

Sprint logboek

Steven Koerts

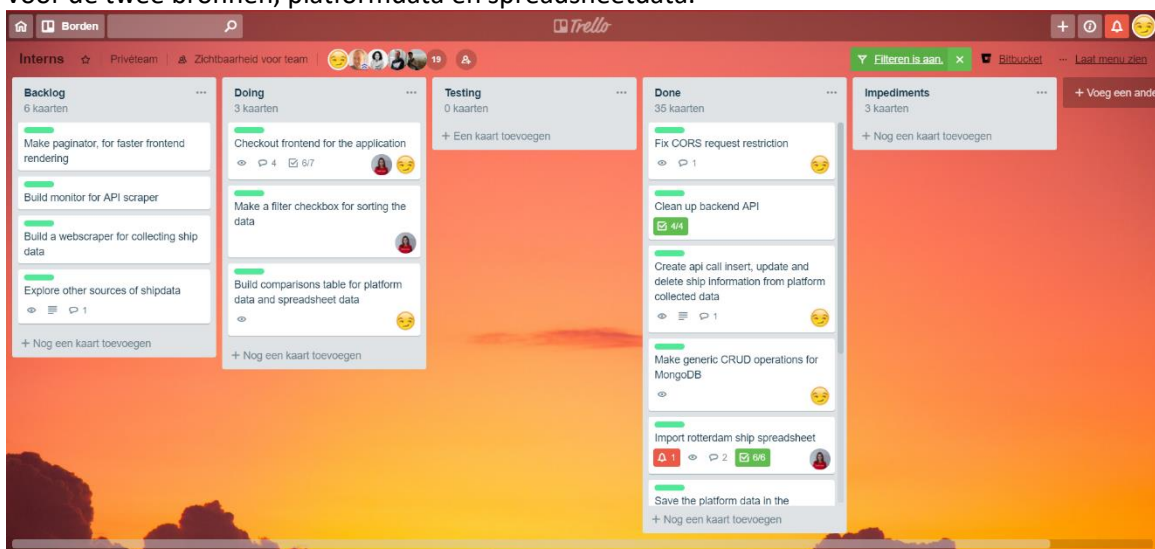
0904861

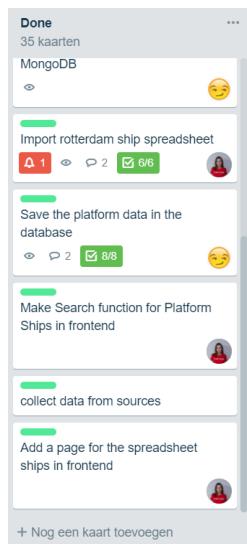
Sprint 1

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 gebruik 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.



