Het Software Delivery Proces

**van Finalist**

### Student: Yassine Ben-MoussaOpleiding: Informatica(information engineering)Studentnummer: 1596411Bedrijfsbegeleider: Brian SnijdersDocentbegeleider: Elfriede KrauthDatum: 17 maart 2015

#  Voorwoord

Ik heb het afgelopen semester met veel plezier mijn afstudeeropdracht bij Finalist mogen doen. De mensen van Finalist hebben me uitstekend behandeld. Ik wil Finalist bedanken dat ze me de kans hebben gegeven om bij hen af te kunnen studeren. In het bijzonder wil ik mijn dank uiten aan Brian Snijders. Brian heeft mij uitstekend begeleid en was bovendien een prettige collega om mee te werken. Ook wil ik Elfriede Krauth bedanken voor de begeleiding die zij mij heeft geboden het afgelopen semester.

#  Samenvatting

Ik heb onderzoek gedaan naar het software delivery proces binnen één van de units van Finalist. Vanuit Finalist is de vraag ontstaan om onderzoek te doen naar dit proces, zodat dit proces voor hen inzichtelijker zou worden en om het te kunnen verbeteren. Dit proces was niet inzichtelijk genoeg. In eerste instantie zou dit onderzoek zowel intern bij Finalist als extern bij klanten van Finalist worden gedaan, maar samen met mijn begeleider hebben we besloten om het alleen intern binnen Finalist te doen, omdat de beschikbare tijd niet voldoende was. Naast het feit dat er onderzoek is gedaan naar het software delivery proces van Finalist, is er ook onderzoek gedaan naar usability en security van webapplicaties.

Het grootste gedeelte van het onderzoek naar het software delivery proces van Finalist is gedaan door de operationele tak van Finalist te interviewen. De operationele tak bestaat uit: Een operationeel directeur, een unitmanager, een projectmanager, developers, testers, informatie analisten en een solution expert. Deze tak van Finalist is immers direct betrokken bij het software delivery proces in tegenstelling tot de andere takken van Finalist. Uit deze interviews is naar voren gekomen dat Finalist gebaat is bij verschillende soorten inzichten. Voor de proof of concept is er gekozen om alleen inzichten mee te nemen naar de realisatie van de proof of concept als deze een directe relatie hebben met het software delivery proces. Een directe relatie houdt in dat de gewenste inzichten een direct verband hebben met deployments en releases van software. Aan de hand van deze criteria is er vervolgens bepaald dat de wensen voor inzichten van de managementleden meegenomen zullen worden naar de realisatie van de proof of concept waarbij de inzichten van deployments van software een hogere prioriteit hebben.

Ik heb ook onderzoek gedaan naar security van webapplicaties, omdat de proof of concept online gaat draaien. Hierbij was het van belang om onderzoek te doen naar de meest voorkomende kwetsbaarheden van webapplicaties. Vervolgens heb ik onderzoek gedaan naar kwetsbaarheden die van toepassing zijn op de proof of concept, Broken Authentication, SQL injection en Cross Site Scripting zijn deze kwetsbaarheden. Aan de hand hiervan is er onderzoek gedaan op welke manier de proof of concept tegen deze kwetsbaarheden beveiligd kan worden.

Onderzoek naar usability van webapplicaties is gedaan, omdat het van belang was dat de proof of concept makkelijk te gebruiken is. Hierbij is er onderzoek gedaan naar welke usability richtlijnen van toepassing zijn op de proof of concept en hoe deze richtlijnen toegepast kunnen worden. Ook heb ik onderzoek gedaan naar op welke manier de inzichten het best gepresenteerd kunnen worden. Hierbij heb ik een usability expert van Finalist geïnterviewd.

Er is een proof of concept gerealiseerd die voortvloeit uit de resultaten van het onderzoek. De proof of concept maakt het mogelijk dat Finalist een beter inzicht heeft over het software delivery proces. Daarnaast is het beveiligd tegen aanvallen van hackers en zijn de resultaten van het onderzoek naar usability hierin toegepast, zodat het makkelijk te gebruiken is voor de gebruikers.

 Inhoud

[Voorwoord 2](#_Toc414309917)

[Samenvatting 3](#_Toc414309918)

[1 Inleiding 7](#_Toc414309919)

[2 Randvoorwaarden, uitgangspunten en afbakening 8](#_Toc414309920)

[2.1 Randvoorwaarden 8](#_Toc414309921)

[2.2 Uitgangspunten 8](#_Toc414309922)

[2.3 Afbakening 8](#_Toc414309923)

[3 Opdracht 9](#_Toc414309924)

[3.1 Aanleiding 9](#_Toc414309925)

[3.2 Huidige situatie 9](#_Toc414309926)

[3.3 Doelstelling 9](#_Toc414309927)

[3.4 Hoofd- en deelvragen 9](#_Toc414309928)

[4 Aanpak 11](#_Toc414309929)

[5 Software delivery proces 14](#_Toc414309930)

[5.1 Wie zijn de belanghebbenden van het software delivery proces en van welke belanghebbende worden de wensen naar de proof of concept meegenomen? 15](#_Toc414309931)

 5.2 Wat is een software delivery proces volgens de literatuur? 15

[5.3 Wat is een software delivery proces van Finalist en waaruit bestaat dit proces? 16](#_Toc414309932)

[5.4 Welke inzichten zijn nodig om het software delivery proces van Finalist inzichtelijker te maken? 17](#_Toc414309933)

[5.4.1 Wie heb ik geïnterviewd Stap 4 17](#_Toc414309934)

[5.4.2 Resultaten van de interviews Stap 5 18](#_Toc414309935)

[5.4.3 Lijst prioriteiten inzichten 19](#_Toc414309936)

[5.5 Conclusie 21](#_Toc414309937)

[6 Security 22](#_Toc414309938)

[6.1 Wat zijn de meest voorkomende kwetsbaarheden van webapplicaties? 23](#_Toc414309939)

[6.2 Welke onderdelen van de proof of concept moeten beveiligd worden? 24](#_Toc414309940)

[6.3 Hoe kunnen de onderdelen van de proof of concept beveiligd worden? 26](#_Toc414309941)

[6.3.1 SQL injection 26](#_Toc414309942)

[6.3.2 Broken Authentication and Session Management 29](#_Toc414309943)

[6.3.3 Cross Site Scripting 30](#_Toc414309944)

[6.4 Conclusie 30](#_Toc414309945)

[7 Usability 32](#_Toc414309946)

[7.1 Welke richtlijnen van usability zijn nodig voor de proof of concept? 33](#_Toc414309947)

[7.2 Hoe kunnen de inzichten het beste gepresenteerd worden? 35](#_Toc414309948)

[7.3 Conclusie 37](#_Toc414309949)

[8 Conclusies en aanbevelingen 39](#_Toc414309950)

[8.1 Conclusies 39](#_Toc414309951)

[8.2 aanbevelingen 42](#_Toc414309952)

[9 Evaluatie 43](#_Toc414309953)

[10 Literatuurlijst 44](#_Toc414309954)

[11 Woordenboek 47](#_Toc414309955)

[12 Bijlage proof of concept 49](#_Toc414309956)

[12.1 Tools 49](#_Toc414309957)

[12.1.1 JAVA, MySQL en Java Database Connectivity 49](#_Toc414309958)

[12.1.2 Spring MVC 49](#_Toc414309959)

[12.1.3 Rundeck en Jenkins 50](#_Toc414309960)

[12.2 Security 51](#_Toc414309961)

[12.3 Software delivery proces(inzichten) 54](#_Toc414309962)

[12.4 Usability 58](#_Toc414309963)

[13 Bijlage Finalist 62](#_Toc414309964)

[13.1 Beschrijving Finalist 62](#_Toc414309965)

[13.2 Software 62](#_Toc414309966)

[13.3 Organisatiestructuur 63](#_Toc414309967)

[14 Bijlage interviews 64](#_Toc414309968)

[Interview: Brian Snijders Functie: Solution expert 64](#_Toc414309969)

[Interview: Ivor Functie: Developer 66](#_Toc414309970)

[Interview Albert Functie:Projectmanager 68](#_Toc414309971)

[Interview: Brian Functie: Solution expert 72](#_Toc414309972)

[Interview: Kolja Functie: Unit manager 76](#_Toc414309973)

[Interview: Mark de Vos Functie: Operationeel directeur 79](#_Toc414309974)

[Interview: Joe Functie: Usability expert 82](#_Toc414309975)

[16 Bijlage security 86](#_Toc414309976)

[OWASP 86](#_Toc414309977)

[Cyber Security Consulting Blog 86](#_Toc414309978)

[Cyber Security Nexus 86](#_Toc414309979)

[Edgescan 87](#_Toc414309980)

[Vergelijken van kwetsbaarheden tussen de verschillende bronnen 87](#_Toc414309981)

[17 Bijlage Plan van aanpak 91](#_Toc414309982)

[Inleiding 93](#_Toc414309983)

[1 Opdracht 94](#_Toc414309984)

[2 onderzoeksvragen 95](#_Toc414309985)

[2.1 Het software delivery proces 95](#_Toc414309986)

[3 Aanpak 97](#_Toc414309987)

[3.1.1 Software delivery proces 97](#_Toc414309988)

[3.2 Realisatie 100](#_Toc414309989)

[4 Doelstellingen/oplevering 101](#_Toc414309990)

[5 Finalist 103](#_Toc414309991)

[5.2 Software 103](#_Toc414309992)

[6 Risicomanagement 105](#_Toc414309993)

[7 Planning 106](#_Toc414309994)

[8 Algemeen 108](#_Toc414309995)

#  1 Inleiding

Finalist is een software bedrijf dat verschillende software oplossingen realiseert, meer over Finalist is in hoofdstuk 13 te vinden. Finalist wilt haar dienstverlening verbeteren. Hierdoor is de opdracht ontstaan om te onderzoeken hoe het software delivery proces van Finalist verbeterd kan worden. In dit document zal uiteengezet worden hoe ik onderzoek heb gedaan en wat de resultaten van het onderzoek zijn. Daarnaast zal uitgelegd worden welke keuzes ik heb gemaakt en waarom.

In hoofdstuk 2 behandel ik randvoorwaarden, uitgangspunten en zet ik de afbakening van mijn onderzoek neer. In hoofdstuk 3 richt ik mij op de opdracht. Hierin behandel ik onderwerpen als de aanleiding van de opdracht, het doel van de opdracht en de hoofd- en deelvragen. In hoofdstuk 4 zal ik uitleggen welke aanpak ik heb gehanteerd voor mijn onderzoek. In hoofdstuk 5 behandel ik het onderzoek naar het software delivery proces binnen één van de units van Finalist. In hoofdstuk 6 richt ik mij op het onderzoek naar security. In hoofdstuk 7 richt ik mij op het onderzoek naar gebruiksvriendelijkheid van webapplicaties.

# 2 Randvoorwaarden, uitgangspunten en afbakening

In dit onderdeel zal ik aangeven met welke randvoorwaarden ik voor mijn opdracht te maken heb en welke uitgangspunten er zijn. Daarnaast zet ik ook de afbakening van mijn opdracht neer.

## 2.1 Randvoorwaarden

Er zijn een aantal randvoorwaarden die voor mijn opdracht van toepassing zijn:

* De informatie over deployments en releases bevinden zich in Rundeck en Jenkins. Jenkins is een release tool en Rundeck is een deployment tool. Voor de proof of concept moeten de informatie over releases en deployments uit deze twee systemen gehaald worden.
* Het realiseren van de proof of concept moet geschieden met behulp van open-source tooling.
* De proof of concept zal online draaien.

## 2.2 Uitgangspunten

Er zijn een aantal uitgangspunten die voor mijn opdracht van toepassing zijn:

* Voor het onderzoek naar het software delivery proces van Finalist zullen interviews worden afgenomen. Hierbij wordt er vanuit gegaan dat er binnen Finalist voldoende kennis van dit proces aanwezig is.
* Er wordt vanuit gegaan dat Jenkins(release tool) en Rundeck(deployment tool) toegankelijk zijn om informatie uit deze systemen te kunnen halen.
* Voor het realiseren van de proof of concept wordt er vanuit gegaan dat dit zal geschieden met de programmeertaal JAVA.

## 2.3 Afbakening

De proof of concept dient met open-source tooling te worden gerealiseerd. Gezien het feit dat het onderzoek naar het software delivery proces een breed karakter heeft, zal dit in hoofdstuk 5 verder worden afgebakend.

# 3 Opdracht

In dit hoofdstuk zullen verschillende onderdelen die met de opdracht te maken hebben worden behandeld. De aanleiding en de doelstelling van de opdracht zijn onderdelen die behandeld worden. Ook de hoofd- en deelvragen zullen in dit hoofdstuk aan bod komen.

## 3.1 Aanleiding

Binnen Finalist is het software delivery proces een belangrijk proces. Finalist wil het software delivery proces verbeteren en dit begint met het inzichtelijker maken van informatie uit dit proces. Door dit proces inzichtelijker te maken kan Finalist het proces beter sturen. Hierdoor kan Finalist zien welke aspecten van dit proces goed en minder goed gaan. Daarnaast wil Finalist door het inzichtelijker maken van het software delivery proces in een later stadium bepaalde onderdelen van dit proces kunnen factureren aan de klant.

## 3.2 Huidige situatie

Op dit moment zijn de inzichten van het software delivery proces niet inzichtelijk genoeg om het proces nog beter te kunnen sturen en om bepaalde onderdelen hiervan kunnen factureren aan de klant. Er zit wel informatie in de hoofden van mensen, maar er is geen centraal punt waar inzichten over dit proces worden weergegeven.

## 3.3 Doelstelling

Het doel is om het software delivery proces inzichtelijker te maken voor Finalist waardoor het proces nog beter gestuurd kan worden en bepaalde onderdelen van dit proces aan de klant gefactureerd kunnen worden. Er dient een proof of concept gerealiseerd te worden die deze inzichten zal weergeven.

## 3.4 Hoofd- en deelvragen

Om het onderzoek gestructureerd te kunnen verrichten heb ik een hoofdvraag geformuleerd die beantwoord dient te worden. De hoofdvraag van het onderzoek is:

#### Hoe kan het software delivery proces inzichtelijker worden gemaakt, op een voor gebruikers veilige en gebruikersvriendelijke wijze, zodat Finalist het software delivery proces kan verbeteren?

Om de hoofdvraag te kunnen beantwoorden heb ik deelvragen geformuleerd die in samenhang antwoord moeten geven op de hoofdvraag. De hoofdvraag bevat drie onderzoekscomponenten: Het software delivery proces***,*** securityenusability. Ik zal hieronder voor ieder onderdeel de deelvragen weergeven en de methode waarmee onderzoek zal worden gedaan.

Software delivery proces*Methode:* Literatuurstudie en het afnemen van interviews.

Literatuurstudie zal worden gedaan om te weten wat een software delivery proces volgens de literatuur is. Het afnemen van interviews zal worden gedaan om te achterhalen wat een software delivery proces bij Finalist is en om te achterhalen welke inzichten nodig zijn om het software delivery proces inzichtelijker te maken.

Hieronder volgen de deelvragen:

* Wat is een software delivery proces volgens de literatuur?
* Wat is een software delivery proces van Finalist en waaruit bestaat dit proces?
* Welke inzichten zijn nodig om het software delivery proces inzichtelijker te maken?

Security*Methode:* Literatuurstudie

Hieronder volgen de deelvragen:

* Wat zijn de meest voorkomende kwetsbaarheden van webapplicaties volgens de literatuur?
* Welke onderdelen van de proof of concept moeten worden beveiligd?
* Hoe kunnen de onderdelen van de proof of concept worden beveiligd?

Usability*Methode: Literatuurstudie en het afnemen van interviews.*Literatuurstudie zal worden gedaan om te onderzoeken hoe de proof of concept gebruiksvriendelijk ontworpen kan worden. Het afnemen van interviews zal geschieden door een usability expert van Finalist te interviewen met als doel het bepalen op welke manier de inzichten van het software delivery proces het best gepresenteerd kunnen worden.

Hieronder volgen de deelvragen:

* Welke richtlijnen van usability zijn nodig voor de proof of concept?
* Hoe kunnen de inzichten van het software delivery proces het best gepresenteerd worden?

#  4 Aanpak

Voor mijn afstudeeropdracht is het van belang om een aanpak op te stellen en volgens zo’n aanpak de opdracht uit te voeren. De opdracht bestaat uit twee fases: Het onderzoek en de realisatie. Deze twee fases zijn ontstaan, omdat onderzoek moet worden gedaan en aan de hand van dit onderzoek een proof of concept moet worden gerealiseerd.

Het onderzoeksgedeelte moet antwoord geven op de deelvragen. Het onderzoek is verdeeld in drie onderdelen:

* Het software delivery proces
* Security
* Usability

Het onderzoek naar het software delivery proces heeft als doel om boven water te krijgen welke inzichten van het software delivery proces nodig zijn om dit proces inzichtelijker te maken. Als eerst is het zaak om te onderzoeken wat een software delivery proces is volgens de literatuur. Vervolgens zal door middel van het afnemen van interviews onderzoek worden gedaan naar wat een software delivery proces is bij Finalist en hoe het in elkaar zit. Daarna komt het belangrijkste gedeelte: Het onderzoeken welke inzichten nodig zijn om het software delivery proces inzichtelijker te maken. Dit zal geschieden met behulp van het afnemen van interviews. Het onderzoek naar het software delivery proces moet antwoord geven op de volgende deelvragen:

* Wat is een software delivery proces volgens de literatuur?
* Wat is een software delivery proces van Finalist en waaruit bestaat dit proces?
* Welke inzichten van het software delivery proces zijn nodig om dit proces inzichtelijker te maken?

Als het onderzoek naar het software delivery proces is afgerond zal er onderzoek worden gedaan naar security van webapplicaties. Dit onderzoek zal worden gedaan met behulp van literatuurstudie. Er zal onderzoek worden naar hoe de proof of concept kan worden beveiligd. Dit onderzoek zal in stappen worden gedaan. De deelvragen geven tevens ook de stappen weer. Dit onderzoek moet antwoord geven op de volgende deelvragen:

* Wat zijn de meest voorkomende kwetsbaarheden van webapplicaties volgens de literatuur?
* Welke onderdelen van de proof of concept moeten worden beveiligd?
* Hoe kunnen de onderdelen van de proof of concept worden beveiligd?

Het onderzoek naar usability zal worden gedaan met behulp van het bestuderen van literatuur en het afnemen van interviews met een usability expert van Finalist. Er zal onderzoek worden gedaan naar welke richtlijnen van usability voor de proof of concept van toepassing zijn en hoe deze gebruiksvriendelijk ontworpen kunnen worden. Ook zal ik door middel van het interviewen van een usability expert van Finalist, onderzoeken op welke manier de inzichten van het software delivery proces het best gepresenteerd kunnen worden. Het onderzoek naar usability moet antwoord geven op de volgende deelvragen:

* Welke richtlijnen van usability zijn voor nodig voor de proof of concept?
* Hoe kunnen de inzichten van het software delivery proces het beste gepresenteerd worden?

Als het volledige onderzoek is afgerond, zal de proof of concept aan de hand van de resultaten van het onderzoek worden gerealiseerd. Het realiseren zal verdeeld worden over een aantal sprints. Hiervoor zal de methode scrum worden gebruikt. Deze methode is een geslaagde methode die wordt gehanteerd bij het realiseren van software. De functionaliteiten zullen iteratief worden gerealiseerd, zodat de belanghebbenden tijdig feedback kunnen geven en de functionaliteiten getest kunnen worden.

**ONDERZOEK**

#  5 Software delivery proces

Het software delivery proces heeft een belangrijke positie in mijn onderzoek. Finalist wil het software delivery proces verbeteren en dit kan door dit proces inzichtelijker te maken. Ik zal in dit hoofdstuk mijn onderzoek met de daarbij behorende resultaten toelichten.

StappenplanVoor dit onderzoek heb ik een stappenplan samengesteld. Hieronder zijn deze stappen met de daarbij behorende methoden te zien.



Figuur 1: Stappenplan onderzoek software delivery proces

## 5.1 Wie zijn de belanghebbenden van het software delivery proces en van welke belanghebbende worden de wensen naar de proof of concept meegenomen?

Voor mijn onderzoek moet ik bepalen wie de belanghebbenden zijn. De reden hiervoor is dat ik dan duidelijk in kaart heb wie ik moet interviewen en welke consequenties dat voor mijn onderzoek zal hebben. In eerste instantie was het de bedoeling dat ik het onderzoek naar het software delivery proces zowel binnen Finalist als bij klanten van Finalist zou doen. Tijdens het schrijven van het plan van aanpak werd al snel duidelijk dat het niet reëel was om ook de klanten van Finalist te betrekken in het onderzoek, omdat de beschikbare tijd niet voldoende was. Samen met mijn bedrijfsbegeleider Brian Snijders heb ik besloten om het onderzoek alleen intern binnen Finalist te doen. Ook heb ik besloten om niet alle medewerkers van Finalist in het onderzoek te betrekken, maar alleen de operationele tak, omdat de operationele tak betrokken is bij het software delivery proces. De wensen van de operationele tak van Finalist zullen dan ook worden meegenomen naar de proof of concept. Dit is dan ook een verdere afbakening van mijn onderzoek ten opzichte van de afbakening in hoofdstuk 2.3.

5.2 Wat is een software delivery proces volgens de literatuur?
Ik heb onderzoek gedaan naar wat de literatuur zegt over het software delivery proces. De reden hiervoor is dat ik dan weet wat een software delivery proces is en hoe het in elkaar zit. Daarnaast ben ik dan ook beter voorbereid op mijn onderzoek dat over het software delivery proces van Finalist gaat en ook ben ik ook beter voorbereid op de interviews die ik ga afnemen.

Uit mijn onderzoek heb ik twee begrippen uit de literatuur kunnen halen die een software delivery proces beschrijven. Software deployment(Dearle, 2007) en continuous delivery(Humble & Farley, 2010; Info Support, 2014) zijn deze begrippen. Ze worden gebruikt om een software delivery proces te beschrijven.

