat je begint met programmeren, maak je een gestructureerd plan van aanpak. Tijdens het ontwikkelen documenteer je jouw keuzes. Al deze stappen zijn opgedeeld in deelopdrachten.

Voordat je kunt starten met de eerste deelopdracht lever je eerst jouw casus ter goedkeuring in bij de docent. Je mag ervoor kiezen om de wensen van de voorbeeldcasus te vertalen naar een idee voor jouw web-API, of om een eigen casus te bedenken. In dit geval zorg je dat je de voorbeeldcasus goed bestudeerd, zodat je weet hoe je alle elementen terug kunt laten komen in jouw eigen casus.

Je beschrijft in maximaal 250 woorden:

- Welke behoefte je wil vervullen met deze web-API/welke functie de web-API heeft;
- Welke gebruikersrollen je wilt gebruiken (minimaal 2, maximaal 3);
- Welke data elementen je wilt kunnen opslaan (minimaal 6);
- Wat de 3 kern-functionaliteiten van jouw web-API zullen zijn. In minimaal één van deze functionaliteiten is het up- en downloaden van bestanden verwerkt.

Na goedkeuring van de docent over jouw idee voor een web-API, kun je aan de slag met de eerste deelopdracht.

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 (45 euro) op voor de keuring, kan de klant de auto komen ophalen en wordt de reparatie op 'niet uitvoeren' gezet.

Wanneer de klant akkoord gaat, voegt de monteur toe aan de bon wat afgesproken is en gaat hij de auto repareren. Elk gebruikt onderdeel en handeling worden toegevoegd aan de reparatie. 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 hoeft deze opgeslagen handelingen en onderdelen alleen maar te selecteren. Omdat een monteur soms iets specifieks moet doen, kan de monteur ook een 'overige' handeling en prijs toevoegen.

Wanneer de klant de auto komt ophalen, zal een kassamedewerker door het systeem een bon laten genereren met alle onderdelen en handelingen die door de monteur zijn toegevoegd. 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 web-API 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 minimale eisen die aan de web-API gesteld worden.

De verschillende user-rollen die genoemd zijn, monteur, kassamedewerker en backoffice medewerker. Deze rollen hebben allemaal taken die alleen zij kunnen uitvoeren. Met 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

Wanneer je een web-API gaat bouwen, kun je niet zomaar beginnen met programmeren. Het is belangrijk eerst te inventariseren hoe het product moet werken door functionaliteiten voor verschillende rollen te beschrijven. Functionaliteiten beschrijven wat er mogelijk is met de web-API, zoals: inloggen, gegevens opslaan, data wijzigen etc. Daarnaast ga je bepalen welke data je nodig hebt. Vervolgens kun je de architectuur uittekenen door middel van een klassendiagram en sequentiediagrammen.

Je gaat een technisch ontwerp maken dat voldoet aan de volgende eisen:

- Bevat een titelblad, inleiding en inhoudsopgave. In het documenten zitten geen verwijzingen naar afbeeldingen en informatie buiten het document zelf.
- Beschrijft het probleem en de manier waarop deze web-API dat probleem oplost.
- Beschrijft wat de web-API moet kunnen middels minimaal 4 overkoepelende user stories en een sommering van 25 functionele en niet-functionele eisen. Daarnaast moet het voldoen aan het format, zoals je geleerd hebt tijdens de leerlijn.
- Bevat één klassendiagram van het domeinmodel. Dit klassendiagram is taal- en platformafhankelijk en hoeft geen methodes te bevatten.
- Bevat minimaal twee uitgewerkte sequentiediagrammen waarin alle architecturale lagen (controller, service, repository) voorkomen. Zorg ervoor dat deze diagrammen klasse- en methodenamen bevatten die overeenkomen met de namen die je gebruikt in de broncode (Deelopdracht 2).

Op te leveren:

- Het technisch ontwerp van de web-API in PDF (.pdf).

Deelopdracht 2 . Verantwoordingsdocument

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. Je begint al tijdens het ontwikkelen van jouw product met het schrijven van het verantwoordingsdocument. Hierin geef je antwoord op vragen als: 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.

