BUSINESS IN CONTROL

Afstudeerscriptie over het herontwerpen en het versnellen van de business dimension navigator

AFSTUDEERSCRIPTIE

R. Paltoe - 15025713

Kuru, K. | Hoek, J.J. van der
De Haagse Hogeschool, HBO-ICT
02 September 2019 – 10 Januari 2020

CERRIX B.V.

DE HAAGSE HOGESCHOOL

CERRIX

REFERAAT

Deze scriptie gaat over het maken van een nieuw ontwerp en het versnellen van de business dimension navigator module in de CERRIX-applicatie. Nadat het nieuwe ontwerp gemaakt was werd dit geïmplementeerd in de applicatie. Voorheen gebruikte CERRIX Ext.Net als haar framework, maar is er een overstap gemaakt naar Angular. Hierdoor veranderde de huisstijl van de applicatie en liep de navigator dus achter op de rest van de applicatie. Daarnaast was het huidige ontwerp onhandig in gebruik en aan vervanging toe.

Voor de versnelling van de business dimension navigator is er zowel gelet op een server-side als een client-side versnelling voor het filteren van de data en zijn deze apart geïmplementeerd. Vervolgens zijn ze tegen elkaar afgewogen om te kijken welke versnelling de beste oplossing bood en is dit gebruikt in de applicatie.

DESCRIPTOREN

Angular	TypeScript
C#	HTML
Performance	User experience
Agile	SCRUM
Visual Studio	Visual Studio Code
Client-side filtering	Server-side filtering

VOORWOORD

Voor u ligt de afstudeerscriptie “Het herontwerpen en het versnellen van de business dimension navigator”. Deze scriptie is geschreven in het kader van mijn afstuderen aan de opleiding HBO-ICT aan de Haagse Hogeschool en is vervuld bij CERRIX B.V. Van september 2019 tot en met januari 2020 is er over een periode van zeventien weken gewerkt aan de afstudeeropdracht en is deze scriptie tot stand gekomen.

Samen met de hulp van mijn bedrijfsmentor, Martin van Leeuwen is er gezocht naar de mogelijkheden voor zowel de versnelling voor het laden van de data, als de totstandkoming van het nieuwe ontwerp van de navigator. De uitvoering van de opdracht is enigszins als pittig ervaren, mede omdat Angular een nieuwe taal was wat aangeleerd moest worden. Na een analyse van de Angular componenten en het maken van enige ontwerpen is het gelukt een verbetering te leveren van de business dimension navigator module t.o.v. de situatie voorheen.

Bij deze wil ik graag mijn bedrijfsmentor, Martin van Leeuwen, en de begeleiders vanuit mijn opleiding, meneer Kuru en mevrouw van der Hoek bedanken voor de aangename begeleiding, ondersteuning en kritische feedback die ik heb mogen ervaren en ontvangen tijdens dit afstudeertraject. Daarnaast wil ik mijn collega’s van CERRIX, en in het bijzonder Jeffrey Borremans bedanken voor het nalezen en verstrekken van richtlijnen voor mijn scriptie. Zonder hen zou de scriptie niet geworden zijn wat het nu is.

Ik wens u, geachte lezer, veel leesplezier toe.

Rohit Paltoe,

Den Haag, 6 januari 2020.

SAMENVATTING

De business dimension navigator is een module binnen de CERRIX applicatie, waarin een gebruiker kan zien welke koppelingen er zijn tussen alle overige modules van de applicatie. In totaal zijn er zes modules: risico's, events, controls, moi's, kri's en data processings. In deze Navigator kan de gebruiker zijn weergave filteren op basis van een selectie voor een organisatie en business dimension. Omdat dit dé plek is waar de data uit alle modules samenkomen is dit het knooppunt binnen de applicatie. Alleen werd dit niet volledig benut, dit kwam door onderstaande redenen:

Ten eerste was de lay-out van de pagina onhandig vormgegeven. Klanten konden de pagina maar moeilijk in gebruik nemen, omdat er een overvloed aan informatie te zien was en ook was het filter onhandig in gebruik. Eveneens was de grafische weergave van de pagina verouderd ten opzichte van de rest van de applicatie. Dit kwam doordat CERRIX een overstap had gemaakt van een oudere en trage framework, Ext.Net naar een snelle en nieuwere, namelijk Angular. Hier is de huisstijl van de applicatie in mee verandert.

Dit probleem is opgelost door samen met de consultants tot een nieuw ontwerp te komen voor de navigator. Dit ontwerp voldeed aan de nieuwe huisstijl van CERRIX en was grafisch simpel vormgegeven, waardoor het gemakkelijker in gebruik genomen kon worden. Vervolgens was dit ontwerp geïmplementeerd in Angular (in HTML/ TypeScript en CSS). Hierdoor oogde de navigator al een flink stuk aangenamer.

Ten tweede duurde het laden van de data uit de zes modules van CERRIX zeer lang. Het kon tot wel 20 seconden duren, wat natuurlijk ongekend is voor een webapplicatie. Klanten zouden dit nooit kunnen gebruiken, en dit was ook het geval. De performance viel tegen en dit kwam doordat er aan de achterkant volledige tabellen sequentieel werden opgehaald per module waarna zij pas server-side gefilterd werden. Dit kostte allemaal extra tijd.

Dit is opgelost door het ophalen van de data per module om te schrijven van een sequentiële, naar een parallelle functie. Hierdoor werden de functies voor het opvragen van data uit de database tegelijkertijd uitgevoerd, en hoeven zij dus niet meer te wachten op elkaar. Ook is het filteren van de data zelf verbeterd. Er zijn twee implementaties gemaakt omdat het filteren om twee manieren kan gebeuren: namelijk één implementatie voor de server-side filtering, hierdoor wordt de server van CERRIX belast met het filteren en een client-side filtering, waardoor de pc van de gebruiker deze filtering zelf uitvoert. Uiteindelijk is er gekozen voor de client-side filtering, omdat de business dimension navigator nu maximaal 2 seconden nodig had voor het weergeven en filteren van data van de zes modules. Dit is een flinke verbetering ten opzichte van de oude situatie (het 20 seconden wachten).

Ten derde moest de te weergeven data voldoen aan de juiste rollen en rechten van de gebruiker. De applicatie bestaat uit een honderdtal van verschillende rollen en rechten door alle modules heen en voorheen kon de

gebruiker alle data zien dat gekoppeld is aan een proces, zelfs confidentiële, zonder dat er rekening werd gehouden met de rollen en rechten van de gebruiker.

Hier hoefde ik uiteindelijk geen rekening mee te houden, omdat de ontwikkelaars van CERRIX al de juiste functies hadden geschreven voor het ophalen van de data uit de zes modules op basis van de rollen en rechten van de gebruiker. Deze functies heb ik wel op mijn eigen manier geïmplementeerd, waardoor de navigator nu data laat zien op basis van de juiste rollen.

De ontwikkelmethode dat gebruikt wordt is SCRUM, omdat hiermee het werk in kleine cycli uitgevoerd kan worden en er hierdoor gemakkelijker en sneller feedback verkregen kan worden op het uitgevoerde werk. Dit is handig, omdat het ontwerp nog niet vaststaat en from scratch gemaakt moest worden. Daarnaast kan er zo ook tijdig terugkoppeling verkregen worden betreft de verbetering voor de performance.

In zeventien weken is het gelukt om een nieuwe versie op te leveren van de navigator business dimension. De pagina voldoet aan de nieuwe huisstijl van CERRIX, is makkelijker te begrijpen, is omgeschreven in Angular en het laden van de data bedraagt nu 2 seconden. Hierdoor is de module nu volledig opgefrist in vergelijking met de situatie hiervoor en is zowel het team van CERRIX, als ik tevreden met het resultaat. Mijn implementatie zal meegenomen worden naar de volgende release van CERRIX en klanten zullen dit dan direct gaan gebruiken.

DE HAAGSE HOGESCHOOL

CERRIX

~~SAMENVATTING~~

Inhoudsopgave

Formatted: Right: 2,41 cm, Top: 2,5 cm, Bottom: 2,6 cm, Footer distance from edge: 1,27 cm, Different first page header

HOOFDSTUK 1: INLEIDING

In deze scriptie zullen de stappen die er naar geleid hebben om de business dimension navigator te innoveren aan bod komen. De business dimension navigator is het knooppunt binnen de CERRIX-applicatie. Hier worden er overzichten getoond van een door de gebruiker gefilterde bedrijfsproces, en de koppelingen tussen de overige modules binnen de applicatie. Het probleem van de navigator was dat het laden van deze overzichten zeer lang duurde, minimaal al 20 seconden en dat de getoonde overzichten onduidelijk waren vormgegeven. Daarnaast had CERRIX een overstap gedaan van het oude JavaScript framework, Ext. Net, naar een nieuwere en snellere variant, Angular. Hierdoor voldeed deze oude navigator niet meer aan de vernieuwde huisstijl. De business dimension navigator was aan opfrissing toe.

Het doel van het afstudeertraject is om de navigator te versnellen, te herontwerpen en dit ontwerp te implementeren in Angular, waardoor deze pagina weer aansluit aan de rest van de CERRIX-applicatie. Om tot de juiste keuzes voor zowel de versnelling als het ontwerp te komen zal er eerst aan bod komen welke ontwikkelmethode er gebruikt was, de voorkeur ging hiervoor uit naar SCRUM, vervolgens werden de nieuwe wensen vast gesteld waar het ontwerp aan moest voldoen, waarna er een nieuw ontwerp uit voortvloeide. Dit ontwerp werd uiteindelijk in Angular geïmplementeerd en dit eindproduct werd opgeleverd aan CERRIX, die hier tevreden mee was en deze vernieuwing mee heeft genomen naar de volgende klantrelease. Tot slot zal het eindproduct geëvalueerd worden, om te reflecteren op zowel het product, als het proces en worden hier de beroepstaken aangetoond.

HOOFDSTUK 2: PROBLEEMDOMEIN

In dit hoofdstuk zal het probleemdomein beschreven worden. Er zal verteld worden wat de business dimension navigator is, waarom de module onbruikbaar is, wat er hiervoor gedaan was om het probleem op te lossen, wat er zal gebeuren als het probleem niet wordt op gelost en tot slot zal de te vervullen afstudeeropdracht toegelicht worden.

Een goedlopende user experience (*nu: UX)* is van groot belang. Het maakt of kraakt een product. Het kan ervoor zorgen dat men klant wil worden en het product in gebruik wil nemen, of het product juist in zijn geheel wil ontlopen. Tijdens het afstuderen stond de module 'Business Dimension Navigator' van de CERRIX-applicatie centraal en moest de UX van deze webpagina verbeterd worden. Klanten konden de pagina maar moeilijk in gebruik nemen en CERRIX besloot dit probleem onder de loep te nemen. ~~,~~ ~~m~~Maar wat maakte de pagina nou volgens klanten 'onbruikbaar'? Wat voor oplossing heeft CERRIX om de pagina weer bruikbaar te maken?

§ 2.1 DE BUSINESS DIMENSION NAVIGATOR

Om te begrijpen waarom de business dimension navigator *(nu: B.D.N.)* niet makkelijk in gebruik te nemen is, moeten wij ons eerst verdiepen in de B.D.N. Waar de module voor dient, wat voor klanten van een organisatie de module kunnen gebruiken, en hoe de data geladen en weergeven wordt. Door hier naar te kijken kunnen wij vaststellen waardoor het op dit moment onbruikbaar is.

De B.D.N. is gemaakt om een overzicht te krijgen van alle bedrijfsprocessen binnen een organisatie (*zie figuur 2.1).* Er wordt een overzicht in tabelvorm weergeven van alle risico's, controles, verbeteringsmaatregelen, events en verwerkingsdoelen die gekoppeld zijn aan een proces, dat eerst geselecteerd moet worden d.m.v. het filter aan de linker kant op figuur 2.1. Voor het filter moet er een selectie worden gemaakt uit organisatie en bedrijfsproces. Nadat er geklikt wordt op een proces dan begint de filtering. Dan worden de gelinkte risico's/ controles/ events etc. geladen. Deze objecten hebben eveneens onderling gelinkte controles, verbeteringsmaatregelen en events. Dit wordt ook allemaal wel weergeven, alleen op een onoverzichtelijke manier. (*zie wederom figuur 2.1).*

Het doel van de pagina is dus om een nette en visueel aantrekkelijke overzicht te krijgen tussen alle losse onderdelen die gekoppeld zijn aan een bedrijfsproces. Als deze module er niet zou zijn, dan zouden gebruikers handmatig elke risico, controle, event etc. af moeten gaan (wel via een filter) om te bekijken bij welk bedrijfsproces zij horen. Ook worden er op de pagina documenten weergeven die gelinkt zijn aan het bedrijfsproces. Hierdoor kunnen medewerkers die vallen onder het proces 'Developers' dus handige huisregels en developer-related documenten zien. Tot slot kunnen gebruikers via de business dimension navigator de 'Process Overview' bekijken. Dit is een visuele weergave van het process, getekend d.m.v. swimlanes. (Dit wordt door hogere medewerkers gemaakt en gedeeld). De B.D.N. is dus een belangrijk knooppunt binnen de CERRIX-applicatie. Het heeft veel potentie, alleen wordt het niet volledig benut.

Formatted: Justified

Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands)

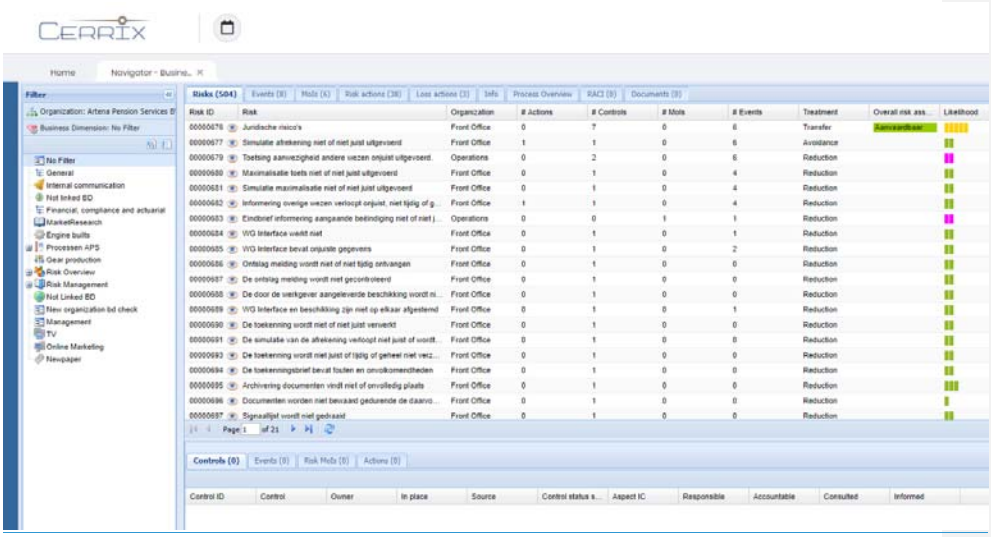
Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands)

Een praktisch voorbeeld van de business dimension navigator:

*Stel je hebt de **organisatie** 'De Haagse Hogeschool', met gekozen **business dimension** 'Studenten' , dan zouden wij hier een lijst met (bestaande) **risico's** zien, bijvoorbeeld: dat het mogelijk is dat een student de kennis niet tot zich kan nemen door afwezigheid. Een **event** zou dan zijn dat de student afwezig is door ziekte, hij heeft weinig fruit gegeten. Een **control** zou dan zijn, dat de student meer fruit zal moeten eten, waardoor hij minder snel ziek wordt. De **measure of improvement** zou dan tot slot zijn dat er wekelijks een fruitmand naar de student bezorgd wordt. Het is de bedoeling dat dit overzicht en deze koppelingen duidelijk weergegeven worden bij het gebruik van de business dimension navigator.*

Een ander voorbeeld van het belang van de business dimension navigator:

Een klant van CERRIX is het bedrijf Stater. Zij hebben grote afnemers, zoals De Nederlandse Bank. Om de communicatie tussen Stater en De Nederlandse Bank goed te kunnen controleren moeten zij hun processen kunnen beheersen. Dit doen ze door gebruik de maken van de business dimension navigator, zodat alles in een oogopslag te zien is, omdat je anders veel processen over het hoofd ziet. Het zorgt voor een goede 'in control' statement, wat De Nederlandse Bank Stater afdwingt. Doordat alle koppelingen met de processen weergeven worden, worden zij beter beheerst, wat zorgt voor een betere controle en terugkoppeling naar De Nederlandse Bank toe.



Figuur 2.1: Huidige Business Dimension Navigator pagina (verouderd en aan omschrijving toe)

In de huidige weergave wordt er een ~~overschot~~overvloed ~~van~~ aan informatie weergeven. Dit is visueel niet aantrekkelijk. Ook is de organisatie en procesfilter (links) gemaakt middels een inklapsysteem, waardoor het niet voor iedereen te begrijpen is en dit visueel onaantrekkelijk is.

§ 2.2 DE ONBRUIKBAARHEID VAN DE BUSINESS DIMENSION NAVIGATOR ~~USER EXPERIENCE~~

De B.D.N. is een belangrijk knooppunt binnen de applicatie, waar alle data in samenkomt, alleen wordt deze potentie niet volledig benut. Dit komt door een aantal oorzaken:

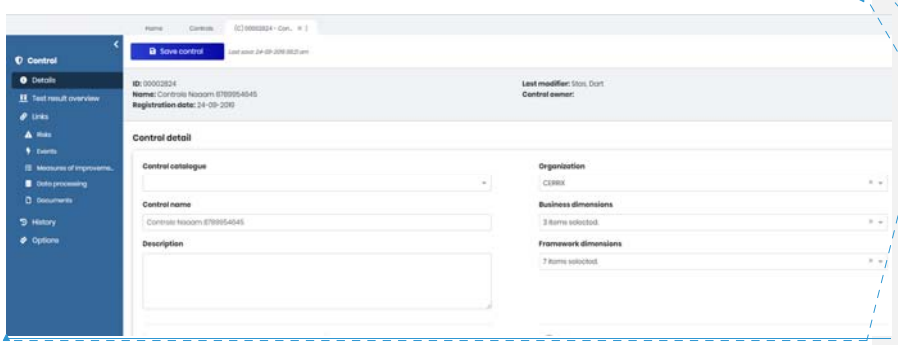
- Als de gebruiker de pagina probeert te laden, kan het tot wel 20 seconden duren voordat er uitvoer op het scherm te zien is en de data geladen is. Nadat de gebruiker klikt op een nieuwe tab (van risico naar controle of event), duurt het wederom +- 20 seconden voordat er nieuwe uitvoer op het scherm is.

- Er wordt te veel data weergeven op het scherm. Het filter aan de linker kant (zie figuur 2.1) is via een inklapsysteem vormgegeven en is onduidelijk. Daarnaast wordt er te veel informatie weergeven over de gekoppelde modules, waardoor de gebruiker overspoeld wordt met informatie.

- De pagina is technisch verouderd. Er wordt gebruik gemaakt van oude schermen die geschreven zijn in Ext.Net. Op dit moment is CERRIX een overstap aan het maken naar Angular 7. Hierdoor voldoet het ontwerp van de pagina niet meer aan de nieuwe huisstijl van CERRIX. (*zie figuur 2.2)*

- De data die momenteel op de pagina weergeven wordt, wordt niet gefilterd op basis van de rollen en rechten van de gebruiker. Hierdoor kunnen gebruikers ook bijvoorbeeld confidentiële risico's zien, wat niet de bedoeling is.

(Voor de volledige uitleg verwijs ik u door naar Bijlage 3: Plan van aanpak)



Figuur 2.2: Voorbeeld van de nieuwe huisstijl van CERRIX (Een control detail pagina) geschreven in Angular, i.p.v. Ext.Net. In vergelijking met figuur 2.1 is dit visueel een groot verschil.

Formatted: Justified

Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands)

Formatted: Font: (Default) +Headings (Calibri Light), 11 pt, Font color: Black, Dutch (Netherlands)

Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands)

Formatted: Justified, Indent: Left: 1,27 cm, No bullets or numbering

Formatted: Font: (Default) +Headings (Calibri Light), Not Italic

Formatted: Justified, Indent: Left: 1,27 cm, No bullets or numbering

Formatted: Justified

Formatted: Font: (Default) +Headings (Calibri Light), Italic

~~Een duidelijke en fijne user expierience is van groot belang.~~

Formatted: Justified

§ 2.3 WAT WAS ER AL GEDAAN OM HET PROBLEEM OP TE LOSSEN?

Formatted: Heading 2, Justified, Indent: Left: 0 cm, Line spacing: single

De afgelopen jaren waren er enkele 'snelle fixes' toegepast op de backend van de B.D.N. De developers van CERRIX hadden enkele methodes gebruikt, om het laden van de data sneller te laten verlopen. Zo had mijn begeleider Martin het laden van de data een stukje herschreven, zodat de data nu **geopende** per tab (risico, controle, event etc.) wordt opgehaald, waardoor de pagina wát sneller was. Voor zijn fix werd alle data van alle modules wel in een keer opgehaald, waardoor er nog langer gewacht moest worden voordat er uitvoer verscheen.

Daarnaast had Martin ook de manier waarop de counts bovenin de tabs werden opgehaald herschreven. (*Bijvoorbeeld:* 'Risks (120)' als titel van de risk tab). Wanneer de tab niet actief was, dan werd er een 'lichtere' manier gebruikt om de counts naast te tabs op te halen. (Hiervoor heeft hij niet alle data opgehaald, maar alleen het aantal aan de database gevraagd). Dit zorgde ook voor wat versnelling.

§ 2.4 WAT GEBEURT ER ALS HET PROBLEEM NIET WORDT OPGELOST?

Formatted: Heading 2, Justified, Indent: Left: 0 cm, Line spacing: single

Niet heel veel klanten gebruiken de business dimension navigator op dit moment, dit komt omdat de pagina niet soepel meewerkt. Echter is dit probleem ontstaan door opgestapelde technical debt. Er werden door de jaren heen alleen snelle fixes toegepast, waardoor de pagina nooit in zijn geheel is verschoond. Daar is mijn afstudeeropdracht voor. Als de pagina niet verbeterd wordt, dan zullen klanten potentiele interesse in de CERRIX-applicatie verliezen. Dit komt doordat niemand een pagina in gebruik wilt nemen waarop je +/- 20 seconden op moet wachten per aanvraag naar de server.

§ 2.5 DE OPDRACHT

Formatted: Justified

De opdracht die gesteld is vanuit CERRIX, is om de B.D.N. te herontwerpen en te versnellen. Aan de achterkant is de code erg verouderd. De manier waarop het ophalen van de data gedaan wordt komt niet overeen met de rest van de methodes vanuit de applicatie. Het merendeel van de applicatie gebruikt op dit moment parallelle functies voor het ophalen van data, wat betekent dat de functies tegelijkertijd worden uitgevoerd, wat sneller is bij grote taken (zoals het per module ophalen van data van de navigator), in plaats van op de oudere manier, waardoor grote en zware functies sequentieel worden uitgevoerd en dus moeten wachten op elkaar.

De business dimension navigator loopt achter op deze implementatie en is hierdoor erg traag. Dit komt doordat er ook op dit moment hele SQL query's uitgevoerd worden om data op te halen, waardoor er **volledige** tabellen worden opgehaald, terwijl het hedendaags veel sneller en simpeler kan. Deze implementatie moet vernieuwd en versneld worden, zodat de pagina weer 'bruikbaar' zal worden.

Deze data dat weergeven wordt moet overigens nog gefilterd worden op basis van de rollen en rechten van de gebruiker die de pagina bekijkt. Op dit moment kan een gebruiker alle risico's, events en controles etc. inzien die er zijn gemaakt. Er zal op gelet moeten worden dat de gebruiker alleen de data mag zien, die hoort bij zijn combinatie van rollen en rechten.

Tot slot zal de visuele weergave van de pagina verbeterd moeten worden. Op dit moment is de pagina geschreven in Ext.Net. Dit is een oud JavaScript framework, en dit zal vernieuwd moeten worden met de door de CERRIX-applicatie heen getrokken nieuwere versie, Angular 7. Hierdoor zal de pagina weer aan de huisstijl van CERRIX voldoen. (Dezelfde applicatie brede Angular-componenten/ tabellen zullen worden toegepast en hierdoor is het consistenter, en sneller). Ik zal dit ontwerp samen met de consultants gaan maken. Hiervoor zullen de nodige analyses gemaakt moeten worden om o.a. de juiste Angular-componenten te vinden, te ontwerpen en te hergebruiken.

HOOFDSTUK 3: ORGANISATIE

In dit hoofdstuk zal de organisatie waar de afstudeeropdracht vervuld is beschreven worden. Er zal uitgelegd worden wat voor organisatie CERRIX is, wat voor product zij haar klanten aanbied, met welke collega's ik direct in contact sta, welke computertalen en programma's er gebruikt worden, hoe de kwaliteitsbewaking van de code wordt gehandhaafd en tot slot welke ontwikkelmethode er gebruikt wordt binnen het team van CERRIX.

§ 3.1 VERWIJZEN NAAR WOORDENLIJST CERRIX

CERRIX B.V. is gevestigd aan de Koninginnegracht 29, 2524 AB te Den Haag en is een software and consultancy bedrijf dat zich specialiseert op het gebied van risico-en procesmanagement. Het doel van CERRIX is om organisaties hun risicobeheersing en processen beter te laten verlopen. Dit vakgebied is momenteel nog volop in ontwikkeling, omdat de meeste bedrijven dit onderhouden middels statische wijzen, zoals losse digitale Excel-sheets of het zelfs los op papier schrijven. CERRIX biedt deze organisaties een transitie van deze oude wijzen naar een dynamische webapplicatie, waarin dit alles netjes en zorgvuldig via een goed lopende userinterface bijgehouden kan worden. Aangezien elke organisatie een solide en kloppende risicomanagement nodig heeft, zijn er genoeg potentiële klanten voor CERRIX en zal deze innovatie ook nooit stilstaan.

§ 3.2 GRC-TOOL

CERRIX biedt haar klanten een '**GRC**-TOOL' aan middels een webapplicatie. Hierin kunnen klanten hun **Governance,** wat staat voor het overzien van de bedrijfsprocessen, **Riskmanagement** om de potentiële risico's te beheersen en managen en **Compliance**, waardoor organisaties hun wettelijke werkwijze (zoals de implementatie van de AVG) in bij kunnen houden. Zo'n tool is zeer gewild, omdat elke bedrijf o.a. wettelijk verplicht is om hun compliance bij te houden, wat perfect in de applicatie gedaan kan worden. De applicatie biedt gebruikers daarnaast rollen en rechten aan, waardoor gebruikers (medewerkers van een organisatie) restricties opgelegd kunnen krijgen, omdat niet elke werknemer overal bij kan.

De applicatie wordt telkens uitgebreid met meerdere features, nu kunnen er bijvoorbeeld forms aangemaakt worden (een soort Google-forms implementatie), waardoor klanten zo organisatie breed hun medewerkers enquêtes, vraagstukken en zelfs uitnodigingen kunnen sturen. Zo is de applicatie dus elke dag nog volop in ontwikkeling.

Tot slot biedt CERRIX haar klanten ook consultancy aan op het gebied van riskmanagement en compliance. Als bedrijven ondersteuning nodig hebben qua risicobeheersing en over de wetgeving, staan de consultants, met expertise op het gebied van risk and projectmanagement voor hun klaar. Natuurlijk staan zij ook klaar voor klanten die vragen hebben over de werking van de applicatie.

Formatted: Heading 1, Justified

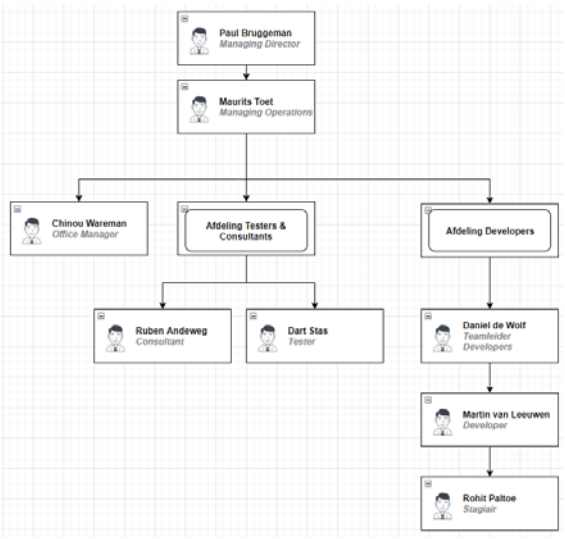
Formatted: Font: (Default) +Headings (Calibri Light), Highlight

§ 3.3 KLANTSEGMENT

De huidige klanten van CERRIX zijn o.a. Heineken, de overheid, ASR-bank, ABN-AMRO, Deloitte en bepaalde pensioenfondsen. Zij hebben voor CERRIX gekozen omdat hun wensen betreft riskmanagement, procesmanagement en compliance het beste passen bij de software en ondersteuning die CERRIX biedt. Naast de Nederlandse markt is CERRIX bezig met een uitbreiding in Curaçao en in Frankrijk, om optimaal van de markt te kunnen genieten. Hieruit blijkt dat CERRIX de focus legt op de financiële sector.

§ 3.4 ORGANIGRAM

CERRIX heeft een klein team van ongeveer 15 man. Hieronder zal een organigram volgen *(zie figuur 3.1)* met de collega's waarmee er direct contact is.



Figuur 3.1 (Gedeelte) Organigram van CERRIX

Het bedrijf is bestaat uit Paul, de managing director, Maurits, de operationele manager, Chinou, de office manager, de testers & consultants en de software developers. Ik zal direct in contact staan met de developers (Daniël, Martin), Paul, Maurits en de testers/ consultants (Dart en Ruben). Mijn bedrijfsmentor is Martin, hij is de voormalige teamleider van de developers en heeft dus genoeg kennis over zowel code als de applicatie mocht ik enige vragen hebben. Naast de developers die weergeven worden in figuur 3.1 zijn er nog tien andere ontwikkelaars die voor ondersteuning kunnen bieden, mochten Martin en Daniël er niet zijn.

DE HAAGSE HOGESCHOOL

CERRIX

INHOUDSOPGAVE

Formatted: Justified, Indent: Left: 0 cm, Right: 0 cm

Formatted: Heading 1, Justified, Space After: 0 pt, Line spacing: single

Formatted: Heading 1, Justified, Line spacing: single

Formatted: Heading 1, Justified, Indent: Left: 0 cm, Space After: 0 pt, Line spacing: single

Formatted: Heading 1, Justified, Line spacing: single

Formatted: Justified, Indent: Left: 0 cm, Right: 0 cm

EXTRA FEATURE: HET REVIEWEN VAN EEN PROCES

RUBEN GEVRAAGD HOE BELANGRIJK PROCESS SCORING IS:

INCONTROL STATEMENT: DAT ZEG EIGENLIJK 'IK BEHEERS DE PROCESSEN DIE IK UITVOER.' HET IS VOORNAMELIJK VAN BELANG RICHTING DE COMMUNICATIE TUSSEN DE KLANTEN DIE JE HEBT, DE GROTE AFNEMERS. RICHTING DNB(DE NEDERLANDSE BANK). BIJV STATER STAAT INCONTROL DOOR DE DNB. VAN HUN KRIJGEN ZIJ DUS BIJV EXTRA CONTROLES. OF AFM, (AUTORITEIT FINANCIELE MARKTEN) OF ISAE PROCESSEN, DOOR CERRIX ZELF. JE ZIET ZO DUS VEEL PROCESSEN OVER HET HOOFD.

''EEN NAVIGATOR HOORT EEN OVERZICHT TE GEVEN''

INRICHGEN MET 2 GROTE DINGEN IN MIND!!

CARLO: RELEVANTE DATA OP 1 PAGINA NU IPV ALLES LOS EN VIA DE WORKSPACES VINDEN IS OOK EEN UX DINGETJE. WNAT KLANTEN WORDEN VOLGHEGOOID MET INFORMATIE, EN DE ONDERHOUDBAARHEID WANT RISK ACTIONS ZIJN NU UITGEFASEERD, DUS DAARDOOR WAS HET SLECHTER ONDERHOUDBAARDER GEWORDEN. WANT NIEUWERE DINGEN WERDEN LOS GEKOPPELD. NU DOOR MIJN CODE ZAL HETMAKKELIJKE RZIJN OM TOE TE VOEGEN. IS OOK BETER EN PERFORMANCE.

HET WAS BEDOELD OM OVERZICHT TEK RIJGEN MAAR JE KREEG ER GEEN OVERZICHT DOOR. NU WORDT ALLES IN 1 OOGOPDSLAG BETER

Formatted: Font: (Default) +Headings (Calibri Light), 28 pt, Bold, Highlight

Formatted: Font: (Default) +Headings (Calibri Light), 28 pt, Bold, Highlight

Formatted: Heading 1, Justified

Formatted: Font: (Default) +Headings (Calibri Light), 28 pt, Bold, Dutch (Netherlands), Highlight

Formatted: Font: (Default) +Headings (Calibri Light), 28 pt, Bold

Formatted: Font: (Default) +Headings (Calibri Light), 28 pt, Not Bold, Not Highlight

Formatted: Heading 1, Justified, Space After: 0 pt, Line spacing: single

Formatted: Heading 1, Justified, Indent: Left: 0 cm, First line: 0 cm

Formatted: Heading 1, Justified, Space After: 0 pt, Line spacing: single

Formatted: Font: (Default) +Headings (Calibri Light), 28 pt, Not Bold

TEGENWOORDIG LEZEN WE HET OVERAL: "WIJ HEBBEN ONZE PRIVACY VOORWAARDEN GEÜPDATET, GA HIER EERST MEE AKKOORD OM VERDER TE GAAN". WAAROM WORDEN WIJ HIER NOU MEE VOLGEGOOID? DAT KOMT DOOR DE NIEUWE AVG WET. TIJDENS MIJN STAGE STOND DIT ONDERWERP CENTRAAL. HOE KONDEN WIJ DE AVG IMPLEMENTEREN, ZODAT ORGANISATIES HUN NIEUWE PRIVACY VOORWAARDES OP ORDE HEBBEN?

AVG

SINDS 25 MEI 2018 IS DE WET ALGEMENE VERORDENING PERSOONSGEGEVENS IN DE HELE EUROPESE UNIE VAN TOEPASSING VOOR ALLE ORGANISATIES DIE PERSOONSGEGEVENS VERWERKEN. DIT IS DE NIEUWE PRIVACYWETGEVING DIE DE NEDERLANDSE WET BESCHERMING PERSOONSGEGEVENS VERVANGT. DE AVG IS OOK WEL BEKEND ONDER DE ENGELSE NAAM: GENERAL DATA PROTECTION REGULATION (GDPR).

WAT VERANDERT ER DOOR DE AVG

DE AVG ZORGT ONDER MEER VOOR DE VERSTERKING EN UITBREIDING VAN DE PRIVACY RECHTEN. ZO ZIJN ER AANTAL AANVULLENDE RECHTEN BIJGEKOMEN: DE RECHT OP VERGETELHEID, WAARDOOR MENSEN NU KUNNEN EISEN DAT EEN ORGANISATIE HUN PERSOONSGEGEVENS VERWIJDERT EN DEZE VERWIJDERING OOK DOORGEEFT AAN ALLE ANDERE ORGANISATIES DIE DEZE GEGEVENS VAN DEZE ORGANISATIE HEBBEN GEKREGEN. OOK DE RECHT OP DATAPORTABILITEIT IS NIEUW. NU HEBBEN MENSEN (ONDER BEPAALDE VOORWAARDEN) HET RECHT OM VAN EEN ORGANISATIE HUN PERSOONSGEGEVENS IN EEN STANDAARDFORMAAT TE ONTVANGEN (ZO KUNNEN WIJ NU OOK HEEL GEMAKKELIJK AL ONZE GEGEVENS DIE FACEBOOK EN/OF GOOGLE VAN ONS HEEFT DOWNLOADEN). HIERDOOR KUNNEN ZIJ HUN GEGEVENS MAKKELIJK DOORGEVEN AAN EEN ANDERE LEVERANCIER VAN DEZELFDE SOORT DIENST.

WAT DIT BETEKENT VOOR ORGANISATIES

DOOR DE INVOERING VAN DE AVG MOETEN ALLE ORGANISATIES HUN PRIVACYWETGING AANPASSEN. ZIJ HEBBEN NU MEER VERPLICHTINGEN BIJ HET VERWERKEN VAN PERSOONSGEGEVENS. DE AVG LEGT NAMELIJK MEER NADRUK OP DE VERANTWOORDELIJKHEID DIE DE ORGANISATIES HEBBEN OM AAN TE TONEN DAT ZIJ ZICH AAN DEZE NIEUWE WET HOUDEN.

DE VERANTWOORDINGSPLICHT HOUDT IN DAT EEN ORGANISATIE MET DOCUMENTEN MOET KUNNEN AANTONEN DAT DE JUISTE ORGANISATORISCHE EN TECHNISCHE MAATREGELEN GENOMEN ZIJN OM AAN DE AVG TE VOLDOEN. DE AVG BIEDT ORGANISATIES TEGELIJKERTIJD MEETINSTRUMENTEN DIE HUN HELPEN OM DE WET NA TE

Formatted: Heading 1, Justified, Indent: Left: 0 cm, Right: 0 cm

Formatted: Heading 1, Justified, Space After: 0 pt, Line spacing: single

Formatted: Heading 1, Justified, Indent: Left: 0 cm, Line spacing: single

Formatted: Heading 1, Justified, Indent: Left: 0 cm, Right: 0 cm, Space After: 0 pt

Formatted: Font: (Default) +Headings (Calibri Light), 28 pt, Font color: Background 1

Formatted: Heading 1, Justified, Indent: Left: 0 cm

Formatted: Heading 1, Justified, Indent: Left: 0 cm, Right: 0 cm

Formatted: Font: (Default) +Headings (Calibri Light), 28 pt, Font color: Background 1

Formatted: Heading 1, Justified, Indent: Left: 0 cm

Formatted: Heading 1, Justified, Indent: Left: 0 cm, Right: 0 cm

LEVEN. BIJVOORBEELD MODELBEPALINGEN VOOR DOORGIFTE VAN PERSOONSGEGEVENS (DIT BLIJKT UIT SANCTIES BINNEN DE AVG. (Z.D.))

- WAT LEVERT DE AVG ORGANISATIES OP

DOOR DE KOMST VAN AVG GELDT ER NOG MAAR ÉÉN PRIVACY WET IN DE HELE EUROPESE UNIE IN PLAATS VAN 28 VERSCHILLENDE NATIONALE WETTEN. INTERNATIONALE ORGANISATIES HOEVEN NU ALLEEN NOG MAAR TE LETTEN OP DE AVG. ALS EEN ORGANISATIE IN MEERDERE EU LIDSTATEN ACTIEF IS, DAN LEVERT DE AVG HET VOLGENDE OP:

- ORGANISATIES HEBBEN MINDER ADMINISTRATIEVE KOSTEN EN NALEVINGSKOSTEN

- ORGANISATIES HEBBEN MEER RECHTSZEKERHEID

- ER IS EEN GELIJK SPEELVELD, OMDAT ALLE REGELS HETZELFDE ZIJN VOOR ALLE BEDRIJVEN IN DE EU

- ORGANISATIES HOEVEN MAAR MET ÉÉN TOEZICHTHOUDER ZAKEN TE DOEN DIT BLIJKT WEDEROM UIT SANCTIES BINNEN DE AVG. (Z.D.).

WAT WAS ER AL GEDAAN OM HET PROBLEEM OP TE LOSSEN?

VOORDAT IK KWAM WAS ER AL EEN CONCEPT GEMAAKT VAN EEN VERWERKINGSDOEL SYSTEEM, WAARIN VASTGESTELD KON WORDEN WAT VOOR INFORMATIE VOOR HOELANG GEBRUIKT WERD, MET WELK DOEL EN WAREN ALLE DETAILS DIE ERBIJ HOORDEN AL GROTENDEELS VASTGELEGD. DIT WAS EEN BESTAAND DOCUMENT EN OOK TE ZIEN ALS CONCEPT IN DE APPLICATIE IN DE VORM VAN INVOERSCHERMEN, MAAR DIT WAS NOG NIET VOLMAAKT.

ALLE TABELLEN/ SOORTEN INFORMATIE DIE DE AVG WET VASTSTELDE WAREN NOG NIET GEÏMPLEMENTEERD IN DE SOFTWARE. DE APPLICATIE HAD AL EEN PAGINA VOOR HET KOPPELEN VAN DEZE VERWERKINGSDOELEN NAAR EEN 'ICT GERELATEERD' SYSTEEM, WAT OOK EEN TABEL IN DE DATABASE WAS MAAR MISTE HET DE ONDERLIGGENDE RELATIES TUSSEN DE ANDERE AVG KOPPELINGEN EN DE OVERIGE WEBPAGINA'S DIE VERBONDEN WAREN AAN EEN VERWERKINGSDOEL.

