Operational Dashboard

Realisatie

Stage Eurofins Digital Testing - Resillion

Jonas Claes

Bachelor in de Toegepaste Informatica
Keuzerichting Digital Innovation

Thomas More
Academiejaar 2022-2023
Campus Geel, Kleinhoefstraat 4, BE-2440 Geel

eurofins | Digital Testing ▸▸ resillion

CREATED BY STUDENTS FROM THOMAS MORE

LID VAN ASSOCIATIE KU LEUVEN

Inhoudsopgave

Inhoudsopgave 1
Voorwoord 3
Abstract 5
Begrippenlijst 7
Figurenlijst 10
1 Inleiding 11
2 Eurofins Digital Testing 13
3 Toelichting gebruikte technologieën 15
3.1 Elastic Stack 15
3.2 Kibana 17
3.3 Docker 18
3.4 TypeScript 18
3.5 Tsoa 19
3.6 Node.js 19
3.7 Azure DevOps JavaScript API library 20
3.8 Python 20
3.9 Filebeat 21
3.10 Bash 21
3.11 Atlassian Jira 22
3.12 SonarCloud / SonarQube 23
3.13 TOML 25
3.14 Microsoft Teams 26
4 Realisatie 27
4.1 Oriëntatie 28
4.2 Fase 1 – Dataopslag en verwerking 29
4.3 Fase 2 – Connectors 32
4.3.1 Atlassian Jira integratie 32
4.3.2 Sonar integratie 34
4.3.3 Voorzien van eigen Sonar library 34
4.3.4 Integratie van gegevens in Elasticsearch 36
4.3.5 Azure DevOps integratie 38
4.3.6 Python integratie 40
4.3.7 Java integratie 41

4.4 Fase 3 – Alerting 42
4.4.1 Microsoft Teams integratie 42
4.4.2 Kibana 43
4.5 Fase 4 – Deployment 44
4.5.1 Server 44
4.5.2 Infrastructure-as-code 45
4.6 Fase 5 – Dashboards 46
4.6.1 Jira 46
4.6.2 Sonar 48
4.6.3 Azure DevOps 50
5 Extra - Philips Hue integratie 51
6 Conclusie 52
7 Aanbevelingen 53
7.1 Let’s Encrypt / ACME SSL certificaten 53
7.2 Elastic Cloud 54
Nawoord 55

Voorwoord

Sinds mijn kindertijd ben ik altijd gefascineerd geweest door computers en programmeren. Mijn passie voor technologie begon op achtjarige leeftijd, toen ik voor het eerst een computer in handen kreeg. Sindsdien heb ik mijn vaardigheden in programmeren ontwikkeld, wat heeft geleid tot mijn huidige carrière. Deze scriptie over het "Operational Dashboard" project voor Eurofins Digital Testing weerspiegelt mijn voortdurende passie voor programmeren en het verkennen van nieuwe technologieën.

Mijn interesse in data-analyse en het genereren van waardevolle inzichten heeft mij ertoe gebracht om dit project te kiezen als het onderwerp van mijn bachelor scriptie. De uitdaging om een dashboard te ontwikkelen dat operationele gegevens aggregeert en visualiseert, sprak me aan vanwege de mogelijkheid om mijn vaardigheden in programmeren te combineren met mijn interesse in data-analyse.

Gedurende het hele proces van het schrijven van deze scriptie en het werken aan het Operational Dashboard-project, ben ik geconfronteerd met diverse uitdagingen en leerervaringen. Ik ben ervan overtuigd dat de kennis en ervaring die ik heb opgedaan tijdens dit project mij in staat zullen stellen om verder te groeien in mijn carrière en nog meer te bereiken op het gebied van technologie en data-analyse.

Ik wil graag van deze gelegenheid gebruik maken om mijn oprechte dank uit te spreken aan iedereen die mij heeft bijgestaan en ondersteund tijdens het schrijven van deze scriptie. Allereerst wil ik mijn begeleiders, Caroline Vanderheyden, Tiziana Treccase en Martijn Degrève bedanken voor hun waardevolle begeleiding, geduld en aanmoediging gedurende het hele proces.

Ook wil ik mijn medestudenten en collega's bij Eurofins Digital Testing bedanken voor hun inzichten, samenwerking en vriendschap, die onmisbaar waren voor het succes van dit project.

Abstract

Het onderwerp van deze bachelor scriptie is de ontwikkeling en implementatie van een "Operational Dashboard" voor Eurofins Digital Testing. Dit dashboard is ontworpen om operationele gegevens uit verschillende platformen zoals Jira, Azure en SonarQube/SonarCloud te aggregeren en visualiseren, en biedt bruikbare inzichten in de prestaties en efficiëntie van het bedrijf. De focus ligt op het integreren van deze gegevensbronnen en het creëren van een gebruiksvriendelijke en aanpasbare interface voor het analyseren van bedrijfsinformatie.

Het project is uitgevoerd met behulp van een ontwikkelingsgerichte aanpak, waarbij verschillende methoden en technologieën zijn toegepast, zoals Kibana voor visualisatie en dashboarding, en integraties met Microsoft Teams voor het genereren van alerts. De ontwikkeling van het dashboard omvatte het opzetten van dataconnecties, het configureren van visualisaties en het ontwikkelen van een flexibele en aanpasbare interface voor het weergeven van de gegevens.

De resultaten van dit project omvatten een succesvol geïmplementeerd Operational Dashboard dat inzicht biedt in verschillende aspecten van de bedrijfsvoering, zoals het aantal open tickets, tickets die al enige tijd open staan, overzichten van pipelines en agent statussen, en resultaten van Sonar-analyses. Het dashboard is geoptimaliseerd voor gebruiksgemak en aanpasbaarheid, en stelt gebruikers in staat om inzichten te verkrijgen die hen kunnen helpen bij het nemen van beslissingen.

De conclusie van deze scriptie is dat het Operational Dashboard een waardevol instrument is voor Eurofins Digital Testing, waarmee het bedrijf zijn operationele gegevens efficiënt kan analyseren en monitoren. Het biedt een geïntegreerde en gebruiksvriendelijke oplossing voor het bijhouden van bedrijfsprestaties en het identificeren van mogelijke verbeterpunten.

Voor toekomstig onderzoek en aanbevelingen wordt voorgesteld om het dashboard verder te ontwikkelen met extra gegevensbronnen en visualisaties, de mogelijkheden voor aanpasbaarheid en gebruikersinteractie te vergroten, en de prestaties en efficiëntie van het dashboard te optimaliseren. Daarnaast kunnen best practices op het gebied van dashboarding en data-analyse worden onderzocht om de bruikbaarheid en effectiviteit van het Operational Dashboard verder te verbeteren.

Begrippenlijst

Operational Dashboard	Een visueel hulpmiddel dat operationele gegevens aggregeert en presenteert om de prestaties en efficiëntie van een organisatie te monitoren en analyseren.
Aggregatie	Het combineren van meerdere gegevensbronnen of datapunten om een samengevatte weergave of statistiek te genereren.
Data-analyse	Het proces van het inspecteren, opschonen, transformeren en modelleren van gegevens met als doel bruikbare informatie, conclusies en ondersteuning van besluitvorming te verkrijgen.
KPI (Key Performance Indicator)	Een meetbare waarde die aangeeft hoe effectief een organisatie is in het behalen van belangrijke bedrijfsdoelstellingen.
Jira	Een softwaretoepassing voor projectmanagement en het bijhouden van problemen, ontwikkeld door Atlassian.
Azure	Een verzameling van cloud services en -infrastructuur, ontwikkeld door Microsoft, die organisaties helpt bij het bouwen, implementeren en beheren van applicaties en services.
SonarQube/SonarCloud	Platformen voor statische code-analyse die ontwikkelaars helpen bij het schrijven van schonere en veiligere code.
Pipeline	Een reeks geautomatiseerde processen.
Agent	Een softwarecomponent die op een server of ander apparaat draait en taken uitvoert namens een groter systeem.
Kibana	Een open-source data visualisatie- en dashboarding-tool voor Elasticsearch, ontwikkeld door Elastic.
Elasticsearch	Een open-source, gedistribueerde zoek- en analyse-engine gebaseerd op de Lucene-bibliotheek, ontwikkeld door Elastic.
Logstash	Een open-source gegevensverwerkingspipeline die gegevens uit verschillende bronnen verzamelt, transformeert en naar Elasticsearch stuurt, ontwikkeld door Elastic.
Beats	Lichtgewicht, open-source gegevensverzenders die gegevens van bronnen naar Elasticsearch of Logstash sturen, ontwikkeld door Elastic.

Connectors	Softwarecomponenten die de integratie tussen verschillende systemen of applicaties mogelijk maken door gegevensuitwisseling en communicatie te faciliteren.
Microsoft Teams	Een communicatie- en samenwerkingsplatform ontwikkeld door Microsoft, dat chat, videoconferenties, bestandsopslag en het delen van bestanden mogelijk maakt.
API (Application Programming Interface)	Een set regels en specificaties waarmee softwaretoepassingen met elkaar kunnen communiceren en gegevens kunnen uitwisselen.
Web UI (User Interface)	Een grafische interface die wordt weergegeven in een webbrowser en waarmee gebruikers kunnen communiceren met een softwaretoepassing of -systeem.
PoC (Proof of Concept)	Een prototype of demonstratie van een project om de haalbaarheid en effectiviteit van een idee of concept te testen en te valideren.
Business Intelligence (BI)	Het proces van het verzamelen, analyseren en presenteren van bedrijfsgegevens om beter geïnformeerde beslissingen te nemen en de bedrijfsprestaties te verbeteren.
Data-visualisatie	De grafische weergave van gegevens en informatie, vaak met behulp van diagrammen, grafieken of andere visuele elementen om complexe gegevens eenvoudiger te begrijpen en te interpreteren.
Datawarehouse	Een grootschalig opslagsysteem voor het verzamelen, opslaan en beheren van gestructureerde en ongestructureerde gegevens uit verschillende bronnen binnen een organisatie, gebruikt voor rapportage en data-analyse.
ETL (Extract, Transform, Load)	Een proces in databeheer waarbij gegevens worden geëxtraheerd uit verschillende bronnen, getransformeerd naar een geschikt formaat en vervolgens geladen in een doelsysteem, zoals een datawarehouse.
Real-time data	Gegevens die onmiddellijk of bijna onmiddellijk beschikbaar zijn na het verzamelen, waardoor gebruikers in staat zijn om snel te reageren op veranderende omstandigheden en trends.
Dashboarding	Het proces van het ontwerpen en ontwikkelen van interactieve en visuele hulpmiddelen, zoals dashboards, die gegevens en informatie op een gebruiksvriendelijke en toegankelijke manier presenteren.

Cloud computing	Een model voor het leveren van IT-services waarbij computerbronnen, zoals rekenkracht en opslag, op afstand en via internet beschikbaar worden gesteld in plaats van op lokale hardware.
DevOps	Een set praktijken en cultuur die de samenwerking tussen softwareontwikkeling en IT-activiteiten bevordert, met als doel het verkorten van de systeemontwikkelingslevenscyclus en het leveren van hoogwaardige software.
Gegevensintegratie	Het proces van het combineren van gegevens uit verschillende bronnen en het presenteren van een uniforme en consistente weergave van die gegevens, waardoor het gemakkelijker wordt om te analyseren en inzichten te verkrijgen.
SIEM	Security Information and Event Management (SIEM) is een beveiligingsbeheerbenadering die real-time beveiligingsinformatie en -gebeurtenissen verzamelt, bewaakt en analyseert. SIEM helpt beveiligingsteams bij het proactief opsporen van bedreigingen en snel reageren op beveiligingsincidenten.
OpenAPI	OpenAPI is een specificatie voor het beschrijven, produceren, consumeren en visualiseren van RESTful-webdiensten. Het vereenvoudigt en standaardiseert het ontwerpen, bouwen, documenteren en gebruiken van RESTful API's door een gestandaardiseerd formaat voor het beschrijven van de API's te definiëren.
Swagger	OpenAPI is een specificatie voor het beschrijven, produceren, consumeren en visualiseren van RESTful-webdiensten. Het vereenvoudigt en standaardiseert het ontwerpen, bouwen, documenteren en gebruiken van RESTful API's door een gestandaardiseerd formaat voor het beschrijven van de API's te definiëren.

Figurenlijst

Figuur 1 Eurofins Digital Testing office space op de Corda Campus 13
Figuur 2 Elasticsearch B.V. 15
Figuur 3 Elastic Stack 15
Figuur 4 Kibana 17
Figuur 5 Docker 18
Figuur 6 TypeScript 18
Figuur 7 Node.js 19
Figuur 8 Microsoft Azure 20
Figuur 9 Python 20
Figuur 10 Elastic Beats 21
Figuur 11 Bash 21
Figuur 12 Atlassian Jira 22
Figuur 13 SonarCloud 23
Figuur 14 Voorbeeld van SonarCloud op mijn stageproject 24
Figuur 15 TOML 25
Figuur 16 Voorbeeld van een TOML-configuratie 25
Figuur 17 Test dashboard om ervaring te krijgen met Kibana 29
Figuur 18 Elastic Stack single node docker-compose.yml met SSL 30
Figuur 19 Adapter pattern - Fetch adapter 35
Figuur 20 Adapter pattern - Axios adapter 35
Figuur 21 Vergelijking populaire web frameworks 36
Figuur 22 Sonar integratie - Voorbeeld dashboard 37
Figuur 23 Azure integratie - Discover - Agents 39
Figuur 24 Azure integratie - OpenAPI documentatie 39
Figuur 25 Teams bericht wanneer een Azure DevOps Pipeline Agent down ging 43
Figuur 26 Ansible 45
Figuur 27 Uitgewerkt Jira dashboard 47
Figuur 28 Uitgewerkt Sonar dashboard met zicht op huidige status van projecten 48
Figuur 29 Uitgewerkt Sonar dashboard met zicht op project status over tijd 49
Figuur 30 Uitgewerkt Azure dashboard met zicht op historiek en huidige status van pipelines en agents 50

1 Inleiding

In een steeds meer data gedreven wereld zijn organisaties continu op zoek naar manieren om hun operationele processen te optimaliseren en bedrijfsprestaties te verbeteren. Het vermogen om relevante en bruikbare informatie uit gegevens te extraheren en te visualiseren is cruciaal geworden voor het succes van bedrijven. Operational Dashboards zijn instrumenten die bedrijven helpen om inzicht te krijgen in hun prestaties en efficiëntie door gegevens uit verschillende bronnen te aggregeren, te analyseren en op een begrijpelijke en visuele manier te presenteren. Deze dashboards stellen managers en teams in staat om weloverwogen beslissingen te nemen op basis van real-time informatie en trends.

De aanleiding voor dit project komt voort uit de behoefte van Eurofins Digital Testing om hun operationele gegevens efficiënter te analyseren en te monitoren. Het bedrijf maakt gebruik van verschillende platformen, zoals Jira, Azure en SonarQube/SonarCloud, om hun workflows en processen te beheren. Echter, het gebrek aan een geïntegreerd dashboard om gegevens uit deze platformen samen te brengen en inzicht te bieden in belangrijke prestatie-indicatoren (KPI's) heeft geleid tot de noodzaak voor een oplossing die deze kloof kan overbruggen.

Het onderwerp van deze bachelor scriptie is de ontwikkeling en implementatie van een "Operational Dashboard" dat gegevens uit verschillende platformen aggregeert en visualiseert om Eurofins Digital Testing bruikbare inzichten te bieden in hun bedrijfsvoering. Dit project omvat het onderzoeken van geschikte technologieën en methoden om deze gegevensbronnen te integreren en een gebruiksvriendelijke interface te creëren voor het analyseren van bedrijfsinformatie.

De onderzoeksvraag die in deze scriptie wordt beantwoord, is: "Hoe kan een geïntegreerd Operational Dashboard worden ontwikkeld en geïmplementeerd voor Eurofins Digital Testing om hun operationele gegevens efficiënt te analyseren en monitoren, en bruikbare inzichten te bieden voor het verbeteren van hun bedrijfsprestaties?"

In eerste instantie zal er een onderzoek worden uitgevoerd om inzicht te krijgen in de best practices en methodologieën voor het ontwerpen en implementeren van operationele dashboards. Dit zal ook helpen om de meest geschikte technologieën en tools te identificeren die kunnen worden gebruikt om gegevens uit verschillende bronnen te integreren en te visualiseren.

De onderzoeksopzet omvat een ontwikkelingsgerichte aanpak. Dit project is daarom ook opgedeeld in enkele grote fasen die na elkaar afgewerkt werden. De ontwikkelingsfasen die dit project bevat kan je hieronder terugvinden met extra uitleg over iedere fase.

