Sprint logboek

Steven Koerts

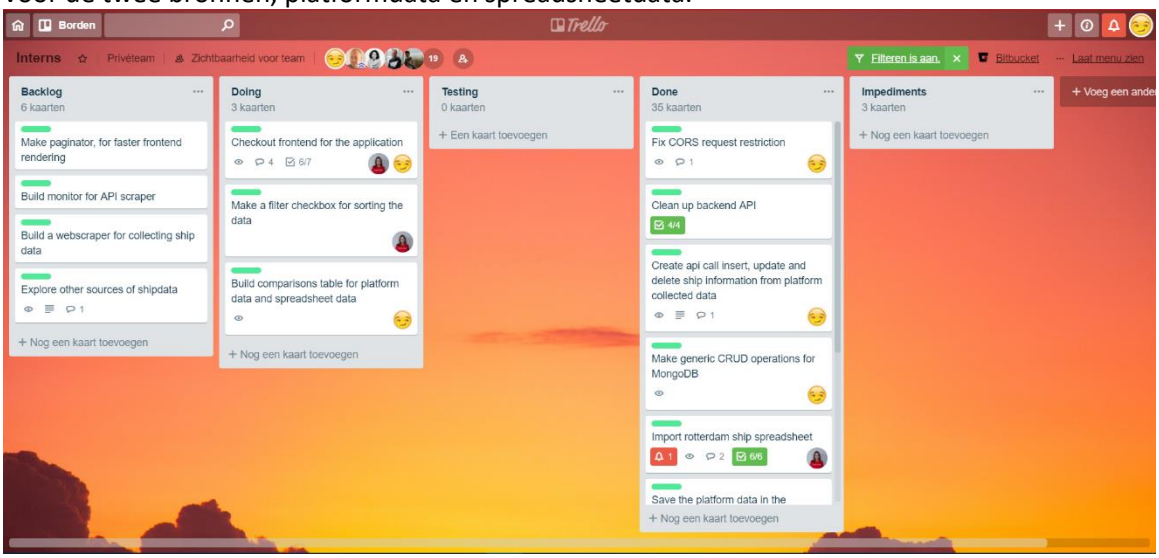
0904861

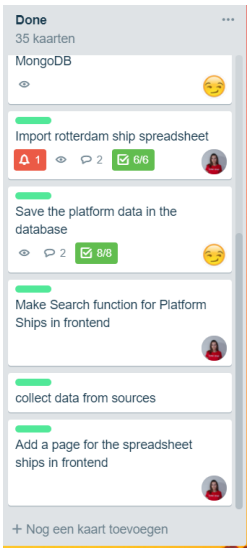
Sprint 1

In de eerste sprint stond vooral het opstarten van de projecten centraal, als eerste hebben we hier het Teqplay platform gedownload en lokaal opgezet. De eerste sprint was vooral een onderzoeksfase naar wat Teqplay al had. Zo hebben we wat geëxperimenteerd met de Teqplay platform API, om scheepsinformatie op te vragen. Na een week hadden we al een backend framework opgezet, we hadden gekozen voor Springboot met de programmeertaal Kotlin. Springboot is een Java webframework die onder andere gebruikt maakt van het (Model View Controller)MVC pattern. Het eerste dat we in Springboot hadden opgezet was een API consumer die een call maakt naar de Teqplay platform API om scheepsinformatie op te halen.

Sprint 2

De tweede sprint heb ik een frontend framework opgezet, React.js in ons geval. Omdat React de bedrijfsstandaard is als het gaat om frontend. Om snel een mooie layout te creëren voor de frontend heb ik Bootstrap gebruikt een mobile friendly CSS framework. Zo heb je snel een mooi opgemaakte website zonder zelf CSS te schrijven. Verder ben ik verder gegaan met het opslaan van de platform data in een Mongo database, zo hoeven we niet elke keer de data uit het platform te halen en kan dat dus via onze eigen database. We maken gebruikt van een frontend en backend die volledig los van elkaar staan, zo hou je clientside en serverside processen uit elkaar. Verder zijn we bezig geweest met het importeren van andere databronnen, zo kregen we van het bedrijf een spreadsheet van statische scheepsdata van de Port of Rotterdam. Dan kunnen we in de volgende sprints de data van verschillende bronnen combineren. Zo had ik in deze sprint ook nog een vergelijkingstabel gemaakt voor de twee bronnen, platformdata en spreadsheetdata.