WAT HIER DUS NOG AAN MISTE WAREN DE STANDING DATA PAGINA'S. DAT WAREN DE PAGINA'S, WAARIN DEZE GEGEVENS INGEVOERD IN KONDEN WORDEN EN MOESTEN

Formatted: Font: (Default) +Headings (Calibri Light), 28 pt, Font color: Background 1

Formatted: Heading 1, Justified, Indent: Left: 0 cm

Formatted: Heading 1, Justified, Indent: Left: 0 cm, Right: 0 cm

Formatted: Heading 1, Justified, Right: 0 cm, No bullets or numbering

Formatted: Font: (Default) +Headings (Calibri Light), 28 pt, Dutch (Netherlands)

Formatted: Heading 1, Justified, Right: 0 cm, Space After: 0 pt, No bullets or numbering

Formatted: Font: (Default) +Headings (Calibri Light), 28 pt, Font color: Background 1, Dutch (Netherlands)

Formatted: Font: (Default) +Headings (Calibri Light), 28 pt, Dutch (Netherlands)

Formatted: Heading 1, Justified, Indent: Left: 0 cm, Line spacing: single

Formatted: Heading 1, Justified, Indent: Left: 0 cm, Right: 0 cm

WORDEN WEERGEVEN, ZODAT ALLES UITEINDELIJK EEN RELATIE HAD MET EEN VERWERKINGSDOEL, WAARDOOR DE AVG CYCLUS COMPLEET WAS.

DEZE VERWERKINGSDOEL MODULE IS EEN EIS DIE DE AVG WETTELIJK STELT. ELK VERWERKINGSDOEL VAN EEN ORGANISATIE MOET OPGESLAGEN EN TRACEERBAAR ZIJN. BINNEN DE AVG HEET DIT HET 'VERWERKINGSREGISTER'.

WAT GEBEURT ER ALS HET PROBLEEM NIET WORDT OPGELOST?

ALS DEZE MODULE ER NIET KOMT DAN ZULLEN ORGANISATIES EEN HOGE GELDBOETE RISKEREN MET EEN MAXIMALE HOOGTE VAN TWINTIG MILJOEN EURO OF 4 PROCENT VAN DE TOTALE WERELDWIJDE JAAROMZET EN MOET ER EEN CORRIGERENDE MAATREGEL PLAATSVINDEN OM DE AVG ALS NOG CORRECT TE KUNNEN IMPLEMENTEREN.

HET IS OOK MOGELIJK OM EEN LAST ONDER BESTUURSDWANG OPGELEGD TE KRIJGEN, WAT EEN AANVULLING OP HET SANCTIESTELSEL VORMT VAN DE AVG EN BIEDT DE AUTORITEIT PERSOONSGEGEVENS EEN LAAGDREMPELIGE MANIER OM HANDHAVING TE BEWERKSTELLINGEN ZONDER GEBRUIK TE HOEVEN MAKEN VAN EEN GELDBOETE. IMMERS, WANNEER IEMAND DOET WAT IS GEVRAAGD, VOLGT ER GEEN SANCTIE (NAAR "SANCTIES BINNEN DE AVG", Z.D.).

DE OPDRACHT

DE GLOBALE OPDRACHT, WAAR IK DEEL AAN HEB GENOMEN WAS 'DE IMPLEMENTATIE VAN AVGREGISTER BIJ CERRIX B.V.' OM DE OPDRACHT TE KUNNEN REALISEREN HEB IK GEWERKT AAN EEN ZESTAL OPDRACHTEN, DIE ALLEMAAL BETREKKING HADDEN TOT DE IMPLEMENTATIE VAN DE NIEUWE WETGEVING AVG IN DE CERRIX APPLICATIE. HET INHOUDELIJKE DOEL VAN DEZE STAGE IS DAN OOK OM DEZE IMPLEMENTATIE ZO COMPLEET MOGELIJK TE MAKEN EN AF TE RONDEN, ZODAT ORGANISATIES DEZE MODULE KUNNEN GEBRUIKEN OM HUN EIGEN AVG ZAKEN OP ORDE TE HOUDEN.

ZO HEB IK PAGINA'S GEMAAKT, WAAROP KLANTEN VAN CERRIX OP AAN KUNNEN GEVEN WAT VOOR DATA ZIJ OPSLAAN, HOE DIT AANGELEVERD WORDT, WAT HUN POLICY IS VOOR HET VERWIJDEREN VAN DEZE GEGEVENS EN HOELANG ZIJ DIT BEWAREN. DAARNAAST HEB DE MODULE GEPROBEERD TE TESTEN EN HEB IK MIJN COLLEGA'S GEHOLPEN MET HET OVERIGE WERK WAT ER NOG GEDAAN MOEST WORDEN, ZOALS HET OPLOSSEN VAN BUGS VOOR IN DE NIEUWE RELEASE VAN DE APPLICATIE.

DOOR HET LOPEN VAN DEZE STAGE HEB IK EEN OVERZICHT GEKREGEN VAN WAT ER ALLEMAAL SPEELT IN HET BEDRIJFSLEVEN EN HOE EEN NORMALE DAG IN EEN ORGANISATIE ERUIT ZIET ALS PROGRAMMEUR. ER ZIJN TESTERS EN PROGRAMMEURS EN ZIJ WERKEN NAUW SAMEN OM DE REQUIREMENTS TE ACHTERHALEN EN OOK OM BUGS IN DE SOFTWARE TE VINDEN. VAN DAGELIJKSE SCRUM STAND-UPS TOT WEKELIJKSE

Formatted: Heading 1, Justified, Space After: 0 pt, Line spacing: single

Formatted: Heading 1, Justified, Indent: Left: 0 cm, Line spacing: single

Formatted: Heading 1, Justified, Indent: Left: 0 cm, Right: 0 cm

Formatted: Font: (Default) +Headings (Calibri Light), 28 pt

Formatted: Heading 1, Justified, Space After: 0 pt, Line spacing: single

Formatted: Heading 1, Justified, Indent: Left: 0 cm

Formatted: Heading 1, Justified, Indent: Left: 0 cm, Right: 0 cm

~~VOORTGANGSMEETINGS, IK HEB HET ALLEMAAL MEEGEMAAKT EN HEB NU EEN CONCRETER BEELD OVER WAT IK KAN VERWACHTEN IN DE NABIJE TOEKOMST.~~

Formatted: Heading 1, Justified, Line spacing: single

HOOFDSTUK 4 AANPAK

In dit hoofdstuk zal de aanpak van het project beschreven worden. Allereerst zal er een risicoanalyse volgen en op basis van deze analyse zal er een ontwikkelmethode afgewogen en geselecteerd worden. Er zal uiteindelijk gekozen worden voor het gebruik van SCRUM als ontwikkelmethode. Daarnaast zullen de SCRUM-activiteiten die doorlopen zullen worden beschreven worden en zal er tot slot een sprint planning volgen om weer te geven hoe de projectindeling in elkaar zal zitten.

~~Projectaanpak / kanban vs scrum/ mijn /pva~~

~~want risk actions zijn nu uitgefaseerd, dus daardoor was het slechter onderhoudbaarder geworden. Want nieuwere dingen werden los gekoppeld. Nu door mijn code zal hetmakkelijke rzijn om toe te voegen. is ook beter en performance.~~

§ 4.1 RISICOANALYSE

Om te bepalen welke ontwikkelmethode er gebruikt zal gaan worden, zal er eerst een risicoanalyse uitgevoerd moeten worden. Deze risicoanalyse zal een beter beeld schetsen over welke risico's er op zouden kunnen spelen tijdens het afstuderen. Mochten deze risico's voorkomen, dan zouden er problemen en verhindering tijdens het afstudeertraject worden ervaren en dus is het handig dat de belangrijkste risico's in kaart worden gebracht, zodat de ontwikkelmethode hierop aan kan sluiten en deze risico's in kan perken. De risicoanalyse wordt weergeven op tabel 4.1

Tabel 4.1 risicoanalyse voor het project

Bedreiging	Kans (1-5)	Impact (1-5)	Risico score (K*I)	Tegenmaatregel	Risico eigenaar
Technisch (middelen)					
TR.1: Niet beschikken over de juiste Visual Studio licenties.	1	5	5	Een gratis versie proberen te vinden (community editie) of anders vragen of ik een licentie via CERRIX kan verkrijgen.	Rohit

Formatted: Font: (Default) +Headings (Calibri Light), Font color: Background 1, Not Highlight
Formatted: Heading 1, Justified
Formatted: Font: (Default) +Headings (Calibri Light), Font color: Background 1, Not Highlight
Formatted: Font: (Default) +Headings (Calibri Light), Not Highlight
Formatted: Normal, Justified, Indent: Left: 0 cm, Right: 0 cm
Formatted: Justified, Space After: 10 pt, Line spacing: Multiple 1,15 li
Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands)
Formatted: Font: (Default) +Headings (Calibri Light), Highlight
Formatted: Justified
Formatted: Font: (Default) +Headings (Calibri Light), Bold
Formatted: Justified
Formatted: Font: (Default) +Headings (Calibri Light), Italic
Formatted: Justified
Formatted: Justified
Formatted: Justified

Organisatorisch (mensen en procedures)					
O.R.1: De communicatie gaat fout: er worden geen/ onjuiste afspraken gemaakt over de contactmomenten	3	4	12	Tijdig afstemmen met het ontwikkelteam/ de verantwoordelijken wanneer de contactmomenten zullen zijn voor het feedback.	Rohit
O.R.2: Ondeskundigheid: ik bezit niet over de juiste kennis voor de technische implementatie.	2	5	6	Bij ondeskundigheid, mijn collega's vragen om raad. (Bijvoorbeeld over vraagstukken over de Angular taal). Als mijn collega's het ook niet weten, zal ik het opzoeken op het internet.	Rohit
Functioneel (wat moet er klaar zijn als het af is)					
F.R.1: De omvang van het project is te groot en het komt niet op tijd af	4	5	20	Tijdig de scope bepalen van de opdracht: niet te veel proberen te doen, maar focussen op de basis en daarna pas op eventuele extra features.	Rohit
F.R.2: De opdracht is niet goed gedefinieerd. Er worden andere uitwerkingen/ implementaties verwacht dan wat er gemaakt zal worden	4	4	16	Veel contactmomenten hebben om de opdracht te bespreken, zodat iedereen op 1 lijn zal zitten.	Rohit
F.R.3: De scope van het project verandert: de baas of de verantwoordelijken willen later iets anders.	4	4	16	Wederom veel contactmomenten hebben om de opdracht te laten zien, feedback te verwerken zodat de scope niet (of zelfs niet al te veel) af zal wijken van het origineel.	Rohit

Formatted: Justified
Formatted: Justified
Formatted: Justified
Formatted: Justified
Formatted: Justified
Formatted: Justified
Formatted: Justified
Formatted: Justified

Deze risico's waren uitgekozen, omdat er onder andere nog geen beschikking was over een Visual Studio licentie en het bedrijf hiervoor zal moeten regelen. Daarnaast is er nog geen idee over hoe de communicatie nou daadwerkelijk in het bedrijf verloopt, hoe de collega's reageren etc. Dit zou pas in de eerste weken duidelijk worden. Ook is er nog geen kennis van Angular. Dit zou pas worden aangeleerd tijdens de uitvoering van het project en zou het ook meer dan mogelijk zijn dat er hier niet genoeg technische ondersteuning voor zou zijn. Tot slot kon de scope of zelfs het project veranderen, omdat het ontwerp nog from scratch gemaakt moest worden en er nog niet echt een idee was hoe dit eruit zou moeten gaan zien. Aangezien dit ontwerp er nog niet was kon er nog van alles boven water komen m.b.t. veranderende requirements en/ of features.

Uit deze risicoanalyse blijkt dat er een 4 van de 6 risico's zijn met een hoge risico score (10+) (**kans*impact**). Dit kwam doordat deze risico's het meeste over de projectcommunicatie en over de requirements/ eisen van het eindproduct gingen. Omdat de business dimension navigator nog herontworpen moest worden en hier veel feedback overheen zou gaan, zodat dit goed zou verlopen, scoorden deze risico's hoog. Om deze redenen zal er rekening gehouden moeten worden met de (hoge) waarschijnlijkheid dat deze risico's konden plaatsvinden. Natuurlijk zou dit tijdig opgevangen moeten worden.

MEETING MET RUBEN VOOR HET PRIORITESEREN(R.1); (R.2); (R.3); (R.4); E, ZOALS DE FRAMEWORK DIMENSION NAVIGATOR.

UIT DEZE REQUIREMENTS KOMEN HETVOLGENDE VOORT:

OP TE LEVEREN PRODUCTEN? § 4.2 ONTWIKKELMETHODE

Het project zal worden uitgevoerd over een periode van 17 weken. Tijdens deze weken kan het, gezien de hoge risicoscores, meer dan voorkomen dat de bovengenoemde risico's kunnen opspelen. Als deze risico's zouden voorkomen, dan zou ik extra veel documentatie moeten schrijven, of de bestaande documentatie moeten aanpassen, omdat de opdracht dan zou veranderen. Dit zou het afstuderen in de weg zitten, omdat het opleveren van deze extra documenten immers veel tijd zal kosten en de soepele doorstroming van het afstudeertraject zou belemmeren.

Hierom lijkt het mij het verstandigst om een ontwikkelmethode te kiezen die mij kan helpen met het beperken van wisselende documentatie, of die mij kan helpen met het vroegtijdig afvangen van dit soort wisselende requirements. Als risico #FR.1 tot en met #F.R.3 zouden voorkomen, zal de opdracht helemaal opnieuw gedefinieerd moeten worden, zouden er extra vergaderingen ingepland moeten worden, en zouden er extra ontwerpen gemaakt moeten worden wat allemaal veel tijd zou kosten.

§ 4.3 SCRUM

De ontwikkelmethode die deze wensen ondersteunt is SCRUM. SCRUM zal ervoor zorgen dat er agile te werk gegaan kan worden, omdat uit de risico's blijkt dat de opdracht heel makkelijk kon veranderen, omdat er nog een nieuw ontwerp gemaakt moest worden van de business dimension navigator en de wensen hiervoor nog besproken moesten worden. Het ontwerp en de opdracht stond dus als het ware nog niet echt 'vast', waardoor SCRUM hier perfect op aansloot.

Mocht het nou zijn dat er wat aspecten zouden veranderen betreft de requirements, dan kon er doordat SCRUM korte cycli bevat er telkens gemakkelijk feedback verkregen worden, waardoor deze wisselingen tijdig kon worden opgevangen. Hierdoor konden eventuele wijzigingen betreft de requirements eenvoudig worden meegenomen naar de volgende sprint om geïmplementeerd te worden. Dit is het voordeel van SCRUM.

Als de waterval-methode gebruikt zou worden, zou dit zich niet in mijn voordeel uitpakken. Dit komt doordat alle stappen van het ontwerp toe, van de implementatie tot het testen eerst uitgevoerd zouden moeten worden en er pas tegen het einde feedback verkregen kon worden. Als dit pas op een veel later stadia zou gebeuren, door wisselende meningen, omdat het ontwerp nog niet vast stond, zou dit het afstuderen belemmeren door het tekort aan tijd.

Formatted: Font: (Default) +Headings (Calibri Light), 11 pt, Font color: Black, Highlight

Formatted: Font: (Default) +Headings (Calibri Light), Highlight

Formatted: Font: (Default) +Headings (Calibri Light), Highlight

Formatted: Justified

Formatted: Font: (Default) +Headings (Calibri Light), Font color: Accent 6

Formatted: Font: (Default) +Headings (Calibri Light), Font color: Auto

Formatted: Font: (Default) +Headings (Calibri Light), Font color: Accent 6

Formatted: Font: (Default) +Headings (Calibri Light), Font color: Accent 6

Formatted: Heading 2, Justified

§ 4.4 HET GEBRUIK VAN SPRINTS

SCRUM zal doorlopen worden in sprints omdat het een grote opdracht was. Er moest immers veel gedaan worden, van een nieuw ontwerp tot de implementatie. Binnen de sprints kon het werk opgesplitst worden, zodat er telkens kleine increments geleverd werd. Na elke increment, aan het einde van de sprint bood SCRUM de mogelijkheid om feedback te verkrijgen over het product en proces, waardoor de belanghebbenden het increment vroegtijdig zien en hun feedback hierover kunnen vellen.

SCRUM bood voldoende feedbackmomenten aan (reviews en retrospectives), waarin het product en het proces van het increment werd besproken. Middels deze momenten werd er aan de stakeholders getoond wat er in de sprint bereikt werd. Ook hadden zij ruimte om feedback voor eventuele aanpassingen en/ of wisselende wensen door te voeren, waardoor dit gemakkelijk meegenomen kon worden naar de volgende sprint. Hierdoor werden de risico's m.b.t. de verandering van de opdracht eenvoudig in bedwang gehouden, omdat zij aangepakt en ingeperkt werden.

~~samenJira.~~

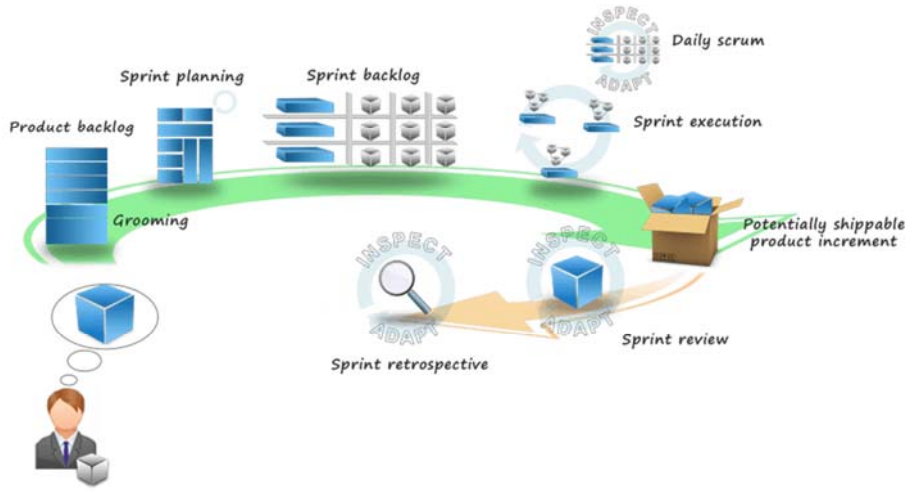
§ 4.5 SCRUM ACTIVITEITEN EN ARTEFACTEN

SCRUM bood mij de gelegenheid om enkele activiteiten uit te voeren en artefacten op te leveren. Deze activiteiten en artefacten ondersteunden het SCRUM proces, door o.a. concrete afspraken op te stellen voor de kwaliteit van de requirements en op te leveren producten.

Tijdens het afstuderen werd er ook gebruik gemaakt van deze activiteiten en artefacten, omdat zij het SCRUM proces eenvoudiger lieten verliepen. Hieronder zal er beschreven worden welke dat waren en waarom de keuze hierop gevallen was. (zie figuur 4.2 en tabel 4.2.1).

Formatted: Heading 2, Justified

Formatted: Justified



Figuur 4.2: Het algemeen model voor sprint activiteiten en op te leveren artefacten. Deze activiteiten en artefacten zal ik ook gebruiken, omdat zij prima aansluiten binnen het afstudeertraject

Naar: SCRUM Activiteiten. (2016). [Foto]

Tabel 4.2.1: Gekozen activiteiten en artefacten tijdens het ontwikkelen

Artefacten:

- Product backlog: In het product backlog waren de requirements opgesplitst in kleine en makkelijk uitvoerbare taken, een tijdsinschatting in uren en een prioriteit volgens de MoSCoW-methode.
- Definition of Ready: Dit was handig, omdat er hierin voorwaardes stonden waaraan de user stories moesten voldoen. Wanneer was een taak zo duidelijk geformuleerd dat het opgenomen kon worden in de product backlog.
- Definition of Done: In de DoD stonden er voorwaardes waaraan het uitgevoerde werk (increment) moest voldoen. Hierdoor werd er gelet op de kwaliteit van de opgeleverde code.
- Sprint planning: Dit was een tabel waarin te zien was welke product backlog items er toegewezen werden per sprint. Het was handig om hier gebruik van te maken, omdat dit een duidelijke overzicht bood van de uit te voeren taken per sprint.

Formatted: Caption, Justified

Formatted: Justified

Formatted: Font: (Default) +Headings (Calibri Light), Bold

Formatted: Justified

Formatted: Font: (Default) +Headings (Calibri Light), Bold

Formatted: Font: (Default) +Headings (Calibri Light), Bold

Voor de volledige uitleg van deze artefacten verwijs ik u door naar:

- Bijlage 3: Plan van Aanpak (betekenis en motivatie SCRUM activiteiten en artefacten)
- Bijlage 4: Requirements document (voor de product en sprint backlog)
- Bijlage 5: Definition of Ready
- Bijlage 6: Definition of Done

Activiteiten (per sprint):

- Daily scrum: Dit was een 5-10 minutengesprek met alle developers van CERRIX. Wij vertelden dagelijks aan elkaar wat wij gister deden, vandaag doen en tegen welke struikelblokken wij liepen. Dit was handig, omdat ik door dit gesprek hulp kon vragen aan mijn collega's als ik vast zat.
- Sprint execution: Dit was de term voor het begin van de sprint en het beginnen met de implementatie.
- 'Done': Dit was het moment wanneer een increment aan het einde van de sprint 'Done' was.
- Sprint review: Dit was het perfecte feedbackmoment. Tijdens de sprint review werd er gezeten met Paul, Ruben, Dart en Maurits en werd het increment besproken. Als er een verandering in wensen was, dan zou dit middels deze feedbackronde meegenomen kunnen worden naar de volgende sprint.
- Sprint retrospective: Dit was het perfecte moment om te reflecteren op mijn eigen proces. Wat ik goed vond gaan, minder goed vond gaan en wat ik geleerd heb.

Deze SCRUM activiteiten werden uitgevoerd en de uitwerking hiervan kunt u terug vinden tijdens de executie van de sprints.

§ 4.5 SPRINTS EN SPRINTPLANNING

Er was gekozen om het werk te verdelen onder 3 sprints. De sprints duurden elk 3 weken, omdat deze kleine iteraties ervoor hebben gezorgd dat er genoeg feedbackmomenten waren, mochten er aspecten of requirements veranderen. Daarnaast was er zo op het einde nog genoeg tijd over voor het maken van de scriptie. (Zie tabel 4.3: sprintplanning). ~~á~~

De nadruk was tijdens dit project gelegd op het verbeteren van de performance. Hierdoor werd er eerst gekeken naar een mogelijkheid om het laden van de data van alle modules te versnellen, en daarna werd er een mogelijkheid voor de versnelling van het filteren gezocht. Uiteindelijk werd dit client-side en server-side gedaan. Door zowel de server als de client-side proberen te verbeteren, werd er gekeken naar alle raakvlakken om de pagina te versnellen.

Formatted: Justified

Formatted: Font: (Default) +Headings (Calibri Light), Bold

Formatted: Font: (Default) +Headings (Calibri Light), Bold

Formatted: Normal, Justified, Indent: Left: 0,04 cm, No bullets or numbering

Formatted: Justified, Indent: Left: 0 cm, Hanging: 0,02 cm

Formatted: Justified

Sprint	Week	Thema	Op te leveren producten
Sprint 0	Week 1 - 3	Het opzetten van SCRUM	• Plan van aanpak • Requirementsanalyse document • Documenten over de opzet van SCRUM • Product backlog n.a.v. de requirements
Sprint 1	Week 4 - 6	Het maken van het nieuwe ontwerp en het onderzoeken van de bestaande codebase.	• Ontwerprapport
Sprint 2	Week 7 - 10	Het implementeren van de server-side versnelling n.a.v. het ontwerp	• Implementatie van het ontwerp in Angular met versnelling aan de server-side.
Sprint 3	Week 11 - 13	Het implementeren van de client-side versnelling n.a.v. het ontwerp	• Implementatie van het ontwerp in Angular met versnelling aan de client-side
	Week 14 - 17	Het afronden van de scriptie	• Volledige scriptie

Tabel 4.3: sprintplanning concept. Het kan natuurlijk verschillen als de sprints daadwerkelijk doorlopen worden, omdat werk naar de volgende sprint meegenomen kan worden.

~~requirementse~~

Formatted Table

Formatted: Caption, Justified

Formatted: Justified

Formatted: List Paragraph, Justified, Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm

Formatted: Justified

Formatted: Justified

Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands)

Formatted: Justified

Formatted: List Paragraph, Justified

Formatted: Justified

Formatted: Font: (Default) +Headings (Calibri Light), Not Bold

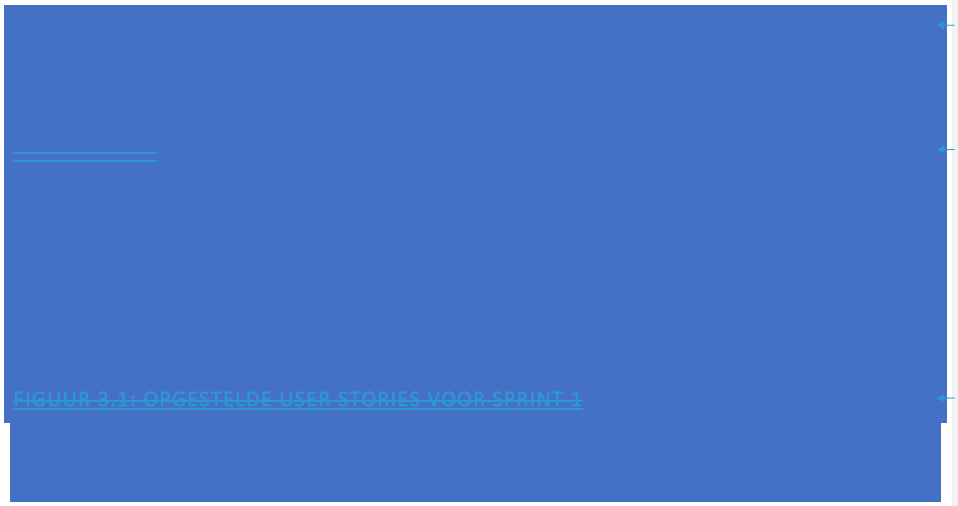
Formatted: List Paragraph, Justified, Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm

Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands)

Formatted: Justified

Refereren naar bijlage

1opzetten van SCRUMInsert bijlage



FIGUUR 3.1: OPGESTELDE USER STORIES VOOR SPRINT 1

Formatted: Heading 1, Justified

Formatted: Heading 1, Justified, Tab stops: Not at 9,51 cm

Formatted: Justified

HOOFDSTUK 5 SPRINT 0 – HET OPZETTEN VAN SCRUM

In dit hoofdstuk zal er aan bod komen hoe sprint 0 doorlopen was. De requirements zijn in dit hoofdstuk achterhaald. Er zal toegelicht worden dat de requirements middels een interview werden vergaard en door middel van een meeting, waarin de wensen van Paul aan bod kwamen. Deze requirements kregen vervolgens een MoSCoW-inschatting en een tijdsinschatting, waarna ze zijn opgenomen in de product backlog in de vorm van user stories.

§ 5.1. HET ACHTERHALEN VAN DE REQUIREMENTS

Voordat er begonnen kon worden met het coderen, moesten er eerst requirements achterhaald worden, zodat er bekend was wat er concreet veranderd moest worden. Hieronder zal uitgelegd worden hoe het vergaren van de requirements was verlopen.

§ 5.1.1 HET VERGAREN VAN DE REQUIREMENTS

(Voor uitleg over de volledige stappen die geleid hebben tot de requirements, verwijs ik u door naar Bijlage 4: Requirements document)

Om zo correct en eenvoudig mogelijk achter de requirements te komen werd er beslist om een interview te houden. Het houden van een interview leek mij de beste oplossing, omdat er kwalitatieve informatie verzameld moest worden, dit leek verbaal het handigst, omdat er genoeg collega's zijn die genoeg wisten over de business dimension navigator. Daarnaast was het kader van de interview explorerend/ verkennend, omdat er nog genoeg requirements verzameld moesten worden en paste een interview hier dus perfect bij. Hierdoor kon er rustig gesproken worden en konden er kwalitatieve vragen gesteld worden voor het achterhalen van de requirements.

Aangezien ik Paul en de andere stakeholders (Maurits en Martin) nog zou vragen wat zij graag verbeterd wouden zien en ik dit altijd nog kon achterhalen, had ik besloten de nadruk van het interview te leggen op wat de klanten zelf van de navigator vonden en graag verbeterd wouden zien. Hierdoor zouden er eventuele verbeterpunten kunnen voortvloeien vanuit de klantperspectief en kon er geverifieerd worden of de aanpassingen die Martin, Paul en Maurits hebben ook overeen kwamen met de wensen van de klanten. Hierom vond ik dit een fijne en nieuwe invalshoek waar naar gekeken werd.

Ik had het interview afgenomen. Rubben en Dart hadden dit afgelegd. Zij waren hier perfect voor, omdat zij beiden Tester en Consultant zijn en direct in contact staan met klanten. Aangezien ik maar een stagiair was en niet in direct contact met de klanten kon staan, was het handig om dit interview aan hun voor te leggen. Omdat zij direct in contact met de klanten staan waren zij voldoende op de hoogte gesteld van zowel de

Formatted: Font: (Default) +Headings (Calibri Light), Not Highlight

Formatted: Font: (Default) +Headings (Calibri Light), Not Highlight

Formatted: Justified

Formatted: Font: (Default) +Headings (Calibri Light), Not Highlight

Formatted: Justified

Formatted: Heading 2

positieve ontwikkelingen als de knelpunten die de business dimension navigator bood. Hierdoor werd het thema van het interview al vrij duidelijk: "*Wat zijn de zowel de negatieve als de positieve aspecten voor het gebruik van de huidige business dimension Navigator?*"

Formatted: Font: (Default) +Headings (Calibri Light), Italic

§ 5.1.2 DE INTERVIEWVRAGEN

(Voor de volledige lijst met vragen verwijs ik u door naar Bijlage 4: Het requirements document)

Om de juiste vragen te bedenken, is er gekeken naar de huidige Navigator en hoe dit de klanten zowel negatief, als positief zou beïnvloeden. Aangezien er niet meteen gevraagd kon worden waar de klanten zich aan storen, maar het onderwerp eerst geïntroduceerd moest worden, waren de eerste twee vragen gesteld. Verder verliep het interview rustig met de gedachte om achter de positieve en negatieve aspecten te komen. Zo werden er de volgende vragen gesteld:

- Waar gebruiken de klanten de business navigator voor?
- Wat voor klanten gebruiken deze module?
- Wat is de positie van deze klanten binnen hun eigen organisatie
- Wat is er al bekend over de ervaring van de klanten bij het gebruik van de module?
- Wat vinden de klanten fijn aan de huidige Navigator?
- Wat vinden de klanten minder fijn aan de navigator?

Formatted: Indent: Left: 1,27 cm, No bullets or numbering

Wat hieruit voortkwam was onder andere het volgende:

De klanten gebruikten de business navigator om op één plek een overzicht te krijgen van hun bestaande bedrijfsproces(sen). Dit hadden zij nodig, omdat hun organisaties opgesplitst werden in meerdere (tot wel 200 bij sommige klanten) bedrijfsprocessen. De navigator bood deze klanten een centraal punt binnen de applicatie, waarbij de risico's, controls, events, moi's en data processings getoond werden. Dit hadden zij nodig, zodat ze hun processen konden verantwoorden naar hun klanten toe en zodat de directie inzag wat voor koppelingen de bedrijfsprocessen hadden met de rest van de applicatie. Hierdoor voelen zij zich 'in control' en hadden zij een duidelijk beeld over de beheersing van hun bedrijfsprocessen.

De verbetering die de klanten zelf wouden zien in de navigator, was dat zij het laden van de data zelf ook versneld wilden hebben. Zij vonden ook dat het te lang duurde voordat er uitvoer op het scherm weergeven werd. Daarnaast vonden zij dat de filter voor het kiezen van de organisatie en business dimension op een onjuiste manier weergeven werd. Het filter zat middels een inklapsysteem moeilijk in elkaar, waardoor het gebruik van de pagina niet begrepen wordt. **(dit was geen nieuwe informatie en was reeds bekend)**

Wat de klanten fijn vonden was het feit dat zij alle informatie betreft hun processen op 1 centraal konden zien. De doorklikfunctionaliteit vonden zij handig, omdat zij hierdoor direct naar de details van hun risks/events etc. konden gaan. **(Dat betekende dat deze functionaliteit overgenomen kon worden in het nieuwe ontwerp).**

§ 5.1.3 WAT UIT DEZE ANTWOORDEN BLEEK

Wat hieruit bleek, was dat de klanten de huidige business dimension navigator onhandig vinden. Zij willen zelf ook graag een verbetering zien voor de werking van de pagina. Het filter zal inderdaad veranderd moeten worden en de wijze waarop de data wordt opgehaald zal versneld moeten worden. Dit klopt ook met wat CERRIX zelf vond en wil verbeteren. Hierdoor zijn zowel CERRIX als haar klanten op één lijn, betreft de grote verbetering.

Nadat het interview was afgerond heb ik een beter beeld gekregen van de navigator, de werking en de impact van de module op de organisaties van klanten. De volgende stap is om aan tafel te zitten met Paul en de consultants, zodat de requirements voor het nieuwe ontwerp hieruit voort vloeien.

§ 5.2 MEETING OM DE REQUIREMENTS NADER TE ACHTERHALEN

Formatted: Heading 2, Justified

Door het houden van het interview is er een beter beeld over welk ergernissen de klanten ervaarden bij het gebruik van de navigator en wat zij graag wilden behouden. Dit was nodig, zodat er zekerheid is dat de klanten en CERRIX op één lijn zijn, zodat dezelfde verbeteringen konden worden gemaakt. Na het interview werd er een meeting gehouden met Paul, Dart, Ruben, Martin en Daniël om hun eigen wensen te bespreken en voor te leggen aan elkaar en het doel was om op uiteindelijke requirements te komen.

(Voor de volledige lijst met wensen en requirements, zie Bijlage 4: Requirements document)

Er werden hier vraagstukken gesteld, die omgezet werden naar de uiteindelijke requirements zoals:

Formatted: Justified

- Kan er een export mogelijkheid komen in excel van de getoonde data?
- Welke kolommen moeten wij weergeven in de nieuwe overzichten?
- Kan het openen van de detail-pagina meegenomen worden in het nieuwe ontwerp?

Het team dat aanwezig was heeft samen gediscussieerd over deze wensen en vraagstukken en ook heb ik hier mijn eigen bevindingen volgens het interview voorgelegd. Hierdoor kwamen wij samen op een aantal requirements. Deze requirements werden vervolgens afgewogen. Er werd gelet op de te besteden uren die ik had en de uren die de implementatie van de wensen vergt, om er zeker van te zijn of deze wensen wel haalbaar waren. Hier werd er een document van gemaakt (Zie Bijlage 4: Requirements document)

§ 5.3. HET OPSTELLEN, INSCHATTEN EN PRIORITEREN VAN DE REQUIREMENTS

Nadat deze wensen een tijdsinschatting in uren hadden gekregen, waardoor er dus een paar wensen af vielen, werden ze onderverdeeld volgens de MoSCoW-methode.

MoSCoW

De MoSCoW-methode is een wijze voor waarop requirements eenvoudig op geprioriteerd kunnen worden. Doordat de wensen die tijdens de meeting afkwamen onderverdeeld werden in de MoSCoW-methode kon er gekeken worden welke wensen daadwerkelijk haalbaar zijn om een minium viable product te leveren, en welke dus, middels de tijdsinschatting daadwerkelijk haalbaar waren om te implementeren binnen het afstudeertraject. Zie tabel 5.2.1.

Tabel *5.2.1 MoSCoW indeling van de requirements*

Must have	- Het systeem moet de data kunnen filteren op basis van organization en business dimension - Het systeem moet de KRI-data kunnen weergeven (dit was er voorheen niet en is er nieuw bijgekomen) - Het systeem moet de event-data kunnen weergeven - Het systeem moet de losse detail pagina van een (risk/control/event etc.) kunnen openen
Should have	- Het systeem moet de gelinkte items tonen van de risks - Het systeem moet de gelinkte items tonen van de controls - Het systeem moet de gelinkte items tonen van de events - Het systeem moet de gelinkte items tonen van de moi's
Could have	- Het systeem moet een afdruk functionaliteit kunnen hebben
Won't have this time	- Het systeem moet een excel uitdraai kunnen maken van de review scores - Het systeem moet een rapportage kunnen tonen van de business dimension, met gekoppelde data

Dit is een deel van mijn uiteindelijke requirements. Voor de volledige lijst met requirements verwijs ik u door naar het requirements document in bijlage 4

§ 5.4. HET OPZETTEN VAN DE PRODUCT BACKLOG

Nadat de requirements waren achterhaald, werden ze opgenomen in de product backlog. Uiteraard werden deze wensen eerst omgeschreven tot user stories die voldeden aan de voorwaardes die beschreven stonden in het DoR. Zo waren er onder andere de volgende user stories gemaakt en toegevoegd aan de product backlog: (zie tabel 5.3.1). Nu was de product backlog compleet en werd er begonnen met de uitvoering van de echte sprints.

Formatted: Font: (Default) +Headings (Calibri Light), Italic
Formatted: Font: (Default) +Headings (Calibri Light), Bold
Formatted Table
Formatted: List Paragraph, Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm
Formatted: Font: (Default) +Headings (Calibri Light), Bold
Formatted: List Paragraph, Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm
Formatted: Font: (Default) +Headings (Calibri Light), Bold
Formatted: List Paragraph, Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm
Formatted: Font: (Default) +Headings (Calibri Light), Italic
Formatted: Font: (Default) +Headings (Calibri Light), Bold
Formatted: List Paragraph, Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm

tabel 5.3.1 user story indeling van de requirements

# User Story ID	Priority (H – L)	User story	Est. time
U07	H	As < an overviewer > I want < to see my Moi's according to the **server-side filter** - when i apply my configuration> so <that I can get an overview>	12
U08	H	As < an overviewer > I want < to see my Data Processings according to the **server-side filter** - when i apply my configuration> so <that I can get an overview>	12

HOOFDSTUK 6: SPRINT 1 - HET MAKEN VAN HET NIEUWE ONTWERP EN HET ONDERZOEKEN VAN DE BESTAANDE CODE BASE

Het probleemdomein was geanalyseerd, de requirements waren achterhaald en de backlog was opgezet. Vanaf dat moment kon er begonnen worden met de uitvoering van de echte sprints. In dit hoofdstuk is allereerst de totstandkoming van het nieuwe ontwerp beschreven. Daarnaast is de bestaande code base geanalyseerd en zijn er vergelijkingen gemaakt tussen Angular en andere frameworks om te kijken of Angular wel de juiste framework was voor het oplossen van dit probleem. Ook zijn de rollen en rechten van de applicatie kort beschreven en is er uitgelegd waarom er zowel een server-side als een client-side implementatie gemaakt zal worden voor het laden en filteren van de data. De user stories die er deze sprint waren uitgevoerd zijn beschreven in tabel 6.1

# User Story ID	Priority (H – L)	User story	Est. time
U01	H	As < a developer> I want< toresearch the performance server side >, < so that I can see how fast this is >	40
U02	H	As < a developer>I want< to create a new design for the webpage >, < so that I know what to implement >	40
U03	H	As <a developer> I want to test whether Angular is the correct chosen framework	16
U04	H	As < a Developer > I want < to have an Angular component analysis based on the design> so < that I know which angular components to (re)use >	40
U05	H	As < a Developer > I want < to know how the Codebase works > so < that I know how the system is set up >	40

Figuur 6.1 Uitgevoerde user stories in sprint 1

§ 6.1. HET MAKEN VAN HET NIEUWE ONTWERP

Wat er nodig was om aan de slag te gaan met programmeren en het implementeren van de herschrijving was het nieuwe grafisch ontwerp. Hiervoor was er gezeten met Paul, Ruben en Dart. Mijn eigen verbeterpunten voor de nieuwe te weergeven kolommen voor in de overzichten werden voorgelegd en meegenomen in het nieuwe ontwerp. Daarnaast waren de wensen van Paul en Ruben ook opgenomen in dit ontwerp.

§ 6.1.1. MIJN EIGEN AANPASSINGEN AAN HET ONTWERP

Uit de requirements bleek dat het ontwerp op een paar aantal aspecten verbeterd moest worden. Zo moest het ingewikkelde filtersysteem verbeterd worden en werd er te veel data weergeven in één keer. Uit eigen initiatief wou ik graag zelf de weergave van de nieuwe kolommen verbeteren voor alle zes modules. Als dit immers gedaan was, dan had ik meer inzicht en begrip over de werking van de applicatie, omdat ik hiervoor door de applicatie zou moeten gaan en zou moeten beslissen welke kolommen de belangrijkste waren per module. Hieronder zal er behandeld worden wat er gedaan was om tot de nieuwe kolommen voor het risicotabel te komen.

