



Blockchain in practice

a proof of concept



Institutt for Informasjonsteknologi
Postadresse: Postboks 4 St. Olavs plass, 0130 Oslo
Besøksadresse: Holbergs plass, Oslo

PROSJEKT NR.
18-25

TILGJENGELIGHET
Offentlig

Telefon: 22 45 32 00

BACHELORPROSJEKT

HOVEDPROSJEKTETS TITTEL Blockchain in practice	DATO 23.05.2018
	ANTALL SIDER / BILAG 188/71
PROSJEKTDeltakere Bjørn Goldberg – s234852 Lars Magnus Henriksen – s305481 Thomas Haugen Guest – s305457 Kim Finnanger Knudsen – s305487 Thomas Langseth - s305469	INTERN VEILEDER Boning Feng
OPPDRAAGSGIVER IBM	KONTAKTPERSON Olga Kravchenko

SAMMENDRAG

Regjeringen ønsker en offentlig digital plattform for innsyn i aksjeregisteret.
Løsningen skal gjøre det enklere for aksjeselskap å oppdatere aksjeeierbøkene sine.

I samarbeid med IBM og Altinn har vi utviklet et proof of concept for digitalisering
av aksjeregisteret ved hjelp av blockchain-teknologi.

3 STIKKORD

Blockchain

Hyperledger

Aksjeeierbok

Preface

This report is made with the intention of documenting the work we did, and the product we developed at Oslo Metropolitan University in the spring of 2018, and will serve as our final bachelor thesis.

The report itself is constructed of several parts which is meant to give the reader insight and understanding of the process behind our work and the systems which has been developed. Although we explain most technical terms, it will be very helpful for our readers to have minor knowledge about technical systems.

In regard to this project we would like to thank our employer, IBM and Olga Kravchenko and Lars Fanebost at IBM for their support, assistance and feedback.

We would also like to thank our faculty guidance counselor, Boning Feng, for his feedback, support and ideas.

Guidance counselor at OsloMet:

Boning Feng, Associate Professor
E-mail: boning.feng@oslomet.no

Contact person at IBM:

Olga Kravchenko, Cognitive consultant
E-mail: olga.kravchenko@no.ibm.com

Any questions regarding this report or the project in its entirety, please do not hesitate to send us an email at ttlbk.bachelorgruppe@gmail.com.

Contents

1	Introduction	1
1.1	Employer	1
1.2	About us	1
1.3	Project context	2
1.4	Presenting the use case and the persona	3
1.5	System description	4
2	Problem statement	6
2.1	Business challenge	6
2.2	Product specification	6
	Functional requirements	7
	Non-functional requirements	8
2.3	Constraints	8
3	Development process	9
3.1	Establishing development environment	9
3.2	Use cases	9
3.3	Specifying requirements	9
3.4	Defining business model	10
3.5	Agile Development methodology	11
	Scrum	11
3.6	Showcase/Sprint review	13
3.7	Retrospective	13
3.8	Planing and method	14
3.9	Going forward with Flowcharts	15
3.10	Wireframes	18
	Low-fidelity	18
	High-fidelity	18
	Our Project	19
	Wireframes for company landing page	19
	Wireframes for company actions	20
	Wireframes for company dashboard	20
3.11	Software and tools	21
3.12	Timetable and progress plan	21
3.13	Work log	23

4	Product documentation	24
4.1	Architecture	24
	Blockchain	24
	Architecture model	25
	Development tools	26
	Development environment	27
4.2	Hyperledger Fabric	28
	Peer nodes	28
	Orderer node	28
	Ca node - Membership Service Provider	28
	CouchDB	28
	Creating the ledger	29
	Docker composer	29
	Fabric tools	29
	Fabric summary	29
4.3	Hyperledger Composer	30
	Composer explained	30
	SDK and API	30
	Run time and connection-profiles	31
	Business network cards	32
	Business Model	33
	Transaction Logic and chaincode	35
	ACL	36
	Other development tools	37
	Altinn Network - Our product	38
	Shareholder registry business model	38
	The smart-contract/chaincode of Shareholder registry	39
	Access control	43
	REST-server	44
	Build	44
4.4	Frontend	45
	Angular applications	45
	Brønnøysundregistrene client	46
	Public web client	51
	Business client	53
	Stock Owner client	59
	Journalist client	67
	Public client	72

GUI	73
Navigation	75
Design components	77
CSS/SASS	81
5 Testing	82
5.1 User Testing	82
The testing process	82
Contract	83
Pre-user test questionnaire	84
User tests and Observational Data	85
Post-user test questionnaire	87
Method	88
Selection	89
Quantitative research	89
Qualitative research	89
Results	90
Conclusion	92
Gathered user-test information	93
5.2 Unit test	93
Testing tools for Hyperledger Composer	93
Mocha	93
Chai	94
Test setup	94
Connection profile	94
Creating a peerAdmin	94
Deploying the business network	95
Seeding the business network	96
Changing identity	98
Unit test execution	98
Arrange	98
Act	99
Assert	99
Executed tests	100
Challenges	101
5.3 System test	102
Listing of functionality that has been tested	103
Test cases step-by-step	104

Results system testing	104
6 Final result and evaluation	105
6.1 Process	105
Agile methodology	105
Sticking to the plan	105
Distributing tasks and responsibility	106
Playbacks	106
Retrospectives	107
Dealing with new technology	107
Uncertain documentation	108
Frequent updates	108
6.2 Product	109
Functional requirements	109
Non-functional requirements	110
6.3 Evaluation	110
7 Future work	111
7.1 User authorization and management frontend	111
7.2 More UX design	111
7.3 Potential features	111
8 Summary	113
9 References	114
10 Appendix	117
10.1 Worklog	117
10.2 Flowcharts and wireframes	150
10.3 Results user testing	160
10.4 Step-by-step guide system tests	168
10.5 Step-by-step guide unit tests	172

1 Introduction

In our bachelor thesis we have developed a solution for a digital register of shareholders by using blockchain technology. The purpose being exploring the value blockchain may offer this field.

1.1 Employer

Our employer for this project has been the well known tech-company IBM (International Business Machines). IBM is one of the leading tech-companies in the world and has been pushing the technological limits with achieving the most patents connected to technology each year, ever since they were founded in 1911 (Om IBM, 2018). They started with creating and selling store scales, time recording devices and tabulators, and have moved onto working with technology such as Blockchain and Artificial Intelligence (Watson)(Rouse, 2016). The technology has changed massively, but IBM is still doing the same; working with the newest and most relevant technology, pushing limits and widening the technological horizon for the benefit of the world.

In this project we teamed up with Olga Kravchenko and Lars Fanebost. Olga is IBM Norway's own Blockchain spokesperson as well as a cognitive consultant who works mainly with Blockchain and Watson. Lars is a well experienced Master IT Architect and has been working for leading companies such as IBM and Bring. We have worked closely together in order to create a product based on Blockchain technology, towards one of their customers (Altinn/Brønnøysundregistrene). Both Olga and Lars have provided us with their unique experience, excellent guidance and both technological and project oriented knowledge throughout the entire process.

1.2 About us

We are a group consisting of five students from Applied Computer Technology at Oslo Metropolitan University (OsloMet). We have known each other from the first year at university, and have for the last two years collaborated and worked together on group projects. The degree has included both technical and theoretical courses. The technical courses has taught us how to code using different technologies and principles, within

both frontend and backend. The theoretical aspect of the degree has equipped us with knowledge and understanding regarding development of prototypes, testing, IoT (Internet of things), as well as how to structure and organize information, both on websites and in databases.

Name	Email
Bjørn Golberg	s234852@oslomet.no
Thomas Langseth	s305469@oslomet.no
Thomas Guest	s305457@oslomet.no
Kim Knudsen	s305487@oslomet.no
Lars Magnus Henriksen	s305481@oslomet.no

1.3 Project context

Brønnøysundregistrene and Altinn are working on solutions to allow more openness and accessibility to information within stock-companies register of shareholders. This is based on the Norwegian parliament's resolution in Prop. 94 LS (Regjeringen, 2013-2014). "The Norwegian parliament requests the government to establish a public solution with information on stock-companies owners that will ensure more accessibility".

Based on this, Brønnøysundregistrene and Altinn have been looking for a digital solution, and decided to work together with IBM to create an appropriate solution for the problem at hand. A part of the process has been to decide and figure out what technology that would solve the issue in the best possible manner for both Brønnøysundregistrene and Altinn, as well as for the users.

In the process of deciding what technology this solution should use, the collaborators are prospecting the possibility to make this a blockchain solution.

As a part of this research we as students were asked to develop a blockchain solution with a relevant use-case by using IBM's Hyperledger tools (Hyperledger Fabrics and Hyperledger Composer). By this the goal is to demonstrate the functionality and thus the possibilities by using blockchain

to establish a public digital solution for making register of shareholders information accessible for the different types of participants.

1.4 Presenting the use case and the persona

As a part of the project commencement, it was important to create clear and suitable use cases in order to properly define what the users will be doing in the solution. It also helped towards clarifying how the different functionality and requirements should be implemented. Furthermore, use cases can be viewed as a way of recognizing essential requirements, depending on different roles or actors in the use case (IBM Knowledge center, 2018).

A "Persona" is a hypothetical person that represents the end user. It should be a realistic representation of a certain user with specific tasks to complete, rather than "an average" user. This will help the developers understand the user's situation and needs, and ultimately guide the developers in regards of how the necessary functionality should be implemented. Furthermore, having a "persona" can reveal necessary functionality as well as inspire how to make the solution fit the user in the best way possible.

The persona for this project is 59 year old Torbjørn Rør Isaksen from Holmestrand. He is a plumber and owns his own plumber-company, as well as being a local politician. He is divorced and has four children. He enjoys sports, especially cross-country skiing. He has sold stock shares from his firm to his ex-wife, and issued stock shares to his employees in a difficult period so that they would work for less money in return. Torbjørn dreams of simplicity, his retirement and to focus on what he should and wish to focus on; work, instead of spending time on administrative things.

After introducing Torbjørn as our persona, it gives us an idea of who he is and what needs he has. In order to create a collective understanding within the team of who this individual is and how we should prioritize his needs, we have used an empathy map. This is a collaborative visualization of what the user "says", "thinks", "does" and "feels". Having information within these topics will contribute to decision-making situations, and ultimately paints a clear picture of the user's needs (Gibbons, 2018).

Trying to get to know Torbjørn better, it was essential to understand what a typical day for him looked like, what he does, what he thinks, feels and says. The following figure is an empathy map with information regarding the mentioned topics, in Torbjørn's case:

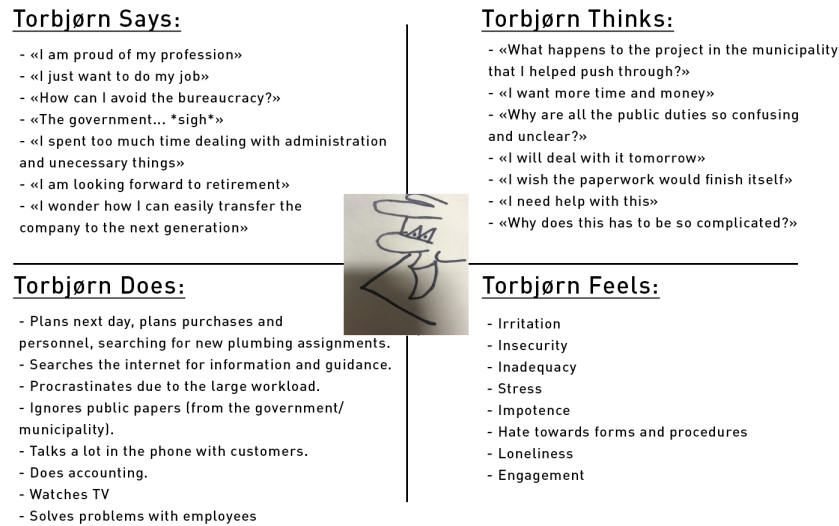


Figure 1: Empathy map of case persona

Based on the information that the team gathered, it was concluded that two essential needs for Torbjørn were the following:

1. Torbjørn needs a way to register his shareholder information only once, so that he can focus on operating the firm.
2. Torbjørn needs a way to confirm that he has provided sufficient information, so that he can relax and sleep at night.

1.5 System description

"Aksjeeierbok" is a system divided into three main parts. Backend with data storage on blockchain, business logic and a REST API. One frontend web application dedicated to Brønnøysundregistrene and one frontend web application open to the public.

The aim of our system is to ease access and insight into different company's

register of shareholders as well as give Brønnøysundregistrene an easier task of responding to different application forms submitted by businesses. The system does also allow private users to trade stocks between each other.

2 Problem statement

In an increasingly digital world, the old way of managing a register of shareholders is quickly becoming outdated and impractical for the users. In 2014 the Norwegian government issued a official mandate to Brønnøysundregistrene to work on forming an official report (Brønnøysundregistrene, 2014) on how the register of shareholders could be digitized. As well as providing a broad sense of transparency in a previously hard to navigate terrain, it would also be giving end users the ability to register and manage their business and/or private portfolios. There is a clear consensus between the different parties involved that there is a better solution out there, but it is not clear on how it would be structured or on what platform it would run. As a direct response to the mandate, we are developing an entry level suggestion on how such a solution could work. Our project could also serve as a proof of concept in future development.

2.1 Business challenge

It is desirable to have an efficient method to register a company and change that information relatively easily online. And then go back to daily business well assured that the process will take care of it.

The problem today is that there isn't a fully digital/automated process for this. This is because the authority's lack of APIs and efficient methods to authenticate the applied information.

2.2 Product specification

This section covers the concrete product requirements. It contains a list of functional and non-functional requirements for the finished product. The functional requirements covers the specific functionality that needs to be implemented for the product to be deemed complete. The non-functional list covers requirements that is not necessarily tied to any specific functionality, but covers higher level and external factors such as system performance, user efficiency and scalability. We divide the non-functional requirements into three main groups (Sommerville, 2010): product requirements, organizational requirements and external requirements.

Functional requirements

Stock owner		
ID	Prio.	Definition
1	A	Allow stock owners to list all their own stocks
2	B	Allow stock owners to see available stocks on the market
3	A	Allow stock owners to approve/decline incoming trade suggestions
4	C	Allow stock owners to see their transaction history
Business		
ID	Prio.	Definition
5	A	Allow business owners to register their company
6	C	Allow business owners to see their transaction history
7	B	Allow business owners to request a capital change
8	B	Allow business owners to request a numeric increase in stock
Brønnøysundregistrene (BR)		
ID	Prio.	Definition
9	A	Allow BR to approve/decline business registration
Journalist		
ID	Prio.	Definition
10	A	Allow journalists to have insights on all trades
11	A	Allow journalists to have insights on all requests

Figure 2: Functional requirements

Non-functional requirements

Product requirements		
ID	Prio.	Definition
1	A	All data should be stored and handled in a secure way.
2	A	All access should be authorized and identifiable.
3	C	The system scales acceptably on computers.
4	A	The system should meet the design requirements set by our client.
5	B	All errors should be handled in a manner that the user will understand.
6	C	The user should not face any unpredictable or inconsistent behaviour based on his or her input.
Organizational requirements		
ID	Prio.	Definition
7	B	The product shall meet the requirements set at the start of the development by our client when the deadline is met.
External requirements		
ID	Prio.	Definition
8	B	All Norwegian laws regarding privacy should be maintained.
9	C	All Norwegian standards concerning web development should be maintained. (Difi, 2014)

Figure 3: Non-functional requirements

2.3 Constraints

Although we did not actively participate in establishing use-cases, our chosen use case was selected out of a pool with two others. Our rather short time line and primordial technical knowledge with relevant technologies (blockchain, Hyperledger, etc), together with the projected profit and level of importance for our client became factors that eliminated the other two use cases.

3 Development process

In this section we will cover the development process, which includes planning and method, timetable and a in-depth look into our agile method of working. In this section you will also find flowcharts and wireframes, detailing our product.

3.1 Establishing development environment

We spent a couple of weeks (three days each week) at IBM to get an introduction and some guidance in the different tools we would be using while developing. We established a developing environment which we would continue to use during the dissertation course.

We were supposed to get VMs (virtual machines) running on IBM Cloud but at this point the VMs was not ready. Instead we were allocated a copy of a local VM to run with VMWare Player (local virtual machine tool) as a temporarily solution.

We installed the required software and tools to start building blockchain solutions with Hyperledger Composer. Furthermore we developed a small car insurance solution for practice. This made us familiar with the environment.

Later on when the dissertation requirements were defined we established an adapted environment as well as a git repository.

3.2 Use cases

The responsible from IBM, Altinn and Brønnøysundregistrene had a meeting to decide what they wanted from the assignment. Furthermore they developed the needed use cases. We didn't participate in this process.

3.3 Specifying requirements

After the use cases were agreed upon we joined IBM and Altinn in a new meeting to define the requirements. Both functional and non functional

requirements.

The participants from Altinn and Brønnøysundregistrene was not the same people as in the use-case meeting. This resulted in minor confusion of what the use-cases should be.

Our impression was that the participant from Brønnøysundregistrene wanted a bigger use-case(s) than first agreed upon. Our supervisor from IBM established the idea that the use-cases had to fit the framework of a bachelor dissertation. That being said it might be possible to extend the cases in the future hence that we are using agile methodology.

After the final agreement about use-cases we went about defining the requirements. We did this by having a brainstorm session consisting of all participants writing requirements on post-it notes. After the session everyone presented the requirements following a plural agreement or discarding of the requirements.

After all the requirements were finished we converted them into tasks and inserted them to Trello. Ongoing we estimated the time of each task and in what sequence they should be executed. We did this by distributing them equally into sprints.

At this point we had defined use cases, base functionality (functional and non functional), and sprints for the whole dissertation course. This would be the foundation execution and progression in the dissertation.

3.4 Defining business model

Based on the requirements and use cases we began to construct a business model. This would be the basis of the model we define in Hyperledger project and the Hyperledger composer use this to compose the blockchain fabric.

We tried to model it in a way that was both intuitive at a personal level as well as it being viable to develop.

3.5 Agile Development methodology

Our chosen development methodology is "Agile". Agile development are ways of development that are characterized by the relative short work/development cycles as well as the close collaboration between the different parts (developers and the customer). Each "cycle" is called a "Sprint" and can last from 1 to 4 weeks each. An important part of Agile development is the close communication internally in the scrum team, as well as between the team and the product owner. This is emphasized by the scrum life-cycle and structure, where daily stand-ups and sprint reviews together with product owner is a natural part of the process (Dan Hellem, 2017).

Scrum

Scrum is a way of working and managing the work at hand. It adopts the basic principles of Agile development and is therefore considered to be an agile methodology. Within the Scrum framework, you will find the following roles and entities:

Product Owner

Andreas Hammes from Brønnøysundregistrene has been this project's product owner. This role is typically held by one person who is the customer, or a representative of the customer. Andreas' responsibility has been to ensure that the scrum team work and prioritize things that maximizes the business value (Product Owner, 2018). Furthermore, in showcases his role has been to make sure that both functional, and non-functional requirements has been met. We would initially present what has been done throughout the most recent sprint, and he would then ask relevant questions regarding these areas to ensure that the important elements has been included and properly implemented. In cases where this were not done to his full satisfaction, we would get feedback on what needed to be done, or what should be done differently.

Product Backlog

The Product Backlog is both owned and kept up-to-date by the Product Owner Andreas Hammes on the behalf of Brønnøysundregistrene in this

project. It is a prioritized list of tasks that are important for the development and the final product. It is meant to help maximize the business value of the product. The backlog will also be the foundation for the scrum team to work from. This emphasizes the importance of a tidy and updated backlog as well as a product owner who is engaged and proactive.

Scrum Master

A Scrum master plays different roles in the process. This person works as someone who are responsible to ensure that the scrum team follows the plan everyone agreed on. IBM's representative and our contact person Olga, has been our Scrum master throughout the project. She has been the connection between the customer (Brønnøysundregistrene) and us (the scrum team). Furthermore, one of her responsibilities has been to keep distractions that might effect the work flow, away from the team, as well as coaching the team and keeping us focused. She has organized and scheduled showcases between the customer and us, and made sure that we as a team have been adequately prepared for each showcase. These tasks are typical responsibilities for a scrum master to make sure the development process happen in a proper manner.

Scrum Team

The Scrum team is the actual group of individuals who are developing the product. They are responsible for both the quality and the functionality of the product that is made (Gregg Boer, 2017). Something that characterizes a scrum team is that the team themselves chooses how to approach the project and delegate areas of responsibilities within the group. We have been able to choose what order to do things in as long as it has been in accordance of the sprint backlog. Working this way has also allowed us as a team to be flexible and respond instantly to changes and issues that has appeared along the way. The scrum team model is designed to do just that; increase the team's flexibility, productivity and creativity (Scrum.org, 2018). The liberty to delegate areas of responsibility between us, has resulted in a more reasonable distribution of focus and manpower, and therefore ensured a high level of production.

Sprint

Scrum is considered to be iterative as projects will be divided into smaller periods of time, that will be repeated over and over again, containing different kind of work and milestones. These periods are usually 1-4 weeks long and are called for Sprints. Each sprint will have been planned in "Sprint planning". This is where the team agrees on what they will be working on, based on the existing backlog items. They will also agree upon what should be completed for that period and estimate how much time that will be needed. (Gregg Boer, 2017)

Stand-up

Stand-ups are short daily meetings with all the team members, where each and every one gives the group a brief report of their standing regarding the sprint. In turns they will inform the group about their progress from the day before, any problems that might have occurred, and what their goal is for the day. That way everyone get to know whats going on in other parts of the project, who they need to talk to if they have questions about a certain thing, and also help each other.

3.6 Showcase/Sprint review

At the end of each sprint we have had meetings called "Showcase" or "Sprint review", together with the customer. In these meetings we have presented the progression, tasks we have done according to the sprint backlog. The meetings have allowed the customer to ask questions and get an understanding of how things work and are connected to each other. Getting this insight is important for the product owner to make sure that we complete the things we have agreed on, as well as how it has been implemented. Based on the information we as the scrum team has provided during these showcases, Andreas (Product owner) has been able to evaluate whether we have met the requirements, and/or if something should be prioritized differently in the process ahead.

3.7 Retrospective

The sprint retrospective takes place at the end of each sprint. During the retrospective the sprint team will reflect on the last sprint through asking

the following questions:

1. What went well during the sprint?
2. What could have been better?
3. What did not work?

The idea behind these questions is to make the group as a whole consider what contributed to the sprint in a positive and negative way, and constructively think about how to improve.

In our retrospective everyone wrote down things or areas of the last sprint on post-it notes. The post-it notes were divided into three categories; what was good, what could have been better, and what was not good. When everyone had done that we would organize the post-its into their respective section on the wall, and then talk about and discuss each and every one of them. As we went along we would group the notes that were similar to each other or within the same area. In the last part of the retrospective everyone would have four stickers each, that we would place on the ones we thought was the most important post-its under "what could have been better"-section. The notes with the most stickers would be our improvement focus for the next sprint. This allowed us to improve and work better as a group.

3.8 Planing and method

During this project we have used an agile software development structure. This means that we divided the work into several sprints. Each sprint lasted for two weeks and was structured around goals we worked towards for that period of time. After getting all the information from Altinn that we needed to start the project, we structured each sprint so that it would contain several milestones as well as estimated time we would spend on the tasks within that sprint. This helped us stay on track regarding doing what we were suppose to do, as well as doing so within reasonable time. To share and collectively organize each sprint we used Trello; an online tool that lets members share, view and organize the project.

Throughout a sprint we started each session with daily stand-ups. In stand-ups each member would tell the group about what he did the day

before, what had been challenging, what was achieved, and tell the group about any problems that might have occurred or been revealed through the process.

Each sprint would end with a showcase with Altinn to give them an update regarding the project. We would inform them about how we were doing, the progress, and if we ran into any problems or questions that we needed to clarify with them. They would get to ask questions about our progress or anything they would wonder about. We would then tell them what was next and what we would be doing the next couple of weeks.

3.9 Going forward with Flowcharts

Flowcharts are diagrams that are created through connecting different kinds of elements (shapes) to each other, to illustrate the flow through a certain process. The flow order is usually illustrated by lines or arrows that will point to the next element, where this element by its form and title, will inform what action or step happens at that certain stage (Frantiska, 2018).

Trying to verbally explain the flow increases the risk of misunderstandings, and therefore also potentially creating unnecessary frustration between the participants. Therefore, the benefit of using flowcharts is that it is a simple way to translate and explain the flow process to other people, regardless of their previous knowledge or experience (Frantiska, 2018). It also allows the participants to ask questions regarding specific areas of the flow process and it is easier for them to explain what they need to know more about. Furthermore, this allows for a quick and efficient response to take place.

In the process of developing flowcharts, it has been crucial for us to focus on what the solution needs to do, based on the use cases. We decided to divide the solution into different parts based on the different participants and their use cases. From there we went into deeper detail about what should be available information- and action-wise. For example, a shareholder should be able to get a list view of his/her stocks, how many, the worth of them, and to what company they belong to. Furthermore, the shareholder should be able to respond to offers on their shareholders, or he/she should be able to either sell or buy more. This view is unique for the shareholder and differentiates from the other participants' view. This

means that some general structure can be seen throughout the different flowcharts, at the same time as they are uniquely designed to fit and explain each participant’s use cases.