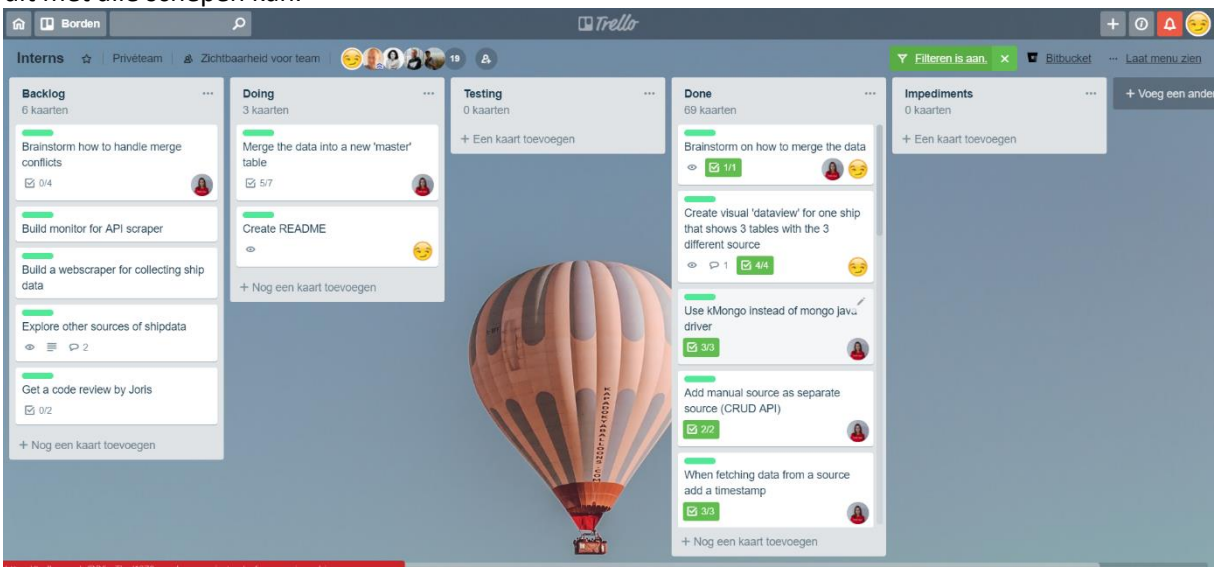


Sprint 3

Wat deze sprint vooral centraal stond was het nadenken over het mergen van de scheepsdata, uit de verschillende bronnen. Op dit moment hadden we twee bronnen verzameld 1 bron is het Teqplay platform en een andere is een spreadsheet met zeeschepen. De spreadsheet komt van Port of Rotterdam. Deze sprint hebben we ook nog een derde bron toegevoegd aan onze applicatie, een handmatige bron. Het idee hierachter is dat we dan ook zelf informatie over schepen kunnen toevoegen en genereren. Het mergen van de data is een van uitdagendste features van de applicaties, ook omdat dit het hart van het systeem vormt. Het uiteindelijke doel van het systeem is dus informatie over schepen binnenhalen en samenvoegen tot een databron.

Verder had ik in deze sprint nog een vergelijkingstool gemaakt om individuele schepen naast elkaar te zetten op het scherm. Dit is een interne tool die overzicht genereerd over de beschikbare informatie en zo het samenvoegen makkelijker maakt.

Aan het eind van de sprint was de applicatie zover dat het schepen handmatig kon mergen, voor de volgende sprint gaat de voornaamste prioriteit naar het automatiseren van dit proces en zorgen dat dit met alle schepen kan.

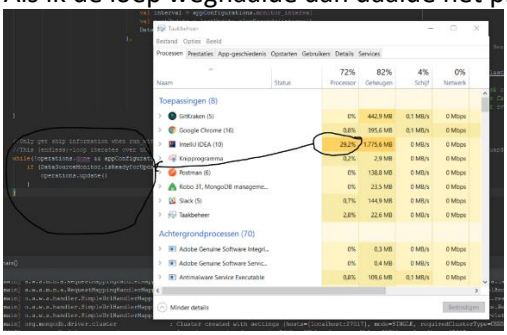


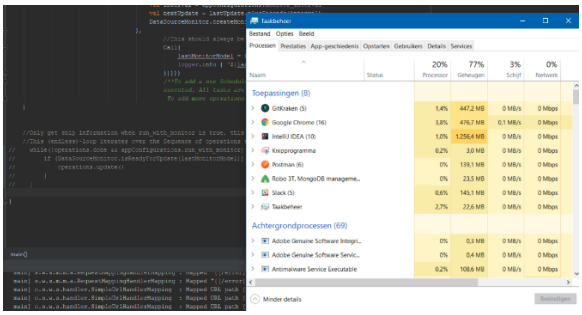
Sprint 4

Deze sprint is er voornamelijk in de backend gewerkt van ons project, vanaf hier heb ik meer duidelijkheid gekregen over het project en waar het project heen gaat binnen Teqplay. Ons systeem moet dus uiteindelijk de hoofdbron worden van statische scheepsinformatie bij het bedrijf waar alle applicaties gegevens vandaan kunnen halen.