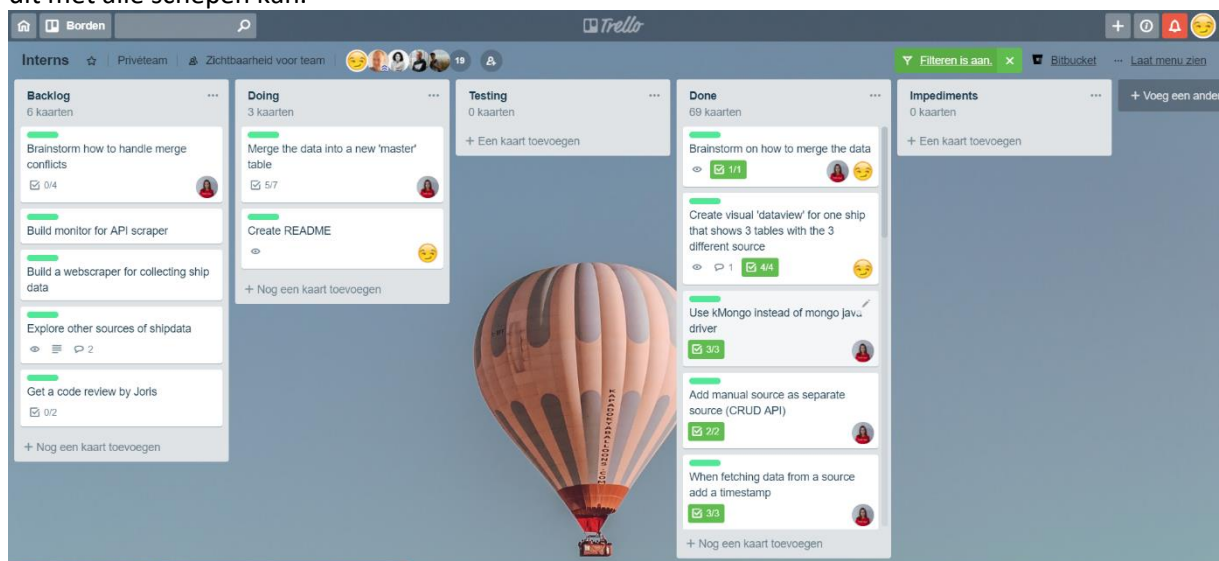


Sprint 3

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 Tegplay 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 genereert 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.

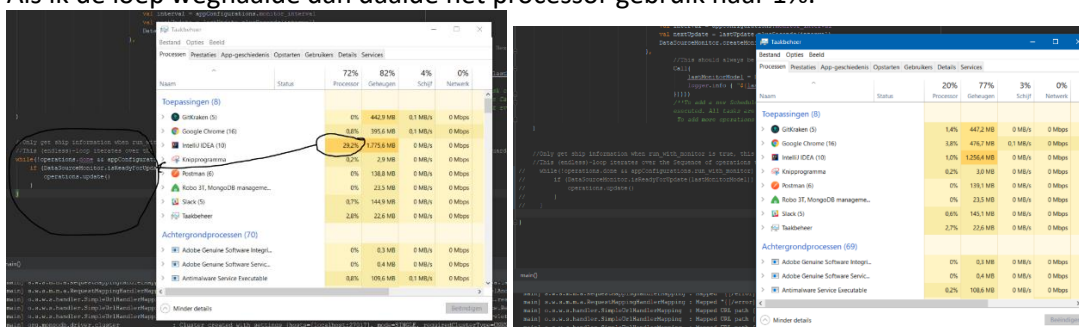


Sprint 4

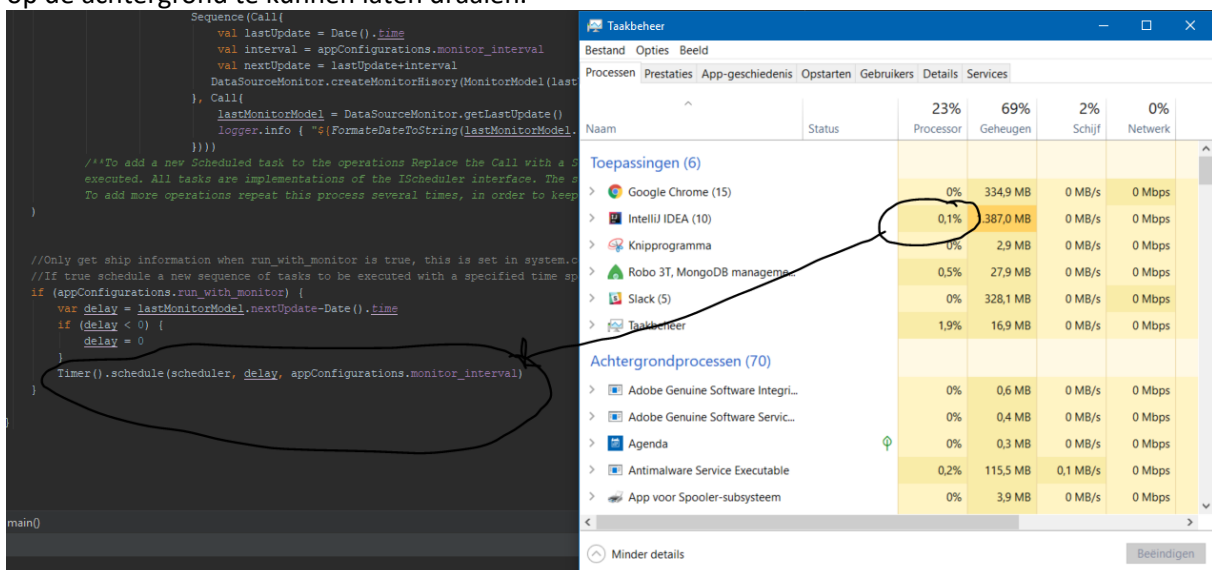
Deze sprint is er voornamelijk in de backend gewerkt van ons project, vanaf hier heb ik meer duidelijkheid gekregen over het project en waar het project heen gaat binnen Teqplay. Ons systeem moet dus uiteindelijk de hoofdbron worden van statische scheepsinformatie bij het bedrijf waar alle applicaties 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 loop. Als ik de loop weghaalde dan daalde het processor gebruik naar 1%.