Op te leveren:

- Verantwoordingsdocument in PDF (.pdf) met daarin:
    - Minimaal 5 beargumenteerde technische keuzes, betreffende de OOP-structuur, de web-API en architectuur.
    - Een beschrijving van minimaal 5 limitaties van de web-API en beargumentatie van mogelijke doorontwikkelingen. Het gaat hierbij om limitaties betreffende de functionaliteit van de web-API, niet om de styling.
    - Een link naar jouw project op Github.

Deelopdracht 3. Broncode Spring Boot

In de eerste opdracht heb je uitgewerkt wat de web-API moet kunnen en hoe de architectuur eruit moet komen te zien. In deze opdracht ga je jouw technisch ontwerp als basis gebruiken om de web-API te implementeren in Spring Boot. Jouw web-API heeft minimaal de volgende eigenschappen:

- Het is een REST web-API die data beheert via endpoints;
- Het implementeert 3 kern-functionaliteiten, waarbij CLEAN code en SOLID zijn toegepast.
- De web-API is beveiligd en bevat minimaal 2 en maximaal 3 user-rollen met verschillende mogelijkheden en rechten;
- De web-API handelt eventuele fouten van de gebruiker, datavalidatie of bugs af door middel van exception handling.
- De web-API en database zijn onafhankelijk van elkaar waardoor het mogelijk is om eventueel naar een ander database systeem te wisselen (zoals MySQL, PostgreSQL, SQLite);
- Communicatie met de database vindt plaats door middel van repositories. De database kan in de vorm van CRUD operaties of complexere, samengestelde, queries bevraagd worden.
- De database is relationeel en bevat minimaal 1 one-to-one relatie en 1 one-to-many relatie;
- De web-API maakt het mogelijk om bestanden (zoals muziek, PDF's of afbeeldingen) te uploaden en te downloaden;
- De web-API beveiligt de data op basis van authenticatie en autorisatie op meerdere rollen.
- De broncode van de web-API wordt getest met Spring Boot test, WebMvc en JUnit. Tevens test je 2 klassen uit de service laag met een line coverage van 100% door middel van minimaal 10 nuttige unit-tests, gebruikmakend van de drie A's.
- Het systeem wordt geleverd met een valide set aan data die automatisch in de database wordt gezet en ook unit-tests worden voorzien van eigen test data.
- Je maakt gebruik van Git om jouw project te beheren en zet daarom jouw project zo snel mogelijk op GitHub. Je zorgt voor minimaal 20 kleine beschrijvende commits, maakt minimaal 5 pull requests voor iedere nieuwe feature en mergt deze naar de main branch.

Op te leveren:

- Projectmap met daarin de broncode, inclusief de link naar de Github repository .

Deelopdracht 4. Installatiehandleiding

In de voorgaande opdrachten heb je jouw ontwikkelwerk afgerond. Om ervoor te zorgen dat ook andere ontwikkelaars jouw project kunnen gebruiken, is het belangrijk een installatiehandleiding te schrijven waarin beschreven wordt wat zij hiervoor nodig hebben. Je schrijft jouw installatiehandleiding die uitlegt hoe de web-API geïnstalleerd en gebruikt kan worden. Dit schrijf je voor een mede-ontwikkelaar zonder enige ervaring binnen het backend-landschap. Het installeren of gebruiken van IDE hoeft dus niet meegenomen te worden.

Jouw installatiehandleiding bevat:

- Een inhoudsopgave en inleiding, met daarin een korte beschrijving van de functionaliteit van de web-API en de gebruikte technieken;
- Een lijst van benodigdheden om de web-API te kunnen runnen (zoals applicaties, runtime environments of andere benodigdheden);
- Een stappenplan met installatie instructies;
- Een lijst met (test)gebruikers en user-rollen;
- Een Postman collectie, die gebruikt kan worden om jouw web-API te testen.
- Een lijst van REST-endpoints, inclusief voorbeelden van de JSON-requests. Deze voorbeelden moeten uitgeschreven zijn zoals in Postman, zodat je ze gemakkelijk kunt selecteren, kopiëren en plakken. Hierin leg je ook uit hoe de endpoints beveiligd zijn.