The flowcharts did not only clarify and help us understand each others ideas and mind-set. The process also revealed several issues in how and when the information and actions should be available to the user. For example, in the figure below you can see that we included several actions that the company could do (in “Handler for Aksjeeierbok”).

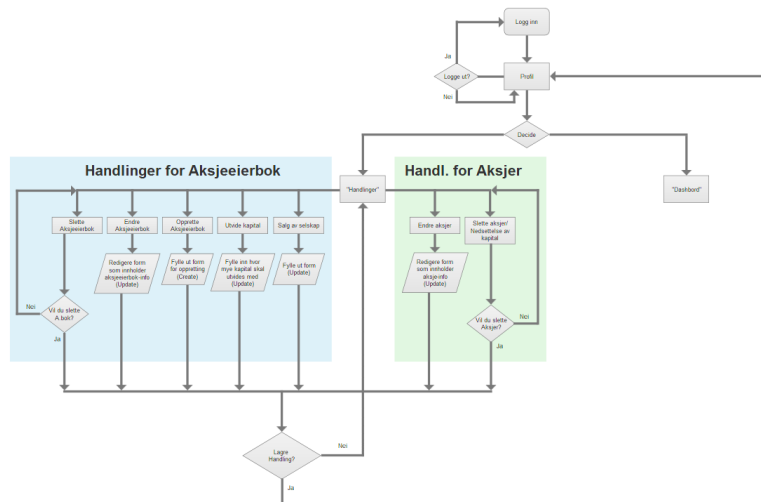


Figure 4: Section-screenshot of Flowchart for companies
(For a closer look, please zoom-in or go to the [appendix](#))

After we sent the flowcharts and presented them in a showcase, we got feedback from the customer regarding legal matters- for example that a company should not be able to delete their shareholder-book. They also gave us response regarding the “Dashboard” part of the flowchart, so that it was easier to go forth with the development and design of the flow. The figure below shows that we took the feedback into consideration and re-configured the flowcharts to fit the additional information we got through the showcase.

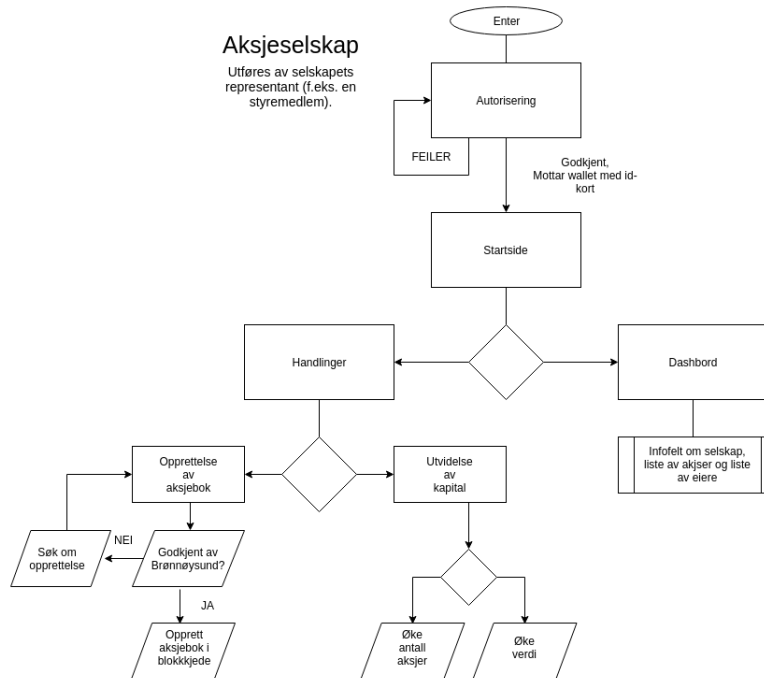


Figure 5: Section-screenshot of revised Flowchart for companies
(For a closer look, please zoom-in or go to the [appendix](#))

In our project the flowcharts have been beneficial within our group to communicate ideas and understanding to each other of how the solution should work, and in what order. Because of this, we have quickly developed an equal understanding of how each process should be carried out. To have a common understanding has been important for us in meetings with the product owner, to communicate the process flow in the best way possible. Especially, if the customer has had any questions regarding this, we have all been capable to further explain or supply information based on the presenter's explanation. Having been able to easily communicate our ideas through flowcharts has played an important role in getting the necessary information from the customer, at the right time. This has furthermore enabled us to take that information into consideration and adjust our solution towards a more solid and correct result.

3.10 Wireframes

Wireframes are visual representations of a design scheme and can act as a blueprint for what the final product will look like (Cao, 2017). The idea is that the wireframes will be used to communicate the content, layout and functionality of an interface. This is usually done through what is called “low-fidelity” (low-fi), and “high-fidelity” (high-fi) wireframes. Both types of wireframes work for the principle of communicating the ideas and plans across to other participants. However, they do have fundamental differences of when and how they should be used.

Low-fidelity

Low-fidelity wireframes usually contains a very low level of details and are made by using computer software or simply hand-sketching. The idea is that they will answer essential structure and layout questions, through using blocks/boxes with descriptive text as placeholders for where elements will be located and shown in the solution (Adiseshiah, 2017). This helps us get a rough picture of the structure and can reveal either changes or new ideas that should be included for the process ahead.

Some of the benefits of low-fidelity wireframes are that because of the low level of details, they are usually quick and easy to make. This means that they are ideal to use when there is little time (Adiseshiah, 2017). Furthermore, there is also a huge value in being able to communicate ideas and layout quickly and then being able to easily replace them with new changes that was revealed through previous low-fidelity wireframes. As they can be done through hand-sketching they do not contribute to increased project costs, meaning they can be an important tool in projects for small companies.

High-fidelity

High-fidelity works towards the same purpose as low-fidelity; communicating a design representation of a final product. The difference lies within the level of details and information. This kind of wireframes tend to be a more complete representation of the product, and the

resemblance between the high-fidelity wireframe and the final product can be quite similar. In the wireframe, the element's size, design, and relative position both on the site and to their neighbouring elements, will be accurate to the final product. High-fidelity wireframes can be interactive for the benefit of seeing how things works, what kind of impression it gives the user, and can give valuable answers in user-testing (Adiseshiah, 2017). An alternative are wireframes with annotations that gives further information or instruction of how things are working.

Our Project

In this project it was necessary for us to get an overview of what functionality the customer wanted in the solution. The use cases helped towards this and was also a good foundation to work from, towards development of the wireframes. The customer expressed during the planning and along the way, that we could experiment and test designs as we pleased. However, they communicated that functionality was of importance and should be prioritized over design and GUI.

As we entered the design process we were handed the design-guide for Altinn and Brønnøysundregistrene. The design-guide emphasized how the different kinds of elements should look and be used. Furthermore, it was stated that using the guide accurately was to be preferred. The idea was that it would give the users an impression of wholeness, and of it being one and the same solution regardless of whether the user was logged in through Brønnøysundregistrene or Altinn. We kept that in mind as well as we wanted to sketch down some ideas.

In the process everyone made low-fidelity wireframes, and later we would vote for which ones we thought had the best potential. Having done so, a part of the group went on to make wireframes that could be considered as rough high-fidelity wireframes. The idea behind this was that we wanted to be able to communicate our ideas and suggestions to the customer in a more detailed manner, without spending too much time and resources.

Wireframes for company landing page

- [Low-fi](#)
- [High-fi](#)
- [Current design: Landing page](#)

Wireframes for company actions

- [Low-fi](#)
- [High-fi](#)
- [Current design: Actions](#)

Wireframes for company dashboard

- [Low-fi](#)
- [High-fi](#)
- [Current design: Dashboard](#)

As seen in the low-fidelity sketches that can be viewed in the links above, they were made by hand-sketching and we used block-elements with descriptive text as placeholders for where the different elements would be placed. The high-fidelity wireframes in the links about were created in Adobe Photoshop CC. In these wireframes we added colours, a higher level of detail and mental cues that suggests interactivity. If we look at the very first high-fidelity figure of the company landing page, there are two elements in the middle of the page. Based on the different colour, the border and the icons used, they suggest that the users will be able to interact with them. Thus, the idea of these two elements being buttons becomes apparent.

The low-fidelity wireframes helped us get a clear idea of where to place each element and the general layout. This enabled us to develop the high-fidelity wireframes, where we would take element-characteristics and accurate placement into consideration. As shown in the figures above, the low-fidelity wireframes worked as guidance rather than showing a “final result”. We made several changes from the low-fidelity to the high-fidelity wireframes, one of them being re-arranging the elements to make the

solution orderly and well organized (referring to the wireframes that illustrate the company actions page). From low-fi to high-fi the layout has stayed consistent and shows how the low-fidelity wireframes has served their purpose throughout the process. From the development of the wireframes to what the product is today, there has been done additional changes. The changes are a result of customer feedback and can be viewed in the links above named "current design".

3.11 Software and tools

- [Trello](#): Web-based project management application, imagine a whiteboard with post-it notes.
- [Box](#): Website for sharing folders and files.
- [Slack](#): Basically a chat website, was used for seamlessly communicating.

3.12 Timetable and progress plan

Our weekly timetable was set rather early on. As some of us worked on Mondays, we agreed to meet and work together four out of the five weekdays. Monday ended up being our "day off", though we did end up working, just not in close proximity. Tuesday was mainly dedicated to tackle the other course we had on the side. Wednesday through Friday was spent working on our bachelor project.

Before starting the development, we tried to divide the project into sprints in cooperation with IBM. These sprints came together to create our overall progress plan, complete with time estimates. It is worth noting that we had little to no experience with such a large scale estimate, and that we deviated from this plan on several occasions. Due to changes in technical understanding, user-story, etc.

Sprints (time estimated after "/"):

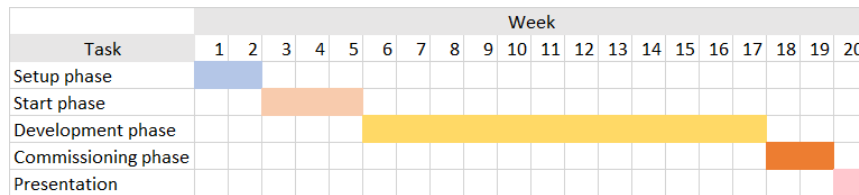


Figure 6: Timetable estimate

Sprint 0 (not included in time estimate, was completed beforehand):

- Setting up development environment
- Hyperledger Composer Intro
- Creating user-stories
- Implement simple use-case for learning

Sprint 1:

- Creating business model / 6hr
- Identity: Owner of register of shareholders / 80hr
- Identity: BR (Brønnøysundregistrene) / 12hr
- Identity: Stock owner / 20hr
- Identity: Journalist / 12hr

Sprint 2:

- Transaction: register of shareholders / 60hr
- Transaction: Transferring information about stock sale / 1hr
- Transaction: Transferring information about stock deletion / 1hr
- Transaction: Transferring information about stock value change / 1hr
- Transaction: Transferring information about representation of stock / 1hr

- Transaction: Capital expansion / 1hr
- Transaction: Sell stock / 30hr
- Transaction: Buy stock / 30hr
- Logic: Stock always needs a owner, i.e can not be created or traded with its owner value to null / 2hr

Sprint 3:

- Transaction + logic: Expanding capital / 80hr
- Setting up user interface (UI) framework / 50hr
- UI: Owner of register of shareholders / 38hr

Sprint 4:

- Easter holiday
- Unfinished parts from previous sprints

Sprint 5:

- Transaction: Changing stock value / 60hr
- UI: Journalist / 24hr
- UI: Stock owner / 38hr
- UI: BR / 38hr

Sprint 6:

- UI: Finishing touches / 38hr
- User testing / 25hr

3.13 Work log

During our work on this project we wrote an extensive work log. We used this to keep track of information from meetings and to keep track of our daily progress. This can be found in the [appendix](#).

4 Product documentation

This documentation covers the product in detail. The purpose is to explain the architecture of the solution, how to build it and brief explanation of the technologies that is used.

4.1 Architecture

The solution uses a blockchain as a fundamental element to store data with applications added to the chain to run a defined functionality. Furthermore a graphical user interface is used to execute the functionality.

Blockchain

Blockchain is a technology to store and handle information in a secure way. The information is not easily manipulated and blockchain solutions are mostly distributed to multiple nodes/hosts that validate each other. Which means manipulated information that doesn't match to agreed upon chaincode will be rejected. Blockchain-solutions have advanced algorithms to ensure this validations. The information is executed as transactions and every transaction has to be signed by the executioner. For a transaction to be signed blockchain use different mechanics to identify the users. Different solutions use different strategies for example if the user should be anonymous.

This can be summarized in three main concepts.

- Secure storing of information (blocks and crypto).
- Agreed upon chaincode (smart contracts validated by orderer).
- User mechanics (anonymous or not).

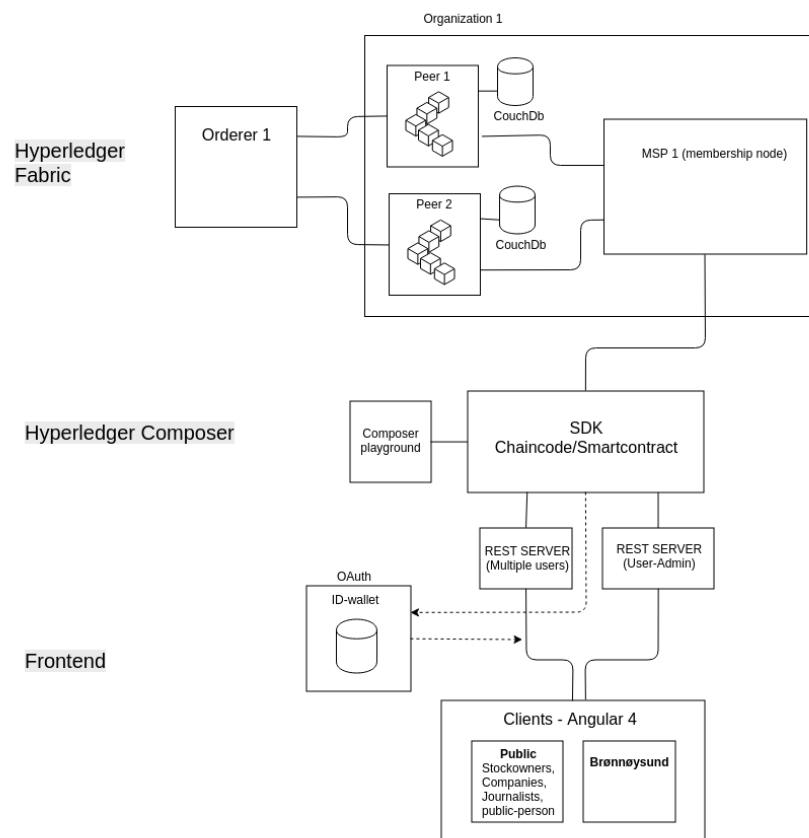
We used Hyperledger to build our blockchain solution. Hyperledger offers different tools to quickly build a blockchain system. The most fundamental is Hyperledger Fabric which run the nodes the blockchain is stored on and creates the blockchain ledger. When the nodes and ledger is up you can install chaincode (smart contracts) which will be the define how the solution will be used.

To develop chaincode we used Hyperledger Composer which is a collection of tools to define a business model and network on blockchain. Furthermore adding permissions and develop chaincode. This chaincode is easily installed on the Hyperledger Fabric.

With these options to execute the chaincode it is easy to develop and connect frontend applications for users to use the blockchain solution.

More in depth on both fabric and composer as well as our frontend applications in sections below.

Architecture model



Development tools

Name	Used for	Description
Ubuntu 16.04.3 LTS	Originally the only OS compatible with Hyperledger	Used as main OS or with a virtual machine
VMWare Workstation 12 player	Used for Ubuntu	VMWare is a software for running virtual machines on a local computer which was necessary for group members who had windows as OS
NodeJs	Used to run necessary node programs and Node Package Manager	Example: Hyperledger-composer-CLI. These programs is installed with Node Package Manager
Docker Composer	Used to create docker images for the fabric nodes	...
Hyperledger Fabric Binaries	Used to create the blockchain	...
Hyperledger Composer CLI	Used to create chaincode etc	Framework for business logic/chaincode, create data
Composer-REST-Server	Used to create a REST-API for the Hyperledger chaincode	Node program that a generate REST-API
Bash	Executing programs and scripts	All the programs mentioned is executed as bash-commands in the terminal. We also created scripts to do this.

Angular 4	Programming frontend applications	Programming language. JavaScript framework
Git	Sharing project files and version control	Git is a software for sharing files in a project with each other so everyone always has the latest version of the project files. Git is also used for version control.
Github.com	Saving the project	Github.com is connected to git and is where the project and the different versions are saved/stored.

Development environment

We use GitHub for a development repository so every group member have access to commit their work. The repository only contains source code which means that development tools like Hyperledger-composer CLI or Hyperledger-fabric binaries must be installed locally.

We host every fabric node and rest server on docker-container images locally. These images are created with docker-composer** (notice the difference between docker- and Hyperledger-composer). When images is running the fabric binaries are used to create the blockchain on peer nodes. We then install the Hyperledger-composer run-time with the Hyperledger-composer CLI. We also usually populate the blockchain with test-data with composer-CLI.

We automated these tasks by using bash scripts.

4.2 Hyperledger Fabric

Hyperledger fabric is one of Linux Foundations blockchain projects. It shares most of the attributes of other blockchain solutions. It uses ledgers, smart-contracts, consensus mechanisms and is participant and transaction based. What differentiates it from other solution is that it's private and permission based.

Hyperledger fabric contain multiple different nodes with different purpose.

Peer nodes

A peer node is a host of one or more of the distributed ledgers. The ledgers contain all information. The current state and the historian. The peer also contain the chaincode that execute the changes on the ledgers. Multiple chaincode can be installed on the peer.

Orderer node

The orderer nodes contain the algorithms that ensures that the same chaincode is executed on all the peers. This ensures that all the peers are synchronized. It is this concept that is known consensus and what makes the chaincode a so called smart contract.

Ca node - Membership Service Provider

The MSP is responsible for handling the members/users and the permissions of the blockchain. Every member gets an unique ID that is used to sign the executed transactions. Members are instantiated as a certain participant and based on what kind of participant the member will be permitted to only do certain transactions.

CouchDB

CouchDB is the integrated database system. The database is generated based on the current state of the blockchain. It uses HTTP and JSON to transfer data between its API and a client. This opens the possibility to quickly query data. But the database only contains current state which means querying historians requires query to the ledger directly.

Creating the ledger

The fabric use a set of executable binaries to build the blockchain. The architecture is depended on multiple configuration-files. The configuration uses YAML language to structure the build. Based on these configurations the binaries build the first blocks of the blockchain.

```
OrdererOrgs:
  - Name: Orderer
    Domain: example.com
    Specs:
      - Hostname: orderer

PeerOrgs:
  - Name: Org1
    Domain: org1.example.com

    Template:
      Count: 2

  Users:
    Count: 0
```

Listing 1: Small example from a YAML-file

Docker composer

When the blockchain is created it is deployed on the nodes which is instantiated as container-images with docker. Docker Composer also uses YAML-files to build the different nodes/containers.

Fabric tools

The Hyperledger Composer project have a getting started tutorial with a set of bash-scripts to build and break a standard single organization fabric. These scripts does the processes explained in the former paragraphs. instance.

Fabric summary

Keeping in mind that our thesis scope emphasizes the development of chaincode/smart-contracts and the permissions of participants and not

distributing the block-chain. By this reason the standard architecture and build scripts provided by fabric tools were sufficient.

4.3 Hyperledger Composer

Composer explained

Hyperledger Composer is a framework and tool set to develop blockchain applications made for Hyperledger Fabric infrastructure and run time. It allows to quickly define a business model/logic and build functionality (chaincode/smart contracts) on it. Composer chaincode is written in JavaScript and is installed and executed on the fabric nodes. The code manage the business definition stored on the fabric distributed ledger. Just as other fabric chaincode the fabric-blockchain consensus algorithms ensures that updates are consistent on all peers.

As well as chaincode Hyperledger Composer offers tools to build Node.js applications and/or build REST-servers to execute the chaincode as transactions and do CRUD operations (Create, Read, Update, Delete). All these tools are offered by the SDK.

SDK and API

The Composer JavaScript SDK is a series of API. It's either used from a Node.js application connected with the node-module 'composer-client' or within the transaction functions (chaincode). It can also be used manually with the Composer-CLI.

The main API are composer-admin, composer-client, common-API and run time-API.

Composer-admin

Composer-admin is used to manage the business network itself. This means installing, updating and starting the network on run times, as well as instantiating participants and issuing business-id-cards . This API is used by the PeerAdmin and if used from a Node application it must be connected to with a PeerAdmin-card. More on different ID-cards in later sections.

Composer-client and common-API

The 'composer-client' is used to connect to a specific business network with a users id-card. It manages the business network with CRUD operations on the different registries (Participant, asset, transactions, events) and by submitting transactions.

The Common-API contains information of the business network's structure and is used by the 'composer-client' to create new resource-elements that is added to the business-network.

Run time-API

The run time-API is used by the transaction functions to access the business registries, issue queries and emit events. It is used similar to composer-client excluding connecting to the network with id-cards. This is because the transactions is being executed from within the peer the business-network is stored on. The user is identified before the transaction is submitted.

The run time-API is accessible as a global class.

Run time and connection-profiles

The different run time composer supports are Hyperledger Fabric, web executed as a playground in web-browser and embedded-run time run as node.js for unit-testing. Each used for different purposes. Each run time will be explained more in depth when relevant.

A connection-profile is defined for composer to connect to a run time. As mentioned in the Fabric section, there are different nodes with different responsibilities. The connection profile specifies the TCP/IP addresses and ports of the different nodes and what kind of node it is (Peer, MSP, etc).

```
{
  "name": "fabric-network",
  "x-type": "hlfv1",
  "version": "1.0.0",
  "peers": {
    "peer0.org1.example.com": {
```

```

        "url": "grpc://localhost:7051",
        "eventUrl": "grpc://localhost:7053"
    },
    "certificateAuthorities": {
        "ca.org1.example.com": {
            "url": "http://localhost:7054",
            "caName": "ca.org1.example.com"
        }
    },
    "orderers": {
        "orderer.example.com": {
            "url": "grpc://localhost:7050"
        }
    }
}

```

Listing 2: Small sample of our connection profile

Business network cards

To connect to the blockchain business network a identification is needed. To provide this information Hyperledger Composer uses Business network cards. There are two different types of cards: peer-admin cards and standard business cards. Peer admin card is used when managing a business network on a specific peer-node. Standard business cards is for the users and is bound to a defined participant. So when an user joins the business-network a participant is instantiated and a network card is handed to the user.

```
bjodn:~/Altinn-Blockchain$ composer card list
The following Business Network Cards are available:
Connection Profile: hlfv1
```

Card Name	UserId	Business Network
Offentlig@altinn-network	Offentlig	altinn-network
Acme@altinn-network	Acme	altinn-network
Frank@altinn-network	Frank	altinn-network
Bronnoysund@altinn-network	Bronnoysund	altinn-network
Eva@altinn-network	Eva	altinn-network
admin@altinn-network	admin	altinn-network
Journalist@altinn-network	Journalist	altinn-network
PeerAdmin@hlfv1	PeerAdmin	

```
Issue composer card list --card <Card Name> to get details a specific card
Command succeeded
bjodn:~/Altinn-Blockchain$
```

Business Model

As previously mentioned participants are instantiated on the network. But for this to happen the business network need to have the participants defined. This is done in the business model.

The business model represent the fundamental structure and logic of the system. The model is structured as participants, assets, transactions, events and queries. These are defined data-structures that describe how data is stored on the distributed blockchain-ledger. What roles these different structures have and their data change depend on the developed chaincode.

Participants

Which actors the network shall contain and what information they will consist of is defined with the participant structure. Participants are as mentioned instantiated by users and bound to a id-card which determines who a participant is. Furthermore the role the user has is depended on the defined participant.

Assets

The asset structure defines what the participants will be transacting. Basically what the business is about. The asset's change state by executing transactions.

Transactions

The transaction structure defines what data a transaction will require to be executed and what will be stored from the execution. For example what asset will be transacted, time-stamp and the participant whom executed the transaction.

Events

As a part of the transaction logic composer gives you the option to emit event signals. What information the event will emit is defined as a event structure in the model. These events can be subscribed to with web-socket technology.

Concept

Concepts in practice is just a ordinary data struct (very simple data type). It is the only structure that can be contained in the other structures. The other structures can only be relations. Contrary to the other structures concepts has no functional purpose.

Enum

The model allows to define enumerated data types.