Software deployment(Dearle, 2007) is een algemeen begrip dat wordt gebuikt voor een software delivery proces. Het begrip wordt uitgelegd als het verrichten van allerlei activiteiten, zodat software gebruikt kan worden. Zaken als het installeren, verwijderen, updaten, configureren, activeren, deactiveren, deploying en undeploying van software vallen onder dit begrip. Dearle (2007) zegt over software deployment het volgende:

“Thus deployment is a post-production activity that is performed for or by the customer of a piece of software. It is at this point in time that all customer centric customization and configuration takes place. Software deployment may be considered to be a proces consisting of a number of inter-related activities including the release of software at the end of the development cycle; the configuration of the software, the installation of software into the execution environment, and the activation of the software. It also includes post installation activities including the monitoring, deactivation, updating, reconfiguration, adaption, redeploying, and undeplying of the software.”

Continious delivery(Humble & Farley, 2007; Info Support, 2014) is een methode die wordt gebruikt om software zo snel mogelijk in productie te krijgen. Het is dusdanig ingericht dat software sneller gereleased en gedeployed kan worden. Hieronder staan een aantal citaten waarmee ik dit heb kunnen vaststellen:

* “Deze methode is er juist op gericht om nieuwe ideeën zo snel en efficiënt mogelijk in productie te krijgen. Ontwikkelen wordt een iteratief proces waarbij continu kleine stukken functionaliteit automatisch worden uitgerold. Hierdoor komen deploy-stappen vaker in het OTAP(Ontwikkel, Test, Acceptatie en Productie)-proces voor, waardoor software releases vaker plaatsvinden en functionaliteit in een doorlopend proces wordt getest en stap voor stap wordt verbeterd.”(Info Support, 2014)
* “You can only start to get a return on your investment once your software is released. So, one of our two overriding goals in this book is to find ways to reduce cycle time, the time it takes from deciding to make a change, wheter a bugfix or a feature, to having it available to users.”(Huble & Farley, 2007, p. 11)
* “The principles, practices, and techniques we describe are aimed at making releases boring, even in complex “enterprise” environments. Software release can—and should—be a low-risk, frequent, cheap, rapid, and predictable process. These practices have been developed over the last few years, and we have seen them make a huge difference.”(Humble & Farley, 2007, p. 10)

## 5.3 Wat is een software delivery proces van Finalist en waaruit bestaat dit proces?

Ik heb onderzoek gedaan naar wat een software delivery proces van Finalist is en waaruit het bestaat. De reden hiervoor is, omdat ik onderzoek doe naar het software delivery proces van Finalist met als uiteindelijke doel om dit proces inzichtelijker te maken. Om te weten wat een software delivery proces van Finalist is en waaruit het bestaat, heb ik Brian Snijders geïnterviewd. Brian Snijders is solution expert bij Finalist en is met alle onderdelen van dit proces betrokken en daarom een geschikte persoon om te interviewen. Het interview met Brian Snijders is in hoofdstuk 14 te lezen

Uit het interview met Brian Snijders is naar voren gekomen dat Finalist op een gestructureerde wijze haar projecten uitvoert. In eerste instantie komt het team samen met de klant bij elkaar om te kijken wat de wensen van de klant zijn. Het team dat in gesprek gaat met de klant bestaat uit: Een informatie analist, developers, testers, architecten en een projectmanager. Finalist vindt het belangrijk dat de klant centraal staat en kiest daarom voor deze aanpak. Als eenmaal de wensen van de klant duidelijk zijn, gaat het proces van start waarin de software voor de klant gerealiseerd gaat worden. Het opleveren van de software wordt in iteraties/sprints van twee weken gedaan. Na deze twee weken komt er een testrelease en wordt de software opgeleverd. Daarna kan de klant aangeven of het naar wens is. Als dat zo is kan de software in productie. 

Figuur 2: Lyfe cycle software delivery proces van Finalist

Het ontwikkelteam ontwikkelt en test de gerealiseerde functionaliteiten. Als de functionaliteiten worden geaccepteerd kunnen ze gereleased en gedeployed worden. Jenkins is een release tool en is verantwoordelijk voor de releases. Rundeck is een deployment tool en is verantwoordelijk voor de deployments.

Aan de hand van het interview heb ik kunnen vaststellen dat releases en deployments van software het hart van het software delivery proces van Finalist is. In hoofdstuk 5.1 heb ik aangegeven dat de wensen van de operationele tak van Finalist worden meegenomen naar de proof of concept. Door dit interview heb ik besloten om alleen wensen van de operationele tak van Finalist naar de proof of concept mee te nemen die een directe relatie hebben met het software delivery proces. Een directe relatie houdt in dat de wensen een harde raakvlak hebben met releases en deployments. Dit betekent dat alleen de wensen die met deze twee aspecten te maken hebben, naar de proof of concept worden meegenomen. Dit is een verdere afbakening van mijn onderzoek ten opzichte van hoofdstuk 2.3 en van hoofdstuk 5.1.

## 5.4 Welke inzichten zijn nodig om het software delivery proces van Finalist inzichtelijker te maken?

In dit onderdeel zal ik uiteenzetten hoe ik onderzoek heb gedaan om te onderzoeken hoe het software delivery proces van Finalist inzichtelijker kan worden gemaakt en wat de resultaten zijn.

### 5.4.1 Wie heb ik geïnterviewd Stap 4

Ik moest bepalen wie ik van Finalist zou interviewen. De reden hiervoor is dat ik dan de juiste mensen interview die mij daadwerkelijk informatie kunnen verschaffen over het software delivery proces en geen mensen interview die niks over dit proces weten of er niks mee te maken hebben. Ik heb samen met mijn bedrijfsbegeleider besloten om alleen de operationele tak van Finalist te interviewen, zij zijn direct betrokken bij dit proces. Binnen deze operationele tak bevindt zich een rangorde van medewerkers. Hierbij kun je denken aan de operationeel directeur, een unitmanager en projectleden. Dit wetende heb ik besloten om de operationeel directeur, de unitmanager en de projectleden te interviewen. Het projectteam bestaat uit developers, testers, informatie analisten, een solution expert en project managers. Ik wilde vanuit verschillende kanten van de operationele tak weten welke inzichten zij van het software delivery proces willen. Informatie analisten zijn alleen in het voortraject van projecten actief in het vertalen van klantwensen en niet daadwerkelijk met het software delivery proces. Hierdoor heb ik de keuze gemaakt om informatie analisten niet te interviewen. Om onderscheid te maken tussen de interviews heb ik besloten om de operationeel directeur en de unitmanager onder managementleden te scharen en de projectteam onder projectleden te scharen.

Voor het afnemen van de interviews heb ik vooraf bepaald dat ik de interviews opneem, zodat ik niet constant aan het schrijven ben en daardoor niet in staat ben om passende vervolgvragen te stellen. Daarnaast heb ik vooraf aan de interviews een aantal vragen opgesteld die ik wilde stellen:

* Aan welke inzichten van het software delivery proces heeft u behoefte?
* Welke irritaties zijn er van de informatie voorziening van het huidige software delivery proces?
* Is er een mogelijkheid dat door het inzichtelijker maken van het software delivery proces, Finalist daar financieel beter van wordt?
* Welke gegevens zijn er nodig van het software delivery proces?
* Hoe kan de communicatie over het software delivery proces verbeterd worden?

### 5.4.2 Resultaten van de interviews Stap 5

Ik heb de projectleden en de managementleden geïnterviewd. Ik heb voor deze methode gekozen, omdat deze twee teams mij informatie konden verschaffen over het software delivery proces. Zij zijn degene die kunnen aangeven welke inzichten zij willen, zodat het proces inzichtelijker kan worden gemaakt. Hieronder zal ik de resultaten van de interviews per team uiteenzetten.

Projectleden
Uit de interviews met de verschillende projectleden is er bij mij een beeld ontstaan dat er behoefte is aan bepaalde inzichten van het software delivery proces waarop dit proces beter gestuurd kan worden. Het interview met de solution expert was hierin doorslaggevend. Op dit moment is het software delivery proces niet inzichtelijk genoeg om het beter te kunnen analyseren en te kijken welke zaken goed gaan en welke minder goed gaan. Hierdoor is het ook niet mogelijk om het proces misschien te kunnen veranderen. Finalist wilt graag haar dienstverlening verbeteren en het software delivery proces speelt daarin een belangrijke rol. Er is aangegeven dat het nu niet mogelijk is om bepaalde beslissingen te nemen. De solution expert geeft vooral aan dat inzicht in de releases en deployments van software belangrijk zijn om het software delivery proces te verbeteren. Hierdoor zou dit proces verbeterd kunnen worden. Er is ook aangegeven dat Finalist verschillende Hippo klanten heeft en dat het niet duidelijk genoeg is hoe de verhoudingen tussen verschillende klanten zijn op het gebied van releases en deployments. Een voorbeeld is dat de releases en deployments voor verschillende klanten met elkaar kunnen worden vergeleken om te kijken waarom het misschien bij de ene klant structureel langzamer is dan bij de andere klant en hoe dat verbeterd kan worden. Er zijn meerdere aspecten waarop gestuurd kan worden.

Nadat ik de interviews heb afgenomen en uitgeschreven heb ik de interviews geanalyseerd. Door deze analyse ben ik tot de conclusie gekomen dat het projectteam behoefte heeft aan de volgende inzichten:

* Inzicht in de maandelijkse releases per klant
* Inzicht in de maandelijkse deployments per klant
* Inzicht in de langste deployment tijd per klant
* Inzicht in de kortste deployment tijd per klant
* Inzicht in de gemiddelde deployment tijd tijd per klant
* Inzicht in de langste release tijd per klant
* Inzicht in de kortste release tijd per klant
* Inzicht in de gemiddelde release tijd per klant
* Inzicht in de verhoudingen van de gemiddelde duur van de releases tussen klanten
* Inzicht in de verhoudingen van de gemiddelde duur van deployments tussen klanten

ManagementledenUit de interviews met de managementleden is er bij mij een beeld ontstaan dat er behoefte is aan inzichten die geen directe relatie hebben met het software delivery proces. Het management fungeert op een hoger niveau. Zij zijn vooral gebaat bij duidelijkheid over de kwaliteit en samenstellingen van projecten. Een voorbeeld is dat de managementleden aangeven dat ze willen weten hoe de samenstellingen van projecten zijn. Hiermee willen ze weten of projecten waar veel juniors onderdeel van zijn beter of slechter lopen dan projecten waar veel seniors onderdeel van zijn. Vervolgens zouden zij de projecten anders kunnen sturen. Als zij dan zien dat projecten waar veel juniors onderdeel van zijn net zo goed draaien als projecten waar veel seniors onderdeel van zijn, zou dat financieel voor Finalist interessant kunnen zijn. Juniors zijn immers goedkoper dan seniors en dat zou betekenen dat projectmanagers vaker projecten met veel juniors moeten samenstellen. Een ander voorbeeld is dat de managementleden inzicht willen hebben in hoe projecten lopen. Ze willen dan met name weten welke projecten er in zwaar weer verkeren. Ook willen ze weten wat de kwaliteit van mensen is, hoe mensen functioneren en hoe het met ze gaat.

Nadat ik de interviews heb afgenomen en uitgeschreven heb ik de interviews geanalyseerd. Door deze analyse ben ik tot de conclusie gekomen dat het managementteam behoefte heeft aan de volgende inzichten:

* Inzicht in de voortgang van projecten
* Inzicht in kwaliteit van mensen
* Inzicht in hoe het met mensen gaat
* Inzicht in aantal wisselingen per project
* Inzicht in hoeveel issues er worden gevonden in een release
* Inzicht in projectmarge

### 5.4.3 Lijst prioriteiten inzichten

Ik heb ervoor gekozen om een lijst van prioriteiten op te stellen. Hiermee stel ik een lijst op waarin ik aangeef welke inzichten een hoge en een lage prioriteit hebben. De reden hiervoor is dat ik dan een beter beeld heb welke functionaliteiten een hogere prioriteit hebben. Uit de interviews met de projectleden is naar voren gekomen dat zij behoefte hebben aan inzichten die een directe relatie hebben met het software delivery proces. Er is behoefte aan duidelijkheid over deployments en releases van software. De managementleden hebben behoefte aan zaken die geen directe relatie hebben met het software delivery proces, maar willen bijvoorbeeld inzicht in hoe projecten lopen, wat de kwaliteit van mensen is, hoe het met de mensen gaat en in projectmarge. In hoofdstuk 5.3 heb ik de opdracht verder afgebakend en aangegeven dat ik alleen inzichten meeneem naar de realisatie van de proof of concept die een directe relatie hebben met het software delivery proces. De inzichten waar de projectleden behoefte aan hebben zal ik meenemen naar de realisatie van de proof of concept. Hetgeen waar de managementleden behoefte aan hebben zal ik niet meenemen naar de realisatie van de proof of concept. Dit wetende kan ik nu een lijst opstellen met prioritering van de functionaliteiten die gerealiseerd moeten worden. Om iets te kwalificeren tot hoge prioriteit of lage prioriteit, heb ik Brian Snijders(Solution Expert) hierover bevraagd. Aan het eind van het interview heb ik hem gevraagd welke inzichten een hoge prioriteit en welke inzichten een lage prioriteit hadden. Zijn antwoord was als volgt: “Als ik niet heel veel inzichten heb, maar ik heb wel de basis inzichten die veilig zijn en ik kan monitoren dan is dat natuurlijk genoeg. En de basisinzichten is niet meer dan hetgeen wat ik genoemd heb. Hoeveel releases per klant doen we, hoelang duren ze gemiddeld.”. Hiermee heeft hij aangegeven dat er geen onderscheid is tussen de verschillende functionaliteiten, maar in een later stadium heb ik dit nog een keer voorgelegd en heeft hij alsnog kunnen aangeven welke zaken een hoge prioriteit hebben. Inzicht in deployments heeft een hogere prioriteit dan inzicht in releases, omdat deployments gefactureerd kunnen worden naar de klant en releases niet. Mede hierdoor heb ik een lijst met prioriteiten kunnen vaststellen. Hieronder is deze te zien.

Hoge prioriteit

* Inzicht in de langste deployment tijd per klant
* Inzicht in de kortste deployment tijd per klant
* Inzicht in de gemiddelde deployment tijd per klant
* Inzicht in de maandelijkse deployments per klant
* Inzicht in de verhoudingen van de gemiddelde duur van deployments tussen klanten

Lage Prioriteit

* Inzicht in de langste release tijd per klant
* Inzicht in de kortste release tijd per klant
* Inzicht in de gemiddelde release tijd per klant
* Inzicht in de maandelijkse releases per klant
* Inzicht in de verhoudingen van de gemiddelde duur van releases tussen klanten

## 5.5 Conclusie

Ik heb onderzoek gedaan naar wat de theorie zegt over een software delivery proces. Hieruit is voortgekomen dat software deployment en continuous delivery begrippen zijn die een software delivery proces beschrijven. Vervolgens heb ik onderzoek gedaan naar hoe het software delivery proces binnen Finalist in elkaar zit. Hierbij heb ik geconcludeerd dat het software delivery proces van Finalist het meest overeenkomt met continuous delivery, omdat de manier waarop software wordt opgeleverd binnen Finalist in grote lijnen hetzelfde overeenkomt.

Voor het afnemen van de interviews heb ik vooraf aan de interviews een aantal vragen opgesteld. Deze vragen heb ik niet allemaal gesteld, omdat gezien de interactie van het interview, het interview anders verliep dan ik vooraf voorspelde. Ik heb wel voor elke geïnterviewde functionaris van Finalist de vragen gesteld die voor die functie relevant zijn. Uit de interviews met de projectleden en de managementleden is naar voren gekomen dat de projectleden behoefte hebben aan inzichten van het software delivery proces om dit proces te beter te kunnen sturen. Deze hebben een directe relatie met het software delivery proces, omdat hun wensen met releases en deployments van software te maken hebben. De managementleden hebben behoefte aan inzichten die geen directe relatie hebben met het software delivery proces, maar willen bijvoorbeeld inzicht in hoe projecten lopen, wat de kwaliteit van mensen is, hoe het met de mensen gaat en projectmarge.

Ik ben tot de conclusie gekomen dat alleen de inzichten waar de projectleden behoefte aan hebben meegenomen zullen worden naar de realisatie van de proof of concept. De inzichten waar de managementleden behoefte aan hebben zullen niet worden meegenomen bij het realiseren van de proof of concept, omdat deze geen directe relatie hebben met het software delivery proces. Verder ben ik tot de conclusie gekomen dat de inzichten over deployments een hogere prioriteit hebben dan releases, omdat deze in tegenstelling tot releases uiteindelijk bijvoorbeeld gefactureerd kunnen worden naar de klant. Hieronder is te zien welke inzichten mee worden genomen naar de proof of concept en welke een hoge prioriteit hebben.

Hoge prioriteit

* Inzicht in de kortste deployment tijd per klant
* Inzicht in de gemiddelde deployment tijd per klant
* Inzicht in de langste deployment tijd per klant
* Inzicht in maandelijkse deployments per klant
* Inzicht in de verhoudingen van de duur van deployments tussen klanten

Lage Prioriteit

* Inzicht in de kortste release tijd per klant
* Inzicht in de gemiddelde release tijd per klant
* Inzicht in de langste release tijd per klant
* Inzicht in maandelijkse releases per klant
* Inzicht in de verhoudingen van de duur van releases tussen klanten

#  6 Security

In hoofdstuk 2.1 is vermeldt dat de proof of concept online zal draaien. Dit heeft als consequentie dat er onderzoek moet worden gedaan naar veiligheid van webapplicaties. In dit hoofdstuk zal ik dit uiteenzetten.

Stappenplan

Figuur 3: Stappenplan onderzoek security

De oriëntatiefase heb ik voornamelijk gebruikt om criteria voor mijn onderzoek vast te kunnen stellen. Voor stap 2, 3 en 4 heb ik een aantal criteria opgesteld. In figuur 5 zijn deze criteria te zien.


Figuur 4: Criteria voor stap 2,3 en 4 uit stappenplan onderzoek security

Ik wil hierbij toelichten dat ik in de criteria voor het beveiligen van de kwetsbaarheden die relevant zijn voor de proof of concept, heb opgenomen dat het verenigbaar moet zijn met de programmeertaal JAVA. Dit heb ik gedaan, omdat ik de proof of concept met JAVA wil realiseren. Als blijkt dat het niet kan dan moet ik een andere programmeertaal gebruiken voor de realisatie. Tevens wil ik aangeven dat als er geen technieken als oplossingen worden geboden, maar algemene richtlijnen voor de beveiliging van een bepaalde kwetsbaarheid dan gelden deze criteria niet en dienen de richtlijnen te worden gehanteerd.

## 6.1 Wat zijn de meest voorkomende kwetsbaarheden van webapplicaties?

Ik heb onderzoek gedaan naar de meest voorkomende kwetsbaarheden van webapplicaties. Dit heb ik gedaan, zodat ik over genoeg kennis beschik om mijn proof of concept te beveiligen tegen deze kwetsbaarheden. Ik heb dit gedaan door middel van literatuurstudie. Hierbij heb ik gekeken welke kwetsbaarheden door de verschillende bronnen als meest voorkomende kwetsbaarheid werden gekwalificeerd. Uit mijn onderzoek(OWASP, 2013; Edgescan, 2014; Kennedy, 2005; Cenzic, 2014; Landsman, 2010) komt naar voren dat er een aantal kwetsbaarheden tot de meest voorkomende kwetsbaarheden kunnen worden gekwalificeerd.

Cross site scripting is een kwetsbaarheid van webapplicaties die door alle bronnen beschouwd wordt als veel voorkomende kwetsbaarheid van webapplicaties. SQL injection is een andere kwetsbaarheid van webapplicaties en ook deze wordt door alle bronnen beschouwd als veel voorkomende kwetsbaarheid van webapplicaties. Bij het analyseren heb ik ook kunnen vaststellen dat Broken Authentication and Session Management wordt beschouwd als veel voorkomende kwetsbaarheid. Onder deze categorie vallen zaken(OWASP, 2013; Edgescan, 2014; Kennedy, 2005; Cenzic, 2014; Landsman, 2010) als: Authentication, authorisation en session management. Sommige bronnen groepeerden bepaalde kwetsbaarheden en schaarden ze onder één naam, andere bronnen noemden de kwetsbaarheden los van elkaar. Ik heb ervoor gekozen om deze kwetsbaarheden samen te voegen en ze onder het begrip ‘Broken Authentication and Session Management’ te scharen. De reden hiervoor is, omdat dat mij meer structuur zal geven bij het kunnen analyseren van de kwetsbaarheden. Broken Authentication and Session Management komt dus ook op de lijst met de meest voorkomende kwetsbaarheden.

Uit mijn onderzoek(OWASP, 2013; Edgescan, 2014; Kennedy, 2005; Cenzic, 2014; Landsman, 2010) concludeer ik dat er drie kwetsbaarheden op de lijst van meest voorkomende kwetsbaarheden komen:

* SQL injection
* Broken Authentication and Session Management
* Cross Site Scripting

## 6.2 Welke onderdelen van de proof of concept moeten beveiligd worden?

Ik heb geanalyseerd en onderzoek gedaan naar welke onderdelen ik van de proof of concept moet beveiligen. Om dit te bereiken heb ik literatuurstudie gedaan. Een kwetsbaarheid uit de lijst van meest voorkomende kwetsbaarheden moet een duidelijke raakvlak hebben met de proof of concept om te kunnen besluiten dat een bepaald onderdeel beveiligd moet worden. Uit mijn onderzoek(Keary, 2015; Siles, 2014; Kennedy, 2005; Landsman, 2010, Stuttard & Pinto, 2011) komt naar voren dat Broken Authentication and Session Management, SQL injection en Cross Site Scripting een duidelijke raakvlak hebben met de proof of concept. Hieronder zal ik voor ieder kwetsbaarheid aangeven hoe ik dit heb kunnen vaststellen.