In de eerste fase van de ontwikkeling van het Operational Dashboard zal ik een data aggregatie en visualisatie oplossing opzetten. Deze zal het mogelijk maken om alle data die we willen verzamelen op te slaan op een centrale plaats waarna deze gevisualiseerd kan worden.

In de tweede fase zal ik een reeks connectoren ontwikkelen en opzetten die de links vormen tussen de verschillende databronnen en onze centrale opslagplaats voor deze gegevens.

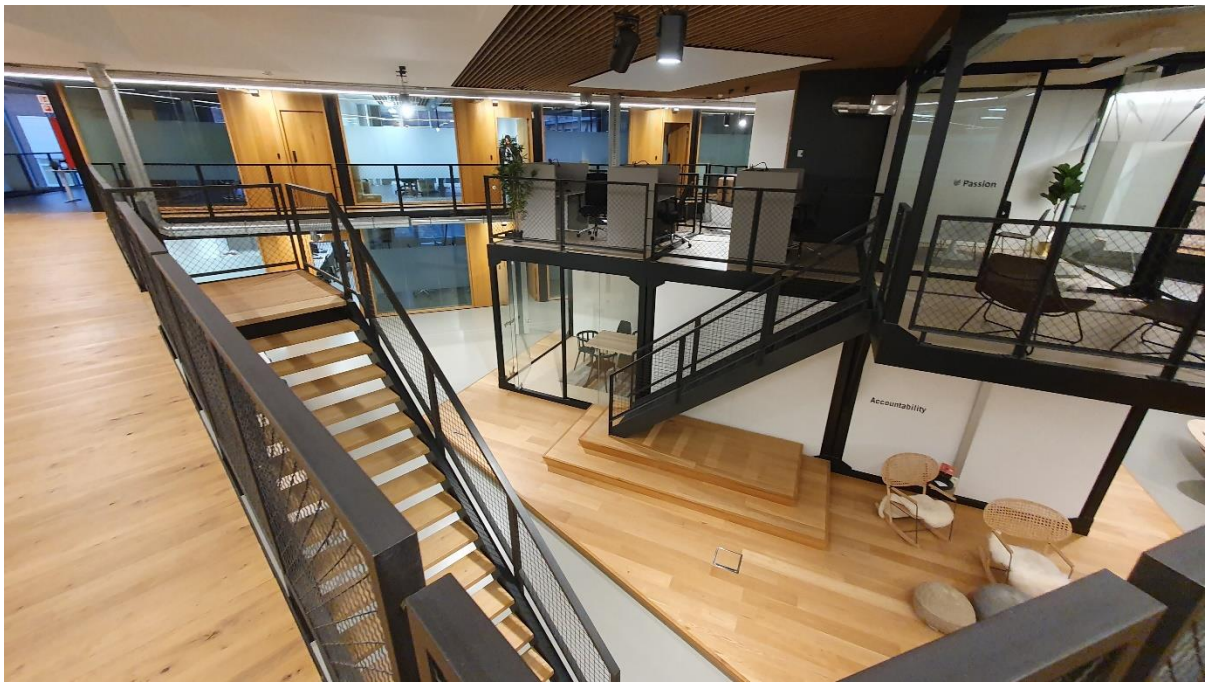
In de derde fase zal ik de alerting mogelijkheden van onze installatie exploreren en configureren. Hier zal ook de koppeling naar Microsoft Teams gemaakt worden zodat alarmen op deze manier in Microsoft Teams terecht kunnen komen.

In de vierde fase zal ik de effectieve plaatsing van de software doen op een server en zal ik er ook voor zorgen dat de configuratie automatisch kan gebeuren door gebruik te maken van Ansible.

In de vijfde fase zal ik de effectieve dashboards uitwerken op maat van Eurofins Digital Testing. Gedurende het ontwikkelingsproces hiervan zullen er iteraties plaatsvinden om het dashboard te verfijnen en te verbeteren op basis van feedback van belanghebbenden en het testen van de functionaliteit. Hierdoor zal het eindresultaat beter aansluiten bij de behoeften en verwachtingen van Eurofins Digital Testing.

Door het volgen van deze onderzoeksopzet zal deze scriptie een praktische oplossing bieden voor het ontwikkelen en implementeren van een effectief Operational Dashboard voor Eurofins Digital Testing, en bijdragen aan het verbeteren van hun bedrijfsprestaties en efficiëntie.

2 Eurofins Digital Testing[1]



Figuur 1 Eurofins Digital Testing office space op de Corda Campus

Eurofins Digital Testing, als wereldwijde speler in digitale testoplossingen en kwaliteitsborging, speelt een cruciale rol bij het ondersteunen van bedrijven bij het ontwikkelen, valideren en optimaliseren van hun digitale producten en diensten.

In de dynamische en snel evoluerende wereld van het Internet of Things (IoT) - waar alledaagse objecten verbonden zijn met het internet om gegevensuitwisseling en communicatie mogelijk te maken - is het essentieel dat bedrijven vertrouwen kunnen hebben in de betrouwbaarheid, veiligheid en efficiëntie van hun digitale systemen, apparaten en content. Eurofins Digital Testing biedt deze zekerheid door middel van uitgebreide test- en validatiediensten.

Het deskundige team van Eurofins Digital Testing, bestaande uit specialisten in diverse disciplines, werkt nauw samen met klanten om hun specifieke behoeften te doorgronden en op maat gemaakte oplossingen aan te bieden die zorgen voor optimale prestaties, veiligheid en efficiëntie van hun digitale systemen. Dit wordt bereikt door een breed scala aan diensten en oplossingen te leveren die gericht zijn op het handhaven van de hoogste kwaliteitsnormen in de industrie.

Een cruciaal aspect van het werk van Eurofins Digital Testing is het ontwerpen en uitvoeren van tests om de prestaties en veiligheid van digitale producten en diensten te beoordelen. Dit kan bijvoorbeeld betrekking hebben op het testen van de snelheid waarmee een apparaat verbinding maakt met het internet, of op het evalueren van de beveiligingsmaatregelen die zijn getroffen om gevoelige gegevens te beschermen tegen onbevoegde toegang.

[1] https://www.resillion.com/

Bovendien biedt Eurofins Digital Testing kwaliteitsborgingsdiensten die bedrijven helpen bij het identificeren en oplossen van eventuele problemen die zich kunnen voordoen tijdens de ontwikkeling en implementatie van hun digitale producten en diensten. Dit kan variëren van het analyseren van de gebruiksvriendelijkheid van een applicatie of website, tot het controleren van de compatibiliteit van een apparaat met verschillende besturingssystemen en netwerken.

Als onderdeel van hun uitgebreide dienstenaanbod helpt Eurofins Digital Testing bedrijven ook om te voldoen aan nationale en internationale regelgeving en normen, en om producten en diensten te ontwikkelen die compatibel zijn met bestaande systemen en apparaten. Hierdoor kunnen bedrijven concurrerend blijven in een steeds veranderende markt en zich aanpassen aan de groeiende eisen van consumenten en zakelijke klanten.

Kortom, Eurofins Digital Testing is een betrouwbare partner voor bedrijven die streven naar uitmuntendheid in hun digitale producten en diensten. Door het bieden van hoogwaardige test- en validatiediensten, kwaliteitsborging en ondersteuning gedurende de gehele levenscyclus van softwareontwikkeling, helpt Eurofins Digital Testing bedrijven om zowel de uitdagingen van de huidige markt als de kansen van de toekomst te benutten.

Het is belangrijk te benadrukken dat Eurofins Digital Testing niet alleen een rol speelt bij het testen van eindproducten, maar ook actief betrokken is bij elke fase van de ontwikkeling. Dit omvat het vaststellen van de vereisten, het ontwerpen van testplannen, het uitvoeren van tests en het analyseren van de resultaten. Door samen te werken met bedrijven, kan Eurofins Digital Testing potentiële problemen vroegtijdig identificeren en oplossen, wat resulteert in kostenbesparingen en een snellere time-to-market.

Naast het leveren van uitgebreide test- en kwaliteitsborgingsdiensten, biedt Eurofins Digital Testing ook opleidingen en consultancydiensten om bedrijven te helpen hun interne processen en vaardigheden te verbeteren. Door het delen van hun expertise en best practices, stelt Eurofins Digital Testing bedrijven in staat om hun eigen test- en kwaliteitsborgingsprocessen verder te verfijnen, wat leidt tot een hogere productkwaliteit en betere concurrentiepositie op de lange termijn.

Om het groeiende aantal cyberdreigingen aan te pakken, heeft Eurofins Digital Testing ook een gespecialiseerde divisie, Eurofins Cyber Security, opgericht. Deze divisie helpt bedrijven om hun digitale systemen, apparaten en content te beschermen tegen cyberaanvallen en gegevensinbreuken, wat een belangrijke zorg is voor bedrijven van elke omvang en in elke sector.

Samenvattend biedt Eurofins Digital Testing een breed scala aan diensten en oplossingen die bedrijven helpen om de kwaliteit, veiligheid en efficiëntie van hun digitale producten en diensten te waarborgen. Door nauw samen te werken met klanten en het bieden van op maat gemaakte oplossingen, speelt Eurofins Digital Testing een cruciale rol in het succes van bedrijven in de informatietechnologiesector en daarbuiten.

3 Toelichting gebruikte technologieën

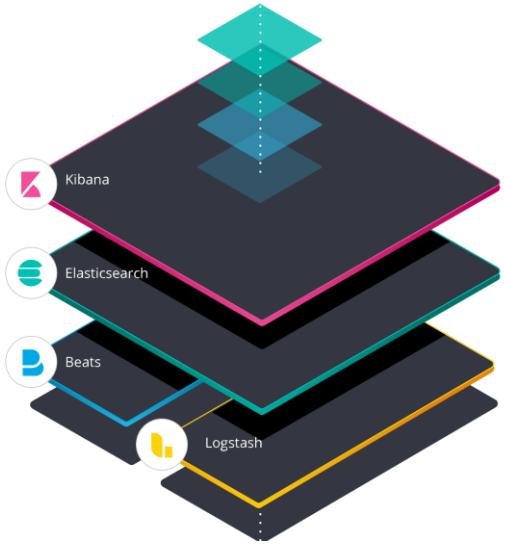
Tijdens het uitwerken van mijn stage ben ik in contact gekomen met verschillende technologieën. Een aantal hiervan beheerste ik al zeer goed, maar een aantal andere waren compleet nieuw voor mij. Hieronder kan je meer informatie terugvinden over alle technologieën en tools waar ik mee in contact ben gekomen tijdens deze stage.

3.1 Elastic Stack[2]

Elastic Stack, ook bekend als ELK Stack, is een krachtige combinatie van drie open-source technologieën: Elasticsearch, Logstash en Kibana. Deze technologieën zijn ontworpen om samen te werken en een geïntegreerde oplossing te bieden voor het verzamelen, analyseren en visualiseren van gegevens, met name loggegevens en events.

Figuur 2 Elasticsearch B.V.

Elasticsearch is een gedistribueerde zoek- en analyse-engine die in staat is om grote hoeveelheden gegevens snel en efficiënt te verwerken. Het is ontworpen om horizontaal schaalbaar te zijn, wat betekent dat het eenvoudig kan worden uitgebreid om grotere datasets te verwerken.



Figuur 3 Elastic Stack

Kibana is een open-source gegevensvisualisatie- en verkenningshulpmiddel gespecialiseerd in grote hoeveelheden streaming- en real-time gegevens. Het helpt gebruikers complexe gegevensstructuren te interpreteren en te begrijpen via visuele weergaven zoals grafieken en histogrammen.

Logstash is een krachtige data-verwerkingspijplijn die gegevens kan verzamelen uit verschillende bronnen, transformeren en doorsturen naar verschillende outputs, zoals Elasticsearch.

Beats zijn lichtgewicht gegevensverzenders die op servers als agents worden geïnstalleerd om verschillende soorten operationele gegevens naar Elasticsearch te sturen, hetzij rechtstreeks, hetzij via Logstash.

[2] https://www.elastic.co/elastic-stack/

De Elastic Stack wordt veel gebruikt in verschillende toepassingen, variërend van monitoring en log-analyse tot beveiligingsinformatie en eventbeheer (SIEM). In dit project wordt Elastic Stack gebruikt om waardevolle inzichten te verkrijgen uit loggegevens en events, wat helpt bij het identificeren van mogelijke problemen en het verbeteren van de algehele prestaties van het systeem.

Door gebruik te maken van de Elastic Stack kunnen we op een vrij eenvoudige en snelle manier onze data gaan verwerken en aggregeren. Dit past perfect bij de noden die Eurofins heeft. Daarnaast geeft deze stack ons ook veel flexibiliteit en kunnen we op korte tijd een krachtige oplossing ontwikkelen.

Tijdens deze stage werd de Elastic Stack gebruikt als centrale oplossing om gegevens op te slaan en te visualiseren.

3.2 Kibana[3]

Kibana is een belangrijk onderdeel van de Elastic Stack, voorheen bekend als de ELK Stack (Elasticsearch, Logstash, Kibana). Het is ontworpen om gebruikers te helpen bij het visualiseren en navigeren door de gegevens die zijn opgeslagen in Elasticsearch indices. Kibana biedt een grafische interface waarmee gebruikers gemakkelijk en snel complexe zoekopdrachten kunnen uitvoeren en de resultaten in verschillende soorten grafieken, tabellen en kaarten kunnen weergeven.

Figuur 4 Kibana

Een van de grootste voordelen van Kibana is dat het naadloos integreert met Elasticsearch, wat betekent dat je real-time zoekopdrachten en analyses kunt uitvoeren op grote hoeveelheden data. Dit is bijzonder nuttig voor tijdreeksanalyse, waarbij je trends en patronen in je gegevens over de tijd kunt onderzoeken.

Kibana biedt een breed scala aan mogelijkheden voor datavisualisatie, waaronder lijndiagrammen, staafdiagrammen, taartdiagrammen, hittekaarten, geografische kaarten, en nog veel meer. Het biedt ook een dashboardfunctie, waarmee je verschillende visualisaties kunt combineren en ordenen om een volledig beeld te krijgen van je gegevens.

Kibana alerting is een specifieke functie binnen de Elastic Stack die gebruikers in staat stelt om waarschuwingen en meldingen te creëren op basis van specifieke gebeurtenissen, drempelwaarden of anomalieën in hun data. Dit is een essentieel hulpmiddel voor teams om proactief te reageren op incidenten, prestatieproblemen of andere belangrijke gebeurtenissen in hun systemen of applicaties. Het voordeel hiervan is dat potentiële problemen kunnen worden geïdentificeerd en aangepakt voordat ze escaleren.

Met Kibana alerting, gebruikers kunnen voorwaarden definiëren die, wanneer aan voldaan wordt, een waarschuwing of melding activeren. Deze voorwaarden kunnen uiteenlopen van het overschrijden van een bepaald foutenpercentage, een ongewoon patroon in logbestanden, tot een verandering in de gemiddelde responstijd van een applicatie. Zodra een waarschuwing wordt geactiveerd, stuurt het systeem een melding naar de gedefinieerde doelgroep. Dit kan individuele teamleden, e-mailgroepen of chatplatforms zoals Microsoft Teams omvatten. De meldingen kunnen in verschillende formaten worden verzonden, zoals e-mails, Slack-berichten of webhook-oproepen naar externe systemen.

Het belangrijkste voordeel van Kibana alerting is dat het teams helpt om snel op de hoogte te zijn van kritieke gebeurtenissen. Dit betekent dat ze onmiddellijk kunnen reageren en problemen kunnen oplossen voordat ze escaleren. Bovendien stelt Kibana alerting teams in staat om hun monitoring- en alarmeringsstrategie te automatiseren, wat leidt tot efficiënter werken.

In deze stage wordt Kibana gebruikt voor de visualisatie van gegevens.

[3] https://www.elastic.co/kibana/

3.3 Docker[4]

Figuur 5 Docker

Docker is een containerisatie platform dat het mogelijk maakt om applicaties en hun afhankelijkheden samen te verpakken in een enkele, draagbare eenheid, bekend als een container. Containers zijn geïsoleerd van het onderliggende besturingssysteem en andere containers, waardoor ze consistent en betrouwbaar kunnen worden uitgevoerd op verschillende platforms en omgevingen.

Docker maakt gebruik van een client-server architectuur, waarbij de Docker-client communiceert met de Docker-daemon om containers te bouwen, uit te voeren en te beheren. Docker maakt het eenvoudig om applicaties te ontwikkelen, testen en implementeren, en verbetert de samenwerking tussen ontwikkelaars en operationele teams.

In dit project wordt Docker gebruikt om het beheer van de applicatie en de bijbehorende services te stroomlijnen. Door gebruik te maken van Docker-containers, kunnen ontwikkelaars hun applicaties snel en eenvoudig implementeren, updaten en schalen, wat resulteert in een snellere time-to-market en verbeterde operationele efficiëntie.