Queries

Hyperledger Composer uses its own query language to define queries to fetch information from the state-database and the ledger. The query is defined in the file 'queries.qry'. When querying current state data the business network path to wanted registry is selected. If querying data which is not current state the path to the ledger's historian is selected.

```

query selectAllStocks {
  description: "Select all Stock belonging to
    RegisterOfShareholders with orgnr _$orgnr"
  statement:
    SELECT org.acme.biznet.Stock
      WHERE (registerOfShareholders == _$orgnr)
}

```

Listing 3: Example of query in model

```

query getStockPurchaseRequest {
  description: "Select a stockPurchaseRequest Transaction"
  statement:
    SELECT org.hyperledger.composer.system.HistorianRecord
      WHERE (transactionId == _$id)
}

```

Listing 4: Example of query-definition in queries.qry

The difference is that the business network's current state is stored on a couchDB-database as well as the ledger and the historian is only on the ledger. This is because of efficiency reasons. It is quicker to fetch and work with current state data on a temporarily generated couch-database and the historian is too large to generate.

Transaction Logic and chaincode

The business model defines the structure of the business network but it doesn't define how the business logic is executed other than with CRUD-operations. This is where chaincode also called smart contracts play a part.

Hyperledger Composer uses JavaScript to define the chaincode as transactions which are executed as a JavaScript function. The function takes an object as a parameter which is the transaction-struct defined in the model.

```

/**
 * Transaction for register a change in the firms
 * stockBook

```

```

    * @param {org.acme.biznet.RegisterChangeOnFirm}
      registerchangeonfirm - changing stockbook for
      firm
    * @transaction
    */
    async function RegisterChangeOnFirm(
      registerchangeonfirm) {

      const factory = getFactory();
      const request = factory.newConcept('org.acme.
        biznet', 'ResponseRequest');

      ...

      const event = factory.newEvent('org.acme.biznet'
        , 'changeOnFirmRegistered');
      event.capitalChange = registerchangeonfirm.
        capitalChange
      event.newCapital = registerchangeonfirm.
        newCapital
      event.amountOfNewStocks = registerchangeonfirm.
        amountOfNewStocks
      event.shareholderRegistryID =
        registerchangeonfirm.shareholderRegistryID
      return emit(event);
    }

```

Listing 5: Extraction of transaction function with transaction object as parameter

As mentioned the functions uses the run time-API to fetch participants and assets from the ledger's different registries to do CRUD, queries and emit events.

It's these defined transaction functions that is the main aspect of smart contracts as it is these sequences of code/operations that the consensus algorithm validate.

ACL

Hyperledger Composer uses Access Control Language (ACL) to define which operations that are permitted by the different participants. This means for example which assets specified participants can do specified CRUD-operations on or specified transactions can only be submitted by

specified participants.

The rules is defined in a file called 'permissions.acl'. The rules are evaluated in order, and the first rule that matches a given condition decides if an user is allowed to perform a given operation.

```
rule FirmReadDeleteOwnRegisterOfShareholders {  
  description: "Allow Firm to read and delete own  
    RegisterOfShareholders"  
  participant(p): "org.acme.biznet.Firm"  
  operation: READ, DELETE  
  resource(r): "org.acme.biznet.RegisterOfShareholders"  
  condition: (r.registerOfShareholders.getIdentifier() == p.  
    getIdentifier())  
  action: ALLOW  
}
```

Listing 6: Example of permissions.acl

Other development tools

Composer-CLI

The composer-CLI is a node.js software containing every composer operation (Installing chaincode, issuing id-cards, creating participants, etc). As it is a command line software. All the commands can be scripted as bash scripts.

Composer-playground

The Composer-playground is run as a web run time in the web browser with the purpose of testing the business network. It allows to manually and temporarily change the chaincode, business model and ACL. It has a test business network option where you connect with optional id-card and execute CRUD-operations and submit transactions.

Yeoman

Yeoman is a Node software which is used to generate all sorts of project templates.

Composer rest server

Composer rest server is a node program that generates REST-api based on the business model. This means CRUD-operations for every participant, asset and transaction as well as executing transactions and queries.

Options:

Standard rest-server - id-card to which will be the id that signs every operation.

Multiple user rest-server - id-card is passed as a parameter during request contrary to it being a static configuration.

Authorization - Every request require a token to be accepted.

Rest-server explorer

A GUI-explorer for the REST-Server-API created by Swagger. It offers a overview of all the operations as well as execution.

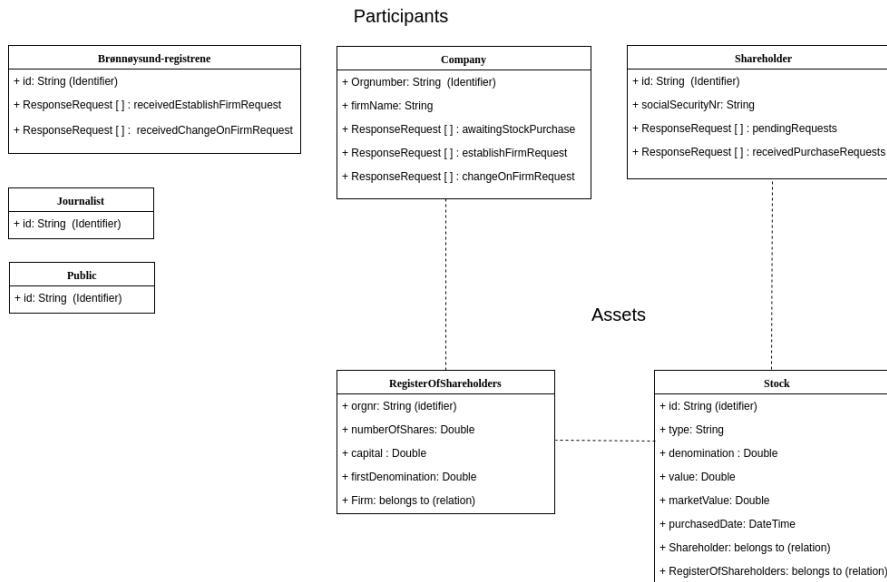
Altinn Network - Our product

The skeleton of composer-project is initiated by yeoman which created the folder-structure with all necessary files. We named the business network 'Altinn-network' as this project is a collaboration between Altinn and Brønnøysundregistrene. The network we developed is defined as follows:

Shareholder registry business model

The model based on the requirements and use-case definition we did early in the project and everything is defined in the model file 'org.example.biznet.cto as demonstrated in the previous section.

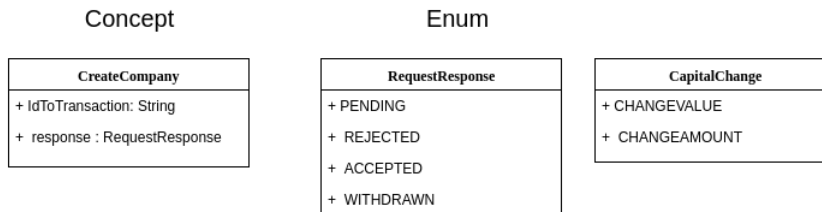
Participants, assets and relations



This model shows all the participants and assets of the business network. Company owns a stockbook which consists of one or more stocks which are all unique and owned by a shareholder.

Concepts and enums

We use a defined concept and a couple of enums in the transactions. They are necessary to be aware of before observing the transaction definitions.



The smart-contract/chaincode of Shareholder registry

What constitutes the smart contract of this shareholder registry network is the following defined transactions.

Transaction models

This is a model of all the transactions structs defined in the model-file. Since the main 'operations' of the system require multiple participants to participate the operations are split into multiple transactions.

Create Stockbook

RegisterFirm
+ Capital : Double
+ numberOfStocks : Integer
+ firmIdentifier : String
+ newStockOwners [] : String
+ distribution [] : Integer

CreateCompany
+ IdToTransaction: String
+ response : RequestResponse

Expand capital

RegisterChangeOnFirm
+ capitalChange : CapitalChange
+ newCapital : Double
+ amountOfNewStocks : Integer
+ shareholderRegistryID : String

ExpandCapital
+ IdToTransaction: String
+ response : RequestResponse

AddStocks
+ IdToTransaction: String
+ response : RequestResponse

Sale of stocks

requestPurchase
+ bid: String
+ quantity : Integer
+ customer : String
+ stockOwner : String
+ registerOfShareholders : String

respondToPurchaseRequest
+ IdToTransaction: String
+ response : RequestResponse

SaleOfStock
+ IdToTransaction: String
+ response : RequestResponse

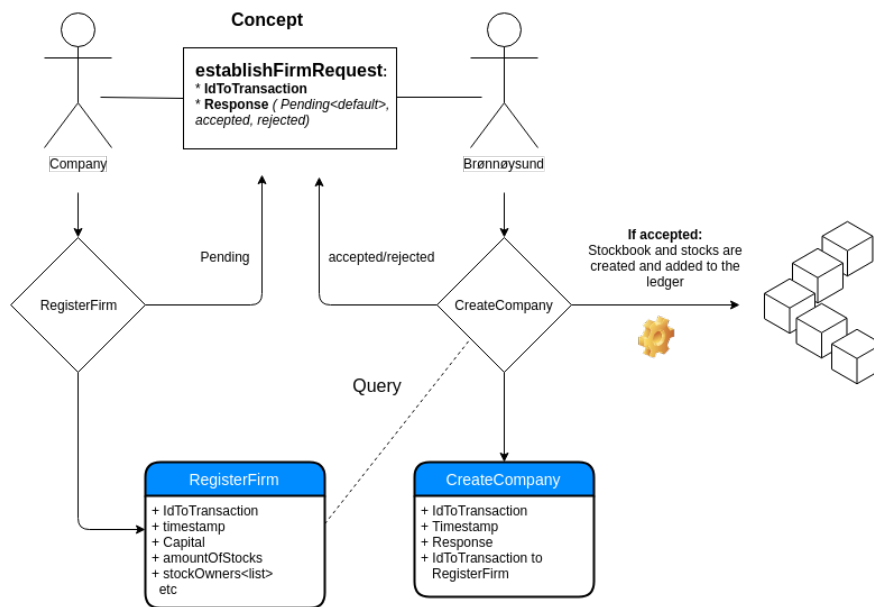
Transaction flow

The logic as mentioned earlier are defined as javascript functions in the logic.js file. To demonstrate how the transactions are executed we have modeled the flow of the transactions.

Create stockbook

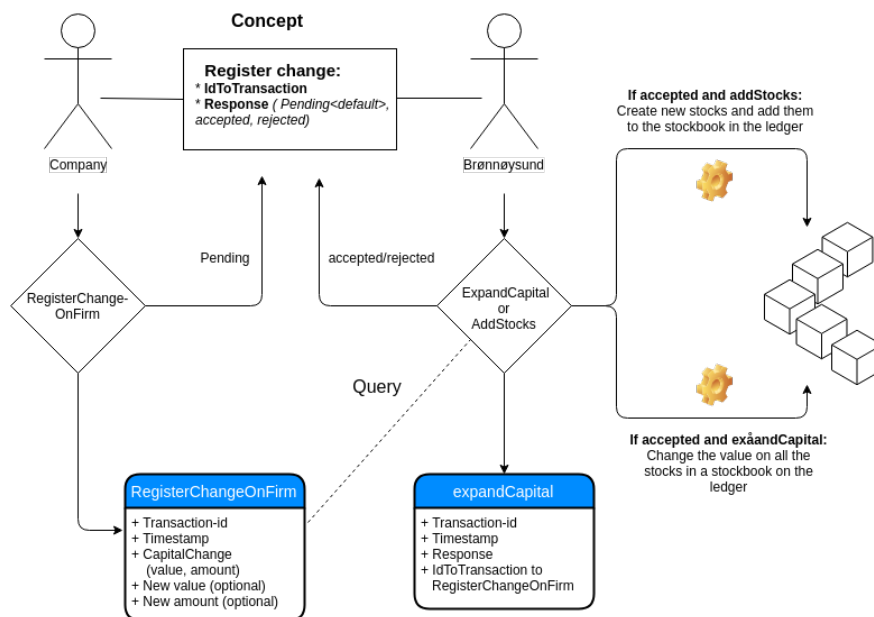
When a company registers a register of shareholders it needs to be accepted by Brønnøysundsregistrene. A company participant executes the transaction 'RegisterFirm' which stores a transaction with all needed data. Also an awaiting response concept is given to both Company and Brønnøysund. Based on the response-concept the Brønnøysund participant

query the registerFirm transactions and respond with either 'REJECTED' or 'ACCEPTED'. If accepted the register of shareholders and shares are created on the ledger.



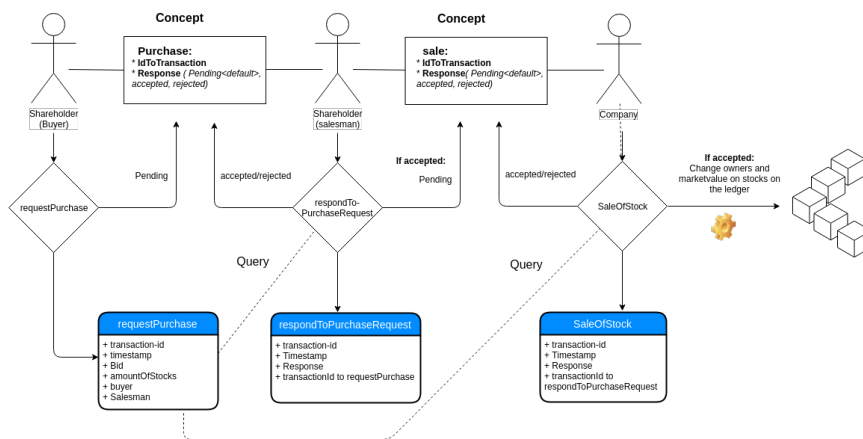
Expand capital

The company participant executes the transaction 'RegisterChangeOnFirm' with necessary data and the transaction is stored as well as a concept given to the participants. In this flow the company defines if the expansions is a 'CHANGEAMOUNT' of stocks or 'CHANGEVALUE' of total stock value. Brønnøysund executes 'expandCapital' with a response 'ACCEPTED' or 'REJECTED' and if accepted expansions is executed on the ledger. Otherwise the response concept is set to 'REJECTED'.



Sale of stocks

The participant StockOwner/shareholder executes 'requestPurchase' and defines amount of stocks, price, and which stockowner to buy from. Transactions is made and concepts are given to participants. Stockowner accepts or rejects by executing 'respondToPurchaseRequest' and if accepted another concept is made for Company who also need to accept the purchase. Company responds by executing 'SaleOfStock' based on given concept. If company accept purchase the given stocks change owner and marketvalue.



Events

Events are defined such that users are able to subscribe to them and receive real time notifications. The events consist mostly of data from the transaction that emits it. For example 'createStockBook' emits an event to Brønnøysund consisting data requesting company.

Access control

Based on our use case we have made 'read' the only CRUD-operation possible for any participants. Every changing operation made by a participant must happen by executing chaincode transactions.

Furthermore, participants only gets access to the transactions relevant to their role.

1. Company

- Read access to own register of shareholders and shares
- Execute access to 'RegisterCompany'-transaction.
- Execute access to 'ExpandCapital'-transaction.

2. Shareholder

- Read access to own shares and stockmarked.
- Execute access to 'RequestPurchase'-transaction.
- Execute access to 'respondToPurchase'-transaction.

REST-server

In our architecture we use REST-server's to expose the networks operations to the user.

Server for useradmin

REST-API generated with Swagger with admin-card as static configuration. Used solely for creating new id-cards and instantiating participants.

Multi-user server with authorization

REST-API for multiple users requiring authorization token. User is authorized and receive id-card frontend. Based on id-card and what kind of participant it is bound to the user get access based on the participants accessibility.

Build

Fabric tools are used to create the ledger and composer CLI is used to install the business network. These steps are scripted.

rebuild.sh Tears down existing fabric and creates new using fabric tools. Installs the business definition and runs the seed.sh and indexDouchDB.sh.

seed.sh Populates the business network with participants and issues id cards.

APIstart.sh Uses the composer rest api program with pre configured options.

indexCouchDB.sh Sorts the data in the current state database with a index.

upgradeRuntime.sh Generates new business definition and install it to the business network.

4.4 Frontend

The frontend consist of two developed web applications.

Brønnøysundregistrene being a public authority is one client, only used by Brønnøysundregistrene. The other being a public client accessible for everyone.

The business networks operations is accessible in the web clients from the REST-API and is exposed to the user based on chosen role/purpose. This is solved by splitting the application into different components exposed with the relevant requests to the REST-API.

Angular applications

Both applications need good support for submitting and showing data therefor Angular is the chosen language used to develop these applications.

Angular is a JavaScript framework developed to ease development of web applications. It can be used to develop all sorts of applications, but it is most suitable for building applications that handles data. Angular does also support other libraries like NativeScript which makes it easier to create cross-platform applications.

Both angular applications consist of components for each page that is displayed. These components consist of a typescript file that defines the logic for the component and a html template that defines how page looks like. In addition to these components the application has a routing module that defines routes to the other components and a app.component file that declares the other components and providers. Listing 7 is an example on

how a routing module can look like. This example is taken from the web application developed for Brønnøysundregistrene.

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { LandingPage } from '../LandingPage/LandingPage.component';
import { Request } from '../Request/Request.component';
import { RegisterOfShareholders } from '../RegisterOfShareholders/RegisterOfShareholders.component';

const routes: Routes = [
  {path: 'LandingPage', component: LandingPage},
  {path: 'Request', component: Request},
  {path: 'RegisterOfShareholders', component: RegisterOfShareholders},
  {path: '**', component: LandingPage}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule],
  providers: []
})
export class AppRoutingModule { }
```

Listing 7: Example of a routing module

Brønnøysundregistrene client

The web application developed for Brønnøysundregistrene is formed by four different Angular components. These components are designed to give Brønnøysundregistrene the opportunity to respond to various applications regarding joint-stock companies, as well as view information about all joint-stock companies registered on the network.

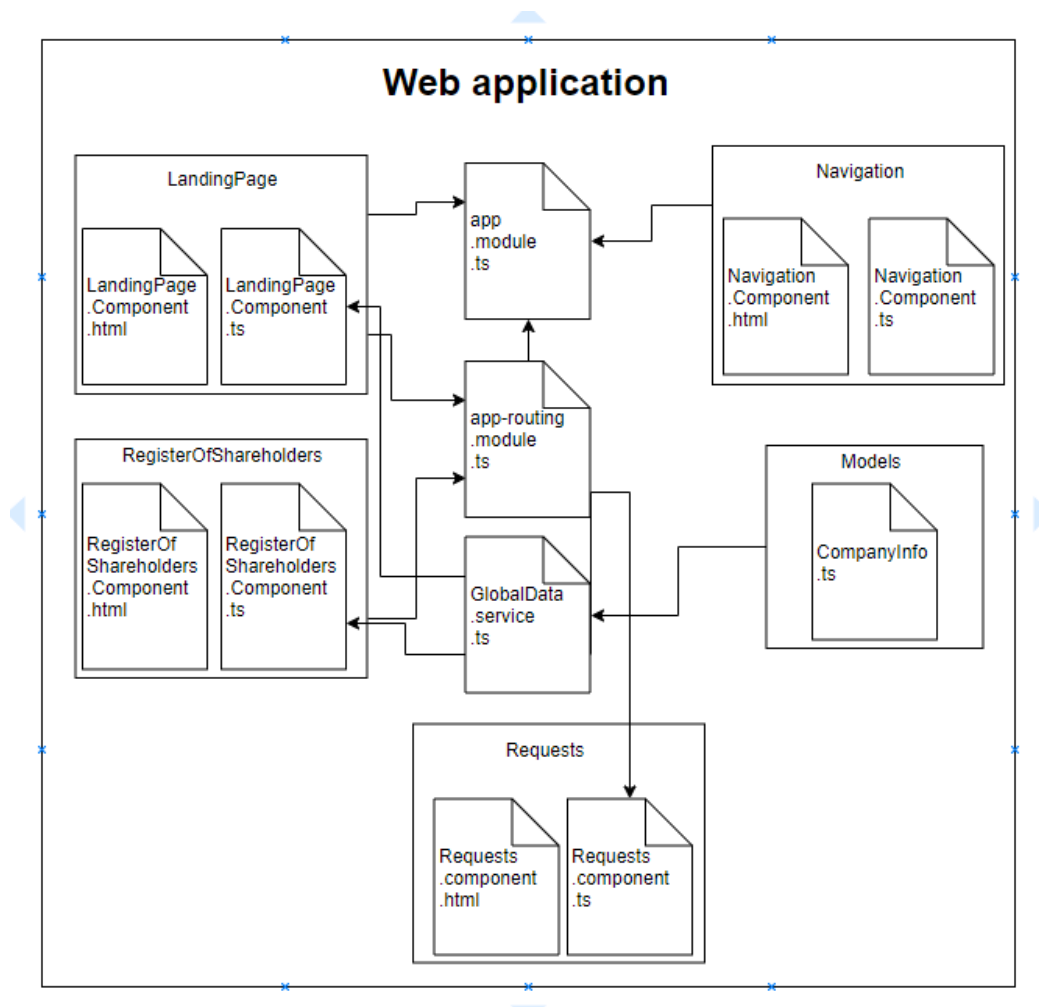


Figure 7: Structure of web application for Brønnøysundregistrene. Arrows symbolizes imports.

Navigation

The navigation component of the web application designed for Brønnøysundregistrene consist of a horizontal menu-bar that links to the other components in the application. The navigation-bar as a stand-alone component so it can be included and reused in other components in the applicaiton.

Landing page

The landing page component is meant as welcome page that easily lets the active user navigate to the component the user is looking for on this visit. It only contains links to other components and no logic.

Requests

The request component of the web application handles application forms delivered to Brønnøysundregistrene. It displays all the applications submitted and lets the user respond to them. All unprocessed applications are retrieved from the blockchain using a call to the API, parsed into JavaScript objects, and sorted into two different lists before they are displayed. The active user can then read the applications and decide whether to approve or reject an application.

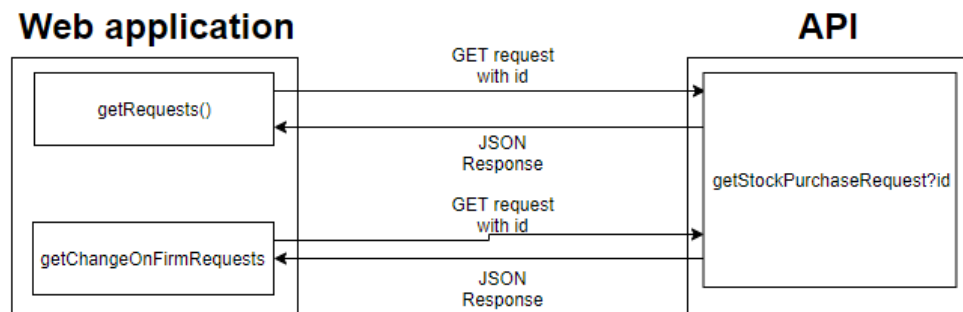


Figure 8: GET request to API

The applications available for processing here is: Application for registration of a Register of shareholders (joint stock company), Application for capital increase and Application for new share issue.

When processing the applications the active user chooses to reject or approve of the given application. This sends a POST request to the API with the information needed on the peer nodes to process the transaction.

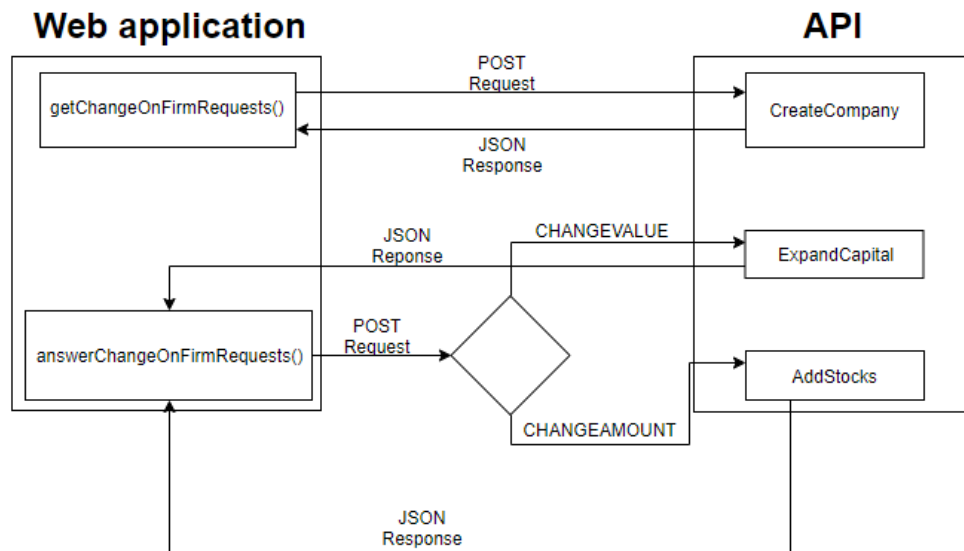


Figure 9: POST request to API

Register of Shareholders

The last component in Brønnøysundsregistrene's web application is a component designed so that the users can get an overview of all the current registered corporations.

The component is connected to the API and runs a query to get a list off all corporations registered. This list consist of JavaScript objects and are later displayed in the HTML template using the Angular directive `*ngFor`.

`*ngFor` is a directive that the application to loop through an array of JavaScript objects or variables and display them programmatically in a HTML template.

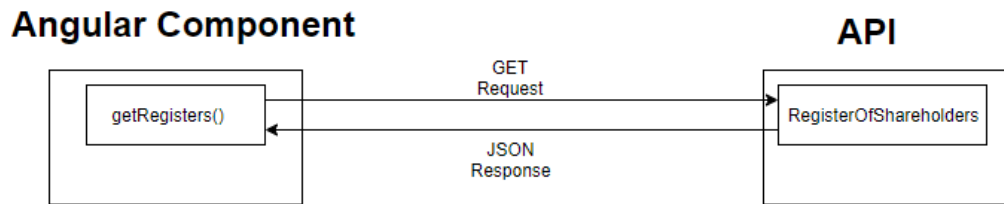


Figure 10: GET request to API

Listing 8 show how the Register of Shareholders component post a GET request to the API and places the response in an array or JavaScript objects.

