Text

Description automatically generated with low confidence

# Portfolio Semester 3 Software



Inleiding

Dit document omschrijft de aanlevering van mijn portfolio, mijn bijdrage tot het bewijsmateriaal & conclusies die ik voor elk leerresultaat heb kunnen trekken.   
De mappenstructuur is zo opgezet dat er voor elk onderdeel 1 map is met bewijzen voor het aantonen van dat leerresultaat.

Contents

[Portfolio Semester 3 Software 1](file:///D:\Fontys\Software%20S3\Leeswijzer.docx#_Toc106365961)

[IPS-LU1 5](#_Toc106365962)

[Bewijslast IPS-LU1 Web application 9](#_Toc106365963)

[User friendly 9](#_Toc106365964)

[Full stack 9](#_Toc106365965)

[Object relation mapping 9](#_Toc106365966)

[Bewijslast IPS-LU2 12](#_Toc106365967)

[IPS-LU4 13](#_Toc106365968)

[Bewijslast CI/CD 14](#_Toc106365969)

[Docker Desktop en Docker runner: 14](#_Toc106365970)

[Dockerfiles 14](#_Toc106365971)

[Docker compose 14](#_Toc106365972)

[Git links 15](#_Toc106365973)

# IPS-LU1

Web application

Canvas pagina: <https://fhict.instructure.com/courses/12097/pages/front-end-development-what-is-javascript-and-what-do-we-mean-by-ux?module_item_id=752605>

https://fhict.instructure.com/courses/12097/pages/back-end-development-which-language-which-framework-and-why-does-it-matter?module\_item\_id=752608

**Expected outcomes:**

|  |  |
| --- | --- |
| **Onderdeel** | **Bewijslast** |
| User friendly: You apply basic User experience testing and development techniques. | * Regelmatig anderen door mijn website laten lopen en de feedback noteren en verwerken. * Integration testing |
| Full-stack: You design and build a full stack application using commonly accepted front end (Javascript-based framework) and back end techniques (e.g. Object Relational Mapping) choosing and implementing relevant communication protocols and addressing asynchronous communication issues. | * Nette code met de juiste structuur. * Feedback docenten. * ORM diagram en/of component diagram maken. * De juiste communicatie protocollen toepassen(bijv. ORM voor SQL injections, of WebSockets). |

Welke JavaScript-frameworks zijn er voor frontend development en wat zijn de voor- en nadelen?

* Er zijn heel veel frameworks beschikbaar voor JavaScript. Hieronder staan de twee

populairste, met een lijst van voor- en nadelen erbij.

* Angular
  + Voordelen:
    - Open source. Hierdoor is het eenvoudiger om je webapp te personaliseren. Ook is er hierdoor veel support te vinden online.
    - Single page application. Omdat er maar één HTML pagina geladen wordt en updates enkel op die pagina weergegeven worden, werkt Angular snel en is het user-friendly
    - Eenvoudig te testen. Door een built in dependency injection is het redelijk eenvoudig om individuele componenten te testen.
  + Nadelen:
    - Beveiliging. Omdat er geen server authorization/authentication is in Angular, is het minder veilig. Er is geen credential validation, waardoor security breaches sneller onstaan.
    - Memory leaks. Door onnodig geheugenverbruik wordt de kans groter dat je webapp crasht of vertraagd wordt.
    - Omdat er veel verschillende manieren om een functionaliteit uit te voeren, is het lastig vast te stellen wat de beste manier is om een functionaliteit uit te werken.
* React
  + Voordelen:
    - Eenvoudig te leren. Voor React heb je niet veel voorkennis nodig. Hierdoor is het relatief makkelijk om de basis te leren.
    - Open source. Hierdoor is het eenvoudiger om je webapp te personaliseren. Ook is er hierdoor veel support te vinden online.
    - Single page application. Omdat er maar één HTML pagina geladen wordt en updates enkel op die pagina weergegeven worden, werkt Angular snel en is het user-friendly
    - Herbruikbare componenten. Een React app bestaat uit meerdere componenten die allemaal verantwoordelijk zijn voor een klein, herbruikbaar stukje HTML code. Omdat deze componenten zo breed toegepast kunnen worden, kun je een component voor verschillende functionaliteiten toepassen.
  + Nadelen:
    - Slechte/weinig documentatie beschikbaar. Omdat React zich zo snel ontwikkelt, is het lastig om documentatie te vinden die helder en up-to-date is.
    - Gebrek aan MVC. Omdat React enkel de UI-laag van je app kan uitvoeren, kan het lastig zijn om het MVC-model toe te passen zonder andere tools toe te passen. Zonder deze extra tools is het lastig om het view-deel van je applicatie te implementeren.