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

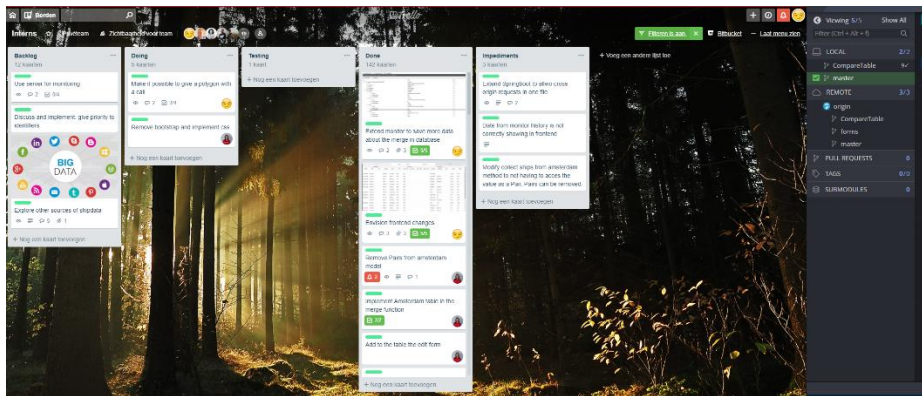


Verder heb ik deze sprint ook besteed om de code op te ruimen, comments te plaatsten waar die ontbraken en het schrijven unit tests. Het doel van deze unit tests is voor het bouwen of overdragen van de applicatie. Als iemand anders in een bepaalde component of unit een wijziging maakt, zal de unit test 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 deployment 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.

```
-----  
/  _  | |  ( )  | |  | |  
| ( _ | | _ _ _ | | | | _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _  
 \ _ \ | ' _ \ | | ' _ \ | _ / _ ' _ \ | ' _ \ / _ \ ' _ \  
 _ ) | | | | | | | | | | | | | | | | | | | | | | | | | | | |  
| _ _ / | | | | | _ / | | | | | | | | | | | | | | | | | | | | | |  
      | |      | |      | |  
      | |      | |      | |  
-----  
Application started v0.1 (beta)  
App configurations:  
Root: http://localhost:8080  
Monitor enabled: true  
Monitor interval: 10000 (milliseconds)  
  
Last database update:  
Last update: 10/19/2018 16:54:01  
Next update: 10/19/2018 17:54:01  
Current time: 10/22/2018 10:10:15  
Ready for next update: true
```

Sprint 5



Deze sprint begon met het toevoegen van een vierde databron, die we gebruiken in de applicatie. Dat is een XML bestand met 27.000 nieuwe schepen die het bedrijf heeft verkregen via het havenbedrijf van Amsterdam. Het leuke aan deze bron is dat die nieuwe info bevat over de schepen, zoals het motorvermogen en de kracht van de boeg- of hekschroef. Het uitdagende van deze bron was dat we weer opnieuw moesten beginnen met analyseren van hoe de bron er precies uit ziet, welke gegevens staan er in, wat slaan we op en wat mag weg? Het scheelde nu wel dat we dit al eerder hadden gedaan met dit project dus al wisten waar we moesten beginnen met zoeken. Zo kan je eerder opgedane ervaringen al snel in praktijk brengen. Zo begin je met het maken van een model voor de databron en vervolgens het opslaan in de database en als laatste een API schrijven die data naar buiten toe vrijgeeft. Daarna is ook deze databron opgenomen in het mergen van de schepen en in de masterdatabase toegevoegd.

Verder ben ik deze sprint ook verder gegaan in de front-end, het bedrijf gaf me een tip voor React-Table een framework waar je heel gemakkelijk data in een mooi opgemaakte tabel kunt laden en zo ook makkelijk schepen kunt vergelijken. Zo een interne vergelijkingstool helpt enorm bij het development proces.