```
import { Component, OnInit, Input } from '@angular/core';
import { Http } from "@angular/http";
import 'rxjs/add/operator/toPromise';
import "rxjs/add/operator/map";
import { Headers } from "@angular/http";

@Component({
  selector: 'app-RegisterOfShareholders',
  templateUrl: './RegisterOfShareholders.component.html'
})

export class RegisterOfShareholders implements OnInit{

  private allRegisters: Array<any>;
  private showContent: boolean;

  ngOnInit(): void {
    this.showContent = false;
    this.getRegisters()
  }

  constructor(private _http: Http) {
  }

  getRegisters() {
    //this._http.get('http://localhost:3000/api/
    RegisterOfShareholders')
    this._http.get('http://aksjebok-hyperledger.localtunnel.
```

```

        me/api/RegisterOfShareholders')
    .map(resultat => {
        let JsonData = resultat.json();
        return JsonData;
    })
    .subscribe(
    JsonData => {
        this.allRegisters = [];
        for(let r of JsonData){
            this.allRegisters.push(r);
        }
        this.showContent = true;
    },
    error => alert(error),
    () => console.log("Info om aksjeboker hentet fra
        systemet")
    );
}
}

```

Listing 8: Register of shareholders component, Brønnøysundregistrene application

Public web client

The public application is developed so that different types of users can register and use the application. Users can choose between three different types of account or use the section that don't require log in. This section do only provide information about registered corporations register of shareholders. The three different types of accounts are a business account for businesses, a stock-owner account for people that trades / owns stocks and a journalist account for journalists that want to check a company's trade and / or capital expansions history.

All the different account types have different roles on the business network and have permission to view different parts of the open web application.

The business account has permission to view the business part of the web application which allows the user to submit applications to Brønnøysundregistrene for registering a register of shareholders, capital expansion and view information about the company and who owns how

many shares in the company.

The stock-owner account has permissions to view the private part of the application. On this part of the application each user is presented with a overview of all the shares they own in different companies, each stocks value as well as a total value owned in each firm. Stock owners can also visit the stock market and submit requests to buy stocks from other users registered on the network.

Public web application

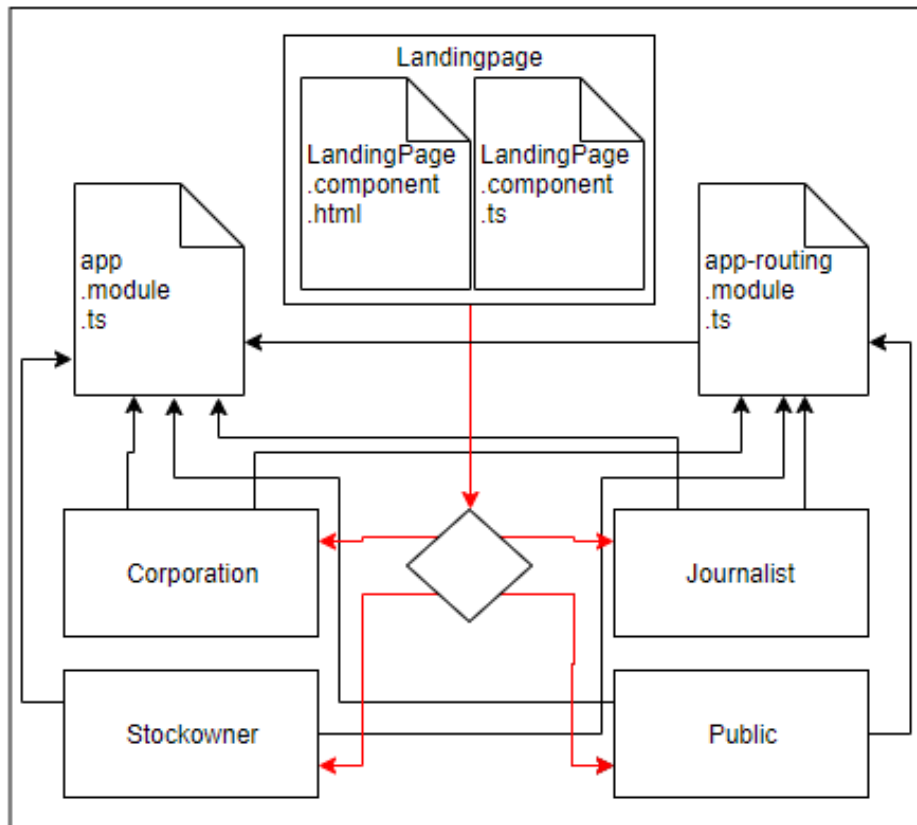


Figure 11: Public web application structure. Red arrows symbolizes flow from login to different parts, black arrows symbolizes imports

Business client

The part of the web application for businesses are meant as a portal for the CEO or any of the board members of a corporation. Here they can find current information about their company or submit new applications to Brønnøysundregistrene. The different kind of applications are registration

of a new joint stock company, capital expansion and new share issue.

Business users have two main components they can visit, a dashboard for overview and an actions tab where they submit applications. As well as these two there is one Navigation component, a footer component and a landing page component.

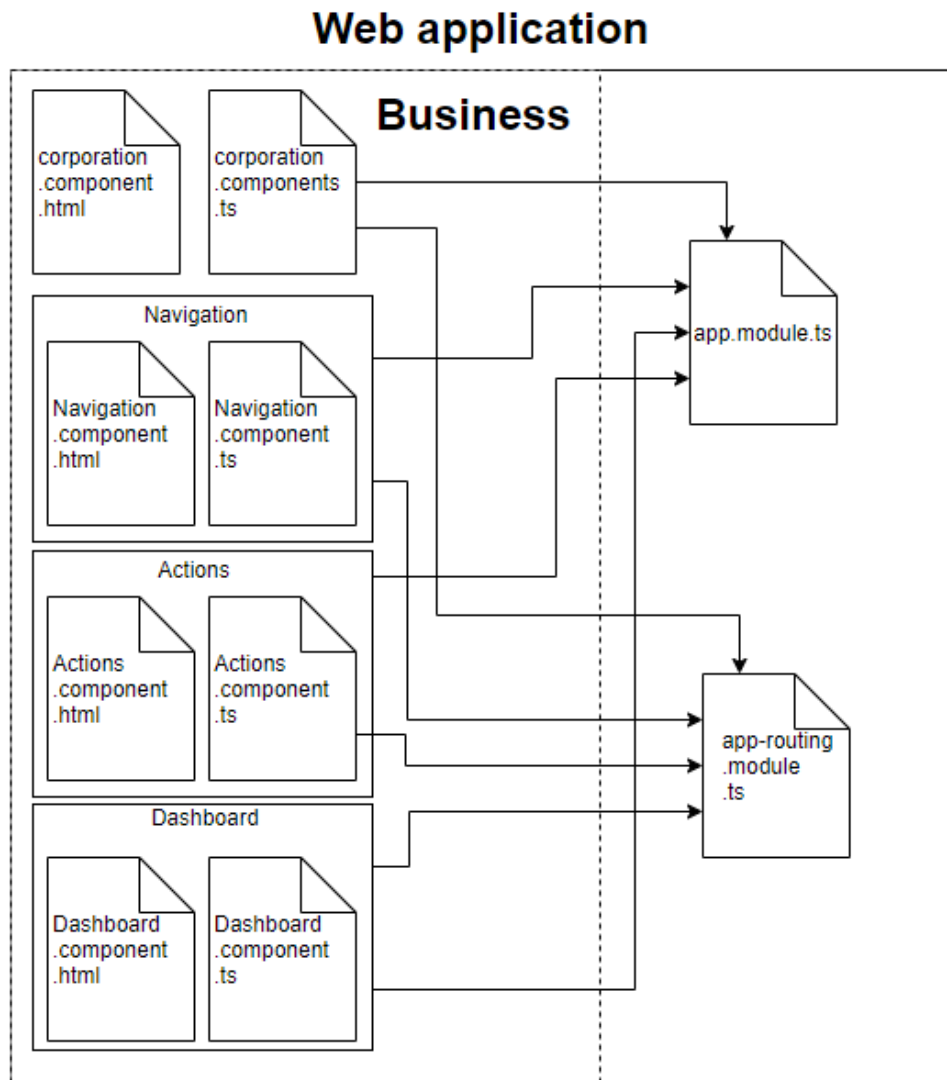


Figure 12: Structuring of business part. Black arrows symbolizes imports.

Dashboard

The dashboard component is designed to give the active user an overview of the current standing of the corporation's shares, capital, number of shares and the shares first denomination, as well as a list of all current

shareholders and how many shares they own.

To get this information the client queries our API and retrieves the company's register of shareholders based on the corporation's organization number (organisasjonsnummer). This information is stored in a JavaScript object and displayed in the components HTML template.

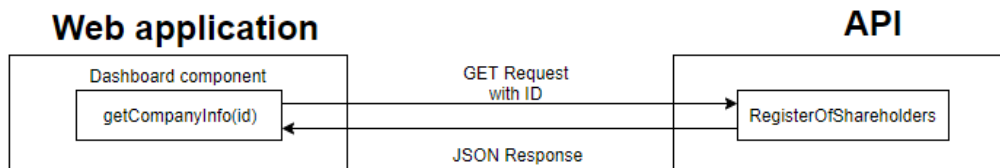


Figure 13: GET Request to get a JSON string with user info for current account of type RegisterOfShareholders (Business)

For the list of current shareholders there is a bit more logic involved. First the client queries the API and get a list off all current shares and its owner, before grouping the shares and sorting out owners. When sorting owners, the logic checks the owner's id and counts how many times this id is stored as owner of a share. Each time the sorting function finds a new id, it stores the last id and how many shares in a JavaScript object and puts this inside an array. After the sorting is done, the client is left with an array of JavaScript objects that contains each shareholder and how many shares each shareholder owns. The list is then display in the components HTML template using Angular's *ngFor directive.

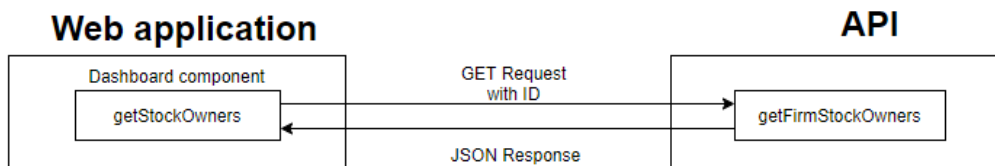


Figure 14: GET Request to get a JSON string with all users that own shares in current corporation

```

groupOwners() {
    var temp: {[k: string]: any} = {};
    temp.id = '';
    temp.numberOfShares = 0;

    this.allStocks.sort(this.compareOwner);
    this.allOwners = [];
    for(let i = 0; i < this.allStocks.length; i++) {
        if(temp.id == this.allStocks[i].owner) {
            temp.numberOfShares += 1;
        } else {
            if(i == 0) {
                temp.id = this.allStocks[i].owner;
                temp.numberOfShares = 1;
            } else {
                temp.id = temp.id.split("#").pop();
                this.allOwners.push(temp);
                temp = {};
                temp.id = this.allStocks[i].owner;
                temp.numberOfShares = 1;
            }
        }
    }

    temp.id = temp.id.split("#").pop();
    this.allOwners.push(temp);

    console.log(this.allOwners);
    this.showStockOwners = true;
}

compareOwner(a,b) {
    if(a.owner < b.owner)
        return -1;
    if(a.owner > b.owner)
        return 1;
    return 0;
}
}

```

Listing 9: Group and sort (compare) functions, Business Dashboard

Actions

On the 'Actions' tab the active user will find all the forms available for a business user, as well as functionality for approving / rejecting stock exchange of the corporation's shares.

The different forms provide the current user with input fields and makes it easy for the user to submit an application. When submitting, the client checks whether the input is in the correct format or not. If it is correct and all required fields are valid the information is parsed into a JSON-string and posted to the API. The API will then return with a message, telling the user whether the application was correctly submitted or not.

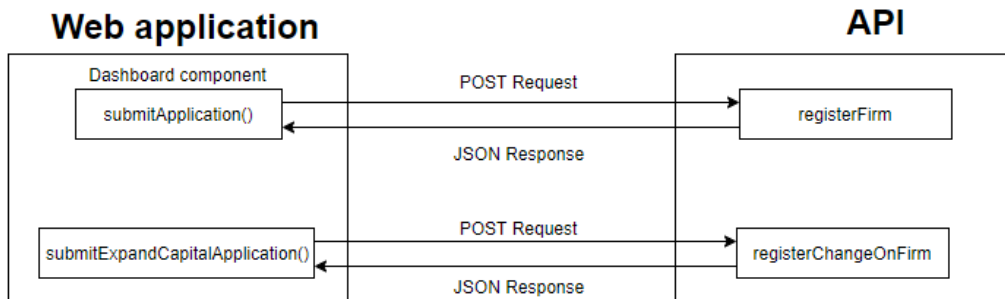


Figure 15: POST Request done to the API from the Actions tab

When loading, the components also makes a call to the API to retrieve a list of unprocessed stock exchange requests. These requests represent a sale of shares between two people and the corporation has the authority to approve or reject these sales. If the corporation chooses to approve of the sale the client connects to the API and makes the changes necessary for the sale to happen. If they reject the sale the client connects to the API and stops the sale from happening. In both scenarios the users affected by this is notified.

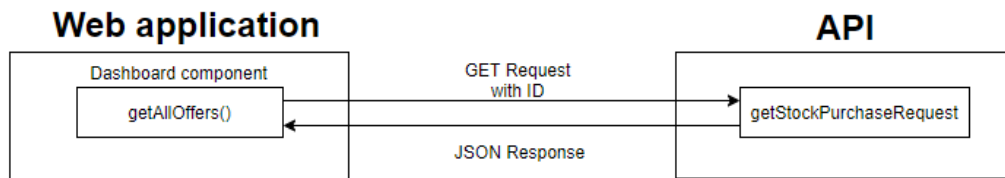


Figure 16: GET Request to get a JSON string containing info about a stock exchange linked to the current active corporation

Navigation

The navigation components are what makes up the visible HTML navigation bar on the business part of the web application. It contains routes to the different components available to business users and are included in the two main components.

Footer

The footer component contains a search field, so the users can search through available documents and pages in the application, a link to Altinn's web page and a link to a support site for the service.

Users can also find information about Altinn, cookies and privacy and operational messages in the footer. This footer is used throughout the web application and holds the same functionality no matter what type of account the user is logged in with. The only thing that differs from account to account is what results that will be displayed after a search.

Stock Owner client

The web application has a part dedicated to users who want to trade stocks. It is open to the public, but you need to register and get a verified account to use it.

When logged in as a stock owner the user has access to the stock market to see what shares are open for sale, a request list of incoming bids on owned shares and a wallet that provides an overview of how many shares the user

owns in different corporations.

This part of the web application consists of five components. The three already mentioned, a navigation component and a footer component. The users are also presented with a landing page.

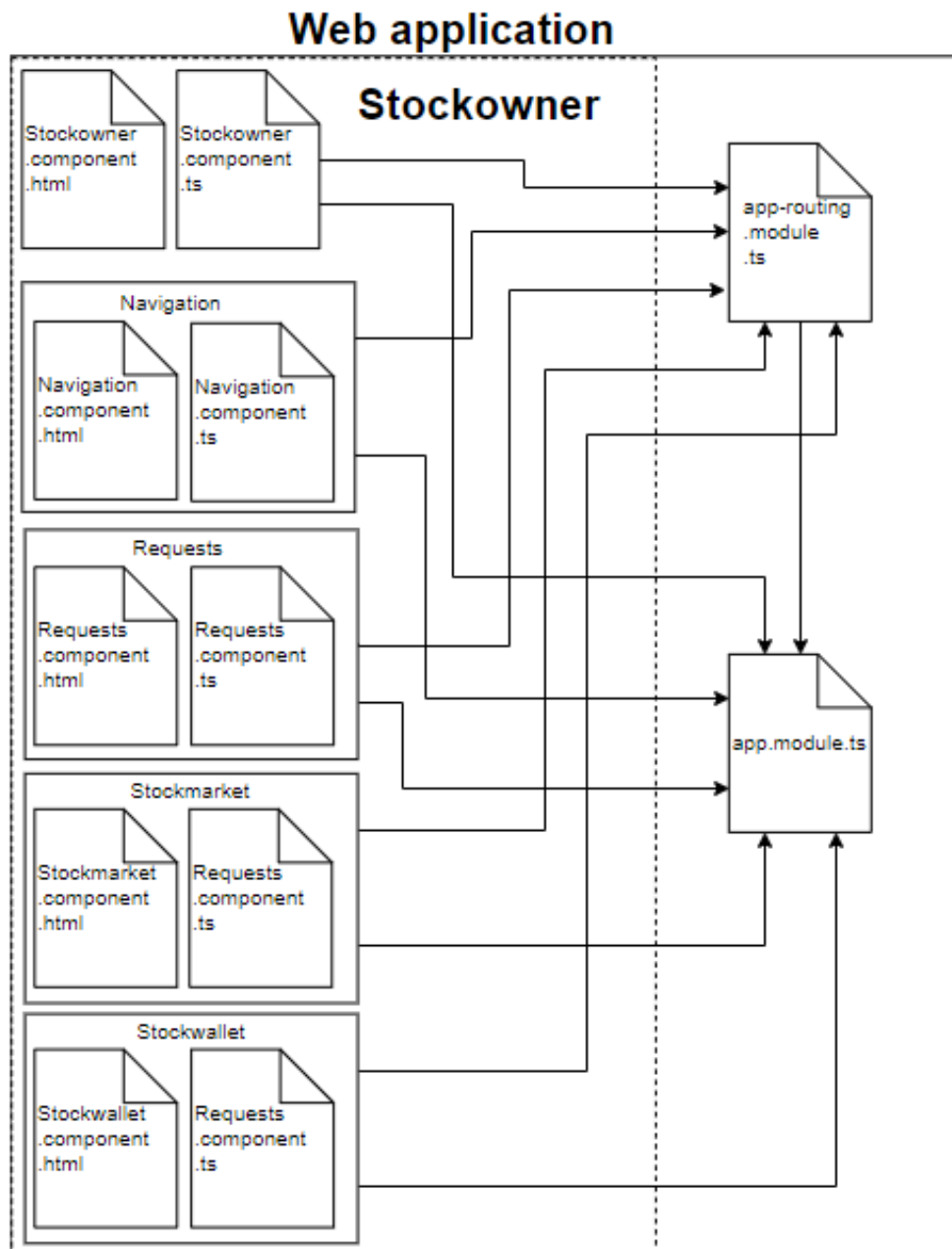


Figure 17: Overview of components available from a stock owner account

Stock wallet

The stock wallet provides the user with an overview of the different corporations he/she owns shares in and how many shares he/she owns in the corporation.

When loaded, the component queries the API for a list of all the shares the current user owns. These shares are placed in an array before they are sorted and grouped on different corporations. The sorting functionality looks at what register of shareholders, identified by organization number, the share is registered to and creates a JavaScript object for each corporation.

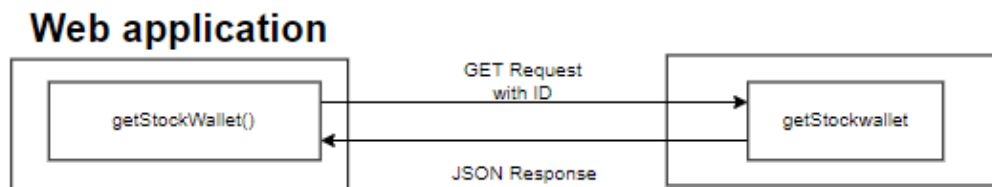


Figure 18: GET Request to receive all shares a person owns

These JavaScript objects contain information about what corporation the shares belong to, how much they are worth, the market value of each share, how many shares the user owns in the corporation and the total market value of all the users owned shares.

```
cleanStockWalletList () {
    this.stockwallet.sort(this.compare);

    this.temp = new StocksPerCompany();
    this.cleanedStockWallet = [];

    for(let i = 0; i < this.stockwallet.length; i++) {
        if(this.stockwallet[i].registerOfShareholders == this.
            temp.registerOfShareholders) {
            this.temp.numberOfShares += 1;
            this.temp.totalMarketValue += this.stockwallet[i].
                marketValue;
        }
    }
}
```

```

    }
    else{
        if( i == 0){
            this.temp.registerOfShareholders = this.stockwallet[
                i].registerOfShareholders;
            this.temp.numberOfShares = 1;
            this.temp.denomination = this.stockwallet[i].
                denomination;
            this.temp.value = this.stockwallet[i].value;
            this.temp.marketValue = this.stockwallet[i].
                marketValue;
            this.temp.totalMarketValue = this.stockwallet[i].
                marketValue;
        }else{
            this.temp.registerOfShareholders = this.temp.
                registerOfShareholders.split("#").pop();
            this.cleanedStockWallet.push(this.temp);
            this.temp = new StocksPerCompany();
            this.temp.registerOfShareholders = this.stockwallet[
                i].registerOfShareholders;
            this.temp.numberOfShares = 1;
            this.temp.denomination = this.stockwallet[i].
                denomination;
            this.temp.value = this.stockwallet[i].value;
            this.temp.marketValue = this.stockwallet[i].
                marketValue;
            this.temp.totalMarketValue = this.stockwallet[i].
                marketValue;
        }
    }
}
this.temp.registerOfShareholders = this.temp.
    registerOfShareholders.split("#").pop();
this.cleanedStockWallet.push(this.temp);
}

```

Listing 10: Function that removes duplicates from stock list

Requests

The request component available to stock owner users, is designed for easy access to all offers other users have made to buy the current users shares.

On load the component makes a call to the API to retrieve all unprocessed

offers. All offers retrieved from the API is then placed inside an array and displayed to the user and each offer is fitted with response buttons. One for accepting and one for rejecting.

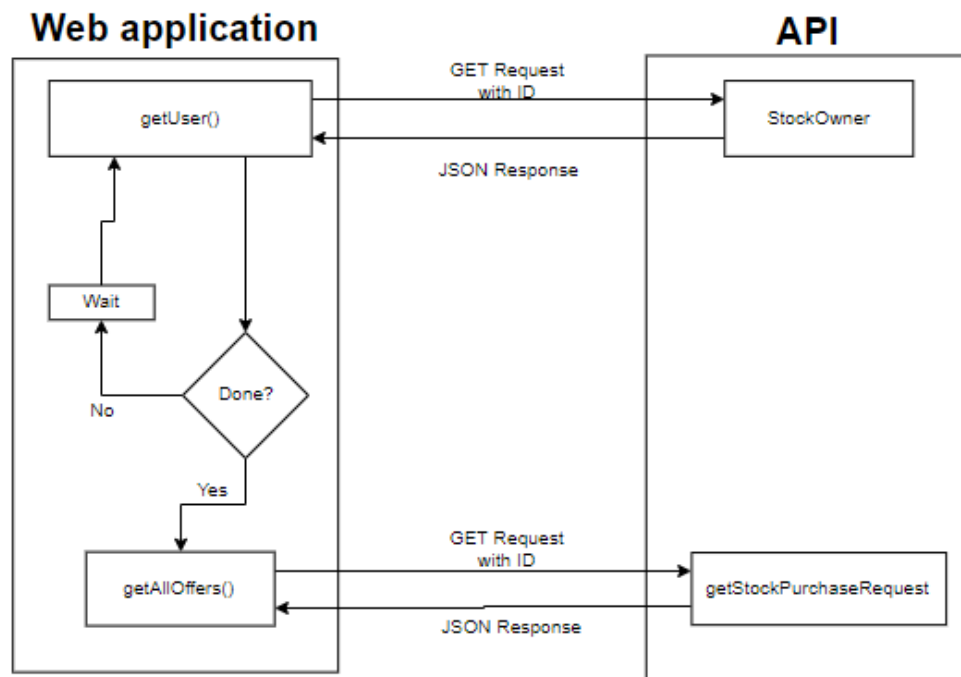


Figure 19: GET Requests to API to get all active offers current user has.

When one of the buttons are pressed, the user is prompted to confirm his / her choice. If confirmed the client submits a POST request to the API so the chosen changes take place. A POST request for rejection will stop the exchange from happening and notify the affected users and an approved request will notify the corporation the shares belong to that someone wants to exchange shares. The corporation must then approve or reject the exchange before the actual exchange happens.

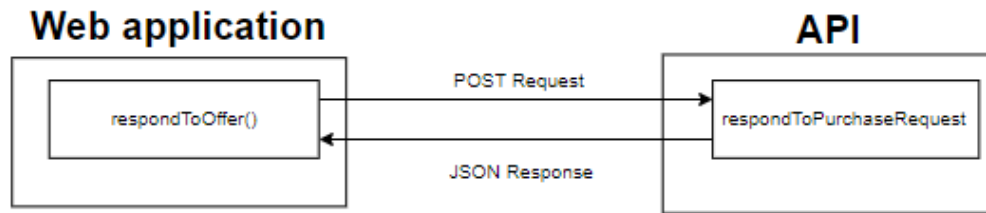


Figure 20: POST Request for responding to a purchase offer.

Stock market

The stock market gives the user a list of all joint stock companies registered on the network. Each corporation is listed out with all its shareholders and how many shares each person own. On the market users can also place offers on all shares.

On load the component send a GET request to the API asking for a list of all stocks on the network. The client then group the shares by corporation and owners inside each corporation. After grouping and sorting the client have an array with JavaScript objects containing all registered joint stock companies. Each of the JavaScript objects contain an array with JavaScript objects representing every shareholder in the corporation and how many shares they own. The market value of the given stock is also stored.

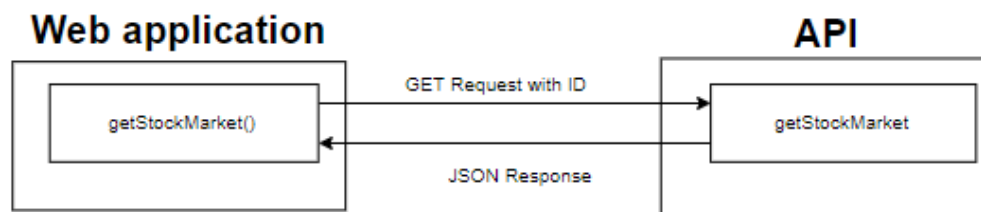


Figure 21: GET Request to get all shares NOT owned by current user. The response is filtered on the current users ID.

Each entry in the corporation's shareholder list is fitted with a button that lets the active user open a form for submitting an offer on a number of

shares. When pressed, a form opens and suppresses the rest of the application so that the user only can interact with the form. This form asks for input on how many shares the active user want to buy from the chosen owner and how much he / she is willing to pay for it. When all required fields are filled and valid the form opens for submission.

When the form is submitted the information inputted in the form is parsed to a JSON string and a POST request is sent to the API containing this JSON string. The API will then process the request and return a status code and message to tell the user if the offer was saved or not. If the offer is saved, the user that owns the shares the active user placed an offer on will get notified.

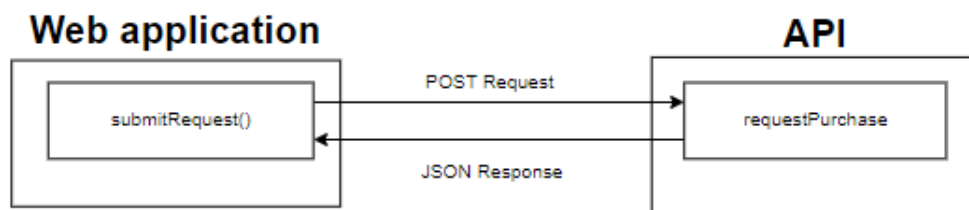


Figure 22: POST Request for giving an offer of given shares a person owns.