Welk framework ga ik gebruiken?

* Omdat ReactJS eenvoudiger is om in te stappen voor iemand met weinig tot geen voorkennis, ga ik het gebruiken. Ik heb nog nooit met JavaScript gewerkt, dus het lijkt mij slim om een framework te kiezen met een relatief laag instapniveau. Angular is bovendien minder stabiel in zowel preformance als veiligheid dan React.

Welke taal en welk framework ga ik gebruiken voor mijn backend?

* Voor de backend ga ik Java springboot gebruiken. Aangezien ik al een aardige tijd in C# heb gewerkt, lijkt het mij goed om eens wat anders te leren. Zo verbreed ik mijn kennis, en het is natuurlijk ook leuk om eens wat nieuws te leren.
* Ik heb gekozen voor springboot, omdat dit framework het eenvoudig maakt om ORM toe te passen. Bovendien is er over springboot veel documentatie te vinden, wat helpt met het leren van nieuwe dingen.

Wat houdt object relation mapping(ORM) in en hoe is het nuttig voor mijn project?

* ORM is een techniek voor het omzetten van gegevens tussen verschillende type systems met behulp van OOP talen. Er wordt, als het ware, een virtuele object database gemaakt, die vanuit een programmeertaal gebruikt kan worden.
* De voordelen van ORM toepassen zijn:
  + Je hoeft zelf geen SQL statements te schrijven. Het ORM-framework genereert deze voor je. Typfouten in de statements worden dus compleet ontweken.
  + Overstappen naar een ander type database wordt heel eenvoudig, omdat het ORM-framework de datatypes van entiteiten automatisch aanpast op je database.
* De nadelen van ORM toepassen zijn:
  + Het configureren van ORM is in eerste instantie ingewikkeld. Hierdoor duurt het mogelijk een tijdje voordat je applicatie naar behoren werkt. Echter kost het je weinig tot geen werk om deze configuratie te onderhouden. Wanneer deze goed uitgevoerd is, hoef je er niet maar naar om te kijken.
  + Het is belangrijk om als programmeur te weten hoe je code precies functioneert. Omdat ORM bijna alle database functionaliteiten binnen je applicatie automatisch afhandelt, worden je skills als SQL-developer er niet beter op.
* Wat is een bean?
  + Een bean is een object dat is beheerd wordt door een Spring IoC container. Een Spring IoC Container is de basis van het spring framework. Het creëert en onderhoudt objecten en hun dependencies tot het einde van hun lifecycle. IoC is een andere naam voor dependency injection, oftewel een design pattern dat ervoor zorgt object geïnstantieerd kan worden, onafhankelijk van eventuele dependencies.

Wat zijn WebSockets en hoe passen ze in mijn project?

* REST('REpresentational State Transfer'-like) is een protocol gebaseerd op een request/response concept, oftewel de client stuurt een request, en de server geeft antwoord. Dit concept wordt Synchronous communication genoemd. Synchrone communicatie bestaat dus altijd uit vraag, gevolgd door een directe reactie. Voor API-calls is dit concept ideaal, maar voor een directe, real-time connectie gaat REST niet op. WebSockets maken het mogelijk om toch via één TCP-verbinding asynchroon te communiceren, waardoor meerdere communicatiestromen tegelijk kunnen functioneren. Asynchrone communicatie wordt bijvoorbeeld gebruikt voor live chats of real-time games, en maakt dus geen gebruik van het vraag/antwoord concept. Bij het gebruik van WebSockets wordt in het begin een verbinding opgesteld, en een kanaal geopend, wat de mogelijkheid aan de server én client biedt om elkaar berichten te sturen op ieder moment.
* Binnen mijn project worden WebSockets toegepast in de vorm van een live-chat. Via deze chat kunnen gebruikers discussiëren over hun favoriete voetbalclubs en competities. Binnen deze chat hebben gebruikers de mogelijkheid om in een publieke chat met een hele groep te communiceren, of om met één specifieke gebruiker een privé chat te voeren.