Broken Authentication and Session ManagementUit mijn onderzoek(Keary, 2015; Siles, 2014; Kennedy, 2005; Landsman, 2010, Stuttard & Pinto, 2011; OWASP, 2013) blijkt dat het onderwerp Broken Authentication and Session Management een raakvlak heeft met het onderzoek, omdat inzichten van het software delivery proces beveiligd gepresenteerd moeten kunnen worden en niet iedereen toegang mag hebben tot deze inzichten. Er zijn een aantal bewijsstukken uit mijn onderzoek(Keary, 2015; Siles, 2014; Kennedy, 2005; Landsman, 2010, Stuttard & Pinto, 2011; OWASP, 2013), waardoor ik heb kunnen vaststellen dat Broken Authentication and Session Managenment een raakvlak heeft mijn onderzoek. Hieronder zijn een aantal citaten hiervan te zien:

* “One of the biggest weaknesses is the failure to provide a means of strong authentication to verify the end user whom he/she claims”(Kennedy, 2005).
* “Without such authentication employed, attackers could acces another user’s account, view sensitive information or perform unauthorizid functions”(Kennedy, 2005).
* “Authentication also lies at the heart of an application’s protection against malicious attack”(Stuttard & Pinto, 2011).
* “Attacker uses leaks or flaws in the authentication or session management functions(e.g., exposed accounts, passwords, session IDs) to impersonate users”(OWASP, 2013)

SQL injectionIk heb de keuze gemaakt om gegevens in een database op te slaan, omdat voor het uitvoeren van berekeningen en het kunnen presenteren van gegevens een database de meest geschikte opslagplaats is. Uit mijn onderzoek(Wichers, Manico & Seil, 2015; OWASP, 2013; Kennedy, 2005; Landsman, 2010; Stuttard & Pinto, 2011) komt naar voren dat webapplicaties die een database als opslagbron gebruiken vatbaar kunnen zijn voor SQL injection. Er zijn een aantal bewijsstukken uit mijn onderzoek(Wichers, Manico & Seil, 2015; OWASP, 2013; Kennedy, 2005; Landsman, 2010; Stuttard & Pinto, 2011), waardoor ik heb kunnen vaststellen dat SQL injection uitgevoerd kan worden op een webapplicatie, waardoor de onderliggende database ongewild benaderd kan worden. Dit is een duidelijke raakvlak met de proof of concept, omdat ik een database als opslagbron ga gebruiken. Hieronder zijn een aantal citaten te zien, die mij hebben doen overtuigen dat een database vatbaar is voor SQL injection:

* “An SQL injection is a specific type of code injection in which the web application’s database is the target of the attack”(Landsman, 2010)
* “SQL Injections are the same as a code injection except that the target of the attack is the database”(Landsman, 2010)
* “In the most serious cases, SQL injection can enable an anonymous attacker to read and modify all data stored wihtin the database”(Stuttard & Pinto, 2011).

Cros Site ScriptingEen onderdeel van de proof of concept is het presenteren van inzichten, oftewel een client. Uit mijn onderzoek(Wiliams, Manico & Mattatall, 2014; OWASP, 2013; Kennedy, 2005; Landsman, 2010; Stuttard & Pinto, 2011) komt naar voren dat hackers client-side aanvallen kunnen uitvoeren op webapplicaties. Hieronder zijn een aantal citaten te zien die mij hebben doen overtuigen dat de proof of concept vatbaar zou kunnen zijn voor cross site scripting:

* “The attacker may modify the web file code on the host server to include Active-X controls, redirection methods, iFramed content or other attack types”(Landsman, 2010).
* “Attackers can execute scripts in a victim’s browser to hijack user sessions”(OWASP, 2013)

Een interface zal onderdeel van de proof of concept zijn, wat in andere woorden een client is. Het feit dat een hacker client-side aanvallen kan uitvoeren is een duidelijke raakvlak met de proof of concept, omdat ik voor de proof of concept een interface ga ontwerpen die inzichten van het software delivery proces moet weergeven.

Uit mijn onderzoek(OWASP, 2013; Edgescan, 2014; Kennedy, 2005; Cenzic, 2014; Landsman, 2010; Stuttard & Pinto, 2011) concludeer ik dat het voor de proof of concept noodzakelijk is om de volgende onderdelen te moeten beveiligen:

* Broken Authentication and session management
* SQL injection
* Cross Site Scripting

## 6.3 Hoe kunnen de onderdelen van de proof of concept beveiligd worden?

Ik heb onderzoek gedaan naar hoe ik de inzichten van het software delivery proces veilig kan presenteren. Dit heb ik gedaan, omdat ik de proof wil beveiligen tegen aanvallen van hackers. Met behulp van literatuurstudie heb ik oplossingen kunnen vinden om mijn proof of concept te kunnen beveiligen. In het vorige onderdeel is aangegeven dat de proof of concept beveiligd moet worden tegen: ***Broken authentication***, ***SQL injection*** en ***Cross Site Scripting***. Voor ieder kwetsbaarheid zal worden uiteengezet hoe de proof of concept er tegen beveiligd kan worden.

### ***6.3.1 SQL injection***

Uit mijn onderzoek(Stuttard & Pinto, 2011; Berkeley Security, z.d.; Sans Software Security; Wichers, Manico & Seil, 2014) zijn drie potentiële technieken naar voren gekomen die SQL injection kunnen voorkomen. Het gebruik van prepared statements, Stored Procedures en invoer valideren zijn deze technieken. Ik zal per techniek toelichten wat mij er toe heeft bewogen om deze technieken als potentiële technieken te kwalificeren.

Prepared StatementEen prepared statement is een techniek waarbij aan een SQL query parameters worden meegegeven. De ontwikkelaar wordt gedwongen om eerst de query te formuleren en vervolgens parameters daar aan mee te geven.

Uit mijn onderzoek(Stuttard & Pinto, 2011; Berkeley Security, z.d.; Jardin, z.d.; Wichers, Manico & Seil, 2014) is naar voren gekomen dat het gebruik van prepared statements een techniek is die SQL injection kan voorkomen. Er zijn een aantal bewijsstukken die mij hebben overtuigd om deze techniek als potentieel te kwalificeren:

* “Prepared statements ensure that an attacker is not able to change the intent of a query, even if SQL commands are inserted”(Wichers, Manico & Seil, 2014)
* “The purpose of a parametherized query is to allow the data source to be able to distinguish executable statements from untrusted data”(Jardin, z.d.)
* “Developers can prevent SQL injection vulnerabilities in web applications by utilizing parametherized database queries”(Berkeley University, z.d.)

Stored ProceduresEen stored procedure is een techniek waarbij SQL code voor het ophalen of bewerken van gegevens uit de database, niet in de applicatie zelf wordt geformuleerd, maar in een programma dat bewaard wordt in een database.

Uit mijn onderzoek(Stuttard & Pinto, 2011; Wichers, Manico & Seil, 2014) is naar voren gekomen dat het gebruik van stored procedures een SQL injection kan voorkomen. Er zijn een aantal bewijsstukken die mij hebben overtuigd om het gebruik van stored procedures als een potentiële techniek te kwalificeren:

* “Stored procedures have the same effect as prepared statements when implemented safely”(Wichers, Manico & Seil, 2014).
* “There is no doubt that customer stored procedures can provide security and performance benefits”(Stuttard & Pinto, 2011)

Invoer validerenInvoer valideren is een techniek waarbij alle invoer gefilterd wordt voordat de query wordt uitgevoerd.

Uit mijn onderzoek(Stuttard & Pinto, 2011; Wichers, Manico & Seil, 2014; Landsman, 2010) is naar voren gekomen dat het valideren van invoer ook als mogelijke techniek wordt aangeduid voor het voorkomen van SQL injection. Er zijn een aantal bewijsstukken die mij hebben doen overtuigen om deze techniek als potentiële techniek te kwalificeren.

* “Employing Data Validation and Data Cleansing techniques from above, SQL injection may be easily mediated”(Landsman, 2010)
* “This third technique is to escape user input before putting it in a query”( Wichers, Manico & Seil, 2014)

In dit geval zijn er meerdere potentiële oplossingen. Dit betekent dat deze oplossingen met elkaar moeten worden vergeleken om de meest geschikte oplossing toe te passen. Zoals aan het begin van het hoofdstuk is vermeld, zijn er een aantal criteria waaraan ik kan toetsen welke oplossing het meest geschikt is. Hieronder zal ik de criteria herhalen:

* De techniek die als oplossing wordt geboden moet voorkomen dat de proof of concept vatbaar is voor de kwetsbaarheid.
* De techniek moet verenigbaar zijn met de programmeertaal JAVA.
* De techniek moet makkelijk toe te passen zijn.

Hieronder zullen de oplossing aan de hand van deze criteria met elkaar worden vergeleken.

Tabel 1: Vergelijking technieken voor het voorkomen van SQL injection

|  |  |  |  |
| --- | --- | --- | --- |
|  | Prepared Statements | Stored Procedures | Invoer valideren |
| **Applicatie kan volledig worden beschermd tegen een SQL injection** | Ja, alle bronnen zijn het er over eens dat prepared statements SQL injections voorkomen | Ja, als er geen gebruik wordt gemaakt van dynamic SQL. Als er gebruik wordt gemaakt van dynamic SQL dan moet de invoer ook gevalideerd worden.  | Nee. Een groot gedeelte van SQL injection aanvallen kunnen met deze techniek worden voorkomen, maar niet alle SQL injection aanvallen kunnen worden voorkomen. |
| **Kan toegepast worden binnen een applicatie dat gebouwd is in de programmeertaal JAVA** | Ja | Ja | Ja |
| **De techniek is makkelijk te begrijpen en makkelijk toe te passen** | Ja. Van alle drie de technieken is het gebruik van Prepared Statements het makkelijkst toe te passen en het makkelijkst te begrijpen. Aan een prepared stament hoef je alleen parameters mee te geven. Het is hetzelfde werk als het schrijven van een SQL query. Het enige wat je moet meegeven aan de query zijn de parameters. | Stored procedures zijn files die in de database worden bewaard en worden geformuleerd. Vanuit de code wordt zo’n stored procedure aangeroepen. Aan zo’n stored procedure moeten ook parameters worden meegegeven.Het kost dus meer tijd om het toe te passen dan prepared statements. Het kost extra werk. Er moet een file aangemaakt worden en geformuleerd worden. Vervolgens dient deze ook vanuit de code aangeroepen te worden. | Ja, de techniek is redelijk makkelijk toe te passen, maar er komt wel extra werk bij kijken. Je kan een filter schrijven om de invoer te valideren. In dat geval kost het dus meer tijd. Het is ook mogelijk om een bestaande filter te gebruiken, maar dat aan die filter het één en ander aan toegevoegd dient te worden.  |

Ik concludeer uit de vergelijking dat het gebruik van prepared statements volgens de criteria die ik heb opgesteld, het sterkst scoort. Het gebruik van stored procedures is ook een techniek die SQL injection kan voorkomen, maar in vergelijking met het gebruik van prepared statements moeilijker te gebruiken is en meer werk kost om het toe te kunnen passen. Invoer valideren is niet geschikt om te gebruiken, omdat niet alle SQL injections kunnen worden voorkomen.

### ***6.3.2 Broken Authentication and Session Management***

Uit mijn onderzoek(Stuttard & Pinto, 2011; Siles, 2014; Keary, 2015; Landsman, 2010) naar hoe Broken Authentication voorkomen kan worden is naar voren gekomen dat het niet het geval is dat er meerdere technieken zijn die met elkaar vergeleken kunnen worden, maar dat er een aantal richtlijnen zijn die in acht dienen te worden genomen om de proof of concept te beveiligen tegen Broken Authentication and Session Management.

Uit mijn onderzoek(Stuttard & Pinto, 2011; Siles, 2014; Keary, 2015; Landsman, 2010) is naar voren gekomen dat sommige aspecten met bepaalde geavanceerde technieken en hulpmiddelen beveiligd kunnen worden, maar ook dat door het nemen van simpele maatregelen de veiligheid van een applicatie vergroot kan worden. Door het onderzoek heb ik kunnen vaststellen dat er drie belangrijke aspecten voor de proof of concept van belang zijn om Broken Authentication te voorkomen: Het beveiligen van wachtwoorden, het beveiligen van session Ids/cookies en het beveiligen van het versturen van gegevens.

Voor elk onderdeel zullen aspecten worden opgenoemd die in acht dienen te worden genomen.

##### Wachtwoorden

* Een gebruiker dient verplicht te worden om een lang wachtwoord met verschillende karakters aan te maken. Een lang wachtwoord met verschillende karakters zorgt er immers voor dat het moeilijker te raden is. Onder een lang wachtwoord wordt verstaan een wachtwoord met minimaal tien karakters. Daarnaast moet het wachtwoord minimaal één hoofdletter, minimaal één kleine letter, minimaal één cijfer en minimaal één speciale karakter bevatten.
* De gebruiker de mogelijkheid bieden om zijn wachtwoord te kunnen veranderen.
* Wachtwoorden moeten op een veilige manier worden opgeslagen. De wachtwoorden moeten gehashed opgeslagen worden in de database.
* Als de inloggegevens niet kloppen mag het systeem niet aangeven of de gebruikersnaam of het wachtwoord foutief is.

##### Session Ids/cookies

* Een session id/cookie moet lang zijn en moeilijk te raden zijn.
* Een session moet van een bepaalde duur zijn voorzien.
* Een session id/cookie mag niet in een URL te zien zijn.

Gegevens versturenOm te kunnen bepalen welke methode ik moet gebruiken bij het versturen van gegevens heb ik een aantal criteria opgesteld die bij deze methodes voorkomen. Om vervolgens een waarde oordeel te kunnen vellen over welke methode het meest geschikt is, heb ik de twee methodes aan de hand van deze criteria met elkaar vergeleken. Hieronder is de tabel te zien.

Tabel 2: Vergelijking GET en POST methode

|  |  |  |
| --- | --- | --- |
|  | **GET** | **POST** |
| **Parameters** | Bij het verzenden van gegevens zijn parameters in de URL te zien en de parameters worden onthouden door de browser.  | Bij het verzenden van gegevens zijn parameters niet in de URL te zien en de parameters worden niet onthouden door de browser |
| **Cache** | Requests kunnen worden gecached. | Requests worden niet gecashed. |
| **Gegevens** | Zijn voor iedereen te zien in URL. | Zijn niet voor iedereen te zien in URL. |
| **Veiligheid** | Niet veilig voor het vesturen van gevoelige informatie, omdat wachtwoorden in de URL zijn te zien. | Veilig voor het vesturen van gegevens, omdat wachtwoorden niet te zien zijn in de URL. |

Uit de tabel is te zien dat voor de veiligheid van de proof of concept, bij het versturen van gegevens gebruik moet worden gemaakt van de POST methode. De GET methode is een onveilige methode, omdat in tegenstelling tot de POST methode gegevens gecashed worden en gegevens in de URL te zien zijn.

### 6.3.3 Cross Site Scripting

Uit mijn onderzoek(Stuttard & Pinto; Wiliams, Manico, & Mattatall, 2014; Kennedy, 2005; Landsman, 2010) naar hoe ik de proof concept kan beveiligen tegen cross site scripting, is naar voren gekomen dat alle invoer goed gevalideerd moet worden en ook dat de uitvoer gedecodeerd moet worden. Hackers zijn in staat om met behulp van HTML elementen en Javascript element aanvallen uit te voeren.

Uit mijn onderzoek(Stuttard & Pinto; Wiliams, Manico, & Mattatall, 2014; Kennedy, 2005; Landsman, 2010) concludeer ik dat de proof of concept tegen Cross Site Scripting beveiligd kan worden door de invoer goed te valideren en de uitvoer te decoderen.

## 6.4 Conclusie

Ik heb onderzoek gedaan naar wat de meest voorkomende kwetsbaarheden van webapplicaties zijn en een lijst met de meest voorkomende kwetsbaarheden samengesteld. Vervolgens heb ik onderzoek gedaan naar welke onderwerpen uit deze lijst voor de veiligheid van de proof of concept van belang zijn. Uit deze analyse heb ik geconcludeerd dat: Broken Authenticationand Session Management, SQL injection en Cross Site Scripting de onderdelen zijn die ik moet onderzoeken.

Om te bepalen met welke technieken SQL injection kan worden voorkomen, heb ik drie potentiële technieken samengesteld. Uit het onderzoek van deze technieken heb ik geconcludeerd dat ik voor het realiseren van de proof of concept prepared statements moet gebruiken, omdat deze techniek een applicatie beschermd tegen SQL injection en, omdat in tegenstelling tot de andere technieken deze makkelijk te gebruiken is.

Om te bepalen hoe de proof of concept beveiligd kan worden tegen Broken Authentication, zijn er drie belangrijke onderdelen: Het beveiligen van wachtwoorden, het beveiligen van session ids/cookies en het beveiligen van het versturen van gegevens. Uit het onderzoek heb ik vastgesteld dat wachtwoorden lang moeten zijn en uit verschillende karakters moeten bestaan. Gebruikers moeten hun wachtwoord kunnen wijzigen en wachtwoorden moeten gehashed in de database worden opgeslagen. Session ids moeten lang zijn, een tijdslimiet hebben en mogen niet in een URL te zien zijn. Voor het versturen van gegevens moet de POST methode worden gebruikt, omdat deze methode in tegenstelling tot de GET methode wel veilig is.

Om te bepalen hoe cross site scripting voorkomen kan worden, heb ik uit onderzoek vastgesteld dat de proof of concept tegen client-side aanvallen beveiligd moet worden. Uit mijn onderzoek heb ik geconcludeerd dat ik de proof of concept kan beveiligen door de invoer goed te valideren en de uitvoer te decoderen.

#  7 Usability

De proof of concept zal bestaan uit inzichten van het software delivery proces. Deze inzichten moeten gebruiksvriendelijk gepresenteerd worden, zodat de gebruikers het makkelijk kunnen gebruiken. Dit is tijdens een interview met Brian Snijders ook nog eens beaamd. Hij zegt hierover: Het succes van de oplossing ligt in het feit hoe veilig en gebruiksvriendelijk het is. Hoe snel mensen wegwijs worden binnen zo’n applicatie. In dit onderdeel zal ik uiteenzetten hoe ik onderzoek heb gedaan en wat hieruit is voorgekomen.

Stappenplan

Figuur 5: Stappenplan onderzoek usability

De oriëntatiefase heb ik voornamelijk gebruikt om criteria voor mijn onderzoek vast te kunnen stellen. Uit die oriëntatiefase kreeg ik als snel het idee dat er richtlijnen zijn en niet zozeer meerdere oplossingen of technieken die met elkaar vergeleken kunnen worden.

CriteriaVoor het kunnen bepalen van welke richtlijnen van usability nodig zijn voor de proof of concept, is als criteria opgenomen dat een usability richtlijn een bepaalde raakvlak moet hebben met de proof of concept.

## 7.1 Welke richtlijnen van usability zijn nodig voor de proof of concept?

Ik heb onderzoek gedaan naar welke richtlijnen van usability nodig zijn voor de proof of concept. Dit heb ik gedaan, omdat er een grote kans zou bestaan dat ik niet alle richtlijnen van usability nodig heb, maar een aantal. Om dit te kunnen doen heb ik geanalyseerd hoe mijn proof of concept er ongeveer uit zou kunnen komen te zien, ik had immers al de operationele tak van Finalist geïnterviewd. Uit mijn eigen analyse ben ik tot de conclusie gekomen dat mijn proof of concept uit de volgende onderdelen zou kunnen bestaan:

* *Het voor een gebruiker mogelijk maken dat hij kan aangeven om gegevens of data in te kunnen zien.*
* *Het in statistiek verband weergegeven van releases en deployments en het weergeven van lijsten of tabellen waarin gegevens van releases en deployments verwerkt zijn.*
* *Een element om te kunnen navigeren door de webapplicatie.*

Aan de hand van deze analyse heb ik een voorselectie gemaakt om gericht onderzoek te doen naar uasbility richtlijnen die een bepaalde raakvlak hadden met deze onderdelen van de proof of concept. Mijn onderzoek(Krug, 2006; Kassenaar, 2010) geeft aan dat de volgende usability richtlijnen voor mijn proof of concept van belang zijn:

1. *Duidelijk maken waar een gebruiker op kan klikken.*
2. *Het creëren van een duidelijke visuele hiërarchie.*
3. *De navigatie element moet op elke pagina op dezelfde plek komen te staan en moet op elke pagina dezelfde uiterlijk hebben.*
4. *De naamgeving van een item van de navigatie element moet naar de bestemming leiden die door de naamgeving is aangegeven.*
5. *De navigatie element moet een home item bevatten om vanuit elke pagina terug te kunnen keren naar het hoofdscherm.*
6. *Minimaliseren van ruis*
7. *Het gebruik maken van conventies*

Het minimaliseren van ruis en het gebruik van conventies zijn twee richtlijnen die voor iedere website gelden. Het minimaliseren van ruis houdt in dat de pagina niet onoverzichtelijk wordt, doordat een webpagina onnodige plaatjes en tekst bevat die een gebruiker af kan leiden van het doel van de webpagina.

Het gebruik van conventies houdt in dat voor de proof of concept zaken worden toegepast waar mensen mee bekend zijn. Een voorbeeld hiervan is een kopje dat boven een stuk tekst staat. Dit kopje geeft aan waar de onderliggende tekst over zal gaan en de gebruiker kan dan gelijk een onderscheid maken tussen de onderdelen van een webpagina die hij wel of niet wil bekijken.

Hieronder zal ik voor elke onderdeel waarvan ik heb aangegeven dat het mogelijk onderdeel van mijn proof of concept is, aangeven welke usability richtlijnen een raakvlak hiermee hebben en waarom.

Het voor een gebruiker mogelijk maken dat hij kan aangeven om gegevens of data te kunnen bekijken.Duidelijk maken waar een gebruiker op kan klikken is de usability richtlijn die onder dit onderdeel valt.Websites bieden gebruikers de mogelijkheid om ergens op te kunnen klikken, zodat ze kunnen aangeven of ze bepaalde gegevens kunnen inzien of bewerken. Dit is dan ook een duidelijke raakvlak, omdat voor de proof of concept ik de gebruiker de mogelijkheid ga bieden om aan te geven welke gegevens hij wilt inzien. Om dit beter te illustreren is hieronder een afbeelding te zien.



Figuur 6: Duidelijk maken wat klikbaar is

Hierboven zijn drie manieren te zien waarop gepresenteerd kan worden dat een gebruiker ergens op kan klikken. De eerste twee zijn onduidelijk en zullen de gebruiker in verwarring brengen. De laatste laat wel duidelijk zien dat er op geklikt kan worden en dat het een knop is.

##### Het in statistiek verband weergegeven van releases en deployments en het weergeven van lijsten of tabellen waarin gegevens van releases en deployments weergegeven kunnen worden.