```
submitRequest () {

    var temp: {[k: string]: any} = {};

    temp.bid = this.buyStockRequest.value.bid;
    temp.quantity = this.buyStockRequest.value.quantity;
    temp.customer = this.userId;
    temp.stockOwner = this.buyStockRequest.value.stockOwner;
    temp.registerOfShareholders = this.buyStockRequest.value
        .registerOfShareholders;

    var body: string = JSON.stringify(temp);
    var headers = new Headers({ "Content-Type": "application
        /json" });
```

```

        //this._http.post("http://localhost:3000/api/
        requestPurchase", body, { headers: headers })
        this._http.post("http://aksjebok-hyperledger.localtunnel
        .me/api/requestPurchase", body, { headers: headers })
        .map(returData => returData.toString())
        .subscribe(
            retur => {
            },
            error => this._errorHandler.checkError(error),
            () => alert("Ditt bud har blitt registrert")
        );

        this.buyStockRequest.patchValue({
            bid: [null],
            quantity: [null],
            customer: [null],
            stockOwner: [null],
            registerOfShareholders: [null]
        });
        this.buyStockRequest.untouched;
        this.buyStockRequest.markAsPristine();

        let el: HTMLElement = this.myButton.nativeElement as
            HTMLElement;
        el.click();
    }

```

Listing 11: POST Request for giving an offer of given shares a person owns

Navigation

The navigation component contains links that route the user to the different pages available for stock owner accounts. It is included in all the components building the stock owner part of the web application.

Journalist client

This part of the web application is dedicated to registered journalists. Since a journalist have special rights of what information they can get from the blockchain, including history, they must be able to prove that they work as a journalist to get a journalist account.

After providing the needed information, the applicant will receive an

account and be able to log into the journalist part of our web application. Here the user can see current standings in the different corporations register of shareholders, all share owners, a transaction history of all exchanges of shares done and history of capital expansion.

The journalist part consists of three main components, a landing page, and a navigation component, as well as the footer we reuse on all the different parts of the web application.

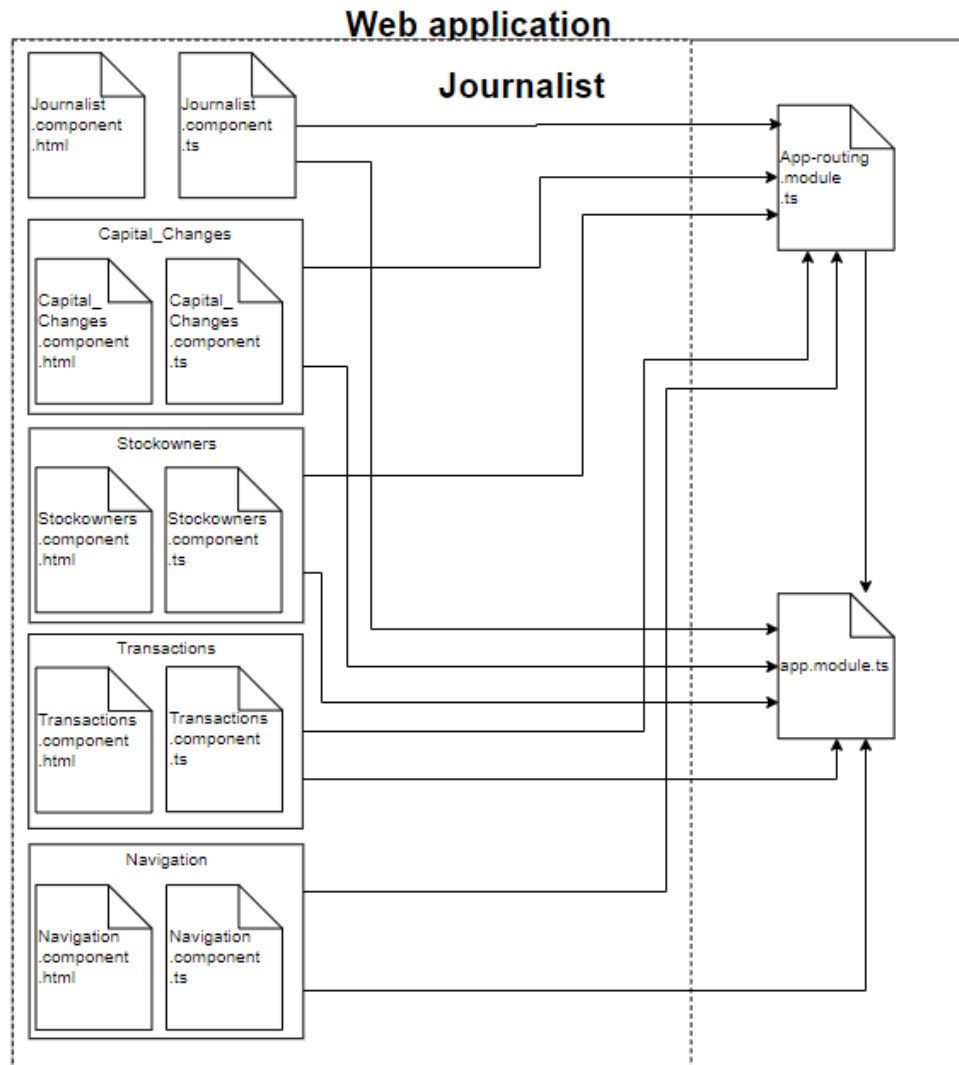


Figure 23: Overview of pages available to journalists. Black arrows symbolizes imports

Transaction history

The transaction history component is designed to let journalists see all stock exchanges that have occurred from birth to death of a share. Each

transaction is stored on the blockchain and contains information about who sells to who, for how much, how many shares and which corporation the shares belong to.

On load the component sends a GET request to the API to receive a list off all transactions of type stock exchange, sort them and puts them in a searchable list consisting of JavaScript arrays. The active user can then search the list for corporations and / or people to see the transaction history he / she wants.

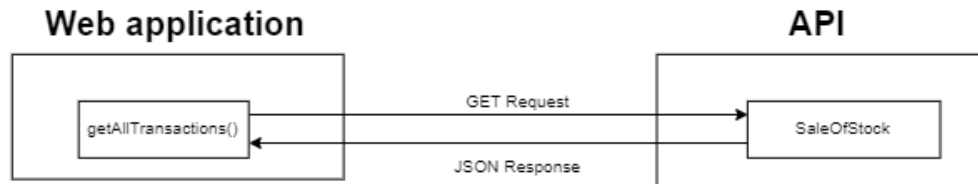


Figure 24: GET Request to the API asking for all stock exchanges that are registered on the network.

Stock owners

On the stock owner component journalists get an overview of all registered corporations, their present status, and a list of all shareholders in the corporation.

```
getStocks () {  
  //this._http.get('http://localhost:3000/api/Stock')  
  this._http.get('http://aksjebok-hyperledger.localtunnel.  
    me/api/Stock')  
  .map(result => {  
    let JsonData = result.json()  
    return JsonData;  
  })  
  .subscribe(  
    JsonData => {  
      if(JsonData) {
```

