MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Alfenas
UNIFAL-MG
Av. Jovino Fernandes Sales, 2.600 - Alfenas - MG CEP 37133-840
http://www.unifal-mg.edu.br

JESPER BØLLING

Development of a Reporting Tool

Alfenas/MG

2020

JESPER BØLLING

Development of a Reporting Tool

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do título de bacharel em Ciêncas da Computação pela Universidade Federal de Alfenas.

Orientador: Flavio Gonzaga

Alfenas/MG

2020

JESPER BØLLING

Development of a Reporting Tool

A Banca Examinadora, abaixo assinada, aprova a dissertação apresentada como parte dos requisitos para a obtenção do título de bacharel em Ciências da Computação pela Universidade Federal de Alfenas.
Orientador: Flavio Gonzaga

Aprovada em: _____/_____/_____

Prof(a). Dr(a). ____

Prof(a). Dr(a). ____

Prof(a). Dr(a). ____

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Universidade Federal de Alfenas - UNIFAL/MG.

Eu agredeco os alunos e os professores do UNIFAL pela ajuda e paciência comigo durante o meu curso. Um estranheiro tentando caber, numa outra cultura.

Eu agredeco Professor Flavio Gonzaga, pelas dicas e ajuda com o presente trabalho. Flavio sempre foi positivo, muito competente, e preparado para ajudar na melhor forma possivel.

ABSTRACT

Business intelligence (BI) has been applied in various domains to take better decisions and it provides different level of information to its stakeholders according to the information needs. This project aimed at developing a generic report platform consisting of a mobile application (reporting tool), to be installed on a tablet or smartphone, and a web application (administration module), that can be run in any internet browser. The report platform can be used to report the state of assets, and helps shortening the process from information gathering to decision making based on the gathered information. The mobile application was developed for Android and iOS using Xamarin (C#), whereas the web application was developed using Angular. Everything is backed up by a COSMOS database and the communication between web application/database and mobile application/database was being placed in a separate API layer backup up by CORE 2.1. The mobile application makes it possible to gather information about assets in the business and generate reports that can be sent to stakeholders. The information gathering process includes evaluating assets, taking photos of assets on the spot and send it all in one package to a central server with the administration module installed. The administration module is where the uploaded reports can be visualized and furthermore PDF-reports can be printed out for later use. In this project different components were explored in Xamarin.Forms, architectural differences in database design and finally, all was put it together in a solution that is implementable in almost every company with assets. The above project was developed using a world leading Tanker company as a use case. After finished developing the solution was integrated, and is being used by the companys engineers in inspecting Tankers around the world.

Keywords: Business intelligence, Assets reporting, reporting tool.

LISTA DE ILUSTRAÇÕES

TABLE OF CONTENTS

1. INTRODUCTION

In a world being digital, companies in general are replacing pen and paper with computers / tablets etc. What is common for them all is, that they all have the need for actionable insights to improve their business. The more information companies have about their business and the better and faster they are to process the data afterwards, the better the companies will be able to optimize their profit.

“As the saying goes, “knowledge is power.” To derive valuable knowledge from your company’s operating data requires the development of an effective business intelligence (BI) program. Simply put, BI is a technology-driven process for analyzing data and presenting actionable information to help corporate executives, managers and others make more informed decisions. For example, if you’re clear about which customers spend the most money and what they buy, you can develop marketing campaigns and promotions focused on selling to these high-value customers.” (Maxwell, Locke & Ritter, 2017)

Locating, gathering, analyzing and reporting data and continuously monitoring are key points in order to archive an effective business intelligence.
Information about its own business is one of the keys on the road to optimized profit, and knowhow in order to streamline their processes about how to obtain, store and process information.

1.1 PROBLEM CHARACTERIZATION

A lot of business today are in a large extend dependent on the collection of information by the use of printed forms.

In business such as Car rental companies, Airline industries, freight industries, real estates, where the core business is based on the how the ongoing maintenance of the business assets, it is of outmost importance that the business administration has information about the whereabouts and the condition of each asset in the company.

In many businesses the process of eventual maintenance includes, filling out a printed form; inspect the asset on site; send the form somewhere; receive the form; process the information on the form; determine the condition of the asset.

A great deal of business today is, as of today still using plain pen and paper in this process.

This can result delay in the time the information is being received by central administration. There is a risk of loosing filled out paper forms, resulting in a new report request. Finally data loss, spelling errors, when copying paper reports to other types of data forms.

The processes for the above items are crucial for every business, as it has a direct impact on the profit if one fails. The faster and more precisely the above process can be executed, the faster and better maintenance can be carried out.

2. LITERATURE REVIEW

"Business Intelligence (BI) software is a collection of decision support technologies for the enterprise aimed at enabling knowledge workers such as executives, managers, and analysts to make better and faster decisions." (Chaudhuri, et al., 2011)**.**

Acording to the article there is a rapid growth in number of products and services offered worldwide and the adoption of these in different industries, which justifies an increasingly need for supporting BI tasks near real time, which make business decisions based on the operational data itself possible.

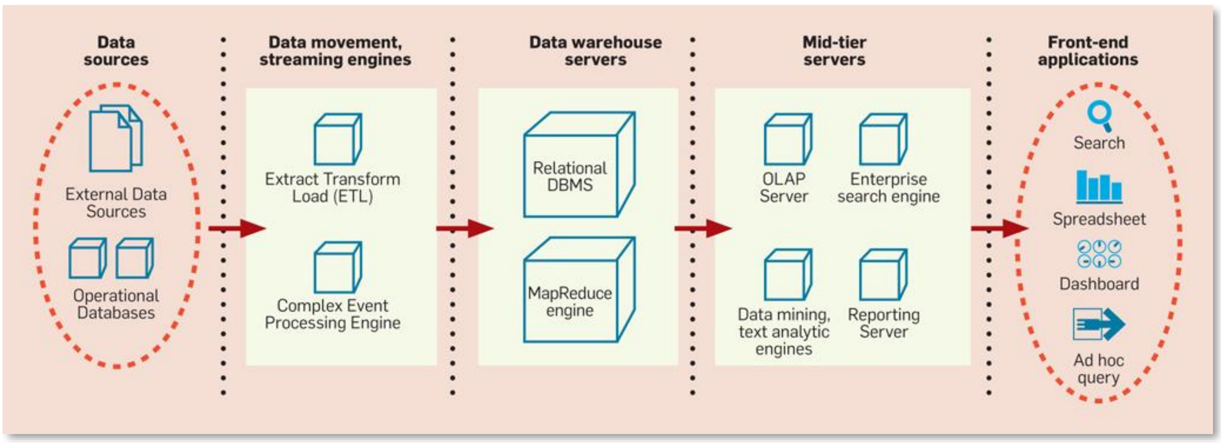


Figure 1 - Typical Business intelligence architecture

Source: https://cacm.acm.org/magazines/2011/8/114953-an-overview-of-business-intelligence-technology/fulltext

The article focuses on the technologies in the shaded boxes in ***Figure 1 - Typical Business intelligence*** architecture**.**

Data movement and how data is being stored has a big impact on the final solution. "The difference in execution time between a good and bad plan for such complex queries can be several orders of magnitudes (for example, days instead of minutes)." (Chaudhuri, et al., 2011)**.**

The goal here is to reduce the latency between when the operational data is acquired from the sources and then when analysis over that data is possible which can be tanslated to "reduce time between data collection and the data processing before being presented", which will be the focus of this work. The data collection and the data presentation are the extremities in ***Figure 1 - Typical Business intelligence*** architecture

A specific Asset management system has been proposed in (COPRPORATION, 2018) where an overall architectural plan has been provided for an asset management system. Despite being in different contexts, the work , presents similarities to the proposed solution. The main difference is, that in the work above, the inspection is to be made in buildings and constructions that have a fixed location and in the proposed solution the inspection is to be made in moving vessels all over the world.

A specific documentation in order to store, process and present the data afterwards, was not encountered. The expected outcome is very similar though:

"A system and method are disclosed herein that provides a property management system for an owner of a property to be able to electronically communicate via a single system with a building management company, inspection company, service company or other user on the system.", which can contribute to achieve the main objectives in this work.

A lot of asset management software tools are to be found on the market, though it is common for many of the tools, to be able to keep track of type, price, buy date, location in company etc.

Or in other words "inventory tracking". The inventory tracking itself is only part of what this work is proposing as it is also needed to be able to track the state of the single inventory item and be able to document it with photos.

A specific software tool that has the following abilities has not been found client, server application with possibility to work on the report "on site"; rating the state of assets and supplying photos along with asset remarks.

3. OBJECTIVES

3.1 GENERAL

The main objective is to develop a generic reporting tool/administration platform, to be able to conduct on site reporting on a mobile device (tablet/smart phone), aligned with the report templates made using the administration platform..

3.2 SPECIFIC

The specific objectives are:

- Develop an administration module where it is possible to verify user with Azure active directory, insert, change and delete items in the report template, visualize uploaded reports for every report object, visualize different versions of reports for different report objects, download a report in PDF-format;
- Develop a mobile application (reporting tool) to be used on iOS/Android tablets/smart phones. The reporting tool shall be able to verify user with Azure active directory, download latest report template, browse through report groups/items, update report item with: Rating, Remark, Images, upload the finished report to a central server;
- Develop a back-end API that is accessible by both the mobile application and the administration platform and stores the all reports in location, independent of different templates.

4. DEVELOPMENT

System functionality overview

In this chapter the details about the developed project is being presented. The development was divided in three parts being the Administration portal, Reporting tool and Client Portal.

Details about the database to hold information and data subject to the above-mentioned solutions is being presented at the end of this chapter.

4.1 ADMINISTRATION PORTAL

The administration portal is where the report template is being configurated. And thereby the base for all the reports. The portal is accessible using a internet browser.

In order to configure the report template, a login with valid credentials is needed. The login is being validated against the company’s Azure Active directory as shown on ***Figure 2 - Login*** window.

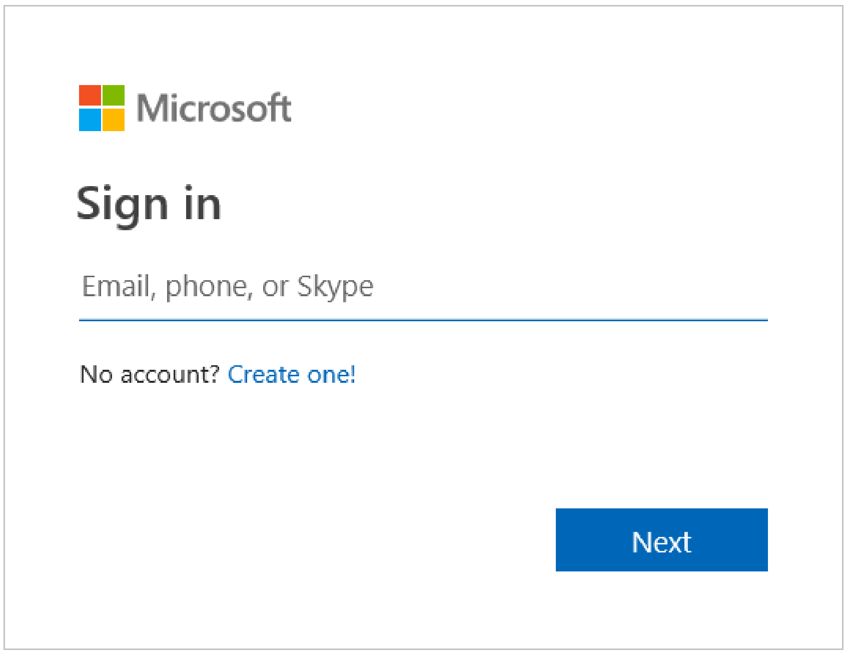
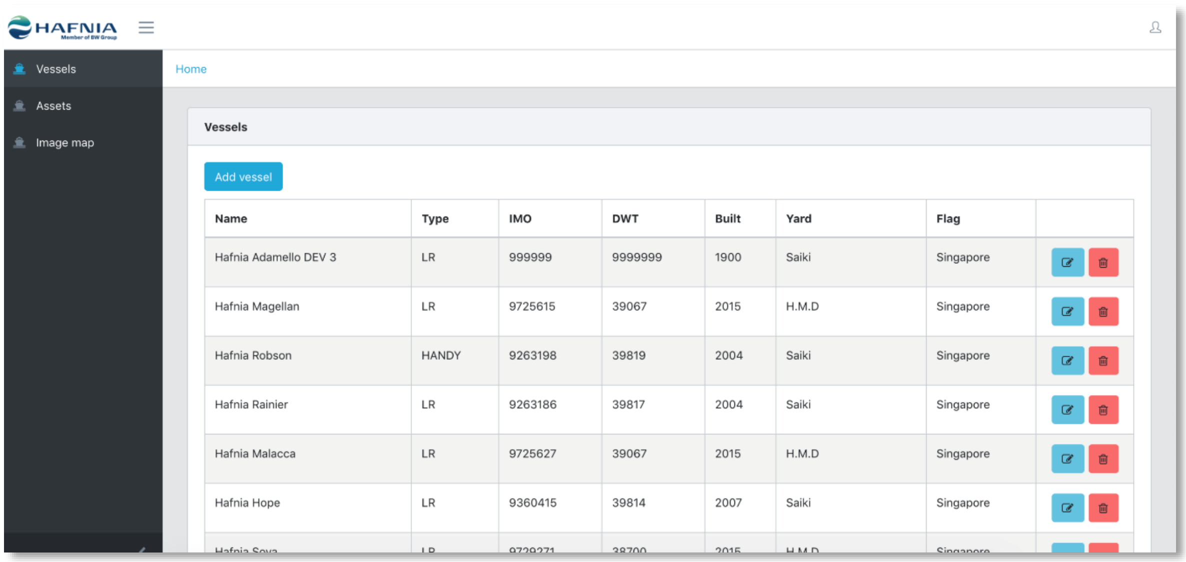


Figure 2 - Login window

Source: Own application

When logged in the user will encounter the configuration section.