De richtlijnen die met dit onderdeel een raakvlak hebben zijn:

* *Het creëren van een duidelijke visuele hiërarchie.*

De reden waarom ik vind dat dit een raakvlak heeft met de proof of concept is, omdat het mogelijk is dat in mijn proof of concept bepaalde statistieken of gegevens een belangrijkere functie hebben en aan de hand hiervan ik kan bepalen welke elementen van de webpagina hoger of lager moeten komen te staan. Daarnaast moet het ook duidelijk zijn welke elementen van een webpagina bij elkaar horen.

Om een duidelijke visuele hiërarchie te creëren dienen de volgende zaken in acht te worden genomen:

* Belangrijke onderdelen staan hoger op een pagina, de koppen ervan zijn groter of vallen meer op.
* Onderdelen die bij elkaar horen bevinden zich in een afgebakend gebied en hebben dezelfde visuele stijl.

Een element om te kunnen navigeren door de webapplicatieDe richtlijnen die met dit onderdeel een raakvlak hebben zijn:

* *De navigatie element moet op elke pagina op dezelfde plek komen te staan en moet op elke pagina dezelfde uiterlijk hebben.*
* *De naamgeving van een item van de navigatie element moet naar de bestemming leiden die door de naamgeving is aangegeven.*
* *De navigatie element moet een home item bevatten om vanuit elke pagina terug te kunnen keren naar het hoofdscherm.*

Al deze richtlijnen hebben een duidelijke raakvlak met dit onderdeel, omdat ze aangeven hoe een navigatie element door de applicatie heen moet functioneren. Als mogelijke navigatie element van de proof of concept kan ik een menu gebruiken.

## 7.2 Hoe kunnen de inzichten het beste gepresenteerd worden?

Ik heb onderzoek gedaan naar hoe de inzichten die uit de interviews naar voren zijn gekomen het beste gepresenteerd kunnen worden. Dit heb ik gedaan, omdat ik wil dat gebruikers de inzichten op de best mogelijke manier kunnen bekijken. Dit onderzoek heb ik gedaan door Joe Tran(usability expert, Finalist) te interviewen, het interview is te lezen in hoofdstuk 14. Hierbij heb ik de inzichten die nodig zijn om het software delivery proces inzichtelijker te maken erbij gepakt en aan de usability expert voorgelegd. Hieronder zijn deze inzichten te zien:

1. Inzicht in de maandelijkse releases per klant
2. Inzicht in de maandelijkse deployments per klant
3. Inzicht in de langste deployment tijd per klant
4. Inzicht in de kortste deployment tijd per klant
5. Inzicht in de gemiddelde deployment tijd per klant
6. Inzicht in de langste release tijd per klant
7. Inzicht in de kortste release tijd per klant
8. Inzicht in de gemiddelde release tijd per klant
9. Inzicht in de verhoudingen van de gemiddelde duur van de releases tussen klanten
10. Inzicht in de verhoudingen van de gemiddelde duur van deployments tussen klanten

Bij het voorleggen van de inzichten aan de usability expert, heeft hij kunnen vast stellen dat de inzichten in twee onderdelen opgedeeld konden worden: Inzichten die gegevens moeten weergeven en inzichten die verhoudingen moeten weergeven. Hieronder is te zien welke inzichten hij tot welke onderdeel kwalificeerde.

Tabel 3: Kwalificatie van de inzichten door de usability expert

|  |  |
| --- | --- |
| **Data** | **Verhoudingen** |
| 1. Inzicht in de maandelijkse releases per klant
2. Inzicht in de maandelijkse deployments per klant
 | 1. Inzicht in de langste deployment tijd per klant
2. Inzicht in de kortste deployment tijd per klant
3. Inzicht in de gemiddelde deployment tijd per klant
4. Inzicht in de langste release tijd per klant
5. Inzicht in de kortste release tijd per klant
6. Inzicht in de gemiddelde release tijd per klant
7. Inzicht in de verhoudingen van de gemiddelde duur van de releases tussen klanten
8. Inzicht in de verhoudingen van de gemiddelde duur van deployments tussen klanten
 |

Aan de hand van deze kwalificatie heeft de usability expert mij vervolgens uitgelegd hoe ik de inzichten het best kan presenteren. Ik zal hieronder voor elke soort inzicht uitleggen wat de resultaten van het interview zijn.

Inzicht in maandelijkse releases en deployments (inzichten: 1 en 2)Uit het interview is naar voren gekomen dat de inzichten waarbij de maandelijkse deployments en
releases moeten worden weergegeven, het beste weergegeven kunnen worden in een tabel vorm. Hierbij moet er voor de releases een apart tabel en voor de deployments een apart tabel worden gemaakt. De usability expert heeft vervolgens ook aangegeven dat het wellicht ook handig zou zijn om de twee tabellen naast elkaar te plaatsen, zodat deze voor een gebruiker in één oogopslag te zien zijn.

Inzicht in de langste, kortste en gemiddelde duur van releases en deployments
(inzichten: 3 t/m 8)
De inzichten waarbij de langste, kortste en gemiddelde duur van releases en deployments per klant moeten worden weergeven, kunnen het beste in een grafiek weergeven worden. Voor deployments moet een grafiek en releases een aparte grafiek getoond worden. Hierbij dienen de kortste, gemiddelde en langste duur van releases van een klant in de grafiek meetbaar te zijn. Voor de volgorde van de weergave van deze drie items geeft de usability expert aan dat ze het beste in de volgende volgorde weergegeven kunnen worden: De kortste duur aan de meest linkerzijde, vervolgens de gemiddelde duur in het midden en aan de meest rechter zijde de langste duur. Hieronder is te zien hoe een grafiek er ongeveer uit moet komen te zien.



Figuur 7: Voorbeeld grafiek voor deployments of releases waarin de kortste, langste en gemiddelde duur van een deployment of release is verwerkt

Inzicht in de verhoudingen van releases en deployments tussen verschillende klanten
(inzichten: 9 en 10)
Voor de inzichten waarbij de verhoudingen van releases en deployments tussen verschillende klanten moeten worden weergegeven is naar voren gekomen dat deze het beste in een grafiek kunnen worden weergegeven. Voor zowel releases als deployments moet een aparte grafiek komen. Daarnaast heeft de usability expert aangegeven dat als er veel klanten zijn, ik een afweging zou kunnen maken om ze niet allemaal in de grafiek weer te geven, omdat ze dan niet allemaal op het scherm kunnen komen. Als mogelijk oplossing hiervoor geeft hij dat bijvoorbeeld de zeven langste releases/deployments en de zeven kortste releases/deployments in de grafiek opgenomen kunnen worden. Hieronder is te zien hoe een grafiek eruit zou kunnen zien.

******

Figuur 8: Voorbeeld grafiek waarin de gemiddelde duur van deployments of releases tussen klanten te zien zijn

## 7.3 Conclusie

Ik heb onderzoek gedaan naar hoe ik de proof of concept gebruiksvriendelijk kan realiseren. Hierbij heb ik geanalyseerd welke elementen mijn proof of concept mogelijk zou kunnen bevatten. Met behulp van deze analyse heb ik kunnen vaststellen dat ik de volgende facetten nodig heb:

1. *Het voor een gebruiker mogelijk maken dat hij kan aangeven om gegevens of data in te kunnen zien.*
2. *Het in statistiek verband weergegeven van releases en deployments en het weergeven van lijsten of tabellen waarin gegevens van releases en deployments verwerkt zijn.*
3. *Een element om te kunnen navigeren door de webapplicatie.*

Vervolgens heb ik onderzoek gedaan naar welke usability richtlijnen voor deze facetten van belang zijn. Hieruit is voortgekomen dat er een aantal richtlijnen zijn die voor de bovengenoemde punten van toepassing zijn

Voor punt 1:

* Duidelijk maken waar een gebruiker op kan klikken.

Voor punt 2:

* *Het creëren van een duidelijke visuele hiërarchie.*

*Voor punt 3:*

* *De navigatie element moet op elke pagina op dezelfde plek komen te staan en moet op elke pagina dezelfde uiterlijk hebben*
* *De naamgeving van een item van de navigatie element moet naar de bestemming leiden die door de naamgeving is aangegeven.*
* *De navigatie element moet een home item bevatten om vanuit elke pagina terug te kunnen keren naar het hoofdscherm*

Daarnaast zijn er twee andere usability richtlijnen die voor elke website gelden: het minimaliseren van ruis en het gebruik maken van conventies.

Naast literatuurstudie heb ik ook een usability expert geïnterviewd om te onderzoeken op welke manier ik de inzichten op de best mogelijke wijze kan presenteren. Hij heeft aangegeven dat er een aantal inzichten zijn die ik in een tabel vorm kan weergegeven, omdat voor die gegevens dat het beste is. Voor de inzichten waarbij er vergelijkingen of verhoudingen dienen te worden weergeven heeft hij mij geadviseerd om dat met behulp van grafieken te doen. Hierbij heeft hij ook enkele belangrijke details aangegeven zoals op welke manier bepaalde items in een grafiek weergegeven kunnen worden. Ik heb besloten om deze adviezen niet in de wind te slaan en deze zoveel als mogelijk proberen toe te passen bij het realiseren van de proof of concept.

#  8 Conclusies en aanbevelingen

In dit onderdeel zal ik uitleggen welke conclusies ik uit het onderzoek heb getrokken en welk aanbevelingen ik heb gedaan.

## 8.1 Conclusies

Het onderzoek bestaat uit drie onderdelen:

* Het software delivery proces
* Security
* Usability

Ik zal hieronder voor elk deelonderzoek aangeven welke conclusies ik heb getrokken.

##### Het software delivery proces

Ik heb onderzoek gedaan naar wat de theorie zegt over een software delivery proces. Hieruit is voortgekomen dat software deployment en continuous delivery begrippen zijn die een software delivery proces beschrijven. Vervolgens heb ik onderzoek gedaan naar hoe het software delivery proces van Finalist in elkaar zit. Hierbij heb ik geconcludeerd dat het software delivery proces van Finalist het meest overeenkomt met continuous delivery, omdat de manier waarop software wordt opgeleverd binnen Finalist in grote lijnen hetzelfde overeenkomt.

Voor het afnemen van de interviews heb ik vooraf aan de interviews een aantal vragen opgesteld. Deze vragen heb ik niet allemaal gesteld, omdat gezien de interactie van het interview, het interview anders verliep dan ik vooraf voorspelde. Ik heb wel voor elke geïnterviewde functionaris van Finalist de vragen gesteld die voor die functie relevant zijn. Uit de gesprekken met de projectleden en de managementleden van Finalist is naar voren gekomen dat de projectleden behoefte hebben aan inzichten van het software delivery proces om dit proces beter te kunnen sturen. Deze hebben een directe relatie met het software delivery proces, omdat deployments en releases van software een directe relatie hebben met dit proces. De managementleden hebben behoefte aan inzichten die geen directe relatie hebben met het software delivery proces, maar willen bijvoorbeeld inzicht in hoe projecten lopen, wat de kwaliteit van mensen is, hoe het met de mensen gaat en projectmarge.

Ik ben tot de conclusie gekomen dat alleen de inzichten waar de projectleden behoefte aan hebben meegenomen zullen worden naar de realisatie van de proof of concept. De inzichten waar de managementleden behoefte aan hebben zullen niet worden meegenomen bij het realiseren van de proof of concept, omdat deze geen directe relatie hebben met het software delivery proces. Verder ben ik tot de conclusie gekomen dat de inzichten over deployments een hogere prioriteit hebben dan releases, omdat deze in tegenstelling tot releases wel gefactureerd kunnen worden naar klanten. Hieronder is te zien welke inzichten worden meegenomen naar de proof of concept en welke een hoge prioriteit hebben.

Hoge prioriteit

* Inzicht in de kortste deployment tijd per klant
* Inzicht in de gemiddelde deployment tijd per klant
* Inzicht in de langste deployment tijd per klant
* Inzicht in de maandelijkse deployments er klant
* Inzicht in de verhoudingen van de duur van deployments tussen klanten

Lage Prioriteit

* Inzicht in de kortste release tijd per klant
* Inzicht in de gemiddelde release tijd per klant
* Inzicht in de langste release tijd per klant
* Inzicht in de maandelijkse releases per klant
* Inzicht in de verhoudingen van de duur van releases tussen klanten

##### Security

Ik heb onderzoek gedaan naar wat de meest voorkomende kwetsbaarheden van webapplicaties zijn en een lijst met deze kwetsbaarheden samengesteld. Vervolgens heb ik geanalyseerd en onderzoek gedaan naar welke onderwerpen uit deze lijst voor de veiligheid van de proof of concept van belang zijn. Uit deze analyse heb ik geconcludeerd dat: ***Broken Authentication and Session Management***, ***SQL injection*** en ***Cross Site Scripting*** de onderdelen zijn die ik moet onderzoeken.

Om te bepalen met welke technieken SQL injection kan worden voorkomen, heb ik drie potentiële technieken samengesteld. Uit het onderzoek van deze technieken heb ik geconcludeerd dat ik voor het realiseren van de proof of concept prepared statements moet gebruiken, omdat deze techniek een applicatie beschermd tegen SQL injection en in tegenstelling tot de andere technieken deze makkelijk te gebruiken is.

Om te bepalen hoe de proof of concept beveiligd kan worden tegen Broken Authentication, zijn er drie belangrijke onderdelen: Het beveiligen van wachtwoorden, het beveiligen van session ids/cookies en het beveiligen van het versturen van gegevens. Uit onderzoek heb ik vastgesteld dat wachtwoorden lang moeten zijn en uit verschillende karakters moeten bestaan. Gebruikers moeten hun wachtwoord kunnen wijzigen en wachtwoorden moeten gehashed in de database worden opgeslagen. Session ids moeten lang zijn, een tijdslimiet hebben en mogen niet in een URL te zien zijn. Voor het verzenden van gegevens moet de POST methode worden gebruikt, omdat deze methode in tegenstelling tot de GET methode wel veilig is.

Om te bepalen hoe cross site scripting voorkomen kan worden, heb ik uit onderzoek vast gesteld dat de proof of concept tegen client-side aanvallen beveiligd moet worden. Uit mijn onderzoek heb ik geconcludeerd dat ik de proof of concept kan beveiligen door de invoer goed te valideren en de uitvoer te decoderen.

Usability
Ik heb onderzoek gedaan naar hoe ik de proof of concept gebruiksvriendelijk kan realiseren. Hierbij heb ik geanalyseerd welke elementen mijn proof of concept mogelijk zou kunnen bevatten. Met behulp van deze analyse heb ik kunnen vaststellen dat ik volgende facetten nodig heb:

1. *Het voor een gebruiker mogelijk maken dat hij kan aangeven om gegevens of data in te kunnen zien.*
2. *Het in statistiek verband weergegeven van releases en deployments en het weergeven van lijsten of tabellen waarin gegevens van releases en deployments verwerkt zijn.*
3. *Een element om te kunnen navigeren door de webapplicatie.*

Vervolgens heb ik onderzoek gedaan naar welke usability richtlijnen voor deze facetten van belang zijn. Hieruit is voortgekomen dat er een aantal richtlijnen zijn die voor de bovengenoemde punten van toepassing zijn.

Voor punt 1:

* Duidelijk maken waar een gebruiker op kan klikken.

Voor punt 2:

* *Het creëren van een duidelijke visuele hiërarchie.*

*Voor punt 3:*

* *De navigatie element moet op elke pagina op dezelfde plek komen te staan en moet op elke pagina hetzelfde uiterlijk hebben*
* *De naamgeving van een item van de navigatie element moet naar de bestemming leiden die door de naamgeving is aangegeven.*
* *De navigatie element moet een home item bevatten om vanuit elke pagina terug te kunnen keren naar het hoofdscherm*

Daarnaast zijn er twee andere usability richtlijnen die voor elke website gelden: het minimaliseren van ruis en het gebruik maken van conventies.

Naast literatuurstudie heb ik ook een usability expert geïnterviewd om te onderzoeken op welke manier ik de inzichten op de best mogelijke wijze kan presenteren. Hij heeft aangegeven dat er een aantal inzichten zijn die ik in een tabel vorm kan weergegeven, omdat voor die gegevens dat het beste is. Voor de inzichten waarbij er vergelijkingen of verhoudingen dienen te worden weergeven heeft hij mij geadviseerd om dat met behulp van grafieken te doen. Hierbij heeft hij ook enkele belangrijke details aangegeven zoals op welke manier bepaalde items in een grafiek weergegeven kunnen worden. Ik heb besloten om deze adviezen niet in de wind te slaan en deze zoveel als mogelijk proberen toe te passen bij het realiseren van de proof of concept.

## 8.2 aanbevelingen

Ik heb alleen de inzichten van de projectleden naar de realisatie van de proof of concept meegenomen. De reden hiervoor was dat ik had besloten om alleen wensen van inzichten die een directe relatie hadden met het software delivery proces mee te nemen naar de proof of concept. De wensen van de managementleden heb ik gekwalificeerd tot inzichten die geen directe relatie hebben met het software delivery proces.

Ik wil Finalist adviseren om een aantal wensen van de managementleden in de toekomst te onderzoeken. Er zijn vier wensen van de managementleden waarvan ik vind dat die hiervoor in aanmerking komen:

* Inzicht in de voortgang van projecten
* inzicht in aantal wisselingen per project
* inzicht in hoeveel issues er worden gevonden in een release
* Inzicht in projectmarge

Deze wensen hebben een indirecte relatie met het software delivery proces, maar ik vind dat ze voor Finalist belangrijk zijn. Met behulp van de bovengenoemde wensen is Finalist beter in staat om haar dienstverlening te verbeteren en om financieel wellicht beter te worden. Het inzicht hebben in de voortgang van projecten kan erbij toedragen dat er sneller kan worden ingegrepen als projecten door welke reden dan ook niet goed lopen. Het inzicht hebben in het aantal wisselingen per project kan erbij toedragen dat er kan worden geanalyseerd of projecten waar veel wisselingen zijn slechter of beter lopen dan projecten waar weinig wisselingen zijn. Het inzicht hebben in hoeveel issues er worden gevonden in een release kan erbij toedragen dat Finalist de vinger erachter kan krijgen wat de reden hiervan is en hoe het verbeterd kan worden. Het inzicht hebben in de projectmarge kan erbij toedragen dat Finalist beter kan analyseren wat de samenstellingen van de projecten zijn en wat de winst van een project is. Hierbij kan worden gekeken of projecten waar veel juniors onderdeel van zijn, beter of slechter lopen dan projecten waar veel seniors onderdeel van zijn. Dit kan kan voor Finalist een financiële verbetering opleveren, omdat juniors goedkoper zijn dan seniors.

Er moet onderzoek worden gedaan hoe deze informatie te verkrijgen is. Staat informatie over de bovengenoemde inzichten in bestaande systemen. Als dat het geval kan er worden gekeken hoe deze informatie uitgelezen kan worden. Als dat niet het geval is dan kunnen functionaliteiten gerealiseerd worden waarbij dit bijgehouden kan worden. Ik adviseer ook om geen nieuw informatie systeem te realiseren voor de bovengenoemde inzichten, maar het aan de proof of concept toe te voegen. Hierdoor zal Finalist een centraal punt hebben waar alle belangrijke inzichten te bekijken zijn.

#  9 Evaluatie

Als eerst wil ik aangeven dat ik de samenwerking met Brian Snijders als prettig en positief heb ervaren. Mijn standplaats was Maarssen en de standplaats van Brian was Rotterdam. Op elke maandag kwamen we een uur bij elkaar om de voortgang te bespreken. Ik heb dit als prettig ervaren, omdat ik vond dat ik hierdoor genoeg feedback kreeg dat ik de rest van de week goed aan de slag kon gaan. Ook de samenwerking met Elfriede Krauth heb ik als prettig ervaren. Zij heeft mij goed begeleid tijdens mijn afstudeerperiode.

Ik ben erachter gekomen dat mijn ambitie ligt in het vakgebied software engineering. Het schrijven van het plan van aanpak en de scriptie zijn facetten die ik minder leuk ervaren heb. In de toekomst wil ik dan ook geen baan waarin ik voor een groot gedeelte van de tijd verslagen aan het schrijven ben. Dat zou ik absoluut niet lang kunnen volhouden. Ook het uitschrijven van de interviews is iets wat ik minder leuk vond. Het kost veel tijd en moeite om dit te doen.

Ik heb geleerd om op een gestructureerde wijze onderzoek te kunnen doen. Ik heb niet eerder mensen moeten interviewen om onderzoek te kunnen doen. In eerste instantie dacht ik dat dat heel makkelijk was, maar dat bleek absoluut niet het geval. De koers van een interview is voor het interview moeilijk vast te stellen. Een interview is een interactie. Dit betekent dat ik tijdens het interview de antwoorden snel moest analyseren om vervolgens een passende vervolgvraag te kunnen stellen. Daarnaast moest ik de interviews analyseren om te kijken wat de gewenste inzichten waren van het software delivery proces. Dit onderdeel heb ik als leerzaam ervaren. Ik kan me ook beter verplaatsen in de rol van een informatie analist die mensen moet interviewen om de wensen boven water te krijgen. Ook heb ik geleerd om met Spring MVC te kunnen werken. Ik heb niet eerder met dit framework gewerkt. Het wordt veel gebruikt in de praktijk en binnen Finalist en dat is de reden waarom ik het ook voor mijn proof of concept heb gebruikt.

Wat ik de volgende keer beter kan doen is sneller knopen doorhakken. Op een gegeven moment was ik klaar met de interviews, maar kon ik op dat moment niet snel genoeg bepalen welke kant ik precies met het onderzoek op wilde. Brian Snijders heeft mij toen duidelijk gemaakt dat het tijd was om de opdracht af te bakenen en te bepalen op welke onderdelen van het software delivery proces ik mij zou richten.

Voor het onderzoek naar usability ben ik een klein beetje afgeweken van mijn plan van aanpak. In het plan van aanpak heb ik aangegeven dat ik zou onderzoeken hoe de usability richtlijnen toegepast kunnen worden. Ik ben hiervan afgeweken, omdat de usability richtlijnen voor zich spreken en hier geen onderzoek voor nodig was. Voor de rest heb ik mijn onderzoek volgens het plan van aanpak gedaan.

Voor mij was het een leerzame ervaring waarin ik als persoon ben gegroeid en het vak software engineering beter ben begrijpen. Deze ervaring zal ik straks als bagage meenemen naar de beroepspraktijk.

#  10 Literatuurlijst

##### Boeken

Krug, S. (2006). Don’t make me think. new Riders Publishing.