Uit de requirements bleek dat het ontwerp op een paar aantal aspecten verbeterd moest worden. Zo moest het ingewikkelde filtersysteem verbeterd worden en werd er te veel data weergeven in één keer. Uit eigen initiatief wou ik graag zelf de weergave van de nieuwe kolommen verbeteren voor alle zes modules. Als dit immers gedaan was, dan had ik meer inzicht en begrip over de werking van de applicatie, omdat ik hiervoor door de applicatie zou moeten gaan en zou moeten beslissen welke kolommen de belangrijkste waren per module. Hieronder zal er behandeld worden wat er gedaan was om tot de nieuwe kolommen voor het risicotabel te komen.

Nieuwe weergave: kolommen voor het risicotabel

In de huidige Navigator werd er een overvloed aan data weergegeven. Bijna alle attributen van de risico's vielen er te zien op de pagina, waardoor het navigeren door de pagina onduidelijk was, omdat er te veel informatie te zien was op het scherm. Ik besloot om een top 8 te maken van attributen om te laten zien op het scherm, die mij het handigst leken om te laten zien op de pagina. Een top 8, zodat de risicotabel dat weergegeven werd nog leesbaar zou zijn, omdat er zo niet te veel informatie op het scherm was.

Dit deed ik door o.a. na te denken wat een overviewer zou verwachten van de B.D.N. Het was een plek waar de gebruiker noodzakelijke informatie hoorde te zien over zijn processen. Wat waren nou de noodzakelijke kolommen om weer te geven van de risico's? Hiervoor had ik Dart gevraagd of hij uitleg kon geven over het nut van de risico's en wat de attributen betekenden, omdat bij het maken van een risico in de applicatie er veel velden (attributen) ingevuld dienden te worden (zie figuur 6.1.2). Dit betekende dus dat het veld wat er momenteel weergeven wordt 'risk description' dus eigenlijk niet nodig was, omdat de nieuwe view algemenere informatie nodig heeft, waardoor er in een oogopslag gekeken kan worden of het proces wel 'in control' is. (Of de risico niet plaats heeft gevonden).

Dit heb ik gedaan door alle attributen die weergegeven werden op te schrijven en met de gedachte van wat een gebruiker van de module verwacht, gedetailleerde kolommen weg te strepen en alleen de noodzakelijke

kolommen, die echt iets over de risico's zeggen te behouden. Hierdoor kwam ik uit op de volgende kolommen: (zie tabel 6.1.1)

Ter illustratie: als wij de belangrijkste gegevens nodig hebben van een student, hebben wij niet zijn of haar telefoonnummer nodig, maar zijn klas/ ID/ opleiding etc. Hetzelfde geldt voor de informatie die de navigator hoort te tonen. Als de overige informatie, zoals het telefoonnummer toch nodig is, dan kan de gebruiker dubbelklikken om verder naar de details pagina verwezen te worden.

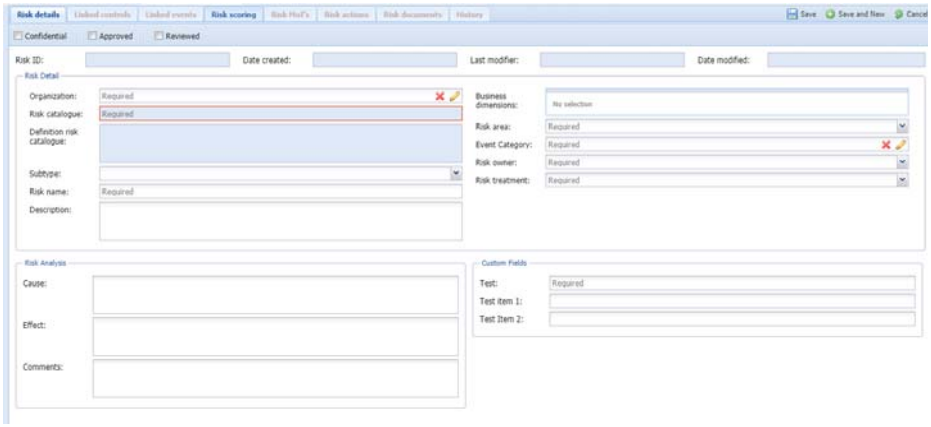
Tabel 6.1.1: Uiteindelijke kolommen voor het risicotabel dat weergeven wordt

Risk ID	Door de applicatie heen kwamen overal de ID's terug. Hetzelfde verwachtte ik dan ook voor de risico's in de navigator
Risk Name	Uiteraard moet het duidelijk zijn over welke risico het gin gen moest de naam terug komen op het scherm
Organization	Hiermee konden de risico's getypeerd worden op organization, aangezien de navigator hierop filterde
Risk owner	Dit is de risicoeigenaar en is belangrijk om meet e nemen omdat er naar deze risicoeigenaar toegegaan moet worden als het risico voorkomt
Gross score	Dit is de bruto score van het risico was hierom een belangrijk detail voor overviewers om naar te kijken.
Net score	Dit is de netto score van het risico en was hierom wederom een belangrijk detail voor de overviewers om naar te kijken
Risk assessment	Dit is het resultaat van het risico, of het wel of niet uitgevoerd werd of aanvaardbaar was. Een overviewer zou dit handig vinden, omdat hij zo de score van de risico kon zien.
Risk appetite	Dit was ook een scoring van de risico op basis van de risk appetite (risicobereidheid). Overviewers zouden dit handig vinden om te zien, zodat ze hierdoor de bereidheid van de organisatie zien om het risico te beheersen.

Deze kolommen waren voorgelegd aan Paul, Ruben en Dart en zij vonden ook dat dit de beste kolommen waren om te laten zien in het nieuwe ontwerp en zij vonden dat er elke tabel een ID, Name en Organization moest laten zien, omdat dit applicatie breed ook gebeurd. Deze 8 kolommen werden dus opgenomen voor in het nieuwe

ontwerp. *(Voor een tabel met de weergave van alle nieuwe kolommen van de zes modules, zie tabel 5 van het ontwerpdocument in de bijlages).*

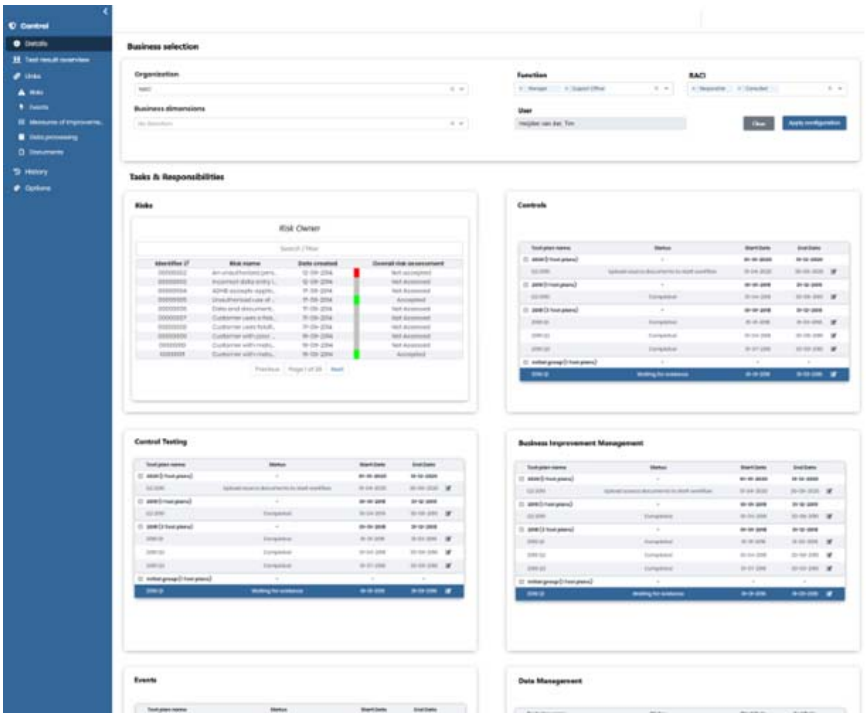
Daarnaast zag ik dat er in de applicatie ook KRI's waren die gekoppeld waren aan organizations. Omdat de navigator filterde op organization en business dimension, konden er in het nieuwe ontwerp ook KRI's meegenomen worden om te laten zien in tabelvorm. Dit werd ook voorgelegd en Paul vond dit een goed idee. Er zou dus ook een overzicht van de KRI's komen in het nieuwe ontwerp. Voorheen bestond dit nog niet in de navigator.



6.1.2: De velden die ingevoerd moesten worden om een risico aan te maken. Er moest een selectie gemaakt worden om de belangrijkste kolommen te kiezen voor de weergave in de navigator.

§ 6.1.2. HET NIEUWE ONTWERP

Naast mijn eigen input voor de invulling van de tabellen op de webpagina, hadden Paul, Ruben en Dart ook hun eigen input gegeven voor het nieuwe ontwerp. Bijvoorbeeld dat het filter voor de selectie van een organization en business dimension een 'clear' button moest krijgen, zodat het filter werd gereset en dat de risk actions uitgefaseerd werden, omdat daar nu de moi's voor zijn gemaakt. Dit werd allemaal in elkaar gezet in paint door Ruben en zag het nieuwe ontwerp er als volgt uit (zie figuur 6.2.1). Aan deze meeting deed ik actief mee en heb ik mee kunnen praten toen de selectie werd gemaakt tussen de lettertype, de grootte hiervan en de grootte van de knoppen.



6.2.1: Het nieuwe ontwerp van de business dimension navigator. Op de afbeelding zijn de zes nieuwe tabellen te zien van de Risk, Control, Moi, Event, KRI en Data Processings module. Daarnaast is er op de linker navigatiebalk een tab voor de documenten te zien.

Na mijn input voor de herziene kolommen werd het ontwerp grafisch samengesteld door Ruben. Verder waren hier nog wensen van Paul in opgenomen, betreft de lettergrote, grootte van de tabellen (deze zijn nu 50% van het scherm, maar misschien wordt dit toch later groter, omdat dat mooier oogt). In combinatie met deze wensen is het nieuwe ontwerp tot stand gekomen (zie figuur 6). Dit zal het nieuwe ontwerp voor de business dimension navigator worden, maar is nog een concept. Het kan nog veranderen, omdat het, eenmaal geïmplementeerd misschien anders kan ogen.

§ 6.2 HET ANALYSEREN VAN DE BESTAANDE CODEBASE

De CERRIX-applicatie is opgebouwd d.m.v. een vier-lagenmodel. Dit betekent dat een aanvraag aan de voorkant door vier lagen moet gaan om de database te bereiken om data op te halen, deze data vervolgens dan weer terug naar boven moet sturen, naar de hoogste laag, zodat er uitvoer weergeven kan worden op het scherm.

Allereerst is er de Application-laag: hier staan de views die weergeven worden op het scherm. Deze laag staat vervolgens direct in contact met de de Business-laag: in deze laag wordt er business logica toegepast en wordt er data gemanipuleerd. De data die er gemanipuleerd of opgevraagd wordt komt uit de De Data Access Layer: deze laag staat direct in contact met de database en heeft als enige de mogelijkheid om data op te halen. Deze database staat in de **DATA**-laag en hierin zijn de database-modellen gedefinieerd.

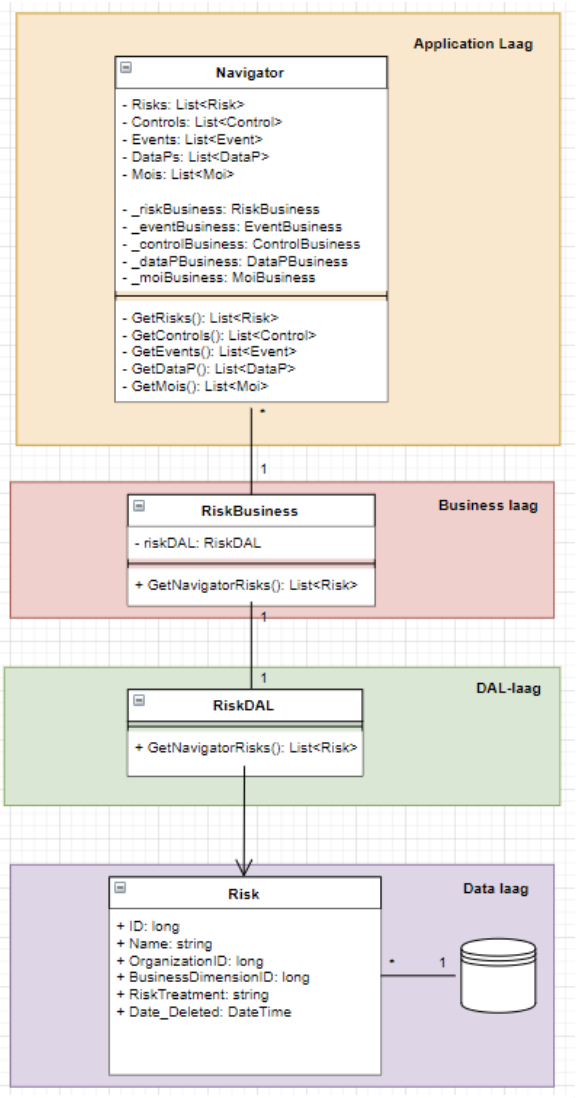
Formatted: Justified

~~Kwaliteitsbewaking~~

Deze lagen werken allemaal samen met elkaar d.m.v. een hiërarchie van boven naar beneden. Nadat er geklikt wordt op de webpagina gaat er via de application laag (view), een call door naar de business laag, die vervolgens naar de DAL gestuurd wordt, die data ophaalt en alles vervolgens weer terug stuurt naar het scherm.

§ 6.3 UML ONTWERP HUIDIGE SITUATIE

Om te begrijpen hoe deze lagen met elkaar samen werken zal er hieronder een UML-diagram van de huidige situatie weergeven worden voor het opvragen van de riskdata. *(zie figuur* 6.3.1).



figuur 6.3.1: (versimpelde) UML-ontwerp van de navigator module van de huidige situatie

In de application-layer staat de huidige view (webpagina) van de navigator. Hier roept de navigator code om de riskdata weer te geven. Deze functie staan in de RiskBusiness.

In de business-layer zit er de functie GetNavigatorRisks(), die opgevraagd wordt aan de DAL-layer. De toegevoegde waarde van de business-layer is dat deze data de vorm van een riskmodel krijgt.

In deze DAL-layer zitten er functies in die de daadwerkelijke database-objecten ophaalt.

De DAL-layer staat direct in contact met de DATA-laag, waar de database en risktabel in staan.

Nadat alle lagen van boven naar beneden zijn afgegaan, wordt de data natuurlijk terug naar boven gestuurd, zodat het weergeven kan worden op de navigator-pagina.

Nu er een UML-diagram is gemaakt snap ik de werking beter tussen de gelaagdheid van de applicatie.

§ 6.4 ROLLEN EN RECHTEN IN DE APPLICATIE

De applicatie toont verschillende soorten data, van confidentiële risico's tot processen die alleen de managers van een afdeling mogen zien. Niet alle werknemers van een organisatie horen toegang te hebben tot deze rechten. Daarom heeft CERRIX een rollen en rechten functionaliteit toegevoegd in de applicatie. Zie figuur 6.4.1.

Ter illustratie: Een student aan de HHS hoort ook geen toegang te hebben tot de module op Osiris om cijfers op in te voeren. Dit hoort afgeschermd te zijn en deze rechten horen alleen toegekend te worden aan docenten of hoger. Hetzelfde gebeurd er in het rollen en rechten systeem van CERRIX.

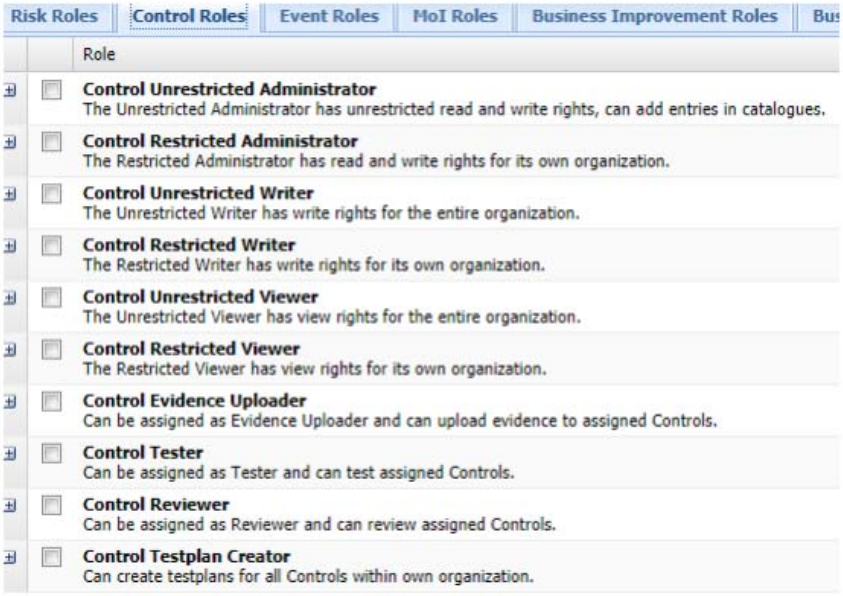


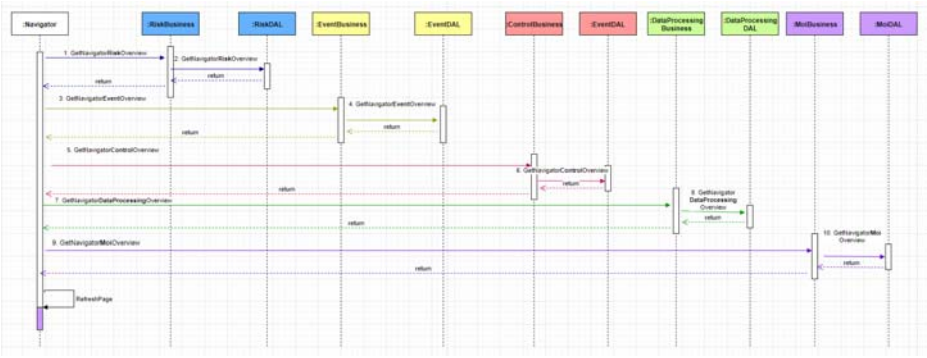
Figure 1 Figuur 6.4.1: Rollen en rechten voor de Control module

Hieruit valt er dus op te merken dat een werknemer bijvoorbeeld een control restricted viewer kan zijn, waardoor hij alleen controls kan zien die alleen binnen zijn eigen organisatie vallen en hierdoor dus geen toegang heeft tot andere data. Zo zijn er complexe rollen en rechten voor alle overige modules van de applicatie.

§ 6.5 SERVER-SIDE VS CLIENT-SIDE FILTERING EN VERSNELLING

Voordat de user stories echter opgepakt kunnen worden, moet er eerst gekeken worden naar de manier hoe de data op dit moment gefilterd wordt en doorgestuurd wordt, zodat er begrepen wordt hoe het ophalen en laden van de data versneld kan worden.

De data van alle zes modules die op het scherm weergeven wordt, wordt op dit moment **sequentieel** opgehaald. Dit houdt in dat er tijdens het laden van de pagina (alvorens er al een filter op organisatie en bedrijfsproces is toegepast) allereerst alle risico's worden opgehaald uit de database, pas nadat de risico's binnen zijn, de controls, events, moi's, en data processings worden opgehaald. Er wordt dus telkens gewacht totdat de data per module is binnengekomen. Dit zorgt ervoor dat het laden van de pagina dus +/- 20 seconden duurt. Om deze gedachte te ondersteunen is er een sequentie diagram getekend (zie figuur 6.5.1), dat laat zien dat alle modules op elkaar wachten voor het ophalen van hun eigen data.



Figuur 6.5.1 Sequentiediagram van de huidige situatie (alles wordt sequentieel opgehaald)

Om dit op te kunnen lossen moeten alle modules dus niet wachten op elkaar om data op te kunnen halen (**sequentieel**), maar moeten deze calls tegelijkertijd, en dus **parallel** uitgevoerd worden. Dit betekent dat er een omschrijving van de code nodig is zodat dit kan gebeuren, zodat zowel het laden als het filteren tegelijkertijd gebeurd met de calls van de andere modules, wat de performance theoretisch al verbeterd. In sprint 2 zal dit verder aan bod komen

§ 6.5.1 SERVER-SIDE DATA FILTERING

Naast het parallel ophalen en filteren van de data, is het ook een requirement om te achterhalen welke implementatie voor het filteren van de data sneller is: **server-side of client-side?**

Het server-side filteren van de data betekent dat er nadat er geklikt wordt op de filter-knop, er een call naar de database gestuurd wordt om alle risico's op te halen en ze vervolgens te filteren op basis van de gekozen selectie. Op dit moment is dit al het geval, maar zal er geprobeerd worden om de server-side filtering toe te passen op de parallelle implementatie voor het laden van de data.

Als het ophalen van de data voor alle zes modules uiteindelijk parallel wordt aangeroepen, dan kan deze data server-side gefilterd, om te kijken of dit veel sneller gebeurd dan voorheen, omdat de data niet meer sequentieel wordt opgehaald.

De functies die de navigator op dit moment heeft voor het ophalen van de data uit de modules is nog op een oude wijze geschreven omdat er niemand in jaren aan de navigator gezeten heeft. Daarom moeten deze functies voor bijvoorbeeld het ophalen van de risks, controls, moi's etc. omgeschreven worden zodat ze de functies gebruiken die de risk-module, control-module etc. op dit moment wordt gebruikt, omdat de developers van CERRIX deze functies al wat hebben versneld. Op deze data zal er vervolgens server-side filtering toegepast worden.

§ 6.5.2 CLIENT-SIDE DATA FILTERING

Naast een server-side versnelling, waarbij de data telkens aan de achterkant opnieuw wordt opgevraagd en gefilterd, is het ook mogelijk om de data aan de voorkant (client-side, in de browser van de gebruiker) te filteren. Dit houdt in dat de data van alle modules client-side wordt opgeslagen (dit wordt nog steeds parallel opgehaald) en tijdens het filteren deze zelfde data gefilterd wordt op basis van de org en het process. Doordat de data van alle modules virtueel in de browser wordt opgeslagen (client-side), hoeft de data niet per keer dat het filtert opnieuw opgehaald worden vanuit de database, maar is het al in de browser opgeslagen en hoeft alleen deze data telkens gefilterd worden. Op dit moment hebeben de overige Angular pagina's van de risk/control/event modules al een client-side filtering implementatie en was het ook een requirement om dit te implementeren in de navigator, om te kijken of dit sneller is dan de server-side implementatie. Door zowel server-side als client-side te implementeren weten wij zeker dat er coverage is op grond van deze twee vlakken voor de versnelling van de navigator.

§ 6.6 VERGELIJKING ICT-GERELATEERDE OPLOSSINGEN

CERRIX heeft in de loop der tijd al veel specifieke en generieke Angular componenten gemaakt. Dit zijn HTML en CSS klasses, die dynamisch werken met TypeScript (een aparte programming language dat compileert naar JavaScript). Dit hebben zij bijvoorbeeld gemaakt om zo bepaalde lijsten met data weer te geven, die middels CSS voldoen aan de CERRIX-huisstijl.

Alleen is Angular wel de juiste framework voor mijn opdracht? Ik zocht het uit om te kijken of het Angular wel de juiste keuze zou zijn om het nieuwe ontwerp in te implementeren. Misschien was er een betere, snellere, nieuwere framework wat gebruikt kon worden aangezien de business dimension navigator module toch helemaal herschreven moest worden en dit dan liever gelijk goed (met de nieuwste ontwikkelingen) kon gebeuren. Mocht het zo zijn dat ik een 'betere' framework gevonden had, dan kon ik dit aanbevelen aan CERRIX en zou ik zo ook een stukje bijdragen aan de verbetering van de ontwikkelstructuur (gekozen framework). **Voor de volledige analyse verwijs ik u door naar Bijlage 8: Angular analyse.**

Dit heb ik uitgezocht door allereerst te kijken naar de drie grootste frameworks die gebaseerd zijn op JavaScript. Deze zijn onder meer: Angular, React en Vue.js. *(Naar 10 Best JavaScript Frameworks to Use in 2019 en Angular vs React vs Vue: Which is the Best Choice for 2019?)* Deze drie frameworks zal ik vergelijken om te zien of er een betere framework beschikbaar is voor CERRIX op dit moment om te gebruiken. Hieronder zal ik per framework een korte introductie geven en daarna deze frameworks (zie figuur 6.3.4), waarna ik mijn aanbevelingen hierover zal uitspreken.

Angular

Angular is uitgekomen in 2010 door Google en is een typescript based js framework. Google is o.a. een van de grote bedrijven die deze framework gebruikt. Angular focust zicht vooral op het gebruik van een single page application. (De webpagina verandert niet. Alle elementen worden dynamisch herladen op dezelfde pagina). *(Naar: Angular).*

React

React is uitgekomen in 2013 door Facebook. Het wordt het meeste gebruikt voor high-traffic websites, zoals websites die veel advertenties gebruiken als Facebook en dus trager zijn. React is zeer dynamisch en focust zich meer op interactieve en simpele user interfaces voor gebruikers. *(Naar: React – A JavaScript library for building user interfaces).*

Vue.js

Vue is uitgekomen in 2014 door een ex-engineer van Google. Websites zoals GitLab en Alibaba gebruiken Vue. Vue noemt zichzelf de progressive JavaScript framework. Dit komt omdat Vue zich ook focust op Single page

applications en op user interfaces. De ex-engineer, Evan You heeft elementen van Angular, React en overige frameworks gepakt en samengevoegd tot een lightweight framework die hij handiger vond. *(Naar: Vue.js).*

Figuur 6.3.4: Vergelijking Angular vs React vs Vue.js

	Performance	**Framework size** (belangrijk voor de scalability en snelheid)	**Learning curve**	**Flexibility**
Angular	Gebruikt de echte DOM (representatie van UI in code), waardoor dit eerst geladen moet worden en trager is dan virtuele DOM.	500 KB 'heavy weight'. Niet goed voor light weight applicaties, omdat Angular veel templates te bieden heeft van het testen tot utilities toe. (CERRIX is heavy weight, omdat er parallel veel data wordt opgehaald en doorgestuurd).	Angular vergt enige tijd om te leren, omdat de taal is opgebouwd uit TypeScript (een set van JS om gemakkelijker te programmeren volgens conventies).	Angular biedt alles aan wat nodig is. Van routing tot templates. Hierdoor heb je geen andere tool nodig om te developen meer.
React	Gebruik van virtuele DOM, dit wordt middels de framework virtueel opgeslagen in het geheugen zonder alle echte zware koppelingen van de echte DOM. Doordat dit virtueel gebeurd is het sneller dan het gebruik van de echte DOM. (lightweight)	100 KB. Het is goed voor light weight applicaties, omdat het niet zoveel templates aanbiedt zoals Angular. Daarom heeft React hulp nodig van losse libraries om extra functies te hebben.	Dit is in JavaScript en dus makkelijk om direct mee aan de slag te gaan.	React biedt niet veel extra tools aal via de React library. Dit betekent dat er third-party libraries gebruikt moeten worden.
Vue.js	Gebruikt eveneens de virtuele DOM, wat sneller is dan de echte DOM.	80 Kb. Dit is een van de smalste frameworks. Hierdoor is het super goed voor light weight frameworks, maar biedt hierdoor geen extra functies aan.	Dit is eveneens in JavaScript.	Vue biedt eveneens niet veel extra tools aan.

Wat hieruit valt te concluderen is dat React en Vue inderdaad goede frameworks zijn. Ze zijn nieuw, sneller dan Angular, omdat er gebruik wordt gemaakt van de virtuele DOM en zijn gemakkelijk aan te leren omdat ze javascript hanteren. Vue is de meest geliefde library op dit moment, omdat zij het goede van Angular en React combineert. Echter zijn Vue en React niet geschikt voor heavy-weight en volledige web-applicaties, omdat zij niet

de juiste tools bieden om dit te ondersteunen. Angular biedt dit wel en daarom is de keuze toch naar Angular gegaan.

De keuze om te werken in Angular vind ik persoonlijk ook handiger, omdat het team van CERRIX ook al programmeert in Angular en zij mij hierom de beste ondersteuning kunnen bieden, mocht ik tegen problemen aanlopen. Ook hebben zij diverse (generieke) componenten ontworpen waarmee er bijvoorbeeld in HTML, CSS en TypeScript tabellen worden weergeven op het scherm en die direct in contact staan met de database. Dit kan ik dan ook mooi combineren met mijn navigator, zodat ik dit alleen nog hoef te implementeren. Er zal binnen het project dus inderdaad gewerkt worden met de framework Angular. Dat heeft deze analyse versterkt.

Nadat ik de drie grootste js frameworks heb vergeleken, viel het toch op dat Angular de geschikte keus is voor CERRIX. CERRIX is namelijk een grote en zware applicatie dat heavy weight is, door zijn parallelle aard en de zware calls voor het ophalen en versturen van data.

Daarnaast is CERRIX al bezig van de grote overstap van Ext.Net naar Angular, waardoor alle developers al gespecialiseerd zijn in TypeScript en dit niet meer aangeleerd hoeft te worden. Ook biedt Angular alles aan wat CERRIX nodig heeft, zo kan de routing gemakkelijk via Angular zelf geregeld worden en is de applicatie van CERRIX een Single Page application, waarbij Angular ook handig voor is. Dit betekent dus dat CERRIX de juiste framework heeft gekozen voor de applicatie en hierom zal ik de navigator nusiness dimension ook implementeren in Angular.

§ 6.7. SPRINT 1: REVIEW

Tijdens de sprint review is het product bekeken. Er is een nieuw ontwerp opgeleverd inclusief een ontwerpdocument **(zie bijlage 7)** voor het ontwerp en de bestaande codebase. Dit zijn de taken die stonden op de product backlog en ze zijn juist uitgevoerd en werkelijkheid geworden. Hierdoor konden deze items afgevinkt worden van de backlog.

Het ontwerp is laten zien aan Paul, Maurits en Ruben en zij vonden het ontwerp goed en willen dit graag geïmplementeerd zien worden. Daarnaast liet in aan mijn begeleider Martin mijn afweging voor de frameworks zien. Hij vond dat ik een goede afweging had gemaakt, waardoor hij zelf ook informatie ging opzoeken over React en Vue uit interesse. Tijdens de volgende sprint zullen er dus alleen items opgepakt worden die behoren bij sprint 2, en niet meer van sprint 1.

De volgende stap zal zijn om dit te implementeren om het ontwerp te realiseren. Dit zal gedaan worden in de volgende sprint.

§ 6.8 SPRINT 1: RETROSPECTIVE

Er is gezeten met Martin om samen te bespreken wat er goed en fout ging tijdens de sprint. Wat er goed ging was dat alle backlog items tijdig zijn afgekomen en dat er dus een mooi ontwerp en codeanalyse is opgeleverd. Dit was van belang, zodat de basis is begrepen van wat en waar er aan gewerkt moet worden. Deze sprint was dus een nuttige sprint. Ik vond zelf niet dat er iets minder goed ging, omdat alle backlog items van sprint 1 voltooid zijn en Paul, Maurits en Ruben tevreden waren over het feit dat ik de codebase heb weten te begrijpen.

Daarnaast is er tijdens deze sprint een duidelijker beeld verkregen over waarom de framework Angular gekozen was, wat de framework inhield en wat Angular extra bood t.o.v. andere frameworks. Aan het begin van deze sprint was dit nog onduidelijk. De naam Angular kwam telkens ter sprake, maar er was nog geen idee wat hier voor gedaan moest worden en in welke taal (TypeScript/HTML/CSS) er geprogrammeerd moest worden.

Ook is het nu duidelijk waardoor de navigator wat trager is, hoe de implementatie hiervan de performance kan verbeteren en hoe dit geïmplementeerd kan worden (middels de overgang van sequentiële naar parallelle methodes voor het ophalen van de data). Hiervoor had ik nog geen idee wat dit allemaal inhield.

HOOFDSTUK 7: SPRINT 2 - HET IMPLEMENTEREN VAN DE SERVER-SIDE VERSNELLING

In dit hoofdstuk zal de daadwerkelijke implementatie van het ontwerp beschreven worden. In dit hooofdstuk zal er één user story door de SCRUM activiteiten heen doorlopen worden. Het ontwerp, de implementatie, het testen, de sprint review en retrospective zullen in dit hoofdstuk verder nog aan bod komen. De user stories die uitgevoerd waren zijn terug te vinden op tabel 7.1.

# User Story ID	Priority (H – L)	User story	Est. time
U06	H	As < an overviewer > I want < to see my Controls according to the **server-side filter** - when I apply my configuration> so <that I can get an overview>	12
U07	H	As < an overviewer > I want < to see my MoIs according to the **server-side filter** - when I apply my configuration> so <that I can get an overview>	12
U08	H	As < an overviewer > I want < to see my Data Processings according to the **server-side filter** - when I apply my configuration> so <that I can get an overview>	12
U09	H	As < an overviewer > I want < to see my KRIs according to the **server-side filter** - when I apply my configuration> so <that I can get an overview>	12
U10	H	As < an overviewer > I want < to see my Risks according to the **server-side filter** - when I apply my configuration> so <that I can get an overview>	12
U11	H	As < an overviewer > I want < to see my Events according to the **server-side filter** - when I apply my configuration> so <that I can get an overview>	12
U12	H	As < an overviewer > I want < to clear my configuration> so <that I can filter on another configuration>	3
U13	H	As < an overviewer >I want< to see the details page of the item I clicked on >, < so that I can get redirected to the details page of the item>	8

Figuur 7.1: De user stories die uitgevoerd zijn in sprint 2

In dit hoofdstuk zal de userstory **U10: As < an overviewer > I want < to see my Risks according to server-side the filter - when I apply my configuration> so <that I can get an overview>** doorlopen worden volgens de SCRUM-activiteiten.

Nadat de user story geïmplementeerd is, hoort deze risk data te tonen op het scherm. De weergegeven data wordt gefilterd op basis van de rechten van de gebruiker. Immers: als een gebruiker een administrator is, kan hij veel meer data inzien, dan een normale (restricted) gebruiker. Er zal rekening gehouden worden met de rollen en rechten van de gebruiker, zodat alleen de risico's weergeven worden die de gebruiker mag zien.

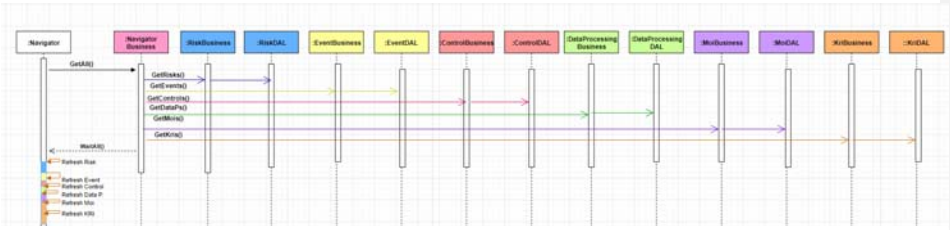
§ 7.1 DE UITVOERING

Deze user story moet er voor zorgen dat een gebruiker alle risico's kan zien van zijn gekozen configuratie. De configuratie is een combinatie van een organisatie en business dimension, waarop de risico's gefilterd worden. Het filteren van de data zal eerst server-side gebeuren. De server-side filtering zal toegepast worden, nadat de data parallel wordt opgehaald.

§ 7.1.1 HET NIEUWE ONTWERP

Tijdens de vorige sprint was het al bekend dat de navigator in de oude situatie alle data ophaalt op een sequentiële wijze (zie figuur 6.5.1), waardoor elke module moest wachten op elkaar voor het ophalen van de data, omdat er maar één module tegelijk haar data op kon halen en dat de oplossing hiervoor een parallelle implementatie zou zijn, zodat de van alle zes modules tegelijkertijd en los van elkaar opgehaald kon worden en los op het scherm weergeven werd. Hieronder zal het ontwerp voor deze parallelle implementatie volgen en zal er beschreven worden hoe dit in de code is verwerkt.

Dit kan verbeterd worden door de sequentiële functies om te schrijven naar parallelle functies. Om deze gedachte visueel te ondersteunen is er hieronder een sequentiediagram gemaakt. Zie figuur 7.1.2. Ik heb gekozen voor de toepassing van parallelle functies, omdat de rest van de applicatie op dit moment ook gebruik maakt van parallelle functies voor het versnellen van het laden van data (voor het ophalen van alle moi's bijvoorbeeld, in de aparte moi module).



Figuur 7.1.2 Sequentie diagram voor de nieuwe parallelle implementatie

Op de afbeelding valt er te zien dat er een verschil is met het sequentiediagram van figuur 6.5.1. De aanvragen naar de database voor het ophalen van de data uit de zes modules worden parallel uitgevoerd. Dat betekent dat er niet langer gewacht hoeft te worden totdat één module zijn data geladen heeft voordat de andere module dit pas kan doen, maar dat alle modules dit tegelijkertijd zullen opvragen. Dit zal ervoor zorgen dat het laden sneller zal verlopen.

Op de afbeelding is er een BusinessNavigatorBusiness bijgekomen, zodat al deze aparte calls vanuit een overzichtelijke business van de navigator aangeroepen kan worden, in plaats van alles direct naar de risk, event, control etc. business te sturen. Ook is dit gedaan omdat er extra code bij komt voor het maken van de parallelle functies en een speciale business hier dus nodig voor is.

De NavigatorBusiness roept vervolgens de risks, controls, events etc. aan en wanneer zij dus allemaal teruggekomen zijn, zal de navigator vervolgens deze data op de pagina herladen. Door het gebruik van deze nieuwe implementatie hoort de pagina sneller te werken. Dit zal geïmplementeerd worden en getest.

§ 7.1.2 DE IMPLEMENTATIE VAN DE BACK-END

Om de back-end te implementeren, moest er allereerst een nieuwe business klasse aangemaakt worden, de 'BusinessNavigatorBusiness.cs'. Hier zou alle code van de nieuwe navigator in staan, zodat er code op één plek zou staan voor alle calls die de navigator maakt om data te laden en te filteren. Op dit moment is er geen speciale business en gaan de calls vanaf de view (in de application-layer) direct naar de bijbehorende risk, control, events businesses om de data uit op te halen. Nu er een lege businessklasse aangemaakt was, werd het tijd om het parallel ophalen van de risico's hierin te implementeren.

De developers van CERRIX hebben al een parallelle functie gemaakt die **alle** risico's ophalen uit de database. Deze functie wordt al gebruikt in de risk-module. De data die deze functie ophaalt checkt eveneens op de rollen en rechten van de gebruiker, waardoor de uitvoer voldoet aan de risico's die een gebruiker wel of niet mag zien. Hier hoef ik verder niets meer voor te doen. Door gebruik te maken van deze functie wordt de uitvoer van de risico's vanuit de navigator al direct gelijkgetrokken met de uitvoer die de gebruiker krijgt als hij alle risico's ophaalt vanuit de risk-module. De methode die de developers zelf al hadden geschreven kunt u terug vinden in bijlage 7: het ontwerpdocument.

Deze functie die **alle risico's** al parallel ophaalden heb ik overgenomen en geïmplementeerd, zodat er al de juiste risico's opgehaald werden die voldeden aan de rollen en rechten van de gebruiker. Alleen moest deze data nog gefilterd worden op basis het filterconfiguratie van een organisatie en business dimension. Dit betekende dat data met **alle** risico's nog (server-side) gefilterd moest worden. Zowel het parallel ophalen van de risico's als het server-side filteren kunt u zien op figuur 7.6.

```
2 references
public IEnumerable<RiskInfoModel> GetLightNavigatorRiskOverview(List<long> orgIds, List<long> bdIds,
    IUserInformation currentUser)
{
    var data = new RiskBusiness(currentUser).GetLightWorkspaceOverview(currentUser).Result;

    return data
        .Where(r => FilterBusinessDimension(r.BusinessDimensionIDs.SplitToLongList(';'), bdIds))
        .Where(r => FilterOrganization(r.OrganizationID, orgIds));
}
```

Figuur 7.6: Mijn eigen functie die eerst de risk data parallel ophaalt en vervolgens de data server-side filtert.

Hierboven valt te zien dat de **data** variabele de data is die uit de RiskBusiness functie komt die al bestaat om **alle** risico's op te halen op basis van de rollen en rechten van de gebruiker (currentUser). Nadat de data binnen is, worden deze risico's gefilterd op basis van de binnengekomen organization ids en business dimension ids, waarna de uitvoer teruggestuurd wordt naar de voorkant van de navigator.