Docker werd in deze stage gebruikt om alle applicaties en integraties eenvoudig te kunnen laten functioneren. Het werd ook gebruikt als manier om gemakkelijk een reproduceerbare omgeving op te zetten.

3.4 TypeScript[5]

Figuur 6 TypeScript

TypeScript is een open-source programmeertaal die is ontwikkeld door Microsoft als een uitbreiding van JavaScript, de meest gebruikte programmeertaal voor webontwikkeling.

TypeScript voegt optionele statische typen toe aan JavaScript, waardoor het gemakkelijker wordt om grootschalige, robuuste en onderhoudbare codebases te bouwen. Statische typen helpen bij het vroegtijdig opsporen van fouten en het verbeteren van de codekwaliteit door ontwikkelaars beter inzicht te geven in de structuur en het gedrag van hun code.

TypeScript biedt ook moderne taalconstructies en functies die nog niet beschikbaar zijn in standaard JavaScript, zoals klassen, modules, interfaces en decorators. In dit project wordt TypeScript gebruikt voor de ontwikkeling van de Microsoft Teams alerting integratie, waarbij het helpt om een meer gestructureerde en onderhoudbare codebase te creëren.

TypeScript werd gebruikt voor de meeste integraties te bouwen tussen verschillende diensten.

[4] https://www.docker.com/
[5] https://www.typescriptlang.org/

3.5 Tsoa[6]

Tsoa is een open-source bibliotheek voor het bouwen van RESTful API's met behulp van TypeScript en de OpenAPI/Swagger-specificatie. Tsoa automatiseert veel aspecten van API-ontwikkeling, zoals het genereren van API-documentatie, het valideren van invoerparameters en het creëren van routehandlers.

Door gebruik te maken van TypeScript en OpenAPI/Swagger, biedt tsoa een sterk getypeerde en goed gedocumenteerde API die gemakkelijk te begrijpen en te consumeren is door zowel ontwikkelaars als eindgebruikers. In dit project wordt tsoa gebruikt voor het ontwikkelen van een gestandaardiseerde en goed gedocumenteerde API voor Sonar en Azure integratie, waardoor de samenwerking tussen teams en de bruikbaarheid van de API wordt verbeterd.

Tsoa werd gebruikt voor de Sonar en de Azure integraties om deze integraties universeel integreerbaar te maken.

3.6 Node.js[7]

Node.js is een open-source, platformonafhankelijke JavaScript runtime-omgeving die het mogelijk maakt om JavaScript-code server-side uit te voeren. Dit maakt het mogelijk om webapplicaties te bouwen met één programmeertaal voor zowel client- als serverzijde, wat resulteert in een meer gestroomlijnde en samenhangende ontwikkelervaring.

Figuur 7 Node.js

Node.js maakt gebruik van de V8 JavaScript-engine van Google, die is geoptimaliseerd voor hoge prestaties en een snelle uitvoering van JavaScript-code. Bovendien maakt Node.js gebruik van een asynchrone, op gebeurtenissen gebaseerde architectuur, waardoor het goed kan presteren bij het afhandelen van een groot aantal gelijktijdige verbindingen en verzoeken.

In dit project wordt Node.js v18 gebruikt als de runtime-omgeving voor het uitvoeren van de applicatie en bijbehorende services. Door gebruik te maken van Node.js, kunnen ontwikkelaars gemakkelijk schaalbare en krachtige server-side applicaties bouwen met behulp van vertrouwde JavaScript-technieken en -bibliotheken.

Node.js werd gebruikt in deze stage om de TypeScript software te kunnen uitvoeren.

[6] https://tsoa-community.github.io/docs/
[7] https://nodejs.org/en

3.7 Azure DevOps JavaScript API library[8]

De Azure DevOps JavaScript API-bibliotheek, ook bekend als “azure-devops-node-api”, is een open-source bibliotheek die een eenvoudige en krachtige JavaScript-interface biedt voor de Azure DevOps REST API. Deze bibliotheek stelt ontwikkelaars in staat om eenvoudig Azure DevOps-services te integreren met hun Node.js applicatie.

Figuur 8 Microsoft Azure

3.8 Python[9]

Python is een populaire, veelzijdige en open-source programmeertaal die bekend staat om zijn leesbaarheid, eenvoud en uitgebreide standaardbibliotheek. Python wordt veel gebruikt in verschillende domeinen, zoals webontwikkeling, data-analyse, kunstmatige intelligentie en automatisering.

Figuur 9 Python

In dit project wordt Python gebruikt voor de Pytest logging integratie met Elasticsearch. Pytest is een krachtig en flexibel testing framework voor Python, dat het eenvoudig maakt om tests te schrijven, uit te voeren en te organiseren. Door Python te gebruiken in combinatie met Pytest, kunnen ontwikkelaars snel en effectief tests uitvoeren en de resultaten opslaan in Elasticsearch voor verdere analyse en visualisatie.

[8] https://github.com/microsoft/azure-devops-node-api
[9] https://www.python.org/

3.9 Filebeat[10]

Filebeat is een lichtgewicht logverzamelaar en -verzender die deel uitmaakt van de Elastic Stack. Het is ontworpen om loggegevens van verschillende bronnen te verzamelen, te verwerken en door te sturen naar Elasticsearch of Logstash. Filebeat is eenvoudig te configureren en te beheren, en kan efficiënt grote hoeveelheden loggegevens verwerken zonder veel systeembronnen te verbruiken.

Figuur 10 Elastic Beats

In dit project wordt Filebeat gebruikt voor zowel de Pytest logging integratie als de JUnit integratie met Elasticsearch. Door gebruik te maken van Filebeat, kunnen ontwikkelaars eenvoudig loggegevens verzamelen van hun testsuites en deze doorsturen naar Elasticsearch voor verdere analyse en monitoring. Dit helpt bij het identificeren van problemen en het verbeteren van de kwaliteit van de code.

3.10 Bash[11]

Bash is een Unix-shell en command-line interface die veel wordt gebruikt in Linux- en macOS-omgevingen. Het is een krachtig en flexibel hulpmiddel voor het uitvoeren van commando's, het schrijven van scripts en het automatiseren van taken.

Figuur 11 Bash

In dit project wordt Bash gebruikt voor de JUnit integratie met Elasticsearch. JUnit is een populair testing framework voor Java, dat ontwikkelaars helpt bij het schrijven en uitvoeren van unit tests voor hun code. Door Bash-scripts te gebruiken om JUnit-testresultaten te verwerken, kunnen ontwikkelaars de testresultaten eenvoudig doorsturen naar Elasticsearch voor verdere analyse en visualisatie. Dit helpt bij het waarborgen van de kwaliteit van de code en het identificeren van eventuele problemen.

[10] https://www.elastic.co/beats/filebeat
[11] https://nl.wikipedia.org/wiki/Bash

3.11 Atlassian Jira[12]

Atlassian Jira is een toonaangevende en populaire projectmanagement- en issue-tracking software die organisaties ondersteunt in het organiseren, plannen, volgen en rapporteren van hun werk. Wat oorspronkelijk werd ontwikkeld als een instrument voor softwareontwikkelingsteams om bugs en problemen bij te houden, heeft Jira zich in de loop der jaren geëvolueerd tot een veelzijdige tool, inmiddels gebruikt door diverse teams van IT tot marketing en HR.

Figuur 12 Atlassian Jira

Jira onderscheidt zich door zijn aanpasbaarheid en flexibiliteit, die teams in staat stelt hun werk te structureren op basis van hun specifieke behoeften en workflows. In de kern van Jira is het concept van een "issue" - een taak, bug, verbetering of ander werkitem. Deze issues kunnen worden toegewezen aan teamleden, prioriteiten krijgen en door het gehele ontwikkelingsproces heen worden gevolgd.

Een cruciaal kenmerk van Jira is de mogelijkheid om aangepaste workflows te creëren. Deze workflows, die de verschillende statussen en transities weergeven die een issue doorloopt tijdens zijn levenscyclus, kunnen op maat worden gemaakt om het proces van issue-tracking nauwkeurig af te stemmen op de behoeften van het team.

Daarnaast biedt Jira uitgebreide rapportage- en visualisatietools. Teams kunnen diverse rapporten genereren, variërend van de voortgang van het project tot teamprestaties en softwarekwaliteit. Deze rapporten kunnen visueel worden weergegeven, waardoor ze eenvoudiger te begrijpen en te delen zijn.

Jira ondersteunt ook een breed scala aan integraties met andere tools, waaronder andere producten van Atlassian zoals Confluence en Bitbucket, evenals externe tools zoals Slack, GitHub, Zendesk en meer. Deze integraties helpen teams hun workflows te stroomlijnen en hun productiviteit te verhogen.

Verkrijgbaar in zowel cloudgebaseerde als zelf-gehoste versies, en met verschillende prijsplannen gebaseerd op teamgrootte en benodigde functies, is Jira toegankelijk voor een breed scala aan organisaties, van kleine bedrijven tot grote ondernemingen.

[12] https://www.atlassian.com/nl/software/jira

3.12 SonarCloud / SonarQube[13]

SonarCloud en SonarQube zijn beide producten van SonarSource, een bedrijf dat gespecialiseerd is in tools voor het analyseren en verbeteren van codekwaliteit.

Figuur 13 SonarCloud

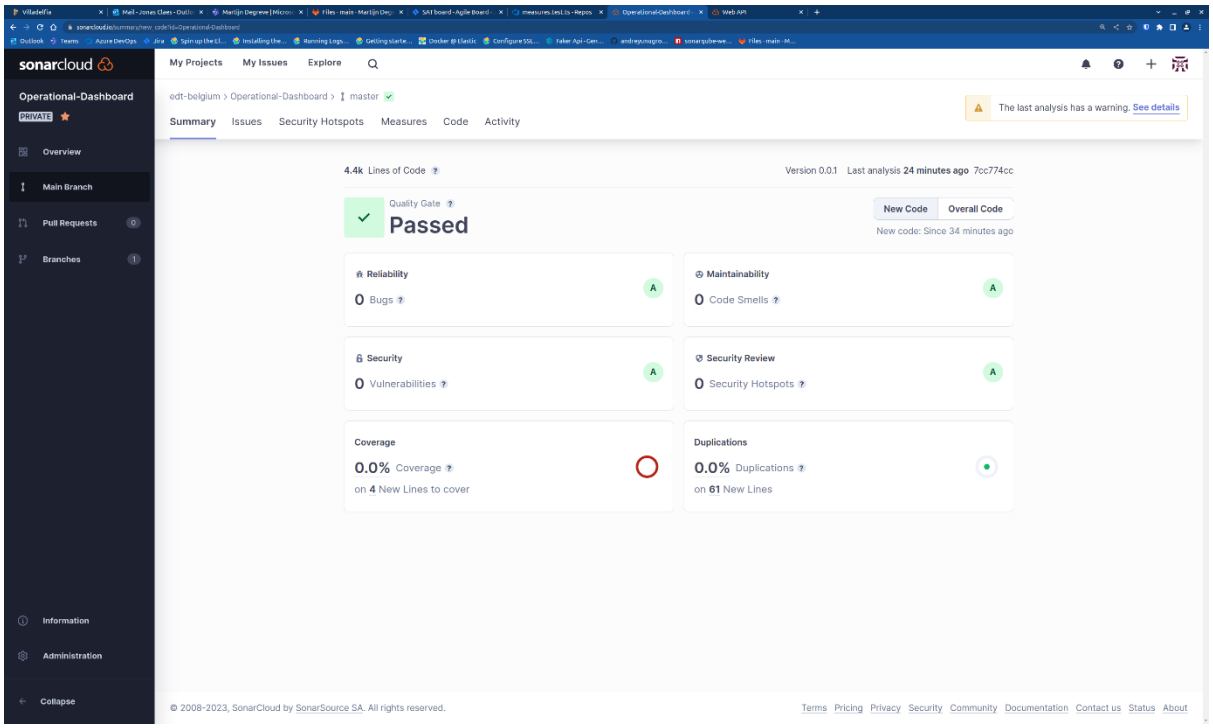
SonarCloud is een cloudgebaseerde service die automatisch broncode analyseert en feedback geeft over codekwaliteit, beveiliging, betrouwbaarheid en onderhoudbaarheid. Het is een software as a service (SaaS)-oplossing die is ontworpen om te integreren met populaire ontwikkelingsplatforms en Continuous Integration (CI) / Continuous Deployment (CD) pipelines, zoals GitHub, Bitbucket en Azure DevOps. SonarCloud kan een verscheidenheid aan programmeertalen analyseren en kan issues opsporen zoals bugs, beveiligingslekken, code smells (code die mogelijk problemen veroorzaakt), en technische schuld (de extra ontwikkelingstijd die in de toekomst nodig zou kunnen zijn als gevolg van het kiezen voor de snelle, maar niet noodzakelijkerwijs beste, oplossing nu).

SonarQube, aan de andere kant, is een open-source platform dat een vergelijkbare set van functies biedt als SonarCloud, maar dat lokaal kan worden geïnstalleerd en geconfigureerd in uw eigen infrastructuur. Het kan ook een breed scala aan programmeertalen analyseren en biedt gedetailleerde rapporten over de kwaliteit van de code, beveiligingsproblemen, duplicatie van code, enz. Het kan ook worden geïntegreerd in CI/CD pipelines voor automatische code-analyse.

Beide tools werken op een vergelijkbare manier. Zodra de code is geschreven en in de repository is gepusht, wordt de code-analyse geactiveerd (dit kan handmatig of automatisch zijn, afhankelijk van hoe de integratie is opgezet). De tool analyseert de code en geeft feedback over potentiële problemen en verbeterpunten. Dit helpt ontwikkelaars om problemen vroeg te identificeren en te corrigeren, wat kan leiden tot een hogere codekwaliteit, betere prestaties, en verbeterde beveiliging.

[13] https://www.sonarsource.com/products/sonarcloud/

Hieronder kan je een voorbeeld zien van hoe SonarCloud eruit ziet op mijn project:



Figuur 14 Voorbeeld van SonarCloud op mijn stageproject

Hier kan je ook al enkele waarden zien terugkomen zoals bugs, code smells, vulnerabilities, security hotspots, coverage en duplications.

3.13 TOML[14]

Figuur 15 TOML

TOML, oftewel Tom's Obvious, Minimal Language, is een eenvoudig configuratiebestandforma at dat is ontworpen om gemakkelijk te lezen en te schrijven door zowel mensen als machines. TOML is gemaakt door Tom Preston-Werner en heeft als doel om een duidelijkere en eenvoudigere manier te bieden voor het opslaan en delen van configuratie-informatie in vergelijking met andere formaten zoals JSON, YAML of INI.

TOML-bestanden zijn tekstbestanden met een. toml-extensie en gebruiken een eenvoudige, op key-value gebaseerde syntax. Ze ondersteunen ook hiërarchische structuren door het gebruik van secties en subsecties, die worden aangegeven met vierkante haakjes ([]).

Enkele basiskenmerken van TOML zijn:

- Commentaren: Commentaren beginnen met een hekje (#) en lopen door tot het einde van de regel.
- Key-value pairs: Keys en values worden gescheiden door een gelijkteken (=).
- Datatypes: TOML ondersteunt diverse datatypes, zoals strings, integers, floats, booleans, datums en tijden.
- Secties en subsecties: Hiërarchische structuren kunnen worden gemaakt met behulp van secties en subsecties, die worden aangegeven door vierkante haakjes ([]).

Een voorbeeld van een TOML-configuratie:

```toml
# This is a TOML document

title = "TOML Example"

[owner]
name = "Tom Preston-Werner"
dob = 1979-05-27T07:32:00-08:00

[database]
enabled = true
ports = [ 8000, 8001, 8002 ]
data = [ ["delta", "phi"], [3.14] ]
temp_targets = { cpu = 79.5, case = 72.0 }

[servers]

[servers.alpha]
ip = "10.0.0.1"
role = "frontend"

[servers.beta]
ip = "10.0.0.2"
role = "backend"
```

Figuur 16 Voorbeeld van een TOML-configuratie

[14] https://toml.io/en/

3.14 Microsoft Teams[15]

Microsoft Teams is een geïntegreerd communicatie- en samenwerkingsplatform dat deel uitmaakt van de Microsoft 365-suite. De essentie van Microsoft Teams is het creëren van een centrale hub voor werkplekcommunicatie en -samenwerking. Het stelt gebruikers in staat om direct te communiceren via chat, audio en video, documenten en bestanden te delen, en gezamenlijk aan documenten te werken.

Webhooks zijn een krachtig hulpmiddel in Microsoft Teams waarmee Teams kan integreren met externe diensten. Een webhook is in wezen een manier voor een app om real-time informatie te verstrekken aan andere apps, waardoor ze automatisch kunnen reageren op bepaalde gebeurtenissen.