Kassenaar, P. (2010). Usability. Duuren Media, van

Humble, J. & Farley, D. (2010). Continuous Delivery, Reliable Software Releases through Build, Test and Deployment automation. Adission-Wesley.

Stuttard, D. & Pinto, M. (2011). The Web Application Hacker’s Handbook(2nd Revised edition). John Wiley & Sons Inc

##### Internetbron

Siles, R. (29-04-2014). Session Management Cheat Sheet
Geraadpleegd op 8 maart 2015, van<https://www.owasp.org/index.php/Session_Management_Cheat_Sheet>

OWASP. (23-06-2013). Top 10 2013-A2-Broken Authentication and Session Management
Geraadpleegd op 8 maart 2015, van
<https://www.owasp.org/index.php/Top_10_2013-A2-Broken_Authentication_and_Session_Management>

Keary, E. (24-02-2015). Autehntication Cheet Sheet
Geraadpleegd op 8 maart 2015, van
<https://www.owasp.org/index.php/Authentication_Cheat_Sheet>

OWASP. (03-02-2014). Top 10 2013-A3-Cross Site Scripting(XSS)
Geraadpleegd op 8 maart 2015, van
[https://www.owasp.org/index.php/Top\_10\_2013-A3-Cross-Site\_Scripting\_(XSS)](https://www.owasp.org/index.php/Top_10_2013-A3-Cross-Site_Scripting_%28XSS%29)

Wiliams, J. Manico, J. Mattatall, N. (04-02-2014). XSS(Cross Site Scripting) Prevention Sheet
Geraadpleegd op 8 maart 2015, van
[https://www.owasp.org/index.php/XSS\_(Cross\_Site\_Scripting)\_Prevention\_Cheat\_Sheet](https://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet)

OWASP. (23-06-2013). Top-10 2013-A1-Injection
Geraadpleegd op 8 maart 2015, van
<https://www.owasp.org/index.php/Top_10_2013-A1-Injection>

Wichers, D. Manico, J. Seil, M. (12-04-2014). SQL Injection Prevention Cheat Sheet
Geraadpleegd op 8 maart 2015, van
<https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet>

Berkeley University.(z.d.). SQL Injection FAQ
Geraadpleegd op 8 maart 2015, van
<https://security.berkeley.edu/content/sql-injection-faq>

James, Jardine. (z.d.). How To Fix SQL Injection: Queries
Geraadpleegd op 8 maart 2015, van
<http://software-security.sans.org/developer-how-to/fix-sql-injection-microsoft-.net-with-parameterized-queries>

Cenzic. (2014). Application Vulnerability Trends Report
Geraadpleegd op 8 maart 2015, van
<http://www.cenzic.com/downloads/Cenzic_Vulnerability_Report_2014.pdf>

Edgescan. (2014). 2014 Vulnerability Statistics Report
Geraadpleegd op 8 maart 2015, van
<http://www.bccriskadvisory.com/wp-content/uploads/Edgescan-Stats-Report.pdf>

Kennedy, S. (2005). Common Web Aplication Vulnerabilities
Geraadpleegd op 8 maart 2015, van
<http://www.isaca.org/Journal/Past-Issues/2005/Volume-4/Pages/Common-Web-Application-Vulnerabilities1.aspx>

Gnanasundar, P. (2011-2015). How To Fix SQL Injection: Statements
Geraadpleegd op 8 maart 2015, van
<http://software-security.sans.org/developer-how-to/fix-sql-injection-in-java-using-prepared-callable-statement>

Landsman, A. (2010). Top Ten Most Critical Web Application Security Vulnerabilities
Geraadpleegd op 8 maart 2015, van
<http://blog.emagined.com/2010/03/22/top-ten-most-critical-web-application-security-vulnerabilities/>

Dearle, A. (2007). Software Deployment Past, Present and Future
Geraadpleegd op 8 maart 2015, van
<http://archive.cs.st-andrews.ac.uk/papers/download/Dea07.pdf>

Info Support. (2014). Whitepaper Continuous Delivery
Geraadpleegd op 8 maart 2015, van
http://blogs.infosupport.com/wp-content/uploads/2014/04/Whitepaper-Continuous-Delivery.pdf

Godtland, A. (2011). What is Software Deployment
Geraadpleegd op 9 maart 2015, van
<http://www.godtlandsoftware.com/word-press/2011/03/26/what-is-software-deployment/>

CrackStation. (2014). Salted Password Hashing – Doing it Right
Geraadpleegd op 16 maart 2015, van
<https://crackstation.net/hashing-security.htm>

Kauffman, L. (2013). About Secure Password Hashing
Geraadpleegd op 16 maart 2015, van
<http://security.blogoverflow.com/2013/09/about-secure-password-hashing/>

#  11 Woordenboek

Tabel 4: uitleg begrippen

|  |  |
| --- | --- |
| ***Begrip*** | ***betekenis*** |
| Jenkins | Een open-source release tool waarmee continuous integration voor het ontwikkelen van software mogelijk wordt gemaakt. |
| Rundeck | Een open-source deployment tool. Het zorgt ervoor dat operationele procedures in een datacenter of een cloudomgeving geautomatiseerd kunnen worden. |
| OTAP | Begrip dat staat voor: Ontwikkelen, Testen, Acceptatie en productie van software. |

**BIJLAGE**

#  12 Bijlage proof of concept

Nadat mijn onderzoek is voltooid is het zaak om de resultaten van mijn onderzoek toe te passen bij het realiseren van de proof of concept. In dit onderdeel zal ik uiteenzetten wat ik heb gerealiseerd en hoe ik dat heb gedaan.

## 12.1 Tools

In dit onderdeel zal ik aangeven welke tools ik heb gebruikt om mijn proof of concept te realiseren.

### 12.1.1 JAVA, MySQL en Java Database Connectivity

Zoals ik in hoofdstuk 2.1 heb vermeld, heb ik aangegeven dat ik de proof of concept met behulp van open source producten zou realiseren. Ik heb voor het realiseren de programmeertaal JAVA gebruikt. Voor het opslaan van gegevens heb ik een MySQL database gebruikt. Deze database is open source. Om ervoor te kunnen zorgen dat mijn proof of concept die in JAVA is gerealiseerd kan communiceren met MySQL heb ik Java Database Connectivity gebruikt. Het is een Java-API die ervoor zorgt dat een Java applicatie met behulp van SQL een database kan benaderen.

### 12.1.2 Spring MVC

Mijn begeleider Brian Snijders heeft mij de vrijheid gegeven om zelf te bepalen op welke manier ik de proof of concept zou realiseren. Hierbij kon ik dus kiezen of ik wel of geen framework zou gebruiken. Ik heb ervoor gekozen om wel een framework te gebruiken en de keuze is op het framework Spring MVC gevallen. Spring MVC is een open source framework waarmee webapplicaties gerealiseerd kunnen worden. MVC staat voor Model View Controller.



Figuur 9: Cycle Spring MVC(Model, View en Controller)

De controller is verantwoordelijk voor het behandelen van requests van gebruikers. De model staat voor het data model. De view geeft de presentatie van webpagina’s weer.

### 12.1.3 Rundeck en Jenkins

Zoals in het hoofdstuk randvoorwaarden staat vermeld, ben ik voor het realiseren van mijn proof of concept afhankelijk van twee systemen: Rundeck en Jenkins. Rundeck is een deployment tool en Jenkins is een release tool.


Figuur 10: Rundeck en Jenkins

De informatie die ik nodig heb voor mijn proof of concept wordt met behulp van API’s die Rundeck en Jenkins beschikbaar hebben gesteld, uitgelezen en naar mijn proof of concept geïmporteerd.

## ***12.2 Security***

Voor het beveiligen van de proof of concept moet ik rekening houden met drie onderwerpen:

* Broken Auhtentication and Session Management
* SQL injection
* Cross Site Scripting

Ik heb aangegeven dat ik voor het realiseren van de proof of concept het framework Spring MVC gebruik. Dit framework biedt ook voor security een aantal functionaliteiten aan. Ik heb zoveel als mogelijk de security functionaliteiten van dit framework toegepast, omdat voor mij als developer dat makkelijk is en het onnodig is om opnieuw het wiel uit te vinden. Ik zal hieronder per kwetsbaarheid aangeven hoe ik het heb beveiligd.

Broken Authentication and Session ManagementVoor deze kwetsbaarheid is in hoofdstuk 6 aangegeven dat er drie belangrijke onderdelen zijn waar rekening mee gehouden dient te worden:

* Wachtwoorden
* Session Ids/cookies
* Versturen van gegevens

***Wachtwoorden***In hoofdstuk 6 is aangegeven dat er een aantal richtlijnen in acht dienen te worden genomen. Hieronder is te zien welke richtlijnen dit zijn.



Figuur 11: eisen voor veiligheid van wachtwoorden

Om ervoor te zorgen dat de gebruiker gedwongen wordt om een lang wachtwoord met verschillende karakters te formuleren, geef ik dat op het scherm aan. Vervolgens valideer ik met behulp van een reguliere expressie of de ingevoerde wachtwoord voldoet aan de eisen.

Voor het opslaan van wachtwoorden had ik de keuze om uit meerdere hash algoritmes te kiezen. Md 5, SHA en Bcrypt. Ik heb ervoor gekozen om Bcrypt te gebruiken, omdat het een betere en stabielere encryptie algoritme heeft(Kauffman, 2013; CrackStation, 2014). Hieronder is te zien hoe de code eruit ziet.

**public** **void** setPassWord(String password) {

 BCryptPasswordEncoder passwordEncoder = **new** BCryptPasswordEncoder();

 String hashedPassword = passwordEncoder.encode(password);

 **this**.passWord = hashedPassword;

}

Voor het kunnen aanpassen van wachtwoorden heb ik een interface gebouwd waarin de gebruiker een nieuw wachtwoord kan aangeven die tevens lang en uit verschillende karakters moet bestaan.

Als een gebruiker foutieve inloggegevens invoert dan wordt er niet aangegeven of het wachtwoord of de gebruikersnaam foutief is. Het systeem geeft alleen aan dat de ingevoerde gegevens foutief zijn. Hieronder is dat te zien.



Figuur 12: Scherm bij foutieve gegevens

***Session IDs/cookies***Voor deze kwetsbaarheid is in hoofdstuk 6 aangegeven dat er drie belangrijke zaken zijn. Hieronder zijn deze te zien.



Figuur 13: Eisen veiligheid session IDs/cookies

Spring MVC biedt voor de beveiliging van session IDs en cookies functionaliteiten aan. Ik heb deze ook toegepast.

Spring MVC genereert voor sessions automatisch een lang en moeilijk te raden session id.

Om te voorkomen dat een session ID of een cookie te zien is in de URL, kan in de security.xml file dit worden aangegeven. Dit kan worden gedaan door ‘disable-url-rewriting’ te formuleren. Hieronder is dit te zien.

<http use-expressions=*"true"* auto-config=*"true"* disable-url-rewriting=*"true"*>

Binnen Spring MVC kan een session van een bepaalde duur worden voorzien door dit in de web.xml file te formuleren. Voor de proof of concept heb ik een session duur van dertig minuten ingesteld. Als de session afloopt moet de gebruiker opnieuw inloggen om gebruik te kunnen maken de applicatie. Hieronder is te zien hoe dit in de web.xml file wordt geformuleerd.

<session-config>

 <session-timeout>30</session-timeout>

</session-config>

***Versturen van gegevens***Voor het versturen van gegevens mocht enkel en alleen gebruik worden gemaakt van de POST methode, de GET methode is een onveilige methode. Ik heb dan ook voor het verzenden van gegevens louter de POST methode gebruikt. Hieronder is een voorbeeld te zien hoe dit in de view aangegeven kan worden.

action=*"*<c:url value=*'/j\_spring\_security\_check'* />*"* method=*'POST'*>

SQL injection
Zoals in hoofdstuk 6 is aangegeven, is voor de beveiliging van de proof of concept tegen een SQL injection gebruik gemaakt van prepared statements. Hieronder is te zien hoe een prepared statement met behulp van Java Database Connectivity in elkaar zit.

String query = "SELECT \* FROM Deployment WHERE executionID = ? and
deploymentJobID = ?";

 **try** {

 PreparedStatement ps;

 con = dataSource.getConnection();

 ps = con.prepareStatement(query);

 ps.setInt(1, executionID);

 ps.setString(2, deploymentJobID);

 }

Cross Site ScriptingZoals in hoofdstuk 6 is aangegeven kan de proof of concept beveiligd worden tegen cross site scripting door de invoer goed te valideren en uitvoer te decoderen. Spring MVC biedt hiervoor een functionaliteit aan die in de web.xml geformuleerd kan worden en voorkomt dat de proof of concept vatbaar is voor deze kwetsbaarheid. Hieronder is te zien hoe dit in de web.xml geformuleerd kan worden.

<context-param>

 <param-name>defaultHtmlEscape</param-name>

 <param-value>true</param-value>

</context-param>

## 12.3 Software delivery proces(inzichten)

Zoals in hoofdstuk 5 is uiteengezet, zijn er een aantal inzichten naar voren gekomen die ik mee zou nemen naar de realisatie van de proof of concept. De inzichten over de kortste, gemiddelde en langste release/deployment worden bij elkaar gevoegd, in hoofdstuk 5 zijn deze los genoemd. Hieronder volgen deze inzichten.

1. Verhouding van de gemiddelde duur van deployments tussen klanten
2. Verhouding van de gemiddelde duur van releases tussen klanten
3. Kortste, gemiddelde en langste duur van deployments per klant
4. Kortste, gemiddelde en langste duur van releases per klant
5. Maandelijkse deployments per klant
6. Maandelijkse releases per klant

Inzicht 1 en 2Inzicht 1 en 2 zijn de volgende inzichten:

* Verhouding van de gemiddelde duur van deployments tussen klanten
* Verhouding van de gemiddelde duur van releases tussen klanten

Ik heb ervoor gekozen om deze twee inzichten op het hoofdscherm weer te geven. Zowel voor releases als voor deployments zijn er twee soorten builds: Continuous build en release build. Dit betekent dat er voor deployments continuous deployment en release deployment zijn, voor releases zijn er continous build en release build. De mogelijkheid was dat ik deze twee soorten builds samen zou voegen voor deployments en ook voor releases. Samen met mijn begeleider heb ik besloten om deze twee type builds apart weer te geven en niet samen te voegen. Hieronder is te zien hoe ik de verhoudingen van releases en deployments heb weergegeven.



Figuur 14: Statistieken van deployments en releases tussen verschillende klanten

Er is te zien dat per klant de gemiddelde duur voor de releases release builds en continuous builds te zien zijn en voor deployments continuous deployment en release deployment te zien zijn. Hiermee kunnen gebruikers de gemiddelde release tijd en deployment tijd van klanten met elkaar vergelijken.

Inzicht 3 en 4Inzicht 3 en 4 zijn de volgende inzichten:

* Inzicht in de kortste, gemiddelde en langste duur van releases
* Inzicht in de kortste, gemiddelde en langste duur van deployments

Voordat de gebruiker deze inzichten kan bekijken heb ik een pagina ontworpen waarmee de gebruiker een klant kan selecteren. Hieronder is deze pagina te zien.



Figuur 15: Scherm voor het selecteren van een klant

Nadat de gebruiker een klant heeft geselecteerd wordt hij doorverwezen naar de pagina waarin hij de statistieken van releases en deployments voor die specifieke klant kan bekijken. Hieronder is deze pagina te zien.



Figuur 16: Statistieken van deployments en releases van een specifieke klant

Ook de statistieken van releases en deployments per klant worden ook weergegeven in continuous build en release build voor releases en voor deployments in continuous deployment en release deployment. Voor de klant ‘Zorg Instituut Nederland’ is te zien dat er voor releases nog geen release builds zijn uitgevoerd. Zodra er release builds worden uitgevoerd zullen de statistieken hiervan wel worden weergegeven. De gemiddelde duur van de releases en deployments worden in minuten aangegeven.

Inzicht 5 en 6Inzicht 5 en 6 zijn de volgende inzichten:

* Maandelijkse deployments per klant
* Maandelijkse releases per klant

Deze inzichten bevinden zich op dezelfde pagina als inzicht 3 en 4. Het kunnen bekijken van de maandelijkse deployments en releases en deployments vergt dus ook als voorwaarde dat er een klant geselecteerd wordt. Hieronder is te zien hoe ik deze inzichten heb weergegeven.



Figuur 17: Gegevens over deployments en releases voor een specifieke klant

Op deze pagina kan de gebruiker een maand en een jaar selecteren. Vervolgens worden het aantal deployments en releases opgehaald en vermeld. Voor de deployments en releases wordt het duur, datum en tijdstip getoond. Hierbij is er geen onderscheid gemaakt tussen continuous build en release build en tussen continuous deployment en release deployment, omdat voor dit onderdeel het alleen interessant is hoeveel releases en deployments voor die maand hebben plaatsgevonden.

## 12.4 Usability

Zoals in hoofdstuk 7 is uiteengezet, zijn er een aantal richtlijnen van usability die ik moet toepassen in mijn proof of concept. De volgende zes richtlijnen zijn hierbij van belang:

1. *Duidelijk maken waar een gebruiker op kan klikken.*
2. *Het creëren van een duidelijke visuele hiërarchie.*
3. *De navigatie element moet op elke pagina op dezelfde plek komen te staan en moet op elke pagina dezelfde uiterlijk hebben.*
4. *De naamgeving van een item van de navigatie element moet naar de bestemming leiden die door de naamgeving is aangegeven.*
5. *De navigatie element moet een home item bevatten om vanuit elke pagina terug te kunnen keren naar het hoofdscherm.*

Daarnaast heb ik ook een aantal adviezen van de usability expert gekregen om de inzichten op de best mogelijke manier weer te geven. Ik zal per pagina uitleggen welke usability richtlijnen en welke adviezen van de usability expert ik heb toegepast en hoe.



Figuur 18: Statistieken van deployments en releases tussen klanten

Voor deze pagina heb ik de usability richtlijnen: Creëer een duidelijke visuele hiërarchie en het gebruik van conventies toegepast. Het creëren van een duidelijke visuele hiërarchie heb ik gedaan door deployments en releases in twee duidelijk afgebakende gebieden te plaatsen. Daarnaast heb ik ervoor gekozen om deployments hoger op de pagina te plaatsen dan releases, omdat deployments een hogere prioriteit hebben. De statistieken van deployments hebben een hogere waarde, omdat deployments in tegenstelling tot releases wel gefactureerd naar klanten kunnen worden. Verder heb ik gebruik gemaakt van een conventie. Boven de statistieken heb ik aangegeven of het om deployments of om releases gaat. In hoofdstuk 7 staat aangegeven dat een kopje aangeeft waar het onderliggende over zal gaan. De gebruiker weet nu meteen welke grafieken onder welk onderwerp vallen. Het type release(continuous build/release build) en het type deployment(continuous deployment/release deployment) heb ik niet boven de grafieken geplaatst, maar onder de grafieken, omdat ik vond dat het er anders druk uit zou zien en ik het belangrijkste vond om aan te geven of het om deployments of om releases gaat. Ook heb ik het menu(navigatie element) bovenaan de pagina geplaatst en laat ik zien dat de gebruiker op de home pagina is door het menu item de kleur oranje te geven.

De usability expert heeft mij geadviseerd om de verhouding van gemiddelde duur van releases en deployments, in grafieken weer te geven en voor beide een aparte grafiek te maken. Ik heb dit ook toegepast. Ik heb voor releases en deployments aparte grafieken gemaakt, waarin ik ook nog eens onderscheid heb gemaakt tussen het type release en het type deployment.



Figuur 19: Scherm voor het selecteren van een klant

Voor deze pagina heb ik het menu boenvaan geplaatst, zodat dit consistent is door de proof of concept. Ik laat zien in welk onderdeel van het menu de gebruiker zich bevindt door het menu item een oranje kleur te geven. De menu items staan op dezelfde plek als op de homepage. Verder heb ik gebruik gemaakt van een conventie door een kopje boven de klanten te plaatsen, zodat de gebruiker gelijk ziet waar het onderliggende over gaat.



Figuur 20: Statistieken en gegevens van deployments en releases van een specifieke klant

Voor deze pagina heb ik ervoor gekozen om de statistieken in een afgebakend gebied te plaatsen. Ook heb ik de tabellen en gegevens over aantallen deployments en releases in een afgebakend gebied geplaatst. Hierdoor is direct duidelijk welke onderdelen bij elkaar horen. Daarnaast heb ik ook een conventie gebruikt. Boven de grafieken en tabellen heb ik kopjes geplaatst, zodat de gebruiker gelijk weet waar het onderliggende over gaat. Het aantal deployments en releases heb ik ook nog eens een oranje kleur gegeven, zodat het opvalt, omdat dit een belangrijk onderdeel van de pagina is. Verder heb ik duidelijk gemaakt wat klikbaar is. In hoofdstuk 7 staat hoe duidelijk gemaakt kan worden waar een gebruiker op kan klikken. Ik heb geen onduidelijke link of tekst gebruikt, maar een duidelijke knop, zodat een gebruiker gelijk weet dat hij erop kan klikken. Verder heb ik een link geplaatst, zodat de gebruiker terug kan keren naar de pagina waar de klanten staan. Ook heb ik het menu bovenaan de pagina weergegeven, zodat dit consistent is. Via het menu kan de gebruiker ook weer terugkeren naar de home pagina.

De usability expert heeft aangegeven dat ik de kortste, gemiddelde en langste duur van deployments en releases in grafieken kan weergeven, voor beide een aparte grafiek. Ik heb dit ook toegepast. Verder heeft hij aangegeven dat de statistieken over de duur van releases en deployments in de volgende weergegeven kunnen worden: kortste, gemiddelde en langste. Ook dit heb ik toegepast. Voor de informatie over deployments en releases heeft hij aangegeven dat dit in tabellen weergegeven kunnen worden. Op het plaatje is te zien dat ik tabellen heb gebruikt.

#  13 Bijlage Finalist

## 13.1 Beschrijving Finalist

Finalist is een bedrijf dat is opgericht in 1988 en ongeveer tachtig medewerkers heeft. Ze hebben een passie voor software ontwikkeling. Om software te ontwikkelen kiezen zij ervoor om gebruik te maken van open source producten. Ze zijn altijd op de hoogte van de laatste technieken op het gebied van open source producten.