De functies die zorgden voor de server-side filtering, namelijk de FilterBusinessDimension() (figuur 7.7) en de FilterOrganization() (figuur 7.8) heb ik zelf gemaakt, door na te denken hoe een filter werkt. Eerst kwamen alle risico's binnen en moesten deze gefilterd worden op business dimension. Dit houdt in dat er per risico gekeken moet worden of de business dimension id overeenkomt met de id's die staan in het filter. Hetzelfde gold voor de organizations. Dit heb ik dus gemaakt.

```
private bool FilterBusinessDimension(IEnumerable<long> bdIds, List<long> bdIdsDown)
    => bdIdsDown.Count == 0 || bdIds.Any(x => bdIdsDown.Contains(x));
```

Figuur: 7.7: De functie voor het filteren op business dimensions.

Op de afbeelding is de functie te zien die ervoor zorgt dat de risico's server-side gefilterd worden op basis van business dimension ids. Per risico wordt deze functie afgegaan en wordt er gekeken of de meegegeven 'bdIdsDown' voorkomt in het lijstje met 'bdIds' , wat de business dimensions van de risico zelf zijn. Als het erin zit dan wordt deze risico teruggestuurd naar de navigator.

```
4 references
private bool FilterOrganization(long orgId, List<long> orgIds)
    => orgIds.Count == 0 || orgIds.Contains(orgId);
```

Figuur: 7.8: De functie voor het filteren op organisatie.

Op de afbeelding is de functie te zien die ervoor zorgt dat de risico's server-side gefilterd worden op basis van hun 'organization id'. Er wordt gekeken of de 'organization id' van het risico overeen komt met het lijstje 'orgIds', dat komt van het filter. Als dit klopt dan wordt dit vervolgens verder doorgestuurd naar de navigator.

De uitvoer van deze functie is een lijst met risico's die al voldoen aan de juiste security-eisen, namelijk, ze voldoen al aan de juiste rollen en rechten van de gebruiker, doordat de reeds bestaande functie die staat in de RiskBusiness om **alle** risico's op te halen dit al zelf opvangt. Ook zijn deze risico's al server-side gefilterd.

Deze functie, die stond in de BusinessNavigatorBusiness, moest nog wel gekoppeld worden aan de voorkant (application-layer, waar de code staat voor de view van de navigator). Dit gedaan op figuur 7.8.2

```
[HttpGet]
[Route("{orgId}/risks/{processId}")]
0 references
public async Task<IHttpActionResult> GetRisks(long orgId, long processId)
{
    try
    {
        var orgIds = _orgBsns.GetOrganizationsDown(orgId).Select(x => x.ID).ToList();
        var bdIds = _businessDimensionBusiness.GetBusinessDimensionsDown(processId).ToList();

            return Ok(await _businessNavBusiness.GetLightNavigatorRiskOverview(orgIds, bdIds,
                UserInformationManager.CurrentUser));
    }
    catch (Exception e)
    {
        _logger.LogError(e);

        return InternalServerError(e);
    }
}
```

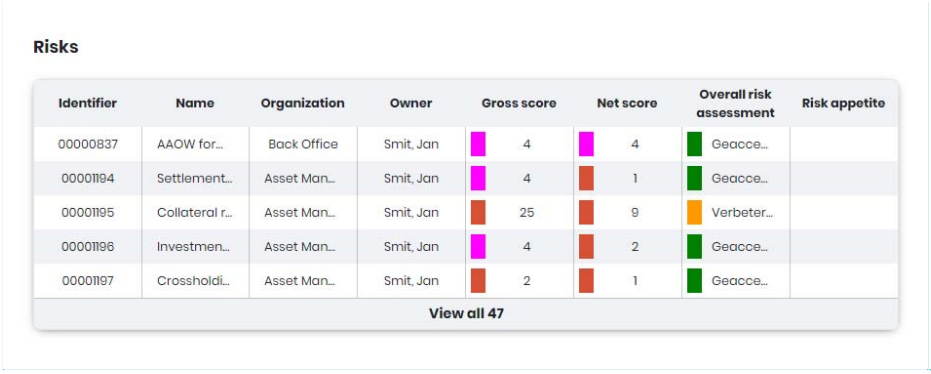
Figuur 2 Parallelle functie in de application-layer die de risico's ophaalt

Op de afbeelding is de GetRisk() functie te zien, die een orgId en een processId verwacht. Deze twee komen beiden uit het filter. Deze functie is parallel geschreven, end at wordt getypeerd door de 'Task' keyword. Doordat er gebruik wordt gemaakt van Tasks, kunnen er parallel taken uitgevoerd worden, omdat C# hier onderwater een aparte thread aanmaakt speciaal voor deze code. De functie roept de code aan die staat op figuur 7.8

De omschrijving voor het maken van een parallelle functie is nu voltooid, alleen wordt de uitvoer nog steeds weergeven op de oude pagina in Ext.Net, omdat uiteraard alleen de back-end op dit moment herschreven is. Hetzelfde is gedaan voor parallel laden en server-side filteren van de controls, events, mois en data processings. De volgende stap is om de tabellen in een nieuw jasje te stoppen, van het oude Ext.Net naar Angular.

§ 7.1.3 DE IMPLEMENTATIE IN ANGULAR (FRONT-END)

De nieuwe weergave van de tabel met risico's aan de voorkant moet er volgens het ontwerp als volgt uit gaan zien (zie figuur 7.9). Zie figuur 7.10 om te zien hoe deze tabel er huidig, in de oude navigator uitziet.



7.9: Ontwerp tabel van de risico's. Dit moet nog geïmplementeerd worden.

Commented [JB1]: Wat doet dit Plaatje hier? Je mist een toelichting/uitleg waarom dit plaatje belangrijk is

Figuur 7.10: Huidige risk-tabel van de navigator business dimension

Commented [JB2]: Zet dit ontwerp en de code bij elkaar, nu is het onduidelijk wat bij wat hoort

Om ervoor te zorgen dat de oude situatie vervangen zou worden met de nieuwe tabel heb ik het één en ander moeten aanpassen. Allereerst heb ik gekeken naar de Angular-components die ik kon gebruiken om de nieuwe tabel weer te geven, zodat ik wist of er bestaande componenten waren of niet dat ik kon overnemen. (Zie bijlage 8: Angular analyse) Hierna heb ik naar de implementatie gekeken van deze componenten in andere gedeeltes van de applicatie en heb ik vervolgens geprobeerd om dit zelf te implementeren.

De tabel-weergave van figuur 7.9 is al een generieke Angular component die vaker terug komt in de applicatie. Dit is een door de developers geschreven 'GenericListManager' component. Dat betekent dat ik dit dus kon implementeren en hier mijn eigen data en velden aan kon geven. Het makkelijkste om hiermee te beginnen was om eerst de HTML te schrijven, omdat ik dit ook kon overnemen van de andere implementaties voor de tabelweergave, daarna eventueel de CSS en tot slot de TypeScript logica om de HTML weergave te laten communiceren in Angular. Allereerst werd er hier dus HTML code voor geschreven (zie figuur 7.12)

```
<!-- row 1 Risk-->
<div class="row">
    <div class="col-7 col-xl-7">
        <generic-list-manager [config]="riskConfig">
        </generic-list-manager>
    </div>
</div>
```

7.12: HTML-code voor het weergeven van het risicotabel

Op de afbeelding valt te zien dat de <generic-list-manager> selector wordt gebruikt en de property [config]=”riskConfig” mee wordt gegeven. Wat er in de <generic-list-manager> staat is beschreven in figuur 7.13

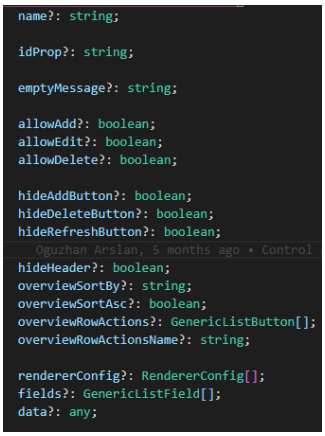
```
<div class="detail-page-card-body overflow-auto" *ngIf="scopedTableData && scopedTableData.length > 0">
    <cerrix-table [config]="config" [data]="scopedTableData" [idProp]="idProp" [activeRow]="activeRow"
        (activeRowChange)="activeRow = $event" (rowDblClick)="editRow(activeRow)"
        (sortingChanged)="sortingChanged()" *ngIf="tableData" [rendererConfig]="config.rendererConfig"
        [NoPadding]="true" [NoBorder]="true" [Centered]="true"></cerrix-table>

    <div class="custom-pagination" (click)="toggleScopedTableData()" *ngIf="scopedToggleAllowed">
        View {{scopedTableExpanded ? 'last ' + this.config.limitViewTo : 'all ' + this.config._visibleRowCount}}
    </div>
</div>
</div>
```

7.13: Snapshot van de HTML van de <generic-list-manager>.

Hier valt te zien dat deze generic-list-component een <(cerrix)-table> en dus een tabel-view weergeeft, met de data uit de **verkregen [config].** Dit zorgt ervoor dat het ontwerp van figuur 7.9 visueel wordt weergeven. Ik moest er wel voor zorgen dat ik de juiste riskdata en de juiste te weergeven riskvelden meegaf aan de **[config] property,** zodat deze risico's weergeven zouden worden op het scherm.

Dit heb ik meegegeven aan de [config]=”riskConfig” van figuur 7.12, zodat de generic-list-manager de juiste velden en data bevat. Deze [config] is een 'GenericListConfig', wat uit de volgende properties bestaat (zie figuur 7.14)



Figuur 7.14: De properties van de GenericListConfig ([config]=”riskConfig”)

Hieruit valt dus op te merken dat de meegegeven riskConfig o.a. een name, idProp, fields (de kolommen in de tabelweergave) en data (de data die correspondeert aan de juiste kolommen) moest hebben om te werken.

Om dit te laten werken en te koppelen met mijn risicodata heb ik eerst geprobeerd om hier zelf uit te komen, hiervoor had ik gekeken naar de implementatie van deze GenericListConfig in de control-module, waar dezelfde tabellen ook worden weergeven. Deze implementatie probeerde ik over te nemen, alleen wou het niet gelijk lukken. Hier moest ik een beetje mee spelen en uiteindelijk had ik Martin gevraagd of hij de werking tussen de GenericListConfigs en de correspondentie met de riskdata kon uitleggen, omdat er op afbeelding 7.14 veel properties waren en ik niet zeker wist welke ik allemaal moest gebruiken/ implementeren. Hierna beschikte ik over genoeg kennis om dit verder, zelfstandig te kunnen implementeren. Zie figuur 7.15, waarop staat wat ik zelf mee moest geven aan mijn eigen implementatie van de risk config.

Figuur7.15: De [config]=”riskConfig” property dat is meegegeven aan de GenericListComponent op figuur 7.12 om te corresponderen met de tabelweergave.

Het creëren van de riskconfig heb ik dus uiteindelijk kunnen implementeren. Op de afbeelding valt dus te zien dat er een <GenericListConfig> gemaakt met de name Risks, de IdProp ‘Guid’ , om de ID gelijk te zetten aan de Guids (unieke identifiers) van de risks. Daarnaast heb ik de kolommen van de tabel gemaakt. Dit is dus het lijstje met fields[].

Zo is er een kolom Identifier, Name, Organization, Overal risk assessment en Risk appetite. Binnen deze fields zijn er nog properties om aan te duiden of de kolommen platte tekst zijn of dat ze een kleur moeten weergeven in de tabel (property **fieldType**) en worden ze middels de fieldname attribute gekoppeld aan de binnengekomen **data.** Deze **riskData** is de data die wordt opgehaald uit de **GetNavigatorRiskOverview()** data. (Zie figuur 7.17). Al deze properties (de Identifier, Guid, Risk Name etc.) zijn gekoppeld aan de **riskData** variabele. Hierdoor weet de riskConfig dus uit welke data hij deze headers aan moet koppelen.

Deze stappen moesten er gezet worden om de de generic-list-manager te laten werken. Hier ben ik dus achter gekomen door, door de applicatie heen te kijken naar implementaties van de generic-list-manager en dit over te nemen. Hier moest wel een beetje voor vallen en opstaan, omdat Angular een nieuwe taal, en TypeScript dus ook. Ik moest een paar tutorials opzoeken over de conventies die er gebruikt werden voor het maken van de genericlistcomponents. Met behulp van een korte uitleg van Martin over de samenhang van de componenten heb ik dit toch vervolgens zelfstandig weten te implementeren.

```
const riskData = this._ds.getRisks(this.selectedOrganizationId, this.SelectedProcessId).toPromise().then(x => {
    x.forEach(d => {
        riskNetScoreColorModel.push({ID: d["Net score"], Name: d["Net score"], Color: d["NetScoreColor"]});
        riskGrossScoreColorModel.push({ ID: d["Gross score"], Name: d["Gross score"], Color: d["GrossScoreColor"] });
        riskOverallRiskAssessmentColor.push({ ID: d["Overall risk assessment"], Name: d["Overall risk assessment"], Color: d["RiskAssessmentColor"] });
        riskAppetiteColor.push({ ID: d["Risk appetite"], Name: d["Risk appetite"], Color: d["RiskAppetiteAssessmentColor"] });
    });
    return x;
});
```

7.17: Snapshot van de riskData variabele.

Op de afbeelding is te zien dat de riskData uit de getRisks() functie komt, die gekoppeld is aan de GetNavigatorRiskOverview() functie vanuit de backend (van figuur 7.6). Verder worden er in deze functie nog een paar lijstjes gemaakt, zodat de juiste kleuren voor de netto en bruto score van een risico gekoppeld worden aan de voorkant. Door deze riskConfig te implementeren en te koppelen aan de <generic-list-manager> wat bestaande code is van CERRIX is het gelukt om de risicotabel van het ontwerp waar te maken en ziet dit er dus als volgt uit (zie figuur 7.17)

Risks

Risk identifier	Name	Organization	Owner	Gross score	Net score	Overall risk assessment	Risk appetite
00000675	Dossier w...	Front Offi...	Smit, Jan	6	3	Verbet...	
00000676	Verwerkin...	Front Offi...	Smit, Jan	9	6	Aanva...	
00000677	Simulatie ...	Front Offi...	Smit, Jan...	4	2		
00000679	Toetsing ...	Front Offi...	Smit, Jan	6	3		
00000680	Maximali...	Front Offi...	Smit, Jan	8	4		

View all 466

7.17: Het risicotabel geïmplementeerd in Angular en versneld vanuit de achterkant, zodat dit tabel de data sneller inlaadt van voorheen.

§ 7.1.4 HET RESULTAAT VAN DE SERVER-SIDE FILTERING

Het resultaat hiervan was dat het inladen van de risico's nu een nieuw jasje heeft aan de voorkant. Deze code is herschreven van Ext.Net naar Angular. Dit heeft ervoor gezorgd dat de pagina voldoet aan de nieuwe huisstijl van CERRIX en oogt de nieuwe tabel veel beter. Deze implementatie is overigens ook gedaan voor het inladen van de controls, events, moi's, data processings en kri's, waarvoor ook een aparte tabel (view) en genericlistconfig implementatie voor gemaakt moest worden per module.

§ 7.2 TESTEN

Voor het volledige testrapport voor deze sprint verwijs ik u door naar bijlage 9: Testrapporten

Natuurlijk dienen er ook testen te worden geschreven en uitgevoerd om te garanderen of wat er gecodeerd is wel klopt. Dit kan zowel functioneel als niet-functioneel gedaan worden. Ik heb ervoor gekozen om zowel functionele, als niet-functionele testen uit te voeren, omdat ik vind dat ik een hele hoop heb veranderd en dit goed getest moest worden.

§ 7.2.1 FUNCTIONELE TESTEN

Voor de functionele tests zijn er in totaal 11 unit testen gemaakt en uitgevoerd, die controleerden of de data van alle zes modules correct gefilterd werden op organisatie, bedrijfsproces en werd er daarnaast getest of een gebruiker wel toegang had tot deze data als hij wel én niet beschikte over de rechten om deze risico's, controls, events etc. te lezen. Op tabel 7.2.1 zijn de test cases beschreven die er werden uitgevoerd voor het functioneel testen van de risico's en events. De keuze voor het uitvoeren van een unittest was gemaakt, omdat de modules los en afzonderlijk data ophalen (in een losse unit dus) en dit dus apart, per module getest kon worden.

Tabel 7.2.1: snapshot van de testcases voor de risks en events

# Test ID	# User story ID	Omschrijving	Verwachte uitvoer	Verwachte resultaat	Resultaat
T09	U10	Selecteer de Identifier van een risk met org = 9, bd = 1, **met** het recht om de risk te kunnen zien (RRV)	Identifier = 23	Identifier ='23	
T10	U10	Selecteer de Identifier van een risk met org = 9, bd = 1, **zonder** het recht om de risk te kunnen zien	Test failed	Test failed	
T11	U11	Selecteer de Identifier van een event met org = 3, bd = 7, **met** het recht om de event te kunnen zien (LERRV)	Identifier = 99	Identifier = 99	
T12	U11	Selecteer de Identifier van een event met org = 3, bd = 7, **zonder** het recht om de event te kunnen zien	Test failed	Test failed	

Deze testcases testen d.m.v. de uitvoering van kleine unittests. Zie figuur 7.2.2 om een voorbeeld te zien van een geschreven unit test.

Het resultaat van de unit tests was dat alle tests slagen. Ik had niet anders verwacht, aangezien de functies die er werden aangeroepen voor het ophalen en controleren op rechten van de te weergeven data al door de ontwikkelaars van CERRIX zelf geschreven waren en ik deze alleen per module had toegepast. Wat eventueel fout zou kunnen gaan was mijn eigen server-side filtering dat op deze data toegepast werd. Hiervoor werden deze unittests ook opgesteld.

```
[TestMethod]
public void TestControls()
{
    /*I'm fetching the control with organization: Cerrix (that is ID = 39)
      With the business dimension:  Internal communication (that is ID = 126)
      I'm expecting to see the control 'Rohit test control' with the Identifier: 2849.
      The user has the 'CMRV' (Control restricted viewer) with the same organizationID as the control.OrganizationID
      So it should work */

    var mockControl = new Mock<ORM_Risk_Control>{
        Identifier = 2849,
        ORM_Organization = 39,
        ORM_Business_Dimension = 126
    }.SaveChanges();

    var mockUser = new Mock<Auth_User>{
        Name = "TestUser",
        ORM_Organization = 39
    }.AddRight("CMRV").SaveChanges();


    var mockBusiness = new Mock<ControlBusiness>().Create();

    var expected = 2849;
    var actual = mockBusiness().GetAll(mockUser)
    .Where(x => x.ORM_Organization = 39
        && X.ORM_Business_Dimension = 126)
        .Select(a => a.Identifier);

    Assert.AreEqual(expected, actual);
}
```

Figuur 7.2.3 Unittest van T01 voor het laden van een control d.m.v. een org en bd filter. De uitvoer van deze test is correct

Deze unittest heb ik gemaakt door te kijken naar de testen die CERRIX al zelf had gemaakt. Hierdoor wist ik welke functies (Assert.AreEqual()) en welke mocking (het maken van testdata) ik kon gebruiken. Op de afbeelding werd er een (mock) control en user aangemaakt. De user had het recht om een control te kunnen zien. Vervolgens werd er getest of de controlbusiness de mock control zou herkennen op basis van de rollen en rechten én de gefilterde organization en business dimension id, dit bleek het geval te zijn, waardoor de unit test was geslaagd.

§ 7.2.2 NIET-FUNCTIONELE TESTEN

Naast de functionele testen, zijn er ook niet-functionele testen uitgevoerd, om te testen of de performance wel überhaupt verbeterd was en of de nieuwe UI wel correct was geïmplementeerd. Hiervoor was er een performance test en een UI-test uitgevoerd.

Performance test:

De performance test werd uitgevoerd in de vorm van een load test, omdat er gekeken werd of ik als tester kon uitvogelen of er een bottleneck in het parallel laden voorkwam. Mocht dit het geval zijn, dan zal dit komen door een incorrecte implementatie van de parallelizing, en kon ik dit nog tijdig oplossen. Zie tabel 7.2.3 voor de testcase.

Tabel 7.2.3: Testcase server-side performance voor het laden en filteren door de risico's in de nieuwe navigator.

T13: Test case: Open de Navigator en filter op organization: CERRIX en business dimension: Wareman Warenhuis. Verifieer dat voor het laden en filteren van de gehele pagina **minder dan 20 seconden duurt** (omdat dit in de oude situatie > 20 sec duurt). (Er worden 2049 risks, 457 events, 16 mois, 120 kri's, 3 data processings en 684 controls doorlopen)
Uitvoering: Vanaf het moment dat de Navigator geopend wordt, houdt Google Chrome dit middels de inspectietool bij en heb ik daarnaast ook een stopwatch. Ik kan zien welke functies er aangeroepen worden en hoelang de uitvoering hiervan duurt. Er zal gelet worden op de inspectietool, om te vergelijken of het herladen van het filter sneller zal verlopen.
Bevindingen attempt 1: Nadat er een selectie in het filter was gemaakt en er op Apply Configuration was geklikt duurde de eerste attempt in totaal: **15 seconden**. Dit is inderdaad een verbetering met de situatie hiervoor.
Bevindingen attempt 2: De tweede keer duurde het **15 seconden.** Dit kan liggen aan het feit dat er in de back-end al bepaalde objecten zijn initialized, waardoor het wat sneller laadt.
Bevindingen attempt 3: De derde keer duurde het laden wederom **15 seconden.**
Bevindingen attempt 4: De vierde keer duurde het laden wederom 15 seconden en het blijkt dat dit het minimaalste is wat ik kan bereiken qua performance.

Conclusie: De performance is door de server-side versnelling verbeterd met 5 seconden en wordt er in ieder geval niet langzamer op. De inspectietool laat geen bottleneck zien, omdat alle calls tegelijkertijd starten en ook niet vast zitten en/of op elkaar wachten.

UI-test:

Tot slot waren er UI-tests uitgevoerd, om te controleren of het ontwerp wel correct was geïmplementeerd. Dit hield in dat de juiste labels, tabellen, kolommen, knoppen etc. wel correct waren vormgegeven en werkten aan de voorkant. Dit was een relatief makkelijke test, omdat het nieuwe ontwerp niet zo groot/ complex was. Het is een kleine pagina met relatief weinig onderdelen (maar 6 tabellen). Hierom werd de UI-test handmatig uitgevoerd. Zie tabel 7.2.4 voor de testcases.

Tabel 7.2.4: UI test cases voor de omschrijving van de view van Ext.Net naar Angular

# Test ID	Omschrijving	Resultaat
T14	Controleer dat de risico tabel zichtbaar is incl. de juiste kolommen	
T15	Controleer dat de moi tabel zichtbaar is incl. de juiste kolommen	
T16	Controleer dat de kri tabel zichtbaar is incl. de juiste kolommen	
T17	Controleer dat de data processings zichtbaar is incl. de juiste kolommen	
T18	Controleer dat de event tabel zichtbaar is incl. de juiste kolommen	
T19	Controleer dat de control tabel zichtbaar is incl. de juiste kolommen	Verbetering: Kolommen: **Control Owner** en **In place** waren aanwezig, maar niet geïmplementeerd (lege uitvoer). Dit is verholpen.
T20	Controleer dat het filter de clear en apply buttons hebben	

§ 7.2.3 CONCLUSIE

Tijdens deze sprint zijn er verschillende testen uitgevoerd. Van functionele unit-testen tot niet-functionele UI en performance testen. Alle unit tests zijn uitstekend verlopen. Er is geen enkele gefaald. Dit had ik ook verwacht, omdat ik niet zelf de code heb geschreven voor het ophalen van de juiste data a.d.h.v. de rollen en rechten van een gebruiker, maar de echte developers dit zelf al hebben gedaan. Deze functie heb ik alleen geïmplementeerd en de server-side filtering heb ik hier ook op toegepast.

Ook waren er performance tests uitgevoerd. Er werd getimed hoelang het duurde voordat alle zes modules gefilterde uitvoer op het scherm lieten zien. Dit duurde 15 seconden om door een flink aantal rows heen te gaan. (2049 risks, 457 events, 16 mois, 120 kri's, 3 data processings en 684 controls). 15 seconden is natuurlijk nog best wel lang, maar is in ieder geval een verbetering met de oude situatie, waar het filteren en laden van alle data 20 seconden duurde.

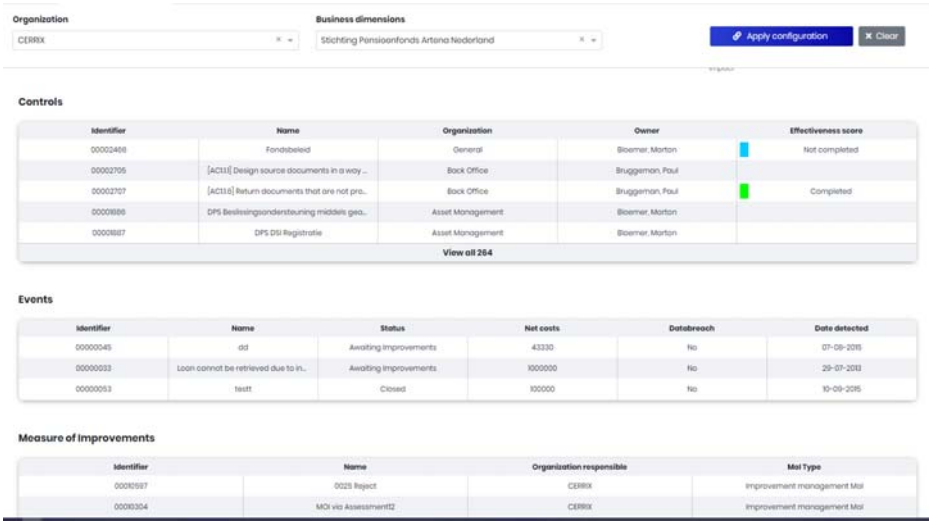
Tot slot waren er UI-tests uitgevoerd. Deze waren uitgevoerd om te controleren of wel alle onderdelen die in het ontwerp stonden waren opgenomen. Uit de UI tests bleken er 2 kolommen te zijn, die nog niet geïmplementeerd waren. Ik ben blij dat er een UI test is gedaan, anders zou dit pas op een later stadia ontdekt worden.

Al met al ben ik zeer tevreden met zowel de resultaten als de gekozen testen. Ik vind dat de tests genoeg coverage bieden over het uitgevoerde werk omdat er verschillende soorten testen worden uitgevoerd die allemaal wat anders testen en met een ander resultaat aankomen.

§ 7.3 SPRINT REVIEW

CERRIX heeft om de twee/ drie weken een 'design Friday' meeting. Dit is een meeting waarin de developers hun producten laten zien aan Paul en de consultants om zo nog extra feedback te ontvangen qua het ontwerp of betreft de functionaliteit.

Aan het einde van deze sprint zag de pagina er als volgt uit: (zie figuur 7.19)



7.19. Snapshot van een deel van de nieuwe Navigator Business Dimension. Op de afbeelding valt er te zien dat de controls, events en moi's al zijn geïmplementeerd (incl. server-side versnelling aan de back-end) en in een nieuw jasje gestopt

Commented [JB3]: Kan ik dat zien op dit plaatje?

Op de afbeelding zijn de risks, controls, moi's, data processings en kri's niet te zien, maar zijn ze wel geïmplementeerd.

Ik heb het zo geregeld dat deze design Fridays altijd vallen tegen het einde van mijn sprints, zo kan ik dit perfect combineren met mijn sprint reviews. Tijdens deze meeting heb ik mijn product laten zien aan Paul en de consultants en zal ik hieronder kort beschrijven wat het zij vonden van het increment:

Commented [JB4]: Heb je dat nu pas geregeld? Hierna heb je nog maar 1 sprint

- Het product ziet er heel netjes uit, het ontwerp komt echt overeen met het ontwerp wat er gemaakt is (dit geldt voor de implementatie van alle 7 modules naar Angular). Echter zijn er een klein aantal aan aanpassingen die doorgevoerd kunnen worden, zoals de ' apply configuration' button dat beter 'apply filter' kan heten en dat de lettertypen van de knoppen wat kleiner kunnen,.

- Het laden van de data duurt toch best wel lang. Het is wel versneld, het duurt nu 15 seconden ongeveer, maar dit moet nog sneller kunnen, omdat de rest van de CERRIX-applicatie client-side filtering gebruikt (wel met parallelle) en dat toch wel wat sneller is.

Verder vind ik zelf dat ik kwantitatief veel gedaan heb. Zo heb ik ervoor gezorgd dat de weergave van de zes modules is geïmplementeerd volgens het nieuwe ontwerp en dat de weergeven data, server-side gefilterd wordt. Echter was de kwaliteit van de code minder. Dit komt omdat ik geen design patterns had gebruikt, waardoor er super veel van dezelfde code gekopieerd en geplakt werd.

Dit kwam omdat ik eerst wou kijken of ik de GenericListComponents en de rest van de Angular kant wel correct kon implementeren. Hier was ik de gehele sprint mee bezig, omdat ik dit voor 7 modules moest doen en deze code bijna voor elke module hetzelfde was. Pas nadat ik aan de voorkant duidelijke uitvoer had wou ik me pas focussen op de implentatie van design patterns.

Het blijkt nu dat de server-side filtering toch aan de trage kant is, en tijdens sprint 3 zal ik dus eerst de focus leggen op het corrigeren van mijn code (kleine visuele aanpassingen zoals hierboven genoemd en de implementatie van design patterns) alvorens ik de user stories gepland voor sprint 3 oppak.

§ 7.4 SPRINT RETROSPECTIVE

Wederom hebben Martin en ik aan tafel gezeten om te bespreken wat er goed en minder goed ging tijdens deze sprint. Ik vind dat het proces matig is verlopen, omdat alle user stories wel correct zijn uitgevoerd en geïmplementeerd, maar alleen missen de ze nodige kwaliteit. De code mistte nog design patterns, omdat ik alles eerst aan de praat wou krijgen voordat ik hierop zou focussen. Dit bleek naderhand toch niet slim, omdat ik nu extra werk heb wat ik moet corrigeren tijdens sprint 3. Als ik in het begin gedacht had aan de toepassing van design patterns, dan hoefde dit niet meegenomen te worden naar de volgende en kon er tijdens de volgende sprint direct met de toegewezen user stories begonnen worden. Dit is iets waar ik in het vervolg rekening mee zal houden. Design patterns zijn er niet voor niets, en moeten dus liever zo vroeg mogelijk geïmplementeerd worden.

Commented [JB5]: Zin klopt niet helemaal:
Ik vind het niet goed, omdat ik van alles goed heb gedaan, maar toch iets slecht.
Ik zou omdat vervangen door ondanks
Ik vind het niet goed, ondanks dat ik van alles goed heb gedaan, is iets toch slecht

Daarnaast is er tijdens deze sprint voor het eerst gewerkt met Angular en TypeScript. Dit was een hele nieuwe taal en hier moest de nodige guides voor opgezocht worden op het internet om te begrijpen welke conventies er waren. Hierdoor weet ik nu hoe TypeScript in elkaar zit en hoe ik dit kan gebruiken voor het maken van mijn eigen classes en componenten. Ook heb ik een beter beeld hoe de genericlistcomponenten worden aangemaakt en gebruikt door de applicatie heen. In het begin snapte ik dit niet goed, maar m.b.v. de uitleg van Martin kon ik de rest zelfstandig nog uit vogelen.

Commented [JB6]: Dit zie ik niet terugkomen in je verslag

HOOFDSTUK 8: SPRINT 3 - HET IMPLEMENTEREN VAN DE CLIENT-SIDE VERSNELLING

Tijdens deze sprint zullen er allereerst de openstaande taken van de vorige sprint opgepakt worden. Zo zal de kwaliteit van de opgeleverde code van sprint 2 eerst verbeterd worden middels twee design patterns. Daarna zal de implementatie van de client-side filtering toegepast worden in de navigator. Vervolgens zal dit getest worden, om te kijken of deze implementatie wel sneller is dan de server-side filtering en zal het uitgevoerde werk onderworpen worden aan een sprint review en retrospective. De uitgevoerde user stories zijn te zien op tabel 8. Wederom zal er in dit hoofdstuk weer één user story behandeld worden, dit zal U17 zijn

Tabel 8: De user stories die uitgevoerd zijn deze sprint

# User Story ID	Priority (H – L)	User story	Est. time
U14	H	As < an overviewer >I want< to see the Risk Matrix>, <so that I can see further details about the Risks >	8
U15	H	As < an overviewer > I want < to see my Controls according to the **client-side filter** - when ¡ apply my configuration> so <that I can get an overview>	12
U16	H	As < an overviewer > I want < to see my Events according to the **client-side filter** - when ¡ apply my configuration> so <that I can get an overview>	12
U17	H	As < an overviewer > I want < to see my Risks according to the **client-side filter** - when ¡ apply my configuration> so <that I can get an overview>	12
U18	H	As < an overviewer > I want < to see my MoIs according to the **client-side filter** - when ¡ apply my configuration> so <that I can get an overview>	12
U19	H	As < an overviewer > I want < to see my Data Processings according to the **client-side filter** - when ¡ apply my configuration> so <that I can get an overview>	12
U20	H	As < an overviewer > I want < to see my KRIs according to the **client-side filter** - when ¡ apply my configuration> so <that I can get an overview>	12

§ 8.1 ONTWERP CODE VERBETEREN

§ 8.1.1 DESIGN PATTERNS

Tijdens sprint 2 waren er geen design patterns gebruikt, omdat ik eerst zeker wou zijn of de functionaliteit wel klopte. Pas daarna zou ik de code netter omschrijven. Nu ik zeker weet dat iedereen tot zover tevreden is met de navigator (qua design) zal ik design patterns selecteren om de code netter te maken, voordat de client-side filtering toegepast kon worden.

§ 8.1.2 WAT ZULLEN DE DESIGN PATTERNS ONDERSTEUNEN

Om de juiste design pattern te selecteren, moet er eerst vastgesteld worden wat de design pattern moet verbeteren. Welke functionaliteit moet de pattern kunnen ondersteunen?

Om te beginnen zijn er 6 'GenericListComponents' objecten gemaakt. Hiervoor is de code telkens gekopieerd en geplakt, terwijl er toch veel attributen overeenkomen per GenericListComponent. Dit zorgt ervoor dat de code voor het weergeven van de 6 tabellen nu grofweg 300 regels aan onnodige code is (zie figuur 8.1). Dit kan veel simpeler gedaan worden door een pattern toe te passen zodat de code veel netter wordt en minder regels bevat. Maar welke design pattern past er hier het beste bij?

Figuur 8.1: GenericListConfigs die 6x gekopieerd en geplakt worden (per module). Terwijl er telkens veel van dezelfde attributen gekopieerd worden en dezelfde waarde hebben

De velden die niet gekopieerd en geplakt werden zijn de lijstjes met de fields, omdat hierin de tabel headers zaten die verschilden per module. Het gebruik van een design pattern kon dit oplossen, zodat er veel minder code geschreven hoefde te worden.

De pattern die dit op kan lossen is de **factory-method pattern**. Dit heb ik gekozen, omdat de factory-method pattern o.a. de volgende oplossingen bood (naar Design Factory Method Design Pattern UML. (2016, 1 oktober)):

De factory-method pattern zorgt ervoor dat er gemakkelijk nieuwe objecten gecreëerd kunnen worden zonder rekening te houden met het type object, omdat deze pattern checkt op een enum, waardoor er per enumtype een aparte initialisatie wordt toegepast op het object.

Dit zorgt ervoor dat ik minder code hoef te schrijven voor het initialiseren van de GenericListsConfigs (en dus niet zoals figuur 8.1). Dit komt omdat elke config wel ongeveer dezelfde waardes bevat. Zo heeft elke config bijvoorbeeld een allowAdd, allowEdit, allowDelete = false waarde. Dit kan dus mooi in 1 generieke methode gezet worden die een 'lege' GenericListsConfigs returned, zodat dit extra regels scheelt. Dit kan ik middels de factory-method pattern implementeren. Deze generieke configs zullen zo per module (risk/event/control) aangemaakt worden. Wat deze configs dan nog missen zijn de fields en de hieraan gekoppelde data. (zie figuur 8.1 voor het lijstje met de fields en data)

Nadat deze generieke configs zijn gemaakt en in principe alleen een allowCreate, allowEdit, allowDelete, name, idProp en limitViewTo property hebben en dus geen data of fields, wordt het tijd om ze toe te kennen. Om dit gemakkelijk te doen zal er gebruik gemaakt worden van een **builder pattern.**

Omdat de fields uit verschillende soorten fieldTypes bestaan (zo is er een **GenericListFieldType.Text** om tekst weer te geven (zie figuur 8.1) of een **GenericListFieldType.Color** om een kleur weer te geven), kan de builder pattern helpen met het toekennen van deze verschillende soorten fields aan een (lege)config. Hiermee worden de fields als het waren los 'gebouwd' aan de config.

Daarnaast bood de builder pattern een nette manier om de fields te chainen aan de lege GenericListConfig, omdat een risk veel velden heeft, ongeveer 8, en deze door de builder pattern mooi aan elkaar gechained kunnen worden tijdens de initialisatie en niet ergens los toegekend hoeft te worden aan deze config in een andere methode.

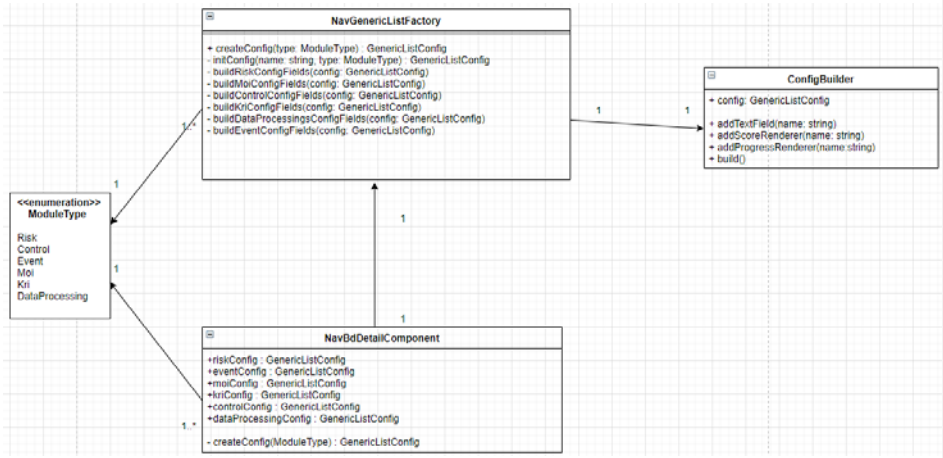
De builder pattern zorgt dus ervoor dat ik gemakkelijk deze verschillende soorten velden, inclusief de prettyName, fieldName en fieldType kan meegeven aan de lege genericListConfig die door de factory-method wordt geïnstantieerd. *(naar Builder pattern).*

§ 8.1.3 HET ONTWERP VAN DE DESIGN PATTERNS

Nu er bekend is dat zowel de factory-method als de builder-pattern zal worden toegepast, kon er hier een ontwerp voor gemaakt worden. Zie figur 8.2 voor het UML diagram.

Op de afbeelding is te zien dat de NavBdDetailComponent klasse, waar de view in zit de NavGenericListFactory aanroept. Dit is de klasse waar de factory-method in is toegepast. Deze gebruikt de enum ModuleType om te kijken welke soort module er geïnstantieerd moet worden, zodat de factory de juiste config kan maken en m.b.v. de ConfigBuilder klasse de juiste velden hieraan op kan bouwen.

Op de afbeelding is te zien dat de NavBdDetailComponent klasse, waar de view in zit de NavGenericListFactory aanroept. Dit is de klasse waar de factory-method in is toegepast. Deze gebruikt de enum ModuleType om te kijken welke soort module er geïnstantieerd moet worden, zodat de factory de juiste config kan maken en m.b.v. de ConfigBuilder klasse de juiste velden hieraan op kan bouwen.



Figuur 8.2 UML diagram voor de factory-method en de builder pattern

§ 8.2.5 IMPLEMENTATIE VAN DE PATTERNS

Door Design Factory Method Design Pattern UML. (2016, 1 oktober) te bestuderen en het UML te volgen kon ik uiteindelijk de factory method pattern implementeren. Dit was niet makkelijk, omdat ik nog nooit een pattern had geïmplementeerd, maar na een aantal een aantal voorbeelden op te hebben gezocht was dit uiteindelijk toch gelukt.

In mijn UML diagram had ik bepaald dat de NavBdDetailComponent alle losse configs zou creëren, omdat dit het component was waar de details (en dus tabellen) zou weergeven. De zes losse configs werden hier aangemaakt, en ze werden getypeerd op basis van de enum ModuleType. De creatie van deze configs gebeurde door in de factory. (Zie figuur 8.3). De volgende stappen zullen dus gezet worden in de factory klasse.