It is here possible to configure the objects to be able to make reports over, which in this case are Vessels as shown in ***Figure 3 - Configuration - Vessels*** overview**.**

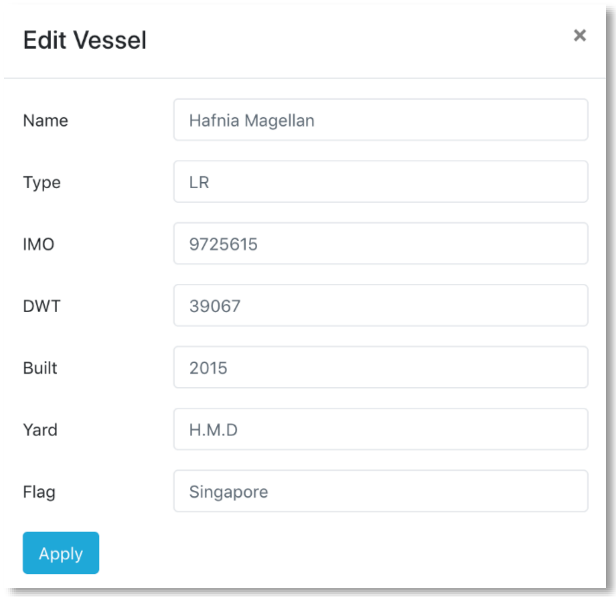


Name	Type	IMO	DWT	Built	Yard	Flag
Hafnia Adamello DEV 3	LR	999999	9999999	1900	Saiki	Singapore
Hafnia Magellan	LR	9725615	39067	2015	H.M.D	Singapore
Hafnia Robson	HANDY	9263198	39819	2004	Saiki	Singapore
Hafnia Rainier	LR	9263186	39817	2004	Saiki	Singapore
Hafnia Malacca	LR	9725627	39067	2015	H.M.D	Singapore
Hafnia Hope	LR	9360415	39814	2007	Saiki	Singapore
Hafnia Sova	LR	9729271	38700	2015	H.M.D	Singapore

Figure 3 - Configuration - Vessels overview

Source: Own application

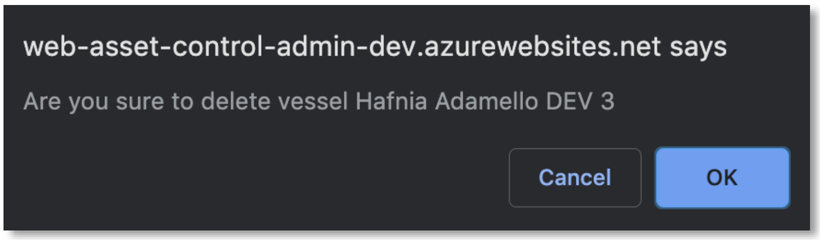
It is possible to add, edit and delete vessels as shown in ***Figure 4 - Add/Edit*** vessel and ***Figure 5 - Delete*** vessel**.**



Name	Hafnia Magellan
Type	LR
IMO	9725615
DWT	39067
Built	2015
Yard	H.M.D
Flag	Singapore

Figure 4 - Add/Edit vessel

Source: Own application



web-asset-control-admin-dev.azurewebsites.net says

Are you sure to delete vessel Hafnia Adamello DEV 3

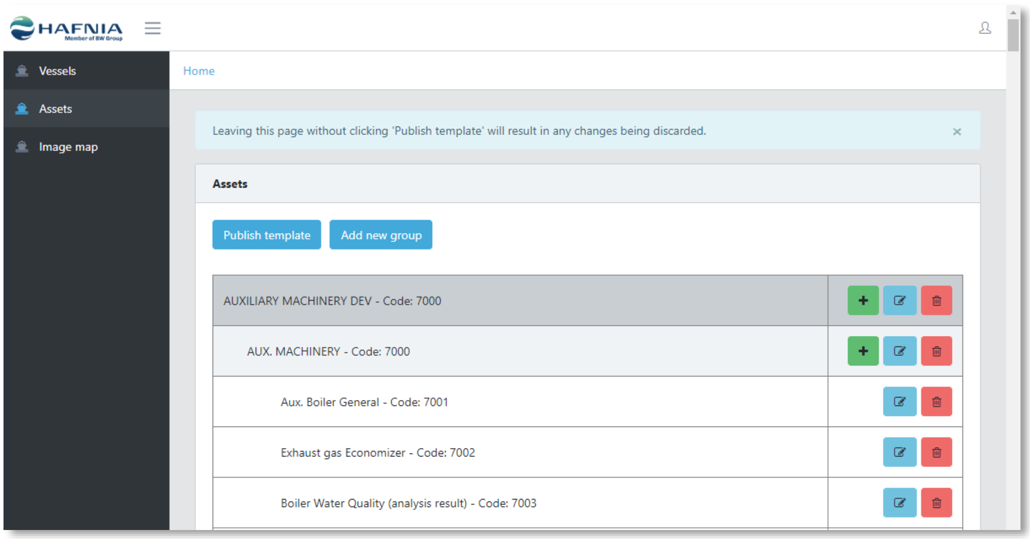
Figure 5 - Delete vessel

Source: Own application

When a vessel is being added, changed or deleted, a single document is being changed in the Vessels collection in the connected Cosmos database.

The next tab on the left is the assets configuration. This is where all the groups, subgroups and assets are created in the order that it will be presented in the reporting tool.

By pressing the button "Add new Group", a new main group can be created as shown ***on Figure 7 - New main group*** creation. After the new main group appears in the overview as shown on ***Figure 8 - New main group*** added**.**



Leaving this page without clicking 'Publish template' will result in any changes being discarded.

Assets

AUXILIARY MACHINERY DEV - Code: 7000
AUX. MACHINERY - Code: 7000
Aux. Boiler General - Code: 7001
Exhaust gas Economizer - Code: 7002
Boiler Water Quality (analysis result) - Code: 7003

Figure 6 - Assets overview

Source: Own application

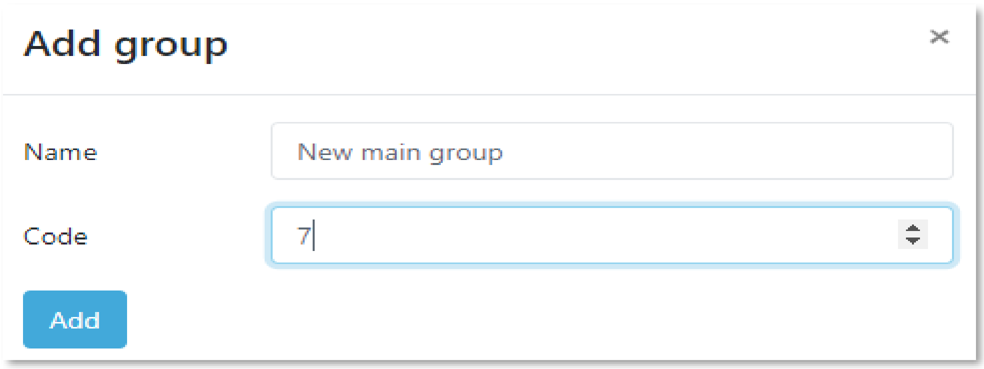
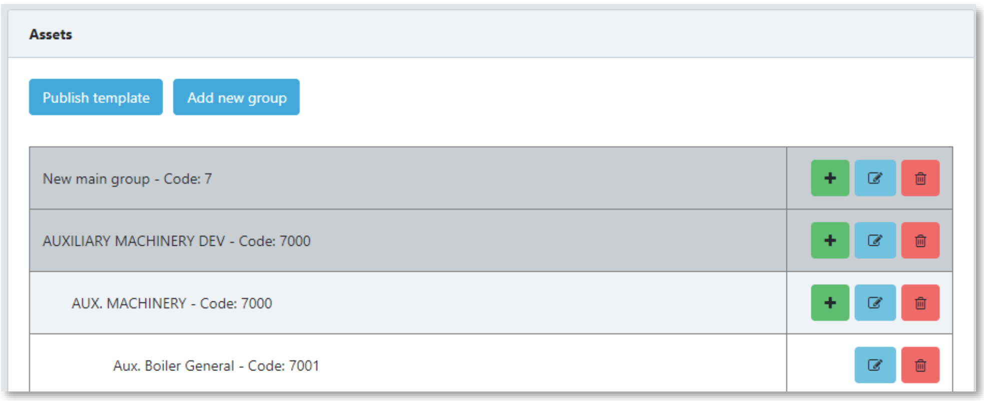


Figure 7 - New main group creation

Source: Own application



New main group - Code: 7

AUXILIARY MACHINERY DEV - Code: 7000

AUX. MACHINERY - Code: 7000

Aux. Boiler General - Code: 7001

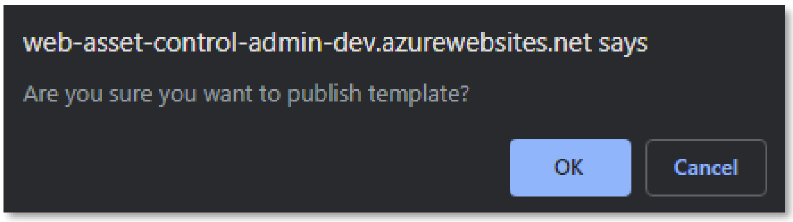
Figure 8 - New main group added

Source: Own application

Clicking the button with the “plus” sign, in the view will add a subgroup or a subitem to the corresponding group/line clicked on.

When the configuration is finished, the “Publish template” button must be pressed in order to save the changes to the database.

When the “Publish template” is pressed, a confirmation dialog as shown on ***Figure 9 - Publish template*** confirmation appears. Changes will be written to the Cosmos database in the template collection, updating the report template, that will be downloaded by the Reporting tool, whenever the Reporting tool synchronizes.


web-asset-control-admin-dev.azurewebsites.net says

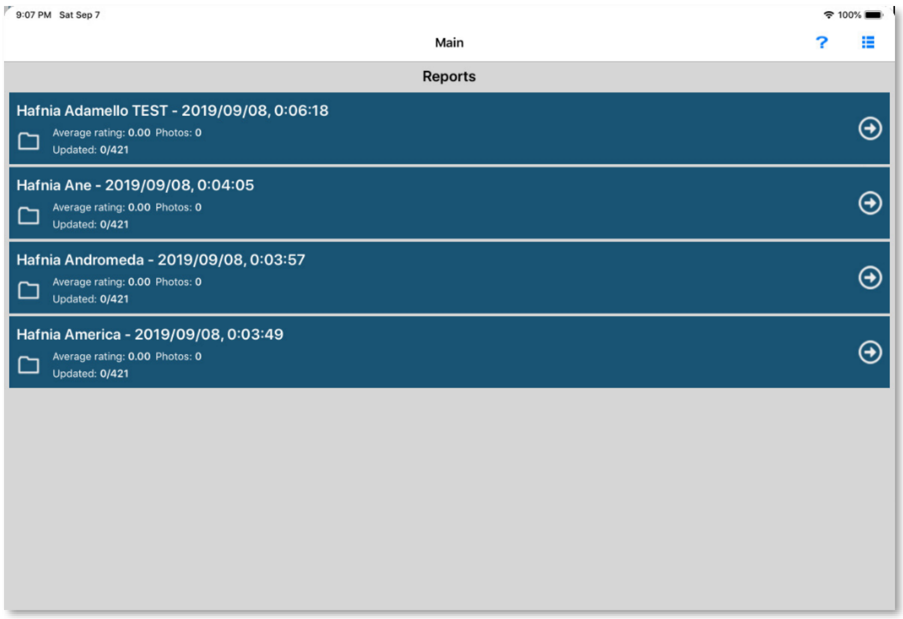
Are you sure you want to publish template?

Figure 9 - Publish template confirmation

Source: Own application

4.2 REPORTING TOOL

The reporting tool can run on either iOS v9.00+ or Android v8.00+. The functions on the 2 platforms are the same. The only thing that differs is the shape of buttons, message boxes, etc. The first time the reporting tool starts up the user will be prompted to sign in with user credentials as shown on ***Figure 2 - Login*** window. The credentials are being verified against the Azure Active Directory belonging to the company. After successful verification, the user will be presented with the Reports List. Ongoing reports and already uploaded reports will be shown in the list as shown on ***Figure 10 - Reports*** list**.**



Reports

Hafnia Adamello TEST - 2019/09/08, 0:06:18
Average rating: 0.00 Photos: 0
Updated: 0/421

Hafnia Ane - 2019/09/08, 0:04:05
Average rating: 0.00 Photos: 0
Updated: 0/421

Hafnia Andromeda - 2019/09/08, 0:03:57
Average rating: 0.00 Photos: 0
Updated: 0/421

Hafnia America - 2019/09/08, 0:03:49
Average rating: 0.00 Photos: 0
Updated: 0/421

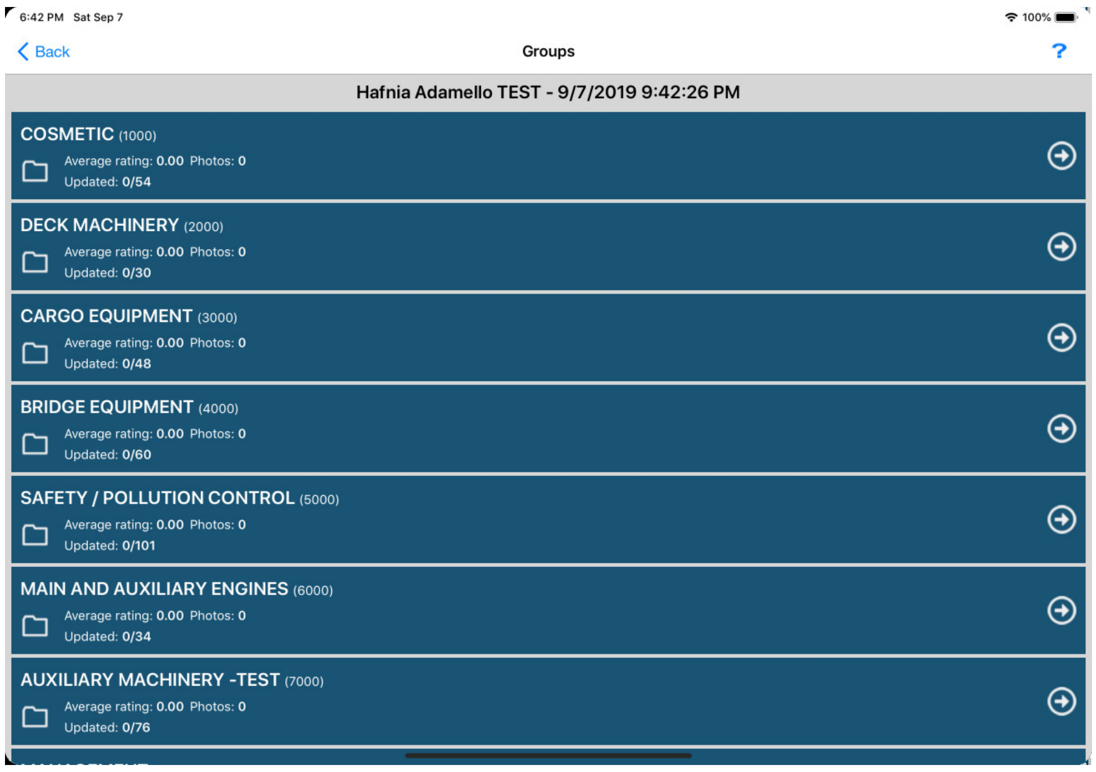
Figure 10 - Reports list

Source: Own application

The first time the user logs on, the Reporting Tool automatically downloads the published report template shown on ***Figure 6 - Assets*** overview. This report template will be used to generate new reports. If a new report template is downloaded, it will only be used as a base for new reports. The reports started before updating the report template remains unchanged.