In de context van Microsoft Teams kunnen webhooks worden gebruikt om meldingen en updates van externe diensten rechtstreeks in een Teams-kanaal te sturen. Dit is bijzonder nuttig voor het volgen van projectupdates, het ontvangen van meldingen van monitoringdiensten, het bijhouden van klantinteracties en nog veel meer.

Bijvoorbeeld, een ontwikkelteam kan een webhook instellen om een melding te ontvangen in hun Teams-kanaal telkens wanneer er een nieuwe bug wordt gemeld in hun bug tracking systeem. Of een verkoopafdeling kan een webhook gebruiken om een melding te krijgen wanneer er een nieuwe verkooplead binnenkomt via hun CRM-systeem.

Het instellen van een webhook in Microsoft Teams omvat in principe het creëren van een webhook-URL voor het specifieke Teams-kanaal waar u meldingen wilt ontvangen. Deze URL wordt vervolgens ingevoerd in de externe dienst die de meldingen zal sturen.

Een van de grote voordelen van webhooks is dat ze een vrij eenvoudige manier bieden om real-time integraties tussen diensten te creëren. Ze vereisen geen polling (het regelmatig checken van een dienst op updates), wat betekent dat ze minder resources verbruiken en sneller kunnen reageren op gebeurtenissen.

[15] https://www.microsoft.com/nl-be/microsoft-teams/group-chat-software

4 Realisatie

In dit hoofdstuk van mijn bachelor scriptie zal ik de implementatie en integratie van verschillende technologieën bespreken die zijn gebruikt om mijn project tot leven te brengen. Het doel van dit hoofdstuk is om de stappen die ik heb genomen en de uitdagingen die ik ben tegengekomen tijdens het realiseren van het project in detail te beschrijven.

Allereerst begin ik met de oriëntatie, waarin ik de benodigde kennis en vaardigheden verwerf om met de gekozen technologieën aan de slag te gaan. Vervolgens zal ik ingaan op de Elastic Stack, de technologie die als basis dient voor het project, en de connectoren die nodig zijn voor de integratie met andere platformen en tools.

In het gedeelte over Atlassian Jira en Sonar zal ik uitleggen hoe deze specifieke tools zijn geïntegreerd en hoe ze een belangrijke rol spelen in het project. Hierbij zal ik ook ingaan op de Sonar library en de Sonar integratie.

De integratie met Azure wordt vervolgens besproken, gevolgd door de verschillende programmeertalen en tools die zijn gebruikt: Python integratie, Java integratie en TypeScript/TestWizard integratie.

In het gedeelte over alerting zal ik toelichten hoe Microsoft Teams is gebruikt om meldingen te sturen. Daarna zal ik ingaan op Kibana, het platform dat wordt gebruikt voor het visualiseren van gegevens dat ook de meldingen genereert.

In de deployment-sectie zal ik de serverinfrastructuur, het gebruik van Infrastructure-as-code en het opzetten van het systeem bespreken. Tot slot zal ik de gecreëerde dashboards beschrijven.

Dit hoofdstuk biedt inzicht in de praktische kant van mijn stage.

4.1 Oriëntatie

Tijdens de oriëntatiefase van het project was het essentieel om goed inzicht te krijgen in zowel Eurofins als de specifieke doelstellingen en uitdagingen van het project. Deze fase was cruciaal om een solide basis te leggen voor het project en om het voorbereidend onderzoek uit te voeren dat nodig was om een effectieve oplossing te ontwikkelen.

Het eerste doel van de oriëntatiefase was om vertrouwd te raken met Eurofins als bedrijf, de bedrijfscultuur en de betrokken teamleden. Dit was belangrijk om een effectieve samenwerking binnen het team te waarborgen en om inzicht te krijgen in de bedrijfsprocessen en -doelen. Het kennen van de organisatie en de mensen met wie ik zou samenwerken, hielp me om het project beter te begrijpen en mijn aanpak af te stemmen op de verwachtingen en behoeften van het bedrijf.

Vervolgens richtte ik me op het verkennen van de wat en de waarom van het project. Dit hield in dat ik de specifieke problemen en uitdagingen onderzocht die het project wilde aanpakken, en de redenen waarom deze kwesties belangrijk waren voor Eurofins. In dit stadium onderzocht ik de technologieën en tools die relevant waren voor het project, zoals Elasticsearch, Elastic Observability, Kibana, Elastic APM, Jira API, Azure API en SonarCloud API. Door deze technologieën te onderzoeken, kon ik beter begrijpen hoe ze konden bijdragen aan het oplossen van de problemen waarmee Eurofins werd geconfronteerd en hoe ze konden worden geïntegreerd om een effectieve oplossing te bieden.

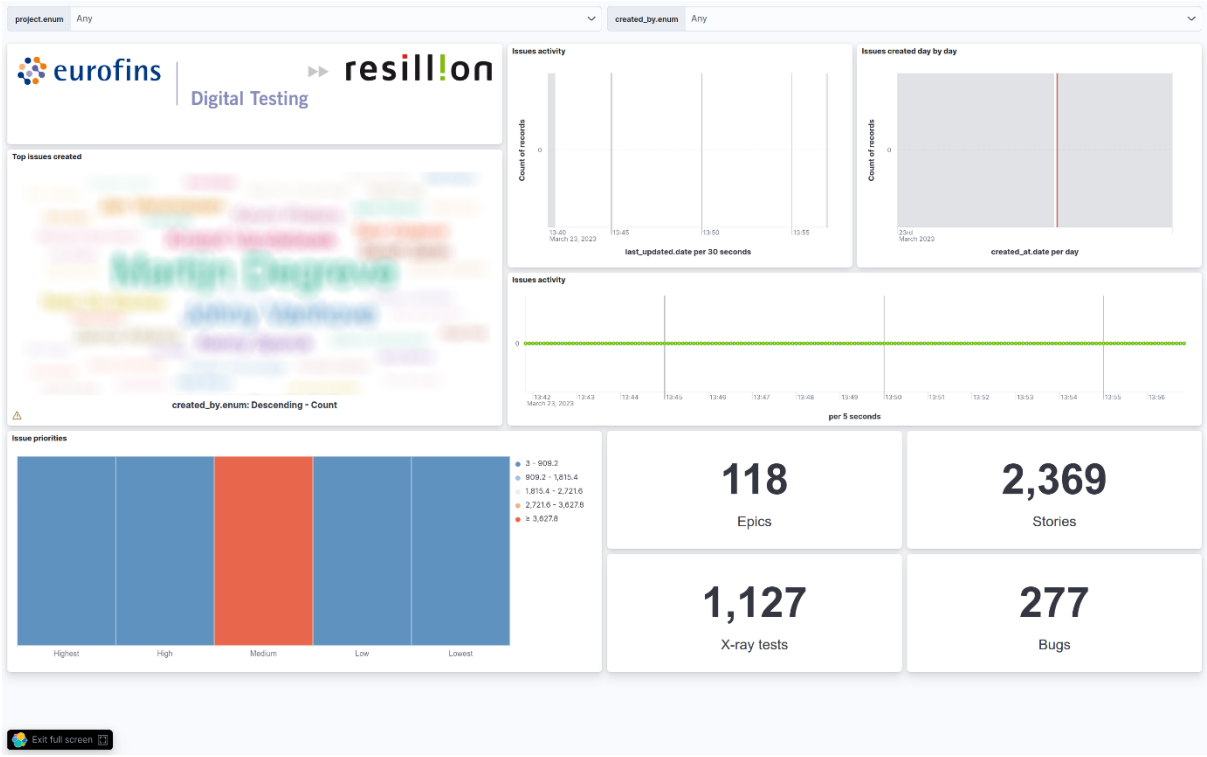
Het onderzoek naar de Jira API en Azure API was bijzonder relevant voor het project, aangezien deze tools een cruciale rol zouden spelen bij het ontwikkelen van een systeem dat de efficiëntie en effectiviteit van de ontwikkelingsprocessen van Eurofins zou verbeteren. Door de werking en mogelijkheden van deze API's te begrijpen, kon ik beter beoordelen hoe ze konden worden geïntegreerd in de oplossing en hoe ze zouden bijdragen aan het bereiken van de doelstellingen van het project.

De oriëntatiefase eindigde met het uitwerken van een duidelijk plan van aanpak, dat teruggevonden kan worden op mijn portfolio, voor het project, met een focus op de technologieën en tools die zouden worden gebruikt en de specifieke stappen die zouden worden genomen om de doelen te bereiken. Deze fase was cruciaal om een sterke basis te leggen voor het verdere verloop van het project en om ervoor te zorgen dat alle betrokken partijen op één lijn zaten met betrekking tot de verwachtingen en doelen.

4.2 Fase 1 – Dataopslag en verwerking

In het begin van deze stage kende ik niets over de Elastic Stack, maar op een week tijd heb ik al een initiële setup kunnen doen van de Elastic Stack. Dit had ik in eerste instantie lokaal op mijn laptop gedaan via Docker en Docker Compose[16]. Vanaf dat dit onderdeel draaiende was heb ik kunnen beginnen experimenteren met de Elastic Stack.

Daarnaast heb ik ook al meteen de Jira integratie kunnen gebruiken en op die manier data kunnen verzamelen. Deze gegevens kwamen later goed van pas omdat ik hiermee de eerste test dashboards heb kunnen bouwen. Hieronder kan je een voorbeeld terugvinden van dit test dashboard. In dit voorbeeld zijn namen onleesbaar gemaakt in verband met GDPR.



Figuur 17 Test dashboard om ervaring te krijgen met Kibana

Gedurende mijn stage heb ik steeds meer bijgeleerd over de Elastic Stack en ook hoe deze onderliggend in elkaar zit. Dit heb ik vooral geleerd uit de documentatie van Elastic en uit problemen die ik ben tegengekomen tijdens het gebruik van de Elastic Stack.

Het heeft mij enorm geholpen dat de documentatie die Elastic aanlevert van de hoogste kwaliteit is. Hier staat duidelijk in hoe bepaalde features werken en hoe je deze kan configureren. Daarnaast geven ze ook heel veel voorbeelden en duidelijke uitleg bij bijvoorbeeld de API's die Elasticsearch aanbiedt.

Uiteindelijk is de Elastic Stack het centrale onderdeel geworden van deze stage. De Elastic Stack speelt een grote rol doordat het alle onderdelen verbindt en een degelijk product is dat onderhouden wordt door Elastic met een breed gamma aan flexibiliteit en mogelijkheden.

16 https://docs.docker.com/compose/

Hieronder staat een voorbeeld van de Docker Compose file die ik heb uitgewerkt om de Elastic Stack lokaal te kunnen uitvoeren. Hier zijn details uitgehaald omdat dit anders niet leesbaar zou zijn gezien de lengte van dit bestand.

```yaml
version: '3'

services:
  es01:
    image: docker.elastic.co/elasticsearch/elasticsearch:${STACK_VERSION}
    volumes:
      - certs:/usr/share/elasticsearch/config/certs
      - esdata01:/usr/share/elasticsearch/data
    profiles:
      - prod
      - setup
    ports:
      - ${ES_PORT}:9200
    environment:
      - node.name=es01
      - cluster.name=${CLUSTER_NAME}
      - cluster.initial_master_nodes=es01
      - ELASTIC_PASSWORD=${ELASTIC_PASSWORD}
      - bootstrap.memory_lock=true
      - xpack.security.enabled=true
      - xpack.security.http.ssl.enabled=true
      - xpack.security.http.ssl.key=certs/es01/es01.key
      - xpack.security.http.ssl.certificate=certs/es01/es01.crt
      - xpack.security.http.ssl.certificate_authorities=certs/ca/ca.crt
      - xpack.security.transport.ssl.enabled=true
      - xpack.security.transport.ssl.key=certs/es01/es01.key
      - xpack.security.transport.ssl.certificate=certs/es01/es01.crt
      - xpack.security.transport.ssl.certificate_authorities=certs/ca/ca.crt
      - xpack.security.transport.ssl.verification_mode=certificate
      - xpack.license.self_generated.type=${LICENSE}
    mem_limit: ${MEM_LIMIT}
    ulimits:
      memlock:
        soft: -1
        hard: -1
    healthcheck:
      test:
        [
          "CMD-SHELL",
          "curl -s --cacert config/certs/ca/ca.crt https://es01:9200 | grep -q 'missing authentication credentials'",
        ]
      interval: 10s
      timeout: 10s
      retries: 120

  kibana:
    ...

  enterprisesearch:
    ...

volumes:
  certs:
    driver: local
  esdata01:
    driver: local
  kibanadata:
    driver: local
  enterprisesearchdata:
    driver: local
```

Figuur 18 Elastic Stack single node docker-compose.yml met SSL

In de uiteindelijke installatie op een Linux server is een 3-node Elasticsearch cluster geconfigureerd.

Een 3-node Elasticsearch cluster betekent dat je drie verschillende servers (of "nodes") hebt waarop Elasticsearch draait. Elke node bevat een deel van je data en kan onafhankelijk van de andere nodes werken. Dit biedt verschillende voordelen:

- Redundantie: Omdat elke node een deel van de data bevat, als er een node uitvalt, zullen de andere twee nodes nog steeds in staat zijn om de dienst voort te zetten. Bovendien, als je replica-shards gebruikt, dan zijn er kopieën van je data op verschillende nodes, wat betekent dat je zelfs bij een storing van een node geen data verliest.
- Beschikbaarheid: Dankzij de redundantie die door meerdere nodes wordt geboden, zijn je gegevens altijd beschikbaar, zelfs als een van de nodes uitvalt. Dit is cruciaal voor systemen die hoge beschikbaarheidseisen hebben.
- Sharding: Met een multi-node cluster kun je sharding toepassen, wat betekent dat je je data kunt splitsen over verschillende nodes. Dit verbetert niet alleen de zoekprestaties (omdat zoekopdrachten parallel over meerdere shards kunnen worden uitgevoerd), maar het stelt je ook in staat om meer data op te slaan dan een enkele node zou kunnen bevatten.
- Horizontale schaalbaarheid: Als je meer data hebt dan je huidige cluster aankan, kun je eenvoudigweg meer nodes toevoegen aan het cluster om de capaciteit te vergroten. Dit staat bekend als horizontale schaalbaarheid en is een krachtig kenmerk van Elasticsearch.

Samengevat, door het configureren van een 3-node Elasticsearch cluster op een Linux-server, is een robuust, betrouwbaar en schaalbaar systeem gecreëerd dat in staat is om grote hoeveelheden data efficiënt te beheren.

4.3 Fase 2 – Connectors

In deze fase kan je meer informatie terugvinden over de verschillende connectoren die data uit verschillende bronnen verzamelen, transformeren en versturen naar de centrale dataopslag.

4.3.1 Atlassian Jira integratie

Binnen Eurofins wordt Jira veel gebruikt door verschillende teams. In ons team wordt het gebruikt voor de tracking van tickets zoals een test die geschreven moet worden of een bug die opgelost moet worden.

Ieder ticket zal uiteindelijk toegewezen worden aan een persoon en een bepaalde prioriteit krijgen. Vanuit een BI perspectief is het dan interessant om te weten hoelang het duurt voor iemand een work item opneemt of hoelang het duurt om een bepaalde feature uit te werken op basis van een aantal story points. Dit kan leiden tot data-driven decisions zoals een extra opleiding geven bijvoorbeeld.

De data die in Jira bestaat, kunnen we uitlezen aan de hand van een API. Deze API is goed gedocumenteerd en heeft ook vele implementaties in verschillende programmeertalen en tools.

Elasticsearch heeft meerdere bestaande integraties voor Jira. Deze integraties vallen onder de Enterprise Search kant van de Elastic Stack. Dit werkt op basis van plugins die geladen kunnen worden in de Enterprise Search software, of via een event-based protocol, losgekoppeld van de Enterprise Search software.

Ik ben begonnen met de native integratie van Jira op te zetten. Dit hield uiteindelijk niet veel in. Hiervoor heb ik op het Atlassian Developer portaal een OAuth 2.0[17] applicatie aangemaakt, waarna ik deze applicatie toegang heb gegeven tot de Jira, en de credentials heb verkregen.

Deze credentials kon ik dan gemakkelijk invullen in de Enterprise Search software, waarna ik de koppeling gemaakt heb, en de data geïndexeerd werd. Dit ging heel snel.

Nadat deze data geïndexeerd werd, heb ik gekeken welke data we exact konden verkrijgen via deze integratie. In eerste instantie leek dit oké te zijn voor onze doeleinden.

Nadien toen ik begon met de dashboards te bouwen, merkten we op dat deze integratie toch niet genoeg data ophaalde. Dit had te maken met de links tussen bijvoorbeeld user stories en epics. Gezien deze link er niet was, was het onmogelijk om per project te zien hoeveel tickets zich hieronder bevinden.