In de factory klasse is er eerst een case/switch om te bepalen welke type config er gemaakt moet worden (op basis van de ModuleType). Hier wordt er eerst een een lege GenericListConfig gemaakt, waar alleen de naam van de config in zit, daarna stuurt de factory de config verder naar de aparte methode per module. (zie figuur 8.4).

De switch/case roept een aparte methode aan om de configs verder te initialiseren (zie figuur 8.5). Aan deze config wordt er middels de implementatie van de builder pattern (zie figuur 8.6) fields en data toegevoegd hieraan. Hierdoor is de GenericListConfig nu gevuld met data en kan het door de NavBdDetailComponent klasse gebruikt worden om data te weergeven op het scherm.

Het resultaat hiervan is dat er nu mooie patterns worden toegepast om de initialisatie en de velden van de GenericListConfig, wat zes keer gemaakt wordt in de NavBdDetailComponent klasse op te vangen. Hierdoor is de code veel kleiner en netter. Deze implementatie scheelt mij nu 300 regels aan code dat voorheen geschreven werd voor het maken van de zes losse configs. Nu het overige werk is gedaan van sprint 2, kan er nu gefocust worden op de client-side filtering.

```
ngOnInit() {
    this.riskConfig = this.navFactory.createConfig(ModuleType.ORM);
    this.controlConfig = this.navFactory.createConfig(ModuleType.MC);
    this.eventConfig = this.navFactory.createConfig(ModuleType.LER);
    this.moiConfig = this.navFactory.createConfig(ModuleType.BIM);
    this.kriConfig = this.navFactory.createConfig(ModuleType.KRI);
    this.dataProcessingsConfig = this.navFactory.createConfig(ModuleType.DM);
}
```

Figuur 8.3: Creatie van alle 6configs.

Deze functie zit in de NavBdDetailComponent en roept de factory klasse aan en heeft een ModuleType als parameter om onderscheid te maken tussen de configs. Hier worden de zes configs los aangemaakt d.m.v. de Factory en Builder classes.

```
createConfig(moduleType: ModuleType): GenericListConfig {
    let config;
    switch (moduleType) {
        case ModuleType.ORM:
            config = this.initConfig(ModuleType.ORMDescription, moduleType);
            this.buildRiskConfigFields(config);
            break;
        case ModuleType.MC:
            config = this.initConfig(ModuleType.MCDescription, moduleType);
            this.buildControlConfigFields(config);
            break;
        case ModuleType.LER:
            config = this.initConfig(ModuleType.LERDescription, moduleType);
            this.buildEventConfigFields(config);
            break;
        case ModuleType.BIM:
            config = this.initConfig(ModuleType.BIMDescription, moduleType);
            this.setMoiConfigFields(config);
            break;
        case ModuleType.KRI:
            config = this.initConfig(ModuleType.KRIDescription, moduleType, "guid");
            this.setKriConfigFields(config);
            break;
        case ModuleType.DM:
            config = this.initConfig(ModuleType.DMDescription, moduleType);
            this.setDataProcessingConfigFields(config);
            break;
    }
    return config;
}
```

8.4: switch/case in de Factory klasse om onderscheid te maken tussen de zes verschillende soorten configs

Hier worden er aparte methodes aangeroepen om de velden van de verschillende configs verder te vullen en initialiseren.

```
private buildRiskConfigFields(config: GenericListConfig) {
    config =
        new ConfigBuilder(config)
            .addTextField("Identifier", "Risk identifier")
            .addTextField("Risk name", "Name")
            .addTextField("Organization")
            .addTextField("Owner")
            .addTextField("GrossScoreColor")
            .addTextField("Gross score")
            .addTextField("Net score")
            .addTextField("NetScoreColor")
            .addTextField("Overall risk assessment")
            .addTextField("RiskAssessmentColor")
            .addTextField("Risk appetite")
            .addTextField("RiskAppetiteAssessmentColor")
            .addScoreRenderer("Gross score", "GrossScoreColor", true)
            .addScoreRenderer("Net score", "NetScoreColor", true)
            .addScoreRenderer("Overall risk assessment", "RiskAssessmentColor", true)
            .addScoreRenderer("Risk appetite", "RiskAppetiteAssessmentColor", true)
            .build();
}
```

8.5: In deze functie die vanuit de switch/case aangeroepen wordt, worden er fields toegekend aan de lege RiskConfig. Dit gebeurd via de Builder.

Op de afbeelding valt er te zien dat er aan de config velden worden gechained (aan elkaar worden gekoppeld). Dit is mogelijk gemaakt door de builder, wat een flink aantal regels in code scheelt.

```
addTextField(fieldName: string, prettyName?: string) {
    this.config.fields.push({ fieldName, prettyName, fieldType: GenericListFieldType.Text });
    return this;
}

addScoreRenderer(textColumn: string, actionColumn?: string, hideActionColumn?: boolean) {
    this.config.rendererConfig.push({ textColumn, actionColumn, hideActionColumn, type: RendererType.Score });
    return this;
}
```

8.6: Dit is de code dat er in de Builder klasse staat

Hierdoor kan elke soort config zijn eigen velden toekennen aan de fields van de lege configs en wordt de config teruggestuurd. Dit scheelt veel regels code, doordat het nu generiek gemaakt is.

§ 8.3 DE CLIENT-SIDE FILTERING

Het verschil tussen de server-side en client-side filtering is dat er via de server-side filtering, per keer dat er gefilterd wordt een nieuwe aanvraag naar de servers gaat om de juiste data op te halen. Bij de client-side filtering, hoort alle data (van alle zes modules) eerst opgehaald te worden in de browser van de gebruiker, en kan er op deze data die maar 1x opgehaald hoeft te worden, gefilterd worden.

Hiervoor moesten de risico's dus eerst opgehaald bij het laden van de navigator, en de functie die ikhiervoor heb gebruikt is dezelfde functie dat gebruikt wordt om de risico's op te halen vanuit de risk-module (net als sprint 2). Deze riskdata wordt er vervolgens meegegeven aan de genericlistconfig data van de risks. (Zie figuur 8.8).

```
this.riskData = this._riskDS.getList().toPromise();
```

8.8: De GetList() functie die de riskData vult

Deze functie bestaat al, zo haalt de GetList() ook de risico's op uit de riskmodule, en heb ik deze functie herbruikt. De functie haalt overigens de risico's op die de gebruiker mag zien volgens zijn rollen en rechten.

In de riskData zitten er dus een paar honderd risico's, die de gebruiker mag zien. Nadat er een filter configuratie is toegepast, gaat de client deze risico's filteren op organisatie en business dimension. Dit heb ik als volgt gedaan: (zie figuur 8.8)

```
filterData(config: GenericListConfig, data: any[]) {
    const filteredData = [];
    const filterBdsFunc = z => z > 0 && this.bdsDown.indexOf(z) >= 0;
    switch (config) {

        case this.kriConfig: {
            data.forEach(y => {
                const organizationsArray = this.toNumberArray(y.organizationIds);

                if (this.businessDimensionId > 0) {
                    const businessDimensionsArray = this.toNumberArray(y.businessDimensionIds);
                    if (organizationsArray.some(b => b > 0 && this.organizationsDown.indexOf(b) >= 0)
                        && businessDimensionsArray.some(filterBdsFunc)) {
                        filteredData.push(y);
                    }
                } else {
                    if (organizationsArray.some(b => b > 0 && this.organizationsDown.indexOf(b) >= 0)) {
                        filteredData.push(y);
                    }
                }
            });
            break;
        }

        default: {
            data.forEach(y => {
                if (this.businessDimensionId > 0) {
                    const businessDimensionsArray = this.toNumberArray(y.BusinessDimensionIDs);
                    if (this.organizationsDown.indexOf(y.OrganizationID) >= 0
                        && businessDimensionsArray.some(filterBdsFunc)) {
                        filteredData.push(y);
                    }
                } else {
                    if (this.organizationsDown.indexOf(y.OrganizationID) >= 0) {
                        filteredData.push(y);
                    }
                }
            });
            break;
        }
    }
    config.data = of(filteredData).toPromise();
    if (config.reload) {
        config.reload();
    }
```

8.8: De client-side filtering methode

De filter verwacht als parameter een config, in dit geval onze RiskConfig en onze RiskData. De functie komt een switch/case binnen. De RiskConfig is in dit geval een KriConfig, waardoor de functie naar de default gaat. Hier werd de data eerst gefilterd op basis van de gekozen businessDimensionIds en vervolgens op organizationId. De gefilterde items wordt vervolgens weer meegegeven aan de riskConfig, waarna hij deze methode verlaat en de config op het scherm herladen wordt.

Dit is een generieke filter functie die elke soort config kan gebruiken. Hierdoor hoeven er geen aparte asynchrone methodes meer gebruikt te worden in de navigatorBusiness.cs, omdat er niet meer server-side gefilterd wordt. De volgende stap was om dit te testen

Het was niet moeilijk om de deze functie te bedenken, omdat het letterlijk hetzelfde doet als de server-side filter functie. Het vergelijkt de gefilterde organisaties en processen met die van de data die (nu client-side) is opgeslagen. Dit lukte natuurlijk niet 1,2,3, omdat ik veel fouten ging maken in TypeScript voor de vergelijkingen tussen de data, maar uiteindelijk is dit toch gelukt en werkt de client-side filter functie.

§ 8.4 TESTEN

Uiteraard zijn er in deze sprint ook een aantal testen uitgevoerd. Wederom waren er functionele en niet-functionele testen uitgevoerd. Hier zal ik de uitvoer geven van de niet-functionele performance test. Voor de overige unit tests en UI tests verwijs ik u door naar bijlage 9: het testrapport.

Om de performance te testen van de client-side filtering is er een test case gemaakt (zie tabel 8.4.1). Hiervoor werd er getimed op het moment dat er een organisatie en bedrijfsproces geselecteerd was en er op het filter knop werd gedrukt. Hierdoor duurde het eerste attempt maar 3 seconden! Dit is een super flinke verbetering. De tweede, derde en vierde keer duurde dit 1.95 seconden, wat nog mooier is. De eerste keer duurde het filteren van de data wat langer, omdat het nog bezig was met het opvragen van de data aan de database en werd dit nog opgeslagen aan de client-side. De keren hierna was de data al opgeslagen en hoefde dit dus alleen nog maar gefilterd te worden.

Tabel 8.4.1: Performance testcase voor het client-side filteren

T14: Test case: Open de Navigator en filter op organization: CERRIX en business dimension: Wareman Warenhuis. Verifieer dat voor het laden en filteren van de gehele pagina **minder dan 15 seconden duurt** (omdat dit in T13 het geval is). (Er worden 2049 risks, 457 events, 16 mois, 120 kri's, 3 data processings en 684 controls doorlopen)
Uitvoering: Vanaf het moment dat de Navigator geopend wordt, houdt Google Chrome dit middels de inspectietool bij en heb ik daarnaast ook een stopwatch. Ik kan zien welke functies er aangeroepen worden en hoelang de uitvoering hiervan duurt. Er zal gelet worden op de inspectietool, om te vergelijken of het herladen van het filter sneller zal verlopen.
Bevindingen attempt 1: Nadat er een selectie in het filter was gemaakt en er op Apply Configuration was geklikt duurde de eerste attempt in totaal: **3 seconden.** Dit is prachtig om te zien, omdat de data nu even snel laad als de rest van de applicatie.
Bevindingen attempt 2: De tweede keer duurde het **1.95 seconden.** Dit is het geval, omat de data van alle modules al aan de client-side zijn ingeladen. Deze data wordt alleen opnieuw gefilterd, wat dus sneller is dan attempt 1.
Bevindingen attempt 3: De derde keer duurde het laden wederom **1.95.**

Bevindingen attempt 4: De vierde keer duurde het laden en filteren wederom 1.95 en het lijkt dat dit het minimaalste aantal seconden is wat ik kan bereiken.
Conclusie: De performance is door de client-side nog veel beter verbeterd dan door de implementatie van de server-side filtering en met een zeer grote marge. Zo duurde het hiervoor 15 seconden, en nu maar 1.95 seconden minimaal, nadat alle data na het eerste attempt al in de client was geladen. Dit is een prachtige verbetering en laat zien dat de performance echt is verbeterd.

§ 8.5 SPRINT REVIEW

Het resultaat van deze sprint was dat ik de kwaliteit van de code heb verbeterd, door de factory-method en de builder pattern te implementeren. Dit heeft ervoor gezorgd dat ik niet meer nodeloos code kopieer en plak, zoals ik bij sprint 2 deed voor het maken van de GenericListConfigs. Hierdoor zijn mijn klasses 300 regels codes kleiner, en is de code aangenaam om te lezen doordat er nu duidelijke structuur aanwezig is, namelijk door de twee design patterns.

Daarnaast heb ik de client-side filtering toegepast. Het was makkelijker om dit te implementeren dan het maken van de server-side filtering, omdat ik maar één functie nodig had voor de client-side filtering. Voor de server-side filtering had ik zes aparte methodes nodig, omdat er zes aparte modules zijn met elk hun eigen implementatie en vertaling voor de filtering. Ik ben zeer onder de indruk dat de client-side filtering maar maximaal 3 seconden in beslag nam. Dit is een aanzienlijke verbetering met de 20 seconden die het hiervoor duurde (de oude situatie). Ik ben tevreden met het uitgevoerde werk deze sprint.

Dit werk heb ik laten zien tijdens de design-Friday aan Paul en Martin. Zij zijn ook tevreden met de aantal seconden dat de filtering duurt. Paul zei dat de navigator met deze snelheid weer verkoopbaar is. Dit kan dus allemaal afgestreept worden van de backlog en dit betekent dat de sprint correct is afgerond.

§ 8.6 SPRINT RETROSPECTIVE

Het proces van sprint 3 is goed ervaren. Alle user stories die gepland stonden zijn afgerond, het vergde wel enige inspanning om af te krijgen, omdat er taken van de vorige sprint meegenomen werden en eerst uitgevoerd moesten worden voordat er begonnen werd met de taken van sprint 3.

Er zijn twee design patterns toegepast, dit heb ik voor de eerste keer gedaan, waardoor het veel tijd kostte om dit uit te zoeken. Martin bood de nodige ondersteuning voor het begrijpen van de builder pattern, zodat dit toegepast kon worden. Nu is er inzicht over het gemak van de implementatie van patterns in zijn geheel en de noodzaak van patterns om de kwaliteit van code te verbeteren.

HOOFDSTUK 9 : EVALUATIE

In het volgende hoofdstuk volgt de evaluatie van zowel het product als het proces. Daarnaast zal er een terugkoppeling met de beroepstaken volgen, om aan te tonen dat deze correct en professioneel zijn uitgevoerd de afgelopen 17 weken.

§ 9.1 PRODUCTEVALUATIE

Het uiteindelijke eindproduct bestond uit een drietal delen: de implementatie van de client-side filtering voor het laden van de navigatordata, de versnelling voor het ophalen van deze data, weergegeven in een nieuw grafisch ontwerp in Angular. Tijdens deze productevaluatie zullen deze drie producten afzonderlijk van elkaar geëvalueerd worden.

Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands), Pattern: Clear

Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands), Pattern: Clear

Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands), Pattern: Clear

§9.1.1 ONTWERPEVALUATIE

Ik ben tevreden over het nieuwe ontwerp, omdat ik hier mijn eigen input ook terug in zie, namelijk de weer te geven kolommen dat ik analyseerde en uitkoos. Daarnaast zag het nieuwe ontwerp er veel mooier uit dan de oude situatie. De pagina leek weer te behoren in CERRIX. Waar ik het ontwerp ook nog wát mooier door vind, is dat ik ook actief mee heb kunnen discussiëren tijdens de ontwerpmeeting. Hierdoor kon ik ook instemmen als ik vond dat bepaalde gekozen lettertypes te groot of te klein waren, hetzelfde gold voor de selectie uit knopjes etc. Daarnaast was het mijn eigen idee om de kri's mee te nemen in het nieuwe ontwerp. Paul zei dat hij hier zelf nog niet eens aan gedacht had en dat dus zeker meegenomen zou worden, wat uiteindelijk ook gedaan is. Om deze redenen ben ik tevreden over het nieuwe ontwerp.

Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands), Pattern: Clear

§9.1.2 VERSNELLINGSEVALUATIE VOOR HET LADEN VAN DATA

Wat ik nog meer had opgeleverd was de implementatie voor de versnelling voor het laden van de data uit de zes modules. Hiervoor heb parallelle functies geschreven die de data per module afzonderlijk van elkaar ophaalt, waardoor het ophalen van de data in ieder geval sneller verliep dan voorheen. Dit was middels de unit tests getest en de uitvoer was dat het laden van de data inderdaad sneller verliep met het server-side filteren van de data, om zo een vergelijking te maken tussen de oude situatie, waarbij de data server-side gefilterd werd, maar sequentieel werd opgehaald. Door deze versnelling te implementeren en te testen bleek, dat het laden en filteren met 5 seconden was versneld (op basis van mijn database). Het voelde goed om daadwerkelijk te kunnen zorgen voor versnelling, en hier ben ik tevreden over.

Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands), Pattern: Clear

§9.1.3 CLIENT-SIDE FILTERINGSEVALUATIE

Waar ik het meest trots op ben is mijn implementatie van de client-side filtering. Het resultaat van deze implementatie was dat het laden van en filteren van de data nu tussen de 2-3 seconden zit. Dit is een prachtige versnelling, omdat dit vroeger 20 seconden duurde. Het implementeren van de client-side filtering was dus het beste wat ik op kon leveren in dit project, omdat dit ervoor heeft gezorgd dat CERRIX de navigator weer kon verkopen aan haar klanten. Hierdoor werd mijn uitproduct meegenomen naar de volgende release van de applicatie. Natuurlijk heb ik veel van deze implementatie geleerd, ik heb de code m.b.v. twee design patterns geschreven en hierdoor weet ik gelijk hoe makkelijk de implementatie en het nut van het gebruik van patterns is, omdat ik hierdoor veel minder code heb hoeven schrijven. Het verschil tussen sprint 2 en 3 tussen de code bedraagt ongeveer 300 regels, en hierdoor is de kwaliteit van de code verbeterd.

Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands), Pattern: Clear

§ 9.2 PROCESEVALUATIE

De gekozen ontwikkelmethode was SCRUM. Uiteindelijk bleek dit de handigste methode te zijn die ik kon gebruiken, omdat ik door de korte sprints genoeg ruimte had om feedback te krijgen op mijn implementaties. Tijdens sprint 2 werd er verteld dat er nog grafisch wat kleine aanpassingen gedaan moesten worden, en deze waren dus dan ook meegenomen naar de volgende sprint. Daarnaast moest de code m.b.v. design patterns geschreven worden en werd dit werk ook meegenomen naar de volgende sprint (sprint 3). In sprint 3 werden deze dingen eerst opgepakt, zodat deze wisselende wensen tijdig waren opgevangen, dit was het beste voordeel wat SCRUM mij aan kon bieden en hier heb ik dus ook gebruik van gemaakt.

Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands), Pattern: Clear

Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands), Pattern: Clear

Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands), Pattern: Clear

De volgende keer zal ik wel letten op het tijdig afmaken van de code. Omdat ik tijdens sprint 2 niet gelet had op de kwaliteit van de code, en alleen op de werking moest ik dit meenemen naar sprint 3, wat voor extra druk zorgde. De volgende keer zal ik gelijk denken aan het gebruik van bepaalde patterns, alvorens ik begin met coderen. Al met al was SCRUM dus de juiste methode, omdat er eventuele kleine wisselingen in requirements voorkwamen en ze dus tijdig opgevangen werden.

§ 9.3 EVALUATIE BEROEPSTAKEN

Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands), Pattern: Clear

Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands), Pattern: Clear

Formatted: Font: (Default) +Headings (Calibri Light), Dutch (Netherlands), Pattern: Clear

A-1: Analyseren probleemdomein

Het probleemdomein is volledig geanalyseerd. Dit is gedaan door een analyse te maken van het probleem dat al bij CERRIX bekend was, er werd er code technisch gekeken waar het probleem voor de performance aan lag. Daarnaast is er een interview afgenomen om vast te stellen wat klanten het probleem van de navigator vonden en was dit overlegd met het team. Vervolgens vloeiden er hier requirements uit. Dit is aangetoond in bijlage 4: het requirements document, in hoofdstuk 5: het opzetten van SCRUM en in hoofdstuk 6: sprint 1.

B-4: Gemotiveerd selecteren van ICT gerelateerde oplossingen

Er is een analyse gemaakt om te kijken of Angular wel de juiste oplossingen bood om de opdracht in te vervullen. Hiervoor werden er drie van de grootste javascript frameworks met elkaar vergeleken en was de conclusie hiervan van Angular inderdaad de beste framework was om het project in uit te voeren. Daarnaast is er ook een angular-componenten analyse uitgevoerd, om te kijken welke bestaande componenten er hergebruikt konden worden bij het implementeren van het nieuwe ontwerp. Dit is aangetoond in bijlage 8: Angular analyse en in hoofdstuk 6: sprint 1.

C-6: Ontwerpen software

De software dat geschreven was, was eerst zorgvuldig ontworpen. Hiervoor zijn er UML-diagrammen en sequentiediagrammen gemaakt en zijn er design patterns toegepast voor het ontwerpen van de software. Dit kunt u terugvinden in bijlage 7: het ontwerpdocument en in hoofdstukken 6, 7 en 8.

D-14: Realiseren software

Er is code geschreven om het laden van de data te versnellen, de filtering te verbeteren en om het nieuwe ontwerp te implementeren in Angular. Dit is aangetoond in hoofdstukken 7 en 8.

D-15: Testen

Om er zeker van te zijn dat mijn implementatie van code klopt, is er uitvoerig getest. Er zijn unit testen, performance testen en UI testen uitgevoerd. Dit kunt u terug vinden in bijlage 9: testrapporten en in hoofdstuk 7 en 8.

G-C: Kritisch en methodisch werken

Natuurlijk zijn alle keuzes die ik gemaakt heb afgewogen. Ik moet er achter kunnen staan en dit kunnen aantonen. Mijn kritische motivatie kunt u terug vinden in vrijwel alle hoofdstukken. Daarnaast is er methodisch, met SCRUM gewerkt. Dit is aangetoond in hoofdstuk 5: het opzetten van SCRUM en in de uitvoering van SCRUM in hoofdstukken 6, 7 en 8.

G-F: Leren leren

In de ICT blijf je door de nieuwste ontwikkelingen in technologie telkens leren. Ook weet ik natuurlijk niet alles en zijn er genoeg dingen dat ik nog niet wist en ik geleerd heb. Zo heb ik voor het eerst in Angular gewerkt, wat een hele nieuwe taal was en heb ik voor het eerst design patterns toegepast. Dit is aangetoond in hoofdstukken 7 en 8.

LITERATUURLIJST

SCRUM Activiteiten. (2016). [Foto]. Geraadpleegd van https://cdnsite3.assist.ro/sites/default/files/promoted_images/blog/book-image.png Goel, A. (2019, 10 oktober).

10 Best JavaScript Frameworks to Use in 2019. Geraadpleegd op 31 oktober 2019, van https://hackr.io/blog/10-best-javascript-frameworks-2019 H., S. (2019, 2 juli).

Angular vs React vs Vue: Which is the Best Choice for 2019? Geraadpleegd op 31 oktober 2019, van https://hackernoon.com/angular-vs-react-vs-vue-which-is-the-best-choice-for-2019-16ce0deb3847

Angular. (z.d.). Geraadpleegd op 31 oktober 2019, van https://angular.io/

React – A JavaScript library for building user interfaces. (z.d.). Geraadpleegd op 29 oktober 2019, van https://reactjs.org/

Vue.js. (z.d.). Geraadpleegd op 29 oktober 2019, van https://vuejs.org/

Design Factory Method Design Pattern UML. (2016, 1 oktober). Geraadpleegd op 15 december 2019, van http://designpattern.co.il/Factory.html

Builder pattern. (2019, 18 december). Geraadpleegd op 15 december 2019, van https://en.wikipedia.org/wiki/Builder_pattern

Daityari, S. (2019, 11 december). *Angular vs React vs Vue: Which Framework to Choose in 2020*. Geraadpleegd op 9 januari 2020, van https://www.codeinwp.com/blog/angular-vs-vue-vs-react/

Kumar, A. (2019, 30 september). *React vs. Angular vs. Vue.js: A Complete Comparison Guide*. Geraadpleegd op 9 januari 2020, van https://dzone.com/articles/react-vs-angular-vs-vuejs-a-complete-comparison-gu

TechMagic. (2019, 19 juni). *React vs Angular vs Vue.js — What to choose in 2019? (updated)*. Geraadpleegd op 9 januari 2020, van https://medium.com/@TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d

Formatted: Justified, Space After: 10 pt, Line spacing: Multiple 1,15 li

BUSINESS IN CONTROL

BUSINESS IN CONTROL

Afstudeerscriptie over het herontwerpen en het versnellen van de business dimension navigator

BIJLAGENBOEKJE

BIJLAGENBOEKJE

R. Paltoe - 15025713

Kuru, K. | Hoek, J.J. van der

De Haagse Hogeschool, HBO-ICT

02 September 2019 – 10 Januari 2020

CERRIX B.V.

INHOUDSOPGAVE BIJLAGES

BIJLAGE 1: DE WOORDENLIJST

CERRIX-module	Betekenis
Controls	Een manier om het risico te beheersen
Documents	Documenten die behoren bij een bedrijfsproces
Events	Een incident (manifestatie van het risico)
Measure of Improvements	Een verbeteringsmaatregel
Process overview	Een visuele weergave van de processen (denk aan swimlanes)
Processing Purposes	Een verwerkingsdoel voor het omgaan van data (vanuit de AVG)
Risk actions	Oudere, uit gefaseerde verbeteringsmaatregelen voor de risico's
Risks	Een risico waar een bedrijf aan blootgesteld kan worden
TypeScript	De programmeertaal van Angular

Formatted: Justified

Formatted: Justified

Formatted: Justified

Formatted: Justified

Formatted: Justified

Formatted: Justified

Formatted: Justified

BIJLAGE 2: HET AFSTUDEERPLAN

Afstudeerplan

Informatie afstudeerder en gastbedrijf

Afstudeerblok: 2019-2.1 (start uiterlijk 2 september 2019)
Startdatum uitvoering afstudeeropdracht: 02-09-2019
Inleverdatum afstudeerdossier volgens jaarrooster: 10 januari 2020

Studentnummer: 15025713
Achternaam: Dhr. Paltoe
Voorletters: R
Roepnaam: Rohit
Adres: Schalk Burgerstraat 124
Postcode: 2572VB
Woonplaats: Den Haag
Telefoonnummer: n.v.t.
Mobiel nummer: 06-84533391
Privé emailadres: rohit_paltoe@live.nl

Opleiding: HBO-ICT
Locatie: Den Haag
Variant: Voltijd

Naam studieloopbaanbegeleider: Mirabai Vosteen
Naam begeleidend examinator: Kadir Kuru
Naam expert examinator: Cobie van der Hoek

Naam bedrijf: CERRIX B.V.
Afdeling bedrijf: Development
Bezoekadres bedrijf: Koninginnegracht 29
Postcode bezoekadres: 2514AB
Postbusnummer: n.v.t.
Postcode postbusnummer: n.v.t.
Plaats: Den Haag
Telefoon bedrijf: 070 363 7733
Telefax bedrijf: n.v.t.

Internetsite bedrijf: https://cerrix.com/

Achternaam opdrachtgever: Dhr. Bruggeman
Voorletters opdrachtgever: P.
Titulatuur opdrachtgever:
Functie opdrachtgever: Project Manager
Doorkiesnummer opdrachtgever: n.v.t.
Email opdrachtgever: paul.bruggeman@cerrix.com

Achternaam bedrijfsmentor: Dhr. Leeuwen, van
Voorletters bedrijfsmentor: M.
Titulatuur bedrijfsmentor:
Functie bedrijfsmentor: Teamleader of development
Doorkiesnummer bedrijfsmentor:
Email bedrijfsmentor: martin.van.leeuwen@cerrix.com

Doorkiesnummer afstudeerder: n.v.t.
Functie afstudeerder (deeltijd/duaal): n.v.t.

Titel afstudeeropdracht:

Het herschrijven van de Business Navigator module voor CERRIX.

OPDRACHTOMSCHRIJVING

1. Bedrijf

CERRIX is een software en consultancy bedrijf dat zich specialiseert op het gebied van Risico- en procesmanagement. Dit is een relatief nieuw vakgebied dat nog volop in ontwikkeling is en waar CERRIX graag haar bijdrage aan wilt leveren. Door deze bijdrage kunnen organisaties hun risicomanagement effectief in kaart brengen en beheersen.

CERRIX kent vele klanten, o.a. grote banken en (internationale) verzekeringsmaatschappijen. Bij deze bedrijven speelt risicomanagement natuurlijk een grote en (volgens de wet) belangrijke rol. Elk bedrijf moet een solide risicomanagement hebben, dit levert goede zaken voor CERRIX op. De software die

CERRIX biedt wordt aangeleverd als SaaS. (Software as a Service) en wordt dus volledig gehost op de servers van CERRIX. Dit zorgt ervoor dat de klanten van CERRIX hun eigen aparte website hebben binnen CERRIX, waar de applicatie op draait.

2. Probleemstelling

GOVERNANCE RISK COMPLIANCE-TOOL

In de GRC-applicatie die CERRIX haar klanten biedt, kunnen klanten verschillende risicomanagement gerelateerde objecten aanmaken die elk gemeten en gemonitord kunnen worden. Zo kunnen er risks, controls, events, KPI's & KRI's, finding reports en MoI's

(verbetermaatregelen) en processing purposes (verwerkingsdoelen n.a.v. de AVG) aangemaakt worden. Al deze objecten bevinden zich in aparte modules binnen de applicatie en hebben elk hun eigen koppelingen, onderlinge relaties met elkaar en hun eigen attributen.

BUSINESS NAVIGATOR

Op dit moment heeft CERRIX ook de module 'Business Navigator'. Dit is een module waarin de gebruiker door middel van filtering op organisatie of business dimension (bedrijfsproces) een overzicht kan krijgen van de verschillende bovengenoemde modules binnen de applicatie en de relaties met elkaar. Hierdoor kunnen op dit moment alleen managers (omdat zij de hoogste rechten hebben binnen de applicatie) in een oogopslag zien hoe hun organisatie of bedrijfsproces ervoor staat. De overzichten die hier getoond worden zijn afhankelijk van de rechten die de gebruiker heeft.

Nadat er bijvoorbeeld op organisatie en business dimension wordt gefilterd zullen er lijsten verschijnen met alle gekoppelde controls/ events/ risks etc.. Wanneer er geklikt wordt op een risk, dan worden de relaties die de risk met de andere modules heeft ook weergeven. Zo worden er dan ook gelinkte controls/ events/ MoI's etc. weergeven per geselecteerde item.

COMPLEXITEIT

De complexiteit hierin is dat een gebruiker van CERRIX per module andere rechten en rollen kan hebben, waardoor het getoonde overzicht verschilt per gebruiker. Per module zijn er verschillende rollen waar een gebruiker aan toegewezen kan zijn. Zo kan een gebruiker bijvoorbeeld alleen controls binnen zijn eigen organisatie zien, maar mag hij wel weer risico's verspreid over alle organisaties zien. Dit terwijl hij events alleen mag zien als hij als een auditor is toegewezen op het event. Dit algoritme zal bedacht moeten worden.

Hoe dit dan in elkaar zit in de business navigator is dat een geklikte risico gelinkte controls en events etc. heeft. Deze controls kan een gebruiker alleen zien als de gebruiker bepaalde Control viewer rechten heeft, de events die daar dan weer aan gekoppeld zijn kan hij alleen zien met de juiste event rechten. Dit zorgt voor de extra laag complexiteit. Op dit moment is dit niet geïmplementeerd, en kan de gebruiker dus alle gelinkte modules zien. Dit zal dus uitgezocht moeten worden.

De invoerschermen die er op dit moment zijn, zijn geschreven in EXT.Net en de backend calls naar de database zijn niet efficiënt genoeg opgezet. Er worden op dit moment hele tabellen en sub tabellen volledig opgehaald, terwijl er alleen bepaalde kolommen nodig zijn. Dit zorgt ervoor dat het ophalen van data tot wel 10-15 seconden duurt, wat de schermen onwerkbaar maakt. Dit komt omdat er jaren niet meer is gewerkt aan deze module. Hierom heeft CERRIX dringend een geheel nieuw ontwerp en verbeterde performance nodig. Dit zal dan dus ook from scratch gemaakt moeten worden en zal hierom ook zorgen voor een extra laagje aan complexiteit.

3. Doelstelling van de afstudeeropdracht

Wanneer de opdracht in januari 2020 voltooid is, dan wordt er verwacht dat de module volledig omgeschreven zal zijn van Ext.Net naar Angular 7 met een nieuw ontwerp, die aan de huisstijl van CERRIX voldoet. Ook zullen de backend calls volledig herzien worden, zodat ze aangepast zijn aan de nieuwe performance criteria. Tenslotte zal de data die getoond wordt volledig gefilterd worden op basis van de x aantal rechten die de gebruiker van de applicatie bezit.

4. Resultaat

Nadat de afstudeeropdracht met succes afgerond zal zijn, zal de nieuwe, verbeterde Business Navigator module de klanten van CERRIX positief verrassen. Hierdoor kunnen niet alleen de managers van een bedrijf deze module gebruiken, maar omdat de data uiteindelijk gefilterd wordt op basis van de rollen en rechten per gebruiker, kunnen normale werknemers ook bij deze data komen en kunnen zij deze module gebruiken.

Doordat de performance van de Business Navigators verbeterd is, zal deze module sneller laden en zal dit sneller te gebruiken zijn. Dit scheelt uiteindelijk tijd en geld, omdat de klanten meer tevreden zullen zijn met het gebruik van de CERRIX-applicatie en dit CERRIX (en de klanten die dit gebruiken) dan ook weer geld oplevert.

De nieuwe UI die ontworpen zal worden, zal ook sterk bijdragen aan de positieve ervaring die de klant zal hebben door het gebruik van de business navigator. Het zal er niet langer verouderd meer uitzien. Alles zal logischer geplaatst worden en intuïtiever aanvoelen, waardoor het coherent is met de helft van de CERRIX applicatie, omdat de overige helft ook op weg is naar Angular schermen.

5. Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

- *Plan van aanpak schrijven (3 dagen)*
- *Aanvullende oriëntatie op het probleemdomein: kijken hoe de gebruikers de business navigator gebruiken en naar de functionaliteit hiervan (onderzoek). (17 dagen)*
- *Gesprekken met de consultants, om een nieuw UI ontwerp vast te kunnen stellen (1 dag)* o *Onderzoeken: Hoe gebruikt de klant de business navigator, waar let hij op. En welke Angular componenten ik kan gebruiken bij het implementeren van de nieuwe design. (5 dagen)*
- *Ontwikkelen van de UI in Angular (15 dagen)*
- *Het herzien en herschrijven van de backend calls (15 dagen)* o *Gesprekken met afstudeer begeleider voor het afstudeerverslag.(1 dag)* o *Bedrijfsbezoek (1 dag)* o *Verslag schrijven (15 dagen)*
- *Gesprekken met afstudeer begeleider voor de voortgang en feedback (1 dag)* o *Gesprek met de manager om de stand van zaken te bespreken (voortgang van opdracht) (2 dagen)*
- *Testen (5 dagen)* o *Adviseren (2 dagen)* o *Overdracht (2 dagen)*
- *(Totaal 85/ 85 dagen)*

6. Op te leveren (tussen)producten

De producten die ik zal moeten opleveren zullen onder meer zijn:

- Onderzoek naar hoe de klant de module gebruikt
- Een nieuw UI-ontwerp dat samen met de consultants is gemaakt
- De gehele nieuwe versie van de business navigator module
- Testrapporten die gemaakt zijn voor het testen van de UI en de security.
- Een overdrachtshandleiding / gesprek
- Een afsluitingspresentatie bij CERRIX

7. Te demonstreren competenties en wijze waarop

Verplichte beroepstaken:

A-1: Analyseren probleemdomein:

Het analyseren zal gebeuren door het kijken wat er misgaat in de huidige code. Waar verloopt alles traag en hoe beïnvloedt dit de performance?. Dit zal geanalyseerd worden, waardoor de backend uiteindelijk herschreven zal gaan worden.

G-C: Kritisch en methodisch werken:

De ontwikkelmethode die gebruikt zal gaan worden is SCRUM. De taken zullen allereerst onderverdeeld worden in epics, die daarna user stories zullen worden. Deze user stories zullen toegewezen worden aan sprints. Dit zal worden verwerkt in het eindverslag.

G-F: Leren leren:

Natuurlijk stopt het leren nooit. Er zal verdiept moeten worden in een nieuwe taal, Angular. Er moet gekeken worden naar wat voor lagenstructuur Angular hanteert en hoe de connecties tot stand komen en wat er gedaan kan worden om de module aan te kunnen aansluiten aan de Angular kant. Hier zal naar gekeken moeten worden en onderzoek naar worden gedaan, waardoor leren leren voortdurend terug zal komen.

Gekozen beroepstaken:

B-4: Gemotiveerd selecteren van ICT gerelateerde oplossingen:

Er moet onderzocht worden welke Angular components er zijn, en welke er het beste gebruikt kunnen worden in de nieuwe front-end. Deze selectie en de overige keuzes om models/ components op deze manier te gebruiken zal gemotiveerd moeten worden.

C-6: Ontwerpen software:

Software kan niet zomaar opgeleverd worden. Hier moet eerst het nodige ontwerp voor opgeleverd zijn. Dit ontwerp zal worden vastgesteld n.a.v. de consultatie van de testers van het bedrijf en door het uitzoeken van de huidige onderlinge verbindingen van de code tussen de andere lagen. Daarnaast zullen er ook use cases en klassendiagrammen opgesteld worden. Dit

zal zorgen voor een betere documentatie, voor duidelijkheid over de applicatie en onderlinge lagen zelf en wat er daadwerkelijk gecodeerd moet worden.

D-14: Realiseren software:

Natuurlijk wordt er software gerealiseerd. De huidige business navigator zal volledig herschreven worden, met nieuwe backend calls. Uiteindelijk zal alles worden opgeleverd in de vorm van werkende software en zal dit terug komen in het eindverslag.

D-15: Testen

Niet alleen het schrijven van software van belang, maar uiteraard zijn de tests die erop volgen zeer belangrijk. Er zullen UI-tests plaatsvinden, om te zorgen dat de Angular kant werkt, maar er zullen ook speciale tests uitgevoerd worden om te testen dat de juiste data weergeven wordt aan de gebruikers met de juiste rollen, omdat dit voor de security van de applicatie van belang is.

BIJLAGE 3: PLAN VAN AANPAK

PROCESSEN IN OVERZICHT

PLAN VAN AANPAK OVER HET VERSNELLEN EN HERONTWERPEN VAN DE BUSINESS DIMENSION NAVIGATOR MODULE IN DE CERRIX-APPLICATIE IN OPDRACHT VAN DE HAAGSE HOGESCHOOL EN CERRIX B.V.

Formatted: Justified

Auteur: Rohit Paltoe

Project: Plan van Aanpak

Herschrijven Business dimension navigator module in de CERRIX-applicatie

Periode: September 2019 – Januari 2020

Onderwijsinstelling: De Haagse Hogeschool

Opdrachtgever: CERRIX B.V.

INHOUDSOPGAVE

INLEIDING

Voor steeds meer organisaties is het hebben van een juiste **GRC** programma van belang, maar waarom is deze GRC-tooling van belang? H, K (2016) legt het op de volgende manier uit.

"To ensure that businesses protect their information, have consistent cohesion departmentally, and follow all governmental regulations, a governance, risk and compliance, (GRC) program is important. Businesses can benefit from a strong GRC program that helps to minimize the threats and risks that companies are exposed to on a daily basis. As GRC guidelines work in conjunction with your business, this increases the safety in your company and it also gives you a safeguard in ensuring your corporate accountability is effectively planned, designed, and implemented across all work platforms"

CERRIX biedt haar klanten een GRC-tool aan. Dit is een software waarin klanten hun Governance (Hoe managers van een bedrijf hun bedrijfsprocessen kunnen inzien en beheersen), Riskmanagement (het beheersen en managen van risico's) en Compliance (het op orde hebben van je data n.a.v. de wetgevingen) kunnen invoeren en beheersen.

Voorheen hielden bedrijven dit bij via een excel-sheet en werd dit dus los en statisch opgeslagen. CERRIX heeft een SaaS ontworpen, waarin klanten bijvoorbeeld risico's aan kunnen maken, de kosten hiervan kunnen invoeren, tegenmaatregelen en controles kunnen vaststellen en hierdoor hun risicomanagement overzichtelijk in orde hebben.