When creating or selecting an existing report entry, the main asset groups are being shown with sub groups statistics as shown on ***Figure 11 - Reporting tool - Asset*** groups**.**

Navigating through subgroups on ***Figure 12 - Reporting tool - Asset*** subgroups**,** and selecting an asset on ***Figure 13 - Reporting tool –*** Assets will lead to the asset overview as shown on ***Figure 14 - Reporting tool - asset*** overview



Hafnia Adamello TEST - 9/7/2019 9:42:26 PM

COSMETIC (1000)
Average rating: 0.00 Photos: 0
Updated: 0/54

DECK MACHINERY (2000)
Average rating: 0.00 Photos: 0
Updated: 0/30

CARGO EQUIPMENT (3000)
Average rating: 0.00 Photos: 0
Updated: 0/48

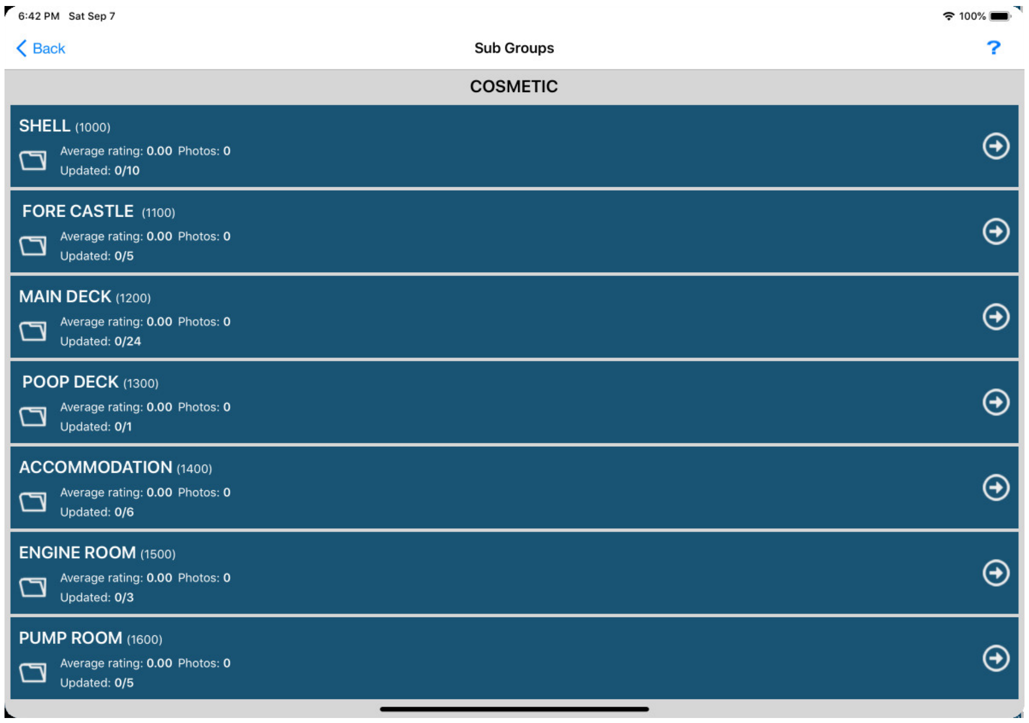
BRIDGE EQUIPMENT (4000)
Average rating: 0.00 Photos: 0
Updated: 0/60

SAFETY / POLLUTION CONTROL (5000)
Average rating: 0.00 Photos: 0
Updated: 0/101

MAIN AND AUXILIARY ENGINES (6000)
Average rating: 0.00 Photos: 0
Updated: 0/34

AUXILIARY MACHINERY -TEST (7000)
Average rating: 0.00 Photos: 0
Updated: 0/76

Figure 11 - Reporting tool - Asset groups
Source: Own application



COSMETIC

SHELL (1000)
Average rating: 0.00 Photos: 0
Updated: 0/10

FORE CASTLE (1100)
Average rating: 0.00 Photos: 0
Updated: 0/5

MAIN DECK (1200)
Average rating: 0.00 Photos: 0
Updated: 0/24

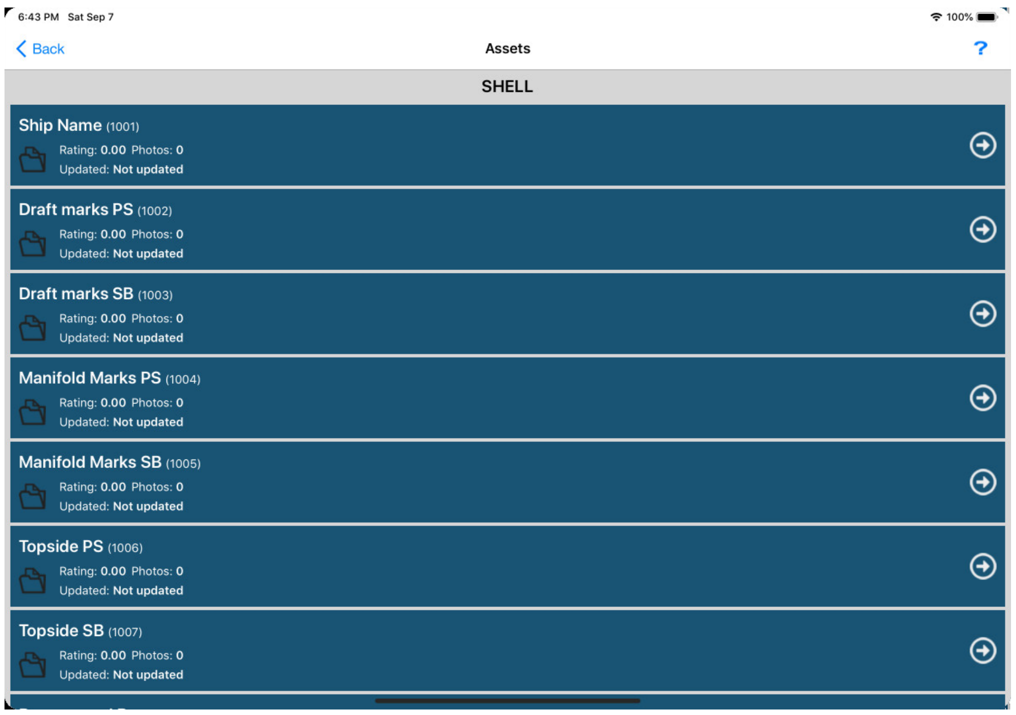
POOP DECK (1300)
Average rating: 0.00 Photos: 0
Updated: 0/1

ACCOMMODATION (1400)
Average rating: 0.00 Photos: 0
Updated: 0/6

ENGINE ROOM (1500)
Average rating: 0.00 Photos: 0
Updated: 0/3

PUMP ROOM (1600)
Average rating: 0.00 Photos: 0
Updated: 0/5

Figure 12 - Reporting tool - Asset subgroups
Source: Own application



SHELL

Ship Name (1001)
Rating: 0.00 Photos: 0
Updated: Not updated

Draft marks PS (1002)
Rating: 0.00 Photos: 0
Updated: Not updated

Draft marks SB (1003)
Rating: 0.00 Photos: 0
Updated: Not updated

Manifold Marks PS (1004)
Rating: 0.00 Photos: 0
Updated: Not updated

Manifold Marks SB (1005)
Rating: 0.00 Photos: 0
Updated: Not updated

Topside PS (1006)
Rating: 0.00 Photos: 0
Updated: Not updated

Topside SB (1007)
Rating: 0.00 Photos: 0
Updated: Not updated

Figure 13 - Reporting tool – Assets
Source: Own application

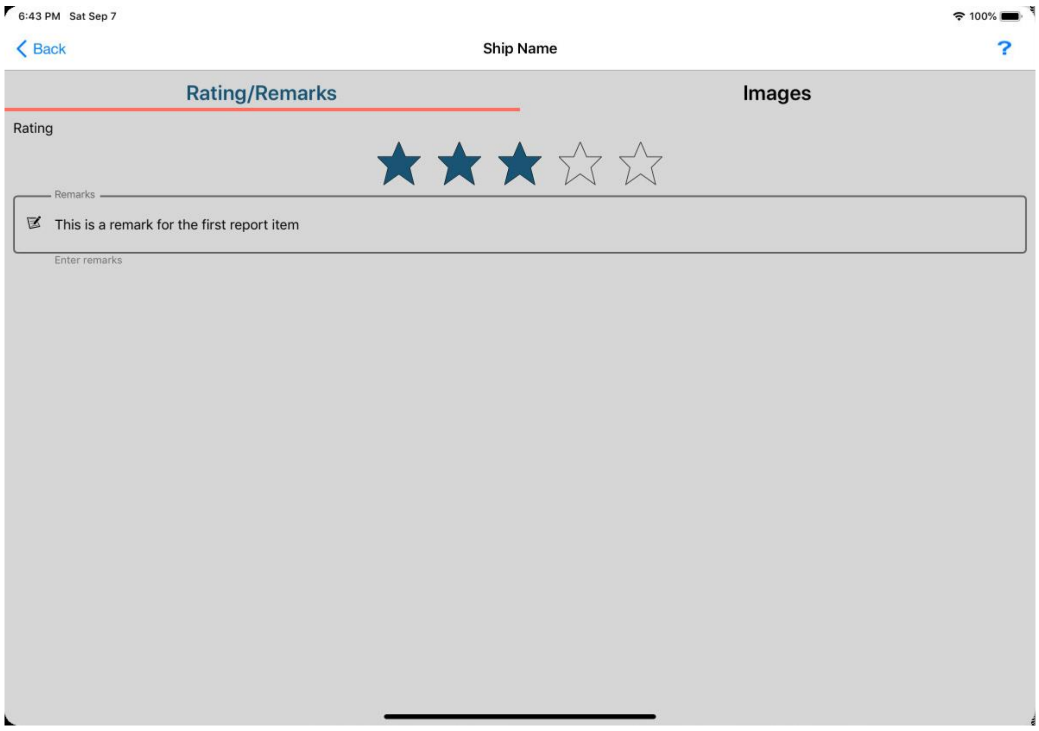


Figure 14 - Reporting tool - asset overview
Source: Own application

On the Ratings/Remarks tab on the assets overview, it is possible to give a rating between 0 and 5, and a remark for the given report item. The rating given to the report item here, is being used to calculate the average for the given asset subgroup, asset group and report. The calculated averages can be seen in each overview.

Selecting the Images tab on assets overview, shows the associated images to the report item as shown on ***Figure 15 - Reporting tool – images*** overview**.** New images can be inserted either by inserting them from the camera roll on the device, or by taking the image directly through the app. An attached image can be manipulated by clicking on the image in the overview.

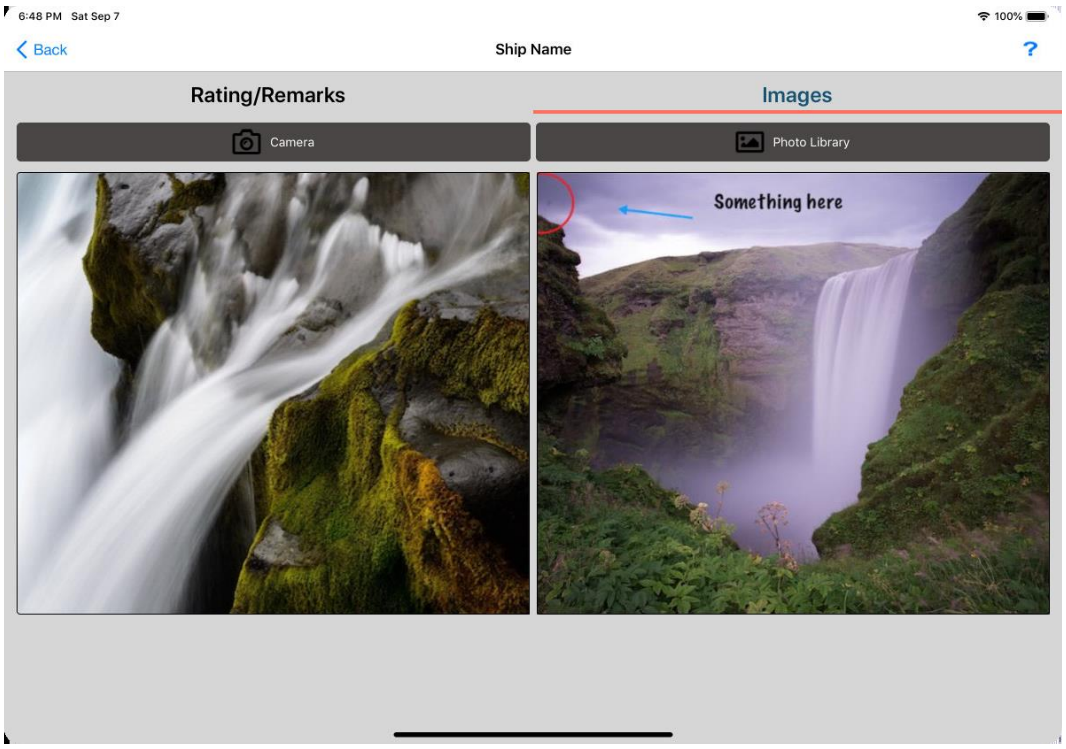


Figure 15 - Reporting tool – images overview
Source: Own application

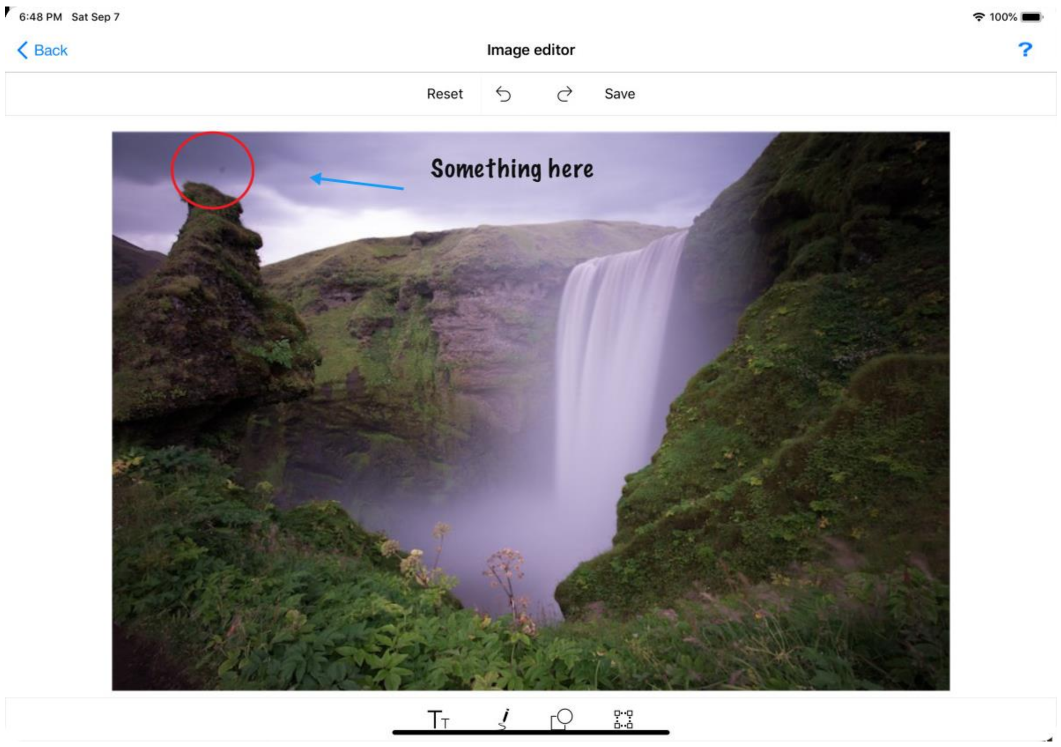


Figure 16 - Reporting tool - image edit
Source: Own application

When manipulating an image as shown on ***Figure 16 - Reporting tool - image*** edit, text, figures and manual drawings can be added to the image. The image can be cropped and rotated as well before saving the image and returning to the image overview.
Before uploading the report, the user must fill out the report overview as shown on ***Figure 17 - Reporting tools - report*** details**.**

After the report has been uploaded as shown on ***Figure 18 - Reporting tool - upload*** report**,** the state of the report on the local device will not be editable anymore. It can be unlocked and uploaded again. By doing that, the report in the Client Portal will be overwritten with the modified report.