Om die reden hebben we besloten om de andere integratie te gebruiken. Deze vereiste meer werk om in te stellen omdat deze integratie werkt met het event-based protocol. Deze integratie heb ik uiteindelijk geïnstalleerd via een Docker container, die verbindt met de Enterprise Search software over het event-based protocol.

[17] https://oauth.net/2/

Nadat deze integratie werkende was, waren we tevreden met de data die we op dat moment konden verzamelen uit Jira.

4.3.2 Sonar integratie

In tegenstelling tot de Jira integratie was de Sonar integratie een pak uitgebreider. Voor Sonar bestonden er geen goede en onderhouden libraries of integraties. Om deze redenen heb ik besloten om een API-client te bouwen en deze daarna te gebruiken in een applicatie die de link legt tussen Sonar en Elasticsearch.

Binnen Eurofins wordt Sonar op dit moment veel gebruikt voor codekwaliteit te evalueren. Sonar biedt hiervoor een aantal simpele grafieken die snel kunnen visualiseren wat de status is. We willen de data uit Sonar halen en integreren in onze eigen oplossing zodat we belangrijke KPI's uit de code kunnen verkrijgen.

4.3.3 Voorzien van eigen Sonar library

Om de gegevens uit de Sonar API te halen, kunnen we werken met HTTP requests. Vaak bieden API's documentatie over hoe deze requests eruit moeten zien en wat ze terugsturen.

Voor Sonar was dit ook het geval, alleen miste deze documentatie een cruciaal onderdeel. Veel API's worden gedocumenteerd aan de hand van een specificatie gekend als OpenAPI[18] of Swagger[19]. Vanaf deze specificatie kunnen we dan libraries genereren voor verschillende programmeertalen.

De Sonar API miste deze specificatie waardoor het niet mogelijk was om een library te genereren. Dit betekende dat ik zelf de API-routes moest gaan implementeren en zelf een library moest gaan bouwen.

Ik ben dan begonnen met deze library op te zetten met TypeScript op een modulaire manier met ondersteuning voor verschillende HTTP-adapters zoals "axios"[20] en "fetch"[21]. In deze library heb ik alle mogelijke HTTP requests toegevoegd zodat we deze in de toekomst ook op andere plaatsen zouden kunnen gebruiken zonder al te veel moeite.

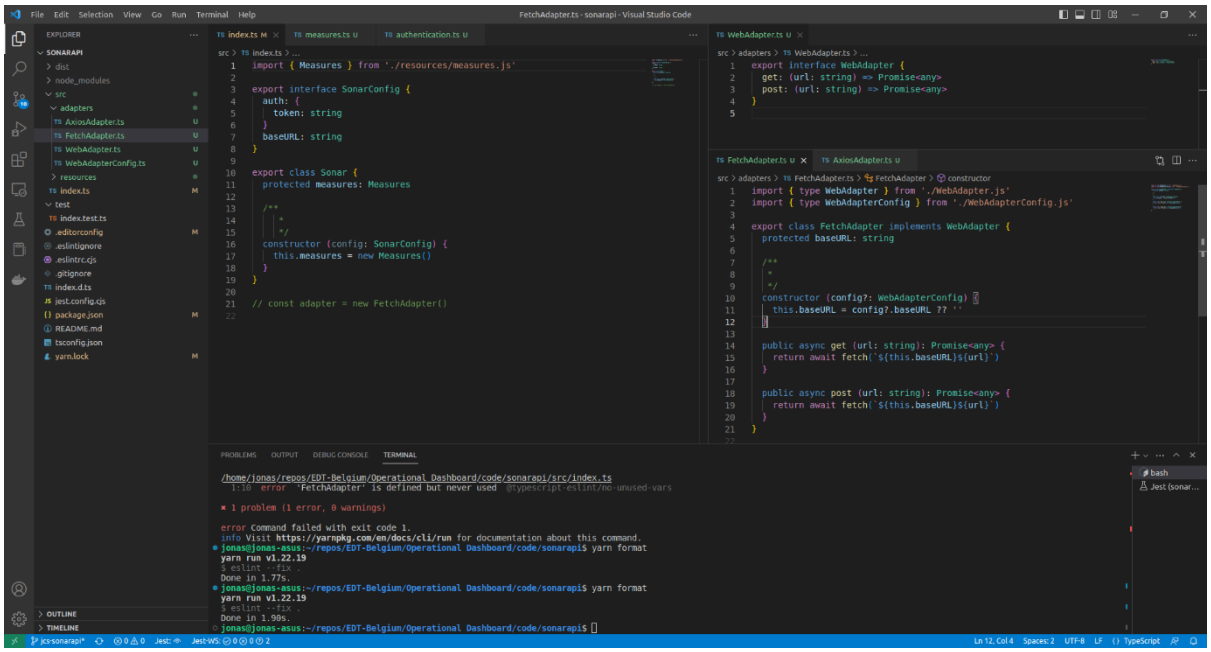
Dit bleek uiteindelijk een goede keuze geweest te zijn, aangezien we voor deze library al snel een tweede toepassing hadden gevonden. Deze tweede toepassing ging over het ophalen van gegevens uit Sonar en deze te plaatsen in een Microsoft SQL Server[22] database, zodat een ander team deze gegevens zou kunnen gebruiken in PowerBI[23].

In totaal heb ik veel tijd gestoken in het bouwen van deze library en ook in het unit-testen van deze library aangezien ik hiervoor veel mocking heb moeten doen. Het effectieve resultaat is uiteindelijk wel een library geworden die op verschillende plaatsen inzetbaar is met weinig moeite in tegenstelling tot handmatig HTTP requests doen.

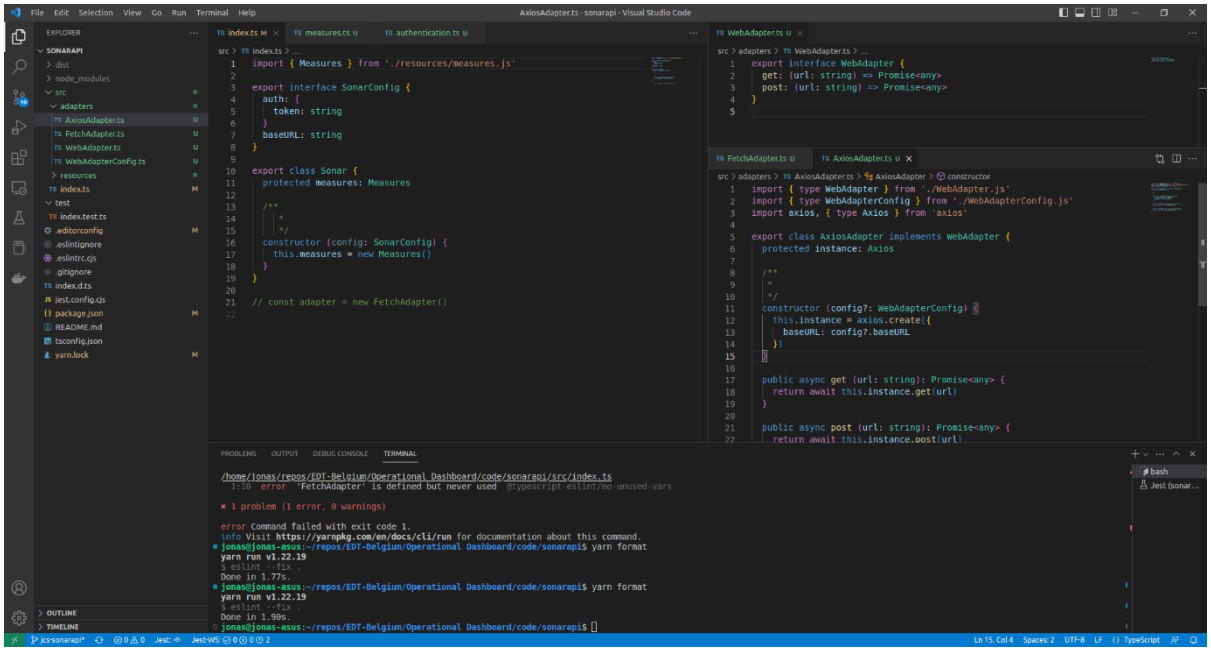
---

[18] https://swagger.io/specification/
[19] https://swagger.io/specification/v2/
[20] https://axios-http.com/docs/intro
[21] https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
[22] https://www.microsoft.com/en-us/sql-server
[23] https://powerbi.microsoft.com/nl-nl/

Hieronder kan je twee afbeeldingen zien van het geïmplementeerde Adapter-pattern voor de HTTP-adapters.



Figuur 19 Adapter pattern - Fetch adapter



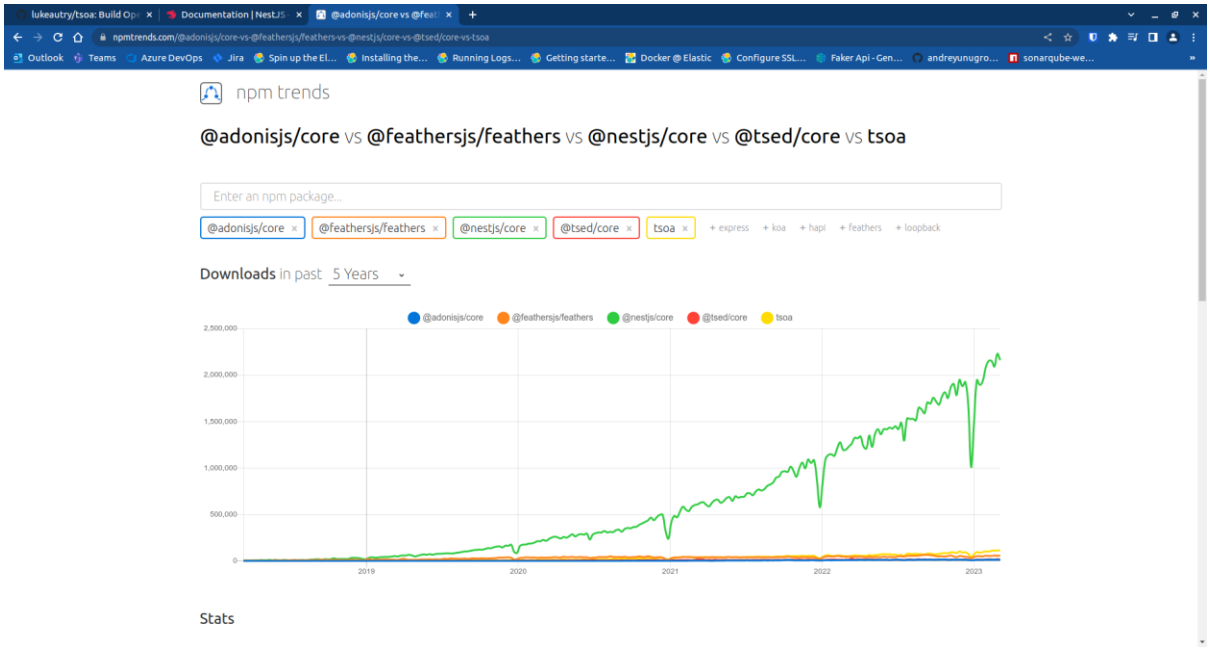
Figuur 20 Adapter pattern - Axios adapter

4.3.4 Integratie van gegevens in Elasticsearch

Voor de integratie van de gegevens in Elasticsearch heb ik een applicatie gebouwd die de data verwerkt en indexeert in Elasticsearch.

Om te bepalen welk framework het best was voor onze use-case heb ik de features van een aantal populaire web frameworks naast elkaar gelegd om dan uiteindelijk te kunnen beslissen welke ideaal is voor onze use-case.

Hieronder kan je een grafiek zien van een aantal populaire web frameworks.



Figuur 21 Vergelijking populaire web frameworks

In dit onderdeel heb ik uiteindelijk gebruik gemaakt van tsoa, een API-first web framework gebaseerd op TypeScript. Dit gaf mij de mogelijkheid om een API te bouwen waarmee we handmatig data kunnen gaan indexeren alsook om dit automatisch op bepaalde tijdstippen te doen.

Verder heeft tsoa geweldige documentatie en genereert het framework ook meteen een OpenAPI/Swagger specificatie. Dit maakt het heel makkelijk om deze web API te integreren met andere tools zoals pipelines.

De gegevens indexeren we in Elasticsearch nadat de index correct is gemaakt. Wanneer de index nog niet bestaat en we indexeren data, dan zal Elasticsearch aannames beginnen maken over de gegevens. Deze aannames zijn soms incorrect en resulteert dan in fouten tijdens het indexeren van de gegevens.

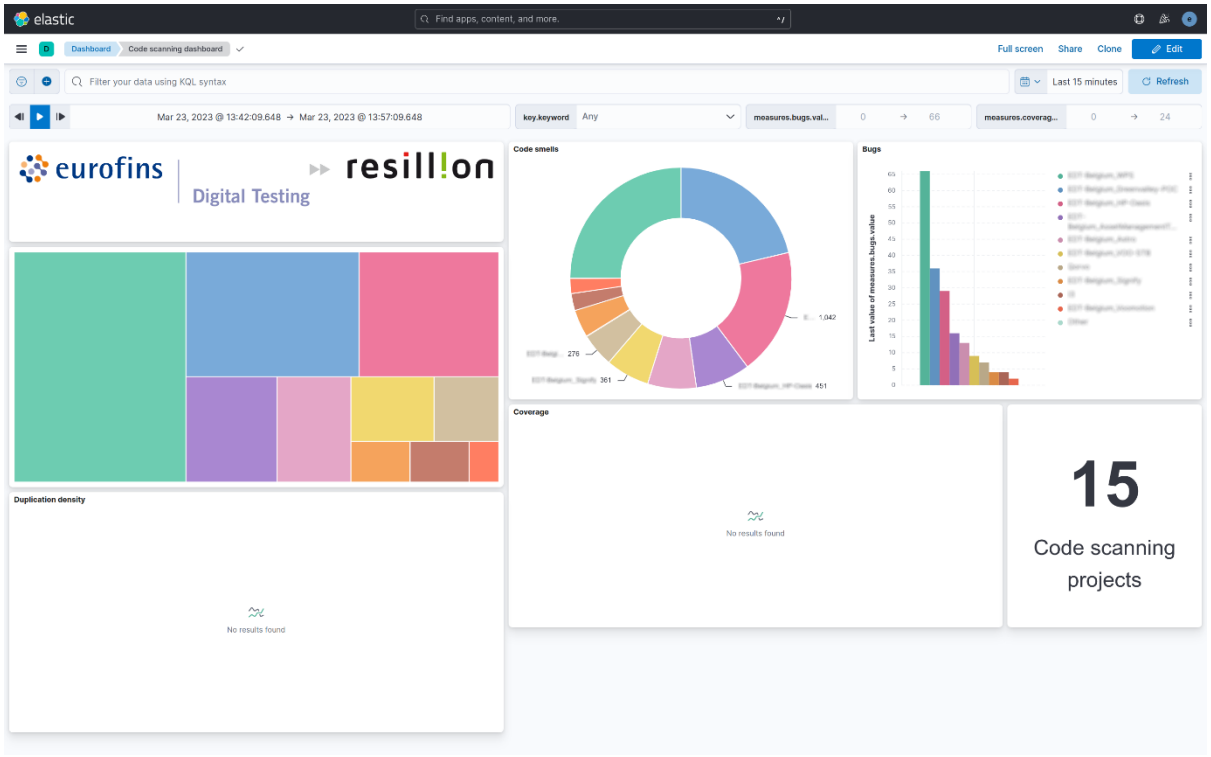
Om dit op te lossen spreken we eerst de Sonar API aan om de mogelijke data types (INT, FLOAT, PERCENTAGE, TEXT…) te achterhalen. Daarna halen we alle mogelijke metrics op en gaan we deze omvormen tot een index mapping met de data types erbij die Elasticsearch

kan inlezen. Nadat dit gebeurd is en de index aangemaakt is, kunnen we de data beginnen te indexeren.

De tijdsbasis waarin de indexering gebeurt, is volledig configureerbaar in de vorm van een Unix cron job. De specifieke cron implementatie die gebruikt werd, is nauwkeurig tot de seconde.

Verder zijn alle instellingen configureerbaar gemaakt via een configuratie file in het TOML-formaat. TOML is een uiterst geschikt formaat voor het maken van configuratiebestanden aangezien het een heel flexibel formaat is met ondersteuning voor de meeste datatypes.

Nadat deze integratie gebouwd was heb ik deze getest en heb ik hier ook al een voorbeeld dashboard mee gebouwd. Deze kan je hieronder terugvinden. De details van klanten en namen heb ik hier uit gelaten in verband met een NDA die ondertekend is.