Daarnaast kunnen organisaties hun compliance, neem de nieuwe privacywetgeving AVG als voorbeeld. Een bedrijf kan via de CERRIX-applicatie een 'verwerkingsdoel' aanmaken en zo vastleggen hoe, wanneer, waarvoor en voor hoelang een gegeven van een klant gebruikt wordt en zal worden bijgehouden. Dit is natuurlijk zeer handig voor bedrijven om in gebruik te nemen. Deze markt is nog volop in ontwikkeling en CERRIX wilt hier heel graag haar bijdrage aan leveren.

Formatted: Heading 2, Justified

Formatted: Justified

CONTEXT

Als bedrijven hun GRC in orde willen hebben, kunnen zij de CERRIX applicatie huren. In de applicatie is het mogelijk om een 'Risico', 'Controle', 'Verbeteringsmaatregel', 'Event', KPI, en KRI (key risk indicator) en 'Verwerkingsdoel' aan te maken in de applicatie. Deze kunnen vervolgens allemaal gelinkt worden aan een organisatie en aan een business dimension (bedrijfsproces). Hiermee kunnen organisaties hun risicomanagement bijhouden. Hieronder volgt een voorbeeld over hoe dit praktisch gebruikt wordt:

Ter illustratie:

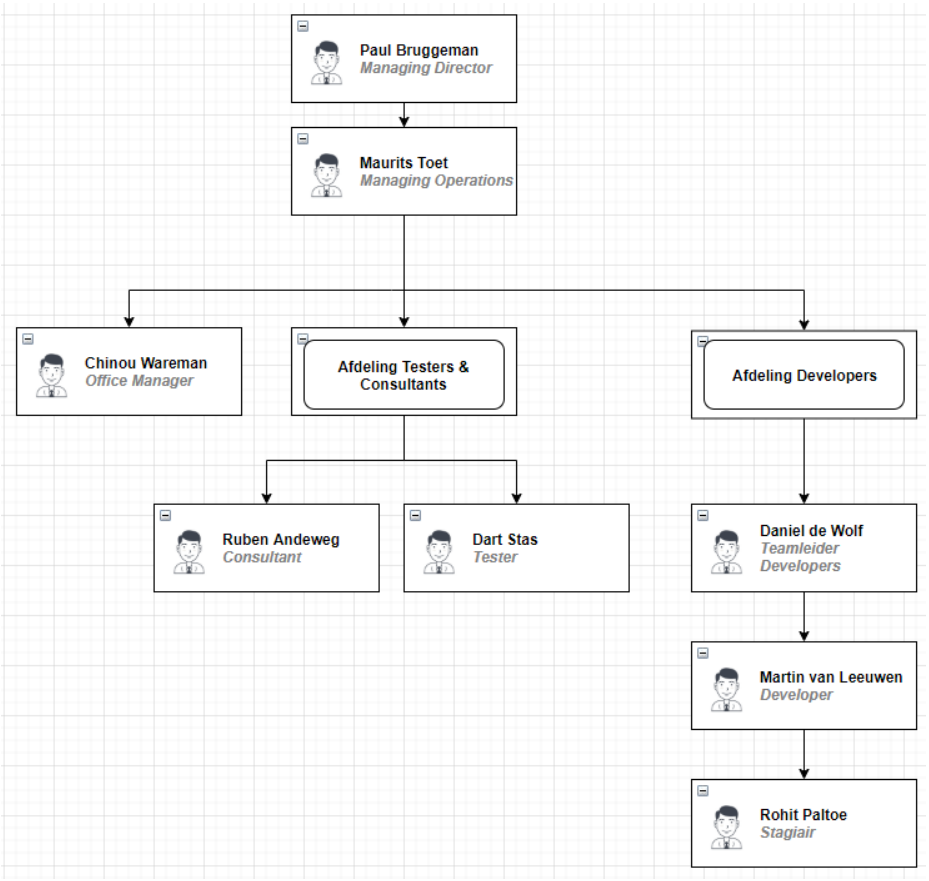
*Stel er is de **organisatie** 'De Haagse Hogeschool', met gekozen **business dimension** 'Studenten' , dan zouden wij hier een lijst met (bestaande) **risico's** zien, bijvoorbeeld: dat het mogelijk is dat een student de kennis niet tot zich kan nemen door afwezigheid. Een **event** zou dan zijn dat de student afwezig is door ziekte. Hij heeft weinig fruit gegeten. Een **control** zou dan zijn, dat de student meer fruit zal moeten eten, waardoor hij minder snel ziek wordt. De **measure of improvement** zou dan tot slot zijn dat er wekelijks een fruitmand naar de student bezorgd wordt.*

Dit wordt dan opgeslagen in de applicatie, zodat er hier rekening mee gehouden kan worden (Risksmagement) een het management de juiste overzichten over hun bedrijfsprocessen verkrijgen (Governance). De verwerkingsdoelen zijn vaststellingen over hoe een bedrijf met haar data omgaat vastgesteld door de AVG. Hiermee kunnen bedrijven auditors laten zien dat zij werken volgens de wetgevingen (Compliance). De CERRIX-applicatie biedt dit dus allemaal.

Formatted: Justified

OPDRACHTGEVER

Het team van CERRIX ziet er als volgt uit (zie figuur 1)



Figuur 4 Organigram van CERRIX

Op de afbeelding zijn de collega's te zien waar ik direct contact mee heb. Martin van Leeuwen, senior developer en voormalig teamleider zal de rol als bedrijfsmentor vervullen. Ik kan naar hem toe als ik vastloop, voor de code reviews en als ik nog andere vragen heb. Er zijn nog tien andere developers die hier niet op staan, maar als ik niet bij Martin kan dan zal ik hen vragen om hun expertise.

Formatted: Heading 2, Justified

Formatted: Justified

Daarnaast zijn er nog consultants en testers binnen het bedrijf. Met hun zal ik aan tafel moeten zitten om tot een definitief ontwerp voor de business navigator te komen. Zij zullen mij ook helpen met het achterhalen van verdere requirements en met het opstellen van testrapporten.

De baas is Paul Bruggeman, hij is de managing director, zijn rechter hand is Maurits Toet. De office manager die onder hen staat is Chinou Wareman. Dit is het team die mij zal ondersteunen en waarmee ik zal werken gedurende mijn afstudeertraject.

PROBLEEMANALYSE EN PROBLEEMSTELLING

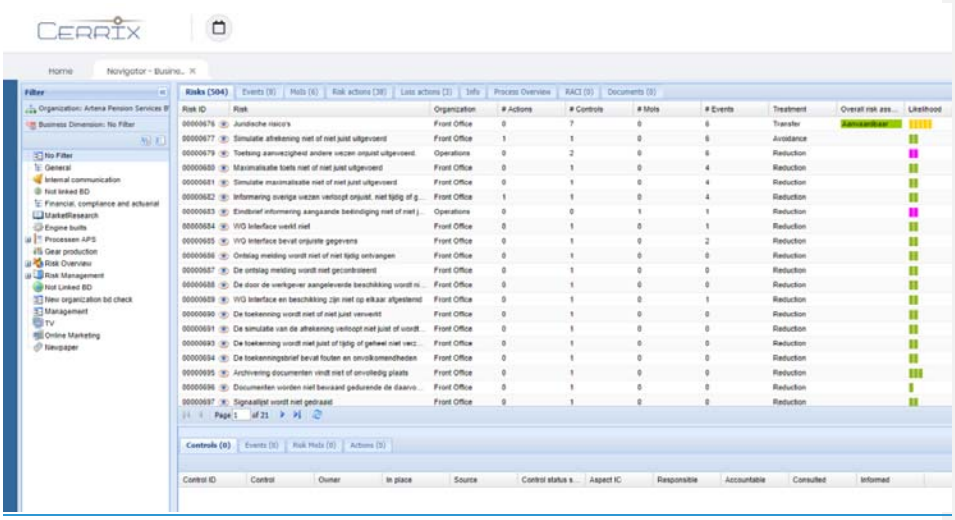
Formatted: Heading 2, Justified

Er zijn dus verschillende objecten en koppelingen die binnen de CERRIX-applicaties gemaakt kunnen worden. Zij zijn dan weer allemaal gekoppeld aan een organisatie en bedrijfsproces (business dimension). De Business dimension navigator module is de plek waar dit allemaal in samenkomt. Hier kun je een organisatie en business dimension selecteren, gekoppeld aan betreffende organisatie, waardoor er hele overzichten te zien zijn met alle koppelingen aan het gekozen bedrijfsproces.

Dit overzicht is handig voor managers, zodat zij hun bedrijfsprocessen kunnen beheersen. Zo kunnen zij zien wat voor soort risico's, controles, verbeteringsmaatregelen, verwerkingsdoelen etc. gekoppeld zijn aan hun processen. Dit wordt allemaal weergeven op 1 pagina in een overzicht in tabelvorm. Dit bestond al in de applicatie. (*Zie figuur 2*)

Het probleem is dat deze overzichten onoverzichtelijk worden weergeven. Het filter aan de linkerkant om een selectie te maken tussen een organisatie en business dimension heeft een ingewikkeld inklapsysteem. Daarnaast wordt er een overschot aan informatie getoond. De gebruikers zwemmen in de data, hierdoor zijn de overzichten onoverzichtelijk. Dit komt grotendeels, doordat er een boven en een onder helft is op het scherm, dat niet intuïtief is (Het onder helft is voor de details van de data). Ook duurt het laden van de pagina heel lang. Het kan tot wel 20 seconden duren voordat er data weergeven wordt op het scherm, dit gebeurt dan weer opnieuw nadat er op een andere tab wordt geklikt. Tot slot is de pagina nog geschreven in een oude taal, Ext.Net. CERRIX heeft een overstap gemaakt naar Angular. Hierdoor moet de pagina aan de voorkant ook nog omgeschreven worden, zodat de pagina weer past bij de rest van de applicatie.

In het ontwerprapport zal het probleem nader worden geanalyseerd.



Figuur 5 Huidige Business dimension navigator pagina (aan omschrijving toe)

Formatted: Justified

DOELSTELLING EN EINDRESULTAAT

Het doel is dus om een nieuwe en verbeterde versie te schrijven van de business dimension navigator. Ik zal de code voor het ophalen van de data moeten herschrijven, waardoor dit sneller gebeurt en de pagina sneller zal gaan laden. Daarnaast zal ik een nieuw ontwerp moeten gaan maken, dat ik zal moeten implementeren in Angular. Tot slot zal ik nog rekening moeten houden met eventuele extra features die ook geïmplementeerd zullen moeten worden.

Tijdens mijn ontwikkeling kan er verwacht worden dat het een tijdje zal duren voordat er een ontwerp tot stand zal komen, omdat dit nog door verschillende medewerkers van CERRIX zal moeten gaan om gekeurd te worden. Nadat er een ontwerp is vastgesteld, zal ik onderzoek moeten doen naar verschillende Angular componenten die ik zou kunnen gebruiken, omdat ik nog over geen Angular-kennis beschik.

RISICOANALYSE

Uiteraard zal ik rekening moeten houden met eventuele risico's die kunnen plaatsvinden. Hiervoor is er een risicoanalyse gemaakt met risico's die eventueel konden plaatsvinden.

Deze risico's waren er gekozen, omdat er onder andere nog geen beschikking is over een Visual Studio licensie en het bedrijf hiervoor zal moeten regelen. Daarnaast is er nog geen idee over hoe de communicatie nou daadwerkelijk in het bedrijf verloopt, hoe de collega's reageren etc. Dit zal in de eerste weken duidelijk worden. Ook is er nog geen kennis van Angular. Dit zal worden aangeleerd tijdens de uitvoering van het project en is het meer dan mogelijk dat er (niet/genoeg) technische ondersteuning is. Tot slot kan de scope of het project zelf veranderen, omdat het ontwerp nog gemaakt moet worden. Aangezien dit er nog niet is kan er van alles boven water komen qua requirements en extra features bijvoorbeeld.

Bedreiging	**Kans** (1-5)	**Impact** (1-5)	**Risico score** (K*I)	**Tegenmaatregel**	**Risico eigenaar**
Technisch (middelen)					
TR.1: Niet beschikken over de juiste Visual Studio licenties.	1	5	5	Een gratis versie proberen te vinden (community editie) of anders vragen of ik een licentie via CERRIX kan verkrijgen.	Rohit
Organisatorisch (mensen en procedures)					
O.R.1: De communicatie gaat fout: er worden geen/ onjuiste afspraken gemaakt over de contactmomenten	3	4	12	Tijdig afstemmen met het ontwikkelteam/ de verantwoordelijken wanneer de contactmomenten zullen zijn voor het feedback.	Rohit
O.R.2: Ondeskundigheid: ik bezit niet over de juiste kennis voor de technische implementatie.	2	5	6	Bij ondeskundigheid, mijn collega's vragen om raad. (Bijvoorbeeld over vraagstukken over de Angular taal). Als mijn collega's het ook niet weten, zal ik het opzoeken op het internet.	Rohit

Formatted: Justified
Formatted: Justified
Formatted: Justified
Formatted: Justified
Formatted: Justified
Formatted: Justified
Formatted: Justified

Functioneel (wat moet er klaar zijn als het af is)					
F.R.1: De omvang van het project is te groot en het komt niet op tijd af	4	5	20	Tijdig de scope bepalen van de opdracht: niet te veel proberen te doen, maar focussen op de basis en daarna pas op eventuele extra features.	Rohit
F.R.2: De opdracht is niet goed gedefinieerd. Er worden andere uitwerkingen/ implementaties verwacht dan wat er gemaakt zal worden	4	4	16	Veel contactmomenten hebben om de opdracht te bespreken, zodat iedereen op 1 lijn zal zitten.	Rohit
F.R.3: De scope van het project verandert: de baas of de verantwoordelijken willen later iets anders.	4	4	16	Wederom veel contactmomenten hebben om de opdracht te laten zien, feedback te verwerken zodat de scope niet (of zelfs niet al te veel) af zal wijken van het origineel.	Rohit

Het project zal worden uitgevoerd over een periode van 17 weken. Gezien de risico's en de op te leveren producten lijkt het mij het handigst om SCRUM te gebruiken als ontwikkelmethode. Dit komt omdat ik een paar risico's heb die hoog gescoord zijn. Om hier rekening mee te houden zal ik iteratief middels SCRUM aan werken, zodat ik hier voldoende documentatie over kan opleveren.

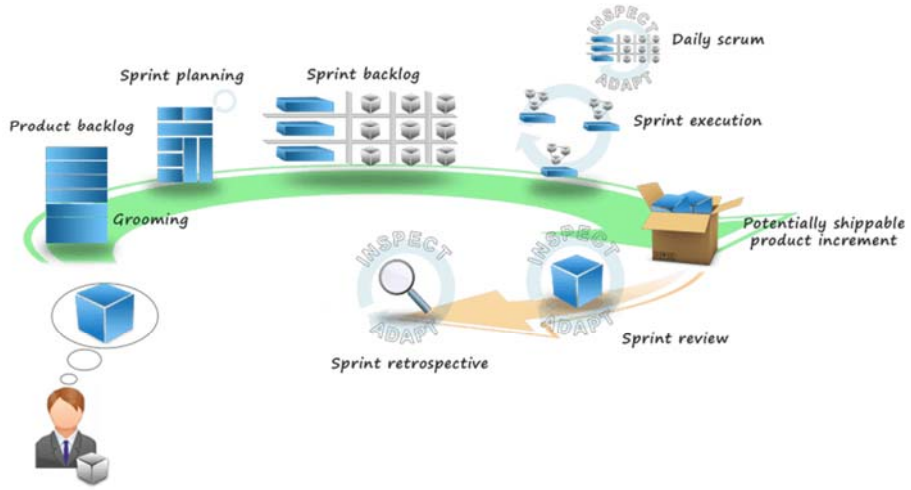
Daarnaast kan ik door het gebruik van SCRUM elke 2 weken een harde deadline zetten om een minimum viable product op te leveren. Dit gaat ook gepaard met meetings die ik kan houden met Paul, de consultants en de testers, zodat ik telkens opnieuw feedback over mijn product kan krijgen, waarna ik klaar ben voor de volgende sprint.

AANPAK EN ONTWIKKELMETHODE

Het project zal worden uitgevoerd over een periode van 17 weken. Gezien de risico's en de op te leveren producten lijkt het mij het handigst om SCRUM te gebruiken als ontwikkelmethode. Dit komt omdat ik een paar risico's heb die hoog gescoord zijn. Om hier rekening mee te houden zal ik iteratief middels SCRUM aan werken, zodat ik hier voldoende documentatie over kan opleveren.

Daarnaast kan ik door het gebruik van SCRUM elke 2 weken een harde deadline zetten om een minimum viable product op te leveren. Dit gaat ook gepaard met meetings die ik kan houden met Paul, de consultants en de testers, zodat ik telkens opnieuw feedback over mijn product kan krijgen, waarna ik klaar ben voor de volgende sprint.

De SCRUM activiteiten die ik zal uitvoeren en artefacten die ik zal opleveren zullen het volgende zijn (zie figuur 3)



Figuur 6 Het algemeen model voor sprint activiteiten en op te leveren artefacten. Deze activiteiten en artefacten zal ik ook gebruiken, omdat zij prima aansluiten binnen het afstudeertraject.

Naar: David, G *(2017)*

Artefacten:

- **Product backlog:**
  Er zal een requirements analyse uitgevoerd worden om de requirements te achterhalen. Deze requirements zullen vervolgens een tijdsinschatting krijgen, zodat er ingeschat kan worden hoelang het uitvoeren hiervan zal duren en het dus wel haalbaar is. Vervolgens zullen zij ingedeeld worden volgens de MoSCoW-manier, zodat er gekeken kan worden welke requirements cruciaal zijn voor een werkend product, welke leuk zijn om erbij te hebben of welke requirements niet meegenomen zullen worden. Deze requirements zullen in de vorm van user stories opgesteld worden, omdat dit de SCRUM-manier is

Formatted: Justified

Formatted: Justified

Formatted: Justified

Formatted: Justified, Indent: Left: 1,27 cm, No bullets or numbering

en ik al enige ervaring heb met het opstellen van de product backlog op deze manier. Dit zal allemaal samenkomen in de product backlog.

- **Definition of Ready:**
  Dit zal een verzameling van afspraken zijn die opgesteld voor het behouden van de kwaliteit van de user stories. Doordat deze user stories in een bepaalde template opgesteld worden, en dus aan de kwaliteitseisen voldoen, kan het development team deze user stories gemakkelijk, snel en effectief oppakken en uitvoeren.

- **Definition of Done:** Dit zal een verzameling zijn van afspraken, waaraan het uitgevoerde werk zal moeten voldoen. Dit kan ook wel de acceptatiecriteria van het opgeleverde werk genoemd worden. Dit is handig, zodat er altijd gecontroleerd kan worden of de taak wel juist en zo compleet mogelijk is uitgevoerd.

- **Sprint planning:** Dit zal een document zijn, waarin de sprints een bepaalde thema en deadline krijgen. Binnen deze thema zal er gestreefd worden naar een bepaald doel. Om het doel te bereiken zullen er product backlogitems toegewezen worden aan deze sprints. Dit zal ervoor zorgen, dat het SCRUM team een goed overzicht krijgt over wat er gedaan moet worden per sprint, zodat er alleen op 1 doel gefocust hoeft te worden per sprint.

Activiteiten (per sprint):

- **Daily scrum:** Een daily scrum is een moment van 5 minuten, waarin er binnen het team van developers in het kort verteld wordt wat er gister was gedaan, tegen welke problemen er aan gelopen worden en wat er vandaag zal worden gedaan. Dit wordt momenteel al dagelijks om 10:00 uitgevoerd onder het team van developers van CERRIX en hier kan ik mooi bij aansluiten. Dit zal gunstig zijn, omdat ik ook altijd tegen problemen aan kan lopen en zo mogelijk hulp kan krijgen van collega's.

- **Sprint execution:** Tijdens de sprint execution zal de daadwerkelijke sprint uitgevoerd worden.

- **'Done':** Dit zal het resultaat (ook wel het 'potentially shippable product increment') zijn, wat opgeleverd zal worden aan het einde van elke sprint. Dit resultaat zal moeten voldoen aan de opgestelde kwaliteitseisen vanuit de definition of done.

- **Sprint review:** Dit zal een review zijn, dat uitgevoerd zal worden aan het einde van elke sprint, nadat het increment is opgeleverd. Tijdens deze review zal het SCRUM team en de belanghebbenden aan tafel zitten om het increment te bekijken en hier zo nodig feedback op te geven. Als het zo mocht zijn dat het product niet voldoet en veranderd moet worden, kan dat meegenomen worden naar de volgende sprint.

  Het houden van een sprint review is wenselijk, omdat de belanghebbenden van CERRIX (de baas Paul, Maurits en de consultants) zo het product kunnen evalueren en eventuele aanpassingen meegenomen kunnen worden naar de volgende sprint.

- **Sprint retrospective:** De sprint retrospective vindt plaats na de review. Dit zal een mogelijkheid zijn om terug te reflecteren op het doorlopen proces van de uitgevoerde sprint. Hier zal besproken worden wat

Formatted: Font: Bold
Formatted: Justified
Formatted: Justified, Indent: Left: 1,27 cm, No bullets or numbering
Formatted: Justified
Formatted: Font: Bold, Dutch (Netherlands)
Formatted: Justified, Indent: Left: 1,27 cm, No bullets or numbering
Formatted: Justified
Formatted: Font: Bold, Dutch (Netherlands)
Formatted: Justified, Indent: Left: 1,25 cm, Hanging: 0,02 cm, No bullets or numbering
Formatted: Justified, Indent: Left: 1,27 cm, No bullets or numbering
Formatted: Justified
Formatted: Font: Bold, Dutch (Netherlands)
Formatted: Justified, Indent: Left: 1,27 cm, No bullets or numbering
Formatted: Justified
Formatted: Font: Bold, Dutch (Netherlands)
Formatted: Justified, Indent: Left: 1,27 cm, No bullets or numbering
Formatted: Justified
Formatted: Font: Bold, Dutch (Netherlands)
Formatted: Justified, Indent: Left: 1,25 cm, Hanging: 0,02 cm, No bullets or numbering
Formatted: Justified
Formatted: Font: Bold, Dutch (Netherlands)
Formatted: Justified, Indent: Left: 1,25 cm, Hanging: 0,02 cm, No bullets or numbering
Formatted: Justified, Indent: Left: 1,27 cm, No bullets or numbering
Formatted: Justified

wel werkte, wat niet en hoe dit de volgende keer dan beter aangepakt zal gaan worden. Dit zal handig zijn, omdat er hierdoor eventuele processen die niet lukten zo liever vermeden kunnen worden in de volgende sprint.

Formatted: Font: Bold, Dutch (Netherlands)

Deze SCRUM activiteiten zullen per sprint uitgevoerd worden. Wat deze activiteiten opleverden zal concreet besproken worden in het volgende hoofdstuk.

OMGEVING PROJECT

Ik zal programmeren in C#. Er wordt gewerkt met ASP .NET (IDE is Visual Studio) en de taal die ik extra zal gaan leren gaat Angular zijn. Om in Angular te kunnen programmeren zal ik de IDE Visual Studio Code gaan gebruiken. Ik zal mijn eigen stage branch krijgen (dit is een branch van master) en hier zal ik alles naartoe pushen. Op het einde van mijn afstuderen zal deze branch gemerged worden met master.

Daarnaast krijg ik ook mijn eigen Azure area binnen CERRIX. Ik mag in de speciaal gemaakte area 'Navigators' mijn work items aanmaken en dit dan verder toekennen aan sprints, zodat ik ook mooie burn down charts etc. kan maken.

KWALITEITSBEWAKING

Om te zorgen dat de kwaliteit van mijn code gewaarborgd blijft, zal er na elk stukje code dat ik oplever een review plaatsvinden. Mijn begeleider vanuit werk zal dan naar mijn pull request kijken en eventuele feedback geven. Pas daarna mag ik het pushen maar mijn stagebranch. Deze stagebranch zal op het einde van mijn afstuderen gemerged worden naar de master.

PLANNING

De algemene planning zal er als volgt uit gaan zien:

Sprint	Week	Thema	Op te leveren producten
Sprint 0	Week 1 - 3	Het opzetten van SCRUM	• Plan van aanpak • Requirementsanalyse document • Documenten over de opzet van SCRUM • PRODUCT BACKLOG N.A.V. DE REQUIREMENTS
Sprint 1	Week 4 - 6	Het maken van het nieuwe ontwerp en het onderzoeken van de bestaande codebase. ~~Het implementeren van de~~	• Ontwerprapport

Formatted: Font: Bold, Dutch (Netherlands)
Formatted: List Paragraph, Justified
Formatted: Justified
Formatted Table
Formatted: Justified
Formatted: Subtitle, Justified, Space Before: 5 pt, Don't add space between paragraphs of the same style, Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm
Formatted: Justified
Formatted: Justified, Space After: 0 pt, Line spacing: single, Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm
Formatted: Subtitle, Justified

		~~vernieuwing van het ontwerp in Angular~~	~~*Een werkende implementatie van het ontwerp~~
Sprint 2	Week 7 - 10	Het implementeren van de server-side versnelling n.a.v. het ontwerp	• Implementatie van het ontwerp in Angular met versnelling aan de server-side.
Sprint 3	Week 11 - 13	Het implementeren van de client-side versnelling n.a.v. het ontwerp	• Implementatie van het ontwerp in Angular met versnelling aan de client-side
	Week 14 - 17	Het afronden van de scriptie	• VOLLEDIGE SCRIPTIE

~~requirementse~~

Formatted: Justified

Formatted: Justified

Formatted: Subtitle, Justified

Formatted: Justified

Formatted: Subtitle, Justified, Space Before: 5 pt, Don't add space between paragraphs of the same style, Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm

LITERATUURLIJST

David, G. (2017, 06 30). *SCRUM Framework*. Retrieved from assist-software.net: https://assist-software.net/blog/scrum-framework-roles-activities-and-artifacts

H, K. (2016, 05 11). *Why a GRC*. Retrieved from Scripted.com: https://www.scripted.com/writing-samples/why-a-governance-risk-and-compliance-c6b37965-433b-4403-a9c4-e219ea58e9c9

BIJLAGE 4: REQUIREMENTS DOCUMENT

In dit document zullen de stappen doorlopen worden die ervoor gezorgd hebben dat de requirements achterhaald zijn. Deze requirements waren vervolgens geprioriteerd volgens de MoSCoW-methode en opgenomen in de product backlog en sprint backlog.

VERGAREN VAN DE REQUIREMENTS

Om requirements zo correct en zo volledig mogelijk te formuleren moet dit onderzocht worden. Hiervoor moeten er eerst een aantal vragen beantwoord worden voor de opzet van het onderzoek:

- Wat is de onderzoeksvraag
- Wat is de soort/ karakter van het onderzoek
- Hoe zal de data verzameld worden
- Welke data zal er geanalyseerd worden
- Hoe wordt de data geanalyseerd

- **Wat is de onderzoeksvraag:**

Het onderzoek zal uitgevoerd worden met de gedachte dat er requirements achterhaald moeten worden. Om de juiste requirements te achterhalen, moet er gekeken worden naar wat er op dit moment goed en minder goed staat. Wat ervaren de gebruikers als positief en wat als negatief aan de navigator? Doordat deze vraag gesteld wordt, kunnen de antwoorden naast elkaar gelegd worden en er verbeteringen komen voor het nieuwe ontwerp in de vorm van requirements.

Hierdoor zal de onderzoeksvraag als volgt luiden:

Wat zijn de zowel de negatieve als de positieve aspecten voor het gebruik van de huidige business dimension navigator?

- **Wat is het soort onderzoek:**

Er moet een keuze gemaakt worden tussen een kwalitatief of kwantitatief onderzoek. Dit legt Baarda, Ben (2014) als volgt uit *(zie figuur 3.3)*

figuur 3.3: kwalitatief versus kwantitatief onderzoek

Kwalitatief onderzoek	**Kwantitatief onderzoek**

Formatted: Justified

Formatted: Justified, Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm

Formatted: Justified

Formatted: Font: Bold

Formatted: Justified, Indent: Left: 1,9 cm, Space After: 7,75 pt, Line spacing: Multiple 1,1 li, Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm

Formatted: Justified

Formatted: Font: Italic, Dutch (Netherlands)

Formatted: Font: Bold, Dutch (Netherlands)

Formatted: Justified, Indent: Left: 1,9 cm, Space After: 7,75 pt, Line spacing: Multiple 1,1 li, Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm

Formatted: Justified

Formatted: Justified

Formatted: Font: Bold

'Bij kwalitatief onderzoek achterhaal je diepgaande informatie over motivaties, gedachten en verwachtingen van mensen. Kwalitatief onderzoek draait om woorden en niet om cijfers.'	*'Met kwantitatief onderzoek verkrijg je cijfermatige inzichten over mensen. Het geeft antwoorden op vragen die in hoeveelheden kunnen worden uitgedrukt.'*

Hierdoor is de keuze gevallen op een kwalitatief onderzoek, omdat er nog niet duidelijk is wat er allemaal aan de nieuwe navigator aangepast moet worden. Er moet dus nog voldoende informatie achterhaald worden, zodat de requirements opgesteld kunnen worden.

Karakter onderzoek:

Naast het soort onderzoek moet het karakter van het onderzoek ook bepaald worden. Er zijn drie soorten karakters volgens Baarda, Ben (2014) Zie figuur 3.4:

figuur 3.4: soorten onderzoek

Beschrijvend onderzoek	**Explorerend/ verkennend onderzoek**	**Verklarend/ toetsend onderzoek**
"Beschrijvend onderzoek is onderzoek waarbij het gaat om registratie en systematische ordening van datgene wat zich voordoet op een bepaald gebied, waarbij niet wordt gestreefd naar de ontwikkeling van een theorie of het formuleren van een hypothese.."	"Explorerend/verkennend onderzoek is onderzoek dat frequenties, samenhangen en verschillen exploreert met als doel om tot een theorie te komen."	"Toetsend onderzoek is onderzoek waarin je toetst of je steun kunt vinden voor een van tevoren geformuleerde verwachting, meestal een hypothese die is gebaseerd op een theorie"

Hierdoor is het karakter van het onderzoek gevallen op een **explorerend/ verkennend onderzoek.** Dit komt doordat er nog vastgesteld moet worden wat precies het probleem is van de huidige navigator en wat de eventuele verbeteringen kunnen zijn. Dit zal verkend worden om tot de requirements te komen.

Formatted: Font: Arial, 11 pt, Dutch (Netherlands)
Formatted: Font: Arial, 11 pt, Dutch (Netherlands)
Formatted: Justified
Formatted: Font: Arial, Dutch (Netherlands)
Formatted: Normal, Justified, Indent: Left: 0,63 cm, Tab stops: Not at 0,63 cm
Formatted: Justified
Formatted: Justified
Formatted Table
Formatted: Justified
Formatted: Justified

- **Hoe zal de data verzameld worden:**

  Er is gekozen voor het verzamelen van de data middels het houden van een **interview** en het **observeren** van de huidige business dimension navigator. Een interview is gekozen, omdat er verschillende testers zijn, zoals Ruben en Dart die de applicatie van top tot teen kennen en zij dus ook genoeg over de navigator weten en wat de klanten eventueel dwars zitten bij het gebruik hiervan.

  Daarnaast is de observatie gekozen, zodat ik de huidige business navigator pagina ook zelf kan zien, hierdoor kan navigeren en hierdoor mijn eigen oordeel kan vellen over de huidige pagina en de werking hiervan.

- **Hoe wordt de data geanalyseerd**

  Op basis van de uitvoer van het interview en de observaties zal de deze informatie geanalyseerd worden, en zal er hier een conclusie uit getrokken worden. Dit zal vervolgens in een meeting met Ruben en Dart besproken worden, zodat samen met het team de requirements opgesteld worden.

HET INTERVIEW

Nu de basis voor het onderzoek vaststaat, is er bekend dat er een **interview** uitgevoerd zal worden. Hieronder zal beschreven worden wat uit het interview naar boven kwam

Het interview is gehouden met Ruben en Dart samen. Zij zijn de testers van de CERRIX-applicatie en daarnaast zijn zij ook consultants en hebben hierdoor direct contact met de klanten van CERRIX. Hierdoor zijn zij de geschikte kandidaten voor het interview.

DE VRAGEN

Om de juiste vragen te bedenken, is er gekeken naar de huidige navigator en hoe dit de klanten zowel negatief, als positief zou beïnvloeden.

Er waren een aantal vragen gesteld, deze waren o.a.:

- Waar gebruiken de klanten de business navigator voor?
- Wat voor klanten gebruiken deze module?
- Wat is de positie van deze klanten binnen hun eigen organisatie
- Wat is er al bekend over de ervaring van de klanten bij het gebruik van de module?
- Wat vinden de klanten fijn aan de huidige navigator?
- Wat vinden de klanten minder fijn aan de navigator?

Formatted: Font: Bold, Dutch (Netherlands)

Formatted: Justified, Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm

Formatted: Font: Bold, Dutch (Netherlands)

Formatted: Font: Bold, Dutch (Netherlands)

Formatted: Justified

Formatted: Font: Bold

Formatted: Justified, Bulleted + Level: 1 + Aligned at: 0,63 cm + Indent at: 1,27 cm

Formatted: Justified, Tab stops: Not at 0,63 cm

Formatted: Justified, Indent: Left: 0,02 cm

Formatted: Normal, Justified

Formatted: Justified, Space After: 7,75 pt

Formatted: Heading 3, Justified

Formatted: Justified, Space After: 7,75 pt

Formatted: Normal, Justified

Formatted: Space After: 7,75 pt, Tab stops: Not at 0,63 cm

Wat uit het interview voortkwam was o.a. het volgende:

De klanten gebruiken de Business Navigator om op één plek een overzicht te krijgen van hun bestaande bedrijfsproces(sen). Dit hebben zij nodig, omdat hun organisaties opgesplitst zijn in meerdere (tot wel 200 bij sommige klanten) bedrijfsprocessen. De navigator biedt deze klanten een centraal punt binnen de applicatie, waarbij de risico's, controls, events, mois en data processings getoond worden. Dit hebben zij nodig, zodat ze hun processen kunnen verantwoorden naar hun klanten toe en zodat de directie inziet wat voor koppelingen de bedrijfsprocessen hebben met de rest van de applicatie. Hierdoor voelen zij zich 'in control' en hebben zij een duidelijk beeld over de beheersing van hun bedrijfsprocessen.

Elke medewerker heeft toegang tot deze module, zodat iedereen in kan zien welke koppelingen er zijn binnen hun bedrijfsproces. Echter de data die hierbinnen weergeven wordt, wordt gefilterd aan de hand van de rollen van de gebruikers.

De verbetering die de klanten zelf willen zien in de navigator, is dat zij het laden van de data zelf ook versneld willen hebben. Zij vinden ook dat het te lang duurt voordat er uitvoer op het scherm weergeven wordt. Daarnaast vinden zij dat de filter voor het kiezen van de organisatie en business dimension op een onjuiste manier weergeven wordt. Het filter zit middels een inklapsysteem moeilijk in elkaar, waardoor het gebruik van de pagina niet begrepen wordt.

Daarnaast vinden de klanten het fijn om alle informatie op 1 centraal punt te kunnen zien. De doorklikfunctionaliteit vinden zij ook fijn, omdat zij hierdoor direct naar de details van hun risks/events etc. kunnen gaan. Wat natuurlijk verbeterd kan worden is de performance van de pagina en het onhandige ontwerp.

Er vond een meeting plaats voor het achterhalen van de wensen van Paul, Ruben en de overige developers voor de functionaliteit voor de nieuwe business dimension navigator (zie hieronder). Hieraan werden er uren gekoppeld. Deze uren laten zien hoeveel uren het implementeren hiervan grofweg vergt.

De functionaliteit werd opgesplitst in 6 grote delen:

- De omschrijving naar Angular
- Het tonen van de linked items
- De rollen en rechten die bij de overzichten komen kijken
- Het maken van rapportages
- Overige algemene functionaliteit

Gelet werd er op de tijd: In totaal zijn er **320** te besteden uren. (10 weken * 32 uur)

Dit is berekend door: **10** weken voor de implementatie (na de 5 weken voor het onderzoeken en analyseren van het probleemdomein en 5 weken voor het afronden van de scriptie) *** 32** (uren per week dat er gewerkt kan worden aan de opdracht (4 dagen in de week)) = **320** te besteden uren.

Hieruit kwam een tabel met de gewenste functionaliteit (requirements) en een schatting van hoeveel uren dit zou duren. Zie tabel 3.5: Tijdsinschatting MoSCoW

Tabel 3.5: Tijdsinschatting van de wensen

Te besteden uren	320
Omschrijven van Navigator business dimensions naar Angular techniek	
Onderzoek performance server side	40
Ontwerp nieuwe scherm	40
Filters Organizations-Business Dimensions(cliënt side en server side)	80
Risk tabel	8
Control tabel	8
Event tabel	8
MoI Tabel	8
Data Processing tabel	8
KRI tabel	8
KRI graphs	8
Openen van items	8
Risk matrix	8
Linked items	
Per risico	8
Per control	8

Per event	8
Per MoI	8
Per Data processing	8
Per KRI	8
Openen van items	2
Rollen en rechten navigator business dimensions	
Tonen tabel per module	8
Inhoud koppelen aan rollen/rechten	8
Openen detailschermen op basis van rechten/rollen	4
Rapprtage	
Reports	40
Exports	24
General	
Documents	8
History	24
Totaal	390

Hieruit valt op te merken is dat er 390 uren geschat zijn voor de totale vervulling van de wensen, terwijl er maar 320 te besteden uren zijn. Hierdoor zijn de wensen voor het maken van de rapportages en het weergeven van de linked items geschrapt en komt het totaal aantal uren nu uit op **320**. Dit moet ongeveer haalbaar zijn.

Deze wensen zijn vervolgens geprioriteerd volgens de MoSCoW methode om te bepalen wat er eerst uitgevoerd moest worden.

MOSCOW

De MoSCoW-methode is een wijze voor het prioriteren van requirements. Doordat de wensen die tijdens de meeting afkwamen onderverdeeld werden in de MoSCoW-methode kon er gekeken worden welke wensen daadwerkelijk haalbaar zijn om een minium viable product te leveren, en welke dus, middels de tijdsinschatting daadwerkelijk haalbaar waren om te implementeren binnen het afstudeertraject.

De MoSCoW-methode bestaat uit requirements die verdeeld zijn onder de volgende prioriteit:

Must have:	Deze requirements moeten in het eindresultaat terugkomen. Zonder deze eisen is het product onbruikbaar. (Minimum viable product)
Should have	Deze requirements zijn zeer gewenst, maar zonder is het product toch wel bruikbaar
Could have:	Deze requirements zullen alleen aan bod komen als er tijd genoeg is om ze te implementeren
Won't have this time:	Deze eisen zullen in dit project waarschijnlijk niet aan bod komen, maar in de toekomst wel.

Formatted: Font: Bold

De hierboven genoemde requirements zijn, gelet op de tijdsinschatting opgenomen in het MoSCoW-model:

Must have

- Het systeem moet de data kunnen filteren op basis van organization en business dimension
- Het systeem moet de KRI-data kunnen weergeven
- Het systeem moet de risk-data kunnen weergeven
- Het systeem moet de control-data kunnen weergeven
- Het systeem moet de MoI-data kunnen weergeven
- Het systeem moet de data processings- data kunnen weergeven
- Het systeem moet de event-data kunnen weergeven
- Het systeem moet de losse detail pagina van een (risk/control/event etc.) kunnen openen
- Het systeem moet een gross/ net risico matrix tabel kunnen tonen

Formatted: Font: Bold

Should have

- Het systeem moet de documenten die horen tot een business dimen tonen

Could have

- Het systeem met een KRI grafiek kunnen tonen

Won't have this time

- Het systeem moet een excel uitdraai kunnen maken van de review scor
- Het systeem moet een raportage kunnen tonen van de business dimension, met gekoppelde data
- Het systeem moet de gelinkte items tonen van de risks
- Het systeem moet de gelinkte items tonen van de controls
- Het systeem moet de gelinkte items tonen van de events
- Het systeem moet de gelinkte items tonen van de mois
- Het systeem moet de gelinkte items tonen van de kris
- Het systeem moet de gelinkte items tonen van de data processings
- Het systeem moet een afdruk functionaliteit kunnen hebben

Deze requirements zijn vervolgens vertaald naar user stories, die voldoen aan de Definition of Ready.

Formatted: Font: Bold

Formatted: Subtitle, Space After: 8 pt, Line spacing: Multiple 1,07 li

Formatted: Font: Bold

Formatted: Font: Bold

PRODUCT BACKLOG

De requirements die uit het MoSCoW-model kwamen zijn omgescreven tot user stories en meegenomen in de product backlog. De backlog items zijn voorzien van de MoSCoW priority en hebben daarnaast nog een estimated tijd in uren.