De klanten van Finalist bevinden zich door heel Nederland. Hierdoor heeft Finalist ervoor gekozen om een vestiging in Maarssen, Rotterdam en Eindhoven te openen. Hiermee willen zij hun klanten op de best mogelijke wijze van dienst te zijn.

Finalist realiseert projecten die vaak groot en complex van aard zijn. Gezien het feit dat deze groot en complex zijn, heeft Finalist verschillende functies binnen een team. Projectmanagers, testers, analisten, ontwerpers en developers zijn functies die binnen een team van Finalist bekleed worden. Hiermee worden de projecten in goede banen geleid.

## 13.2 Software

Finalist realiseert software voor veel klanten. De software die zij realiseren kunnen in drie categorieën worden verdeeld. Hieronder worden deze categorieën beschreven.

Websites en portalen
Voor het realiseren van websites en portalen gebruikt Finalist vaak ook een CMS. Ook dit zijn Open Source producten. Voorbeelden van deze CMS’en zijn: Hippo(JAVA), Dupal(PHP) en Liferay Portal.

Applicatie integratie
Met zowel de ontwikkelkennis als de specifieke integratiekennis in huis, zorgt Finalist er ervoor dat de applicatie integratie van de software vlekkeloos verloopt en alle systemen moeiteloos met elkaar kunnen “praten”. Hiervoor maken de experts koppelingen tussen software-systemen, beantwoorden zij integratie-vraagstukken en krijgen de klanten alle hulp met de authenticatie.

Maatwerk software
De wensen en eisen van de organisatie’s staan centraal bij de bouw van gebruiksvriendelijke webapplicaties, zodat deze voor de klanten volledig op maat zijn gemaakt. Zo heeft Finalist myTomorrows ontwikkeld: een applicatiesysteem waarmee uitbehandelde en chronisch zieke patiënten versneld toegang krijgen tot innovatieve medicijnen. Met deze op exclusief ontwikkelde applicatie worden de patiënten simpel door het complexe aanmeldsysteem geleid. Hierna informeert myTomorrows de patiënt op basis van hun gegevens over behandelingen waarvoor hij of zij in aanmerking kan komen.

Finalist Heeft groot aantal klanten. Hieronder zullen een aantal klanten worden vermeld waar Finalist software voor realiseerd.

## 13.3 Organisatiestructuur

Om een beter beeld te geven van hoe de organisatie van Finalist eruit ziet, zal hieronder een organogram te zien die de structuur van de organisatie weergeeft.


Het management is de hoogste rang binnen Finalist en is verantwoordelijk voor de sturing van Finalist. Binnen zijn er verschillende functies.

Het bedrijfsbureau is verantwoordelijk voor de uitvoering van projecten. Het ervoor zorgen dat er genoeg projectleiders en developers zijn, zijn voorbeelden van de verantwoordelijkheden van het bedrijfsbureau.

Een unit bestaat uit een unit manager, een solution expert en de uitvoerings functies. Developers, analisten en testers zijn voorbeelden van uitvoerende functies. Unit PII staat voor portals integration and information en is verantwoordelijk voor data en visualisatie en integratie. Unit CD staat voor Content Delivery en is verantwoordelijk voor cmsen, websites, online strategie en ontwerp. Unit CS staat voor Custom Solutions en is verantwoordelijk voor maatwerk klussen. De beheer unit is verantwoordelijk voor alle projecten die in beheer zijn gegaan.

#  14 Bijlage interviews

### Interview: Brian SnijdersFunctie: Solution expert

***Interviewer*** Wat is een software delivery proces?

***Brian*** Een software delivery proces is het gehele proces waarin software wordt opgeleverd. Als eerst komt het team samen met de klant bij elkaar. Samen wordt dan gekeken naar de wensen van de klant. Hieruit komen de requirements naar voren. Het team bestaat uit: Een informatie analist, projectmanager, developers, tester en een architect. Nadat de requirements duidelijk zijn wordt de software in iteraties van twee weken gebouwd. Vervolgens komt er een testrelease en wordt de software opgeleverd. De klant kan dan aangegeven of het naar wens is en als dat zo
is kan de software via acceptatie naar productie. Voor het realiseren van software wordt het begrip ‘OTAP’ gebruikt. Dit begrip staat voor ontwikkelen, testen, acceptatie en productie.

***Interviewer*** Welke gedeelte van de organisatie van Finalist heeft te maken met het software delivery proces en wie zijn eventueel geschikt om te interviewen.

***Brian*** Het uitvoerend personeel is een gedeelte die zeker met dit proces te maken heeft. De mensen die verantwoordelijk zijn voor de software komen hierin voor. Hierbij kun je denken aan developers, analisten, testers en architecten. Projectmanagers zijn ook direct betrokken bij het software delivery proces. Binnen Finalist zijn verschillende units en iedere unit heeft een manager. Deze unitmanagers zijn verantwoordelijk voor een unit en zouden misschien verbeteringen van de communicatie van het software delivery proces willen doorvoeren. Ook hebben we een operationeel directeur. De operationeel directeur is eindverantwoordelijk voor de gehele uitvoering van projecten en zou dus ook misschien verbeteringen willen over de communicatie van het software delivery proces. Uiteraard kunnen aan de hand het onderzoek en de interviews meerdere rollen uitkomen die geschikt zijn om te interviewen

### Interview: IvorFunctie: Developer

***Interviewer*** Ik doe onderzoek naar hoe de communicatie over het software delivery proces verbeterd kan worden en welke inzichten van dit proces gewenst zijn. Welke inzichten van dit proces zou jij als developer willen hebben?

***Ivor*** Wat bedoel je precies met inzichten.

***Interviewer*** Jij als developer bent betrokken bij het realiseren van software. Welke inzichten van het gehele traject van het ontwikkelen van software mis jij momenteel of zouden het leven van een developer vergemakkelijken.

***Ivor*** Ik vind het best moeilijk om dat te weten, misschien iets met issues van de projecten. Waarbij er kan worden aangegeven wat voor issue het is en welke developers daar eerder aan hebben gewerkt.

***Interviewer*** Houden jullie die gegevens momenteel bij?

***Ivor*** Wij gebruiken Jira. Dat is een tool waar een developer kan zien aan welke use cases hij moet werken en welke issues er al zijn. Het is een goede tool.

***Interviewer*** Welke inzichten mis jij in Jira?

***Ivor*** Ja uhmm. Ik denk misschien de issues dat die beter gepresenteerd kunnen worden. Misschien dat de status van het ontwikkeltraject beter weergegeven kan worden.

***Interviewer*** Wordt dat in Jira niet goed weergegeven?

***Ivor*** Ja, eigenlijk wel. Ik kan de status van de projecten zien en weet ook wat de issues zijn.

***Interviewer*** Ik merk eigenlijk dat jij best tevreden bent over Jira.

***Ivor*** Ja, dat klopt.

***Interview*** Ik merk aan jou dat jij eigenlijk tevreden bent over Jira en dat jij geen behoefte hebt aan bepaalde inzichten. Klopt dat?

***Ivor*** Ja, dat klopt. Ik denk dat het management meer gebaat is bij bepaalde inzichten. Wij als developers weten precies wat we moeten doen en Jira is een goede tool voor ons.

***Interviewer*** Ik denk dat de conclusie is dat jij als developer geen behoefte hebt aan bepaalde inzichten van het software delivery proces.

***Ivor*** Dat klopt eigenlijk.

### Interview AlbertFunctie:Projectmanager

***Interview*** Welke inzichten zou jij van het software delivery proces willen? Welke inzichten zouden het leven van een projectmanager makkelijker maken?

***Albert*** Heb je het dan over het deployment proces?

***Interviewer*** Het gaat echt om het software delivery proces. En het gaat om jou als projectmanager binnen dit proces. Waar heb jij behoefte aan?

***Albert*** Dat is wel heel breed. Dan denk ik dan vooral aan de inschattingen op het moment dat er een offerte gemaakt wordt, met goede opslagen waarbij ik kan zeggen bij die klant heb je meer uren nodig.

***Interviewer*** Kan je dat nu niet goed zien?

***Albert*** Dat is een proces wat nog loopt. We hebben een sheet om schattingen in te doen, maar heel vaak wordt dat gebruikt in het begin en vervolgens gaat een account-manager naar een klant en komt dan terug en zegt dat we toch te duur zijn. Dan moet er wat van af en vervolgens wordt er zomaar wat geschrapt. Dan gelden hele andere processen opeens. Dan moet ik bij een intake zien wat de schatting was en waarop hij was gebaseerd. Wat is er terecht gekomen van die schattingen? Wat is er terecht gekomen van wat we zouden gaan bouwen?

***Interviewer*** Dat is nu niet inzichtelijk?

***Albert*** Ik had laatst een offerte waar de projectmanagement uren niet in de begroting waren gekomen. Nu kan je zeggen we doen dit project zonder projectmanager, dat is geen handige keus. Je kan ook niet terug naar de klant om te zeggen dat er zoveel aan project management bij komt. De accountmanager wilde niet terug naar de klant. Dan ga je die uren tegen nul tarief doen. Wil je dat soort processen ook meenemen?

***Interviewer***  Ja, dat zou kunnen.

***Albert*** Dan maak je je onderzoek heel erg breed.

***Interviewer*** Ja, ik houd me alleen met het software delivery proces bezig.

***Albert***  Dan valt het hele verhaal over de offertes natuurlijk buiten. Dan is het mijn probleem als de project management uren niet zijn meegenomen. Dan moet ik dat met de accountmanager regelen.

Is alles wat we moeten bouwen ook ingeschat en zijn er dingen bijgekomen? Ik werk vooral veel in projecten met flinke agile en scrum elementen erin. De meeste offertes zijn bij onze projecten fixed budgets. Er is wel ruimte om een bepaald onderdeel te laten vallen. Ik wil graag zien waar we staan. Het mooiste is als we van te voren een compleet backlog hebben met alle functionaliteiten die gerealiseerd moeten worden. Hiermee heb je ook gevoel van wat je allemaal moet doen.

***Interviewer*** Je wilt graag waar je staat?

***Albert*** Ja. Wat moet er idealiter allemaal gebouwd worden. Vervolgens kijken of het kans van slagen heeft. Daar heb je je inschattingen voor die gemaakt zijn voor de offerte. De sprint doelen van te voren vast stellen. Dan heb ik in ieder geval een beeld van past dit met de uren die we hebben ingeschat en de uren die vrij hebben gemaakt. Dan heb je sneller een beeld als we uitlopen. Dan kun je functionaliteiten van de volgende sprint misschien verplaatsen of eruit gooien. Dan weet je in ieder geval dat als we op deze voet doorgaan dan gaan we het nooit redden.

***Interviewer*** Dan kun je dus zien waar jullie staan.

***Albert*** Het is geen exacte wetenschap, maar het geeft jou en de klant het gevoel dat er wat veranderd moet worden. Dit is vooral ook richting de klanten prettig werken.

***Interviewer*** Ben jij ook bij meerdere projecten betrokken?

***Albert*** ja.

***Interviewer*** Zou je dat voor elke project apart willen hebben?

***Albert*** ja.

***Interviewer*** Zou je dan een lijst willen hebben met projecten die je kunt selecteren?

***Albert*** Ik heb Jira en daar staan al m’n projecten in. Die ga je toch allemaal langs om te kijken waar we staan. Bij een sprintmeeting zie je ook dat we deze sprint zouden doen, maar ik zie toch een aantal dingen door schuiven naar de volgende sprint.

***Interviewer*** Kan je dat allemaal in Jira zien?

***Albert*** Ja. In jira maak je sprints aan, houd je issues en taken aan. Bij de start van een sprint kijk je hoeveel we kunnen doen. Dat check je met je sprint doelen, hebben we alle doelen hiermee gedekt. Aan het eind van de sprint zie je dus of je dat allemaal gehaald hebt.

***Interviewer*** Zou je dit in een applicatie willen hebben?

***Albert*** Ik vind Jira daar prima geschikt voor. Daar heb ik niet iets voor nodig. Dit werkt tijdens het bouwen gewoon prima.

***Interviewer*** Kun je in Jira ook alle functionaliteiten zien?

***Albert*** Ja, dat kan allemaal in Jira.

***Interviewer*** Dus je bent tevreden over Jira?

***Albert*** Ja, dat werkt prima.

***Interviewer*** Welke inzichten heb je nu niet die je misschien wel in een applicatie zou willen hebben?

***Albert*** Ik hoef niet zo zeer een applicatie. Wat ik merk is dat inschattingen soms niet kloppen en dat is niet door een applicatie te voorkomen. Voor mijn werk heb ik niet een applicatie nodig.

***Interviewer*** Wat ik bij jou merk is dat je eigenlijk best tevreden bent over Jira en dat je alle inzichten van het software delivery proces tot je beschikking hebt met behulp van Jira.

***Albert*** Ja. Ik wil ook niet informatie van het ene systeem naar andere systeem over kloppen.

***Interviewer*** Kun je ook statistieken in Jira bekijken?

***Albert*** ja, maar ik maak er nauwelijks gebruik van.

***Interviewer*** Ik kom eigenlijk tot de conclusie dat je alle inzichten die jij nodig hebt allemaal met behulp van Jira tot je beschikking hebt en dat je daar tevreden over bent.

***Albert*** Ja, dat klopt.

### Interview: BrianFunctie: Solution expert

***Interviewer*** Welke inzichten van het software delivery proces die jij niet via Jira kunt verkrijgen zou jij graag willen hebben.

***Brian*** Feitelijk zou ik het meest geïnteresseerd zijn in bijvoorbeeld de hoeveelheid releases die wij maandelijks doen, de hoeveelheid deployments die wij maandelijks doen. En dan bijvoorbeeld uitgesplitst per klant of per omgeving. Daarnaast wil je weten als je dan die inzichten die aantallen hebt, hoe lang duurt het gemiddeld, wat is de langste tijd, wat is de kortste tijd. Eigenlijk een stukje performance matrix over je hele software delivery proces. Dat begint natuurlijk met bouwen op je notebook, bouwen op continious integration machine. Vervolgens wordt het getest en ga je een release draaien, die worden op een testomgeving of A omgeving gezet. Vervolgens wordt die geaccepteerd. Dan zet je hem op de P omgeving, een stukje deployment van een release. Dat zou je ook willen meten. Puur en alleen om te kijken, heb ik bij een bepaalde klant stelselmatig hele trage deployments, hele trage releases?. En hoe verhoudt dat zich tot een klant waar het allemaal heel snel gaat. Is er iets in te ontdekken wat we vanuit preventief beheer en kwaliteitsverbetering kunnen oplossen. Als ik een klant heb waarbij een deployment heel lang duurt dan kan dat verschillende redenen hebben. Misschien dat wij het proces niet goed geregeld hebben. Misschien dat de infra waar wij naar toe deployen niet voldoende responsive of te traag is. Misschien dat we een trend opmerken dat als we naar externe partijen deployen dat dat altijd langzamer gaat en waar dat aan ligt. Ik zou over klanten bepaalde doorsnedes willen maken. Ik heb een gegeven dat bepaalde klanten een bepaalde versie van Hippo gebruikt, dus eigenlijk een filtering zou ik zeggen. Omdat we niet alleen Hippo doen, maar ook andere dingen moet dat wel flexibel zijn. Je zou eigenlijk iets willen hebben wat eigenlijk een soort portal is, waarbij je verschillende informatie uit verschillende systemen kunt onttrekken. Bijvoorbeeld uit Jira, Rundeck en Jenkins. Als je dat allemaal bij elkaar kunt aggregeren in één informatie portal, dat zou op zich heel veel inzicht kunnen leveren.

***Interviewer*** Jij ziet dus eigenlijk een systeem voor je dat met andere systemen kan communiceren.

***Brian*** Ja, uiteindelijk wel.

***Interviewer*** Het moet dus een tool worden die informatie kan ophalen uit verschillende systemen.

***Brian*** Ja en die dat dan overzichtelijk kan presenteren. Waarmee je ook eigenlijk verhoudingen zou kunnen bepalen. Hoe draaien drupal en hippo projecten gemiddeld. Een stukje zien van, want allebei de teams hebben een eigen bouwstraat daar vandaan kun je misschien een trend herkennen. Is de ene bouwstraat sneller dan de ander. Dan zou het Hippo team misschien naar de bouwstraat van Drupal kijken en daar van leren. Dat soort inzichten zijn er nu helemaal niet en we kunnen er niet op sturen.

***Interviewer*** Nu zeg jij dat er een tool moet komen die met die verschillende systemen moet communiceren. Nu ben jij als solution expert technisch onderlegd. Is het technisch haalbaar dat je met Jira bijvoorbeeld kunt communiceren.

***Brian*** Ja, Jira heeft zelf een API waarmee met een webservice client of rest client informatie op kan halen. Jenkins heeft ook een API die uitgevraagd kan worden om metrieken te krijgen. Rundeck heeft ook een API die uitgevraagd kan worden. Het bouwen gebeurt in Jenkins, het testen gebeurt in Jenkins, deployen gebeurt in Rundeck. Dat zijn eigenlijk alle metrieken die je zou moeten verzamelen.

***Interviewer*** De opslag blijft Jira bijvoorbeeld. Jullie willen dus geen applicatie waarbij je deze gegevens kunt invoeren?

***Brian*** Jenkins en Rundeck zijn in principe de authentieke bron voor alle informatie die je nodig hebt om die metrieken te bepalen en inzicht te geven. Het kan zijn dat je misschien een database wilt hebben of dat je alles real time wilt doen. Dat is een afweging die jij kan maken. Het gaat er om dat wij die inzichten tot onze beschikking hebben. Voor jou zou dat een technische overweging kunnen zijn. Ga je tussentijds alles vast leggen of ga je het real time doen.

***Interviewer*** Jira blijft dan als input instrument?

***Brian***  ja, dat zoiezo. Of je dat tussentijds vastlegt dat is een afweging die je moet maken.

***Interviewer*** Ik heb voor jou een projectleider en een developer geïnterviewd en die stonden niet op een nieuwe applicatie te wachten.

***Brian*** Ja, maar er is wel degelijk behoefte aan een centraal punt van projectinformatie. Welke plugins gebruiken we allemaal binnen een project. Het staat nu wel ergens, maar men moet het opzoeken in één of ander tekst bestand.

***Interviewer*** Hoe zie jij die inzichten voor je? Als los gegevens of in statistiek verband?

***Brian***  Allebei. Actuele en historische gegevens heb je nodig om te kijken hoe doe ik het nou over de tijd heen. In het algemeen zou ik willen weten hoe het in de huidige maand gaat. Uiteindelijk ook per maand factureren en per maand willen weten hoeveel releases je hebt gedraaid.

***Interviewer*** Kun je deze statistieken in Jira zien.

***Brian*** Nee, dat is niet strak gestandaardiseerd in Jira.

***Interviewer*** Welke inzichten hebben voor jou de hoogste prioriteit?

***Brian***  Dat vind ik vrij lastig. Dat kan ik niet opnoemen. Wat ik welk belangrijk vind is dat het veilig is. Dat is een prioriteit. De gevoelige informatie moet beveiligd worden. Je moet een mechanisme hebben om klanten toegang te geven en toegang in te trekken. En uiteindelijk is het ook belangrijk hoe je de inzichten presenteert. De succes van de oplossing ligt aan hoe veilig en gebruiksvriendelijk het is. Hoe snel mensen wegwijs worden binnen zo’n applicatie. Als ik niet heel veel inzichten heb, maar ik heb wel de basis inzichten die veilig zijn en ik kan monitoren dan is dat natuurlijk genoeg. En de basisinzichten is niet meer dan hetgeen wat ik genoemd heb. Hoeveel releases per klant doen we, hoelang duren ze gemiddeld. Dit soort dingen allemaal.

### Interview: KoljaFunctie: Unit manager

***Interviewer*** Ik heb de afgelopen weken met het project team gesproken, hieruit is het één en ander uitgekomen. Jij bent unit manager en functioneert op een hoger niveau. Het doel is om het software delivery proces te verbeteren en daarvoor zijn bepaalde inzichten nodig. Ik ben dan ook benieuwd welke inzichten jij van het software delivery proces wilt hebben.

***Kolja*** Ik wil inzicht hebben in kwaliteit van de software. Dat heeft ook te maken met releases, iets moet komen te hangen in het kwaliteitssysteem als het nieuw is. Vervolgens moet daar een trend rapportage zichtbaar worden voor mij. Kwaliteit is een moment opname.

***Interviewer*** Wat versta jij onder kwaliteit?

***Kolja*** Kwaliteit is dat we binnen zekere grenzen optimaliseren op een pragmatische manier, dat we goede code opleveren, goede functionaliteiten. Dat onze processen zodanig zijn dat we dat makkelijk kunnen doen.

***Interviewer*** Code is niet echt meetbaar?

***Kolja*** Dat is niet helemaal waar. We spreken bijvoorbeeld styling regels af. We zeggen bijvoorbeeld dat we drie spaties gebruiken. Hoe vaak wordt dat overtreden? Je kan dit bijvoorbeeld ook uitbreiden met code checkers die letten op veel gemaakte fouten. Je hebt bijvoorbeeld design patterns en je hebt daar code checkers voor die zeggen je probeert iets te doen en daar is een design pattern voor en dat doe je niet goed, die pas je niet toe. We hebben daar ook tools voor, Jenkins bijvoorbeeld. Daar hangen dingen in, maar daar moet ik toch weer ergens heen om dat te krijgen. Ik zou daar wel inzicht in willen hebben.

***Interviewer*** In code kwaliteit.

***Kolja*** De huidige stand en de trend zou ik willen weten. Vooral de trend vind ik belangrijk.

***Interviewer*** Je noemt code kwaliteit. Is er nog een andere onderwerp die van belang is?

***Kolja***  Projecten daar worden voortgangsrapportages van gemaakt, ik zou ook wel willen weten wat de trend is van projecten. Daarnaast zou ik de excepties ervan willen weten.

***Interviewer*** Wat versta jij onder excepties?

***Kolja*** Welke projecten zijn er in zwaar weer? In januari verdwijnt de operationele verantwoordelijkheid van de unit manager. Dit gaat naar het bedrijfsbureau. Ik word dan verantwoordelijk over hoe het met de mensen gaat. Dat is misschien ook wel iets waar ik inzicht in wil hebben. Wat is de kwaliteit van mensen en hoe het met ze gaat.