Figuur 22 Sonar integratie - Voorbeeld dashboard

4.3.5 Azure DevOps integratie

Binnen Eurofins worden vooral de repos, pipelines en artifacts functionaliteiten gebruikt van Azure DevOps. Testen worden automatisch gerund via een pipeline en worden automatisch gestart op basis van een tijdschema. Een aantal pipelines starten automatisch wanneer nieuwe code gepushed wordt. Deze pipelines zijn vaak gericht op het builden van de software en de code kwaliteit checken via Sonar.

Alle pipelines die testen uitvoeren, worden op on-premise infrastructuur gedraaid, op die manier blijft de testdata intern en kan er ook getest worden op IoT hardware. Deze pipelines worden gerund via een zogenaamde pipeline agent.

Een pipeline agent, ook wel bekend als een build agent of een deployment agent, is een softwarecomponent binnen een Continuous Integration (CI) en Continuous Deployment (CD) systeem zoals Azure Pipelines. De agent is verantwoordelijk voor het daadwerkelijk uitvoeren van de taken die zijn gedefinieerd in een pipeline, zoals het bouwen van code, het uitvoeren van tests en het implementeren van software.

Pipeline agents kunnen zelf gehost (self-hosted) of gehost door de service (cloud-hosted) zijn. Bij zelf gehoste agents installeer en beheer je de agent op je eigen infrastructuur, wat je meer controle en flexibiliteit biedt. Cloud-hosted agents worden daarentegen door het CI/CD-systeem zelf beheerd en vereisen geen extra inspanning voor installatie en onderhoud.

Deze agents kunnen soms crashen of offline gaan door verschillende redenen. Als dit ’s nachts gebeurt wanneer er testen moeten lopen, dan kan dit leiden tot gemiste test runs en uiteindelijk data.

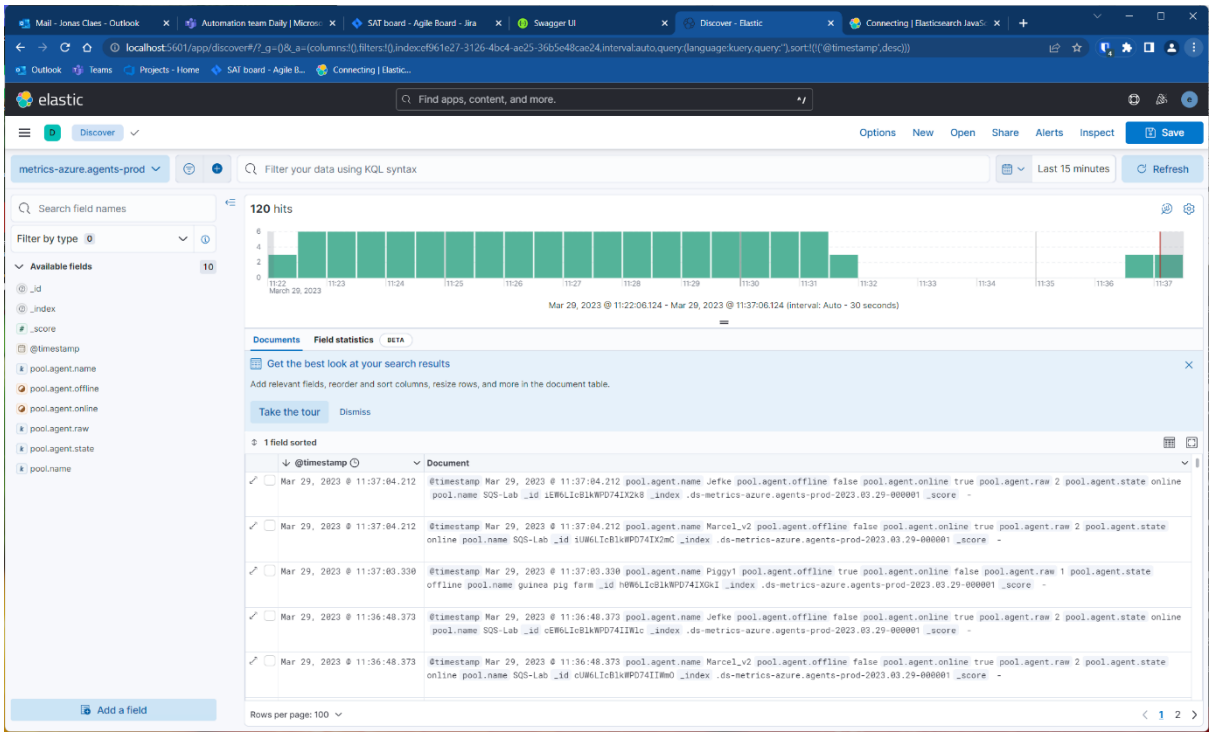
Vanuit de Azure DevOps API kunnen we data ophalen over de status van agents en pipelines. Deze gegevens willen we ook kunnen indexeren in Elasticsearch en mocht een agent down gaan, hier ook een alarm over genereren.

Dit doen we aan de hand van de Azure DevOps API en door een software te bouwen die de integratie van deze gegevens faciliteert. Deze software werkt uiteindelijk dus als ETL-pipeline en is gebaseerd op dezelfde core concepts als de Sonar integratie.

Omdat dit op dezelfde basis is gebaseerd hebben we ook opnieuw een volledig gedocumenteerde API met mogelijkheden om Elasticsearch te configureren voor de Azure API gegevens. Daarnaast biedt de API ook mogelijkheden om handmatig gegevens op te halen uit de API.

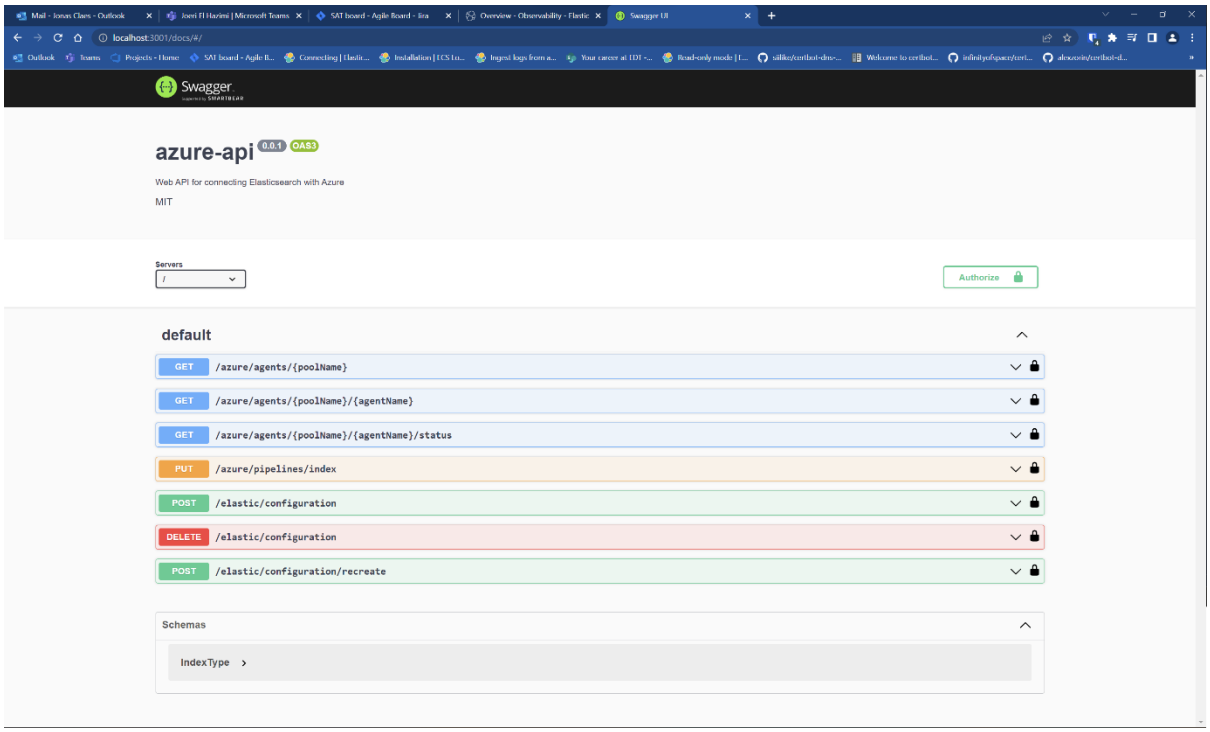
Nadat deze integratie gebouwd was heb ik deze getest, maar nog geen voorbeeld dashboard van gemaakt omdat ik deze ervaring al twee keer heb opgedaan met de vorige integraties.

Hieronder kan je een voorbeeld vinden van de data die geïndexeerd wordt in Elasticsearch voor de agents statussen.



Figuur 23 Azure integratie - Discover - Agents

Hieronder kan je een voorbeeld vinden van de documentatie die gegenereerd wordt door tsoa.



Figuur 24 Azure integratie - OpenAPI documentatie

4.3.6 Python integratie

Pytest[24] is een populair testframework voor Python dat wordt gebruikt om unit tests en functionele tests voor Python-applicaties te schrijven, uit te voeren en te organiseren. Het biedt een eenvoudige en flexibele manier om tests te schrijven met behulp van Python's ingebouwde assert statement en ondersteunt zowel Python 2 als Python 3.

Er is een testing framework opgezet binnen Eurofins dat Python testen kan uitvoeren met behulp van Pytest. Deze testen worden automatisch gestart vanuit de Azure Pipelines zoals hierboven beschreven.

Uit Pytest komt veel data, waarvan een deel geüpload wordt naar Report Portal, een open-source tool om test resultaten te delen met bijvoorbeeld klanten. Om de data in Elasticsearch te krijgen, konden we de data van Report Portal halen of rechtstreeks met Pytest integreren.

Uiteindelijk heb ik een integratie met Pytest gebouwd die zo de gegevens kan verwerken en indexeren in Elasticsearch. Deze maakt gebruik van hooks in Pytest om zo de gegevens te ontvangen en te verwerken. Deze gegevens worden dan genormaliseerd in het ECS-formaat[25] en weggeschreven naar een configureerbare opslaglocatie. Hierna verwerkt Filebeat de file en wordt deze geïndexeerd in Elasticsearch.

Nadat ik deze integratie gebouwd had, heb ik samen met een collega van Eurofins, Muhammed, deze plugin geïnstalleerd in een bestaande testing pipeline. Deze loopt sindsdien iedere dag mee en genereert data voor het dashboard.

[24] https://docs.pytest.org/en
[25] https://www.elastic.co/guide/en/ecs/current/index.html

4.3.7 Java integratie

JUnit[26] is een populair testframework voor Java dat wordt gebruikt om unit tests te schrijven, uit te voeren en te organiseren voor Java-applicaties. Het is een open-source project dat een belangrijk onderdeel vormt van de Java-ontwikkelingsomgeving en helpt ontwikkelaars bij het schrijven van betrouwbare en robuuste code.

SureFire[27] is een Maven plugin die wordt gebruikt om testen uit te voeren en testrapporten te genereren in Java-projecten. In combinatie met JUnit wordt de SureFire plugin gebruikt om JUnit-tests automatisch uit te voeren als onderdeel van het Maven build-proces en om testresultaten te rapporteren in een gestandaardiseerd formaat, zoals XML.

De XML die gegenereerd wordt door SureFire kunnen we inlezen aangezien dit een standaardformaat is. Dit maakt het koppelen van JUnit aan Elasticsearch al relatief simpel.

Om de effectieve integratie te maken, heb ik gebruik gemaakt van Filebeat, die de logs inleest, verwerkt en indexeert in Elasticsearch.

Hier had ik in het begin redelijk veel problemen mee omdat Filebeat normaal gezien bestanden lijn-per-lijn verwerkt, in plaats van een volledige blok tekst. Hiervoor heb ik gebruik moeten maken van de multiline module. Deze module kan een aantal lijnen groeperen op basis van patronen. Dit werkte heel goed voor onze use-case, mits er een newline karakter aan het einde van de file staat.

Om dit karakter toe te voegen aan de file heb ik een klein bash-scriptje geschreven dat wijzigingen monitort aan de map waar de rapporten in terecht komen. Als de map inhoud verandert, neemt het bash-scriptje de file, voegt hier een newline karakter aan toe, en zet deze in de map van Filebeat om geïndexeerd te worden.

[26] https://junit.org/junit5/
[27] https://maven.apache.org/surefire/maven-surefire-plugin/

4.4 Fase 3 – Alerting

In deze fase kan je meer informatie terugvinden over hoe we met Microsoft Teams verbinden en hoe de alarmen vanuit Kibana in Microsoft Teams terecht geraken.

4.4.1 Microsoft Teams integratie

Binnen Eurofins wordt Microsoft Teams veel gebruikt voor communicatie tussen collega's en zijn er verschillende teams en kanalen waarin informatie gedeeld kan worden en meldingen geplaatst kunnen worden.

Omdat alle collega's toegang hebben tot Microsoft Teams is het bij uitstek een goede tool om ook meldingen over infrastructuur in te plaatsen.

De Elastic Stack heeft intern ook ondersteuning om alarmen en meldingen in Microsoft Teams te plaatsen. Dit is een betalende feature waarvoor een licentie gekocht zou moeten worden.

Omdat we hierin geïnteresseerd waren, heb ik prijsinformatie opgevraagd bij Elastic. Een week later heb ik antwoord gekregen en heb ik de offerte gedeeld met Martijn. De prijs die op dit gamma aan features geplakt werd ging boven de 20.000 euro per jaar. Dit is natuurlijk overdreven veel voor enkel de alerting en meldingen die we willen gebruiken. Om deze reden hebben we beslist om zelf een integratie te bouwen met Microsoft Teams en de Elastic Stack.

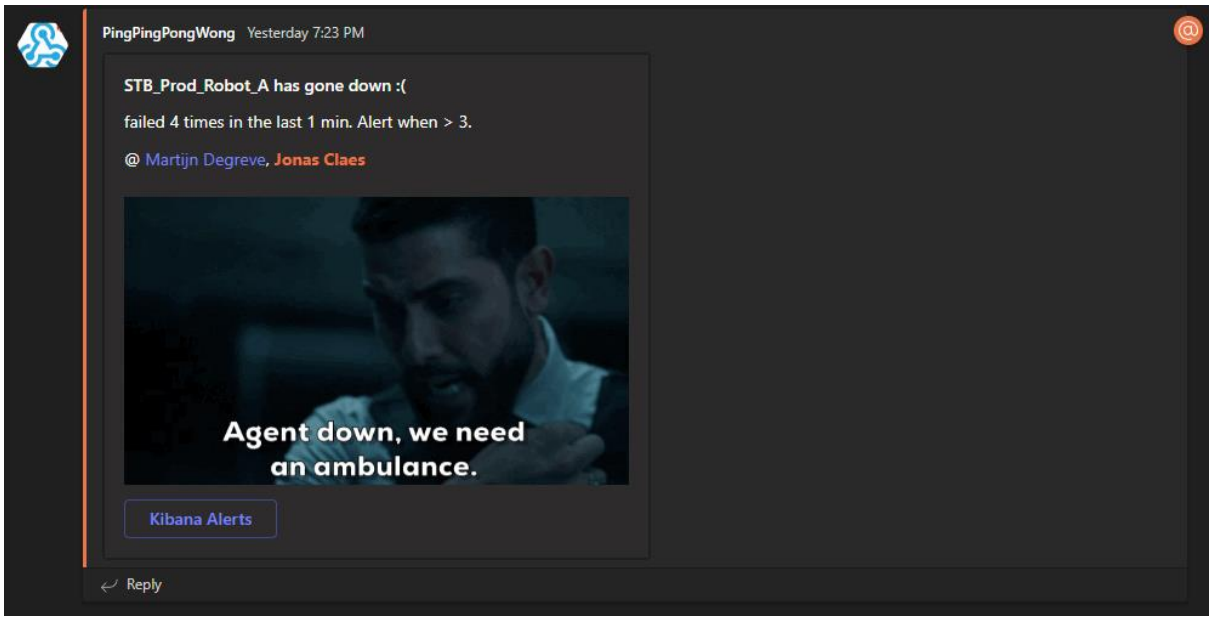
Voor deze integratie te bouwen, heb ik opnieuw de Sonar integratie als basis genomen. Hier heb ik het web API-gedeelte uit gelaten omdat dit niet nodig was voor deze integratie. Wat er uiteindelijk overbleef, was een TypeScript project met linting, formatting en testing.

Ik heb dan gebruik gemaakt van cron jobs om op een tijdschema de Elasticsearch API aan te spreken. Tijdens iedere run ga ik alle documenten in een bepaalde index ophalen, deze worden dan apart verwerkt en verstuurd over Microsoft Teams. Wanneer het bericht succesvol verstuurd is, wordt het document verwijderd uit de Elasticsearch index.