# User Story ID	Priority (H – L)	User story	Est. time
U01	H	As < a developer> I want< to research the performance server side >, < so that I can see how fast this is >	40
U02	H	As < a developer>I want< to create a new design for the webpage >, < so that I know what to implement >	40
U03	H	As <a developer> I want to test whether Angular is the correct chosen framework	16
U04	H	As < a Developer > I want < to have an Angular component analysis based on the design> so < that I know which Angular components to (re)use >	40
U05	H	As < a Developer > I want < to know how the Codebase works > so < that I know how the system is set up >	40
U06	H	As < an overviewer > I want < to see my Controls according to the **server-side filter** - when i apply my configuration> so <that I can get an overview>	12
U07	H	As < an overviewer > I want < to see my Mois according to the **server-side filter** - when i apply my configuration> so <that I can get an overview>	12
U08	H	As < an overviewer > I want < to see my Data Processings according to the **server-side filter** - when i apply my configuration> so <that I can get an overview>	12
U09	H	As < an overviewer > I want < to see my KRIs according to the **server-side filter** - when i apply my configuration> so <that I can get an overview>	12

U10	H	As < an overviewer > I want < to see my Risks according to the **server-side filter** - when I apply my configuration> so <that I can get an overview>	12
U11	H	As < an overviewer > I want < to see my Events according to the **server-side filter** - when i apply my configuration> so <that I can get an overview>	12
U12	H	As < an overviewer > I want < to clear my configuration> so <that I can filter on another configuration>	3
U13	H	As < an overviewer >I want< to see the details page of the item I clicked on >, < so that I can get redirected to the details page of the item>	8
U14	H	As < an overviewer >I want< to see the Risk Matrix>, <so that I can see further details about the Risks >	8
U15	H	As < an overviewer > I want < to see my Controls according to the **client-side filter** - when i apply my configuration> so <that I can get an overview>	12
U16	H	As < an overviewer > I want < to see my Events according to the **client-side filter** - when i apply my configuration> so <that I can get an overview>	12
U17	H	As < an overviewer > I want < to see my Risks according to the **client-side filter** - when i apply my configuration> so <that I can get an overview>	12
U18	H	As < an overviewer > I want < to see my Mois according to the **client-side filter** - when i apply my configuration> so <that I can get an overview>	12
U19	H	As < an overviewer > I want < to see my Data Processings according to the **client-side filter** - when i apply my configuration> so <that I can get an overview>	12

U20	H	As < an overviewer > I want < to see my KRIs according to the **client-side filter** - when i apply my configuration> so <that I can get an overview>	12

SPRINT BACKLOG

De product backlog items zijn toegewezen aan de sprints die hetzelfde thema hebben. Hierdoor zal het sprint thema bereikt worden d.m.v. de uitvoering van de toegewezen user stories.

Sprint	**Toegewezen user stories**
Sprint 1: Het maken van het nieuwe ontwerp en het onderzoeken van de bestaande codebase.	U01, U02, U03, U04, U05
Sprint 2: Het implementeren van de server-side versnelling n.a.v. het ontwerp	U06 t/m U13
Sprint 3: Het implementeren van de client-side versnelling n.a.v. het ontwerp en de linked details	U14 t/m U20

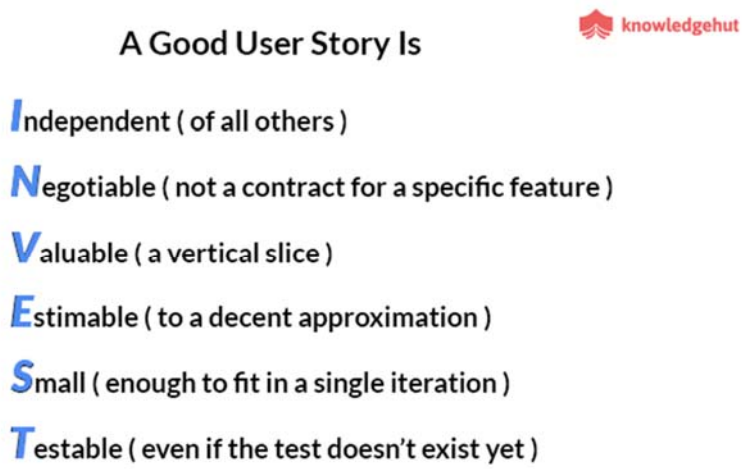
LITERATUURLIJST

Baarda, B. (2014). *Dit is onderzoek! incl. toegang tot Prepzone* (2de editie). Den Haag: Noordhoff.

BIJLAGE 5: DEFINITION OF READY

De requirements moeten voldoen aan een bepaald template, zodat zij makkelijk opgepakt kunnen worden. De definition of ready legt uit aan welke voorwaardes zij moeten voldoen. De user stories die gemaakt worden a.d.h.v. de voorwaardes die in de definition of ready staan moeten volgens het standaard SCRUM model als volgt worden opgebouwd: *(zie figuur 1.1)*

figuur 1.1: Opzet van een goede user story (naar A good user story (z.d.)). voldoen aan het 'INVEST'-model:



Het 'INVEST'-model legt uit dat de user story opgebouwd moet zijn uit het volgende: (volgens Scrum User stories (z.d.)) zie figuur 1.2

Figuur 1.2 INVEST - uitgelegd

Independent	Losstaand van elkaar zijn. Zonder al te veel samenhang, zodat de user story niet te gecompliceerd wordt als het te veel onderliggende relaties krijgt.
Negotiable	Het moet mogelijk zijn om te discussiëren over user stories. Ze staan niet vast, omdat de story altijd moet kunnen veranderen indien nodig.

Valuable	De user story moet toegevoegde waarde bieden aan de gebruiker.
Estimable	Er moet een tijdsinschatting in uren worden toegevoegd aan de story. Hierdoor kan het team de user story een prioriteit geven voor de implementatie.
Small	De user story moet niet te groot zijn, zodat het in de sprint werkelijk uitgevoerd kan worden.
Testable	De juiste acceptatiecriteria moeten vastgesteld worden per story, zodat zij gemakkelijk getest kunnen worden

Dit zal toegepast worden op de user stories. Hierdoor zullen de user stories aan de volgende voorwaardes moeten voldoen:

1. Een user story zal opgebouwd worden in het volgende format:
   As a <type gebruiker>, I want to <do something>, so that I <can achieve this>.
2. De acceptatiecriteria per user story moeten opgesteld en begrepen worden door het team
3. De user story moet een tijdsinschatting krijgen
4. De user story moet een prioriteit inschatting krijgen.
5. De user story moet klein zijn.
6. De user story moet niet te gecompliceerd zijn (weinig onderliggende relaties bevatten)

LITERATUURLIJST

A Good User Story. (z.d.). [Foto]. Geraadpleegd van https://s3.amazonaws.com/mobtvse_masukomi/2014_2_20_user_story_decomposition.png

Scrum User Stories. (2015, 10 juni). Geraadpleegd van https://www.knowledgehut.com/tutorials/scrum-tutorial/user-stories

BIJLAGE 6: DEFINITION OF DONE

In de Definition of Done staan er voorwaardes, waaraan het increment moet voldoen om ervoor te zorgen dat de kwaliteit van het uitgevoerde werk gewaarborgd blijft. Als de wekelijkse increments aan deze voorwaardes voldoen dan is het zeker duidelijk dat wat er gemaakt is echt zal werken en dat zowel CERRIX als de developers die mijn werk zullen controleren hier tevreden mee zullen zijn. Zo bestaat de D.O.D. uit de volgende eisen:

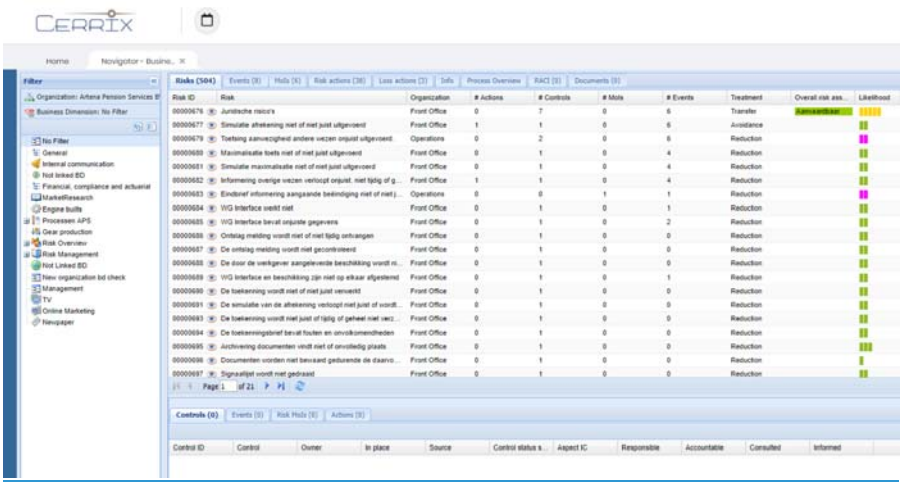
- Alle testen moeten voltooid zijn of in ieder geval uitgevoerd (unit/ui/performance). Als ze niet voltooid zijn, dan moeten ze aangepast worden en opnieuw uitgevoerd worden.
- Er is gecodeerd a.d.h.v. de acceptatiecriteria en het werk voldoet hie rook aan
- Er heeft een code review plaatsgevonden op het werk
- De automatische testen van CERRIX zelf slagen (Continuous Integration)
- Het werk voldoet aan wat er vermeld is in de user stories
- Het werk is gepusht op de servers van CERRIX

BIJLAGE 7: ONTWERPDOCUMENT

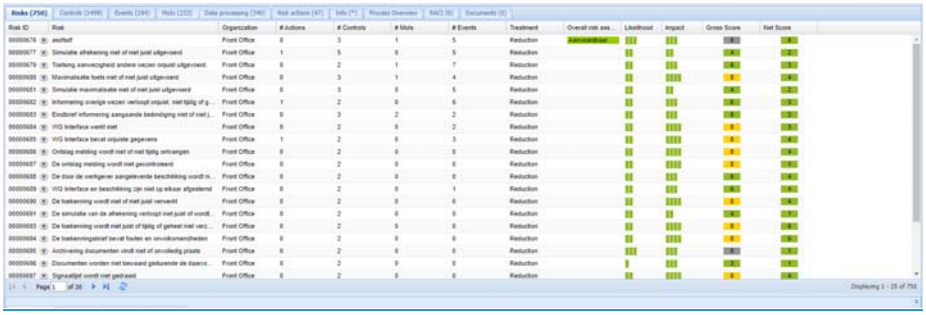
In dit document zal de totstandkoming van het nieuwe ontwerp uitgelegd worden. Er wordt beschreven welke bijdrage ik zelf heb geleverd aan het nieuwe ontwerp, welke overige wensen er zijn meegenomen en hoe het nieuwe ontwerp eruit zal gaan zien. Ook zal er een technische analyse plaatsvinden om inzicht te krijgen op de bestaande codebase, zodat het probleem met de traagheid van de performance van de navigator beter geanalyseerd kan worden. Dit zal worden ondersteund met diverse UML en sequentiediagrammen. Tot slot zal er een server-side als client-side oplossing geboden worden voor het probleem van de performance van de navigator business dimension.

ONTWERP

Het was uit de requirements duidelijk dat de Navigator op een aantal punten qua ontwerp verbeterd moest worden. Zo was het filter ontworpen door een inklap systeem (zie figuur 1). Daarnaast was er een overvloed aan informatie te zien op het scherm. De lezer zwom erin. Dit betekende dat de kolommen die weergegeven werden op het scherm herzien moesten worden, omdat een de risicotabel bijvoorbeeld al 14 attributen liet zien. Ik besloot om de attributen die de tabellen lieten zien te herzien, omdat ik hierdoor door de zes modules heen zou moeten gaan en ik beter zou begrijpen waar een risk, control, moi etc. voor dienen in de applicatie en welke koppelingen ze verder hebben, waardoor ik de CERRIX applicatie beter zou begrijpen. Figuur 2 laat zien hoe de huidige risicotabel eruit ziet.

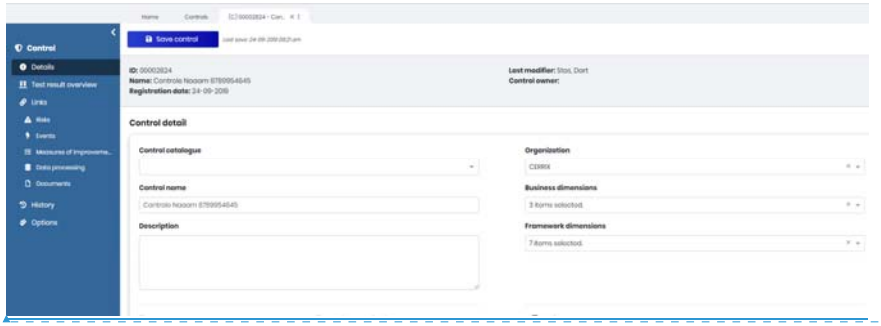


Figuur 1 Huidige Business dimension navigator pagina. Verouderd en aan omschrijving toe



Figuur 2 Het huidige risicotabel dat weergeven wordt in de Navigator. Hier wordt er te veel informatie getoond.

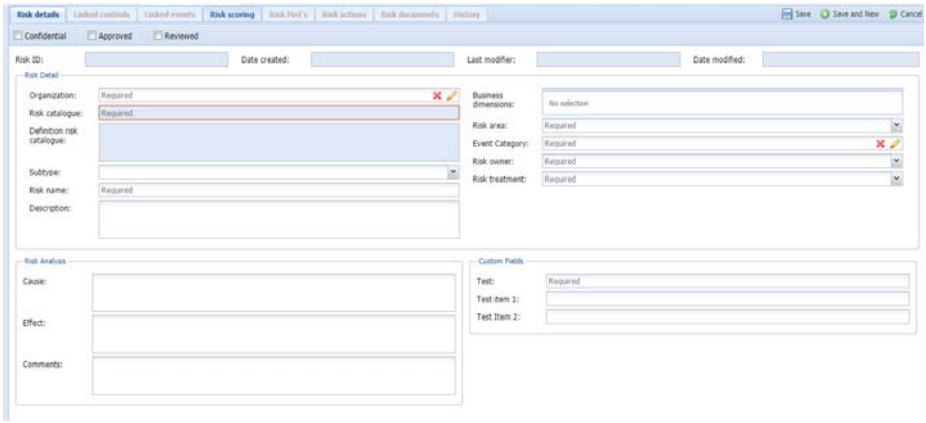
Het nieuwe ontwerp voor de filters in hoefde ik zelf niet te veranderen, omdat er door de applicatie heen al nieuwe Angular filters worden gebruikt en dat design zou dan vervolgens overgenomen worden. Ook het tabje voor de documenten hoef ik niet te ontwerpen, omdat dit aan de linker kant van de blauwe navigeerbalk zou komen te staan, net zoals de andere nieuwere Angular pagina's, zoals de Control detail pagina (zie figuur 3).



Figuur 3 De control detail pagina geschreven in de Angular huisstijl. Op de pagina is een links een navigeerbalk te zien, waar ook het knopje voor de documenten staan en de Navigator zal over dezelfde balk beschikken.

Formatted: Font: Italic

Wat ik moest doen om de nieuwe kolommen te bedenken was om allereerst te begrijpen wat alle kolommen betekenden. Hiervoor heb ik naar de Risk, Control, Event, Moi, KRI en Data Processing module gekeken en de pagina voor het creëren van een lege item bestudeerd. (Zie figuur 4 voor de pagina voor het creëren van een risico)



Figuur 4: De velden die ingevoerd moesten worden om een risico aan te maken. Er moest een selectie gemaakt worden om de belangrijkste kolommen te kiezen voor de weergave in de navigator.

Op de pagina zijn er een flink aantal attributen te zien. Ik ging er van uit om maximaal acht kolommen te maken in het nieuwe ontwerp, omdat dat nog overzichtelijk zou zijn om op het scherm weer te geven. Hierdoor zou het scherm dus nog leesbaar zijn. Om de juiste kolommen uit te kiezen moest er eerst begrepen worden waar de navigator voor gebruikt zou worden. Wat verwachten gebruikers als ze de Navigator business dimension openen?

De navigator is de plek waar een gebruiker de noodzakelijke informatie hoort te zien over zijn processen en de gelinkte items die hierbij horen (dus een risk/ event/ control/ moi dat gekoppeld is aan een business dimension). Als een gebruiker dit verwacht, dan hoef ik in het nieuwe overzicht bijvoorbeeld geen 'Risk description' te tonen, omdat er algemene informatie m.b.t. het ' in control ' houden van de processen verwacht wordt.

Ter illustratie: als wij de belangrijkste gegevens nodig hebben van een student, hebben wij niet zijn of haar telefoonnummer nodig, maar zijn klas/ ID/ opleiding etc. Hetzelfde geldt voor de informatie die de Navigator hoort te tonen. Als de overige informatie, zoals het telefoonnummer toch nodig is, dan kan de gebruiker dubbelklikken om verder naar de details pagina verwezen te worden.

Zo heb ik dus naar de kolommen gekeken die er op dit moment weergeven werden en bleek het dat er wel belangrijke kolommen weergeven werden die ik kon laten staan en er een paar waren die ik kon wegstrepen, dit heb ik nagevraagd aan Dart en Ruben, of zij dit ook vonden en hierdoor zag het risicotabel er als volgt uit (zie tabel 5).

Tabel 5: De uiteindelijke kolommen die weergeven zouden worden in het risicotabel.

Risk ID	Door de applicatie heen kwamen overal de ID's terug. Hetzelfde verwachtte ik dan ook voor de risico's in de navigator
Risk Name	Uiteraard moet het duidelijk zijn over welke risico het gin gen moest de naam terug komen op het scherm
Organization	Hiermee konden de risico's getypeerd worden op organization, aangezien de navigator hierop filterde
Risk owner	Dit is de risicoeigenaar en is belangrijk om meet e nemen omdat er naar deze risicoeigenaar toegegaan moet worden als het risico voorkomt
Gross score	Dit is de bruto score van het risico was hierom een belangrijk detail voor overviewers om naar te kijken.
Net score	Dit is de netto score van het risico en was hierom wederom een belangrijk detail voor de overviewers om naar te kijken
Risk assessment	Dit is het resultaat van het risico, of het wel of niet uitgevoerd werd of aanvaardbaar was. Een overviewer zou dit handig vinden, omdat hij zo de score van de risico kon zien.
Risk appetite	Dit was ook een scoring van de risico op basis van de risk appetite (risicobereidheid). Overviewers zouden dit handig vinden om te zien, zodat ze hierdoor de bereidheid van de organisatie zien om het risico te beheersen.

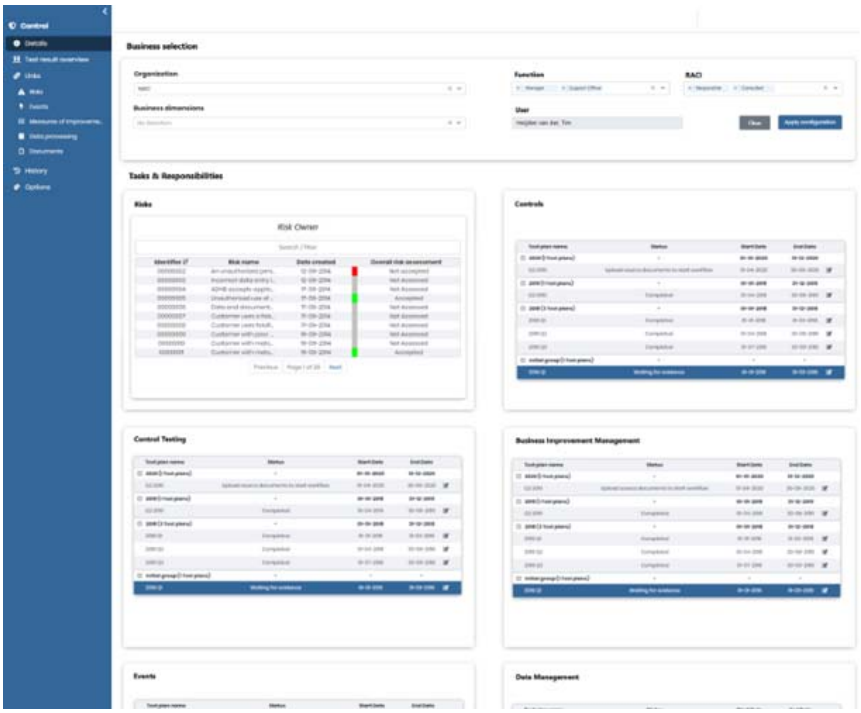
Dit is ook voor de overige modules gedaan, waardoor de volgende kolommen weergeven zullen worden in het nieuwe ontwerp (zie tabel 6).

Tabel 6: De uiteindelijke kolommen voor alle zes modules die in het ontwerp zijn meegenomen.

Module	Kolommen (Verplicht)	Overige kolommen				
Risk	ID, Name, Organization	Risk owner	Owner	Gross score	Net score	Appetite
Controls	ID, Name, Organization	Control owner	Effectiveness score	In place	Control period score	Type
Events	ID, Name, Organization	Status	Progress	Date detected	Responsible	
Mois	ID, Name, Organization	Status	Progress	Users responsible	MoI Type	
KRIs	ID, Name, Organization	KRI Frequency	Latest data point			
Processing Purposes	ID, Name, Organization	Structures				

HET NIEUWE ONTWERP

Na mijn input voor de herziene kolommen werd het ontwerp grafisch samengesteld door Ruben. Verder waren hier nog wensen van Paul in opgenomen, over de lettergrote, grootte van de tabellen (deze zijn nu 50% van het scherm, maar misschien wordt dit toch later groter, omdat dat mooier oogt). In combinatie met deze wensen is het nieuwe ontwerp tot stand gekomen (zie figuur 7). Dit zal het nieuwe ontwerp voor de Business dimension navigator worden, maar is nog een concept. Het kan nog veranderen, omdat het, eenmaal geïmplementeerd misschien anders kan ogen.



Figuur 7 Het nieuwe ontwerp van de Business dimension navigator. Op de afbeelding zijn de zes nieuwe tabellen te zien van de Risk, Control, Moi, Event, KRI en Data Processings module. Daarnaast is er op de linker navigatiebalk een tab voor de documenten te zien.

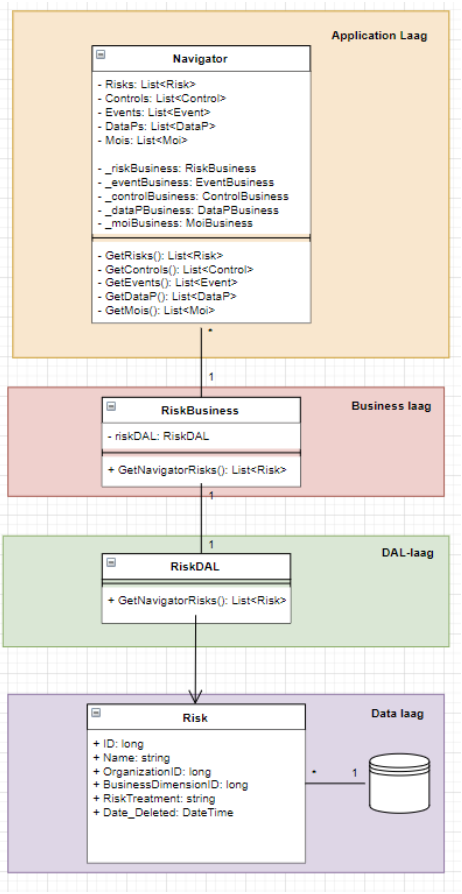
CODE TECHNISCH

De CERRIX-applicatie is opgebouwd d.m.v. een vier-lagenmodel. Dit betekent dat een aanvraag aan de voorkant door vier lagen moet gaan om de database te bereiken, deze vervolgens dan weer terug moet sturen voordat er uitvoer te zien is op het scherm.

Zo is er de Application laag, in deze laag staan de views die weergeven worden op het scherm. Deze laag staat in direct contact met de Business laag. Hierin staat er businesslogica voor het opvragen en manipuleren van data. Deze data komt uit de Data Access Laag, waarin functies staan om data op te halen uit de Database. De Database en haar modellen staan in de DATA-laag. Om dus een aanvraag te doen vanaf een webpagina moeten deze vier lagen eerst doorlopen worden om de database te bereiken, en moeten zij vervolgens weer teruggestuurd worden alvorens er uitvoer is op het scherm.

ONTWERP VAN DE HUIDIGE SITUATIE

Om een beter beeld te schetsen over hoe de vier lagen met elkaar samenwerken, wordt er hieronder een UML-diagram weergeven van de huidige situatie van de Business dimension navigator, voor het opvragen van de riskdata. (Zie figuur 8)



In de application-layer staat de huidige view (webpagina) van de Navigator. Hier roept de Navigator code om de riskdata weer te geven. Deze functie staan in de RiskBusiness.

In de business-layer zit er de functie GetNavigatorRisks(), die opgevraagd wordt aan de DAL-layer. De toegevoegde waarde van de business-layer is dat deze data de vorm van een riskmodel krijgt.

In deze DAL-layer zitten er functies in die de daadwerkelijke database-objecten ophaalt.

De DAL-layer staat direct in contact met de DATA-laag, waar de database en risktabel in staan.

Nadat alle lagen van boven naar beneden zijn afgegaan, wordt de data natuurlijk terug naar boven gestuurd, zodat het weergeven kan worden op de Navigator-pagina.

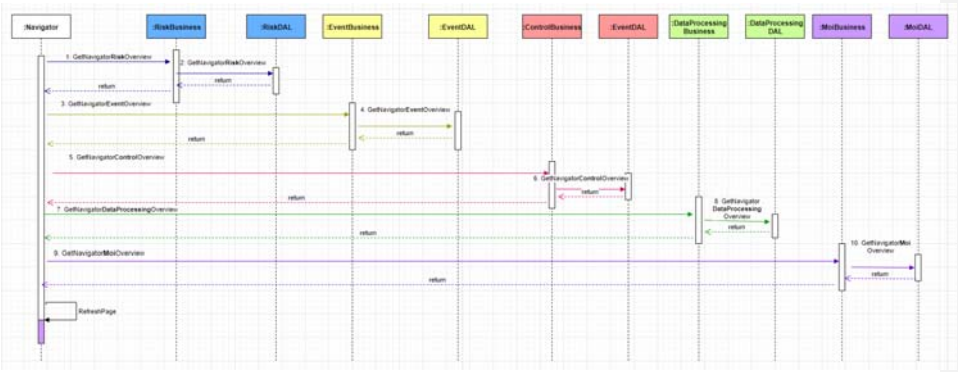
Nu er een UML-diagram is gemaakt snap ik de werking beter tussen de gelaagdheid van de applicatie.

Figuur 8 UML-diagram van de huidige navigator voor het opvragen van de riskdata. Op de afbeeldig is de communcatie tussen de vier lagen weergeven.

TRAAGHEID VAN DE NAVIGATOR

De data van alle zes modules die op het scherm weergeven wordt, wordt op dit moment **sequentieel** opgehaald. Dit houdt in dat er tijdens het laden van de pagina (alvorens er al een filter op organisatie en bedrijfsproces is toegepast) allereerst alle risico's worden opgehaald uit de database, pas nadat de risico's binnen zijn, de controls, events, mois, en data processings worden opgehaald. Er wordt dus telkens gewacht totdat de data per module is binnengekomen. Dit zorgt ervoor dat het laden van de pagina dus +/- 20 seconden duurt. Om deze gedachte te ondersteunen is er een sequentie diagram getekend (zie figuur 9), dat laat zien dat alle modules op elkaar wachten voor het ophalen van hun eigen data.

Wanneer het filter wordt toegepast op de data van de zes modules, dan worden al deze risks, controls, events, mois en data processings wederom opgevraagd vanuit de database, maar worden ze in de bijbehorende DALs gefilterd op basis van hun geselecteerde organisatie en proces. Dit is een heel erg slechte design, omdat er hierdoor maar één module per keer opgehaald en gefilterd kan worden, waardoor de business dimension navigator zo traag is.



Figuur 9 Sequentiediagram van de huidige situatie

Op de afbeelding valt er te zien dat de navigator eerst de Risks ophaalt, pas als de risico's terug in de navigator komen, de events worden opgehaald, daarna pas de controls, data processings en tot slot de mois. Dit zorgt ervoor dat het laden en filteren van de pagina zo lang duurt, omdat tijdens het filteren alle modules ook weer op elkaar wachten.

OPLOSSING VOOR DE VERSNELLING

Om dit op te kunnen lossen moeten alle modules dus niet wachten op elkaar om data op te kunnen halen (**sequentieel**), maar moeten deze calls tegelijkertijd, en dus **parallel** uitgevoerd worden. Dit betekent dat er een omschrijving van de code nodig is zodat dit kan gebeuren, zodat zowel het laden als het filteren tegelijkertijd gebeurd met de calls van de andere modules, wat de performance theoretisch al verbeterd.

Server-side vs client-side filteren van de data

Naast het parallel ophalen en filteren van de data, is het ook een requirement om te achterhalen welke implementatie voor het filteren van de data sneller is: server-side of client-side?

SERVER-SIDE

Het server-side filteren van de data betekent dat er nadat er geklikt wordt op de filter-knop, er een call naar de database gestuurd wordt om alle risico's op te halen en ze vervolgens te filteren op basis van de gekozen selectie. Op dit moment is dit al het geval, en zal ondersteund worden met figuren 10 en 11.

```
public List<RiskNavigatorModel> GetNavigatorRiskOverview(ServerSearchFilter<RiskSearchFilter> searchFilters)
{
    var riskData = RiskDAL.GetNavigatorRiskOverview(searchFilters);

    var riskMatrix = new OrganizationRiskMatrix( initDictionaries: true);

    return riskData.Select(x => MapToRiskNavigatorModel(x, new RiskNavigatorModel(), riskMatrix))
        .ToList();
}
```

Figuur 10 Huidige bestaande functie die de risks voor de navigator ophaalt.

Op de afbeelding valt er de functie te zien die de risico's voor de navigator ophaalt. Deze functie staat in de RiskBusiness en wordt aangeroepen vanaf de Navigator zelf. De functie heeft een searchFilter als parameter, waarin de ID's staan voor de organisatie en business dimension. Zie figuur 11 om te bekijken hoe deze risico's nader worden opgehaald vanuit de RiskDAL.

```
1 reference
public IList<ORM_Risk> GetNavigatorRiskOverview(ServerSearchFilter<RiskSearchFilter> searchFilters)
{
    return GetList(searchFilters, includeSlowlyChangingDimensions: false, pointInTime: null,
            params navigationProperties: x => x.ORM_Organization,
            x => x.ORM_Risk_Risk_Control,
            x => x.LER_Event_Risk,
            x => x.ORM_Risk_Treatment,
            x => x.ORM_Risk_Assessment,
            x => x.ORM_T_Impact,
            x => x.ORM_T_Impact_Net,
            x => x.ORM_T_Likelihood,
            x => x.ORM_T_Likelihood_Net,
            x => x.ORM_Risk_Moi,
            x => x.ORM_Risk_Actions
        );
}
```

Figuur 11 Huidige bestaande functie in de RiskDAL die de risico's opvraagt aan de database.

Op deze afbeelding vraagt de RiskDAL de riskdata op aan de database door de GetList() functie. Deze functie vraagt naast het searchFilter ook o.a. de koppeltabellen Organization, RiskControlm EventRisk, RiskAssessment etc. op, zodat deze dependencies ook geladen kunnen worden (lazy loading).

IMPLEMENTATIE VOOR SERVER-SIDE FILTERING EN VERSNELLING

Als het ophalen van de data voor alle zes modules parallel wordt aangeroepen, dan kan deze data server-side gefilterd, om te kijken of dit veel sneller gebeurd dan voorheen, omdat de data niet meer sequentieel wordt opgehaald.

De functies die de navigator op dit moment heeft voor het ophalen van de data uit de modules is nog op een oude wijze geschreven omdat er niemand in jaren aan de navigator gezeten heeft. Daarom moeten deze functies voor bijvoorbeeld het ophalen van de risks, controls, mois etc. omgeschreven worden zodat ze de functies gebruiken die de risk-module, control-module etc. op dit moment wordt gebruikt, omdat de developers van CERRIX deze functies al wat hebben versneld. Op deze data zal er vervolgens server-side filtering toegepast worden.

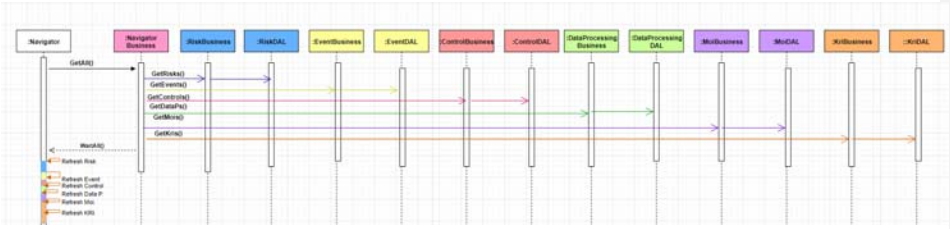
CLIENT-SIDE

Naast een server-side versnelling, waarbij de data telkens aan de achterkant opnieuw wordt opgevraagd en gefilterd, is het ook mogelijk om de data aan de voorkant (client-side, in de browser van de gebruiker) te filteren. Dit houdt in dat de data van alle modules client-side wordt opgeslagen (dit wordt nog steeds parallel opgehaald) en tijdens het filteren deze zelfde data gefilterd wordt op basis van de org en het process. Doordat de data van alle modules virtueel in de browser wordt opgeslagen (client-side), hoeft de data niet per keer dat het filtert opnieuw opgehaald worden vanuit de database, maar is het al in de browser opgeslagen en hoeft alleen deze data telkens gefilterd worden. Op dit moment hebeben de overige Angular pagina's van de risk/control/event modules al een client-side filtering implementatie en was het ook een requirement om dit te implementeren in de navigator, om te kijken of dit sneller is dan de server-side implementatie. Door zowel server-side als client-

side te implementeren weten wij zeker dat er coverage is op grond van deze twee vlakken voor de versnelling van de navigator.

VAN SEQUENTIEEL NAAR PARALLEL

Om het ophalen van de data van allle modules om te schrijven van de sequentiele wijze naar een parallelle wijze moet er eerst geschetst worden hoe wat voor werking dit dan zal hebben t.o.v. de huidige wijze. (zie figuur 12)



Figuur 12 Sequentie diagram voor de nieuwe parallelle implementatie

Op de afbeelding valt er te zien dat er een verschil is met het sequentiediagram van afbeelding 9. De aanvragen naar de database voor het ophalen van de data uit de zes modules gebeurd nu parallel. Dit betekent dat er niet langer gewacht hoeft te worden totdat één module zijn data geladen heeft voordat de andere module dit kan doen, maar dat alle modules dit tegelijkertijd gaan opvragen. Dit zal ervoor zorgen dat het laden sneller zal verlopen.

IMPLEMENTATIE PARALLEL OPHALEN VAN DE DATA

Voor het gemak zal alleen het ophalen van de risico's uitgelegd worden. Dezelfde implementatie kan er gebruikt worden voor de overige modules. Om de sequentiele wijze voor het ophalen van de risico's te herschrijven naar de parallele manier, moet er eerst begrepen worden hoe C# dit interpreert, dit is beschreven volgens ASP. NET (2019, 3 augustus):

Om parallelle functies te maken, dient er gebruik te worden gemaakt van multithreading. Dit houdt in dat C# een taak aan gaat maken per grote call/ functie. Deze taak bevat functionaliteit die afzonderlijk (parallel) van de andere code uitgevoerd zal worden. Dit kan, doordat C# automatisch aparte threads aanmaakt per Task. Om dit te implementeren zal er dus een Task gemaakt worden, waarin alleen staat dat de risico's opgehaald moeten worden. Voor het ophalen van de control-data zal er dus ook een nieuwe Task aangemaakt moeten worden, zodat deze taken allemaal afzonderlijk uitgevoerd zullen worden van elkaar en apart een resultaat zullen krijgen.

De implementatie voor het herschrijven van het ophalen van de data op een parallelle wijze komt terug in sprint 2. Dit zal daar verder ondersteund worden, tevens zal de implementatie van het ontwerp naar Angular ook in deze sprint behandeld worden en zal er laten zien hoe de filtering server-side is toegepast. Voor de implementatie van de client-side filtering en versnelling verwijs ik u door naar sprint 3.

DE EXT.NET CODE

Deze huidige risk-tabel is opgebouwd uit de volgende code, geschreven in Ext.Net (zie figuur 13):

```
<BaseParams>
    <coolite:Parameter Name="organizationID" Value="selectedOrganizationID" Mode="Raw" />
    <coolite:Parameter Name="businessDimensionID" Value="selectedBusinessDimensionID" Mode="Raw" />
</BaseParams>
<Reader>
    <coolite:JsonReader ReaderID="RISK_GUID">
        <Fields>
            <coolite:RecordField Name="ID" Mapping="ID" />
            <coolite:RecordField Name="RiskGuid" Mapping="GUID" />
            <coolite:RecordField Name="RiskIdentifier" Mapping="Identifier" />
            <coolite:RecordField Name="RiskConfidential" Mapping="RiskConfidential" />
            <coolite:RecordField Name="RiskName" Mapping="RiskName" />
            <coolite:RecordField Name="RiskOrganizationName" Mapping="OrganizationName" />
            <coolite:RecordField Name="RiskLikelihoodCode" Mapping="GrossLikelihood" />
            <coolite:RecordField Name="GrossLikelihoodColorCode" Mapping="GrossLikelihoodColorCode" Type="String" />
            <coolite:RecordField Name="RiskImpactCode" Mapping="GrossImpact" />
            <coolite:RecordField Name="GrossImpactColorCode" Mapping="GrossImpactColorCode" Type="String" />
            <coolite:RecordField Name="RiskScore" Type="Int" Mapping="GrossRiskScore" />
            <coolite:RecordField Name="GrossScoreColorCode" Mapping="GrossRiskScoreColor" />
            <coolite:RecordField Name="RiskLikelihoodText" Mapping="GrossLikelihoodText" />
            <coolite:RecordField Name="RiskImpactText" Mapping="GrossImpactText" />
            <coolite:RecordField Name="ActionCount" Type="int" Mapping="ActionCount" />
            <coolite:RecordField Name="ControlCount" Type="int" Mapping="ControlCount" />
            <coolite:RecordField Name="LossCount" Type="int" Mapping="EventCount" />
            <coolite:RecordField Name="MoiCount" Type="int" Mapping="MoiCount" />
            <coolite:RecordField Name="RiskMitLikelihoodCode" Mapping="NetLikelihood" />
            <coolite:RecordField Name="RiskMitImpactCode" Mapping="MIT_IMPACT_CODE" />
            <coolite:RecordField Name="RiskMitScore" Type="Int" Mapping="NetRiskScore" />
            <coolite:RecordField Name="NetScoreColorCode" Mapping="NetRiskScoreColor" />
            <coolite:RecordField Name="RiskMitLikelihoodText" Mapping="NetLikelihoodText" />
            <coolite:RecordField Name="RiskMitImpactText" Mapping="NetImpactText" />
            <coolite:RecordField Name="RiskTreatmentName" Mapping="RiskTreatmentName" />
            <coolite:RecordField Name="RiskTreatmentDescription" Mapping="RiskAssessmentDescription" />
            <coolite:RecordField Name="RiskAssessmentSortOrder" Mapping="RiskAssessmentSortOrder" Type="String" />
            <coolite:RecordField Name="RiskAssessmentDescription" Mapping="RiskAssessmentDescription" Type="String" />
            <coolite:RecordField Name="RiskAssessmentColor" Mapping="RiskAssessmentColor" Type="String" />
        </Fields>
    </coolite:JsonReader>
</Reader>
```

7.13: Snapshot van de Ext.Net code om de huidige Risk-tabel te tonen.

Hierboven valt er de risicotabel te zien geschreven in Ext.Net. Dit moet omgeschreven worden naar Angular en zal tijdens sprint 2 gedaan worden.

LITERATUURLIJST

ASP .NET. (2019, 3 augustus). *Parallel For Method in C#*. Geraadpleegd op 8 januari 2020, van https://dotnettutorials.net/lesson/parallel-for-method-csharp/

BIJLAGE 8: ANGULAR ANALYSE

In dit document zal er een Angular analyse volgen. Er zal geanalyseerd worden of Angular wel de juiste framework is om de afstudeeropdracht in te bouwen. Deze analyse is gedaan, om er voor te zorgen dat de framework met de beste, nieuwste of zelfs gemakkelijkste technologie in ieder geval in kaart gebracht is, alvorens er in Angular geprogrammeerd zal gaan worden. De drie grootste javascript frameworks zullen vergeleken worden en uiteindelijk zal de keuze uitgaan naar Angular. Hierna zal er een Angular-componenten analyse plaatsvinden op basis van het nieuwe ontwerp, om te kijken welke bestaande componenten er hergebruikt kunnen worden tijdens mijn vernieuwing van de voorkant van de navigator.