Op te leveren:

- Installatiehandleiding in PDF (.pdf);
- Postman collectie in JSON (.json);

Quickscan

Hieronder vind je een aantal randvoorwaarden waaraan de eindopdracht moet voldoen. De beoordelaar kan op basis van deze eisen de eindopdracht teruggeven en zal deze niet verder nakijken tot aan de eisen zijn voldaan.

Gebruik de quickscan om te zien of je voldoet aan de inlevereisen.

Algemene eisen:

- ☐ Documentatie ingeleverd als .pdf.
- ☐ Document bevat geen bronnen of verwijzingen buiten het document (behalve wanneer hier expliciet naar gevraagd wordt zoals bijvoorbeeld de link naar het Github-project.).
- ☐ De eindopdracht is goed leesbaar zonder storende aanwezigheid van grammatica- en spellingsfouten.
- ☐ Het volledige project en bijbehorende documenten wordt aangeleverd d.m.v. een ZIP-bestand van maximaal 50 MB. (geen .rar).
- ☐ Het ZIP-bestand bevat de volgende elementen:
  - Technisch ontwerp (in .pdf)
  - Verantwoordingsdocument (in .pdf)
  - Broncode van het project (Let op: dus niet alleen de link naar het Github-project)
  - Installatiehandleiding (in .pdf)
  - Postman collectie in JSON (.json);
- ☐ Indien je een herkansing inlevert na het krijgen van feedback, is het ingevulde 'Template herkansingsfeedback' bijgevoegd.

Inhoudelijke eisen:

- ☐ Alle deelopdrachten zijn uitgewerkt en de gevraagde deelproducten zijn aanwezig.
- ☐ De web-API is geprogrammeerd met Springboot en een versie van Java voor long term support, zoals Java 17 of Java 21.
- ☐ Er wordt gebruikgemaakt van DTO's om data te valideren en vervuiling van de database te voorkomen.
- ☐ Het project is geüpload naar een GitHub repository: deze repository staat op public. De link is toegevoegd in jouw verantwoordingsdocument.
- ☐ Het project wordt ingeleverd zonder out-map/target-map/.idea-map en .iml-bestand.
- ☐ Maven is gebruikt als dependency manager.
- ☐ De web-API start op zonder te crashen.

Structuur

De integrale eindopdracht bestaat uit verschillende documenten. Houd hierin de volgende structuur aan:

Technisch ontwerp

- Inleiding

De inleiding fungeert net zoals de inhoudsopgave als een routekaart voor de lezer: hier leg je uit wat de lezer kan verwachten van jouw verslag.

- Algemene omschrijving web-API

Voordat je direct de diepte in duikt, leg je globaal uit welk probleem je hebt willen oplossen met jouw web-API en wat de belangrijkste kernfunctionaliteiten zijn.

- User stories

De user stories zijn beschreven in het format als <rol> wil ik <functionaliteit> zodat <reden/doel> …
Functionele en niet-functionele eisen
Eisen zijn beschreven in het format "Als [rol] kan/wil/doe/gebruik ik…" en voorzien van nummering en duidelijke categorisering.

- Klassendiagram

Je maakt één klassendiagram van alle entiteiten. Dit diagram is taal- en platformafhankelijk, goed leesbaar en tekstueel uitgelegd of beschreven.

- Sequentiediagrammen

In deze sequentiediagrammen komen alle architecturale lagen voor. Daarnaast zijn ze goed leesbaar en tekstueel uitgelegd of beschreven.

Verantwoordingsdocument

- Inleiding
- Verantwoording keuzes
- Mogelijke doorontwikkelingen
- Template herkansingsfeedback*

Installatiehandleiding

1. Inleiding
2. Benodigdheden
3. Installatie instructies (in de vorm van een stappenplan)
4. Testgebruikers
5. Postman collecties
6. REST-endpoints
7. Overige commando's

Alle documenten zijn netjes, verzorgd en voorzien van titelblad, inhoudsopgave en duidelijke formatting wanneer bijvoorbeeld commando's in de terminal moeten worden ingevoerd.