Als laatste heb ik de monitor uitgebreid om ook informatie op te slaan over hoeveel schepen er per keer worden geüpdatet en worden toegevoegd. Dit wordt in de toekomst nog meer, zo zal ik volgende sprint werken aan het opslaan van schepen per gebied op de kaart. Zodat je dus kunt zien hoeveel schepen er in een bepaald gebied zijn.

Sprint 6


Deze sprint stond voornamelijk het focussen van het gebruik van het systeem op de planning, dus ervoor zorgen dat het systeem een duidelijk doel krijgt en alle functionaliteiten gebruikt kunnen worden. Deze sprint heb ik gewerkt aan het visueel maken de veranderingen die plaats vinden in de database. Hiervoor heb ik een design pattern gebruikt, de observer. De observer is een object dat andere objecten in de gaten houdt en als er een verandering van een bepaalde waarde plaatsvindt een notificatie stuurt. Die notificatie kan van alles zijn, een email, sms-bericht of iets opslaan in een database. Op dit moment worden alle veranderingen opgeslagen in een database, dus kan je zien welk attribuut is veranderd naar wat.

Veel voorkomende dingen in de database zijn dat de lengte en breedte opeens wisselen, dan heb je opeens een schip van 120 meter breed en 6 meter lang. Het is dan relevant om de schipper een bericht te sturen over de wijziging en dit terug te laten draaien. Aangezien de route informatie ook gebaseerd is op de breedte van een schip, zou dat schip nergens meer doorheen passen.

▼ (14) ObjectId("5be443e4de306f618e0e62ce")	{ 7 fields }
_id	ObjectId("5be443e4de306f618e0e62ce")
id	211564050
attribute	positionOfTransponder
oldValue	{ 4 fields }
distanceToBow	28.0
distanceToStern	3.0
distanceToPort	5.0
distanceToStarboard	1.0
newValue	{ 4 fields }
distanceToBow	28.0
distanceToStern	3.0
distanceToPort	1.0
distanceToStarboard	5.0
source	TEQPLAY_PLATFORM
date	2018-11-08 14:10:44.652Z

Handwritten red text: "Value Swap" with an arrow pointing from the oldValue to the newValue for distanceToPort.


Verder ben ik in de front-end bezig geweest met het maken van een uitgebreide detail pagina met alle informatie over een schip die we tot nu toe hebben. Ook een afbeelding van het schip erbij die rechtstreeks van marine traffic komt. Dit was nog een uitdaging aangezien je de afbeelding kan opzoeken met twee nummers een imo en mmsi nummer en je weet van tevoren niet welke beschikbaar is. Ook ben ik afgestapt van het CSS framework bootstrap en heb mijn eigen CSS geschreven. Volgende sprint ga ik hiermee verder en de lay-out verder uitwerken, uiteindelijk moet een catalogus worden voor schepen, dat als je iets zoekt over een schip je het dan zal vinden.

 ShipHappens

[Home](#) [Available sources](#) [Create manual ship](#)

CASTORO SEI - [BHS]

OFFSHORE PIPELAYER Pipe layer Platform, semi submersible



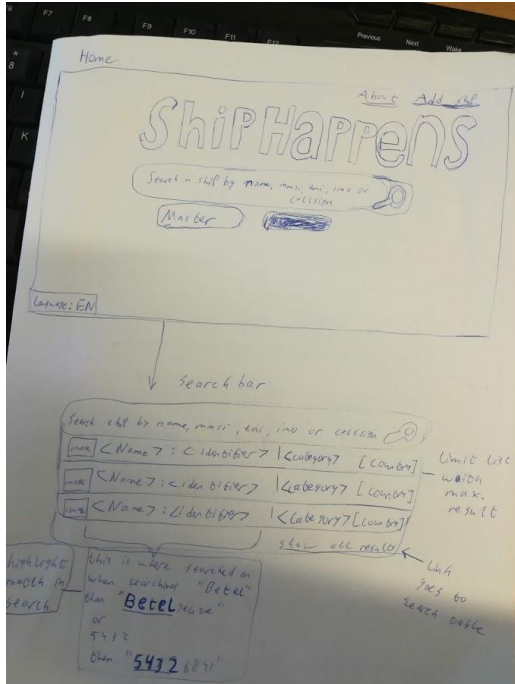
© Ria Maat
MarineTraffic.com

Static Info

MMSI: 308162000
IMO: 8758603
ENI:
CallSign: C6DF4
Flag Code: BHS
AIS Version:
Year of Construction: 1978