SELECTEREN VAN ICT-GERELATEERDE OPLOSSINGEN

CERRIX heeft een overstap gemaakt van het oude Ext.Net framework naar de nieuwere en snellere Angular framework. Het voordeel hiervan is dat Angular een veel lichtere framework is, waardoor de webpagina's sneller geladen worden. Toen er nog gebruik werd gemaakt van Ext.Net was de applicatie zeer traag en moest er ook 5+ seconden gewacht worden om bijvoorbeeld een risico te openen. Hedendaags d.m.v. de implementatie en de optimalisatie van Angular duurt dit slechts enkele ogenblikken. Alleen is Angular wel de juiste framework om mijjn project in te vervullen?

Aangezien de Business dimension navigator module compleet herschreven wordt, wordt alle bestaande code ook herschreven. Dit biedt mij de gelegenheid om rond te kijken naar andere frameworks, die misschien nét wat meer bieden dan Angular, om dit voor de zekerheid afgewogen te hebben. Hiervoor zal er een analyse volgen tussen de drie grootste javascript frameworks op dit moment: Angular, React en Vue.js. *(Naar 10 Best JavaScript Frameworks to Use in 2019 (07-2019) en Angular vs React vs Vue: Which is the Best Choice for 2019?, (07-2019)). Hieronder zal er in een kort overzicht uitgelegd worden waar deze drie frameworks voor zijn.*

Angular

Angular is uitgekomen in 2010 door Google en is een typescript based js framework. Google is o.a. een van de grote bedrijven die deze framework gebruikt. Angular focust zicht vooral op het gebruik van een single page application. (De webpagina verandert niet. Alle elementen worden dynamisch herladen op dezelfde pagina. Hierdoor verandert de URL ook niet). *(Naar: Angular, z.d.).*

React

React is uitgekomen in 2013 door Facebook. Het wordt het meeste gebruikt voor high-traffic websites, zoals websites die veel advertenties gebruiken als Facebook en dus trager zijn. React is zeer dynamisch en focust zich meer op interactieve en simpele user interfaces voor gebruikers. *(Naar: React – A JavaScript library for building user interfaces, z.d.).*

Vue.js

Vue is uitgekomen in 2014 door een ex-engineer van Google. Websites zoals GitLab en Alibaba gebruiken Vue. Vue noemt zichzelf de progressive JavaScript framework. Dit komt omdat Vue zich ook focust op Single page applications en op user interfaces. De ex-engineer, Evan You heeft elementen van Angular, React en overige frameworks gepakt en samengevoegd tot een lightweight framework die hij handiger vond. *(Naar: Vue.js, z.d.).*

Nu er bekend is wat deze frameworks zijn zal er een vergelijking worden gemaakt op basis van de performance, grootte van de framework, de kennis dat er nodig is en de flexibiliteit van de frameworks, zodat er gekeken kan worden of sommige frameworks uitblinken. (Zie tabel 1 en figuur 2). Dit tabel is gebasseerd op de volgende internetbronnen: Daityari, S. (2019, 11 december), Kumar, A. (2019, 30 september). en TechMagic (2019, 19 juni). Deze bronnen hebben o.a. dezelfde aspecten gebruikt waaruit tabel 1 en figuur 2 uit zijn opgebouwd.

Tabel 1: Vergelijking Angular vs React vs Vue.js

	Performance	**Framework size** (belangrijk voor de scalability en snelheid)	**Learning curve**	**Flexibility**
Angular	Gebruikt de echte DOM (representatie van UI in code), waardoor dit eerst geladen moet worden en trager is dan virtuele DOM.	500 KB 'heavy weight'. Niet goed voor light weight applicaties, omdat Angular veel templates te bieden heeft van het testen tot utilities toe. (CERRIX is heavy weight, omdat er parallel veel data wordt opgehaald en doorgestuurd).	Angular vergt enige tijd om te leren, omdat de taal is opgebouwd uit TypeScript (een set van JS om gemakkelijker te programmeren volgens conventies).	Angular biedt alles aan wat nodig is. Van routing tot templates. Hierdoor heb je geen andere tool nodig om te developen meer.
React	Gebruik van virtuele DOM, dit wordt middels de framework virtueel opgeslagen in het geheugen zonder alle echte zware koppelingen van de echte DOM. Doordat dit virtueel gebeurd is het sneller dan het gebruik van de echte DOM. (lightweight)	100 KB. Het is goed voor light weight applicaties, omdat het niet zoveel templates aanbiedt zoals Angular. Daarom heeft React hulp nodig van losse libraries om extra functies te hebben.	Dit is in JavaScript en dus makkelijk om direct mee aan de slag te gaan.	React biedt niet veel extra tools aal via de React library. Dit betekent dat er third-party libraries gebruikt moeten worden.

Vue.js	Gebruikt eveneens de virtuele DOM, wat sneller is dan de echte DOM.	80 Kb. Dit is een van de smalste frameworks. Hierdoor is het super goed voor light weight frameworks, maar biedt hierdoor geen extra functies aan.	Dit is eveneens in JavaScript.	Vue biedt eveneens niet veel extra tools aan.

	Angular	React	Vue
Type	A Framework	Library to build UI	A library
Why Choose	If you want to use TypeScript	If you want to go for "everything-is-JavaScript" approach	Easy JavaScript and HTML
Founders	Powered by Google	Maintained by Facebook	Created by Former Google Employee
Initial Release	September 2016	March 2013	February 2014
Application Types	If you want to develop Native apps, hybrid apps, and web apps	If you want to develop SPA and mobile apps	Advanced SPA and started supporting Native apps
Ideal for	If you want to focus on large-scale, feature-rich applications	Suitable for modern web development and native-rendered apps for iOS and Android	Ideal for web development and single-page applications
Learning Curve	A steep learning curve	A little bit easier than Angular	A small learning curve
Developer-friendly	If you want to use the structure-based framework	If you want to have flexibility in the development environment	If you want to have separation of concerns
Model	Based on MVC (Model-View-Controller) architecture	Based on Virtual DOM (Document Object Model)	Based on Virtual DOM (Document Object Model)
Written in	TypeScript	JavaScript	JavaScript
Community Support	A large community of developers and supporters	Facebook developers community	Open-source project sponsored through crowd-sourcing
Language Preference	Recommends the use of TypeScript	Recommends the use of JSX – JavaScript XML	HTML templates and JavaScript
Popularity	Widely popular among developers	More than 27,000 stars added over the year	More than 40,000 stars added on GitHub during the year
Companies Using	Used by Google, Forbes, Wix, and weather.com	Used by Facebook, Uber, Netflix, Twitter, Reddit, Paypal, Walmart, and others	Used by Alibaba, Baidu, GitLab, and others

Figuur 2 Vergelijking tussen de frameworks volgens Kumar. A. (Aug, 20, 2018))

Uit bovenstaande tabel en figuur valt op te merken dat Angular verschilt in programmeertaal. Er wordt gebruik gemaakt van TypeScript, wat een soort superset aan conventies is, waardoor het javascript framework wordt afgedwongen om zicht te houden aan bepaalde code conventies, omdat javascript opzichzelf een hele losse taal is, waar niet veel regels aan gebonden zijn in vergelijking tot C# en Java. Dit moet wel door de programmeurs aangeleerd worden.

Daarnaast is Angular wel wat trager qua performance dan de overige twee. Dit komt omdat Angular een zwaardere framework is, die simpelweg meer bestanden heeft dan React en Vue. De bestanden die Angular extra biedt zijn o.a. extra tools voor de routing, pipes en voor de testing, dit is het geval, omdat Angular ook gemaakt is voor large-scaled en heavy-weight web-apps. React en Vue missen dit, waardoor er gebruik gemaakt moet worden van externe libraries om deze functies te ondersteunen, waar nog onderzoek over moet worden gedaan m.b.t. de betrouwbaarheid en onderhoudbaarheid. React en Vue dus kleiner en compacter en bieden het minimale wat een applicatie nodig zou hebben.

De drie frameworks worden nog ondersteund en regulier bijgewerkt en geüpdatet, zodat de security op orde is en worden allemaal door zéér grote bedrijven gebruikt (Google, Twitter, Netflix, Alibaba etc.) en hebben een grote supportbase.

Wat hieruit valt te concluderen is dat React en Vue inderdaad goede frameworks zijn. Ze zijn nieuw, sneller dan Angular, omdat er gebruik wordt gemaakt van de virtuele DOM en zijn gemakkelijk aan te leren omdat ze javascript hanteren. Vue is de meest geliefde library op dit moment, omdat zij het goede van Angular en React combineert. Echter zijn Vue en React niet geschikt voor heavy-weight en volledige web-applicaties, omdat zij niet de juiste tools bieden om dit te ondersteunen. Angular biedt dit wel en daarom is de keuze toch naar Angular gegaan.

De keuze om te werken in Angular vind ik persoonlijk ook handiger, omdat het team van CERRIX ook al programmeert in Angular en zij mij hierom de beste ondersteuning kunnen bieden, mocht ik tegen problemen aanlopen. Ook hebben zij diverse (generieke) componenten ontworpen waarmee er bijvoorbeeld in HTML, CSS en TypeScript tabellen worden weergeven op het scherm en die direct in contact staan met de database. Dit kan ik dan ook mooi combineren met mijn navigator, zodat ik dit alleen nog hoef te implementeren. Er zal binnen het project dus inderdaad gewerkt worden met de framework Angular. Dat heeft deze analyse versterkt.

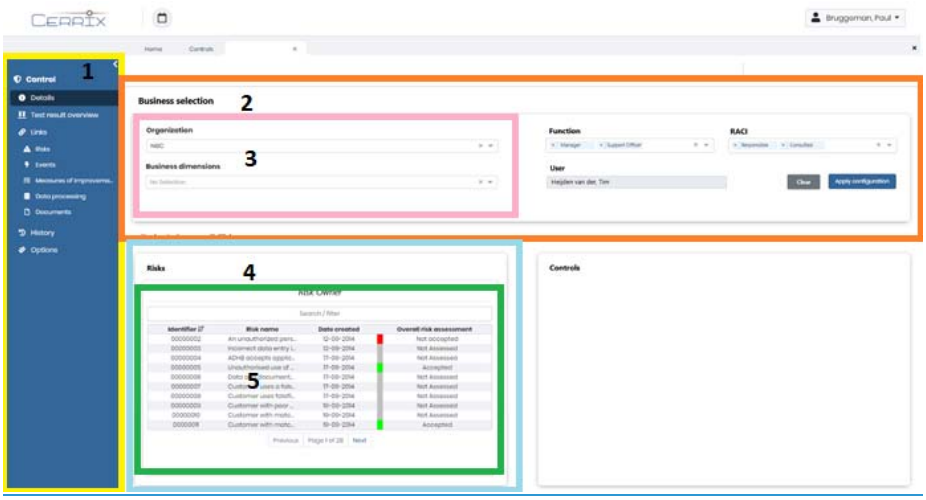
ANALYSE ANGULAR-COMPONENTEN IN HET NIEUWE ONTWERP

De developers van CERRIX hebben al het een en ander gemaakt in Angular. Zo hebben zij al meer dan de helft van de applicatie omgebouwd van de oude Ext.Net naar Angular schermen. Hiervoor moesten zij bepaalde componenten hergebruiken, bijvoorbeeld voor het weergeven van tabeldata. Ook is er al code geschreven voor het maken van een organization en business dimension filter, omdat dit terug komt op de control details pagina.

Het nieuwe ontwerp is niet zomaar verzonnen. Alle elementen (filter/ tabel/ knoppen) die terugkomen in het nieuwe ontwerp zijn door Ruben geselecteerd, omdat ze voldoen aan de huisstijl van CERRIX en dus ook (generiek) terugkomen in de applicatie. Deze generieke componenten kan ik hergebruiken als ik de pagina ga omschrijven naar Angular. Dit zal ervoor zorgen dat ik niet opeens code op zal gaan leveren voor het weergeven van een tabel, terwijl die al bestaat en alleen nog geïmplementeerd moest worden in mijn Navigator. Hierom zal er op een rijtje gezet worden welke Angular-componenten ik kan hergebruiken om het nieuwe ontwerp te kunnen bouwen.

INVENTARISERING TE GEBRUIKEN ANGULAR-COMPONENTEN

Allereerst zal er gekeken worden naar het nieuwe ontwerp. (Zie figuur 3). Dit ontwerp is gesplitst in 5 grote onderdelen. Dit zijn onderdelen die verder in CERRIX voor komen in de vorm van generieke componenten. Dit heb ik gevonden door de applicatie van CERRIX te inspecteren via de webbrowser. Hierdoor kwam ik op de volgende generieke componenten uit die CERRIX gebruikt in de rest van de applicatie en die ik ook kan toepassen om het nieuwe ontwerp waar te maken.



Figuur 3 Analyse van de Angular-componenten die voorkomen in het ontwerp

De nummers die op figuur 3 worden vermeld worden in figuur 4 uitgelegd en beschreven (zie tabel 4)

Tabel 4: Legenda van de nummers die voorkomen op figuur 3.

# Angular component	

Formatted: Justified

Formatted: Font: Bold

1	Dit is een menusysteem, ontworpen door de developers van CERRIX zelf. (“<menusystem> </menusystem>”). Hier zijn sub menu’s, zoals de documents, history en details van de business dimension navigator.
2	Dit is een grafische ‘card’ weergave, waarin items mooi met een schaduw worden weergeven. Dit is gemaakt middels een ‘details-page-card’ component van CERRIX.
3	Dit zijn de CERRIX-angular filters. Deze zijn al ontworpen middels de CERRIX ‘<cerrix-tree> </cerrix-tree>’component, omdat de organisaties en business dimensions gelaagdheid hebben en in een tree zitten. Door gebruik te maken van dit component is de styling gelijk goed met de rest van de applicatie.
4	Dit is weer een CERRIX ‘detail-page-card’ klasse, gemaakt voor een HTML (Een blok in HTML). Hierdoor zijn er mooie onderscheidingen gemaakt tussen de pagina en de blokken met data.
5	Dit is een CERRIX ‘<generic-list-component> </generic-list-component>’. Door het implementeren van dit component kunnen er lijsten met informatie (zoals de velden van het risico) mooi weergeven worden zoals de manier die door de rest van de applicatie heen gebruikt wordt.

Formatted: Justified

Formatted: Justified

Formatted: Justified

Formatted: Justified

Formatted: Justified

Tabel 4 beschrijft de vijf grote, generieke Angular-componenten die ik kan hergebruiken voor het bouwen van het nieuwe ontwerp in Angular. Dit tabel heb ik geverifieerd bij Martin en hij zei dat dit inderdaad de juiste Angular-componenten zijn die gebruikt worden. Dat betekent dat ik tijdens de uitvoering van de sprints in de scriptie deze componenten zal toepassen. Ze zijn nu in ieder geval geïdentificeerd.

LITERATUURLIJST

10 Best JavaScript Frameworks to Use in 2019. Geraadpleegd op 31 oktober 2019, van https://hackr.io/blog/10-best-javascript-frameworks-2019 H., S. (2019, 2 juli).

Angular vs React vs Vue: Which is the Best Choice for 2019? Geraadpleegd op 31 oktober 2019, van https://hackernoon.com/Angular-vs-react-vs-vue-which-is-the-best-choice-for-2019-16ce0deb3847

Angular. (z.d.). Geraadpleegd op 31 oktober 2019, van https://Angular.io/

React – A JavaScript library for building user interfaces. (z.d.). Geraadpleegd op 29 oktober 2019, van https://reactjs.org/

Vue.js. (z.d.). Geraadpleegd op 29 oktober 2019, van https://vuejs.org/

Daityari, S. (2019, 11 december). *Angular vs React vs Vue: Which Framework to Choose in 2020*. Geraadpleegd op 9 januari 2020, van https://www.codeinwp.com/blog/Angular-vs-vue-vs-react/

Kumar, A. (2019, 30 september). *React vs. Angular vs. Vue.js: A Complete Comparison Guide*. Geraadpleegd op 9 januari 2020, van https://dzone.com/articles/react-vs-Angular-vs-vuejs-a-complete-comparison-gu

TechMagic. (2019, 19 juni). *React vs Angular vs Vue.js — What to choose in 2019? (updated)*. Geraadpleegd op 9 januari 2020, van https://medium.com/@TechMagic/reactjs-vs-Angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d

BIJLAGE 9: TESTRAPPORTEN

In dit document zullen de testcases uitgewerkt worden. Er zal uitgelegd worden welke testen er zijn uitgevoerd en waarom. Deze testen zullen gekoppeld worden aan de user stories voor een duidelijke traceerbaarheid. Weergeven zal worden wat de invoer, verwachtte uitvoer, werkelijke uitvoer, en conclusie is uit de testen.

TESTEN VOOR SPRINT 2: IMPLEMENTATIE V.D. SERVER-SIDE VERSNELLING N.A.V. HET ONTWERP

In deze paragraaf zullen de testen worden besproken die er zijn doorlopen tijdens de uitvoering van sprint 2. Er zal besproken worden welke functionele en niet-functionele testen er zijn uitgevoerd.

FUNCTIONELE TEST

Er is de keuze gemaakt voor het uitvoeren van functionele testen, om te controleren dat de juiste data wordt opgehaald uit de database. Dit is uitgevoerd voor user story U06 t/m U11. (zie tabel 1)

Tabel 1: Lijst met user stories die een functionele test hebben gekregen.

U06	H	As < an overviewer > I want < to see my Controls according to the **server-side filter** - when i apply my configuration> so <that I can get an overview>	12
U07	H	As < an overviewer > I want < to see my MoIs according to the **server-side filter** - when i apply my configuration> so <that I can get an overview>	12
U08	H	As < an overviewer > I want < to see my Data Processings according to the **server-side filter** - when i apply my configuration> so <that I can get an overview>	12
U09	H	As < an overviewer > I want < to see my KRIs according to the **server-side filter** - when i apply my configuration> so <that I can get an overview>	12
U10	H	As < an overviewer > I want < to see my Risks according to the **server-side filter** - when I apply my configuration> so <that I can get an overview>	12
U11	H	As < an overviewer > I want < to see my Events according to the **server-side filter** - when i apply my configuration> so <that I can get an overview>	12

Het doel van de uitvoering van deze functionele testen is om ervoor te zorgen dat de juiste risks, controls, mois, data processingsm events en kri's opgevraagd worden uit de database. Omdat elke module beschouwd zal worden als een kleine aparte unit die los van elkaar getest moet worden, is er gekozen voor een unittest.

Er zullen per module twee unittests uitgevoerd worden om te bepalen of er wel de juiste data wordt opgehaald a.d.h.v. de rollen en rechten van de gebruiker en de selectie van organization en business dimension waarop gefilterd wordt. De rollen en rechten die er meegegeven worden aan de mockdata verschilt natuurlijk ook per module daarom verschilt het ook per 2 testcases Zie tabel 2 voor de volledige test cases en uitvoer van de unit tests.

# Test ID	# User story ID	Omschrijving	Verwachte uitvoer	Verwachte resultaat	Resultaat
T01	U06	Selecteer de Identifier van een control met org = 39, bd = 126, **met** het recht om de control te kunnen zien (CMRV)	Identifier = 2849	Identifier = 2849	
T02	U06	Selecteer de Identifier van een control met org = 39, bd = 126, **zonder** het recht om de control te kunnen zien	Test failed	Test failed	
T03	U07	Selecteer de Identifier van een moi met org = 40, bd = 129, **met** het recht om de moi te kunnen zien (MRV)	Identifier = 112	Identifier = 112	
T04	U07	Selecteer de Identifier van een moi met org = 39, bd = 126, **zonder** het recht om de moi te kunnen zien	Test failed	Test failed	
T05	U08	Selecteer de Identifier van een data processings met org = 42, bd = 99, **met** het recht om de moi te kunnen zien (DPRV)	Identifier = 25	Identifier = 25	
T06	U08	Selecteer de Identifier van een data processings met org = 42, bd = 99, **zonder** het recht om de data processings te kunnen zien	Test failed	Test failed	
T07	U09	Selecteer de Identifier van een kri met org = 26, bd = 122, **met** het recht om de kri te kunnen zien (KRV)	Identifier = 07	Identifier = 07	

T08	U09	Selecteer de Identifier van een kri met org = 26, bd = 122, **zonder** het recht om de kri te kunnen zien	Test failed	Test failed	
T09	U10	Selecteer de Identifier van een risk met org = 9, bd = 1, **met** het recht om de risk te kunnen zien (RRV)	Identifier = 23	Identifier =`23	
T10	U10	Selecteer de Identifier van een risk met org = 9, bd = 1, **zonder** het recht om de risk te kunnen zien	Test failed	Test failed	
T11	U11	Selecteer de Identifier van een event met org = 3, bd = 7, **met** het recht om de event te kunnen zien (LERRV)	Identifier = 99	Identifier = 99	
T12	U11	Selecteer de Identifier van een event met org = 3, bd = 7, **zonder** het recht om de event te kunnen zien	Test failed	Test failed	

Hieronder zal T01 uitgewerkt worden (zie figuur 3). De bedoeling van deze test is dat er een control bestaat met organization ID = 39 en business dimension ID = 126. Deze control heeft als Identifier 2849. Om te testen of er aan de juiste security eisen voldoen is er een gebruiker aangemaakt met het Control restricted viewer recht, waardoor hij alleen controls binnen zijn eigen organisatie mag zien (hier zorgt de ControlBusiness voor). Organisatie 39 valt onder zijn organisatie (dat is immers gedefinieerd in de code). Hierdoor zou de

mockBusiness de controls ophalen die de gebruiker mag zien d.m.v. zijn rechten. De test slaagt en de uitvoer is correct.

```
[TestMethod]
public void TestControls()
{
    /*I'm fetching the control with organization: Cerrix (that is ID = 39)
      With the business dimension:  Internal communication (that is ID = 126)
      I'm expecting to see the control 'Rohit test control' with the Identifier: 2849.
      The user has the 'CMRV' (Control restricted viewer) with the same organizationID as the control.OrganizationID
      So it should work */

    var mockControl = new Mock<ORM_Risk_Control>{
        Identifier = 2849,
        ORM_Organization = 39,
        ORM_Business_Dimension = 126
    }.SaveChanges();

    var mockUser = new Mock<Auth_User>{
        Name = "TestUser",
        ORM_Organization = 39
    }.AddRight("CMRV").SaveChanges();


    var mockBusiness = new Mock<ControlBusiness>().Create();

    var expected = 2849;
    var actual = mockBusiness().GetAll(mockUser)
    .Where(x => x.ORM_Organization = 39
        && X.ORM_Business_Dimension = 126)
        .Select(a => a.Identifier);

    Assert.AreEqual(expected, actual);
}
```

Figuur 3 Unittest van T01 voor het laden van een control d.m.v. een org en bd filter. De uitvoer van deze test is correct.

Uiteraard zal ik ook een test laten zien die faalt en die ook hoort te falen. Zie figuur 4 voor de uitwerking van T02. Dit is bijna dezelfde code als figuur 3, alleen heeft de gebruiker het recht niet om de control te kunnen zien. Hij heeft helemaal geen rechten. Hierdoor kan de mockBusiness().GetAll(mockUser) niet bij de gewenste control en faalt de test. Dit laat zien dat de implementatie voor de juiste data a.d.h.v. de rollen en rechten v.d. gebruiker i.c.m. de server-side filtering goed gaat.

```
[TestMethod]
public void TestControls()
{
    /*I'm fetching the control with organization: Cerrix (that is ID = 39)
      With the business dimension:  Internal communication (that is ID = 126)
      I'm expecting to see the control 'Rohit test control' with the Identifier: 2849.
      The user has the 'CMRV' (Control restricted viewer) with the same organizationID as the control.OrganizationID
      So it should work */

    var mockControl = new Mock<ORM_Risk_Control>{
        Identifier = 2849,
        ORM_Organization = 39,
        ORM_Business_Dimension = 126
    }.SaveChanges();

    var mockUser = new Mock<Auth_User>{
        Name = "TestUser",
        ORM_Organization = 70
    }.SaveChanges();


    var mockBusiness = new Mock<ControlBusiness>().Create();

    var expected = 2849;
    var actual = mockBusiness().GetAll(mockUser)
    .Where(x => x.ORM_Organization = 39
        && X.ORM_Business_Dimension = 126)
        .Select(a => a.Identifier);

    Assert.AreEqual(expected, actual);
}
```

Figuur 7 Unittest van T02: Proberen een control te kunnen openen, waarbij de gebruiker geen toegang tot heeft omdat hij niet genoeg rechten heeft. De test faalt en dat is de bedoeling

NIET-FUNCTIONELE TEST

Naast de functionele unittesten zullen er ook performance tests worden uitgevoerd, omdat de snelheid van het laden en server-side filteren van de data een belangrijk gedeelte is van de afstudeeropdracht.

PERFORMANCE TEST

De performance test werd uitgevoerd in de vorm van een load test, omdat er gekeken zal worden of ik als tester kon uitvogelen of er een bottleneck in het laden voorkomt. Mocht dit het geval zijn, dan zal dit komen door een incorrecte implementatie van de server-side versnelling, en kan ik het dus tijdig oplossen. Zie tabel 5 voor de testcase.

Dezelfde test zal meerdere keren uitgevoerd worden om te bepalen of de performance verbetert, naar mate er vaker gefilterd wordt. Het kan misschien zo zijn dat het filteren dan sneller verloopt, omdat er al bepaalde business en DAL instanties gecreëerd zijn en niet nog een keer aangemaakt hoeven te worden. Hierom zal de test vier keer herhaald worden, om te testen wat het verschil tussen de pogingen zijn.

Tabel 5: Testcase server-side performance voor het laden en filteren door de risico's in de nieuwe navigator.

T13: Test case: Open de Navigator en filter op organization: CERRIX en business dimension: Wareman Warenhuis. Verifieer dat voor het laden en filteren van de gehele pagina **minder dan 20 seconden duurt** (omdat dit in de oude situatie wel het geval is). (Er worden 2049 risks, 457 events, 16 mois, 120 kri's, 3 data processings en 684 controls doorlopen)
Uitvoering: Vanaf het moment dat de Navigator geopend wordt, houdt Google Chrome dit middels de inspectietool bij en heb ik daarnaast ook een stopwatch. Ik kan zien welke functies er aangeroepen worden en hoelang de uitvoering hiervan duurt. Er zal gelet worden op de inspectietool, om te vergelijken of het herladen van het filter sneller zal verlopen.
Bevindingen attempt 1: Nadat er een selectie in het filter was gemaakt en er op Apply Configuration was geklikt duurde de eerste attempt in totaal: **15 seconden**. Dit is inderdaad een verbetering met de situatie hiervoor.
Bevindingen attempt 2: De tweede keer duurde het **15 seconden.** Dit kan liggen aan het feit dat er in de back-end al bepaalde objecten zijn initialized, waardoor het wat sneller laadt.
Bevindingen attempt 3: De derde keer duurde het laden wederom **15 seconden.**
Bevindingen attempt 4: De vierde keer duurde het laden wederom 15 seconden en het blijkt dat dit het minimaalste is wat ik kan bereiken qua performance.
Conclusie: De performance is door de server-side versnelling verbeterd met 5 seconden en wordt er in ieder geval niet langzamer op. De inspectietool laat geen bottleneck zien, omdat alle calls tegelijkertijd starten en ook niet vast zitten en/of op elkaar wachten.

UI-TEST

Tot slot zullen er UI tests plaatsvinden, om te checken of het ontwerp wel correct geïmplementeerd is en er geen tabellen, labels, knoppen of kleurtjes missen. Dit is een relatief makkelijke test, omdat het nieuwe ontwerp niet zo groot/ complex is. Het is een kleine pagina met relatief weinig onderdelen. Hierom zal de UI-test handmatig uitgevoerd worden en voor de test cases verwijs ik u door naar Tabel 6.

Tabel 6: UI test cases voor de omschrijving van de view van Ext.Net naar Angular

# Test ID	Omschrijving	Resultaat
T14	Controleer dat de risico tabel zichtbaar is incl. de juiste kolommen	
T15	Controleer dat de moi tabel zichtbaar is incl. de juiste kolommen	
T16	Controleer dat de kri tabel zichtbaar is incl. de juiste kolommen	
T17	Controleer dat de data processings zichtbaar is incl. de juiste kolommen	
T18	Controleer dat de event tabel zichtbaar is incl. de juiste kolommen	
T19	Controleer dat de control tabel zichtbaar is incl. de juiste kolommen	Verbetering: Kolommen: **Control Owner** en **In place** waren aanwezig, maar niet geïmplementeerd (lege uitvoer). Dit is verholpen.
T20	Controleer dat het filter de clear en apply buttons hebben	

CONCLUSIE TESTS VOOR SPRINT 2

Tijdens deze sprint zijn er verschillende testen uitgevoerd. Van functionele unit-testen tot niet-functionele UI en performance testen. Alle unit tests zijn uitstekend verlopen. Er is geen enkele gefaald. Dit had ik ook verwacht, omdat ik niet zelf de code heb geschreven voor het ophalen van de juiste data a.d.h.v. de rollen en rechten van een gebruiker, maar de echte developers dit zelf al hebben gedaan. Deze functie heb ik alleen geïmplementeerd en de server-side filtering heb ik hier ook op toegepast.

Ook er performance tests uitgevoerd. Er was getimed hoelang het duurde voordat alle zes modules gefilterde uitvoer op het scherm lieten zien. Dit duurde 15 seconden om door een flink aantal rows heen te gaan. (2049 risks, 457 events, 16 mois, 120 kri's, 3 data processings en 684 controls). 15 seconden is natuurlijk nog best wel lang, maar is in ieder geval een verbetering met de oude situatie, waar het filteren en laden van alle data 20 seconden duurde.

Tot slot zijn er UI testen gedaan. Deze waren uitgevoerd om te controleren of wel alle onderdelen die in het ontwerp stonden waren opgenomen. Uit de UI tests bleken er 2 kolommen te zijn, die nog niet geïmplementeerd waren. Ik ben blij dat er een UI test is gedaan, anders zou dit pas op een later stadia ontdekt worden. (Misschien pas bij de volgende sprint?).

Al met al ben ik zeer tevreden met zowel de resultaten als de gekozen testen. Ik vind dat de tests genoeg coverage bieden over het uitgevoerde werk omdat er verschillende soorten testen worden uitgevoerd die allemaal wat anders testen en met een ander resultaat aankomen.

TESTEN VOOR SPRINT 3: IMPLEMENTATIE V.D. CLIENT-SIDE VERSNELLING N.A.V. HET ONTWERP

In deze paragraaf zullen de testen worden besproken die er zijn doorlopen tijdens de uitvoering van sprint 3 en welke functionele en niet-functionele testen er waren uitgevoerd.

FUNCTIONELE TEST

Omdat er tijdens sprint 2 unit tests zijn gemaakt en de code voor het ophalen van de risico's aan de niet is aangepast, kunnen de zelfde tests (T01 t/mT12) wederom uitgevoerd worden. De uitvoer hiervan zal beschreven worden in table 7.

Tabel 7: De uitvoer van de bestaande unit tests T01 t/m T12

# Test ID	# User story ID	Omschrijving	Verwachte uitvoer	Verwachte resultaat	Resultaat
T01	U06	Selecteer de Identifier van een control met org = 39, bd = 126, **met** het recht om de control te kunnen zien (CMRV)	Identifier = 2849	Identifier = 2849	
T02	U06	Selecteer de Identifier van een control met org = 39, bd = 126, **zonder** het recht om de control te kunnen zien	Test failed	Test failed	
T03	U07	Selecteer de Identifier van een moi met org = 40, bd = 129, **met** het recht om de moi te kunnen zien (MRV)	Identifier = 112	Identifier = 112	
T04	U07	Selecteer de Identifier van een moi met org = 39, bd = 126, **zonder** het recht om de moi te kunnen zien	Test failed	Test failed	
T05	U08	Selecteer de Identifier van een data processings met org = 42, bd = 99, **met** het recht om de moi te kunnen zien (DPRV)	Identifier = 25	Identifier = 25	
T06	U08	Selecteer de Identifier van een data processings met org = 42, bd = 99, **zonder** het recht om de data processings te kunnen zien	Test failed	Test failed	
T07	U09	Selecteer de Identifier van een kri met org = 26, bd = 122, **met** het recht om de kri te kunnen zien (KRV)	Identifier = 07	Identifier = 07	

T08	U09	Selecteer de Identifier van een kri met org = 26, bd = 122, **zonder** het recht om de kri te kunnen zien	Test failed	Test failed	
T09	U10	Selecteer de Identifier van een risk met org = 9, bd = 1, **met** het recht om de risk te kunnen zien (RRV)	Identifier = 23	Identifier =`23	
T10	U10	Selecteer de Identifier van een risk met org = 9, bd = 1, **zonder** het recht om de risk te kunnen zien	Test failed	Test failed	
T11	U11	Selecteer de Identifier van een event met org = 3, bd = 7, **met** het recht om de event te kunnen zien (LERRV)	Identifier = 99	Identifier = 99	
T12	U11	Selecteer de Identifier van een event met org = 3, bd = 7, **zonder** het recht om de event te kunnen zien	Test failed	Test failed	

Het resultaat van deze tests zijn dat ze allemaal wederom geslaagd zijn. Ik had niet anders verwacht omdat ik niet aan de code die staat in de back-end heb gezeten, waardoor deze tests geheid zouden slagen. Dit is gebeurd en hier ben Ik tevreden mee.

NIET-FUNCTIONELE TEST

Tijdens deze sprint is de implementatie voor het filteren van de data aangepast. Nu filtert de navigator de data client-side en er zal moeten getest worden hoe veel sneller, of trager deze implementatie is t.o.v. de server-side filtering van zowel de nieuwe als de oude situatie.

PERFORMANCE TEST

De performance test die er uitgevoerd zal worden zal dezelfde test zijn als T13, alleen moet het laden en filteren van de data nu <15 seconden duren, omdat dat de uitvoer was van T13, zodat er echt kan blijken dat de client-side versnelling het problem heeft opgelost.

T14: Test case: Open de Navigator en filter op organization: CERRIX en business dimension: Wareman Warenhuis. Verifieer dat voor het laden en filteren van de gehele pagina **minder dan 15 seconden duurt** (omdat dit in T13 het geval is). (Er worden 2049 risks, 457 events, 16 mois, 120 kri's, 3 data processings en 684 controls doorlopen)
Uitvoering: Vanaf het moment dat de Navigator geopend wordt, houdt Google Chrome dit middels de inspectietool bij en heb ik daarnaast ook een stopwatch. Ik kan zien welke functies er aangeroepen worden en hoelang de uitvoering hiervan duurt. Er zal gelet worden op de inspectietool, om te vergelijken of het herladen van het filter sneller zal verlopen.
Bevindingen attempt 1: Nadat er een selectie in het filter was gemaakt en er op Apply Configuration was geklikt duurde de eerste attempt in totaal: **3 seconden.** Dit is prachtig om te zien, omdat de data nu even snel laad als de rest van de applicatie.
Bevindingen attempt 2: De tweede keer duurde het **1.95 seconden.** Dit is het geval, omat de data van alle modules al aan de client-side zijn ingeladen. Deze data wordt alleen opnieuw gefilterd, wat dus sneller is dan attempt 1.
Bevindingen attempt 3: De derde keer duurde het laden wederom **1.95.**
Bevindingen attempt 4: De vierde keer duurde het laden en filteren wederom 1.95 en het lijkt dat dit het minimaalste aantal seconden is wat ik kan bereiken.
Conclusie: De performance is door de client-side nog veel beter verbeterd dan door de implementatie van de server-side filtering en met een zeer grote marge. Zo duurde het hiervoor 15 seconden, en nu maar 1.95 seconden minimaal, nadat alle data na het eerste attempt al in de client was geladen. Dit is een prachtige verbetering en laat zien dat de performance echt is verbeterd.

UI-TEST

Zoals eerder vermeld had Paul tijdens de review van sprint 2 nog extra feedback, betreft het ontwerp. Hij wou graag dat de knoppen 'Apply configuration' veranderde naar 'apply filter' en had nog andere kleine aanpassingen. Een UI-test biedt de perfecte oplossing om te testen of deze wensen wel zijn geïmplementeerd en niet over het hoofd zijn gezien. Hiervoor zijn de volgende test-cases opgesteld en uitgevoerd (Zie tabel 8)

Tabel 8: Uitgevoerde UI-tests voor sprint 3

# Test ID	Omschrijving	Resultaat
T21	De apply configuration knop heet nu apply filter	
T22	De lettertype van de knoppen moet wat kleiner	

CONCLUSIE UITGEVOERDE TESTEN SPRINT 3

Ik ben zeer tevreden met het resultaat uit de tests van sprint 3. De unittesten slaagden allemaal, en dit verwachtte ik vanaf het begin al, omdat ik deze code niet had aangepast. Daarnaast waren alle verbeteringen voor de UI doorgevoerd en tot slot is de performance van de pagina zeer verbeterd. Het laden en filteren duurt client-side maar 2 seconden maximaal! Hier ben ik zeker tevreden mee.

BIJLAGE 10: EVALUATIE BEDRIJFSMENTOR

Faculteit IT & Design

Delft

Evaluatieformulier afstuderen

In te vullen door opdrachtgever c.q. bedrijfsmentor(en)

Student: Rohit Paltoe

Periode: 2 sep 2019 - 10 jan 2020

Bedrijf c.q. instelling: Cerrix B.V.

Bedrijfsmentor: Martin van Leeuwen

Plaats: Den Haag

Datum: 06-01-2020

1. **Heeft de student zichzelf snel en goed ingewerkt in het bedrijf en de uit te voeren afstudeeropdracht?**

   Goed. De student is direct begonnen met het uitwerken van de opdracht.

2. **Hoe beoordeelt u de communicatieve vaardigheden van de student (in de samenwerking met collega's, in contacten met de opdrachtgever, bij mondelinge presentaties, schriftelijke rapportages)?**

   Gemiddeld. De student stelt duidelijke vragen bij interviews en gesprekken, maar mist nog structuur bij zijn presentaties waardoor niet altijd de gewenste informatie naar voren komt.

3. **Hoe heeft de student tijdens het uitvoeren van de opdracht gefunctioneerd?**

- **Qua verantwoordelijkheid** goed
- **Qua zelfstandigheid** voldoende
- **Qua planmatig werken** voldoende
- **Qua creativiteit** goed
- **Qua productiviteit** goed
- **Qua samenwerken met collega's** goed
- **Qua draagvlakontwikkeling** voldoende
- **Qua inspelen op bedrijfscultuur** goed

- **Qua rekening houden met de specifieke context van het bedrijf** voldoende
- **Qua het op gang brengen van de nodige veranderingen** matig

4. **Hoe beoordeelt u de kennis en kunde van de student in verhouding tot wat u verwacht van een bijna afgestudeerde?**

   De student had bij aanvang van het afstuderen voldoende kennis om gelijk aan de slag te kunnen met de opdracht. De kennis die de student nog miste heeft hij opgedaan tijdens het afstuderen.

5. **Hoe beoordeelt u de kwaliteit van de opgeleverde (tussen)producten?**

   Goed, uit de tussenproducten kon ik goed opmaken waar de student mee bezig was en verder mee aan de slag zou gaan.

6. **Bent u tevreden over het opgeleverde (eind)product?**

   - **In hoeverre heeft u gekregen wat is afgesproken?**

     Qua documentatie en source code is alles opgeleverd wat vooraf was afgesproken.

   - **In hoeverre voldoet het (eind)product aan uw verwachtingen?**

     De source code die de student heeft opgeleverd voldoet aan de verwachtingen en vormt een goede basis voor ons om op door te ontwikkelen.

- **Wat is de bruikbaarheid en onderhoudbaarheid hiervan?**

  De code van de student is beoordeeld door twee medewerkers van het bedrijf en de bevindingen daarvan zijn doorgevoerd door de student. De kwaliteit van de code voldoet hierdoor aan onze eisen, zowel voor bruikbaarheid en onderhoudbaarheid.

- **Wat gebeurt er met het opgeleverde (eind)product?**

  Het (afstudeer)product wordt meegenomen in de volgende release van ons product (Cerrix).

- **Kunt u direct met het opgeleverde product aan de slag?**

  Ja wij kunnen er direct mee verder.

7. Zijn er nog aspecten voor u van belang die nog niet aan de orde zijn geweest?

Nee

~~Refereren naar bijlage~~

~~1opzetten van SCRUMInsert bijlag~~