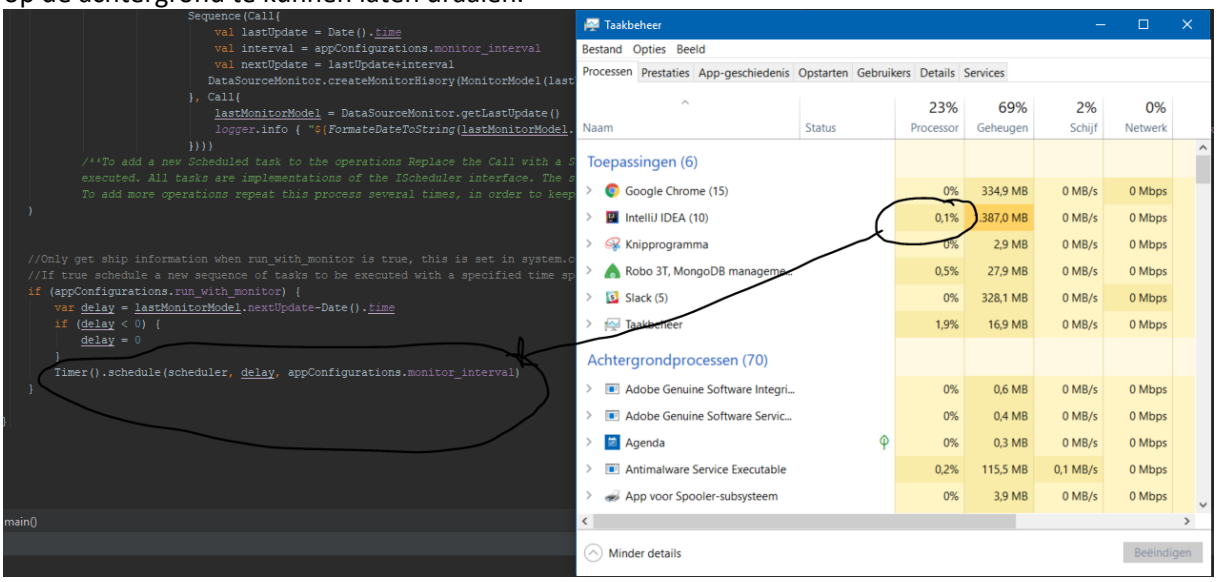
In de eerdere sprints hadden we al een applicatie die gegevens van verschillende bronnen binnenhaalden, alleen om dit te bereiken moest er handmatig een functie worden aangeroepen om gegevens op te halen of te updaten. Deze sprint heb ik daarvoor een monitor of task scheduler(taakplanner) gebouwd die om een van te voren bepaalde functies aanroept. Dus als nu de applicatie uitgevoerd wordt dan wordt de taakplanner op de achtergrond uitgevoerd en om de zoveel tijd worden dan nieuwe gegevens binnengehaald. Het proces van het samenvoegen van schepen is ook geautomatiseerd en dynamisch gemaakt. Dit is een mooi begin van het opbouwen van de zo gewilde master database.

Verder heb ik deze sprint ook gewerkt aan het documenteren en testen van stukken code. Een van testen die ik heb uitgevoerd is een performance test, hierbij kan ik zien hoeveel processor kracht gebruikt wordt door de applicatie. Wat bleek was dat de app 30% van de rekenkracht van de processor gebruikte wanneer er alleen achtergrond processen draaide. Dan heb ik het dus niet over het eenmalig opstarten, maar het constant stand-by runnen van het systeem. Dit bleek te komen door de taakmonitor die ik had gebouwd en het onnodig gebruik van een eindeloze loep.
Als ik de loep weghaalde dan daalde het processor gebruik naar 1%.





Uiteindelijk heb de task scheduler herbouwd zonder een eindeloze loep erin om hem stabiel rustig op de achtergrond te kunnen laten draaien.



Verder heb ik deze sprint ook besteed om de code op te ruimen, comments te plaatsten waar die ontbraken en het schrijven unit tests. Het doel van deze unit tests is voor het bouwen of overdragen van de applicatie. Als iemand anders in een bepaalde component of unit een wijziging maakt, zal de unit test vervolgens falen en hoogstwaarschijnlijk ook andere tests. Hier zie je dan precies wat er fout er fout gaat, want de test geeft aan wat het verwachte resultaat is en het werkelijke resultaat. Als die twee niet met elkaar overeenkomen dan faalt dus de test.

Deze sprint zijn er goede stappen gezet richting een eerste live deployement van het systeem, daar dragen de taakmonitor, het automatisch samenvoegen van scheepsdata en de unit test allemaal aan bij. Als afsluiter heb ik ook wat tijd vrijgemaakt om een mooi startup scherm te maken in de command line.