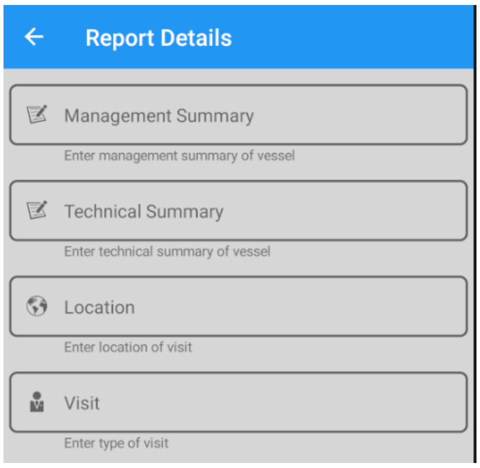
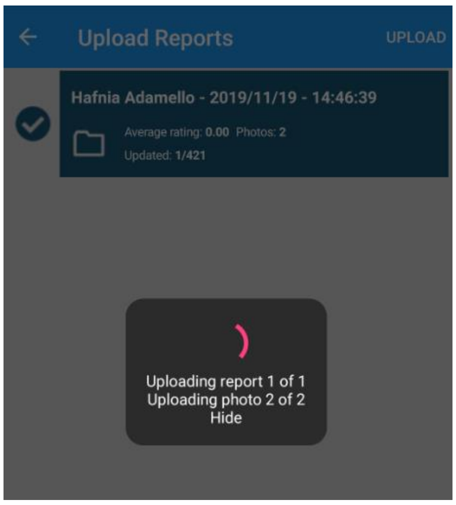


Figure 17 - Reporting tools - report details
Source: Own application



Hafnia Adamello - 2019/11/19 - 14:46:39
Average rating: 0.00 Photos: 2
Updated: 1/421

Uploading report 1 of 1
Uploading photo 2 of 2
Hide

Figure 18 - Reporting tool - upload report
Source: Own application

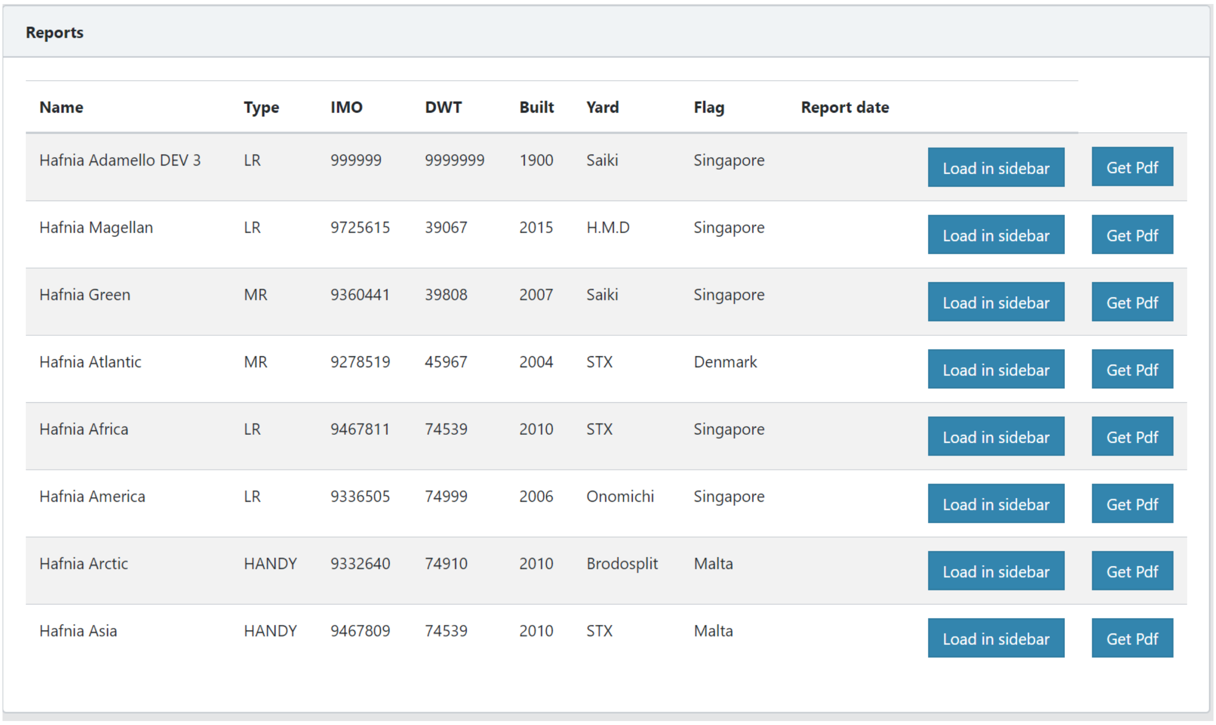
4.3 CLIENT PORTAL

The client is where the reports uploaded from the "Reporting Tool", can be accessed. The portal is accessible using a web browser.

The portal is secured in the same way as the Administration portal, so when entering the portal, the user will be asked to login as shown on ***Figure 2 - Login*** window**.**

After a successful login the user will have an overview of all the report objects as shown on ***Figure 19 - Client portal - report objects*** overview. In this case "Vessels".

Here it is possible to review the latest report for each vessel. That can either be shown on the page by clicking "Load in sidebar", or downloaded as a pdf file.

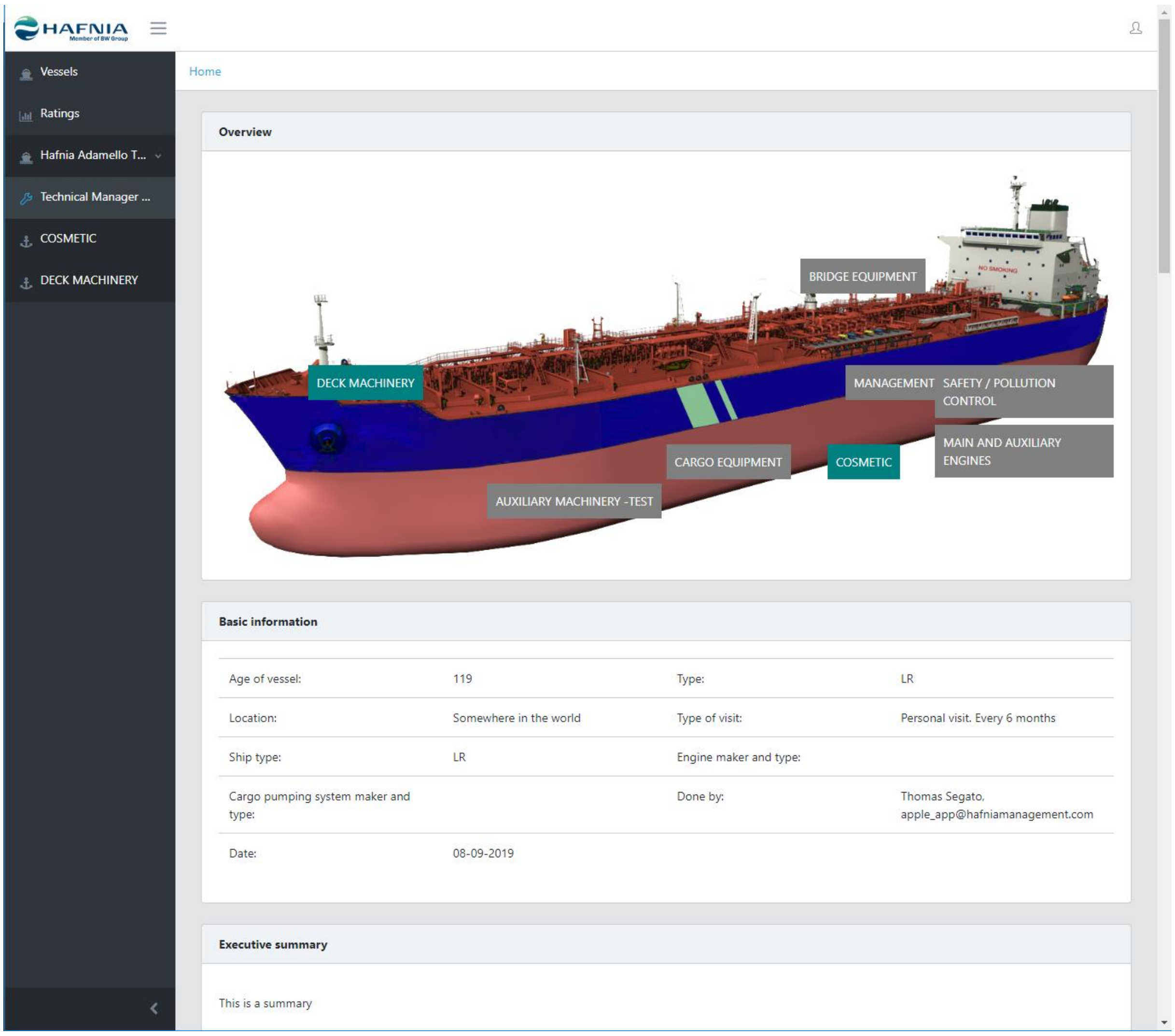


Reports

Name	Type	IMO	DWT	Built	Yard	Flag	Report date		
Hafnia Adamello DEV 3	LR	999999	9999999	1900	Saiki	Singapore		Load in sidebar	Get Pdf
Hafnia Magellan	LR	9725615	39067	2015	H.M.D	Singapore		Load in sidebar	Get Pdf
Hafnia Green	MR	9360441	39808	2007	Saiki	Singapore		Load in sidebar	Get Pdf
Hafnia Atlantic	MR	9278519	45967	2004	STX	Denmark		Load in sidebar	Get Pdf
Hafnia Africa	LR	9467811	74539	2010	STX	Singapore		Load in sidebar	Get Pdf
Hafnia America	LR	9336505	74999	2006	Onomichi	Singapore		Load in sidebar	Get Pdf
Hafnia Arctic	HANDY	9332640	74910	2010	Brodosplit	Malta		Load in sidebar	Get Pdf
Hafnia Asia	HANDY	9467809	74539	2010	STX	Malta		Load in sidebar	Get Pdf

Figure 19 - Client portal - report objects overview

Source: Own application



Age of vessel:	119	Type:	LR
Location:	Somewhere in the world	Type of visit:	Personal visit. Every 6 months
Ship type:	LR	Engine maker and type:	
Cargo pumping system maker and type:		Done by:	Thomas Segato, apple_app@hafniamanagement.com
Date:	08-09-2019		

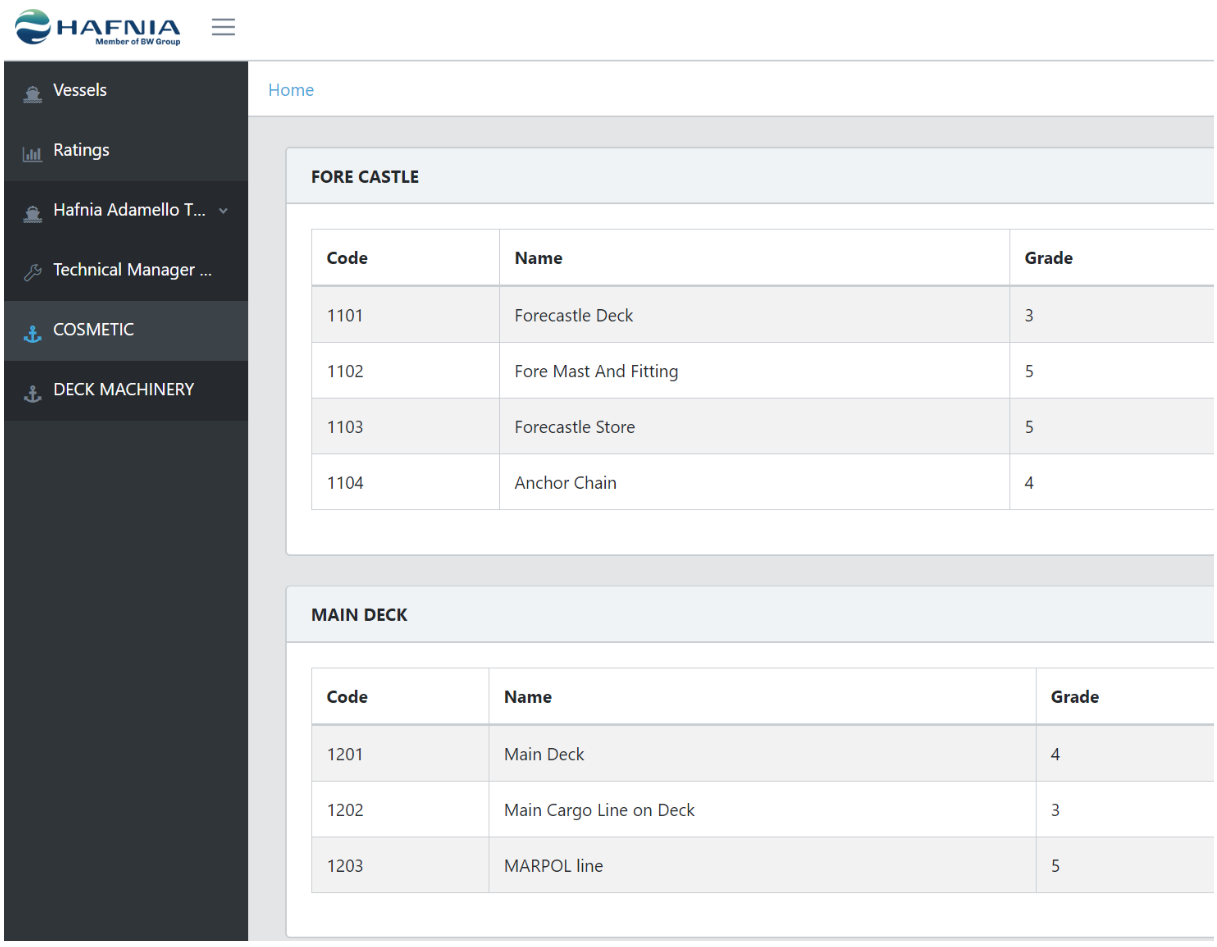
This is a summary

Figure 20 - Client Portal – vessel overview

Source: Own application

Figure 20 - Client Portal – vessel overview shows an overview on the vessel. There are labels with different colors on the vessel indicating what groups have a rating or not. Asset groups that have ratings are having a green label active and can clicked.