***Interviewer*** Wil je dat per werknemer weten?

***Kolja*** Dat zou op zich wel grappig zijn.

***Interviewer*** Houden jullie ook cijfers van werknemers bij?

***Kolja*** Ik houd ze niet bij. Ik heb dan wel een goed geheugen, dus dat onthoud ik wel.

***Interviewer*** Als jij bijvoorbeeld straks vertrekt dan heeft jou opvolger daar geen zicht op.

***Kolja***  ja.

***Interviewer*** Hoe zie jij het dan voor je?

***Kolja*** Je tikt gewoon even in en het wordt bijgehouden in een systeem.

***Interviewer***  Het is best wel lastig als de verantwoordelijkheden vanaf januari veranderen.

***Kolja*** Ja. Het is ook wel interessant om de status van een project te weten. Als een project in zwaar is, dan kan ik daar misschien uit aflezen dat de mensen van dat project het niet naar hun zin hebben en dat is wel degelijk interessante informatie.

***Interviewer*** Hoe zou je kunnen meten dat een project in zwaar is?

***Kolja*** Dat is iets wat van een project manager moet komen. Die houdt voortgangsverslagen bij en daarin kan hij aangeven dat er dingen niet lekker gaan. Dat zie je daarin terug. En als je dat daar vandaan zou kunnen uitlezen is dat interessant.

### Interview: Mark de VosFunctie: Operationeel directeur

**Interviewer** Jij bent verantwoordelijk voor de operationele tak van Finalist. Welke inzichten zou jij als operationeel directeur willen hebben om het software delivery proces te kunnen sturen?

**Mark** Er zijn twee aspecten waarvan ik denk dat die van invloed zijn op de kwaliteit van de software. Één daarvan is het aantal wisselingen op een project team.

**Interviewer** Wat houdt het aantal wisselingen per project team in?

**Mark**  Je ziet dat er binnen project teams veel gewisseld wordt. Dan vertrekt die ontwikkelaar en komt er een andere ontwikkelaar.

Interviewer Wat voor voordeel zou jij uit zo’n inzicht kunnen halen?

**Mark**  Dan hebben we dat inzichtelijk. We zouden dan kunnen zien of er projecten zijn waar er veel problemen zijn. Is er een link te vinden met het aantal wisselingen per project plaats vind. Iedereen denkt wel dat het zo is, maar we hebben dat niet inzichtelijk. Een andere aspect die ik zou willen meten is hoeveel issues worden er gevonden in een release de eerste week na de productie name. Dus we hebben een release gemaakt en opgeleverd en hoeveel issues vinden gemiddeld de eerste week of de eerste twee weken na productie name. Hoe minder dat is hoe beter we het zelf getest hebben.

**Interviewer** Zou je als operationeel directeur niet willen weten bij welke klanten het allemaal goed gaat.

**mark** Ja klopt. De twee aspecten die ik heb genoemd die zou je dan per klant of per project moeten kunnen zien.

**Interviewer** Zou je op deze twee aspecten kunnen sturen?

**Mark** Ja. Wat ook wel interessant is voor het bedrijfsbureau is projectmarge.

**Interviewer** Wat is een projectmarge?

**Mark**  Stel je verkoopt een project voor een ton en dan gaan er allerlei mensen aan werken. Wat we willen is kijken hoeveel winst we maken op een project. We kijken nu alleen van je komt niet boven de ton. Je kunt je voorstellen dat als we projecten uitvoeren met alleen juniors, dat zou meer geld moeten opleveren.

**Interviewer** Dat kost meer geld toch?

**mark** Je wilt weten wat kost een project gemiddeld en dan kun je projectmanagers sturen van je hoeft niet altijd seniors in je team te hebben. Je kunt bijvoorbeeld één senior en een paar juniors hebben. Het zou voor de kosten interessant zijn. Als we nu een project draaien dan willen we de vijf meest ervaren mensen hebben.

**Interviewer** Dit wil jij inzichtelijk hebben?

**mark** Ja. Sterker nog daar gaat het hele bedrijf op gestuurd worden.

**Interviewer** Dit is nu niet inzichtelijk?

**mark** We krijgen vanaf januari wel een nieuw systeem. Daar kun je tarieven van juniors, mediors en senioren bij houden en kun je sturen op project marge.

**Interviewer** Dan zou dat middels het nieuwe systeem inzichtelijk worden.

**mark**  Ja, als het goed is wel.

**Interviewer** Alleen dit onderdeel komt in het nieuwe systeem, de voorgenoemde twee onderdelen niet?

**mark** Ja, de twee voorgenoemde niet. Daar is helemaal niks voor. Dit is misschien meer voor het bedrijfsbureau, maar we moeten eigenlijk meer 20 en 80 procent reviews doen. Dit om er voor te zorgen dat we vroegtijdig weten of iets goed gaat of niet.

**Interviewer** Dit is meer wat voor het bedrijfsbureau?

**mark** Ja dat klopt.

**Interviewer** Dit aspect komt niet in het nieuwe systeem?

**mark**  Nee. Een aspect wat we nog beter kunnen doen is als we gaan opleveren dat we niet alleen de software opleveren, maar ook het testplan(onze eigen test resultaten), kloppende release notes. Dus dat we een complete set opleveren getest door ons en dat we het dan pas aan de klant gaan overdragen. Wat nu vaak voorkomt is dat onze ontwikkelaars alles programmeren en dat het dan al heel snel naar de klant gaat. Er zou eigenlijk een stap tussen moeten zitten.

**Interviewer** Dit is meer iets dat het proces aangepast moet worden.

**mark**  Ja.

**Interviewer** Worden trouwens de wisselingen van de projecten bijgehouden?

**mark** Ja, in FUO en dat gaat naar het nieuwe systeem. Je zou eigenlijk op dashboard per klant willen zien op trend van het aantal issues, de trend van de project wisselingen, de trend van project marge.

### Interview: JoeFunctie: Usability expert

**Interviewer** Ik doe onderzoek naar het software delivery proces en hiervoor heb ik een aantal interviews afgenomen. Uit deze interviews zijn er een aantal inzichten naar voren gekomen. Gezien het feit dat ik graag zou willen weten hoe ik deze inzichten het beste kan presenteren, kom ik bij jou terecht.

Dit zijn de inzichten die naar voren zijn gekomen. Mijn vraag is dan ook hoe kan ik deze inzichten het beste presenteren?

**Joe** Als ik de lijst met inzichten zie, dan zie ik eigenlijk twee soorten type. Er zijn inzichten die data moeten weergeven en er zijn verhoudingen die weergegeven dienen te worden.

**Interviewer** Dat klopt inderdaad. Heb jij ook een idee op welke wijze dat het best gepresenteerd kunnen worden.

**Joe** De inzichten die moeten worden weergeven zijn eigenlijk ook te scheiden in twee soorten. We zien de maandelijkse inzichten en de inzichten over kortste, langste en gemiddelde duur van deployments en releases. Zou je mij kunnen vertellen wat er
precies onder maandelijkse releases/deployments wordt verstaan?

**Interviewer** Onder maandelijkse releases wordt verstaan dat de gebruikers moeten kunnen zien hoeveel releases/deployments er in een bepaalde maand zijn geweest. Er kunnen gegevens als de datum van de release/deployment, het tijdstip van de release/deployment en hoe lang zo’n release/deployment heeft geduurd worden vermeld.

**Joe** Als ik het goed begrijp moeten er dus voor een bepaalde maand alle releases/deployments bepaalde gegevens worden getoond.

**Interviewer** Ja dat klopt.

**Joe** In dat geval is het beste om deze gegevens in een lijst of een tabel vorm weer te geven, voor deployments een aparte tabel en voor releases een aparte tabel. Hiermee hebben de gebruikers dan inzicht in alle releases/deployments die in een bepaalde maand hebben plaats gevonden. Je gaf aan dat er voor elke release/deployment meerdere aspecten zullen worden laten zien. Dan is het beste om inzichten in een tabel vorm te presenteren waarin het voor de gebruikers wat bijvoorbeeld de waarde is van de datum van een release en de tijdstip wanneer zo’n release heeft plaatsgevonden te zien is. Ik weet niet hoe de structuur van je applicatie eruit komt te zien, maar je zou bijvoorbeeld de twee tabellen naast elkaar kunnen weergeven, zodat een gebruikers allebei de tabellen in één oogopslag kunnen zien.

**Interviewer** Dat is een helder antwoord. Laten we dan nu kijken naar de inzichten waarbij wordt aangegeven dat de gebruikers inzicht willen in de kortste, langste gemiddelde release/deployment per klant. Hoe kunnen die het beste gepresenteerd worden?

**Joe** Wat ik dus begrijp is dat gebruikers per klant de langste, kortste en gemiddelde duur van een release/deployment willen kunnen inzien.

**Interviewer** Ja, dat klopt. Er is vanuit die interviews aangegeven dat ze dat per klant willen kunnen inzien.

**Joe** In dat geval zou je deze inzichten het beste kunnen samenvoegen dat de gebruikers dit in één oogopslag kunnen zien. Je geeft aan dat het drie onderdelen zijn die de gebruikers willen kunnen zien. In dit geval zou ik zeggen dat je deze onderdelen het beste in een grafiek kunt weergeven, waarbij je deze drie items weergeeft. Dit zou bijvoorbeeld met behulp van een staaf diagram of een lijndiagram weergegeven kunnen worden. Vervolgens zou je deze items op een bepaalde manier kunnen weergeven. Je zou de kortste aan de meest linkerzijde, het gemiddelde duur in het midden en de langste duur aan de meest rechterzijde weergeven. Dus in de volgorde komt als eerst de kortste, dan de het gemiddelde en dan de langste duur.

**Interviewer** Ok, dat is een helder antwoord. Op deze manier had ik er nog niet naar gekeken. Je gaf net al aan dat er ook inzichten zijn die verhoudingen moeten weergeven. Hoe zouden die het beste gepresenteerd kunnen worden?

**Joe** Ik zie dat er inzichten in de verhouding van de duur van de releases en deployments moeten worden weergegeven.

**Interviewer** ja, dat klopt inderdaad.

**Joe** In dit geval zou je de verhoudingen het beste in grafieken kunt weergeven. Ik zou hierbij wel een aparte grafiek voor releases en een aparte grafiek voor deployments maken. Aan de hand van de hoeveelheid klanten kan je dan vervolgens ook nog besluiten om niet alle klanten te tonen in de grafiek, omdat als er veel klanten zijn dan komen ze niet allemaal op het scherm. Als dat het geval is dan zou je ervoor kunnen kiezen om bijvoorbeeld zeven klanten met de hoogste duur van releases/deployments en zeven klanten met de laagste releases/deployments kunnen opnemen in de grafiek. Wat ook nog vervolgens zou kunnen, ik weet natuurlijk niet wat precies de structuur wordt van je applicatie, maar je zou dan de twee grafieken naast elkaar weergeven Hiermee heeft de gebruiker dan gelijk inzicht in en de releases en de deployments.

**Interviewer** Ik weet niet precies hoeveel klanten er zijn, maar er kunnen natuurlijk in de toekomst veel klanten bijkomen.

**Joe**  Inderdaad.

**Interviewer** Hoe kunnen de andere verhoudingen het beste gepresenteerd worden?

**Joe** De andere verhoudingen zijn eigenlijk vergelijkbaar met die we zojuist hebben besproken. Inzicht in de verhoudingen tussen projecten is vergelijkbaar. Ook die zou je het beste in een grafiek kunnen weergeven. Ook hierbij moet je nagaan hoeveel projecten er zijn en kunnen ze wel allemaal op het scherm worden getoond. Hierin kun je vervolgens een afweging maken of je ze allemaal laat zien of alleen bijvoorbeeld zeven projecten met de langste duur van deployments/releases en zeven projecten met laagste duur van deployments/releases.

#  16 Bijlage security

In dit onderdeel zullen de meest voorkomende kwetsbaarheden volgens verschillende bronnen worden vermeld.

### OWASP

1. Injection
2. Broken Authentication and Session Management
3. Cross-Site Scripting
4. Insecure Direct Object References
5. Security Misconfiguration
6. Sensitive Data Exposure
7. Missing Functional Level Acces Control
8. Cross-Site Request Forgery
9. Using Components with Known Vulnerabilities
10. Unvalidated Redirect and Forwards

bron: <https://www.owasp.org/index.php/Top10#OWASP_Top_10_for_2013>

###  Cyber Security Consulting Blog

1. Lack of Data Validation and Data Cleansing
2. Broken or Lacking Acces Controls
3. Broken Authentication and Session Management
4. Cross-Site Scripting
5. Buffer OverFlow Vulnerabilities
6. Code Injection and SQL injections Vulnerabilities
7. Poor or Lacking Error Handling
8. Insecure & Improper Data & File Storage
9. Denial of Service Vulnerability
10. Lacking or Poor Configuration Management

Bron: <http://blog.emagined.com/2010/03/22/top-ten-most-critical-web-application-security-vulnerabilities/>

### Cyber Security Nexus

* Authentication
* Session Security and Session IDs
* SQL injection
* Buffer overflows
* Cross- Site Scritping
* Error handling
* Remote web site administration
* Denial of service
* Storage

Bron: <http://www.isaca.org/Journal/Past-Issues/2005/Volume-4/Pages/Common-Web-Application-Vulnerabilities1.aspx>

### Edgescan

1. Cross-Site Scripting(XSS)
2. Content Injection
3. Authorisation
4. CSRF
5. Session Management
6. Information Leakage
7. Authentication
8. SQL Injection
9. Other Injection

Bron: <http://www.bccriskadvisory.com/wp-content/uploads/Edgescan-Stats-Report.pdf>

**Cenzic**

1. Cross-Site Scripting(XSS)
2. Information Leakage
3. Authentication and Authorisation
4. Session Management
5. SQL Injection
6. CSRF
7. Other

Bron: <http://www.cenzic.com/downloads/Cenzic_Vulnerability_Report_2014.pdf>

## Vergelijken van kwetsbaarheden tussen de verschillende bronnen

In dit onderdeel zullen de verschillende termen voor de kwetsbaarheden van webapplicatie door verschillende bronnen zijn genoemd met elkaar worden vergeleken. Om dit te doen is er gekozen om de top tien kwetsbaarheden van webapplicaties op te sommen en te beschrijven. Hierdoor kan vervolgens de beschrijvingen van de kwetsbaarheden vergeleken worden met de beschrijvingen van OWASP.

Tabel 1: Beschrijvingen van kwetsbaarheden van webapplicaties

|  |  |  |
| --- | --- | --- |
| **Nummer** | **kwetsbaarheid** | **Beschrijving** |
| 1 | Injection | Een injection is een aanval waarbij een hacker gemanipuleerde code naar een interpreter stuurt om data die opgeslagen is in een bepaalde bron te manipuleren. Injection aanvallen worden bijvoorbeeld veelal bij SQL, LDAP en Xpath aangetroffen. Afhankelijk van de bron waar data wordt bewaard kunnen injection aanvallen worden uitgevoerd.  |
| 2 |  Broken Athentication and Session Management | Hackers proberen middels het uitvoeren van aanvallen om specifieke gevoelige gegevens van gebruikers te verkrijgen. Wachtwoorden en session ids zijn hier voorbeelden van. Als een hacker deze gegevens verkrijgt kan hij namens een gebruiker verschillende handelingen op een website verrichten. Een hacker is in staat om deze gegevens te verkrijgen, omdat session ids en wachtwoorden niet goed beveiligd worden. |
| 3 |  Cross Site Scripting | Dit is een techniek waarbij hackers client-side scripts invoeren om gevoelige informatie van gebruikers te verkrijgen. De reden waarom een applicatie vatbaar kan zijn voor deze aanval is, omdat de invoer van gebruikers niet goed wordt gevalideerd en hierdoor kan het bij de uitvoer van de eindgebruiker komen. Een hacker zou middels het toepassen van deze techniek bijvoorbeeld een sessie van een gebruiker over kunnen nemen of een Cookie kunnen achterhalen. |
| 4 |  Insecure Direct Object References | Deze aanval kan worden uitgevoerd door een geautoriseerde gebruiker en ontstaat doordat een applicatie niet keer op keer controleert wie de gebruiker is of hij op een bepaalde pagina mag komen. Een voorbeeld hiervan is dat een geautoriseerde gebruiker zich op een pagina bevind waar hij één van zijn gegevens kan veranderen, maar deze gebruiker verandert in de URL het id en kan de gegevens van een andere gebruiker veranderen. |
| 5 | Security Misconfiguration | Dit kan ontstaan als een software component vatbaar is voor aanvallen door een niet veilige configuratie optie. |
| 6 | Missing Functional Level Acces Control | Een geautoriseerde gebruiker kan een URL aanpassen of een parameter van een functie die specifiek voor bepaalde gebruikers is. Hierdoor kan een hacker op een webpagina komen waar hij niet mag komen. |
| 7 | Sensitive data exposure | Dit ontstaat als gevoelige informatie niet met cryptografische technieken worden beveiligd. Daarnaast kan het ontstaan, omdat er bijvoorbeeld zwakke algoritmes worden gebruikt. |
| 8 | Using components with known vulnerabilities | Een applicatie maakt gebruik van een bepaalde framework libraries. Deze libraries kunnen bepaalde lekken waardoor een hacker hiermee bepaalde aanvallen kan uitvoeren. Een applicatie dient gebruik te maken van framework libraries waarvan het bekend is dat ze veilig zijn. |
| 9 | Cross Site Request Forgery | Een hacker formuleert valse requests en probeert vervolgens om ervoor te zorgen dat gebruikers deze valse requests uitvoeren. |
| 10 | Unvalidated redirects and forwards | Webapplicaties sturen gebruikers na een bepaalde request door naar andere webpagina’s. Een hackers kan door middel van het meegeven van een andere URL een gebruiker naar een andere website lokken. |

Tabel waarin de beschrijvingen van de kwetsbaarheden van de verschillende bronnen vergeleken worden met de beschrijvingen van de kwetsbaarheden van OWASP. Het nummer geeft het nummer weer van de tabel waarin de beschrijvingen van de kwetsbaarheden staan volgens OWASP.

Tabel 2: Vergelijking begrippen kwetsbaarheden

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ***Cyber Security Consultant Blog*** | ***Cyber Security Nexus*** | ***Edgescan*** | ***Cenzic*** |
| 1 | Code injection and SQL injection vulnerabilities | SQL injection | SQL injection | SQL injection |
| 2 | Broken authentication and session management | -Authentication-Session Security and Session IDs | -Authorisation-Session management-Authentication | -Authentication and authorization-Session management |
| 3 | Cross-Site Scripting | Cross-Site Scripting | Cross-Site Scripting | Cross-Site Scripting |
| 4 | Broken or Lacking Acces Controls |  |  |  |
| 5 | Lacking or Poor Configuration Management |  |  |  |
| 6 | Broken or Lacking Acces Controls |  |  |  |
| 7 | Insecure & improper Data & File Storage | Storage | Information Leakage | Information Leakage |
| 8 | Buffer Overflow vulnerabilities |  |  |  |
| 9 |  |  | Cross Site Request Forgery(CSRF) | Cross Site Request Forgery(CSRF) |
| 10 |  |  |  |  |

In de tabel is te zien dat sommige onderwerpen uit andere bronnen bij OWASP juist samengevoegd zijn. De onder waar een streep voor staat zijn deze onderwerpen.

#  17 Bijlage Plan van aanpak

Plan van Aanpak

*Het software delivery proces*

Student: Yassine Ben-Moussa
Studentnummer: 196411
Bedrijf: Finalist
Docentbegeleider: Elfriede Krauth
Bedrijfsbegeleider: Brian Snijders

InhoInhoudsopgaveopgave

[Inhoudsopgave 92](#_Toc406506643)

[Inleiding 93](#_Toc406506644)

[1 Opdracht 94](#_Toc406506645)

[1.1 Aanleiding 94](#_Toc406506646)

[1.2 Doel van de opdracht 94](#_Toc406506647)

[2 onderzoeksvragen 95](#_Toc406506648)

[2.1 Het software delivery proces 95](#_Toc406506649)

[2.2 Hoofdvraag 96](#_Toc406506650)

[2.2 Deelvragen 96](#_Toc406506651)

[3 Aanpak 97](#_Toc406506652)

[3.1 Onderzoek 97](#_Toc406506653)

[3.1.1 Software delivery proces 97](#_Toc406506654)

[3.1.2 Software delivery proces binnen Finalist 98](#_Toc406506655)

[3.1.2 Management Finalist 98](#_Toc406506656)

[3.1.3 Project teams 98](#_Toc406506657)

[3.1.4 Security 99](#_Toc406506658)

[3.1.5 Usablility 99](#_Toc406506659)

[3.2 Realisatie 100](#_Toc406506660)

[4 Doelstellingen/oplevering 101](#_Toc406506661)

[4.1 Doelstelling 101](#_Toc406506662)

[4.2 Resultaat 101](#_Toc406506663)

[5 Finalist 103](#_Toc406506664)

[5.1 Beschrijving organisatie 103](#_Toc406506665)

[5.2 Software 103](#_Toc406506666)

[5.3 Organogram 104](#_Toc406506667)

[6 Risicomanagement 105](#_Toc406506668)

[7 Planning 106](#_Toc406506669)

[8 Algemeen 108](#_Toc406506670)

#  Inleiding

Voordat ik daadwerkelijk ga starten met het uitvoeren van de afstudeeropdracht, is het belangrijk en een vereiste om een goed plan van aanpak op te stellen. Dit heeft als doel om gestructureerd de opdracht te kunnen uitvoeren. Binnen dit document worden verschillende aspecten uitgelegd en geformuleerd. Dit moet er toe leiden dat de opdrachtgever, docentbegeleider en ik op één lijn zitten om vervolgens de opdracht succesvol te kunnen realiseren.

#  1 Opdracht

In dit hoofdstuk wordt de opdracht uitgelegd. Het doel van dit hoofdstuk is om aan te geven waarom de opdracht tot stand is gekomen, het probleem te verduidelijken, het doel van de opdracht te formuleren en om de gewenste situatie kenbaar te maken.

1.1 Aanleiding
Op dit moment is het software delivery proces niet inzichtelijk. Hierdoor heeft Finalist niet de mogelijkheid om dit proces te sturen. Aangezien Finalist haar dienstverlening verder wilt verbeteren, wil Finalist inzichten van het software delivery proces tot haar beschikking hebben om te kunnen analyseren welke aspecten van het software delivery proces verbeterd moeten worden.