```

        this.allStocks = [];
        for (let s of JsonData) {
            this.allStocks.push(s)
        };
    };
    if(this.allStocks.length > 0){
        this.orderStocks();
    }
},
error => this._errorHandler.checkError(error),
() => console.log("Klientet har hentet alle aksjene som er registrert")
);
}

```

Listing 12: GET Request to retrieve all shares valid on the network

On load the web applications sends a GET request to the API asking for all shares registered on the world state of the blockchain and sorts by which corporation they belong to. This information is stored in an array of JavaScript objects which contain information about the current standing of the register of shareholders, a list of all shares and a list of all shareholders. The list of shareholders is empty at this point.

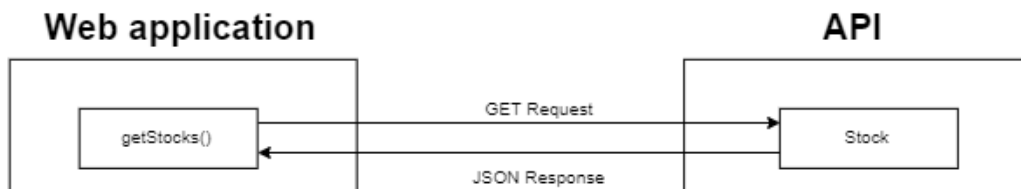


Figure 25: GET Request to API asking for all registered stocks

When all the shares have been placed in the correct JavaScript object, a sort function loops the list of all shares inside each JavaScript object (corporation) and creates new JavaScript objects for each shareholder inside the corporation and saves the object inside the list of all shareholders. These JavaScript objects contain information about who the owner is, how many shares he / she holds and the value of these shares.

Capital expansion

The capital expansion component is designed for journalists that wants to look at all the capital expansions a corporation has done.

On load the components sends a GET request to our API and asks for a list off all corporations registered on the blockchain. After sorting and placing this list in an array of JavaScript objects, a loop sends a GET requests and ask for a list of all capital expansions a given corporation has done. The response from the API is then stored inside an array of JavaScript objects inside its respectful corporation object.

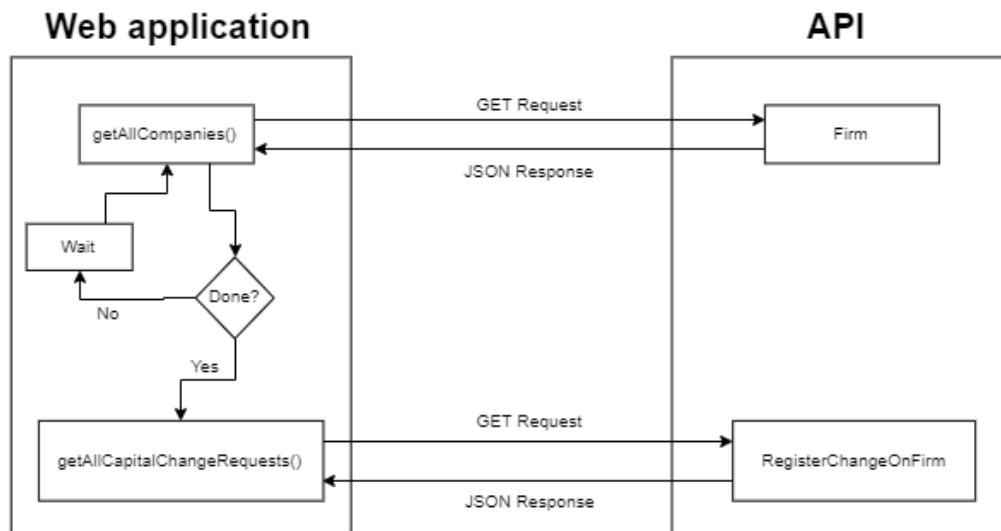


Figure 26: GET Request flow for all capital changes done on all registered joint stock companies on the network.

Navigation

The navigation component provides its HTML template with routes that link to the other components a journalist has permission to access.

Public client

The last part of the web application is an open part. It is design to give the public an easy way to get insights about all corporations and their shareholders as stated by law (Aksjeloven, 1997, §4-6).

On load the components sends a GET request to the API asking for a list of registered corporations on the blockchain network. The result from this request is stored as JavaScript objects in an array. After the response is received, the client loops the array and sends new GET requests and asks for a list of all shares registered to each corporation. The results are stored inside a new array of JavaScript objects inside the corporation object.

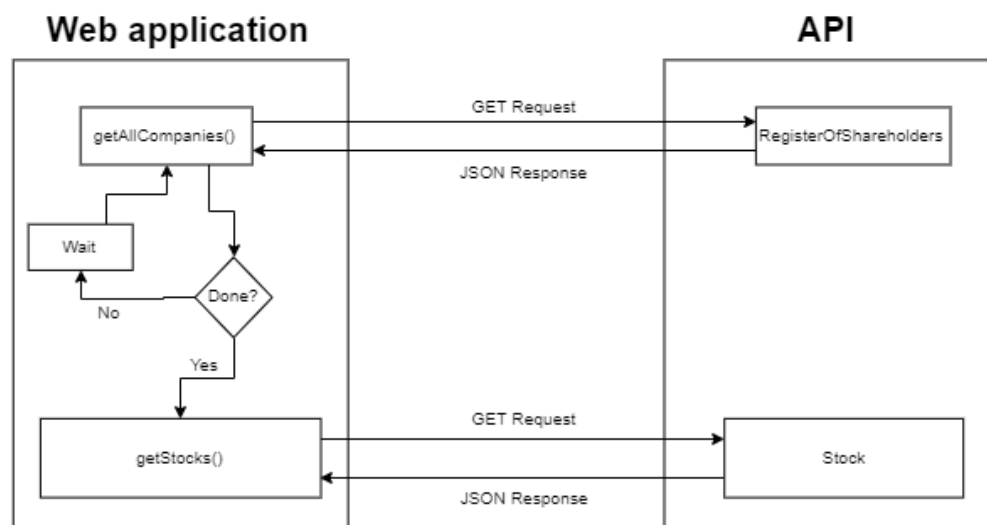


Figure 27: GET Request flow for the open web application.

When this response is received, a new loop sorts and organizes all shareholders inside each corporation and creates a new array of JavaScript objects that contain information of the owner, how many shares he / she holds and the value of these shares.

GUI

The graphic user interface (GUI) is designed to be in style with Altinn's (the client) styleguide. This styleguide is open source and can be found on

Github Altinn (Altinn, Design system - Styleguide). The guide is still in development but the developers have tried to use it in best possible manner.

Altinn's styleguide provides rules and components for their design and by using this we have managed to give a more consistent feel to users who use different web applications provided by Altinn or Brønnøysundregistrene.

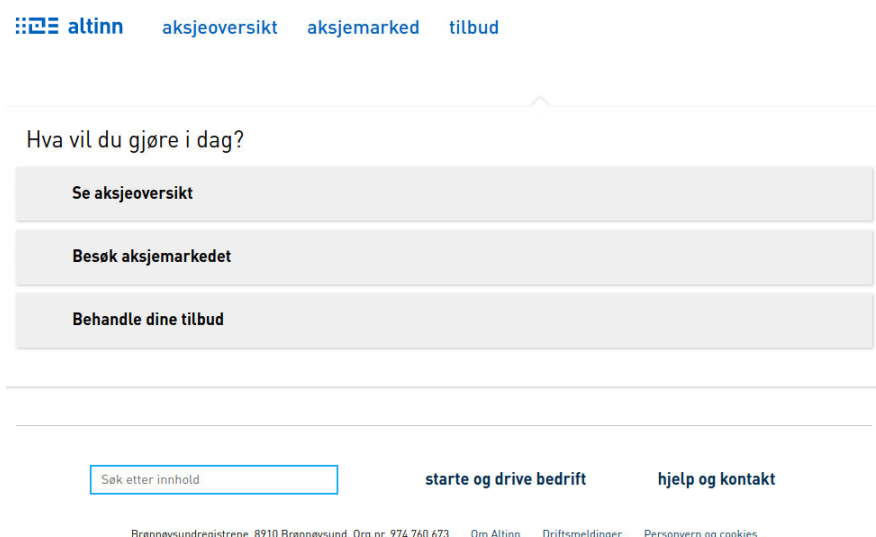


Figure 28: Screenshot: Landing page for stock owners

In figure 28 we see a screenshot of the landing page for an user logged in with a stock owner account. We can clearly see similarities to Altinn's official web application in figure 29.

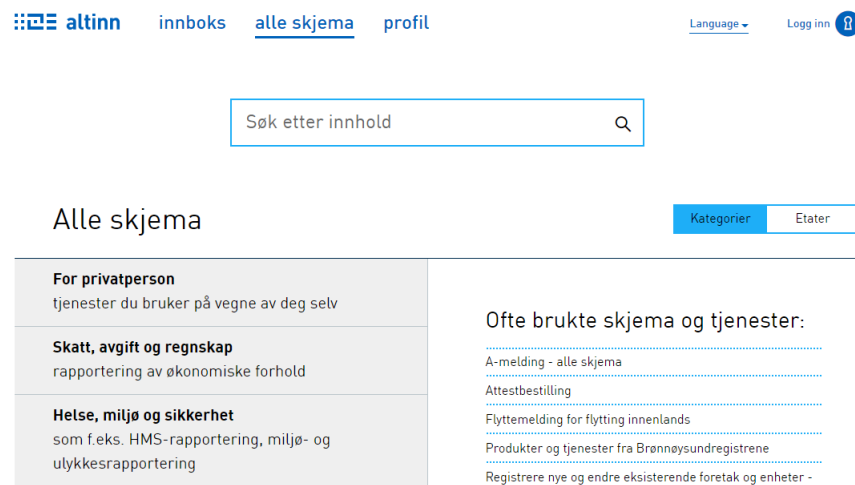


Figure 29: Screenshot: Altinn's official web application

Navigation

When designing the navigation bar, developers followed the guidelines provided by Altinn's styleguide (Altinn, Design system - styleguide). The styleguide provided the developers with predefined CSS classes and rules of where the different classes should be used. One of the rules that must be followed is that the website / application must follow the rules of universal design. This means that the application should be developed with diversity in mind. Therefore tab indexes should be logical and the contrast between text and background color should be high.

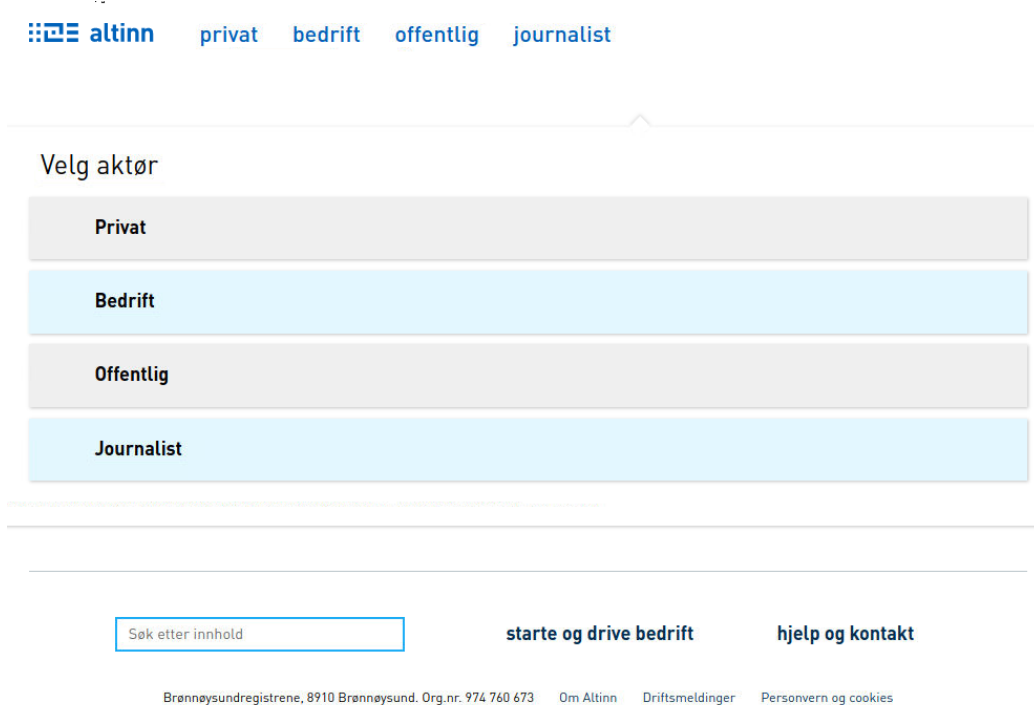


Figure 30: Screenshot: Landing page for stock owners

As we can see in figure 30 the background color is white (hex code: FFFFFFFF) and the text color is blue (hex code: 0062BA). This gives a contrast ratio of 6.08:1 and passes the AA requirement for normal text and both AA and AAA requirement for large text in the WCAG standard. (W3, WCAG Contrast minimum)

In figure 30 we can also see that options have different colors. This is due to a rule found in Altinn's styleguide (Altinn, Design system - styleguide) which says that options that are for private purposes should be of color gray and options for business purposes should be marked blue.



Figure 31: Screenshot: Landing page for stock owners

Another point is that the focused element is highlighted so that the user easily can identify what element currently is focused. This is solved by giving the focused element an orange border as seen in figure 31

Design components

Altinn's styleguide (Altinn, Design system - styleguide) do also provide different components and guidelines one where and how these components should be used.

Accordion

One of these components is an accordion component. This component is to be used when you want to give the user a choice of what information he / she wants to see on the website. For example, the developers used this component to display a corporation's register of shareholders as well as a list of all the current shareholders in the corporation on the same page.

As seen in figure 32 the user is presented with two accordions. If the user opens the accordion labeled 'Din aksjebok' a table with information about the registers current standings will be displayed. If the user opens the accordion labeled 'Liste over aksjonærer' will a list of all current shareholders, with information about how many shares they hold, be displayed. Only one of these accordions can be open at the same time, so if the user has one 'Din aksjebok' open and tries to open 'List over aksjonærer' the 'Din aksjebok' accordion will be closed.

Din firmaoversikt

Din aksjebok

ORGNR	KAPITAL	PÅLYDENDE	ANTALL AKSJER
123456789	80000	80	1000

Liste over aksjonærer

Søk etter innhold

starte og drive bedrift

hjelp og kontakt

Brønnøysundregistrene, 8910 Brønnøysund. Org.nr. 974 760 673 [Om Altinn](#) [Driftsmeldinger](#) [Personvern og cookies](#)

Figure 32: Screenshot: Business dashboard

Breadcrumb navigation

A breadcrumb navigation is a list that show where in the hierarchy the user currently is and provides links to the parent of the current displayed page. As seen in figure 33 Altinn's breadcrumb is styled with a blue underline on all parent pages and the current page is greyed out without underline.

[Startside](#) / Aksjeoversikt

Din aksjeoversikt

FIRMA	PÅLYDENDE	MARKEDSVRDI
123456789	80	80

Figure 33: Screenshot: Breadcrumbs

Table

In figure 34 we can see Altinn's design for tables. This is a responsive table that breaks on smaller screens as seen in figure 35. This component is used on pages where the user is presented with a list of information. This can for example be on in the stockowners stockwallet or on a corporations dashboard listing all current shareholders.

TH	TH 1	TH 2	TH 3	TH 4
body th 1	body td 1	body td 2	body td 3	body td 4
body th 1	body td 1	body td 2	body td 3	body td 4
body th 1	body td 1	body td 2	body td 3	body td 4
body th 1	body td 1	body td 2	body td 3	body td 4

Figure 34: Screenshot: Example of table layout

body th 1	
datatitle 1	body td 1
datatitle 2	body td 2
datatitle 3	body td 3
datatitle 4	body td 4

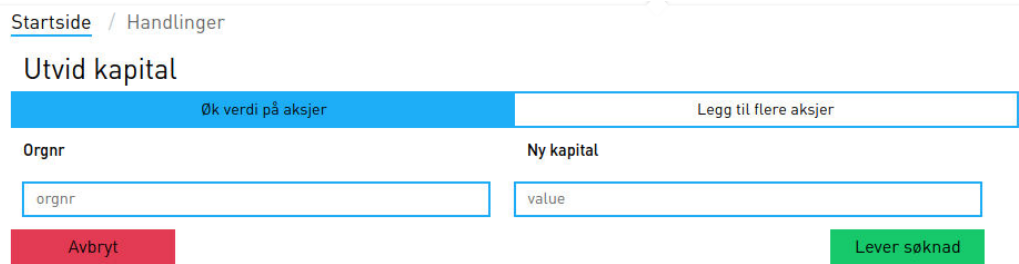
body th 1	
datatitle 1	body td 1
datatitle 2	body td 2
datatitle 3	body td 3
datatitle 4	body td 4

Figure 35: Screenshot: Table component small screen

Switch

The switch component is designed so that the user easily can switch between two different views. It should be used wherever the user should be able to decide what content he / she wants to display, but it is not suitable if there are more than two different views available to the user.

As seen in figure 36 developers chose to use the switch component to display the two different forms available for a corporation to expand its capital. On one form we ask for inputs on how many new shares are to be issued and the other asks for how much the corporation wants to increase their stock value with. When using the switch the application switches between these two forms.



Startside / Handlinger

Utvid kapital

Øk verdi på aksjer

Legg til flere aksjer

Orgnr

Ny kapital

orgnr

value

Avbryt

Lever søknad

Figure 36: Screenshot: Switch usage, expand capital

CSS/SASS

All stylesheets were provided by Altinn's styleguide on Github (Altinn, Design system - Styleguide). The stylesheet files are written in SASS (Syntactically awesome style sheets) and is free to use, edit and publish.

5 Testing

Testing is an important part of the development process. It helps developers identify bugs and remove them. This will in turn increase the overall quality of the product.

In this section we will cover what tests we have performed as well as their results.

5.1 User Testing

User testing is an important tool to ensure that new solutions work as intended but can also be used towards improving the efficiency of older and already existing solutions (Direktoratet for forvaltning og IKT, 2015). User tests are conducted by testers within the development team, who will then design what the tests should be and find suitable test subjects. In order to find relevant test subjects, it is important to carefully identify who the user group is. Depending on the product that is being developed, a detailed description might be necessary- identifying technical knowledge or any physical limitations can be of interest. Identifying what functionality that should be tested is another important step and is usually shaped or decided by the user case. An accurate analysis of who the user group is and what functionality that should be tested, is essential to retrieve data that will be relevant and generalisable. Data that is generalisable supports the external validity, and refers to whether the gathered data applies to other situations and persons (Svartdal, 2009).

The testing process

In this project we have used a mix between qualitative and quantitative methods through questionnaires and observation, to gather useful information. We split the information gathering into three parts, a pre-user test questionnaire, Observation and post-user test questionnaire. Prior to the user testing we had a test-agreement that introduced us and our project to the test subjects.

Contract

In the agreement we informed the test subjects of who we were, what we were working on and what the purpose of both the test and our product was. They were informed that they would stay anonymous and that any information that was gathered from them would not be traceable back to them. We informed them that any collected information would only be used for the purpose of this project. Furthermore, we informed the test subject that the information would be deleted within 30th of June 2018 and would only exist in the final bachelor thesis hand-in after that. Finally, they would have to give their consent through giving their signature and location. They were also given an email address they could use if they had any inquiries.

We are a group of 5 bachelor-students who are working on our Bachelor thesis in Applied Computer Technology (Anvendt Datateknologi) at OsloMet. The solution is based on the Blockchain-technology together with IBM's Hyperledger (Composer/Fabrics), and aims to make a decentralized system between public participants, shareholder and companies. The goal is to prevent fraudulent activity as well as increasing the process/communication-efficiency between the participants.

With this pre-questionnaire we wish to gather information that identifies who you are as a user, as well as factors that might influence the results. The experiment will contain a set of premade usecases and tasks that you will try to complete in the solution that we have made. By doing this we get to see if our solution work as intended, and if there is a natural and good flow, from a user's perspective.

NOTE: Information and results gathered through the questionnaire and the experiment will be stored temporarily, and used in our Bachelor Thesis. You will remain anonymous throughout this questionnaire and experiment- therefore any information or result will not be able to be traced back to you. After the thesis is completed and handed in the information that we have gathered through this questionnaire and experiment will only exist in the final hand-in, and will be deleted elsewhere. The information will be deleted by 30th of June 2018 and no later than this date.

Please sign if you have understood and consent to the use of information we get through the experiment with you as a test subject.

I understand and I consent: _____ Location: _____

Figure 37: User test contract

Pre-user test questionnaire

After the contract was signed by the test subject, he/she would independently answer the pre-user test questionnaire by circling around the suitable answer.

Gender: Male Female Other	Age group: 18-28 29 – 38 39 – 48 49 – 58 58+
Eyesight: Nearsighted Long-sighted Blind Uncorrected	Color blindness: Yes (What type?): _____ No

Handedness: Right Left Ambidextrous	Mobility impairment: Yes (What type?): _____ No
Comfort level with computers: Strongly dislike Dislike OK Like Strongly Like	

Figure 38: pre-user test questionnaire

The pre-user test questionnaire was primarily quantitatively oriented and meant to gather general physical information about the test subject. The two first questions would help us physically identify the person. The

following questions were directed towards potential physical limitations and whether the person was comfortable with using computers or not. The idea with these questions was to reveal potential factors that could alter or influence the test results. The question regarding the test subject's comfort level with computer, could reveal whether difficulties or significant efficiency regarding solving the given tasks could be explained by the familiarity of using computers. A high level of comfort would most likely be a result extensive knowledge and experience with computers. Thus, this can affect how fast or how easy a test subject solves a given task.

User tests and Observational Data

When the pre-user test questionnaire was completed we would have three participants from our group help with the user tests and gathering of observational data. One of us would orally instruct the test subject to conduct a variety of tasks, to the best of their ability. The tasks were the following:

- Create a stakeholder book
- Increase the amount of shares
- Add value to the shares
- Buy/sell shares from/to another shareholder
- Find an overview of all your shares and information about them
- Approve/Decline creation of stakeholder book

When the test subject would conduct tasks within the company scope of the application, the test-leader would need to take on the role of Brønnøysundregistrene to either accept or reject the request of, for example, creating a stakeholder book to achieve a complete and dynamic process.

While the test subject was conducting the test, another participant from our group would observe the test subject and look for any cues through body language or nonverbal communication that would imply that the subject was either struggling, distracted, confused or seemed to understand

how to use the application. According to Birdwhistell, 1952, 1970; Efron, 1941; Ekman and Friesen, 1969b; Exline and Winters, 1965; Hall, 1959, 1963, 1966; Kendon, 1967a; Mehrabian, 1971c, 1972; Schefflen, 1964, 1965, 1966 and Sommer, 1969 (as cited in Mehrabian, 2007, p. 1) the body language can through posture, positioning and facial expressions work as an external reflection of the test subjects emotional state and therefore give a genuine picture of how the user experiences both the application and the process through each task. It can reveal whether a person is comfortable and confident, or if the person is in some sort of distress that might affect the test performance.

The questions were therefore shaped towards what kind of emotions and body language the subject expressed throughout the test. Furthermore, we looked for factors such as if the person seemed relaxed, if the person asked many questions- and if so, about what. Answering these questions would reveal if there was something that was not easy for the user to understand, and then specifically what that was. This would clarify if it was the application or if it was the instructions that were the problem.

Questions	Observations
Does the subject seem relaxed/at ease in the situation?	
Does the subject appear to be used to using computers?	
Does subject ask a lot of questions? If so, specify if questions are regarding the solution, or the given instructions	
Does the subject talk during the test? If so, what is the test subject saying?	
What kind of emotions does the subject express?	
What kind of body language does the subject have during the test?	
Does the subject spend much time trying to solve each task?	
Was there any specific point throughout the test that the test subject seemed to struggle more than expected?	
Did the subject complete the test? If not, please specify why not- and how far the subject got	

Figure 39: Observational data form

The last participant from our group would have a secondary observant role as well as helping or give the test subject technical guidance if needed. Having a secondary observant meant that we got observational data from two different points of view and would therefore either support or contradict the observations that were made. Consistency in the observations would argue that the observations had been correct, while inconsistency could mean that the observations were rather subjectively influenced and incorrect.

Post-user test questionnaire

This part of the user-testing would be conducted by the test subject

individually, after completing the tasks. In this questionnaire we wanted the subject to evaluate how they experienced the application. The questions asked for the test subject's subjective and detailed opinion about the application. This provided us with valuable information and feedback. We wanted to know what the subject liked and disliked about the application, if something could have been done differently- how, and so on. Finally, we asked for the subject's overall evaluation of the application, and if he/she had any additional comments. These qualitatively oriented questions would give us deeper and more detailed information about the application. Not only does it provide essential information towards improving the application, it is the key information that we look for through user testing.

Questions	Answers
What did you like about the product?	
What did you not like about the product?	
What could have been done differently, and how?	
Did the product have a natural and logic flow to it?	
If not, what should be done to achieve this?	
Does the product lack any important elements from your point of view?	
If so, please specify.	
What is your overall evaluation of the product?	
Any additional comments?	

Figure 40: Post-user test questionnaire

Method

The methods used to gather information through the user testing has been a mix between qualitatively and quantitative research methods and will be

further explained in the following paragraphs.

Selection

The test subjects have been individuals from Brønnøysundregistrene, Altinn and IBM, as we conducted the user test at the end of one of the showcases. This means that we have had test subjects that are a part of the user group, however the selection has not been randomized. Ideally, we would have had a randomized selection to include the average user to test people who might not have the same technical competence as our test subjects. Furthermore, we would have benefited from having a higher number of test subjects. This means that the data would have a better chance of being generalisable the user group and essentially lead to us making more accurate and useful decision in the development of our application, or future changes (Svartdal, 2009). However, we have benefited from highly competent people who has given valuable feedback and advises based on years of knowledge and experience.

Quantitative research

The pre-user test questionnaire is characterized by confined questions as we wanted short and accurate answers that would help identify the test subject as well as survey any physical limitations. At the end of this questionnaire we used a 5-step Likert-scale where the test subject was asked to rate their comfort level regarding computers. The scale goes from Strongly dislike to Strongly like and does not give any further details why the subject feels the way he/she does. It simply indicates to what degree the test subject is comfortable with computers. Yet it provides information that might explain any potential issues with completing the tasks that were given.

Qualitative research

The qualitative methods used in the user testing has been through observation and questionnaire. The way these two parts of the test differ from the pre-user test questionnaire, is that they have been designed to

gather more detailed information about each of their experience with the application. They have been used in a different area of the testing with a purpose of giving us essential information regarding what is working well, and what should be differently. Furthermore, it has given us information about how the test subjects thinks we should have done things. This is feedback that is crucial towards making a product that provides the future users with a comfortable and efficient user experience.

Results

Pre user test questionnaire

The pre-user test questionnaire indicates that the test subjects have been of both genders and all in the age-group of 39 to 48. All the test subjects are near-sighted which could affect the performance. However, as the observational data will reveal, none of the subjects complained or had difficulties connected to eyesight and are therefore likely not to have affected the test subjects significantly. None of them suffered from colour-blindness either. Two out of three subjects were right-handed. The subject who was left-handed did not report any difficulty using the application with a mouse and keyboard. This subject also reported that her comfort level with computers were equal to “Strongly like”, which indicates that she is used to and experienced with using computer. This might explain why being left-handed was not an issue. All subjects reported that did not have any mobility impairment. Two out of three reported that they “strongly like” computers, regarding the comfort level, while the last subject reported he would rank his comfort level as “liked”.

This information creates a picture and confirms that these subjects offers a representation of the user group. Even though the test group was small, there is some diversity in gender, hand preference and comfort level regarding computer use.

Observational data

For the observational data there were some observations that was consistent across the subjects. They were all curious and interested in testing the

application. One seemed to be a little stressed and expressed that she was under time pressure as she had to leave for a meeting. Other than this, they appeared to be confident and comfortable in the test situation as well as how to use computers. This aligns and confirms the information the subjects gave in the pre-user test questionnaire. Regarding whether the subject asked questions throughout the test, there was some diversity. One subject asked several questions about what to fill into the different input-fields in the application and received further instructions. This signifies that it could be useful with informational cues, in shape of pop-up info-boxes or similar notifications to clarify and guide the user through the first time. Two of the subjects did not talk much during the tests except for the questions, and that one of the subjects discovered an issue regarding calculations based on user input in two fields. The last subject talked some during the test and could appear slightly unfocused at times.

Regarding their emotional state they appeared to be positive and eager to participate. This became apparent through their body language too. Most of the time they were leaning towards the computer, looking for clues and seemed engaged in solving the tasks. In addition to this the subject that specified she was on a time schedule, revealed some level of stress and distraction by checking what the time was regularly. Regarding the two last questions in the observation part, the subjects spent about as much time as each other and did not get locked at a certain point during the test. One of the subjects was noticeable more thorough and wanted to explore beyond the tasks that were given. It was interesting to see the subject explore as this meant he would test out application outside the intended scope. This was the same subject who discovered an error connected to calculations. Two out of three subjects completed the user test, while the application crashed during the user test to the subject who had little time to complete it.

Post user test questionnaire

The test subjects thought it was easy to get an overview in the application, that it had a good flow and that it had a good potential for the future. Regarding things they did not like, they thought there were too few indicators- that it was either lacking or too small, the application was

missing authentication, some GUI error and some functionality was not working the way it does in real life. Two of the subjects reported that they would have preferred written use case for the test rather than getting the instructions verbally. The third subject did not report any constructive criticism towards what could have been done differently. On the question if they thought the product had a natural and logic flow to it- one of the subjects thought it did. The others wanted “the logic more woven together” (being able to register capital and shareholder at the same point and time in the application), as well as the last subject wanted a simpler way to get to the approval page and suggested a link in the navigation bar. On the question whether they thought the application was lacking any important elements, two left a blank answer while the last subject stated that the application was unfinished.

Their overall evaluation of the application was that it was “good being a bachelor thesis”, “Very good” and that “it can be used to explain the technical solution”, referring to the use of blockchain-technology. Regarding the question if the test subjects had any additional comments, they said they “would have liked to be able to see previous value after changing it” (stock value), and that the application was lacking some logic concerning adding stocks. The test subject wanted to be able to increase capital and add stocks in one action. Furthermore, he wanted to be able to decide who owns what stocks. And finally, he thought the application needed more custom error-handling towards the user, and that the stock-owner listing was clunky.

Conclusion

As the information throughout “User testing”-section reveals, we have tested test subjects that represents the user group well. They have provided us with useful feedback and information, both through testing our application but also by answering our questionnaires. Their background with years of knowledge and experience makes them more than qualified towards giving feedback on what should be included and what should be done differently. This accompanies the user testing in a fulfilling matter. The information we have gotten through user testing would be used for further development and finalizing the application. As a group we have

reflected our user-testing approach and see that by increasing the number of test subjects the data would be more generalisable regarding the user group. Furthermore, the questions in the questionnaires could have been more specific and asked more directly about certain elements as this would clarify and make the improvement process further on, easier. In the pre-user test questionnaire we could also have included a question regarding what "participant-role" they identified themselves with. For example, if they identified themselves as a shareholder, company owner, or someone from Brønnøysundregistrene. This would help us further in understanding who our test subjects were, and what functionality they might want or expect.

Specific changes to be made to the application based on information from the user testing:

- Additional GUI cues or information to the user when processes have been completed, and error-handling
- Show previous stock value after changing it
- Allowing the user to register capital and shareholder in the same process instead of in two separate processes.
- Add a navigation option to the navigation bar so that the user has easy access to the "Approval Page".

Gathered user-test information

Go to the [appendix](#) to view each of the test subjects answers and information

5.2 Unit test

Unit testing is used to test the behavior of a small specific part of the code, such as a function/method. The unit test asserts that the expected result matches the actual result. If this is not the case, the test will fail, and you can easily find the code that failed.

Testing tools for Hyperledger Composer

Mocha

Mocha is a JavaScript test framework running on Node. It is used to run our chaincode as an embedded runtime and is executed as unit tests. This allows for unit testing without running it on the fabric runtime.

Chai

Chai is an assertion library for Node that can be paired with any JavaScript testing framework.

Test setup

To have a isolated testing environment for composer code must one create a simulated/embedded runtime with the business network. Different steps are required for this to happen.

Connection profile

Connection profile is used when creating the different identity cards. It is necessary so the connections to the business network know that it is a embedded runtime.

```
// Embedded connection used for local testing
const connectionProfile = {
  name: 'embedded',
  'x-type': 'embedded'
};
```

Listing 13: Js code of function 'before'

Creating a peerAdmin

Embedded runtime is similar to the fabric runtime so the business network must be deployed to it. And this is done by the peerAdmin.

To begin the creation of a peerAdmin the method 'before()' is used with a callback function. It creates the peerAdmin identity card and a 'adminConnections'.

```

before(async () => {
  // Generate certificates for use with the embedded
  connection
  const credentials = CertificateUtil.generate({ commonName: '
    admin' });

  // PeerAdmin identity used with the admin connection to
  deploy business networks
  const deployerMetadata = {
    version: 1,
    userName: 'PeerAdmin',
    roles: [ 'PeerAdmin', 'ChannelAdmin' ]
  };
  const deployerCard = new IdCard(deployerMetadata,
    connectionProfile);
  deployerCard.setCredentials(credentials);

  const deployerCardName = 'PeerAdmin';
  adminConnection = new AdminConnection({ cardStore: cardStore
    });

  await adminConnection.importCard(deployerCardName,
    deployerCard);
  await adminConnection.connect(deployerCardName);
});

```

Listing 14: Js code of function 'before'

Deploying the business network

With the established peerAdmin one can go ahead deploying the business network. This is done by the method 'beforeEach()', also used with a callback function.

Both a 'businessNetworkDefinition' (referring to the model file) and a 'businessNetworkConnection' is created. This is used by the peerAdmin to deploy and start the business network.

```

beforeEach(async () => {
  businessNetworkConnection = new BusinessNetworkConnection({
    cardStore: cardStore });

  events = [];

```

```

businessNetworkConnection.on('event', event => {
    events.push(event);
});

const adminUserName = 'admin';
let adminCardName;
let businessNetworkDefinition = await
    BusinessNetworkDefinition.fromDirectory(path.resolve(
        __dirname, '..'));
businessNetworkName = businessNetworkDefinition.getName();

// Install the Composer runtime for the new business network
await adminConnection.install(businessNetworkDefinition);

// Start the business network and configure an network admin
identity
const startOptions = {
    networkAdmins: [
        {
            userName: adminUserName,
            enrollmentSecret: 'adminpw'
        }
    ]
};
const adminCards = await adminConnection.start(
    businessNetworkDefinition.getName(),
    businessNetworkDefinition.getVersion(), startOptions);

// Import the network admin identity for us to use
adminCardName = `${adminUserName}@${
    businessNetworkDefinition.getName()}`;
await adminConnection.importCard(adminCardName, adminCards.
    get(adminUserName));

// Connect to the business network using the network admin
identity
await businessNetworkConnection.connect(adminCardName);

...
});

```

Listing 15: Method 'beforeEach()' deploying business network

Seeding the business network

The 'beforeEach()' method callback function allows for creating other data as well. We use this opportunity to create test data for later use. This does so we do not have to initialize new data for each single test.

```
beforeEach(async () => {  
  
    ...  
  
    // Get the factory for the business network.  
    const factory = await businessNetworkConnection.  
        getBusinessNetwork().getFactory();  
  
    // Create Firm "ACME AS" #112233 with RegisterOfShareholders  
    and two Stocks  
    const firm1 = await factory.newResource(namespace, "Firm", "  
        112233");  
    firm1.firmName = "ACME AS";  
    firm1.awaitingStockPurchase = [];  
    firm1.establishFirmRequest = [];  
    firm1.changeOnFirmRequest = [];  
  
    const firmRegistry = await businessNetworkConnection.  
        getParticipantRegistry(namespace + '.Firm');  
    await firmRegistry.add(firm1);  
  
    // Issue the identities.  
    let identity4 = await businessNetworkConnection.  
        issueIdentity("org.acme.biznet.Firm#112233", "ACME");  
    await importCardForIdentity("ACME", identity4);  
  
    // Create and add the assets.  
    const registerOfShareholders = await factory.newResource(  
        namespace, "RegisterOfShareholders", '112233');  
    registerOfShareholders.numberOfShares = 2;  
    registerOfShareholders.capital = 100000;  
    registerOfShareholders.firstDenomination = 50000;  
    registerOfShareholders.approved = true;  
    registerOfShareholders.belongsTo = await factory.  
        newRelationship(namespace, 'Firm', '112233');  
  
    const registerOfShareholdersRegistry = await  
        businessNetworkConnection.getAssetRegistry(namespace + '.  
        RegisterOfShareholders');  
    await registerOfShareholdersRegistry.add(  
        registerOfShareholders);  
}
```



```
});
```

Listing 16: Method 'beforeEach()' using commonAPI to create test data

Changing identity

Access control should be a part of the unit testing as well as the functionality. But an extra step is required for this to happen. As demonstrated in the previous paragraph the runtime is executed, deployed and connected to as a admin. This means that every transaction executed is done so by the admin. And we want to execute transaction as ordinary users to test access control. So a identity change is required.

The method 'useIdentity(card)' is used for the identity change (17). It disconnects as admin and connects with a given identity card. For example as the firm "ACME" that we created in 'beforeEach()'.

This is done before submitting transaction during tests.

```
async function useIdentity(cardName) {
  await businessNetworkConnection.disconnect();
  businessNetworkConnection = new
    BusinessNetworkConnection({ cardStore: cardStore });
  events = [];
  businessNetworkConnection.on('event', (event) => {
    events.push(event);
  });
  await businessNetworkConnection.connect(cardName);
}
```

Listing 17: Js code of the function 'useIdentity'

Unit test execution

Traditionally unit tests are divided into the steps 'arrange', 'act' and 'assert'.

Arrange

This part consist of choosing identity and creating the transaction to be tested. This is done by executing 'useIdentity()' with a identity card and 'factory.newTransaction()' with required input.

```

const factory = businessNetworkConnection.
  getBusinessNetwork().getFactory();
const stockRegistry = await businessNetworkConnection.
  getAssetRegistry(namespace + '.Stock');

// create and submit the changeStockValue transaction
const changeStockValue = factory.newTransaction(
  namespace, 'ChangeStockValue');
changeStockValue.newValue = 10000;
changeStockValue.registerOfShareholders = "112233";

```

Listing 18: Js code of the test 'changeStockValue'

Act

Submitting the transaction.

```

await businessNetworkConnection.submitTransaction(
  changeStockValue);

```

Listing 19: Js code of the test 'changeStockValue'

Assert

In the final part we assert the values we expect the transaction will produce. As transactions might emit events we have to assert expected event values as well.