Clicking the label will result in a list with assets that have been evaluated in the particular group as shown on ***Figure 21 - Client portal – evaluated asset*** list**.**



FORE CASTLE

Code	Name	Grade
1101	Forecastle Deck	3
1102	Fore Mast And Fitting	5
1103	Forecastle Store	5
1104	Anchor Chain	4

MAIN DECK

Code	Name	Grade
1201	Main Deck	4
1202	Main Cargo Line on Deck	3
1203	MARPOL line	5

Figure 21 - Client portal – evaluated asset list

Source: Own application

Each evaluated asset is listed with name, code, grade and any remarks that may have been added. If there are any images associated with the asset, a button will appear on the end of the line. Clicking the button will show the associated image(s) in a modal as shown on ***Figure 22 - Client portal - image*** modal.

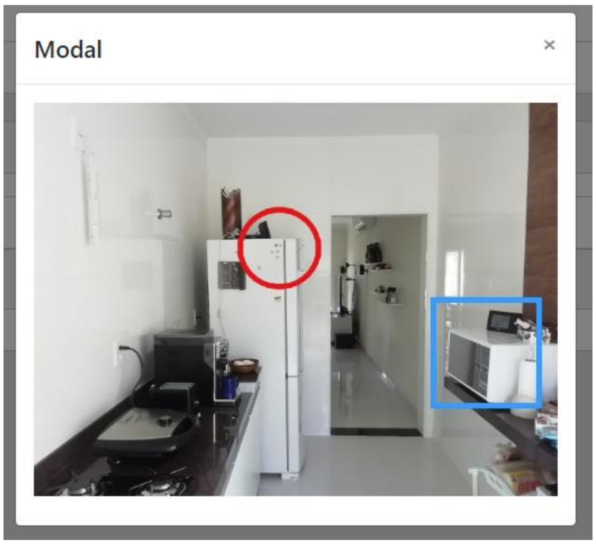
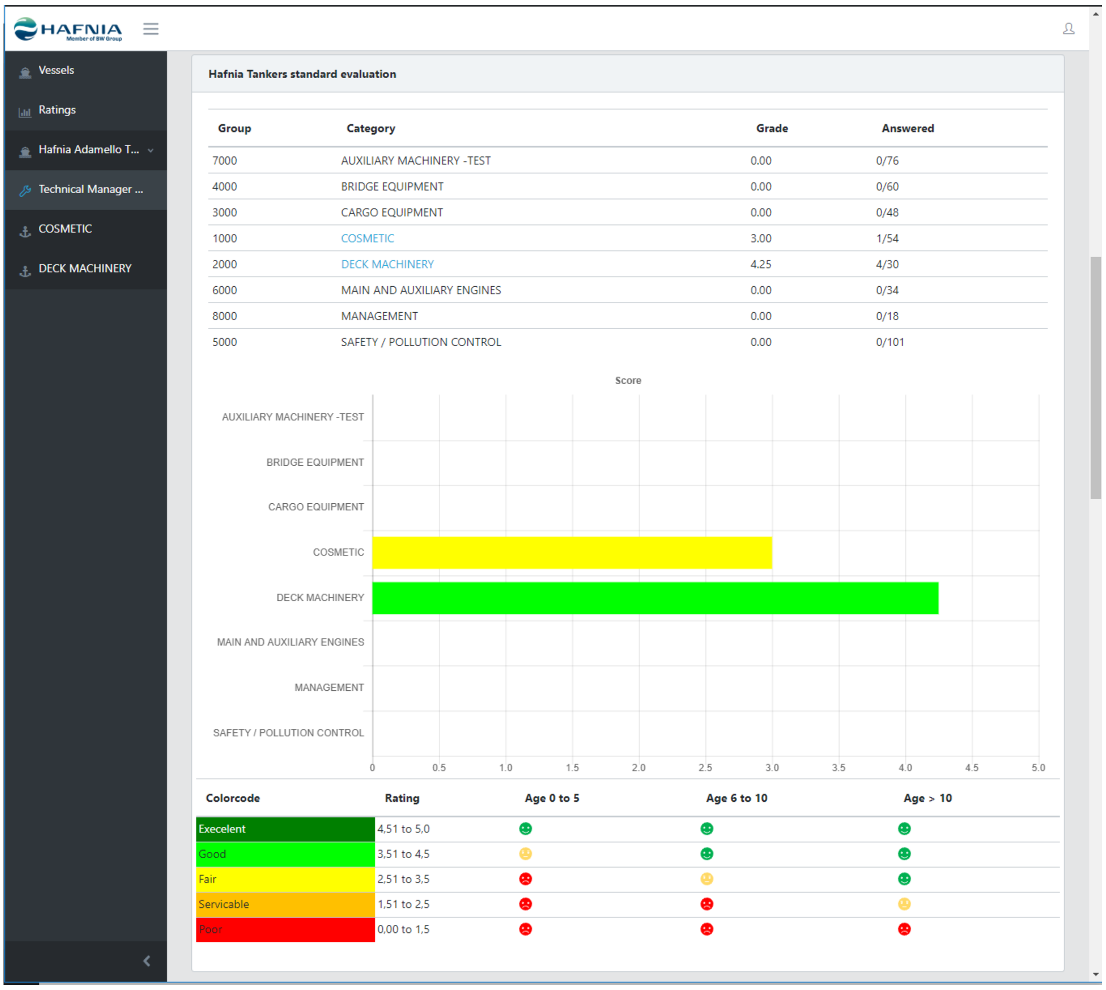


Figure 22 - Client portal - image modal

Source: Own application



Hafnia Tankers standard evaluation

Group	Category	Grade	Answered
7000	AUXILIARY MACHINERY -TEST	0.00	0/76
4000	BRIDGE EQUIPMENT	0.00	0/60
3000	CARGO EQUIPMENT	0.00	0/48
1000	COSMETIC	3.00	1/54
2000	DECK MACHINERY	4.25	4/30
6000	MAIN AND AUXILIARY ENGINES	0.00	0/34
8000	MANAGEMENT	0.00	0/18
5000	SAFETY / POLLUTION CONTROL	0.00	0/101

Colorcode	Rating	Age 0 to 5	Age 6 to 10	Age > 10
Excelent	4,51 to 5,0	🟢	🟢	🟢
Good	3,51 to 4,5	🟡	🟢	🟢
Fair	2,51 to 3,5	🔴	🟡	🟢
Servicable	1,51 to 2,5	🔴	🔴	🟡
Poor	0,00 to 1,5	🔴	🔴	🔴

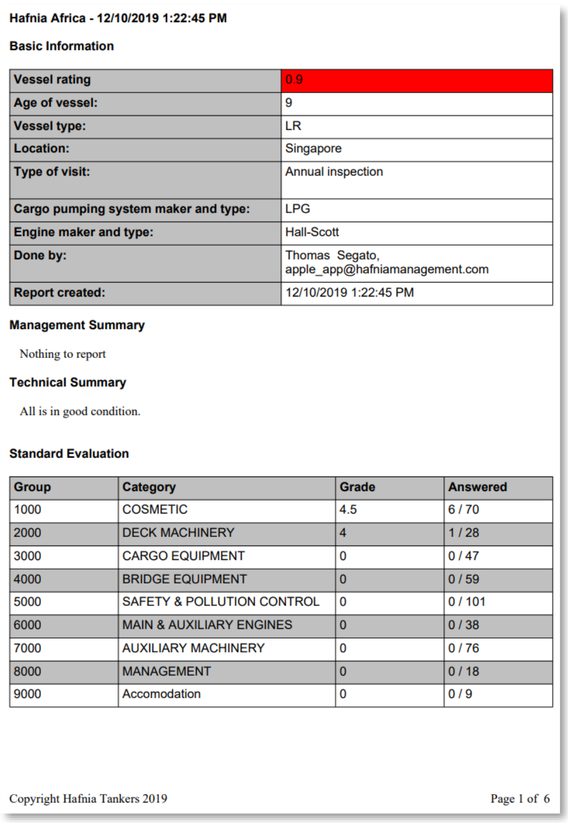
Figure 23 - Client portal - vessel report on page

Source: Own application

Figure 23 - Client portal - vessel report on page is the core of the whole application. The main asset groups are listed with their respective name, and calculated average. Furthermore, amount of evaluated assets and total assets in each group are listed. The average grade is based solely on the report items that have been evaluated.

The classification of each group can be read in the Score-table and is based on the average score for each group. Based on that, and the age of the ship, the color code table tells the condition of the given area, and if service is needed.

It is possible to save the report as a pdf as well. A such is shown on ***Figure 24 - Client portal - pdf*** report**.**



Hafnia Africa - 12/10/2019 1:22:45 PM

Basic Information

Vessel rating	🟥 0.9
Age of vessel:	9
Vessel type:	LR
Location:	Singapore
Type of visit:	Annual inspection
Cargo pumping system maker and type:	LPG
Engine maker and type:	Hall-Scott
Done by:	Thomas Segato, apple_app@hafniamanagement.com
Report created:	12/10/2019 1:22:45 PM

Management Summary

Nothing to report

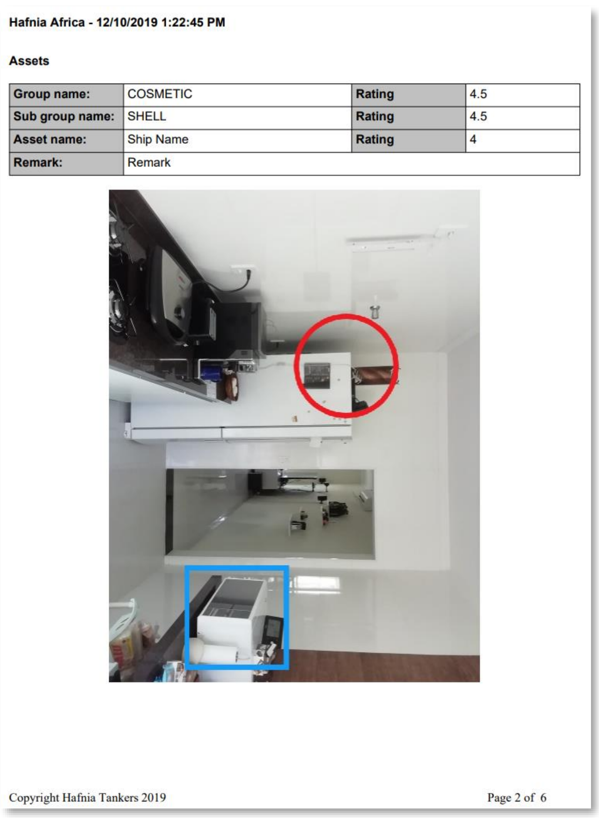
Technical Summary

All is in good condition.

Standard Evaluation

Group	Category	Grade	Answered
1000	COSMETIC	4.5	6 / 70
2000	DECK MACHINERY	4	1 / 28
3000	CARGO EQUIPMENT	0	0 / 47
4000	BRIDGE EQUIPMENT	0	0 / 59
5000	SAFETY & POLLUTION CONTROL	0	0 / 101
6000	MAIN & AUXILIARY ENGINES	0	0 / 38
7000	AUXILIARY MACHINERY	0	0 / 76
8000	MANAGEMENT	0	0 / 18
9000	Accomodation	0	0 / 9

Copyright Hafnia Tankers 2019 Page 1 of 6

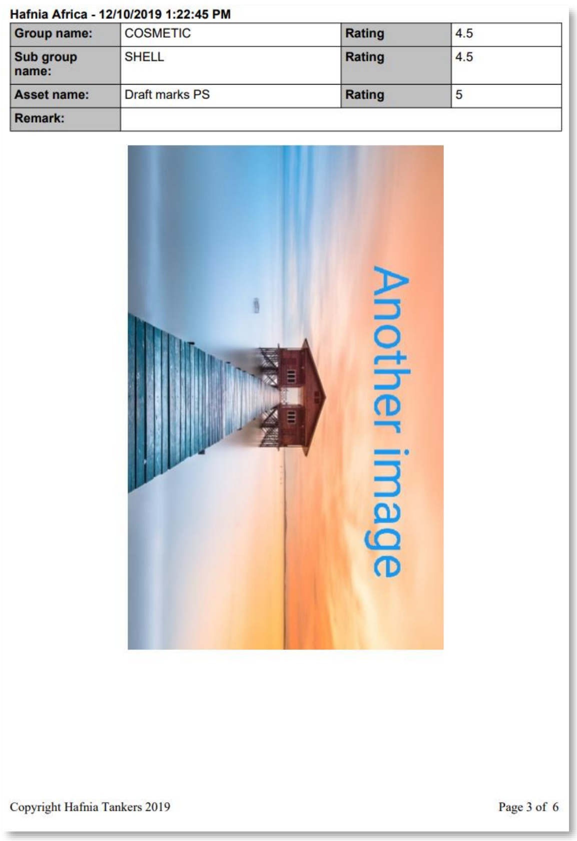


Hafnia Africa - 12/10/2019 1:22:45 PM

Assets

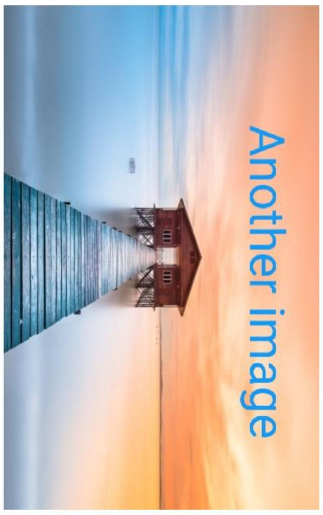
Group name:	COSMETIC	**Rating**	4.5
Sub group name:	SHELL	**Rating**	4.5
Asset name:	Ship Name	**Rating**	4
Remark:	Remark		

Copyright Hafnia Tankers 2019 Page 2 of 6

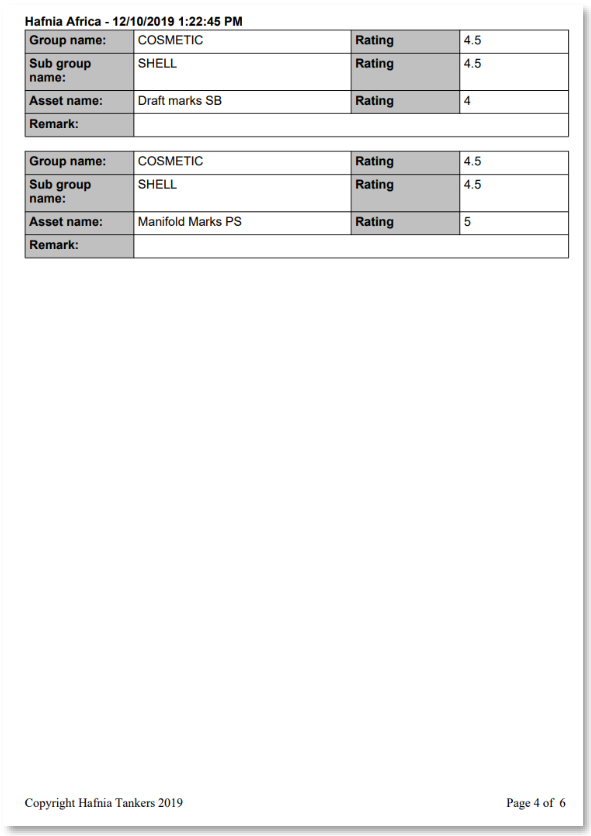


Hafnia Africa - 12/10/2019 1:22:45 PM

Group name:	COSMETIC	**Rating**	4.5
Sub group name:	SHELL	**Rating**	4.5
Asset name:	Draft marks PS	**Rating**	5
Remark:			