Conclusie

* Omdat React een goed punt is om te beginnen met JavaScript, lijkt dit me een goede keuze om te starten.
* Mijn backend gaat in Java gemaakt worden, omdat het heel handig is voor ORM, met gebruik van het Springboot-framework. ORM zorgt ervoor dat alle database-gerelateerde zaken in je applicatie minder vatbaar zijn voor menselijke fouten(bijv. typfouten). Ook is het goed om mijn kennis te verbreden door een nieuwe taal te leren.
* WebSockets worden toegepast om asynchrone communicatie mogelijk te maken. In mijn project wordt dit in de vorm van een live-chat geïmplementeerd.

# Bewijslast IPS-LU1 Web application

## User friendly

You apply basic User experience testing and development techniques.

Om aan te tonen dat ik user experience testen uit kan voeren, en de juiste vervolgstappen onderneem, heb ik een aan een aantal medestudenten gevraagd om feedback te geven over de UX van onze proftaak(Refactoring Flow). De feedback die ik ontvangen heb van mijn medestudenten heb ik verwerkt in een verslag. In dit verslag is ook beschreven hoe deze feedback verwerkt dient te worden. Dit verslag staat in de folder ‘documentatie’ en heet ‘UX testing verslag’.

## Full stack

You design and build a full stack application using commonly accepted front end (Javascript-based framework) and back end techniques (e.g. Object Relational Mapping) choosing and implementing relevant communication protocols and addressing asynchronous communication issues.

### Object relation mapping

Om uit te leggen hoe ORM toegepast wordt in mijn project, lopen we hieronder door de opzet van één functionaliteit heen. Alle functionaliteiten bestaan uit dezelfde componenten, zoals ze hieronder beschreven staan. Het onderstaande stappenplan beschrijft hoe een lijst met clubs uit een specifieke competitie opgehaald wordt.

#### Stappenplan ORM

1. Configuration.

Text

Description automatically generatedAls eerste wordt gedefinieerd dat transacties afgehandeld worden door de bestanden in de Repository folder met behulp van de annotation @ComponentScan. In deze folder staan de repositories die communiceren met de database. Vervolgens wordt gedefinieerd hoe een Bean van een entiteit eruit moet zien. Dit wordt uitgevoerd door een entityManager methode. In deze methode wordt een lokale instantie van de entityManager aangemaakt. Vervolgens wordt aangegeven dat enkel de bestanden in het mapje Model als entiteit herkend worden door de Bean. Door in de configuratie te definiëren hoe deze beans eruit moeten zien, en waar ze horen te staan, is het eenvoudig om nieuwe entiteiten toe te voegen en af te handelen.

1. Entiteiten.

Text

Description automatically generatedZoals in stap één te lezen is, herkent de applicatie een entiteit alleen als deze in het mapje ‘Model’ staat. Iedere klasse in dit mapje is dan ook een entiteit/model. Aangezien we een lijst met clubs op willen halen, gaan we in op de entiteit ‘Club’. Om ervoor te zorgen dat Spring het model herkent als entiteit, staat de annotation @Entity bovenaan. Een regel eronder wordt @Table aangeroepen. Zo weet Spring dat er een database-tabel hoort te zijn voor deze entiteit, en hoe deze hoort te heten. Wanneer deze niet gevonden wordt in de database, wordt deze automatisch gegenereerd. Vervolgens geef ik aan welke properties een Club moet hebben, en met de @Column annotation geef ik aan dat er voor deze property een column moet bestaan in de tabel van de entiteit, en hoe de column hoort te heten. Omdat een League bestaat uit meerdere clubs, wordt er een ManyToOne relatie gedefinieerd. Dit houdt in dat een League meerdere clubs kan hebben, en een club maar één League heeft. Vervolgens wordt aangegeven dat de column in de Club-tabel “league\_id” heet, en dezelfde waarde heeft als de “id” column in de League-tabel. In de kleine afbeelding hierboven is te zien hoe deze relatie eruit ziet in de League-entiteit. De relatie wordt hier omgedraaid naar OneToMany, en in plaats van een enkele entiteit, heeft League een lijst met alle clubs die bij de League horen.