```

// Stock value should return 10000 after transaction
let registerCap = await stockRegistry.get("1");
registerCap.value.should.equal(10000);

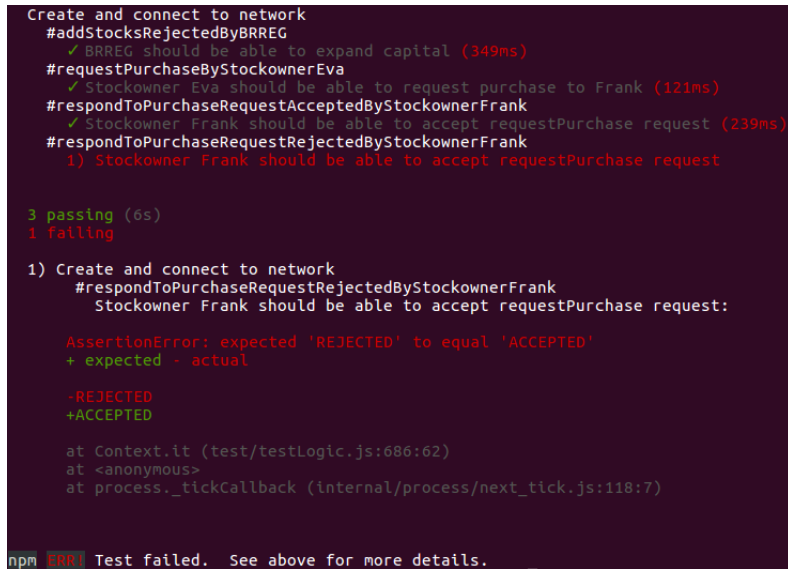
let registerCap2 = await stockRegistry.get("2");
registerCap2.value.should.equal(10000);

// Check event data after transaction
events.should.have.lengthOf(1);
const event = events[0];
event.oldValue.should.equal(50000);
event.newValue.should.equal(10000);

```

```
event.message.should.equal("Stock has changed value.");
```

Listing 20: Js code of the test 'changeStockValue'



```
Create and connect to network
#addStocksRejectedByBRREG
  ✓ BRREG should be able to expand capital (349ms)
#requestPurchaseByStockownerEva
  ✓ Stockowner Eva should be able to request purchase to Frank (121ms)
#respondToPurchaseRequestAcceptedByStockownerFrank
  ✓ Stockowner Frank should be able to accept requestPurchase request (239ms)
#respondToPurchaseRequestRejectedByStockownerFrank
  1) Stockowner Frank should be able to accept requestPurchase request

3 passing (6s)
1 failing

1) Create and connect to network
   #respondToPurchaseRequestRejectedByStockownerFrank
     Stockowner Frank should be able to accept requestPurchase request:

    AssertionError: expected 'REJECTED' to equal 'ACCEPTED'
    + expected - actual

    -REJECTED
    +ACCEPTED

    at Context.it (test/testLogic.js:686:62)
    at <anonymous>
    at process._tickCallback (internal/process/next_tick.js:118:7)

npm ERR! Test failed.  See above for more details.
```

Figure 41: Screenshot: Expected value not equal to actual value

Executed tests

In figure 42 you can see all the executed unit tests, and which ones failed and passed. In the appendix under [unit tests step-by-step](#) you will find an in depth description of all the tests, structured with arrange, act and assert.

```

Create and connect to network
#CreateCompanyByBrreg
1) BRREG should be able to submit transaction CreateCompany
#registerChangeOnFirmCHANGEVALUEByStockOwnerFrank
✓ StockOwner Frank should not be able to RegisterChangeOnFirm (90ms)
#registerChangeOnFirmCHANGEVALUEByFirmTestfirma
✓ Firm Testfirma should be able to RegisterChangeOnFirm (122ms)
#registerFirmWrongAmountOfStocks
✓ should not be able to RegisterFirm with distribution with wrong amount of stocks (79ms)
#registerFirmWrongAmountOfStockowners
✓ Should not be able to RegisterFirm with wrong distribution (82ms)
#registerFirmByTestfirma
✓ Firm Testfirma should be able to use transaction RegisterFirm (154ms)
#expandCapitalAcceptedByBRREG
2) BRREG should be able to expand capital
#expandCapitalRejectedByBRREG
3) BRREG should be able to expand capital
#expandCapitalWrongCapitalChange
✓ BRREG should be able to expand capital (227ms)
#addStocksWrongCapitalChange
✓ BRREG should be able to expand capital (198ms)
#addStocksRejectedByBRREG
4) BRREG should be able to expand capital
#requestPurchaseByStockownerEva
✓ Stockowner Eva should be able to request purchase to Frank (119ms)
#respondToPurchaseRequestAcceptedByStockownerFrank
✓ Stockowner Frank should be able to accept requestPurchase request (199ms)
#respondToPurchaseRequestRejectedByStockownerFrank
5) Stockowner Frank should be able to reject requestPurchase request
#SaleOfStockAcceptedByFirmACME
6) Firm ACME should be able to accept SaleOfStock request