Copyright Hafnia Tankers 2019 Page 3 of 6

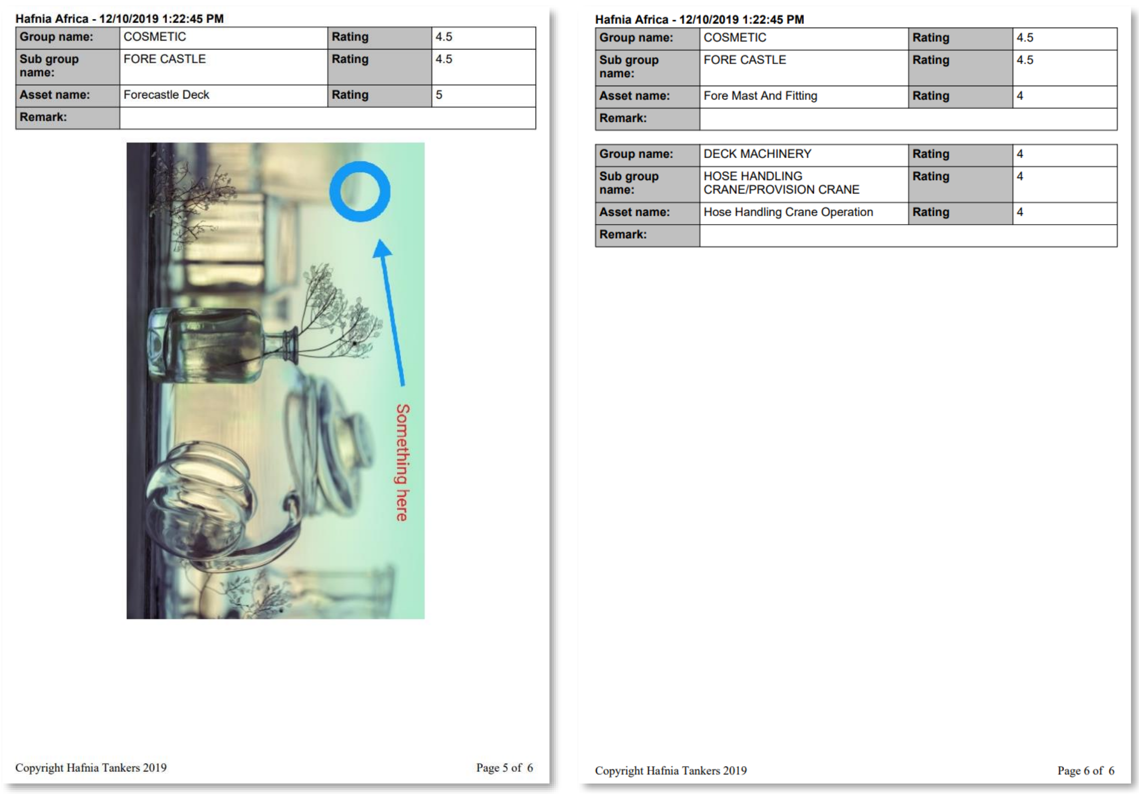


Hafnia Africa - 12/10/2019 1:22:45 PM

Group name:	COSMETIC	**Rating**	4.5
Sub group name:	SHELL	**Rating**	4.5
Asset name:	Draft marks SB	**Rating**	4
Remark:			

Group name:	COSMETIC	**Rating**	4.5
Sub group name:	SHELL	**Rating**	4.5
Asset name:	Manifold Marks PS	**Rating**	5
Remark:			

Copyright Hafnia Tankers 2019 Page 4 of 6



Hafnia Africa - 12/10/2019 1:22:45 PM

Group name:	COSMETIC	**Rating**	4.5
Sub group name:	FORE CASTLE	**Rating**	4.5
Asset name:	Forecastle Deck	**Rating**	5
Remark:			

Copyright Hafnia Tankers 2019 Page 5 of 6

Hafnia Africa - 12/10/2019 1:22:45 PM

Group name:	COSMETIC	**Rating**	4.5
Sub group name:	FORE CASTLE	**Rating**	4.5
Asset name:	Fore Mast And Fitting	**Rating**	4
Remark:			

Group name:	DECK MACHINERY	**Rating**	4
Sub group name:	HOSE HANDLING CRANE/PROVISION CRANE	**Rating**	4
Asset name:	Hose Handling Crane Operation	**Rating**	4
Remark:			

Copyright Hafnia Tankers 2019 Page 6 of 6

Figure 24 - Client portal - pdf report

Source: Own application

4.4 GENERAL STRUCTURE OF THE PROJECT

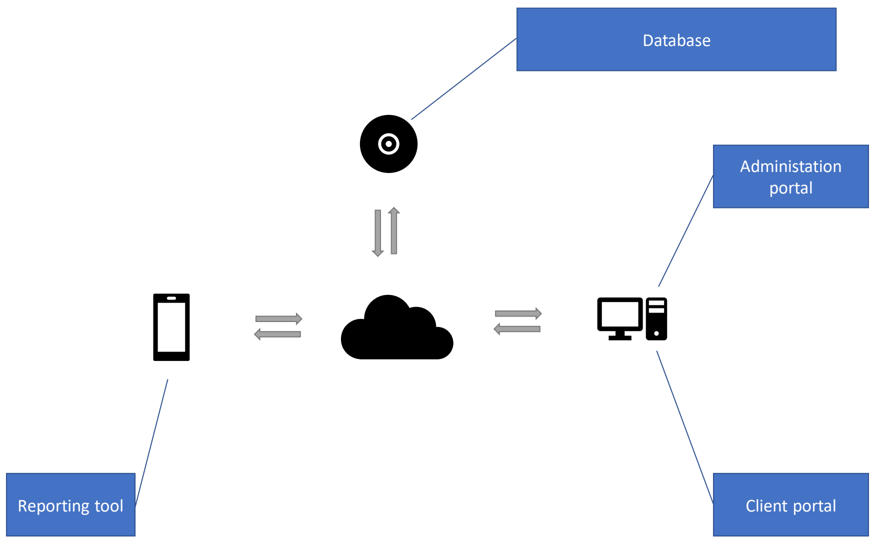


Figure 25 - Overall structure of project

Source: Own application

The project is structured as shown above on ***Figure 25 - Overall structure of*** project**.**

The Database is where all the information is being stored. That is: Report Templates, report objects and filled out reports for every report object.

The Database is only being accessed by the back-end API. The back-end API is responsible for the communication to/from the Reporting tool, Client portal, administration portal and the Database.

4.4.1.Security

In order not to expose the data in the Database, access to the back-end *API* is being secured by a security layer based on an Active Directory (**Error! Reference source not found.**), *OAuth* and a provider using Active Directory Federation Services (**Error! Reference source not found.**).

The RootViewController provides the content view of the window.

Error! Reference source not found. works as a proxy service between the **Error! Reference source not found.** and the application where authentication is needed. Through **Error! Reference source not found.**, the application is granted a federated trust, so the users can log into the application through **Error! Reference source not found.**.

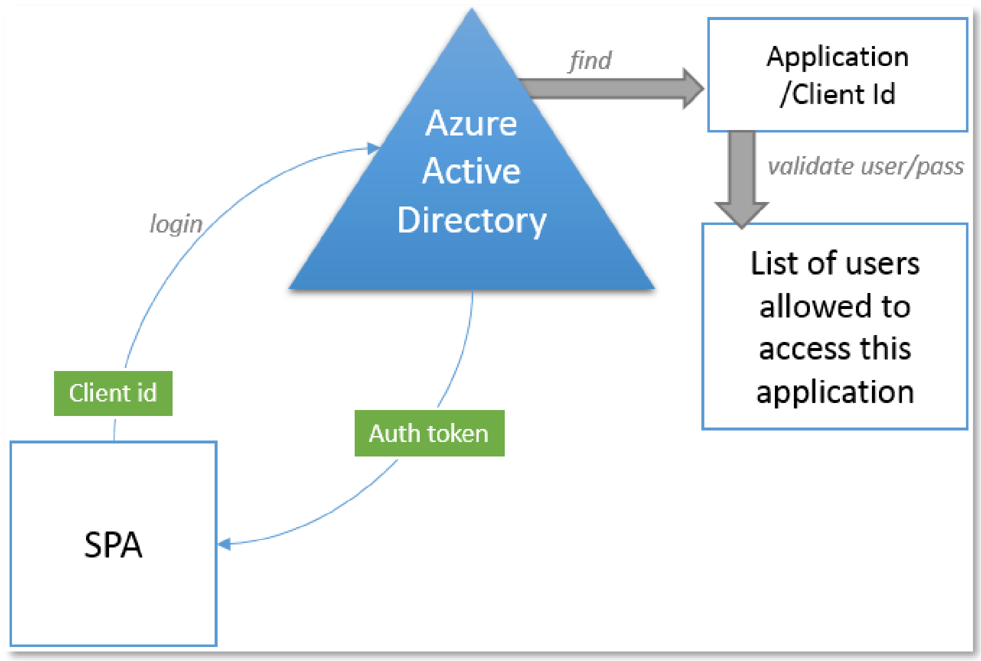


Figure 26 - ADFS Process

Source: Own application

The process for authentication shown on ***Figure 26 - ADFS*** Process is as follows:

- The *SPA* (Wikipedia, 2019) or Mobile device is being accessed by the user and client ID, domain and redirect URL is sent to the *ADFS*
- The received client ID is being used to authenticate the user through *SSO* against the AD using the domain.
- The user successfully authenticates on *AD*, and it is verified if the user has access to the application and if given redirect URL is registered within the application.
- The user is now being forwarded to the redirect URL together with an authentication claim (token), which either grants or denies access based on the Federated Trust service created.

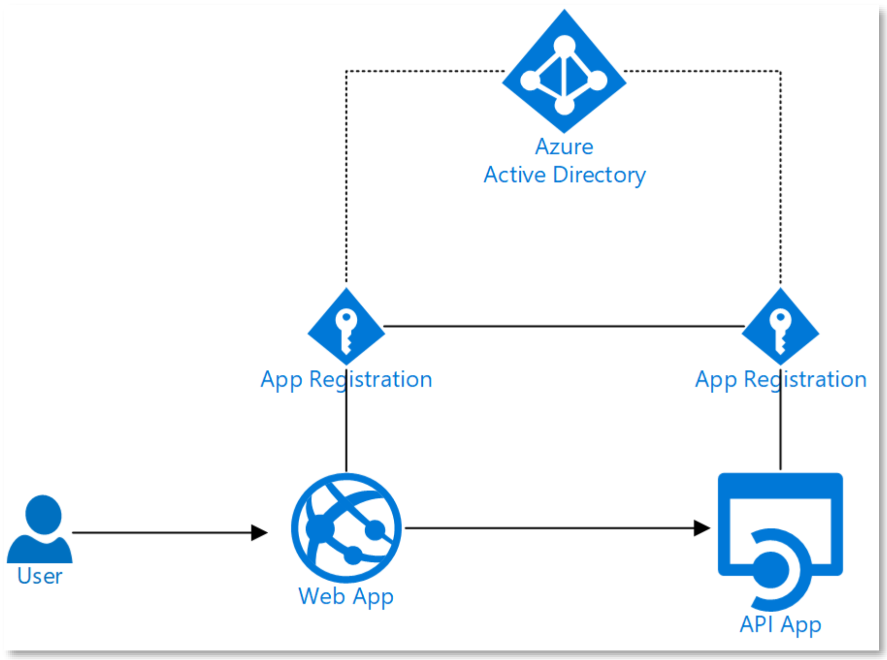


Figure 27 - Securing web API with AFDS

Source: Own application

Figure 27 is showing how the *Web API* is being secured by the *AFDS* process.

When the user accesses the webpage / app and already has a token, the request is being forwarded to the *Web API.*

If the user does not have a token, the *AFDS* process will commence.

4.4.2 Choice of database

Choosing the right database is very crucial for the performance in the solution. With a lot of different database structures to choose from, tests are being carried out, in order to find the best performing database.

A normalized model (3rd normal form) in a relational database would look as on ***Figure 28 - Report Data*** model**.** A report can have various Asset groups, and each asset groups can either have asset groups or report assets.

In this case we will need a report template that can change. And when the template is being changed, the data in old instances of reports should not change. Ex. A building has a report with 1 door as an entrance. A year after, the door has been changed to a double door, but the rest is the same.

In the first report the name should be “Entrance door”, and the second report “Entrance, double door”.

To do this in a relational database, it would be required to keep track of all the changes in the model names and instances, and store this in the database as well. To benefit the most from a relational database, it requires that all the data must follow the same structure and be stored in a very organized fashion, which makes working with our dynamic report templates a bit more complicated.

A none relational database on the other hand is based on a collection of documents, where every document consists of only two columns ('key' and 'value'), with more complex information sometimes stored as BLOBs within the 'value' columns. The data in each document is stored as JSON, and values are nested hierarchically. That means that a whole report can be stored in a document and it is not required to keep track of when names for the asset’s changes.

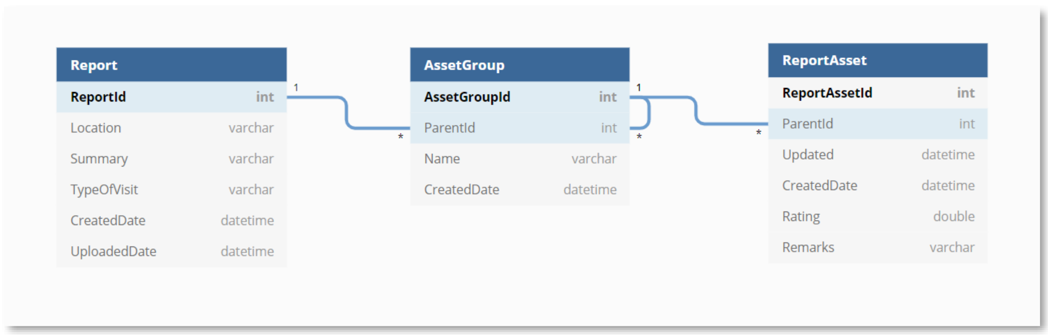


Figure 28 - Report Data model

Source: Own application

Based upon these facts a non-relational database is chosen for the project. In this case Microsoft Azure Cosmos DB is being used.