Sprint 7

Deze sprint heb ik de applicatie uitgewerkt vanuit een gebruikersperspectief, de sprint begon met het maken van een schets voor de front-end van de applicatie. Ons systeem heeft hetzelfde idee als Google, we verzamelen een hoop informatie. In ons geval over schepen. Die informatie geven we niet direct vrij aan de gebruiker, we laten de gebruiker zoeken naar een schip. Op voorwaarde dat de gebruiker zelf moeten weten waar die naar op zoek is, we geven dus niet meer de vrijheid om rond te zoeken in de database.



ShipHappens

Search ship by name, mmsi, eni or callsign

Na de schetsen ben ik het design gaan realiseren tot een website. Hierbij zat ook in het maken van een nieuwe detail pagina, waar alle schip eigenschappen in categorieën zijn ingedeeld. Ook kan je een schip nu aanpassen in de detail pagina, op dit moment wordt de gebruiker doorverwezen naar een nieuwe pagina. Op die pagina zijn alle stukken tekst vervangen door een input veld waar een schip wordt aangepast. Als een schip is aangepast wordt die opgeslagen in een aparte collectie en het schip staat gemarkeerd als handmatig. Voor volgende sprint wordt ook gewerkt aan een manier op schepen op dezelfde pagina door te wijzigen.

RAINBOW WARRIOR - [NLD]

YACHT SAIL/YACHT - Yacht (Sailing)

Identifiers:

MMSI: 244163000

IMO: 9575363

Callign: PF7187

Flag Code: NLD

Static Info:

Six Code: Z

Home Port: Amsterdam

Position sensor type: -

AIS Version: -

Year of Construction: 2011

Last Update: Sun Nov 25 20:36:12 CET 2018

Shiphappens source: MERGED

View how org got merged

Categories:

Category: YACHT

Sub-Category: SAIL/YACHT

Sub-Category Code: X11A2YS

HBH Type: Yacht (Sailing)

Role: -

Shiptype: -

Measurements:

Gross Tonnage(Ton): 855 Ton

Net Tonnage(Ton): 855 Ton

Height(m): -

Draught(m): -


Dead Weight(Ton): 855 Ton

Main Engine power(KW): 1425 KW KW

Thruater alert power(KW): 1 x 110 KW

Thruater low power(KW): -

Length X Width(m): 57.92 M X 11.3 M



RAINBOW WARRIOR NLD

YACHT SAIL/YACHT ShipType: YACHT Yacht (Sailing)

Identifiers:

MMSI: 244163000

IMO: 9575363

Callign: PF7187

Flag Code: NLD

Static Info:

Six Code: Z

Home Port: Amsterdam

Position sensor type: -

AIS Version: -

Year of Construction: 2011

Last Update: Sun Nov 25 20:36:12 CET 2018

Shiphappens source: MERGED

Categories:

Category: YACHT

Sub-Category: SAIL/YACHT

Sub-Category Code: X11A2YS

HBH Type: Yacht (Sailing)

Role: -

Shiptype: -

Measurements:

Gross Tonnage(Ton): 855


Net Tonnage(Ton): 257

Height(m): -

Draught(m): -

Dead Weight(Ton): 855


Main Engine power(KW): 1425



Als laatste hebben we een volledige live deployment gedaan op de student servers van Teqplay. De applicatie draait met een losse backend en een losse front-end, er zijn nu nog problemen met doorverwijzen naar verschillende pagina's. Dit komt doordat Amazon's S3 buckets niet goed samen gaan met React router. Dit is een issue waar komende sprint naar gekeken gaat worden. Voorlopig kan je de website bekijken op <http://demo.teqplay.nl/shiphappens/>.

Sprint 8

Deze sprint heb ik verder gewerkt aan het gebruik van applicatie, zo is het voor de applicatie dus mogelijk om schepen toe te kunnen voegen en wijzigen. Na het toevoegen van een schip wordt dit direct naar de master database gemerged zodat de veranderingen direct zichtbaar zijn. Voor het gebruikersgemak heb ik gewerkt aan een React component die kan wisselen tussen een gewone tekst en een invoer veld. Zo kan de gebruiker dus alleen het veld aanpassen dat hij expliciet kiest om te veranderen en hoeft niet door te gaan naar een nieuwe pagina.