1. Controllers.

Text

Description automatically generatedDe Controller vangt API-calls op vanuit de front-end. De Controller haalt vervolgens de benodigde data, in dit geval het “leagueid”, zet deze om van een JSON-property naar een bruikbaar datatype, in dit geval een “Long”. Vervolgens wordt de Service aangeroepen.

1. Services.

Text

Description automatically generatedNadat de Controller een request ontvangen heeft, komt deze in de Service terecht. De service implementeert de Custom repository interface. Vanuit de service wordt de Repository aangeroepen.

1. Custom repositories.Text

   Description automatically generated

Zoals hierboven vermeld, is de Custom Repository een interface. In deze interface worden alle methodes die niet door de standaard JPA-repository herkend worden gedefinieerd. De JpaRepository bevat bijvoorbeeld geen methode om alle clubs op te halen op basis van het ID van een competitie, zoals gevraagd in de beschreven functionaliteit. Daarom wordt deze in de Custom Repository gedefinieerd. Vanuit de Custom Repository wordt er nog geen database call uitgevoerd. Dit gebeurt pas in de standaard Repository.

1. Standaard repositories.

Text

Description automatically generatedIn een standaard Repository staan geen methodes gedefinieerd. Voor de veelgebruikte methodes wordt de JpaRepository aangeroepen. Voor de meer abstracte, specifieke methodes wordt een Custom Repository, zoals hierboven beschreven, toegepast. De standaard Repository ‘extend’ de JpaRepository en de Custom Repository, waardoor alle methodes van beide interfaces herkend worden door de Repository. Vanuit deze Repository worden alle database calls uitgevoerd. Door de @Repository en @EnableJpaRepositories herkent Springboot dat deze interface gerechtigd is om database calls te maken. Wanneer een methode aangeroepen wordt via de ClubRepository, gaat deze een database call maken. Als er data gevonden wordt die voldoet aan de meegegeven parameters, wordt deze via alle bovenstaande onderdelen, dus van Repository tot Controller, teruggestuurd. Vanuit de controller wordt de data opgestuurd naar de front-end.

IPS-LU2

Software quality

Canvas pagina: <https://fhict.instructure.com/courses/12097/pages/what-is-quality-assurance-and-why-is-it-important?module_item_id=752617>

**Expected outcomes:**

|  |  |
| --- | --- |
| **Onderdeel** | **Bewijslast** |
| Tooling and methodology: Carry out, monitor and report on unit integration, regression and system tests, with attention for security and performance aspects, as well as applying static code analysis and code reviews. | * Unit testing * Systeem testen * Testplan |

# Bewijslast IPS-LU2

Ik heb een onderzoeksdocument geschreven, waarin ik beschrijf hoe ik de software kwaliteit waarborg in mijn project door middel van geautomatiseerde testen. Dit onderzoeksdocument is te vinden onder Portfolio S3 Luc van den Acker -> Documentatie -> Onderzoeksdocumenten -> ResearchDocumentTestplan.

# IPS-LU4

CI/CD

Canvas pagina: <https://fhict.instructure.com/courses/12097/pages/what-is-software-release-management?module_item_id=752622>

**Expected outcomes:**

|  |  |
| --- | --- |
| **Onderdeel** | **Bewijslast** |
| You implement a continuous integration and deployment solution (using e.g. Gitlab CI and Docker). | * CI/CD pipeline in GitLab * Docker image aanmaken en lokaal runnen. |

Wat is Continuous Integration?

* Continuous Integration gaat over het automatisch integreren van wijzigingen in je code naar een specifieke branch, enkel wanneer het voldoet aan bepaalde kwaliteitscontroles. Als het niet voldoet aan de eisen, wordt de code afgewezen.
* Alle code die je naar de desbetreffende branch pusht, wordt automatisch gebuild en getest.
* Hierdoor weet je zeker dat alle code op een bepaalde branch goed functioneert.