Het versturen naar Microsoft Teams verloopt via een webhook die aangemaakt kan worden op een kanaal-per-kanaal basis. Naar deze webhook kan dan data gestuurd worden via een zogenaamde POST-request. De gegevens die hiermee meegegeven dienen te worden zijn gespecifieerd als een standaardformaat, gebouwd door Microsoft. Dit standaard formaat heet Adaptive Cards en laat ons toe om berichten te sturen met een opmaak.

Op die manier kunnen we makkelijk alerts en meldingen sturen naar Microsoft Teams. Deze zijn ook volledig customizable en kunnen per type melding aangepast worden.

Hieronder kan u een voorbeeld zien van een bericht dat verstuurd werd toen een Azure DevOps Pipeline Agent down ging.



Figuur 25 Teams bericht wanneer een Azure DevOps Pipeline Agent down ging

Ieder bericht dat verstuurd wordt heeft een passende GIF[28] voor de gebeurtenis die plaats vindt, bijvoorbeeld, wanneer een agent down gaat zal er een GIF gebruikt worden met betrekking tot iets dat down gaat. Wanneer een agent dan weer herstelt na down geweest te zijn zal hier een positieve GIF aan toegevoegd worden die weergeeft dat er iets terug werkt.

Op die manier hebben we een heel erg snelle manier om te zien of er iets mis gaat of dat er iets herstelt zonder dat je de volledige tekst gelezen moet hebben.

Daarnaast zijn al deze GIFs volledig configureerbaar via een configuratiebestand dat later gegenereerd wordt door onze infrastructure-as-code toepassing.

4.4.2 Kibana

De alerting mogelijkheden in Kibana laten ons toe om bij te houden of de Azure DevOps Pipeline Agents nog werken, en als deze stoppen met werken, voor hoelang deze niet meer werken. Daarnaast laat dit ons ook toe om alarmen te genereren naar externe platformen toe.

De alarmen die Kibana genereert worden geplaatst in een Elasticsearch index, waarna de Microsoft Teams integratie die ik gebouwd heb, deze kan uitlezen en versturen naar Microsoft Teams.

In het Kibana Uptime Monitors dashboard kan dan makkelijk gekozen worden om voor een bepaalde agent alarmen in- of uit te schakelen.

[28] https://en.wikipedia.org/wiki/GIF

4.5 Fase 4 – Deployment

In deze fase kan je meer informatie terugvinden over het installeren en configureren van de volledige softwareoplossing. Daarnaast kom je ook meer te weten over de infrastructure-as-code met Ansible[29].

4.5.1 Server

In dit project heb ik een server opgezet met Ubuntu Server 22.04 LTS[30] als besturingssysteem. Ubuntu is een populaire en gebruiksvriendelijke Linux-distributie ontwikkeld door Canonical. Ik koos voor de LTS-versie (Long Term Support) omdat deze gedurende vijf jaar wordt voorzien van regelmatige updates en beveiligingspatches. Hierdoor ben ik verzekerd van een betrouwbare en veilige basis voor mijn serverinfrastructuur.

Om veilige toegang op afstand tot de server te bieden, heb ik een SSH-server[31] (Secure Shell) geïnstalleerd. SSH is een protocol dat versleutelde verbindingen tussen computers tot stand brengt, waardoor ik op een veilige manier op afstand beheertaken kan uitvoeren, zoals het installeren van software, het beheren van services en het controleren van systeemlogboeken.

Om mijn server te beschermen tegen kwaadwillende pogingen om toegang te krijgen, heb ik Fail2ban[32] geïmplementeerd. Fail2ban is open-source software die logbestanden monitort en IP-adressen blokkeert die herhaaldelijk inlogpogingen met onjuiste inloggegevens uitvoeren. Dit vermindert de kans op ongeautoriseerde toegang tot mijn server aanzienlijk. Daarnaast heb ik UFW[33] (Uncomplicated Firewall) geconfigureerd, een eenvoudig te gebruiken firewall voor Linux-systemen. UFW helpt bij het beheren van inkomende en uitgaande netwerkverbindingen en zorgt ervoor dat alleen geautoriseerde toegang wordt toegestaan en ongewenst netwerkverkeer wordt geblokkeerd.

Om de beveiliging van de toegang tot mijn server verder te versterken, heb ik twee-factor authenticatie (2FA) over SSH geïmplementeerd. Twee-factor authenticatie is een beveiligingsmethode waarbij een extra verificatiestap wordt toegevoegd aan het inlogproces. Gebruikers moeten, naast hun wachtwoord of SSH-sleutel, ook een tijdgebonden eenmalig wachtwoord (OTP) invoeren. Dit wachtwoord wordt gegenereerd door een authenticatie-app of hardware-token en biedt een extra beveiligingslaag tegen ongeautoriseerde toegang.

[29] https://www.ansible.com/
[30] https://ubuntu.com/
[31] https://nl.wikipedia.org/wiki/OpenSSH
[32] https://www.fail2ban.org/
[33] https://help.ubuntu.com/community/UFW

4.5.2 Infrastructure-as-code

Voor de configuratie en het beheer van de serverinfrastructuur heb ik Ansible gebruikt, een open-source automatiserings- en configuratiebeheertool. Ansible stelt mij in staat om eenvoudig en consistent serverconfiguraties en taken te beschrijven met behulp van zogenaamde playbooks, die geschreven zijn in de gemakkelijk te lezen YAML-taal[34]. Door gebruik te maken van Infrastructure-as-code (IaC), kan ik de infrastructuur definiëren en beheren via code, wat zorgt voor consistentie, eenvoudig onderhoud en betere samenwerking tussen teamleden.

Figuur 26 Ansible

In dit project heb ik ook Docker en Docker Compose geïntroduceerd voor het uitrollen van applicaties in containers. Containers zijn lichtgewicht, draagbare eenheden waarin applicaties en hun afhankelijkheden kunnen worden verpakt. Dit zorgt voor betere schaalbaarheid en efficiëntere workflows, omdat containers gemakkelijk kunnen worden verplaatst, opgeschaald en geïsoleerd van andere processen op de server. Docker Compose is een hulpmiddel om meerdere containers samen te definiëren en te beheren, waardoor ik complexe applicaties met meerdere onderdelen eenvoudiger kan uitrollen en beheren.

Om de configuratie van de applicaties verder te automatiseren en te vereenvoudigen, heb ik gebruikgemaakt van Jinja-templates[35]. Jinja is een template-engine voor Python die mij in staat stelt om dynamisch configuratiebestanden te genereren op basis van vooraf gedefinieerde sjablonen. Dit bespaart tijd en vermindert de kans op fouten, aangezien ik slechts één sjabloon hoef bij te werken in plaats van meerdere afzonderlijke configuratiebestanden.

Tot slot heb ik het gehele implementatieproces geautomatiseerd met behulp van Azure DevOps Release pipelines en Ansible. Release pipelines stellen mij in staat om automatisch wijzigingen in de code te testen, valideren en implementeren op de serverinfrastructuur. Dit zorgt voor een gestroomlijnd proces waarbij ik sneller en betrouwbaarder updates kan uitrollen en mijn serverinfrastructuur up-to-date en veilig kan houden.

Door al deze technologieën en beveiligingsmaatregelen te combineren, heb ik een robuuste en veilige serverinfrastructuur gecreëerd die gemakkelijk te onderhouden en uit te breiden is. Hierdoor kan ik mij richten op het ontwikkelen en implementeren van hoogwaardige applicaties, terwijl ik de veiligheid en betrouwbaarheid van de onderliggende infrastructuur waarborg. Bovendien zijn de gekozen oplossingen en methoden toegankelijk en begrijpelijk, zelfs voor mensen met beperkte technische kennis, wat de samenwerking en communicatie binnen het team ten goede komt.

[34] https://en.wikipedia.org/wiki/YAML
[35] https://jinja.palletsprojects.com/en/3.1.x/

4.6 Fase 5 – Dashboards

4.6.1 Jira

Dit dashboard is ontworpen om een volledig overzicht te geven van de activiteiten van het Software Automation Team (SAT). Het is bedoeld om de workflow, de voortgang van taken en de kwaliteitscontrole te monitoren. Hierdoor kunnen projectmanagers en teamleden op de hoogte blijven van de voortgang van projecten en taken, en kunnen ze proactief mogelijke problemen aanpakken.

Binnen het Jira-gedeelte van het dashboard zijn er twee belangrijke secties: SAT en algemene ticket status.

De SAT-sectie bevat informatie over de verschillende projecten die door het Software Automation Team worden uitgevoerd, automatiseringstaken en de algemene teamproductiviteit.

De algemene ticket status-sectie biedt een overzicht van de huidige status van de tickets. Er is een diagram dat de verdeling toont van gesloten, open, en andere soorten tickets binnen het SQS team. Het laat ook tickets zien die momenteel in behandeling zijn door het SAT, met verdere details zoals de beschrijving van elk ticket, de toegewezen persoon, de bijbehorende Epic, en het tijdstip en de categorie van de laatste statuswijziging.

Verder zijn er aparte secties voor tickets die momenteel worden gevalideerd en een overzicht van alle tickets per Epic in een taartdiagram. Dit maakt het gemakkelijk om te zien welke Epics de meeste tickets hebben, en dus mogelijk meer aandacht nodig hebben.

Daarnaast is er een speciale tabel die tickets toont met een prioriteit hoger dan medium. Dit zorgt ervoor dat teamleden en projectmanagers belangrijke taken niet uit het oog verliezen, en maakt het mogelijk om prioriteit te geven aan werkzaamheden op basis van hun belangrijkheid.

Hieronder kan je een afbeelding terugvinden van het uitgewerkte dashboard.

Figuur 27 Uitgewerkt Jira dashboard

Dit dashboard bevat gevoelige informatie die niet zomaar gedeeld kan worden. Om deze reden zijn de gevoelige visualisaties onleesbaar gemaakt.

4.6.2 Sonar

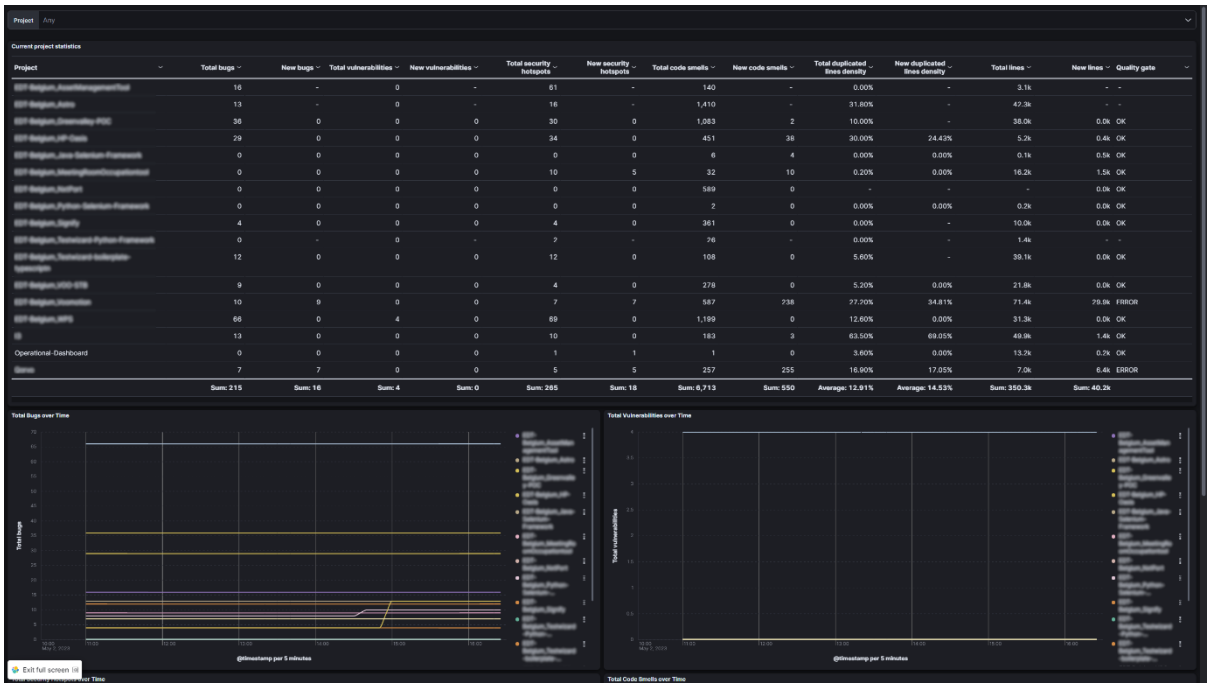
Dit Sonar-dashboard is ontworpen om een uitgebreid overzicht te geven van de kwaliteitsstatus van verschillende projecten. Het maakt gebruik van Sonar's functionaliteiten voor statische code-analyse om het team te helpen de codekwaliteit te beoordelen, bugs te identificeren, en voortdurende verbeteringen te waarborgen.

Het dashboard biedt twee verschillende weergaven: 'nu' en 'over tijd'. De 'nu'-weergave toont de huidige status van de projecten, terwijl de 'over tijd'-weergave de veranderingen in de kwaliteit van de projecten over een bepaalde periode laat zien. Dit helpt teams te begrijpen hoe hun inspanningen om de codekwaliteit te verbeteren de loop van de tijd hebben beïnvloed.

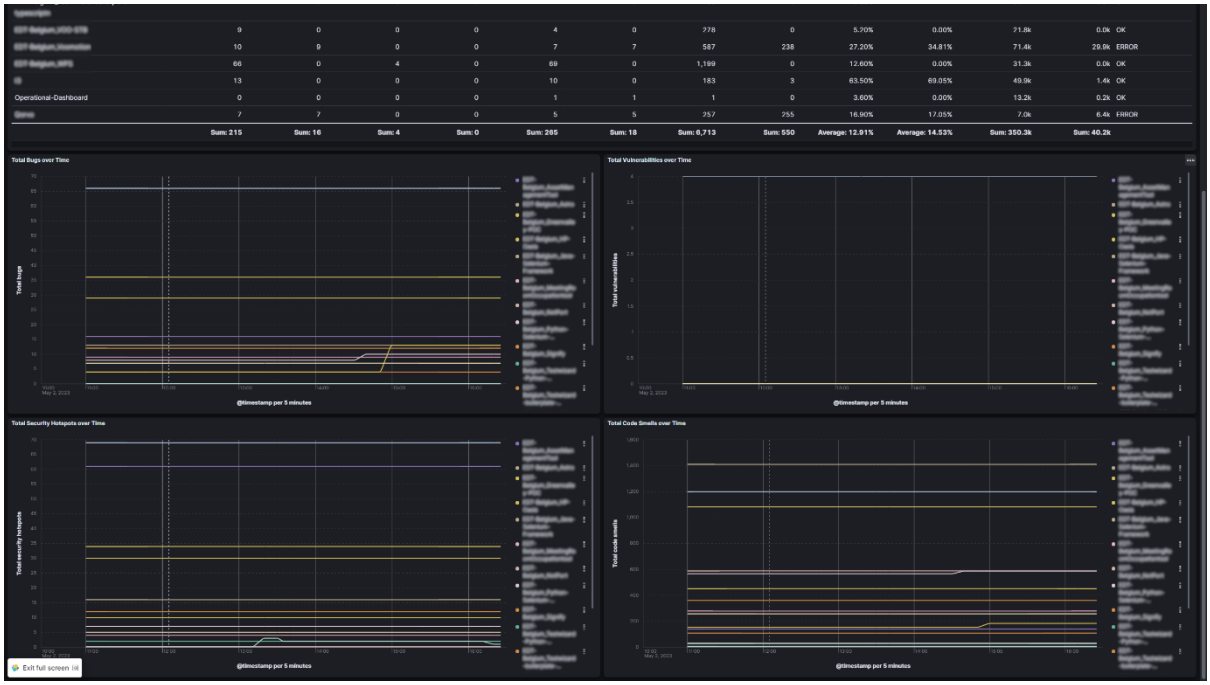
Daarnaast worden er per project statistieken weergegeven. Deze statistieken kunnen inzicht geven in verschillende aspecten van de codekwaliteit, zoals duplicatie, technical debt, en dekking van de code door tests. Dit geeft teams een gedetailleerd beeld van de specifieke sterke en zwakke punten van elk project.

Tot slot bevat het dashboard ook een onderdeel dat de bugs per project op tijdsbasis toont. Dit onderdeel is essentieel om te begrijpen waar en wanneer bugs ontstaan, en om de effectiviteit van bug fixing inspanningen te beoordelen.

Hieronder kan je afbeeldingen terugvinden van het uitgewerkte dashboard.



Figuur 28 Uitgewerkt Sonar dashboard met zicht op huidige status van projecten



Figuur 29 Uitgewerkt Sonar dashboard met zicht op project status over tijd

Dit dashboard bevat gevoelige informatie die niet zomaar gedeeld kan worden. Om deze reden zijn de gevoelige visualisaties onleesbaar gemaakt.