NERO  NGA 

Dit is in read only modus.

NERO	 	Nigeria ▾	 
------	---	-----------	---

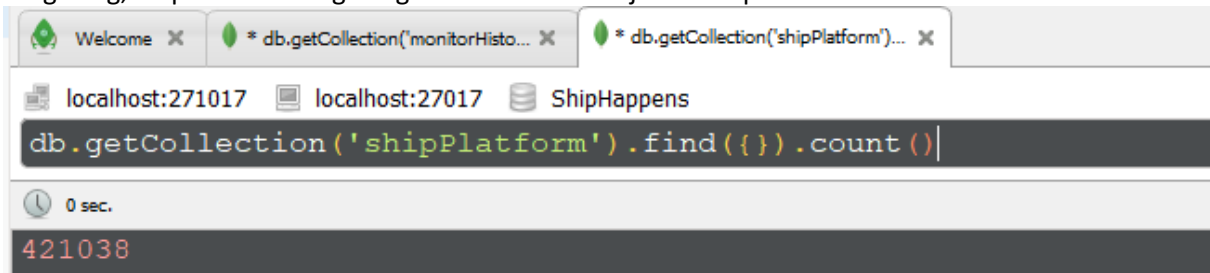
Door op de pen te klikken kom je bij de bewerk modus.

Verder nog wat veranderingen in de styling van de front-end op de detail pagina en een aantal kleine issues in code, bug fixus etc. Dit komt de performance van het systeem ten goede en zorgt voor het stabiliseren van het systeem. Dat is ook waar ik de komende laatste weken mee aan de slag ga, het verbeteren en optimaliseren van het systeem. Om het project goed af te ronden.

Sprint 9

Tot zover hebben we een hoop gedaan aan het project er is een hoop werk verzet en een hoop gedaan. Vanaf hier hebben we besloten dat we twee richtingen op kunnen gaan, één is om de breedte in te gaan en verder uit te gaan bouwen en twee is het huidige systeem wat we tot noch toe hebben te optimaliseren. We hebben de richting gekozen om het systeem verder te optimaliseren en het breder maken van het systeem op een lagere prioriteit te zetten.

Verder ben ik wel gaan kijken naar wat er gebeurt als je meer schepen probeert binnen te halen van Teqplay platform. Op dit moment halen we zo een 400.000 schepen binnen van de development omgeving, de productie omgeving heeft zo een 5 miljoen schepen.



The screenshot shows a MongoDB shell window with three tabs: 'Welcome', '* db.getCollection('monitorHisto...', and '* db.getCollection('shipPlatform')...'. The active tab shows a query: `db.getCollection('shipPlatform').find({}).count()`. Below the query, it indicates '0 sec.' and the result '421038'.

Dit is om de performance van het systeem te testen, verder met het oog op een live deployment is het belangrijk dat het systeem stabiel kan draaien op de studentserver. Op dit moment gebruikt het systeem te veel werkgeheugen en krijgt voortdurend een `OutOfMemoryException`, waardoor ons systeem er voortdurend uitligt. Dus hier gaan we komende weken mee aan de slag.

Verder heb ik in de front-end een aantal kleine dingen opgepakt om componenten er beter uit te laten zien. Eén van die dingen is het tonen van landvlag afbeeldingen bij alle schepen, hiervoor heb ik een API gevonden die deze functionaliteit ondersteunt. <https://restcountries.eu/> We hadden al beschikking tot de landcodes van de schepen. In de API kan je zoeken op landcodes en de API geeft vervolgens alle informatie over dat land terug. Voorlopig gebruik ik alleen de afbeelding die de API terug geeft, maar er zit ook informatie in over de volledige landnaam. Dit is interessant voor het project omdat gewoon de volledig naam van een land naast het schip staat in plaats van een niets zeggende afkorting.

Bijvoorbeeld: Landcodes ZWE en SWE, kunnen beide worden aangezien voor het land Zweden.

Afhankelijk van de taal die je gebruikt. In het Engels in SWE Zweden en ZWE is Zimbabwe. Zo zijn er een hoop verschillende gevallen, dus daarom is het handig om de volledige landnaam neer zetten op het scherm.