Wat is Continuous Delivery?

* Continuous Delivery is eigenlijk Continuous Integration, maar dan een stapje verder. Hierbij wordt namelijk gekeken naar het functioneren van je applicatie.
* Continous Delivery maakt gebruik van end-to-end testing. Als deze testen slagen, wordt de software gelijk op een server geïnstalleerd.
* Continuous Delivery zorgt ervoor dat er altijd een werkende versie van je applicatie beschikbaar is op je server, waardoor je als programmeur zelf niet constant bezig bent met nieuwe versies controlleren en uploaden naar je server.

Pipeline opzetten via gitlab:

Om een CI/CD Pipeline op te zetten via gitlab, zijn een aantal dingen nodig: Docker Desktop, een Docker runner, een YAML-file en een pom.xml file.

*pom.xml file:*

De pom.xml file bevat informatie over het project die maven nodig heeft om het project te bouwen, zoals dependencies en build directories.

*YAML-file:*

Een YAML file is een door mensen leesbare data serialization file. In de YAML file configureer je welke taken je runner uit moet voeren, en hoe je runner dit moet doen.

# Bewijslast CI/CD

## Docker Desktop en Docker runner:

Zoals hierboven uitgelegd, is CI/CD simpelweg een geautomatiseerd proces om kwaliteitscontroles uit te voeren, voordat een stuk code naar een server gestuurd wordt. Docker is een tool, bedoeld om dit proces zo eenvoudig mogelijk te laten verlopen. Docker pakt een stuk code, inclusief depencencies, in en stopt dit in een container image. Een container image is een klein pakket, met alle onderdelen die nodig zijn om dat onderdeel van de applicatie te runnen. De runner zorgt ervoor dat je builds en deployments kan draaien in je Docker container. Om een image lokaal te kunnen draaien, zijn er een aantal benodigdheden:

### Dockerfiles

Om een image te kunnen maken van een project, wordt een Dockerfile gebruikt. Voor iedere Git-Repository dient een Dockerfile gemaakt te worden. In deze Dockerfile wordt gespecificeerd welk framework gebruikt wordt, en welke versie van dit framework. Docker maakt een image van een

Text

Description automatically generatedjar-file. In de Dockerfile staat aangegeven hoe deze jar-file heet en waar deze opgeslagen is. Daarnaast wordt een entrypoint aangegeven, waarin gedefinieerd wordt welke programmeertaal gebruikt wordt, dat er gebruik gemaakt wordt van een jar-file. In de entrypoint wordt ook aangegeven wat de executable is, oftewel de jar-versie van het project waarvan een image gemaakt moet worden. Tot slot wordt aangegeven welke op welke poort de image moet draaien.

### Docker compose

Text

Description automatically generatedNadat er van alle repositories een container gemaakt is, worden deze samengevoegd en gedraaid met behulp van Docker compose. Natuurlijk weet Docker van zichzelf niet welke containers er samengevoegd moeten worden. Daarom heb ik een Docker-compose.yml file gemaakt, waarin ik aangeef welke services meegenomen moeten worden in de compose. Met build wordt aangegeven in welke map de repository staat op mijn PC. Met image geef ik aan hoe de Docker image van de specifieke repository heet. Vervolgens specificeer ik op welke poort de image moet draaien. Helaas is het niet gelukt om Docker compose aan de gang te krijgen. Bij iedere poging zie ik “Could not find or load main class” als foutmelding. Zeer waarschijnlijk komt dit door een fout in de Maven build stage in mijn pipeline. Tijdens deze job wordt mijn project gecompileerd in de pipeline. De kans is natuurlijk aanwezig dat er tijdens deze job iets verkeerd gaat, waardoor, tijdens de Docker compose fase, de main-klasse niet gevonden kan worden.

# Git links

Frontend: <https://git.fhict.nl/I434575/individuele-opdracht-s3-software.git>

Backend: <https://git.fhict.nl/I434575/s3_frontend.git>