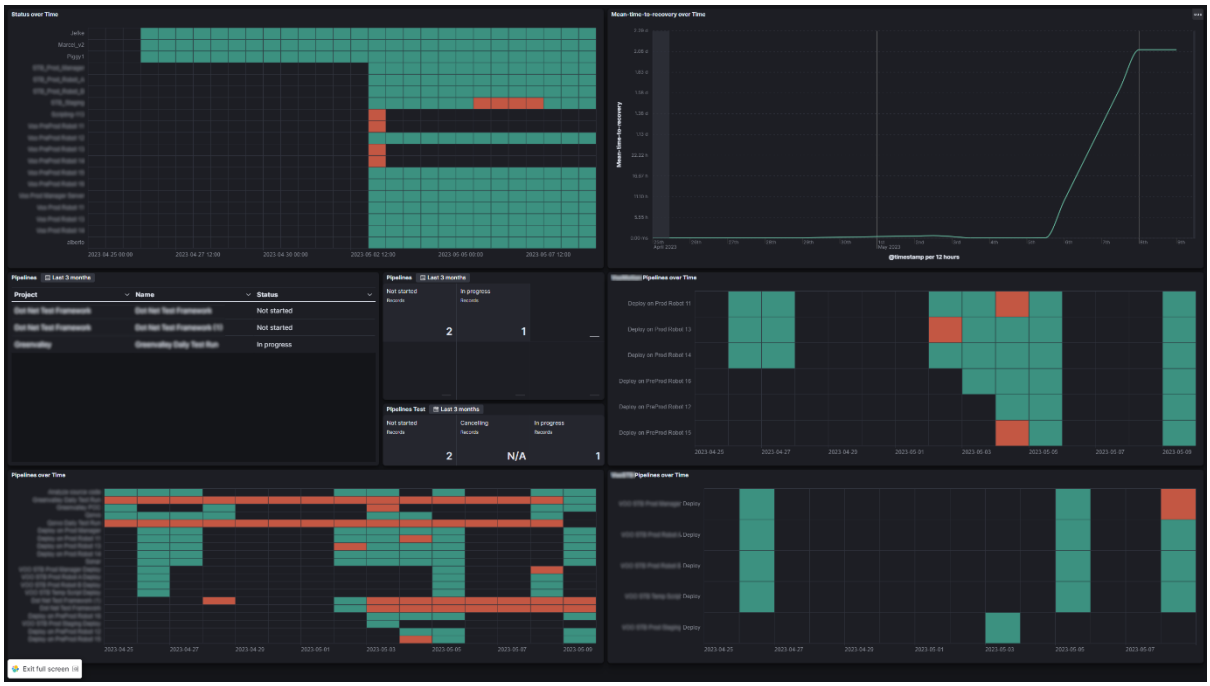
4.6.3 Azure DevOps

Het Azure DevOps-dashboard is ontworpen om een gedetailleerd overzicht te geven van de status de Azure DevOps-resources. Het geeft inzicht in de operationele gezondheid, uptime, en de voortgang van jobs om een effectief beheer en snelle probleemoplossing mogelijk te maken.

In de 'Status heatmap' sectie van het dashboard wordt de huidige gezondheidsstatus van de Azure DevOps-services weergegeven. Het gebruikt een eenvoudige kleurcodering - groen betekent dat een service goed werkt, terwijl rood wijst op een probleem. Dit zorgt voor een direct visueel overzicht van de status van de agents, waardoor snel en gemakkelijk eventuele problemen geïdentificeerd kunnen worden.

Onder 'Running jobs' vindt bevindt zich een tabel met details over de taken die momenteel worden uitgevoerd. Informatie zoals de projectnaam, de pipeline en de status (bijvoorbeeld 'in afwachting') van elke taak wordt vermeld. Dit geeft inzicht in welke taken actief zijn en wat hun huidige status is.

Hiernaast zijn er nog andere heatmaps die aantonen of pipelines geslaagd zijn of gefaald zijn in het verleden. Deze informatie is belangrijk gezien het feit dat wanneer een pipeline faalt, er mogelijks geen testen gerund hebben.



Figuur 30 Uitgewerkt Azure dashboard met zicht op historiek en huidige status van pipelines en agents

Dit dashboard bevat gevoelige informatie die niet zomaar gedeeld kan worden. Om deze reden zijn de gevoelige visualisaties onleesbaar gemaakt.

5 Extra - Philips Hue integratie

In de laatste fase van mijn stage heb ik een extra onderdeel aan mijn werk toegevoegd: het bouwen van een visuele indicator voor alarmmeldingen. Dit element stelt ons in staat om op een eenvoudige, intuïtieve manier te zien of er al dan niet alarmen actief zijn in ons systeem.

Om deze functionaliteit te realiseren, heb ik een integratie ontwikkeld met behulp van de programmeertaal Python, waarbij ik de libhueble bibliotheek gebruikte. Deze specifieke bibliotheek biedt de mogelijkheid om Philips Hue slimme lampen aan te sturen via Bluetooth Low Energy[36] (BLE).

Met deze integratie kunnen we de kleur van de slimme lamp aanpassen op basis van de alarm statussen in Kibana, ons monitoringsplatform. Wanneer een alarm actief is, zal de lamp rood oplichten. Zodra alle alarmen zijn opgelost, verandert de kleur van de lamp naar groen.

Ik heb echter ook rekening gehouden met gebruikers die gedeeltelijk kleurenblind zijn. Na feedback van mijn collega's heb ik de integratie aangepast en verbeterd om aan deze behoefte te voldoen. Een belangrijke wijziging was het instellen van de lamp om rood te knipperen in plaats van een continue rode kleur, bij een actief alarm. Deze aanpassing zorgt ervoor dat ook collega's met kleurenblindheid de alarmstatus gemakkelijk kunnen waarnemen.

Na het voltooien van de ontwikkeling en het testen van deze integratie, heb ik het uiteindelijk als een systemd[37] unit op onze Linux-server geïmplementeerd. Het gebruik van systemd stelt ons in staat om de service eenvoudig te beheren.

[36] https://en.wikipedia.org/wiki/Bluetooth_Low_Energy
[37] https://nl.wikipedia.org/wiki/Systemd

6 Conclusie

In deze bachelor scriptie werd de vraag gesteld hoe een geïntegreerd Operational Dashboard kan worden ontwikkeld en geïmplementeerd voor Eurofins Digital Testing, om hun operationele gegevens efficiënt te analyseren en monitoren en zo bruikbare inzichten te bieden voor het verbeteren van hun bedrijfsprestaties. Dit doel werd voortgestuwd door de noodzaak van een effectief dashboard dat de organisatie in staat stelt data-gedreven beslissingen te nemen.

Het onderzoek dat is uitgevoerd, heeft overtuigend aangetoond dat het project haalbaar is. Er waren echter technische uitdagingen te overwinnen. Bijvoorbeeld, het creëren van een naadloze koppeling tussen een dataopslag zoals Elasticsearch en API's zoals die van SonarCloud of Azure DevOps, bleek een uitdaging te zijn. Maar met volharding en technisch vernuft werden deze obstakels succesvol genavigeerd, waardoor het eenvoudiger werd om gegevens te visualiseren vanuit diverse bronnen.

De diepgaande verkenning heeft mij in staat gesteld belangrijke ontdekkingen te doen. Het meest opvallende onder deze bevindingen is de immense kracht en capaciteit van de Elastic Stack. Dit, gecombineerd met de efficiëntie van de API's bij het ingesteren van data in Elasticsearch, opent nieuwe mogelijkheden voor geavanceerde data-analyse en -monitoring.

De implicaties van deze ontdekkingen zijn veelzeggend. Ze tonen aan dat de Elastic Stack een cruciale rol kan gaan spelen in de bedrijfsvoering van Eurofins. De brede interesse in deze technologie, zoals getoond door verschillende teams, afdelingen en managers, voorspelt een toenemend gebruik van de Elastic Stack binnen de organisatie in de nabije toekomst.

Desondanks was dit onderzoek niet zonder beperkingen. De scope bleef beperkt tot de Elastic Stack, waardoor een vergelijking met andere potentieel nuttige tools, zoals PowerBI, ontbrak. Toekomstig onderzoek kan dit opvullen door het uitvoeren van een grondige vergelijking van de functionaliteiten van de Elastic Stack en andere data-analyse en -visualisatie tools. Dit zou Eurofins een completer beeld kunnen geven van de beste methoden voor datavisualisatie en -analyse, passend bij hun specifieke behoeften.

Concluderend heeft dit onderzoek een waardevolle bijdrage geleverd over de ontwikkeling en implementatie van een operationeel dashboard binnen Eurofins. Het heeft een solide basis gelegd voor toekomstig onderzoek en voor de mogelijke uitbreiding van het gebruik van de Elastic Stack binnen de organisatie.

Het succes van dit project, gecombineerd met de geuite interesse binnen het bedrijf, suggereert dat de Elastic Stack een cruciale rol kan gaan spelen in het verbeteren van de bedrijfsprestaties van Eurofins. Deze verbetering zal worden bereikt door efficiënte analyse en monitoring van operationele gegevens, waardoor het bedrijf de vruchten kan plukken van data gedreven besluitvorming.

7 Aanbevelingen

In deze sectie kan je meer te weten komen over aanbevelingen die ik naar Eurofins Digital Testing gedaan heb. Deze aanbevelingen gaan vooral over security, beheer en schaalbaarheid.

7.1 Let’s Encrypt / ACME SSL certificaten

Een Let's Encrypt[38] SSL certificaat is een gratis, automatisch vernieuwbaar certificaat dat wordt verstrekt door Let's Encrypt, een non-profit certificeringsinstantie (CA). SSL staat voor Secure Sockets Layer en is een standaard beveiligingstechnologie die wordt gebruikt om een versleutelde verbinding tot stand te brengen tussen een webserver en een browser. Dit zorgt ervoor dat alle gegevens die tussen de server en de browser worden uitgewisseld, privé en beschermd blijven tegen onderschepping door derden.

Let's Encrypt is opgericht om het gebruik van SSL/TLS (Transport Layer Security, de opvolger van SSL) op het internet te bevorderen en het eenvoudiger en toegankelijker te maken om een veilige verbinding te bieden. Door een Let's Encrypt SSL certificaat te gebruiken, kan een website-eigenaar de vertrouwelijkheid van gegevensuitwisseling waarborgen en bezoekers verzekeren dat hun verbinding met de website veilig is. Bovendien zullen moderne browsers de website als 'veilig' markeren, wat het vertrouwen van gebruikers in de website versterkt.

Tijdens het project hebben we de afweging gemaakt tussen self-signed certificaten en Let’s Encrypt certificaten. Uiteindelijk is de keuze gemaakt voor self-signed certificaten omdat als we dit via IT zouden moeten regelen, dit niet meer haalbaar zou zijn voor deze stage.

Om deze reden is dit een aanbeveling die ik toch wil meenemen omdat een geldig SSL-certificaat de privacy en beveiliging van de gegevens beter kan garanderen dan een self-signed certificaat.

[38] https://letsencrypt.org/

7.2 Elastic Cloud

Tijdens mijn stageopdracht heb ik de Elastic Stack geïmplementeerd als een self-hosted oplossing. Hoewel dit een waardevolle leerervaring was en ons inzicht gaf in de werking van Elastic, zijn er enkele belangrijke redenen waarom we zouden moeten overwegen om over te stappen naar Elastic Cloud[39] in plaats van onze huidige aanpak. Hier zijn enkele redenen die de voordelen van Elastic Cloud benadrukken ten opzichte van onze self-hosted oplossing:

- Tijdsbesparing: Door over te stappen op Elastic Cloud, hoef ik niet langer tijd te besteden aan het beheren en onderhouden van onze Elastic Stack. Het Elastic-team neemt deze verantwoordelijkheden op zich, waardoor ik me kan concentreren op andere belangrijke taken binnen de organisatie, zoals het ontwikkelen van nieuwe functies en het verbeteren van bestaande processen.
- Schaalbaarheid: Met onze huidige self-hosted oplossing kan het uitdagend zijn om de Elastic Stack te schalen wanneer dat nodig is. Elastic Cloud maakt het eenvoudig om resources, zoals opslag, geheugen en rekenkracht, op- of af te schalen om aan onze behoeften te voldoen. Hierdoor kunnen we efficiënter omgaan met veranderende workloads en tegelijkertijd kosten optimaliseren.
- Betrouwbaarheid en beschikbaarheid: Elastic Cloud biedt ingebouwde redundantie en hoge beschikbaarheid, wat betekent dat het risico op downtime aanzienlijk wordt verminderd in vergelijking met onze self-hosted oplossing. Bovendien worden back-ups automatisch gemaakt en opgeslagen in de cloud, waardoor we onze gegevens snel kunnen herstellen in geval van een storing.
- Geoptimaliseerde prestaties: Elastic Cloud wordt voortdurend geoptimaliseerd en bijgewerkt door het Elastic-team, wat resulteert in betere prestaties en stabiliteit in vergelijking met onze self-hosted oplossing. Dit betekent dat onze Elastic Stack sneller en betrouwbaarder zal werken, wat een positieve impact heeft op onze gebruikerservaring.
- Toegang tot de nieuwste functies en beveiligingsupdates: Met Elastic Cloud hebben we automatisch toegang tot de nieuwste functies en beveiligingsupdates, zonder dat we handmatig upgrades hoeven uit te voeren. Dit zorgt ervoor dat onze Elastic Stack altijd up-to-date en veilig blijft, wat essentieel is voor het beschermen van onze gegevens en het voldoen aan de vereisten op het gebied van gegevensbescherming.

Gezien de bovenstaande voordelen raad ik aan om te overwegen om over te stappen van onze huidige self-hosted Elastic Stack naar Elastic Cloud. Dit zal niet alleen onze efficiëntie en schaalbaarheid verbeteren, maar ook de betrouwbaarheid en beveiliging van onze gegevens waarborgen.

Om deze initiële stap makkelijker te maken heb ik ook een prijsvoorbeeld gemaakt voor Eurofins om snel te weten wat het zou kosten om de Elastic Stack te deployen in de cloud.

Voor deze calculatie heb ik gebruik gemaakt van verschillende opslag mogelijkheden met Azure. Op die manier zouden we data ouder als 1 jaar kunnen verhuizen van hot-storage naar warm-storage. Na 3 jaar zou deze data dan kunnen verhuizen naar cold-storage. Op die manier behalen we de hoogste kosten efficiëntie.

[39] https://www.elastic.co/cloud/

Nawoord

Het is met een gevoel van voldoening en dankbaarheid dat ik deze bachelor scriptie afrond, die het resultaat is van maandenlang onderzoek, leren en groeien. Het proces van deze stage heeft mij onmisbare kennis en inzichten opgeleverd, zowel op technisch als persoonlijk vlak, en het was een bijzonder lonende ervaring.

Allereerst zou ik mijn stagebegeleider, Caroline Vanderheyden, bij Thomas More willen bedanken voor haar ondersteuning en begeleiding gedurende dit hele proces. Haar adviezen en feedback waren onmisbaar en hebben mij enorm geholpen in het uitvoeren van deze stage.

Ik wil ook graag mijn dank uitspreken aan Martijn Degrève en Tiziana Treccase voor hun waardevolle technische en sociale begeleiding. Zonder hun deskundigheid en ondersteuning had ik niet het niveau van inzicht en begrip kunnen bereiken dat ik nu heb.

Bovendien gaat mijn oprechte waardering uit naar mijn collega's en vrienden bij Eurofins / Resillion. Hun vriendelijkheid, professionaliteit en collegialiteit hebben enorm bijgedragen aan mijn positieve ervaring en hebben het mogelijk gemaakt dat ik me echt deel van het team voelde.

Ik ben bijzonder verheugd om te melden dat ik, na een succesvolle stageperiode, een aanbod van Resillion heb ontvangen en aanvaard. Ik kijk ernaar uit om mijn professionele reis met hen voort te zetten en ik ben enthousiast over de uitdagingen en kansen die deze nieuwe rol zal bieden.

Tot slot wil ik mijn ervaring en aanbevelingen delen met toekomstige stagiaires en medestudenten. Deze stage was een ongelooflijk leerzame en verrijkende ervaring, en ik raad Resillion van harte aan voor iedereen die op zoek is naar een stageplaats waar je zowel professioneel als persoonlijk kunt groeien.

Het afronden van deze bachelor scriptie markeert niet alleen het einde van een belangrijke fase in mijn academische carrière, maar is ook het begin van mijn professionele reis in de wereld van de technologie.

Ik ben dankbaar voor de kennis en ervaring die ik heb opgedaan, en ik kijk ernaar uit om deze te gebruiken als basis voor toekomstige successen.