1.2 Doel van de opdracht
Finalist wilt haar dienstverlening verder verbeteren. Het software delivery proces speelt hierin een belangrijke rol. Om het software delivery proces te verbeteren heeft Finalist inzichten nodig om dit proces te kunnen sturen. Het doel is om te bepalen welke inzichten nodig zijn om dit proces te kunnen sturen. Vervolgens moet er een proof of concept gerealiseerd worden die deze inzichten weergeeft. Daarnaast moet de proof of concept veilig en makkelijk te gebruiken zijn voor gebruikers.

1.3 Kwaliteitseisen
Het is belangrijk om zorg te dragen voor het feit dat de gebruikers optimaal van de applicatie gebruik kunnen maken. Om dit te bewerkstelligen is het noodzakelijk om vooraf te bepalen wat de eisen zijn. De eisen die hieraan worden gesteld zijn:

* De proof of concept moet de juiste inzichten van het software delivery proces weergeven. Deze inzichten moeten uit het gebruikersonderzoek herleidbaar zijn.
* De proof of concept dient goed te werken. Dit houdt in dat er geen foutmeldingen moeten zijn en dat de functionaliteiten niet weigeren te werken.
* De proof of concept moet makkelijk te gebruiken zijn voor de gebruikers. Dit houdt in dat de applicatie aan de normen van usability moet voldoen.
* De proof of concept dient beveiligd te worden tegen hackers, omdat de applicatie online zal draaien.

#  2 onderzoeksvragen

Het onderzoek zal uit twee fases bestaan: Het gebruikersonderzoek en de onderzoeken naar usability en security. Het gebruikersonderzoek wordt gedaan om te achterhalen naar welke inzichten de gebruikers op zoek zijn, de onderzoeken naar usability en security worden gedaan, omdat de proof of concept makkelijk te gebruiken moet zijn en de proof of concept online zal draaien. Hieronder zullen de onderzoeksvragen worden vermeld.

## 2.1 Het software delivery proces

Binnen bedrijven worden wordt gekeken op welke manier software zo snel mogelijk opgeleverd kan worden. Het opleveren van software kan uit verschillende fases en onderdelen bestaan. Het proces waarin software wordt opgeleverd, wordt het software delivery proces genoemd. Iedere organisatie kan een andere implementatie hebben voor het opleveren van software.

Finalist heeft ook een proces ingericht voor het opleveren van software. Binnen Finalist bestaat het software delivery proces uit een aantal fases en onderdelen.

Het onderzoek zal zich op het software delivery proces richten. De fases die hierboven zijn getoond zijn hierin cruciaal. Alleen de zaken die een directe relatie hebben met de fases van het software delivery proces en die duidelijk meetbaar zijn zullen in de resultaten worden meegenomen. Het onderzoek zal zich dus niet richten op het verbeteren van alle processen binnen Finalist of op zaken die een indirecte relatie hebben met het software delivery proces. Het kan zijn dat er uit de interviews informatie naar voren komt die wel te maken heeft met het software delivery proces, maar een indirecte relatie heeft. Zulke zaken zullen niet worden meegenomen bij het realiseren van de applicatie, maar kunnen misschien wel van belang zijn bij het geven van een aanbeveling.

***Bron***: Interview met Brian Snijders

2.2 Hoofdvraag
Hoe kan het software delivery proces inzichtelijk worden gemaakt, op een voor gebruikers veilige en gebruikersvriendelijke wijze, zodat Finalist het software delivery proces kan verbeteren?

2.3 DeelvragenOm de hoofdvraag te kunnen beantwoorden dienen er een aantal deelvragen te worden beantwoord. Hieronder volgen de deelvragen.

* Wat is een software delivery proces en waaruit bestaat dit proces?
* Wat is een software delivery proces volgens de literatuur?
* Welke inzichten zijn nodig om het software delivery proces inzichtelijk te maken?
* Wat zijn de meest voorkomende kwetsbaarheden van webapplicaties volgens de literatuur?
* Welke onderdelen van de proof of concept moeten worden beveiligd?
* Hoe kunnen de onderdelen van de proof of concept worden beveiligd?
* Welke aspecten van usability zijn volgens de literatuur nodig voor de proof of concept?
* Hoe kunnen de usability aspecten volgens de literatuur toegepast worden, zodat de proof of concept gebruiksvriendelijk is?
* Hoe kunnen de inzichten van het software delivery proces het beste gepresenteerd worden?

#  3 Aanpak

De uitvoering van het project zal bestaan uit twee fases: Het onderzoeksgedeelte en de realisatiegedeelte.

De onderzoeksfase bestaat uit twee onderdelen: Gebruikersonderzoek en onderzoek naar security en usability. Dit is een fase waarin achterhaald dient te worden welke inzichten nodig zijn om het software delivery proces inzichtelijk te maken. In deze fase is het zaak om potentiële gebruikers te interviewen om deze inzichten boven water te krijgen en te kijken hoe dit het best geautomatiseerd kan worden. De volgende stap is om te onderzoeken hoe de applicatie beveiligd kan worden tegen aanvallen van hackers en om te onderzoeken hoe de applicatie gebruiksvriendelijk gerealiseerd kan worden.

De realisatiefase zal verdeeld worden in een aantal sprints. Om deze fase goed te laten verlopen zal er gebruik worden gemaakt van de methode scrum. Met behulp van deze methode zal er iteratief worden gewerkt. Het voordeel hiervan is dat functionaliteiten die af zijn worden opgeleverd en aan de belanghebbende worden tentoongesteld. Hierdoor kunnen de belanghebbende tijdig hun feedback op de functionaliteiten geven.

3.1 Onderzoek
In deze fase is het zaak om alle benodigde informatie te vergaren die nodig zijn om een proof of concept te bouwen. In deze fase moeten de deelvragen beantwoord worden. Voor het realiseren van de proof of concept zal er gebruik gemaakt worden van de methode scrum. Met behulp van deze methode krijgen de belanghebbende tijdig de gerealiseerde functionaliteiten te zien, kunnen de belanghebbende hierop feedback geven en kunnen de functionaliteiten naar aanleiding van de feedback aangepast worden. Daarnaast is het een vereiste dat er een bepaalde set aan functionaliteiten gerealiseerd word, een volledig uitgedachte applicatie zonder gerealiseerde functionaliteiten willen we voorkomen.

## 3.1.1 Software delivery proces

* Wat is een software delivery proces en waaruit bestaat dit proces volgens de literatuur?

**Methodiek** Het is zaak om te achterhalen wat volgens de literatuur een software delivery
proces, omdat ik onderzoek ga doen naar het software delivery proces en hiervoor moet ik weten wat een software delivery proces is en waaruit het bestaat. Hierdoor weet ik wat een software delivery proces is.

**Deliverable** Definitie met een duidelijke beschrijving van het software delivery proces. Daar naast zal er ook een flow model worden opgeleverd waarin duidelijk te zien hoe het opleveren van software in zijn werking gaat. De deelvraag is goed beantwoord als er een duidelijke beschrijving is van dit proces.

3.1.2 Software delivery proces binnen Finalist
Het doel is om boven water te krijgen wat er verstaan wordt onder het software delivery proces en op welke manier er software wordt opgeleverd binnen Finalist.

* Wat is een software delivery proces en waaruit bestaat dit proces binnen Finalist?

**Methodiek** In eerste instantie is het zaak om met Brian Snijders(begeleider) te achterhalen wat er wordt verstaan onder een software delivery proces, omdat ik onderzoek ga doen naar het software delivery proces en hiervoor moet ik weten wat een software delivery proces is en waaruit het bestaat. Hierdoor weet ik wat een software delivery is en kan ik uit de interviews weten wat ik in de resultaten kan meenemen.

**Deliverable** Definitie met een duidelijke beschrijving van het software delivery proces. Daar naast zal er ook een flow model worden opgeleverd waarin te zien hoe op welke manier software wordt opgeleverd. Dit moet duidelijk uit het interview herleidbaar zijn. De deelvraag is goed beantwoord als er een goede beschrijving van het software delivery proces is.

3.1.3 Management Finalist
Het doel is om vast te stellen welke inzichten het management van het software delivery proces wil hebben. De deelvragen die bij dit onderdeel horen zijn:

* Welke inzichten van het software delivery proces willen medewerkers van Finalist?

**Methodiek** Het management van Finalist zal worden geïnterviewd. De operationeel directeur en de unit manager zijn de twee functies uit het management die zullen worden geïnterviewd, omdat zij betrokken zijn bij het software delivery proces.

**Deliverable** Een lijst van wensen en eisen van het management waarin er onderscheid wordt gemaakt tussen welke wensen eisen een hoge prioriteit hebben en welke een lage prioriteit. Dit moet duidelijk uit de interviews herleidbaar zijn. Alleen de zaken die een directe relatie hebben met het software delivery proces hebben worden meegenomen in de resultaten. De interviews zullen als bijlage worden bewaard. De deelvragen zijn juist geantwoord als het duidelijk is welke inzichten van het software delivery proces medewerkers van Finalist willen.

3.1.4 Project teams
Het doel is om te achterhalen welke inzichten zij van het software delivery proces willen. Dit onderdeel moet antwoord geven op de deelvragen:

* Welke inzichten van het software delivery proces willen de medewerkers van Finalist?

**Methodiek** Het projectteam van Finalist zal worden geïnterviewd. Onder een project team wordt verstaan: Projectmanager, solution expert, developer en analist.

**Deliverable** Een lijst met wensen en eisen van het projectteam waarin er onderscheid wordt gemaakt tussen welke wensen en eisen een hoge prioriteit hebben en welke een lage prioriteit. Dit moet duidelijk herleidbaar uit de interviews zijn. De interviews zullen als bijlage worden bewaard.

3.1.5 Security
Het doel is om te achterhalen hoe de applicatie beveiligd kan worden tegen hackers. Niet alle zaken van security zullen onderzocht worden, maar alleen de zaken die van belang zijn voor het realiseren van de applicatie. Mogelijke onderwerpen waarnaar onderzoek zal worden gedaan zijn: SQL injection, Cross Site scripting en Broken authentication. Dit onderdeel moet antwoord geven op de deelvraag:

* Wat zijn de meest voorkomende kwetsbaarheden van webapplicaties volgens de literatuur?
* Welke onderdelen van de proof of concept moeten worden beveiligd?
* Hoe kunnen de onderdelen van de proof of concept beveiligd worden?

**Methodiek** De website van OWASP zal worden geraadpleegd. OWASP is een open source organisatie die als doel heeft om organisaties te helpen om de veiligheid van software te verbeteren. Grote bedrijven en banken zoals: ING, Rabobank en Capgemini hanteren de beveiligingsrichtlijnen van OWASP. Ook binnen Finalist wordt de site van OWASP geraadpleegd.

**Deliverable** Rapport(onderdeel van scriptie) waarin de bevindingen van de literatuurstudie wordt vastgelegd. De deelvragen zijn juist beantwoord als het duidelijk is wat de meest voorkomende kwetsbaarheden van webapplicaties zijn en beveiligd moet worden tegen aanvallen van hackers.

3.1.6 UsablilityHet doel is om te onderzoeken welke richtlijnen gehanteerd moeten worden om er voor te zorgen dat een applicatie gebruiksvriendelijk is. Niet alle zaken van usability zullen onderzocht worden, maar alleen de zaken die van belang zijn voor het realiseren van de applicatie.

Deze onderdelen zullen worden onderzocht. Dit onderdeel moet antwoord geven op de volgende deelvragen.

* Welke aspecten van usability zijn volgens de literatuur nodig voor de proof of concept?
* Hoe kunnen de usability aspecten volgens de literatuur toegepast worden, zodat de proof of concept gebruiksvriendelijk is?
* Hoe kunnen de inzichten van het software delivery proces het beste gepresenteerd worden?

**Methodiek** Er zal literatuurstudie worden gedaan om te achterhalen hoe een applicatie gebruiksvriendelijk gerealiseerd kan worden. Het boek “Don’t make me think” van Steve Krug zal geraadpleegd worden. Steve Krug is een Informatie architect en user experience professional. Dit boek staat in de top 10 beste usability boeken. In dit boek staan alle usability onderdelen om een gebruiksvriendelijke website te kunnen maken. Daarnaast zullen ook usability experts geraadpleegd worden om te achterhalen hoe de inzichten van het software dellivery proces het beste gepresenteerd kunnen worden.

**Deliverable** Rapport(onderdeel van scriptie) waarin de bevindingen van de literatuurstudie wordt vastgelegd. Een bepaalde aantal aspecten van usability dienen opgeleverd te worden. De interviews met de usability experts zullen als bijlage worden bewaard. De deelvragen zijn juist beantwoord als het duidelijk is hoe een applicatie gebruiksvriendelijk gerealiseerd kan worden en hoe de inzichten van het software delivery proces het best gepresenteerd kunnen worden.

## 3.2 Realisatie

De tools die zullen worden gebruikt zijn voor het realiseren van de applicatie zijn:

Programmeertaal: JAVA
Database: MySQL
Framework: Spring MVC
IDE: Eclipse

Gezien het feit dat Finalist software ontwikkelt met open source producten, zal ik ook voor het realiseren van de proof of concept gebruiken maken van open source producten. MYSQL, Spring MVC en Eclipse zijn allen open source producten. JAVA is een programmeertaal die verenigbaar is met veel open source producten.

#  4 Doelstellingen/oplevering

Voor het afstudeerproject dienen er doelstellingen vastgesteld te worden. Daarnaast dienen er ook een aantal zaken opgeleverd te worden.

4.1 Doelstelling

***Opdrachtgever***Met behulp van dit onderzoek moet er achterhaald worden naar welke inzichten de verschillende belanghebbende op zoek zijn en een proof of concept gerealiseerd worden. Daarnaast met de applicatie beveiligd zijn en makkelijk te gebruiken zijn.

***Opdrachtnemer***Middels dit project wil ik bewijzen dat ik in staat ben om individueel een onderzoek te doen, het onderzoek op een duidelijke manier te rapporteren en aan de hand van de resultaten van het onderzoek een proof of concept te realiseren. Uiteraard wil ik middels dit project de opleiding succesvol afronden.

4.2 resultaat

***Rapport***Finalist wilt weten hoe het software delivery proces inzichtelijk kan worden gemaakt en hoe de applicatie op een veilige en makkelijke manier kan worden gebruikt. Het rapport zal antwoord geven op de hoofd en deelvragen. Binnen het rapport worden alle bevindingen vastgelegd en worden de keuzes onderbouwd.
 ***Proof of concept***Naast het rapport zal er een proof of concept opgeleverd worden. De proof of concept zal voortvloeien uit de resultaten van het onderzoek.

***Producten***Om de opdracht succesvol af te kunnen worden wordt er van mij verwacht dat ik een aantal producten oplever.

Voor de hogeschool Utrecht

* Plan van aanpak
* Scriptie
* Presentatie

*Voor de opdrachtgever*

* Proof of concept die de inzichten van het software delivery proces weergeeft
* Onderzoeksrapport

#  5 Finalist

In dit hoofdstuk zal ik wat vertellen over Finalist. Ik zal een beschrijving geven van wat Finalist doet en hoe de organisatie in elkaar zit.

5.1 Beschrijving organisatie
Finalist is een bedrijf dat is opgericht in 1988 en ongeveer tachtig medewerkers heeft. Ze hebben een passie voor software ontwikkeling. Hun doel is om software te realiseren die aan de gebruikerseisen voldoen. Om software te ontwikkelen kiezen zij ervoor om gebruik te maken van open source producten. Ze zijn altijd op de hoogte van de laatste technieken op het gebied van open source producten.

De klanten van Finalist bevinden zich door heel Nederland. Hierdoor heeft Finalist ervoor gekozen om een vestiging in Maarssen, Rotterdam en Eindhoven te openen. Hiermee trachten zij hun klanten op de best mogelijke wijze van dienst te zijn.

Finalist realiseert projecten die vaak groot en complex van aard zijn. Gezien het feit dat deze groot en complex zijn, heeft Finalist verschillende functies binnen een team. Projectmanagers, testers, analisten, ontwerpers en developers zijn functies die binnen een team van Finalist bekleed worden. Hiermee wordt getracht om de projecten in goede banen te leiden.

## 5.2 Software

Finalist realiseert software voor veel klanten. De software die zij realiseren kunnen in drie categorieën worden verdeeld. Hieronder worden deze categorieën beschreven.

***Websites en portalen***
Voor het realiseren van websites en portalen gebruikt Finalist vaak een CMS. Dit zijn Open Source producten. Voorbeelden van deze CMS’en zijn: Hippo(JAVA), Dupal(PHP) en Liferay Portal.

***Applicatie integratie***
Met zowel de ontwikkelkennis als de specifieke integratiekennis in huis, zorgt Finalist er ervoor dat de applicatie integratie van de software vlekkeloos verloopt en alle systemen moeiteloos met elkaar kunnen “praten”. Hiervoor maken de experts koppelingen tussen software-systemen, beantwoorden zij integratie-vraagstukken en krijgen de klanten alle hulp met de authenticatie.

***Maatwerk software***
De wensen en eisen van de organisaties staan centraal bij de bouw van gebruiksvriendelijke webapplicaties, zodat deze voor de klanten volledig op maat zijn gemaakt. Zo heeft Finalist myTomorrows ontwikkeld: een applicatiesysteem waarmee uitbehandelde en chronisch zieke patiënten versneld toegang krijgen tot innovatieve medicijnen. Met deze op exclusief ontwikkelde applicatie worden de patiënten simpel door het complexe aanmeldsysteem geleid. Hierna informeert myTomorrows de patiënt op basis van hun gegevens over behandelingen waarvoor hij of zij in aanmerking kan komen.

5.3 Organogram
Finalist heeft inmiddels tachtig medewerkers. Om een beter beeld te geven van hoe de organisatie in elkaar zit zal hieronder een organogram te zien zijn.


Het managements is de hoogste rang binnen Finalist en is verantwoordelijk voor de sturing van Finalist. Binnen het management zijn er verschillende functies.

Het bedrijfsbureau is verantwoordelijk voor de uitvoering van projecten. Het ervoor zorgen dat er genoeg projectleiders en developers zijn, zijn voorbeelden van de verantwoordelijkheden van het bedrijfsbureau.

Een unit bestaat uit een unit manager, een solution expert en de uitvoerings functies. Developers, analisten en testers zijn voorbeelden van uitvoerende functies. Unit PII staat voor portals integration and information en is verantwoordelijk voor data en visualisatie en integratie. Unit CD staat voor Content Delivery en is verantwoordelijk voor cmsen, websites, online strrategie en ontwerp. Unit CS staat voor Custom Solutions en is verantwoordelijk voor maatwerk klussen. De beheer unit is verantwoordelijk voor alle projecten die in beheer zijn gegaan.

#  6 Risicomanagement

Binnen elk project kunnen er zaken optreden die van te voren niet gepland zijn. Het is daarom handig om rekening te houden met zaken waar je mee te maken kan krijgen tijdens het uitvoeren van een opdracht. Deze zaken worden ook wel als risico’s aangeduid. Hieronder zal ik een aantal risico’s vermelden.

|  |  |  |
| --- | --- | --- |
| ***Oorzaak*** | ***Maatregel*** | ***Wanneer*** |
| De toegevoegde functionaliteiten zullen niet helemaal goed werken. | Testen. | Na het realiseren van een functionaliteit. |
| Belanghebbende willen niet meewerken met interviews | Tijdig benaderen | Ruim op tijd benaderen. |
| De functionaliteiten zullen niet binnen de tijd gerealiseerd worden. | Prioriteiten stellen en in de avonduren doorwerken. | Als blijkt dat de planning niet gehaald gaat worden. |
| Onjuiste interpretatie van gebruikerswensen. | Functionaliteiten tijdig aan belanghebbende laten zien en feedback van hen verwerken. | Na het realiseren van een functionaliteit. |
| Belanghebbende zijn niet tevreden met resultaat | Aanpassen naar wens van belanghebbende | Als blijkt dat belanghebbende niet tevreden zijn met resultaat |

#  7 Planning

Om het afstudeertraject in goede banen te leiden is het noodzakelijk om een planning te maken. Een planning biedt structuur en een houvast voor mij. Hieronder is te zien dat ik een planning heb gemaakt met de onderdelen die gedaan moeten worden.

|  |  |  |
| --- | --- | --- |
| ***Week*** | ***Product*** | ***Opleverdatum*** |
| 46 t/m week 51 | Schrijven plan van aanpak | 17 december |
| 48  | Interview Project team | 27 november |
| 49 | Concept securityInterview managementInterview project team |  |
| 52 t/m 2 | Onderzoek Interview projectteamInterview managementteamSprint 1 |  |
| 3 | Onderzoek security onderzoek verwerken in scriptieOpleveren gebruikersonderzoekOpleveren onderzoek securitySprint 2 |  |
| 4 | Onderzoek usabilityOnderzoek verwerken in scriptieOpleveren onderzoek usabilitySprint 3 |  |
| 5 | Opleveren concept scriptieSprint 4 |  |
| 5  | Sprint 5Sprint 1,2 en 3 verwerken in scriptie |  |
| 6 | Sprint 5 |  |
| 7 | Sprint 4 en 5 verwerken in scriptie |  |
| 8 t/m 10 | Scriptie schrijvenConcept Scriptie inleveren |  |
| 11 | Scriptie  | 17 maart |
| 14 | Presentatie | 28 maart |

#  8 Algemeen

In dit hoofdstuk worden de contactgegevens van mij, mijn docentbegeleider en bedrijfsbegeleider vastgelegd.

**Gegevens opdrachtgever en opdrachtnemer**
De opdrachtnemer voor dit in plan van aanpak beschreven delen is:

|  |  |
| --- | --- |
|  Adres: | Hogeschool UtrechtNijenoord 13552AS Utrecht |
|  Student: | Yassine Ben-Moussayassine.ben-moussa@student.hu.nl0646867763 |
|  Docentbegeleider: | Elfriede Krauthelfriede.krauth@hu.nl |

De opdrachtgever voor de in plan van aanpak beschreven delen is:

|  |  |
| --- | --- |
|  Adres: | FinalistSafariweg 39/403605 MA Maarssen |
|  Contactpersoon: | Brian Snijdersbrian@finalist.nl0645504083 |