* Indien je een herkansing inlevert na het krijgen van feedback, lever je ook het ingevulde 'Template herkansingsfeedback' in. Dit Word-document kun je vinden in Teams.

Beoordelingscriteria

De eindopdracht wordt beoordeeld op basis van de volgende beoordelingscriteria. Per criterium kent de beoordelaar een aantal punten toe.

Deelopdrachten/ producten		***Weging***
1. Technische ontwerp	**Met deze opdracht worden aspecten van de leeruitkomsten van de cursussen Software design & documentatie (LU2) en Spring Boot (LU3) getoetst.**	**20%**
Criterium 1.1	De student vertaalt de casus (of een eigen idee voor een web-API) naar 4 user stories en vertaalt deze in een opsomming van tenminste 25 relevante functionele en niet-functionele eisen.	5%
Criterium 1.2	De student maakt een overzichtelijke en logisch gestructureerde klassendiagram van het domeinmodel, waarbij de kardinaliteiten wordt aangegeven tussen verschillende klassen en daarmee inzicht geeft in de database structuur.	10%
Criterium 1.3	De student legt op logische en correcte wijze alle verschillende architecturale lagen (controller, service en repository) vast in twee duidelijke sequentiediagrammen. Deze sequentiediagrammen bevatten klasse- en methodenamen die overeenkomen met die in het ingeleverde project.	5%
2. Verantwoordings-document	**Met deze opdracht worden aspecten van de leeruitkomst van de cursus Programmeren met Java (LU1) getoetst.**	**10%**
Criterium 2.1	De student beargumenteert minimaal 5 gemaakte technische keuzes (bijvoorbeeld over OOP-structuren, design patterns, Web-API of architectuur) en reflecteert op ieder punt.	5%
Criterium 2.2	De student beschrijft minimaal 5 realistische limitaties van de functionaliteit van de web-API en beargumenteert welke doorontwikkelingen mogelijk en/of wenselijk zijn	5%

3. Broncode Spring Boot	**Met deze opdracht worden aspecten van de leeruitkomsten van alle 3 de cursussen getoetst.**	**65%**
Criterium 3.1	De student implementeert 3 belangrijke kern-functionaliteiten (naast authenticatie en autorisatie) op correcte en kwalitatieve wijze en past de principes van Clean Code en SOLID correct toe.	15%
Criterium 3.2	De student voert 2 geslaagde integratie-tests uit door de broncode van de web-API te testen. Tevens test de student op correcte wijze 2 klassen uit de service laag met een line coverage van 100% door middel van minimaal 10 nuttige unit-tests, gebruikmakend van de drie A's.	10%
Criterium 3.3	De student past exception handling op effectieve wijze toe bij het beheren van mogelijke errors binnen de web-API.	5%
Criterium 3.4	De student beheert de code met correct gebruik van Git. De student maakt minimaal 20 kleine commits met compacte en zinvolle commit-messages en maakt minimaal 5 pull requests die naar de main branch gemerged zijn.	5%
Criterium 3.5	De student stelt modellen en data constraints op om de integriteit van de databasegegevens te waarborgen, vult de databases met testdata via data.sql en valideert inkomende data op effectieve wijze.	10%
Criterium 3.6	De Student past op correcte en veilige wijze authenticatie en authorisatie toe en maakt gebruik van meerdere rollen binnen de web-API.	10%
Criterium 3.7	De student implementeert een kwalitatieve web-API volgens de REST richtlijnen.	10%
4. Installatiehandleiding	**Met deze opdracht worden aspecten van de leeruitkomst van de cursus Software design & documentatie (LU2) getoetst.**	**5%**
Criterium 4.1	De student maakt een zelfgeschreven installatiehandleiding waarmee iemand zonder kennis van het project de gemaakte web-API zelfstandig kan draaien.	5%
Totaal		100%