4.4.2.1.Project data model and tests

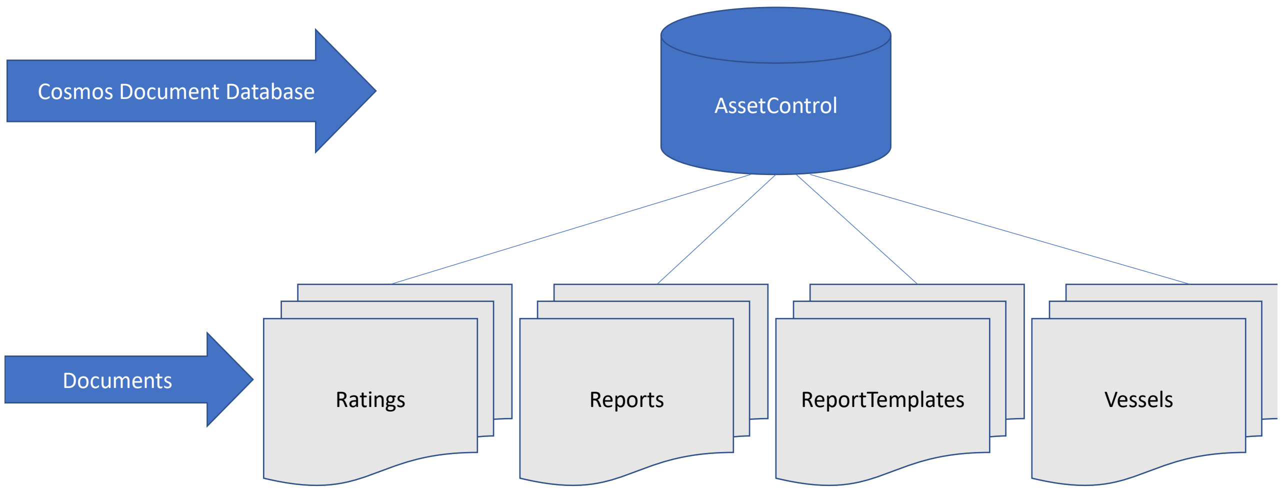


Figure 29 - Cosmos Database Layout

Source: Own application

Figure 29 - Cosmos Database Layout shows the database layout where the database "AssetsControl" has several containers underneath for storing data.

The "Vessels" container is where the report objects are stored. In this case it consists of documents with detailed information about each vessel.

The “ReportTemplates” container contains a document with the enabled report ´template edited in the administration portal. The reporting tool downloads the template from this container.

The “Reports” container contains all the uploaded reports.

The “Ratings” contains descriptions of the rating for the ships. Different ship types can have different ratings.

A report document withholds all information about the report. Images that are uploaded along with the report are stored in Azure Blob storage and a reference to the image is kept within the report document.

4.4.2.2 Limitations within Cosmos Database and the use of it

It is important to bear in mind that each document in a Cosmos Database container has a maximum size of 2 MB. In our use case, there are about 420 assets, that can be evaluated. The empty report template for these 420 assets gives a file size of 210 KB. With 10 images for each report Asset and a random text of 3219 characters in as a remark gives a total size of 2,05 MB.

It is most unlikely that in most cases these limits will not be exceeded. In the case that the limit will be exceeded the solution will be to change the data structure a bit, so a single report will be contained in several documents.

4.5 DEVELOPMENT OF THE REPORTING TOOL

Development of the reporting tool was done in Xamarin.Forms (Using Visual Studio 2017), which consist of various elements that allows development of applications for iOS and Android sharing the same Code-Base.

- **C#- Language** – Allows features like Generics, LINQ and the Task Parallel Library and syntax is similar to C# elsewhere.
- **Mono .NET framework** – Provides a cross-platform implementation of the extensive features in Microsoft’s .NET framework.
- **Compiler** – Depending on the platform, produces a native app (e.g. iOS) or an integrated .NET application and runtime (e.g. Android). The compiler also performs

many optimizations for mobile deployment such as linking away un-used code and libraries depending on the platform compiled to.

- **IDE tools** – The Visual Studio on Mac and Windows allows you to create, build, and deploy Xamarin projects.

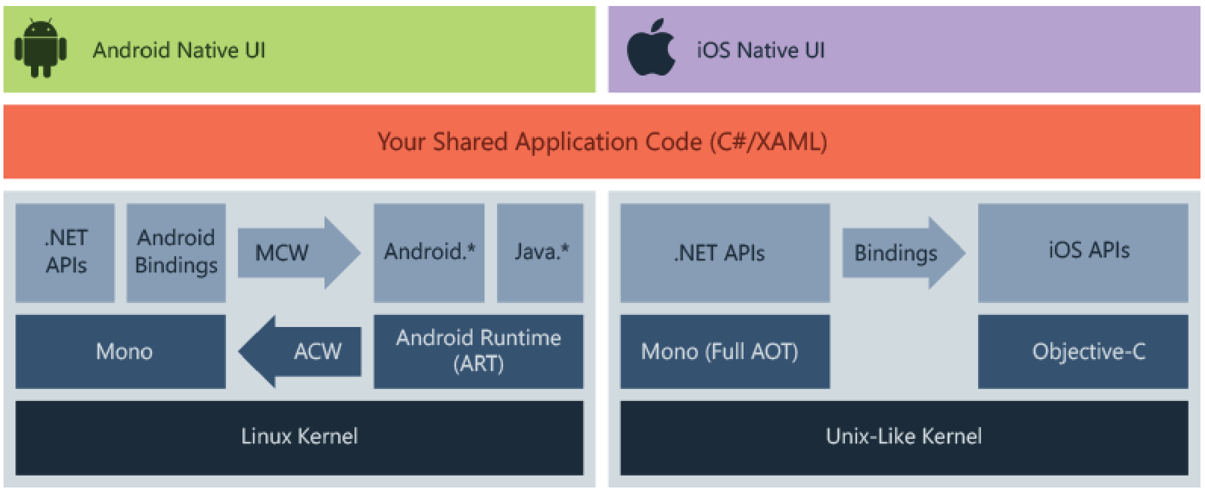


Figure 30 - Xamarin structure

Source: https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin

Using Xamarin one uses a single programming language and runtime that works on three mobile platforms. The C# code base will have access to all features that are available for native SDK's. Interfaces can be designed individually for each platform as seen on Figure 30 - Xamarin structure.

4.5.1 Structure

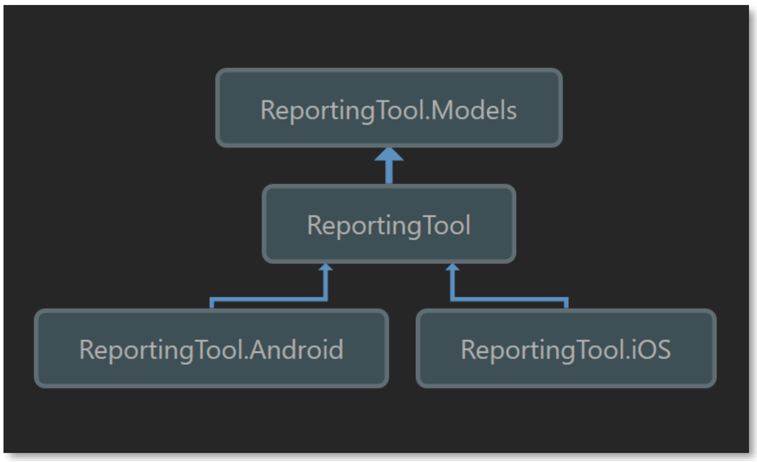


Figure 31 - Project dependency diagram

Source: Own application

The ReportingTool platform was projected to work only on Android and iOS why the structure is as shown on Figure 31 - Project dependency diagram**.**

“ReportingTool” is a .NET standard 2.0 library and is the Core Library in the project. This is where the business layer, service access layer and data access layer go. At compile time this will either be compiled to run on iOS or on Android. This project is dependent on ReportingTool.Models where all the data models are stored.

The ReportingTool.Android and ReportingTool.iOS is where specific code for Android or iOS goes, where necessary. That can for example be a navigation page

A navigation page in Xamarin Forms, is at compile time being compiled into a **ViewController** in iOS and into an **Activity** in Android. If there are any device specific customizations it must be handled in the specific target project, since the code to any customization is different.

4.5.1.1 Compiling to Android Devices

Code of C# is compiled to intermediate language with a package of MonoVM + JIT. If there are any unused classes of framework then it is stripped out when linking. The application runs, interact with the java native types through JNI (Java Native Invoke).

4.5.1.2 Compiling to iOS devices

C# source code is compiled using Xamarin.iOS. The Mono framework allows access to iOS platform specific features. The Xamarin.iOS compiler compiles source code into an intermediate language that is known as ECMA CIL (common intermediate language).

After compilation of Xamarin.iOS application code into CIL there is needs to compile it again into native machine code that can run on an iOS device.

4.8 DEVELOPMENT OF THE CLIENT/ADMINISTRATION PORTAL

Development of the client and administration portal was carried using Angular CLI and Typescript. There is no specific reason for choosing Angular other than trying something new.

4.7 DEVELOPMENT OF A BACK-END API

The Reporting tool mentioned in

4.5 DEVELOPMENT OF THE REPORTING TOOL and the client/administration portal mentioned in ***4.8 DEVELOPMENT OF THE CLIENT/ADMINISTRATION*** PORTAL are all communicating with a back-end API in order to deliver/retrieve data from the database.

The backend API was developed as a REST-API, based on Microsoft .NET Core 2.1. All methods are secured as described in the

4.4.1.Security section. A big advantage of making a separate REST-API is that front end and back end is separated, which makes it very easy to construct a new front-end platform and after just “connect” it to the backend. Other advantages to this approach:

- Authentication

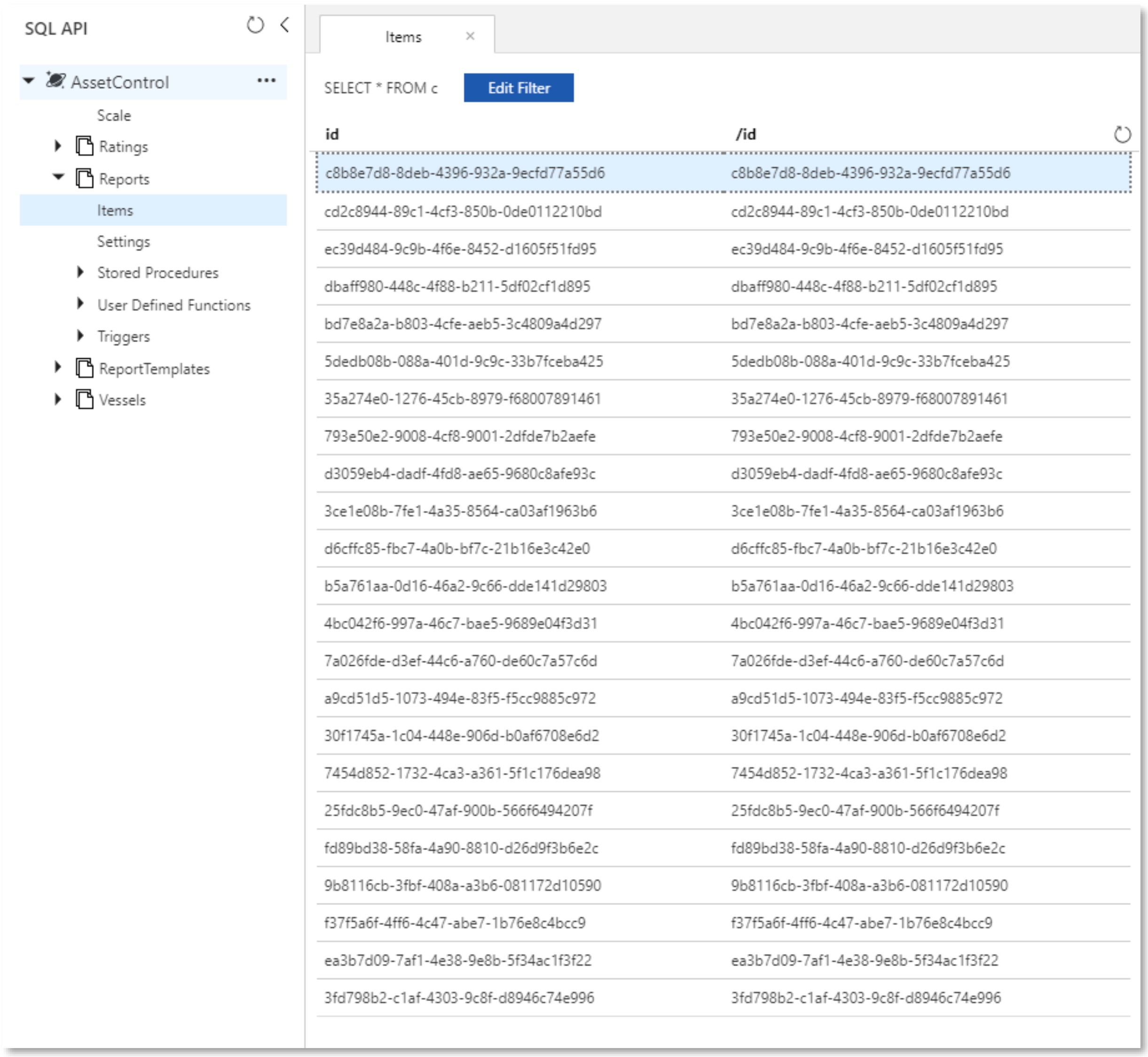
    - If a client needs to access data from multiple devices, it only must be authenticated once at the gateway. This reduces latency and ensures authentication process are consistent across the application e.g. ReportingTool and portals.
- Input Validation
    - The API itself can perform simple logic as well. The API can ensure that the request contains all the necessary information in order to complete the request before it reaches the services behind.
- Metrics collection
    - When all the requests are funneled through the API, it is the ideal place to collect analytics information such as:
        - Where the user is connection from
        - How many times a day a user is connecting
        - What service behind is being called the most
- Response Transformation
    - When having different devices and users needing different information, the API can be used to effectively present a unique API to each client type. For example, does it make sense to differentiate calls from the ReportingTool and the Portals since they usually are called from either mobile device or desktop devices.

5. RESULTS

This project with the Vessels use case is being tested live by the client. As being the world's leading product tanker company carrying cargoes such as naphtha, gasoline and jet oil, this is not a process that is easily done.