NERO 

Tunisia ▾



Sprint 10 – De Laatste en de afronding

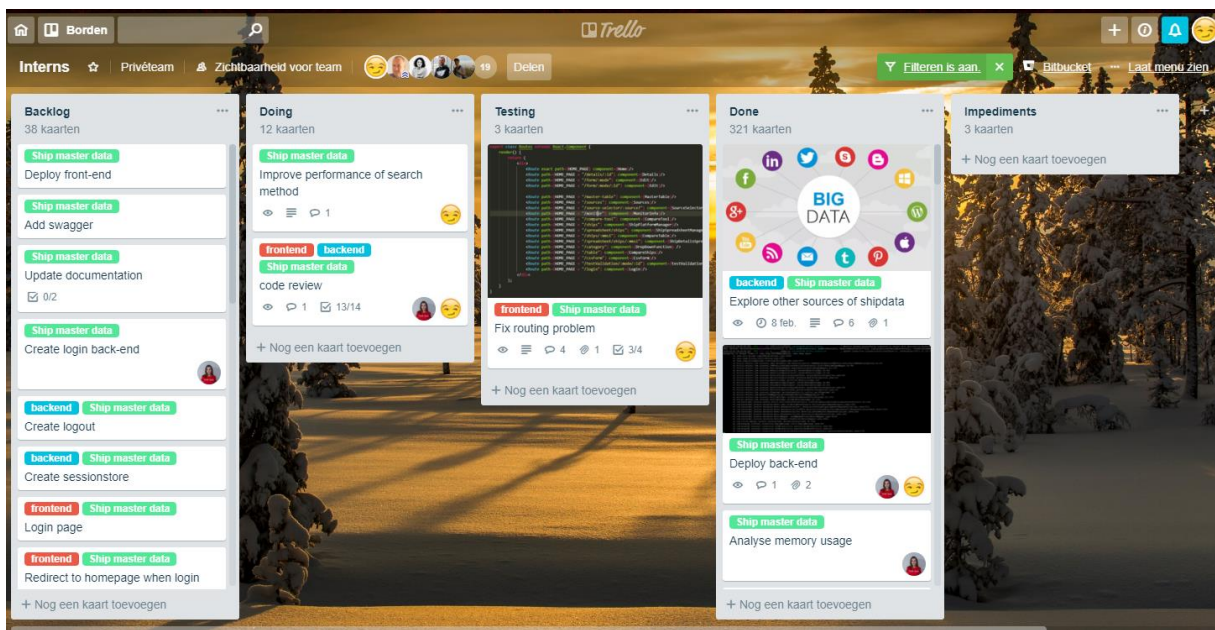
De laatste sprint staat vooral het optimaliseren, testen en verbeteren centraal. Zo heb ik een performancetest uitgevoerd om de snelheid van de zoekfunctie te testen. Deze was namelijk wat aan de trage kant en er moest worden uitgezocht hoe dat kwam. Er staan als laatste nog een aantal kaarten op het scrum bord, waar we waarschijnlijk niet meer aan toe komen. Onder anderen het maken van een login systeem en een categorie samen-voeger. In alle databronnen zitten een hoop verschillende scheepscategorieën, in totaal 7. Het mooiste zal zijn als deze samengevoegd kunnen worden tot één categorie.