Each inspection engineer has been equipped with a device with the reporting tool installed on, which provides data to the database. Inspecting the data as shown in the Cosmos Database, it is possible to see that the amount of data is increasing as seen on

Figure 32 - Reports in production**.**



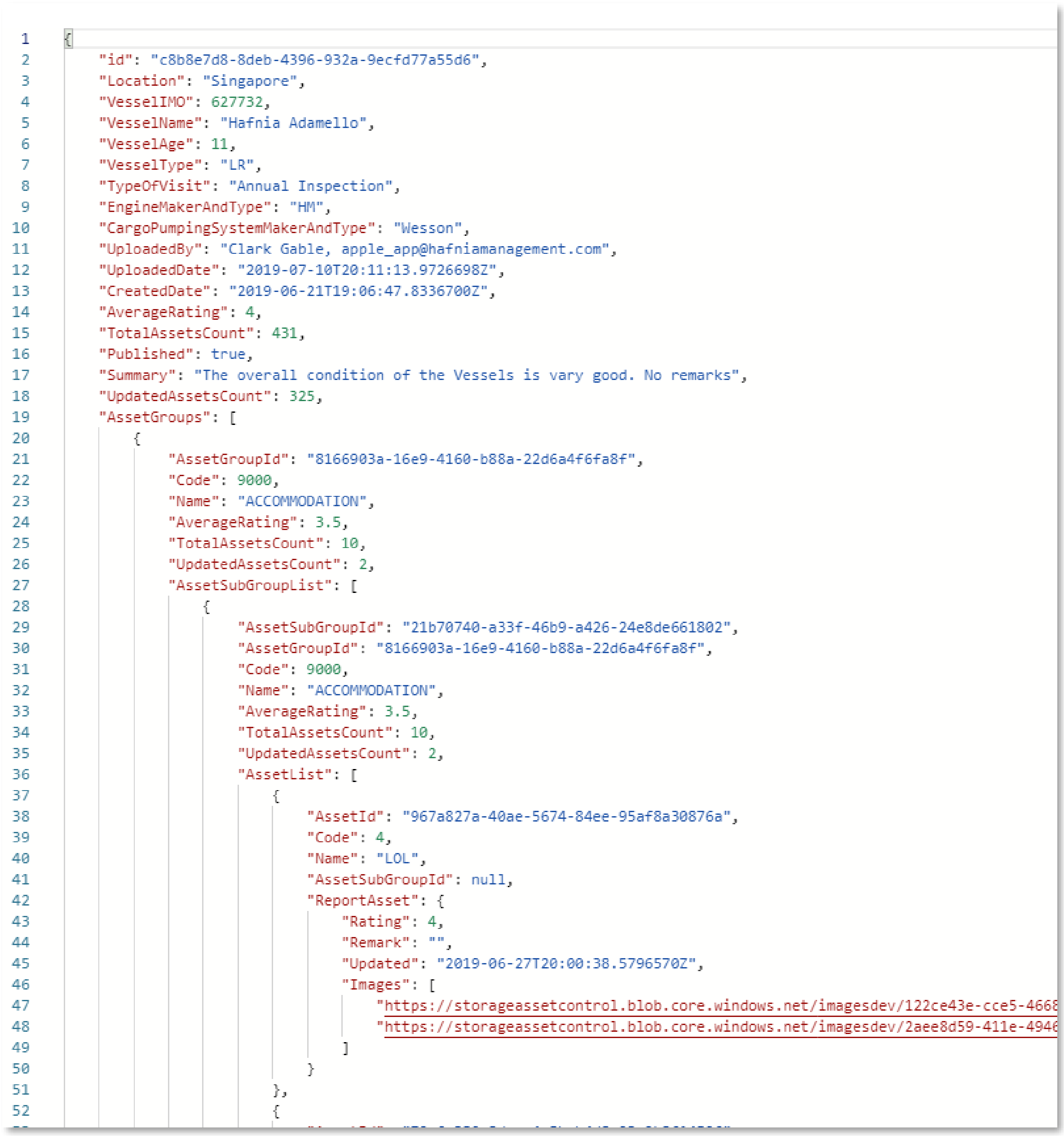
```
SELECT * FROM c
```

id	/id
c8b8e7d8-8deb-4396-932a-9ecfd77a55d6	c8b8e7d8-8deb-4396-932a-9ecfd77a55d6
cd2c8944-89c1-4cf3-850b-0de0112210bd	cd2c8944-89c1-4cf3-850b-0de0112210bd
ec39d484-9c9b-4f6e-8452-d1605f51fd95	ec39d484-9c9b-4f6e-8452-d1605f51fd95
dbaff980-448c-4f88-b211-5df02cf1d895	dbaff980-448c-4f88-b211-5df02cf1d895
bd7e8a2a-b803-4cfe-aeb5-3c4809a4d297	bd7e8a2a-b803-4cfe-aeb5-3c4809a4d297
5dedb08b-088a-401d-9c9c-33b7fceba425	5dedb08b-088a-401d-9c9c-33b7fceba425
35a274e0-1276-45cb-8979-f68007891461	35a274e0-1276-45cb-8979-f68007891461
793e50e2-9008-4cf8-9001-2dfde7b2aefe	793e50e2-9008-4cf8-9001-2dfde7b2aefe
d3059eb4-dadf-4fd8-ae65-9680c8afe93c	d3059eb4-dadf-4fd8-ae65-9680c8afe93c
3ce1e08b-7fe1-4a35-8564-ca03af1963b6	3ce1e08b-7fe1-4a35-8564-ca03af1963b6
d6cffc85-fbc7-4a0b-bf7c-21b16e3c42e0	d6cffc85-fbc7-4a0b-bf7c-21b16e3c42e0
b5a761aa-0d16-46a2-9c66-dde141d29803	b5a761aa-0d16-46a2-9c66-dde141d29803
4bc042f6-997a-46c7-bae5-9689e04f3d31	4bc042f6-997a-46c7-bae5-9689e04f3d31
7a026fde-d3ef-44c6-a760-de60c7a57c6d	7a026fde-d3ef-44c6-a760-de60c7a57c6d
a9cd51d5-1073-494e-83f5-f5cc9885c972	a9cd51d5-1073-494e-83f5-f5cc9885c972
30f1745a-1c04-448e-906d-b0af6708e6d2	30f1745a-1c04-448e-906d-b0af6708e6d2
7454d852-1732-4ca3-a361-5f1c176dea98	7454d852-1732-4ca3-a361-5f1c176dea98
25fdc8b5-9ec0-47af-900b-566f6494207f	25fdc8b5-9ec0-47af-900b-566f6494207f
fd89bd38-58fa-4a90-8810-d26d9f3b6e2c	fd89bd38-58fa-4a90-8810-d26d9f3b6e2c
9b8116cb-3fbf-408a-a3b6-081172d10590	9b8116cb-3fbf-408a-a3b6-081172d10590
f37f5a6f-4ff6-4c47-abe7-1b76e8c4bcc9	f37f5a6f-4ff6-4c47-abe7-1b76e8c4bcc9
ea3b7d09-7af1-4e38-9e8b-5f34ac1f3f22	ea3b7d09-7af1-4e38-9e8b-5f34ac1f3f22
3fd798b2-c1af-4303-9c8f-d8946c74e996	3fd798b2-c1af-4303-9c8f-d8946c74e996

Figure 32 - Reports in production

Source: Own application

Each line in the ‘Items’ column is a report and has all the information related to the report, which can be seen on ***Figure 33 - Report details in Production*** DB.



```
{
    "id": "c8b8e7d8-8deb-4396-932a-9ecfd77a55d6",
    "Location": "Singapore",
    "VesselIMO": 627732,
    "VesselName": "Hafnia Adamello",
    "VesselAge": 11,
    "VesselType": "LR",
    "TypeOfVisit": "Annual Inspection",
    "EngineMakerAndType": "HM",
    "CargoPumpingSystemMakerAndType": "Wesson",
    "UploadedBy": "Clark Gable, apple_app@hafniamanagement.com",
    "UploadedDate": "2019-07-10T20:11:13.9726698Z",
    "CreatedDate": "2019-06-21T19:06:47.8336700Z",
    "AverageRating": 4,
    "TotalAssetsCount": 431,
    "Published": true,
    "Summary": "The overall condition of the Vessels is vary good. No remarks",
    "UpdatedAssetsCount": 325,
    "AssetGroups": [
        {
            "AssetGroupId": "8166903a-16e9-4160-b88a-22d6a4f6fa8f",
            "Code": 9000,
            "Name": "ACCOMMODATION",
            "AverageRating": 3.5,
            "TotalAssetsCount": 10,
            "UpdatedAssetsCount": 2,
            "AssetSubGroupList": [
                {
                    "AssetSubGroupId": "21b70740-a33f-46b9-a426-24e8de661802",
                    "AssetGroupId": "8166903a-16e9-4160-b88a-22d6a4f6fa8f",
                    "Code": 9000,
                    "Name": "ACCOMMODATION",
                    "AverageRating": 3.5,
                    "TotalAssetsCount": 10,
                    "UpdatedAssetsCount": 2,
                    "AssetList": [
                        {
                            "AssetId": "967a827a-40ae-5674-84ee-95af8a30876a",
                            "Code": 4,
                            "Name": "LOL",
                            "AssetSubGroupId": null,
                            "ReportAsset": {
                                "Rating": 4,
                                "Remark": "",
                                "Updated": "2019-06-27T20:00:38.5796570Z",
                                "Images": [
                                    "https://storageassetcontrol.blob.core.windows.net/imagesdev/122ce43e-cce5-4668
                                    "https://storageassetcontrol.blob.core.windows.net/imagesdev/2aee8d59-411e-4946
                                ]
                            }
                        },
                        {
```

Figure 33 - Report details in Production DB

Source: Own application

This is the result of the reporting tool idea implemented for use in the shipping business. With minor adjustments it can be implemented in any business to do the reporting of any object.

REFERENCES

Chaudhuri Surajit , Narasayya Vivek and Dayal Meshwar An Overview of Business Intelligence Technology [Online] // Communications ACM. - Communications of the ACM, 2011. - 12 13, 2019. - https://cacm.acm.org/magazines/2011/8/114953-an-overview-of-business-intelligence-technology/fulltext.

COPRPORATION BUILDING INFORMATION MANAGEMENT SYSTEMS PROPERTY MANAGEMENT SYSTEM AND METHOD THEREOF [Patent]. - NY, USA, 09 20, 2018.

Hansen Michael S. [Online] // blogs.msdn.microsoft.com. - march 25, 2018. - 04 02, 2019. - https://blogs.msdn.microsoft.com/mihansen/2018/03/25/azure-active-directory-authentication-easy-auth-with-custom-backend-web-api/.

Maxwell, Locke & Ritter USE BUSINESS INTELLIGENCE TO MAKE PROFITABLE DECISIONS [Online]. - Maxwell Locke & Ritter, 11 02, 2017. - 12 13, 2019. - https://www.mlrpc.com/articles/use-business-intelligence-make-profitable-decisions/.

Microsoft Active Directory Domain Services Overview [Online]. - 05 30, 2017. - 12 13, 2019. - https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview.

Microsoft Active Directory Federation Services [Online] // Microsoft Docs. - Microsoft, 05 30, 2017. - 12 13, 2019. - https://docs.microsoft.com/en-us/windows-server/identity/active-directory-federation-services.

Microsoft Integrated Windows Authentication [Online]. - 12 17, 2012. - 12 13, 2019. - https://docs.microsoft.com/en-us/aspnet/web-api/overview/security/integrated-windows-authentication.

GLOSSARY

AD

Active Directory (AD) is a directory service developed by Microsoft for Windows domain networks. It is included in most Windows Server operating systems as a set of processes and services. Initially, Active Directory was only in charge of centralized domain management. Starting with Windows Server 2008, however, Active Directory became an umbrella title for a broad range of directory-based identity-related services.

ADFS

Active Directory Federation Services (ADFS) is a Single-sign on (**Error! Reference source not found.**) solution created by Microsoft. As a component of Windows Server operating systems, it provides users with authenticated access to applications that are not capable of using Integrated Windows Authentication (IWA) (Microsoft, 2012) through Active Directory (AD) (Microsoft, 2017).

ADFS gives organizations the ability to control their employees' accounts while simplifying the user experience: employees only need to remember a single set of credentials to access multiple applications through **Error! Reference source not found.**.

Android Activity

The Activity class takes care of creating a window where in which the UI can be placed. Activities are often being presented as full screen windows, but can also be presented as floating windows / embedded windows etc.

Azure Blob Storage

Azure Blob Storage is a service for storing large amounts of unstructured object data, such as text or binary data. You can use Blob Storage to expose data publicly to the world, or to store application data privately. Common uses of Blob Storage include:

1. Serving images or documents directly to a browser
2. Storing files for distributed access
3. Streaming video and audio
4. Storing data for backup and restore, disaster recovery, and archiving
5. Storing data for analysis by an on-premises or Azure-hosted service

Dependency Injection

Dependency injection is a design pattern that helps developing loosely coupled code. The purpose of developing loosely coupled code is to make code maintainable and easy to update.

A dependency or dependencies are created outside the class that uses it/them. The dependency / dependencies is/are injected from the code that calls the class any information about the instantiation is kept away from the class itself.

Generics

Generics were added to version 2.0 of the C# language and the common language runtime (CLR). Generics introduce to the .NET Framework the concept of type parameters, which make it possible to design classes and methods that defer the specification of one or more types until the class or method is declared and instantiated by client code. For example, by using a generic type parameter T you can write a single class that other client code can use without incurring the cost or risk of runtime casts or boxing operations.

LINQ

Language-Integrated Query (LINQ) is the name for a set of technologies based on the integration of query capabilities directly into the C# language. Traditionally, queries against data are expressed as simple strings without type checking at compile time or IntelliSense support. Furthermore, one must learn a different query language for each type of data source: SQL databases, XML documents, various Web services, and so on. With LINQ, a query is a first-class language construct, just like classes, methods, events.

For a developer who writes queries, the most visible "language-integrated" part of LINQ is the query expression. Query expressions are written in a declarative query syntax. By using query syntax, you can perform filtering, ordering, and grouping operations on data sources with a minimum of code. You use the same basic query expression patterns to query and transform data in SQL databases, ADO .NET Datasets, XML documents and streams, and .NET collections.

OAUTH

OAuth is an open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords. This mechanism is used by companies such as Amazon, Google, Facebook, Microsoft and Twitter to permit the users to share information about their accounts with third party applications or websites.

RootViewController

The RootViewController is the first ViewController on the Application Stack on iOS The RootViewController provides the content view of the window.

SPA

A single-page application (SPA) is a web application or web site that interacts with the user by dynamically rewriting the current page rather than loading entire new pages from a server. This approach avoids interruption of the user experience between successive pages, making the application behave more like a desktop application. In an SPA, either all necessary code – HTML, JavaScript, and CSS – is retrieved with a single page load, or the appropriate resources are dynamically loaded and added to the page as necessary, usually in response to user actions. The page does not reload at any point in the process, nor does control transfer to another page, although the location hash or the HTML5 History API can be used to provide the perception and navigability of separate logical pages in the application. Interaction with the single-page application often involves dynamic communication with the web server behind the scenes.

SQL

Structured Query Language is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). It is particularly useful in handling structured data, i.e. data incorporating relations among entities and variables.

SSO

Single Sign On (SSO) is a process that permits a user to access multiple services after going through user authentication (i.e. logging in) only once. This involves authentication into all services the user has given permission to, after logging into a primary service. Among other benefits, SSO avoids the monotonous task of confirming identity repeatedly through passwords or other authentication systems.

Task Parallel Library (TPL)

The Task Parallel Library (TPL) is based on the concept of a task, which represents an asynchronous operation. In some ways, a task resembles a thread or Thread Pool work item, but at a higher level of abstraction. The term task parallelism refers to one or more independent tasks running concurrently. Tasks provide two primary benefits:

1. More efficient and more scalable use of system resources.

Behind the scenes, tasks are queued to thread pool which has been enhanced with algorithms that determine and adjust to the number of threads and that provide load balancing to maximize throughput. This makes tasks relatively lightweight, and you can create many of them to enable fine-grained parallelism.

2. More programmatic control than is possible with a thread or work item.

Tasks and the framework built around them provide a rich set of APIs that support waiting, cancellation, continuations, robust exception handling, detailed status, custom scheduling, and more.

For both reasons, in the .NET Framework, TPL is the preferred API for writing multi-threaded, asynchronous, and parallel code.