Verder is er ook gewerkt aan deployen van de applicatie op de student server van het bedrijf, in het begin waren er constant `OutOfMemoryExceptions`. Dit werd veroorzaakt door dat alle schepen tegelijkertijd werden ingeladen en al het werkgeheugen van de server in beslag nam.

```
2018-12-21 14:38:38.381 INFO 9059 --- [127.0.0.1:27017] org.mongodb.driver.cluster : Monitor thread successfully connected to server with description ServerDe
TBD, ok=true, version=ServerVersion{versionList=[2, 6, 12]], minWireVersion=0, maxWireVersion=2, maxDocumentSize=16777216, logicalSessionTimeoutMinutes=null, roundTripTimeNa
2018-12-21 14:38:38.387 INFO 9059 --- [Timer-0] org.mongodb.driver.connection : Opened connection [connectionId(localValue:17, serverValue:147)] to 127.0
Exception in thread "Timer-0" java.lang.OutOfMemoryError: Java heap space
    at java.util.Arrays.copyOfRange(Arrays.java:3664)
    at java.lang.String.<init>(String.java:207)
    at java.lang.StringBuilder.toString(StringBuilder.java:407)
    at kotlin.reflect.jvm.internal.impl.metadata.JvmMemberSignatureMethod.asString(JvmMemberSignature.kt:20)
    at kotlin.reflect.jvm.internal.JvmFunctionSignature$KotlinConstructor.<init>(RuntimeTypeMapper.kt:66)
    at kotlin.reflect.jvm.internal.RuntimeTypeMapper.mapSignature(RuntimeTypeMapper.kt:175)
    at kotlin.reflect.jvm.internal.KFunctionImpl$caller$2.invoke(KFunctionImpl.kt:60)
    at kotlin.reflect.jvm.internal.KFunctionImpl$caller$2.invoke(KFunctionImpl.kt:54)
    at kotlin.reflect.jvm.internal.ReflectProperties$LazySoftVal.invoke(ReflectProperties.java:92)
    at kotlin.reflect.jvm.internal.ReflectProperties$Val.getValue(ReflectProperties.java:31)
    at kotlin.reflect.jvm.internal.KFunctionImpl$caller$2.invoke(KFunctionImpl.kt:60)
    at kotlin.reflect.jvm.internal.KParameterImpl$2$1.invoke(KParameterImpl.kt:44)
    at kotlin.reflect.jvm.internal.KParameterImpl$2$1.invoke(KParameterImpl.kt:25)
    at kotlin.reflect.jvm.internal.ReflectProperties$LazySoftVal.invoke(ReflectProperties.java:92)
    at kotlin.reflect.jvm.internal.ReflectProperties$Val.getValue(ReflectProperties.java:31)
    at kotlin.reflect.jvm.internal.KTypeImpl.getJavaType$Kotlin_reflect_api(KTypeImpl.kt)
    at kotlin.reflect.jvm.ReflectJvmMapping.getJavaType(ReflectJvmMapping.kt:70)
    at kotlin.reflect.jvm.internal.KCallableImpl.callDefaultMethod(KCallableImpl.kt:133)
    at kotlin.reflect.jvm.internal.KCallableImpl.callBy(KCallableImpl.kt:110)
    at com.fasterxml.jackson.module.kotlin.KotlinValueInstantiator.createFromObjectWith(KotlinValueInstantiator.kt:135)
    at com.fasterxml.jackson.databind.deser.impl.PropertyBasedCreator.build(PropertyBasedCreator.java:138)
    at com.fasterxml.jackson.databind.deser.BeanDeserializer.deserializeUsingPropertyBased(BeanDeserializer.java:471)
    at com.fasterxml.jackson.databind.deser.BeanDeserializerBase.deserializeFromObjectUsingNonDefault(BeanDeserializerBase.java:1191)
    at com.fasterxml.jackson.databind.deser.BeanDeserializer.deserializeFromObject(BeanDeserializer.java:314)
    at com.fasterxml.jackson.databind.deser.BeanDeserializer.deserialize(BeanDeserializer.java:140)
    at com.fasterxml.jackson.databind.ObjectMapper._readMapAndClose(ObjectMapper.java:3814)
    at com.fasterxml.jackson.databind.ObjectMapper.readValue(ObjectMapper.java:2945)
    at org.litote.kmongo.jackson.JacksonCodec.decode(JacksonCodec.kt:164)
    at com.mongodb.internal.connection.ReplyMessage.<init>(ReplyMessage.java:48)
    at com.mongodb.internal.connection.GetMoreProtocol.execute(GetMoreProtocol.java:95)
    at com.mongodb.internal.connection.GetMoreProtocol.execute(GetMoreProtocol.java:52)
    at com.mongodb.internal.connection.DefaultServer$DefaultServerProtocolExecutor.execute(DefaultServer.java:172)
```

Nu is het zo gebouwd dat de schepen in kleinere batches worden ingeladen en de merge functie is uit de scheduler gehaald.

De laatste stand van zaken zijn op het Trello bord te zien. Onder anderen het verbeteren en sneller maken van de applicatie.



De applicatie ShipHappens is beschikbaar op: <http://demo.teqplay.nl/shiphappens/>