9 passing (16s)
6 failing

```

Figure 42: Screenshot of all tests executed

Challenges

There were some problems that occurred that made the test process challenging.

Dependency versions

As the different tools often were dependent on each other conflicts often occurred. A tool might require a certain version of a dependency but that very version might not be compatible some other dependency etc.

When the technology is new as well means that proper documentation is hard to come by. This forced us to go about it with a 'trial and error' approach.

Code coverage

One example of a tool we were not able to implement is 'Istanbul' which is a code coverage with JavaScript. The version of 'Istanbul' that were

available did not support asynchronous functions. All the created tests are asynchronous which made 'Istanbul' unusable.

Unindexed database

On the fabric runtime we configure the database to index its data. We were not able to do this on the embedded runtime. This resulted in failing some of the tests. The transactions that consist of queries with the command 'ORDER BY' return data in the indexed order. This order is wrong on the embedded runtime.

Concepts

Participants consists of different response concepts that we created. The concept structure were first introduced in the Hyperledger composer v0.19 version. We started unit testing at v0.16. This means that concepts are not available in the embedded runtime we use. Which results in the test alert failure on updating participants even though it works on the fabric runtime.

5.3 System test

The purpose of system testing is to confirm that all specified requirements are met, and that the product behaves in an expected manner. System testing is a form of "black box" testing, which means that the internal structure of the system that is being tested is not necessarily known to the tester. And that the tests focus on finding errors that a user would be able to see, for example interface errors (Software Testing Fundamentals, 2018).

We built the tests in such a way that we can simulate an end-user scenario. In these tests we used a browser plug-in called Selenium IDE, which lets us create predefined sequences of actions that simulate real use. It also has a record function, that lets us record a run-through of the application, which then Selenium stores as a complete test. This alongside the possibility for custom scripted tests makes for a very powerful tool. Selenium is built so you can run your tests multiple times with little to no input needed from our side. This allows us to run the same test multiple times and makes it easier to see if changes done to the system have introduced new bugs.

The tests is built to cover all the functionality of our application, and uses two of the main clients to do so: stock owner and business.

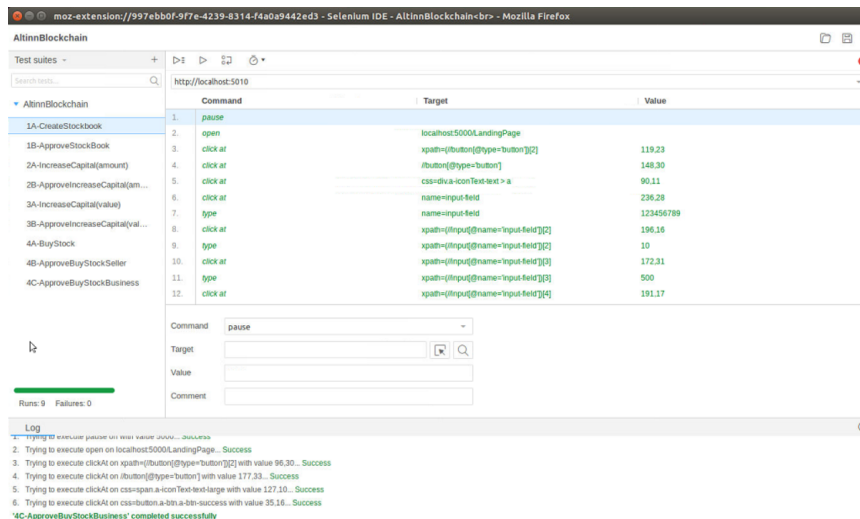


Figure 43: Screenshot: After a successfully test run in Selenium

Figure 43 will show you a overview of Selenium. On the left you can see our "test suite" AltinnBlockchain. Test suites is a collection of single tests that can be run in quick succession. In total we have 13 different tests, which when run together covers all of the functionality in our application. Using a naming convention we can order them as we wish. In this test we used a number followed by a letter to group the different tasks and order them correctly. You can see that the two tests that create the register of shareholders are named 1A and 1B, because they need to be run before any of the other tasks, and because they need to be run in that specific order. As we also had to test the denial of different actions, such as creating a register of shareholders, we had to add extra tests with prerequisites. In the cases where a test have a prerequisite, the corresponding prerequisite runs before the test itself.

Listing of functionality that has been tested

- Public web application
 - Business user

- * Submit application for registration of a register of shareholders
- * Submit application for issue of new shares
- * Submit application for capital expansion
- * Approve stock exchange between two users on company shares
- * Deny stock exchange between two users on company shares
- Private user
 - * Submit offer on shares listed on the stock market
 - * Respond to offer on shares by approving it
 - * Respond to offer on shares by denying it
- Brønnøysundregistrene web application
 - Approve application for registration of a register of shareholders
 - Approve application for issuing new shares
 - Approve application for capital expansion
 - Deny application for registration of a register of shareholders
 - Deny application for issuing new shares
 - Deny application for capital expansion

Test cases step-by-step

To system test our two application we created multiple test cases. In the [appendix](#) you will find a step-by-step guide that shows the different steps each selenium test was programmed to complete on the different web applications.

Results system testing

The system tests was completed without any critical errors or bugs. This confirms that our platform runs in a expected manner. The only hurdle was that our application is not as fast as we would have wanted it to be, mostly due to running on virtual machines, we added five second pauses into the Selenium sequences as a workaround, so that the test would wait for the program to load properly.

6 Final result and evaluation

In this section we will focus on highlighting the results of this project. The developed product, further potential and the process. Both what we did and did not manage to do as well as what we learned from it.

6.1 Process

During the course of this project we have gained a lot of new insight and experience about development as a whole. Using what we had previously learned about agile development methodology combined with good guidance, we planned and carried out a development project.

Agile methodology

As mentioned in 'the process' section we planned the whole process and divided it into sprints. Every sprint had tasks with time estimation and every sprint was going to be concluded in a playback with the customer. We believe this methodology helped us achieve the results that we are satisfied with. Even so that we could have committed more to it.

Sticking to the plan

We tried to stick to the plan as much as possible. The original tasks and estimations were in general helpful to get going. But throughout the project we gained new knowledge of the technologies. This resulted often in exposing ignorance the original tasks were based on and required us to adapt and change some of the tasks.

Here is an example of a trello-board we corrected:

'Establish shareholder registry' were assumed to consist these tasks

- Define the shareholder registry asset in the model
- Define the transaction
- Create the transaction function
- Consensus from Brønnøysundregistrene

Most of these tasks were correct with the exception of 'Consensus with Brønnøysundregistrene'. The process of establishing a shareholder registry requires a consent from Brønnøysundregistrene. As we were ignorant on how consensus actually worked in a blockchain we assumed that consensus should play a part in the consent. This was not entirely true by the fact that the consensus mechanism only assures that the same pre-agreed code is executed on every peer and not a real time consent. We learned that the consent from Brønnøysundregistrene had to be a part of the transaction flow as demonstrated in the product documentation.

We had multiple tasks assuming that consensus had to be applied and not a inherit functionality of the hyperledger blockchain. Discovering things like this made a lot of the tasks invalid as well as declaring new tasks we were not previously aware of.

As pursuing to solve these invalid tasks resulted in the discovery of new knowledge and more valid tasks means that even though they were invalid it does not mean that they did not serve a purpose. With newly gained knowledge we were able to adapt the development process accordingly. With this being said we could have committed to updating the trello-boards more often. Some of the new tasks had a tendency to be a vocal agreement between the group members.

Distributing tasks and responsibility

Every group member had a area of responsibility in the project. Even though everyone contributed more or less in every area. This responsibility consisted of understanding the technology in use as well as making sure of progress. The different responsibilities were distributed by itself in a natural manner when every group member pursued the different tasks. This worked out fine for us as no one were particularly unhappy with their responsibilities and all did a good job.

Playbacks

Every sprint was finalized with a playback with the customer. We managed to have playback for all the five sprints we completed. The playback consisted of a walk through of the trello-tables and tasks of the sprint.

Furthermore what tasks were accomplished, delayed or even deprecated.

The customer would then express their opinions of the work as well as instructing what they wanted us to prioritize. For example that they wanted us to emphasize the blockchain/hyperledger aspect of the project more than the frontend aspect. Even though they understood that both were important for the project as a whole.

We finished every playback with a Q/A session as the customer were eager to gather information on blockchain and the potential it brings.

All in all the playbacks helped us organize the development throughout the project as well as keeping us on edge and willing to keep every deadline the sprints consisted. It also gave a lot of experience presenting our work and communication with a customer.

Retrospectives

Retrospectives was not used as much as we planned. This was unfortunate because when we did we found it helpful.

Different problems became apparent to us through retrospective, and helped us change "unhealthy" working habits. Furthermore, this improved our work flow as well as the quality of our work. We experienced less problems in the same areas as before. Typically these problems were connected to being unfocused, as a result of too much work throughout a day, and too few and short breaks.

We started using retrospectives rather late in the project. After experiencing the effects it gave we regretted not doing it more frequently from the beginning.

Dealing with new technology

Blockchain is a technology that has existed for some time. But the blockchain tools we used has not. Hyperledger fabric is the only tool that was production ready when we started the project while Composer became production ready at the end of it.

Being that most of our tools were so fresh and little tested some challenges naturally appeared.

Uncertain documentation

The general documentation for both Fabric and Composer were indeed helpful. We came a long way with the documentation itself. The problems appeared when unexpected errors and bugs started turning up. Or even when it did not turn up but code not working as intended. It turned out that documentation not always was up to date which meant that we had to go about finding solutions by our self. Hyperledger forums like 'rocketchat' and 'slack-sub' were helpful but receiving valuable responses could take a while.

This was particularly problematic in the early phases of the project. When we were supposed to have a development environment ready and start developing chaincode for deployment to the blockchain. Everything did not always work out and we spent a lot of time getting started. The consequence of this being delaying some of the sprint tasks as it was hard to estimate how much time these problems consumed.

Frequent updates

New updates to the tools were released frequently. Especially Hyperledger Composer. We tried to version lock as much as possible but at times we were more or less forced to update our system. At some point halfway in the project Hyperledger Composer released version 0.19.1. This version was released as 0.19 for a test period only to be renamed version 1.0 being the first production ready version. Being such a big change the Composer team removed all the previous documentation and changed it with the new 0.19 documentation. Since so many of the changes were breaking and the documentation for our version was out off reach we had no choice other than upgrading our system.

This update required us to spend some time changing our code. Our queries had to be restructured, the connection profiles changed, etc. This was also something we had failed to account for in our estimate.

All though the updates were time consuming we still found the new version better than the previous. The time spent was more or less worth it. One example being that the chaincode upgraded the JavaScript from ES5 to ES6 which gave us the possibility to use arrow functions, async and await, etc. This gave us more opportunities when developing chaincode.

6.2 Product

The product result is based on the product specification requirements. We evaluated whether a requirement were met based on the different tests that we did.

Functional requirements

Id	Prio	result
1	A	Both backend and frontend functionality allows users to list their own stocks
2	B	Both backend and frontend functionality allows users to see stocks on the stock market
3	A	Both backend and frontend functionality allows users to approve/decline trade suggestions
4	C	Queries for Stock owners transaction history is developed backend but are currently not available frontend.
5	A	Both backend and frontend functionality allows business owners to register their company
6	C	Queries for business owners transaction history is developed backend but are currently not available frontend.
7	B	Both backend and frontend functionality allows business owners to change capital
8	B	Both backend and frontend functionality allows business owners to numeric increase in stock
9	A	Both backend and frontend functionality allows Brønnøysund to approve/decline business registrations.
10	A	Queries for transaction history is developed backend but are currently not available frontend.
10	A	Queries for request history is developed backend but are currently not available frontend.

Non-functional requirements

Id	Prio	result
1	A	The blockchain inherently ensures secure storing of data. No entry level (frontend) security has been implemented.
2	A	Access control and access identifying are implemented, but authentication is missing.
3	C	Front end application is scaleable on different desktops, but not on other devices such as phones.
4	C	Static styling requirements is met.
5	B	Minimal error handling for user have been implemented.
6	C	The user is not faced with any unwanted behaviour, though some feedback is not fully implemented.
7	B	Most of the functional requirements is met.
8	B	No Norwegian laws have been breached by our application.
9	A	Norwegian Difi standard is met.

6.3 Evaluation

We have managed to develop a solution that allows creation of a shareholder registry and has a small stock market. The solution has different participants with different rules which the users get controlled access to. And we developed all this with blockchain and smartcontracts.

Being partly a blockchain proof of concept where most of the requirements were met we choose to conclude that with our given use case blockchain is a proven concept. Even though our solution as a whole is not a finished solution. There is for example instances of bugs that must be fixed. And more potential in UX-design. But the most prioritized goals are achieved.

7 Future work

In this section we will cover the planned functionality we ended up not implementing, mainly due to time constraints. We also discovered potential new features during our work with the project that we did not plan to add, but would be interesting to add in the future.

Most of the planned features that were not implemented was due to running out of time. The reason being underestimating the time it takes to figure out and implement technology without having any prior knowledge.

7.1 User authorization and management frontend

Every functionality needed to implement this were set. Hyperledger handling users as participants with id-cards, the rest-api set on multi-user mode and an external wallet stored at github requiring authorization. The only thing remaining was implementing the github authenticating middleware in the angular application.

This was planned as a task for the last sprint, but we did not have enough time to complete this task before our last playback. The same day we had planned to do user tests which we chose to prioritize over frontend authentication.

This resulted in having to dummy the id of the user in the angular application.

7.2 More UX design

The frontend applications we developed mostly met the functional requirements. But as we discovered from the user tests that the GUI is not as intuitive and user friendly as it should be. Because of this more focus on UX design would be a natural next step in further development.

7.3 Potential features

If we were to continue to work on this project, these are some features we would consider adding:

Login w/ BankID

One thing that we and our interested parties would have liked to see implemented was a extra login security measure through the use of BankID (BankID, 2018). BankID is a authentication service that banks and business use to confirm identity in Norway. Proving to be very stable and secure, it would definitely lift our application to a more real world state.

Dashboard with statistics and visualization graphs

Adding a more complex dashboard that gives the user more data and graphs on how their shares and the overall business environment has changed could be a interesting feature to add.

8 Summary

Blockchain is a exciting technology with immense potential. We are very happy to have had the opportunity to explore blockchain in this bachelor project.

The project process being executed as a realistic development process granted us with valuable experience. We have learned a lot from cooperating in a group as well as using development methodology. Developing for a real customer has given us experience with communication and pitching ideas as well as expectation management. Along the way receiving excellent mentoring from IBM as well as our menthor at OsloMet. At times we have shown capability of independence as well when we were required to figure out use of different technology by our own.

All in all we are delighted with the whole process and the experiences it gave us as well as being content with our effort.

9 References

- Abrahamsson, P., Salo, O., and Ronkainen, J. (2002). Agile software development methods: Review and analysis. Espoo: VTT. <https://arxiv.org/ftp/arxiv/papers/1709/1709.08439.pdf>AgileSoftwareDevelopmentMethods:ReviewandAnalysis
- Advanced Development Methods. (n.d.). What is a product owner?. Retrieved April/May, 2018, from <https://www.scrum.org/resources/what-is-a-product-owner>
- Aksjeloven. (1997). Lov om aksjeselskaper. Retrieved February 5, 2018, from https://lovdata.no/dokument/NL/lov/1997-06-13-44/*
- Altinn. (2018). Design system - Styleguide. Retrieved April/May, 2018, from <https://altinn.github.io/designsystem-styleguide/>
- BankID. (2018). We ensure safe and secure identification and signing. (n.d.). Retrieved April/May, 2018, from <https://www.bankid.no/en/about-us/>
- Boer, G. (2017, April 04). What is Scrum? - Azure DevOps. Retrieved April/May, 2018, from <https://www.visualstudio.com/learn/what-is-scrum/>
- Brønnøysundregistrene (2010). Elektronisk aksjeeierbok i Altinn. Brønnøysund. <https://www.regjeringen.no/contentassets/7cff1a12a8494404b386a9b512e76brreg-rapport.pdf>
- Direktoratet for forvaltning og IKT. (2014). Løsningsforslag for Web. Retrieved April/May, 2018, from <https://uu.difi.no/krav-og-regelverk/losningsforslag-web>
- Direktoratet for forvaltning og IKT. (2015, February 03). Brukertesting. Retrieved May 03, 2018, from <https://www.difi.no/fagomrader-og-tjenester/tidstyver/tidstyvdatabasen-verktoy-og-metoder/identifisere/brukertesting>

Gibbons, S. (2018, January 14). Empathy Mapping: The First Step in Design Thinking. Retrieved April/May, 2018, from <https://www.nngroup.com/articles/empathy-mapping/>

Hellem, D. (2017). (n.d.). What is Agile Development? - Azure DevOps. Retrieved April/May, 2018, from <https://www.visualstudio.com/learn/what-is-agile-development>

Hyperledger Project. (n.d.). Architecture Explained. Retrieved April/May, 2018, from <http://hyperledger-fabric.readthedocs.io/en/release-1.1/arch-deep-dive.html>

IBM Knowledge Center. (n.d.). Defining use cases. Retrieved April/May, 2018, from https://www.ibm.com/support/knowledgecenter/SSWSR9_11.0.0/com.ibm.pim.dev.doc/pim_tsk_arc_definingusecases.html

IBM. (2018, April 19). En Nærmere titt på IBM. Retrieved April/May, 2018, from <https://www.ibm.com/ibm/no/no/>

Mehrabian, A. (2007). Nonverbal communication (1st ed., pp. 1). New York: Routledge. Retrieved May 09, 2018, from <https://www.taylorfrancis.com/books/9781351308717>

Regjeringen. Prop. 94 LS (2013–2014). Retrieved May/April, 2018, from <https://www.regjeringen.no/no/dokumenter/Prop-94-LS-2013-2014/id759716/>

Rouse, Margaret (2016). What is IBM (International Business Machines) - Definition from WhatIs.com. (n.d.). Retrieved April/May, 2018, from <https://searchitchannel.techtarget.com/definition/IBM-International-Bu>

Software Testing Fundamentals. (2018, March 03). Black Box Testing. Retrieved April/May, 2018, from <http://softwaretestingfundamentals.com/black-box-testing/>

Sommerville, I. (2010). Software engineering 9th Edition. Boston, MA:

Addison-Wesley <https://inspirit.net.in/books/academic/Ian%20Sommerville%20Software%20Engineering,%209th%20Edition%20%20%20%202011.pdf>

Svartdal, F. (2009). Psykologiens forskningsmetoder (3rd ed.). Bergen: Fagbokforlaget.

The Linux Foundation. (n.d.). Hyperledger Composer. Retrieved April/May, 2018, from <https://www.hyperledger.org/projects/composer>

The Linux Foundation. (n.d.). Hyperledger Fabric. Retrieved April/May, 2018, from <https://www.hyperledger.org/projects/fabric>

W3. (2018). WCAG Contrast Minimum Requirements. Retrieved May 11, 2018 From <https://www.w3.org/TR/WCAG21/#contrast-minimum>

10 Appendix

10.1 Worklog

Loggbok

(Mal som vi kan fylle ut for hver økt vi har hatt)

Dato:	Hvor:
--------------	--------------

Hvem:

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:

Resultat:

Dato: 16.01.2018	Hvor: IBM - Lakkegata 53
-------------------------	---------------------------------

Hvem: Lars, Bjørn, Kim, Thomas L.
--

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:

Fått innføring i IBM Cloud og oppdatering på hvordan det ligger an med rammer og krav til prosjektet.		Invitert av IBM til innføring i teknologier som kan være interessante å bruke til prosjektet.
---	--	---

Resultat:
Fått oppdatering på når vi skal ha første Blockchain workshop og at vi skal bruke IBM Blockchain Developer sammen med HyperLedger Composer og Angular. Vi har også opprettet kontoer på HyperLedger forumet på rocketchat.

Dato: 26.01.18	Hvor: OsloMet
-----------------------	----------------------

Hvem: Lars, Bjørn, Kim, Thomas L. og Thomas G.

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
Møte med veileder.		For å møte veileder og gi informasjon om hvor langt vi har kommet.

Resultat:

Et møte på ca. 30 min med veileder hvor vi fortalte hvor langt vi hadde kommet og hvordan forholdet vårt er med IBM. Fikk tips fra tidligere erfaring fra veileder.

Dato: 01.02.18

Hvor: IBM - Lakkegata 53

Hvem: Lars, Bjørn, Kim, Thomas L. og Thomas G.

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
Innføring i IBM sin blockchain teknologi		For å lære å sette opp prosjekt og lære hvordan de forskjellige teknologiene fungerer før vi begynner med den faktiske oppgaven.

Resultat:

Modellert et dummy-prosjekt. Opprettet blockchain-nettverk.

Dato: 02.02.18

Hvor: IBM - Lakkegata 53

Hvem: Lars, Bjørn, Kim, Thomas L. og Thomas G.

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
Fortsette med opplæring		Videre innføring i blockchain teknologi

Resultat:
Vi delte opp oppgavene og jobbet med disse individuelt og som gruppe. Jobbet med front-end(Angular).

Dato: 08.02.2018	Hvor: IBM - Lakkegata 53
------------------	--------------------------

Hvem: Bjørn, Kim, Thomas L. og Thomas G.
--

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
Sette opp user-stories og kravspeks.	Agile-utvikling / møte / samtale	Legge basen for hva som skal utvikles og hvilke funksjonaliteter og aktører / assets som skal inn i Blockchain nettverket.

Resultat:

Vi fikk satt opp forskjellige requirements for systemet og ble enige om hva som skal være fokus for oppgaven.

Dato: 09.02.2018

Hvor: IBM - Lakkegata 53

Hvem: Lars, Bjørn, Kim, Thomas L. og Thomas G.

**Mål: Hva skal
gjøres/Har blitt
gjort?**

Hvordan:

Hvorfor:

Videre definisjon av
requirements.
Planlegge første
sprint.

For å planlegge sprinten som
skal startes på neste uke.

Resultat:

Videre definisjon og lagt inn tilleggsinformasjon på requirements i Trello. Estimert tiden på requirements. Blitt enig om hvor mye tid vi skal sette av til hver sprint. Planlegging av første sprint.

Dato: 14.02.2018

Hvor: Grupperom på skolen

Hvem: Bjørn, Kim, Lars og Thomas G.

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
Starte på sprint 1. Lage business model		

Resultat:
Begynt på modellering av business modell. Venter fortsatt avklarelse på noen punkter fra Olga. Skal vi bruke channels?

Dato: 15.02.2018	Hvor: Grupperom på skolen
------------------	---------------------------

Hvem: Bjørn, Kim, Thomas L., Lars og Thomas G.
--

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
Ferdigstille business modell		

Resultat:

Jobbet videre med business model. Venter fortsatt på svar fra Olga. Business model er ikke helt ferdig. Sett på ACL og opprettelse av identity cards.

Dato: 20.02.2018

Hvor: Pilestredet 35, kontoret til Boning

Hvem: Bjørn, Kim, Thomas L. og Lars (Thomas G. på kurs)

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
Møte med veileder	via møte face-to-face på skolen.	Gi oppdatering på hvordan prosessen går.

Resultat:

Hadde et kort møte med Boning hvor vi informerte han om vår progress.

Fikk svar fra Olga med oppklaring rundt spørsmål vi hadde.

- Fortsatt problemer med VM'ene
- Skal bruke channels.
- Fått oversendt use-casene fra møtet med altinn.
- IBM må avklare IP(?) med kunden før kontrakt kan underskrives.

Dato: 21.02.2018

Hvor: Grupperom på skolen

Hvem: Bjørn, Kim, Thomas L. og Lars (Thomas G. på kurs)

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
<ul style="list-style-type: none"> - Gjøre ferdig business modellen etter svar fra Olga. - Sette opp miljøet lokalt på datamaskinene våre. 		

Resultat:

Gjort ferdig business modellen. Satt opp miljøet lokalt på datamaskinene våre. Nesten hele modellen er lagt inn.

Dato: 22.02.2018

Hvor: Grupperom på skolen

Hvem: Bjørn, Kim, Lars, Thomas L og Thomas G

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
<ul style="list-style-type: none"> - Jobbet med Sprint 1 - Starte med ACL 	Lese dokumentasjon, prøve oss frem med å lage regler.	For å få bedre og nødvendig kunnskap for oppgavens fremgang.

Resultat:

- Vi har lest mye dokumentasjon på peers, consensus, transactions og ACL. Vi har klart å sette opp enkelte regler for forskjellige aktører i Playground. Målet var å få en forståelse for dette og komme litt i gang med det, slik at vi kunne bruke morgendagen til å få reglene på plass.
- Fått avklart møte/Stand-up med Altinn Onsdag 28.02.18.

Dato: 23.02.2018

Hvor: Grupperom på skolen

Hvem: Bjørn, Kim, Lars, Thomas L og Thomas G

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
<ul style="list-style-type: none"> - Ferdigstillelse av oppgavene satt i sprint 1. - Skrive videre på bachelor-rapport. - Lage Script som "seeder" business nettverket med brukere og ID'er. 	<ul style="list-style-type: none"> - Bruke dokumentasjon og HyperLedger Composer Fabric til å teste oss frem med forskjellige ACL-regler for å få det til å fungere. - Fortsette på rapport startet i Latex. 	<ul style="list-style-type: none"> - For å nå kravene som er satt til Sprint 1. - For å gjøre det enklere for oss å skrive en god rapport og få ting på "papiret" mens det fortsatt ligger friskt i minnet. - Hver gang vi gjør endringer i business-modellen / ACL eller annet som påvirker .bna fila som skal publiseres må vi recompile og re-deploye nettverket. Dette gjør at vi mister participants/assets etc. som er lagret på nettverket og vi ønsker derfor å ha et script som kan "populate" nettverket med de participantene og assetene vi trenger.

Resultat:

- Vi er i stor grad ferdig med oppgavene som er satt i sprint en, men noe har vi satt på vent da vi må få til å sette opp konsensus mekanismer som sikrer transaksjoner m.m. på blockchain.
- Ryddet i rapportoppsett m.m.
- Fått lagd shell-script som kjører composer kommandoer for å opprette participants og tildele ID-kort. Dette forenkler prosessen med å teste / endre business modellen senere (for deployment til web) da vi slipper å opprette participants og ID-kort manuelt.

Dato: 28.02.18

Hvor: IBM - Lakkegata 53

Hvem: Thomas L, Kim, Bjørn og Thomas G

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
<ul style="list-style-type: none"> - Første stand-up med Altinn og IBM. - Gå igjennom første sprint og vise frem hva vi har gjennomført. 	Møte med Altinn og Br.sund over skype, hos IBM.	Dette for å vise frem og forklare til Altinn + Br.sund både hva som er gjort, men også for å forklare hvordan ting er konstruert og henger sammen.

Resultat:

De som deltok på møtet var følgende: Thomas L, Kim, Bjørn og Thomas G fra bachelorgruppen, sammen med Olga, Thaline og Henning fra IBM, Sverre, Leo og (+1) fra Altinn, og Andreas fra Brønnøysundregisteret

- Litt tekniske problemer ved oppstart, møtet startet opp før vi egentlig var klare for å vise så noe av tiden gikk til teknisk oppsett.
- Da vi kom i gang forklarte Bjørn de forskjellige elementene og viste frem instanser av de forskjellige aktørene. Videre så forklarte han til Altinn at de forskjellige aktørene ville ha forskjellige rettigheter bestemt på "hvilken type aktør" de er.
- Altinn fikk stille forskjellige spørsmål underveis. Dette gjaldt både ifht til hva som ble demonstrert men også ifht til eventuelle uklarheter ("Offentlig"-participant, og Brreg -> myndighets-register).
- Altinn var alt-i-alt veldig fornøyd med progresjonen vi hadde hatt og følte de forstod gangen i det vi hadde å vise. De fortalte at de så et stort potensiale med det vi hadde gjort, og så positivt på det videre arbeidet.

- IBM uttrykte at de var fornøyde med møtet de også, og anbefalte til neste stand-up at vi alle introduserte oss slik at deltakerne fra Altinn og Br.sund får sjansen til å bli kjent med hver av oss.

Dato: 01.03.18

Hvor: Datalab på OsloMet (p35)

Hvem: Bjørn, Thomas L, Kim, Lars og Thomas G

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
<ul style="list-style-type: none"> - Oppstart på ny sprint. - Tilegne oss informasjon om transaksjoner og konsensus / "Endorsement" 	<p>Bruker Trello-oversikten og se hva som skal gjøres, og bruker dette som arbeidsplan/To-do list.</p> <p>Lese oss opp på dokumentasjon og forsøke å få satt opp transaksjoner slik at det fungerer.</p>	<p>Trello gir bra oversikt og gjør det enkelt å jobbe strukturert.</p> <p>Viktig å lese dokumentasjon for å få grunnleggende og viktig kunnskap om transaksjoner i Blockchain. Det vil være helt essensielt for å komme i mål med denne sprinten, og prosjektet i seg selv.</p>

Resultat:

Gruppen i sin helhet brukte dagen på å lese og forsøke å forstå hvordan transaksjoner fungerer, hvordan det skal struktureres og hvordan det henger sammen. I løpet av dagen har vi fått noe kunnskap og satt igang med å strukturere queries, samtidig som vi opplever noe utfordringer med å få det til å fungere. Teoretisk sett så opplever vi å ha fått noe kunnskap, men at det er mye å sette seg inn i og at vi må bruke ytterligere tid på å tilegne oss kunnskap. Morgendagen vil derfor bestå av samme type arbeid for å fortsette der vi slapp i dag.

Dato: 02.03.18	Hvor: Grupperom på OsloMet (p35)
-----------------------	---

Hvem: Bjørn, Thomas L, Kim, Lars og Thomas G

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
<ul style="list-style-type: none"> - Fortsette med arbeidet i fra gårdsdagen - Vi trenger mer tid på å tilegne oss informasjon om transaksjoner og konsensus / "Endorsement" - Jobbe med rapportskriving 	<p>Lese oss opp på dokumentasjon og forsøke å få satt opp transaksjoner slik at det fungerer.</p> <p>Skrive på rapporten vår, som vi skriver på sharelatex.com- tilføye nye ting som har skjedd.</p>	<p>Viktig å lese dokumentasjon for å få grunnleggende og viktig kunnskap om transaksjoner i Blockchain. Det vil være helt essensielt for å komme i mål med denne sprinten, og prosjektet i seg selv.</p> <p>Vi skriver på rapporten for å ha en naturlig flyt i arbeidet med den. Ved å skrive litt hver uke så håper vi å få med alt som er relevant og viktig for utviklingen av rapporten, og at ikke noe skal falle utenfor fordi vi har drøyd arbeidet med å skrive på den.</p>

Resultat:
<p>Dagen i dag har vært svært lik gårdsdagen. Tiden har gått til å lese videre på dokumentasjon, samt problemløsning av kode vi har skrevet men som ikke har fungert som vi ønsket. Pr dags dato så har vi ikke fått det til, men kunnskapsmessig så har vi noe fremgang. "Query'en funker, men transaksjonene fungerer ikke" - Thomas Langseth. Vi må prioritere dette og implementering av konsensus videre, frem til det fungerer.</p>

Dato: 07.03.18	Hvor: Grupperom på OsloMet (p35)
-----------------------	---

Hvem: Bjørn, Thomas L, Kim, Lars og Thomas G

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
-Fikse transaksjonen "expandCapital" -Skrive på rapporten	Lese dokumentasjon. Lete etter svar på forum(github, stackoverflow)	

Resultat:
Skrevet på rapporten. Fikset query så "expandCapitol" transaksjonen fungerer. Funnet nytt problem: hyperledger støtter ORDER BY, men couchDB støtter det ikke.

Dato: 08.03.18	Hvor: Grupperom på OsloMet (p35)
----------------	----------------------------------

Hvem: Bjørn, Thomas L, Kim, Lars og Thomas G
--

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
- Møte med Altinn og IBM for å gå igjennom sprint 1	-Møte på skype	-Repeterende gjennomgang av sprint 1 med nye personer fra Altinn

Resultat:

Gjennomførte repeterende gjennomgang av sprint 1 med Altinn og IBM med Olga og Henning fra IBM, og Øyvind Huseby, Sverre og ?? fra Altinn.

Dato: 09.03.18

Hvor: IBM - Lakkegata 53

Hvem: Bjørn, Thomas L, Kim og Thomas G

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
- Standup med Olga og Altinn - Gjennomgang av presentasjon fra Olga	- Møte foregikk over skype/spark - Olga gikk gjennom en presentasjon hun hadde om blockchain	- For å gå gjennom og vise hva vi hadde gjort i sprint 2

Resultat:

Deltagere; Olga fra IBM. Sverre, duden med langt hår(Leo?) og Andreas Hamnes fra Altinn.

Spørsmål fra Altinn:

- Hva skjer hvis to selskaper merger?
- Hvordan sjekke om selskap allerede har aksjebok?
- Hvordan koble seg opp mot andre blockchains eller sjekke register hos BR?
- Hvordan overføre aksjeeierbok til en annen blockchain?

Andre kommentarer fra Altinn:

- Må ha mulighet for å legge til flere eiere når man oppretter aksjeeierbok når dette skal presenteres.
- Pålydende verdi er feil. Annet felt med reel verdi. Pålydende verdi er den verdien vedtektene sier at aksjene har. Verdien vil alltid være lik eller mer enn pålydende. Pålydende er verdien på aksjen når selskapet ble stiftet.
- Legge inn parameter for:
 - hva den koster nå
 - hvor mye forrige eier betalte

- pålydende
- Første aksje skal først ut. Har noe å si om skatt. Må tenke på at regelverket kan endre seg. F. eks. velge hvilken av aksjene som skal selges. Tjene mindre på en aksje ved å selge den du kjøpte dyrest for å skatte mindre.
- Sette opp query og frontend hvor man kan se historikken til en aksje. Kunne vise analytics på aksjer. Kunne se hvor mye de har blitt solgt for hver gang, om det er norske eller utenlandske statsborgere, mm.
- Prisen jeg har betalt for aksjen skal ikke være tilgjengelig for selskapet?
 - Vil kanskje ikke vise omsetningsverdien til selskapet. Skal vises for skatteetaten, kjøper og selger. De andre har ikke noe med det å gjøre.
- Informasjon som ikke er tilgjengelig for offentligheten:
 - Pålydende er tilgjengelig for alle på blockchain, mens omsetningsverdi er kun tilgjengelig for kjøper, selger og offentlige myndigheter med rett på denne informasjonen.

Dato: 13.03.18

Hvor: Datalab på OsloMet (p35)

Hvem: Bjørn, Thomas L, Kim, Lars og Thomas G

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
<ul style="list-style-type: none"> - Sette opp rest-API med autorisering og flere brukere - Begynte å jobbe med GUI 		

Resultat:

Tegnet opp hvordan GUI'et skal se ut.

Dato: 15.03.18	Hvor: Datalab på OsloMet (p35)
-----------------------	---------------------------------------

Hvem: Bjørn, Thomas L, Kim, Lars og Thomas G

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
- Jobbe videre med GUI		

Resultat:
<p>Jobbet videre med transaksjoner hvor man skal oppdatere flere typer assets i en transaksjon. Fortsatt stuck på den. Kanskje dette kan fikses når man oppdaterer composer eller CouchDB? Lese dokumentasjon på endringer i nye versjoner av composer og CouchDB.</p>

Dato: 16.03.18	Hvor: Datalab på OsloMet (p35)
-----------------------	---------------------------------------

Hvem: Bjørn, Thomas L, Kim, Lars og Thomas G

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:

<ul style="list-style-type: none"> - Jobbe videre med GUI - Lese dokumentasjon 		
--	--	--

Resultat:
Lest MASSE dokumentasjon. Jobbet videre med flytskjema og GUI.

Dato: 21.03.18	Hvor: Datalab på OsloMet (p35)
-----------------------	---------------------------------------

Hvem: Lars, Thomas G., Thomas L., Bjørn og Kim

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
<ul style="list-style-type: none"> -Møte med Lars Fanebost -Lage GUI 		<ul style="list-style-type: none"> -For å vise hva vi har gjort til nå

Resultat:

Møte med Lars Fanebost og Olga. Forklarte Lars hva vi har gjort til nå og viste han modellen vi har laget. Fikk oppklaring i hva peers er og hvor mange peers vi skal ha. Fikk ikke sett på transaksjoner. Skal se på dette på fredag.

Begynt å lage gui.

Dato: 22.03.18

Hvor:

Hvem:

**Mål: Hva skal
gjøres/Har blitt
gjort?**

Hvordan:

Hvorfor:

Resultat:

Dato: 23.03.18

Hvor: IBM - Lakkegata 53

Hvem: Bjørn, Thomas G., Thomas L. og Kim

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
-Gjennomgang av sprint med IBM og Altinn		

Resultat:
<p>Leo og Andreas fra Altinn. Olga og Lars F. fra IBM.</p> <p>Gikk gjennom oppgavene på Trello. Gikk gjennom sketcher/bilder av GUI og flytskjema. Viste GUI'et som vi har laget(uten CSS).</p> <p>-Diskutert med Skatteetaten:</p> <ul style="list-style-type: none"> -Beregne skatt på salg av aksjer. -Ikke enten eller på relasjonsdatabase og blockchain, men begge. <p>Block chain blir et bindeledd mellom relasjonsdatabaser(Andreas) Overføre informasjon mellom etater med blockchain</p> <p>Fra Altinn:</p> <ul style="list-style-type: none"> -Salg av selskap? Hvordan fungerer dette? Skal vi ha det med? -Legge ut sketcher/bilder og flytskjema på box -Leo: Synliggjøre interaksjoner mellom aktører.(Hyperledger explorer) -Fokusere mer på logikken og ikke fokusere så mye på GUI. -Validering på frontend. Har lagt det til i backend. -OWASP -> valider så tidlig som mulig. Skal ha validering på frontend <p>Presentasjon den 11. april</p>

Dato: 04.03.18	Hvor: Datalab på OsloMet (p35)
----------------	--------------------------------

Hvem: Bjørn, Thomas L., Lars og Kim

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:

-Oppdatere VM'er		
------------------	--	--

Resultat:

Oppdatert til hyperledger composer v0.19 på VM'ene og lokale maskiner. Fikset indeksering i CouchDB etter oppdatering. Automatisk indeksering med bash-script. Fikset transaksjonen addStocks.

Dato: 05.04.18

Hvor: Datalab på OsloMet (p35)

Hvem: Bjørn, Thomas L., Thomas G., Lars og Kim

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
<ul style="list-style-type: none"> - Begynne på testing - Events 		

Resultat:

Startet på og utført første test. Rydda og opprettet scripts. Fikset på transaksjoner. Lagd flytskjema for journalister. Jobbet med events mm. LANG diskusjon rundt hvor mange orgs vi trenger.

Dato: 06.04.18

Hvor: Datalab på OsloMet (p35)

Hvem: Bjørn, Lars, Thomas L. og Kim. (Thomas G. jobbet hjemmefra)

**Mål: Hva skal
gjøres/Har blitt
gjort?**

Hvordan:

Hvorfor:

Resultat:

Skypemøte med Altinn og lbm.
Olga fra lbm(Thaline også?). Andreas, Sverre og Kristine fra Altinn.
Gått gjennom det vi har gjort og gitt et overblikk over hvor vi er.

Dato: 09.04.18

Hvor: Datalab på OsloMet (p35)

Hvem: Bjørn, Thomas L. og Kim

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:

Resultat:
Laget flow chart for aksjeeier, journalist og offentlig. Lagt til nytt id felt på stockowner i modellen og endret seed.sh.

Dato: 11.04.2018	Hvor: OsloMet og House of Innovation Oslo
------------------	---

Hvem: Thomas L, Kim Knudsen, Lars Magnus Henriksen og Bjørn Golberg

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
<ul style="list-style-type: none"> - Forberede presentasjonen for bedrifter på HIO - 		<ul style="list-style-type: none"> - Vise til offentligheten hva vi har gjort i samarbeid med IBM og BrReg

Resultat:
<ul style="list-style-type: none"> - Mye god tilbakemelding. Imponert over vår forståelse og utførelse på kort tid.

Dato: 13.04.18	Hvor: Datalab på OsloMet (p35)
-----------------------	---------------------------------------

Hvem: Thomas L., Thomas G., Bjørn og Kim

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
-Fikse import av scss -Dele opp transaksjoner		

Resultat:
Delt opp transaksjoner. Skrevet rapport om use case og persona. Fikset litt import av scss. Skjønner ikke hva som er galt. Får importert, men får 404 selv om linken er riktig.

Dato: 17.04.18	Hvor: Datalab på OsloMet (p35)
-----------------------	---------------------------------------

Hvem: Thomas L., Lars Magnus, Bjørn og Kim

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
-Transaksjoner - Autentisering - Testing		

Resultat:

- Ferdigstille transaksjoner
- Jobbet med autentisering, ingen fungerende versjon
- Noe frontend logikk er gjort

Dato: 17.04.18

Hvor: Datalab på OsloMet (p35)

Hvem: Thomas L., Bjørn, Lars, Thomas G. og Kim

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:

Resultat:
Forstått hvordan enhetstester skal kobles opp mot systemet og begynt å skrive test-caser.

Dato: 18.04.18	Hvor: Datalab på OsloMet (p35)
-----------------------	---------------------------------------

Hvem: Thomas L., Bjørn, Lars, Thomas G. og Kim

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:

Resultat:
Enhetstester. Refaktorering og splitting av transaksjoner. Jobbet meg GUI.

Dato: 19.04.18	Hvor: Datalab på OsloMet (p35)
-----------------------	---------------------------------------

Hvem: Thomas L., Bjørn, Lars, Thomas G. og Kim

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:

Resultat:

Enhetstester. Refaktorering og splitting av transaksjoner. Jobbet meg GUI.

Dato: 20.04.18

Hvor: Datalab på OsloMet (p35)

Hvem: Thomas L., Bjørn, Lars, Thomas G. og Kim

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:

Resultat:

Møte hos IBM hvor vi viste fram det vi hadde gjort i denne sprinten. Olga fra IBM, Andreas og Leo fra Altinn/Brreg. Utført retrospective hvor vi skrev på post-it lapper. Det vi skrev ble enten puttet under "What worked in this sprint", "What didn't work in this sprint" og "What could be better". Vi fikk alle fire klistremerker hver som vi skulle putte på lappene under "What could be better". Så skulle vi ta de tre med flest klistremerker, og prøve å gjøre det bedre på disse punktene til neste gang. Det endte opp med at vi skulle forbedre oss på fire punkter siden to av lappene fikk fem klistremerker og to lapper fikk tre klistremerker.

De punktene vi skal forbedre oss på er:

- Jobbe til sent på kvelden(3)
- Oppmøte til oppsatt tid(3)
- QA på kode(5)/vise hverandre hva vi har gjort når vi er ferdige. For eksempel når man har laget en ny metode eller fikset noe nytt på frontend.
- Oppfølging av hverandre(5)

Gjennomgang av rapporten våres med Olga, hvor vi fikk innspill fra henne om hva vi kan gjøre bedre, legge til eller fjerne. Gått gjennom strukturen på rapporten.

Dato: 23.04.18**Hvor: Datalab på OsloMet (p35)****Hvem: Thomas L., Bjørn, Lars, Thomas G. og Kim**

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
<ul style="list-style-type: none">- Fullføre klient	<ul style="list-style-type: none">- Ferdigstille klient for aksjonær og br, mangler litt logikk og pop ups som viser om noe gikk bra / feilmeldinger	<ul style="list-style-type: none">- Klientene må "ferdigstilles" nok så de kan brukes til å gjennomføre brukertesting på applikasjonene / systemet.

Resultat:

Web applikasjoner nærmer seg ferdig, men noe logikk og design på sidene må inn så det er mer intuitivt for brukere.

Dato: 24.04.18

Hvor: Datalab på OsloMet (p35)

Hvem: Thomas L., Bjørn, Lars, Thomas G. og Kim

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
<ul style="list-style-type: none">- Fullføre klient- Enhetstester	<ul style="list-style-type: none">- Ferdigstille klient for aksjonær og br	

Resultat:

Dato: 25.04.18

Hvor: Datalab på OsloMet (p35)

Hvem: Thomas L., Bjørn, Lars, Thomas G. og Kim

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
<ul style="list-style-type: none"> - Fullføre klient - Enhetstester - Rapportskrivning 		

Resultat:
Skriver rapport og jobber med siste som trengs av utvikling

Dato: 26.04.18	Hvor: Datalab på OsloMet (p35)
----------------	--------------------------------

Hvem: Thomas L., Bjørn, Lars, Thomas G. og Kim
--

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
<ul style="list-style-type: none"> - Skrive rapport 	<ul style="list-style-type: none"> - Rapport 	

Resultat:

Rapport

Dato: 27.04.18

Hvor: Datalab på OsloMet (p35)

Hvem: Thomas L., Bjørn, Lars, Thomas G. og Kim

**Mål: Hva skal
gjøres/Har blitt
gjort?**

Hvordan:

Hvorfor:

- Fikse transaksjon
- Rapport

- Legge til aksjer har en feil, finne den å fikse

Resultat:

Rapport
API-url var feil, fikset

Dato: 30.04.18

Hvor: Datalab på OsloMet (p35)

Hvem: Thomas L., Bjørn, Lars, Thomas G. og Kim

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
<ul style="list-style-type: none">- Utforme brukertester- Rapport	<ul style="list-style-type: none">- Manus, oppgaver og diverse rundt brukertester	

Resultat:

Dato: 3.05.18	Hvor: Datalab på OsloMet (p35)
----------------------	---------------------------------------

Hvem: Thomas L., Bjørn, Lars, Thomas G. og Kim

Mål: Hva skal gjøres/Har blitt gjort?	Hvordan:	Hvorfor:
<ul style="list-style-type: none">- Skrive rapport- Planlegge brukertester	<ul style="list-style-type: none">- Rapport- Definere oppgaver til brukertest- Lage questionnaire	<ul style="list-style-type: none">- For å få gjennomført gode brukertester som gir oss de svarene vi trenger.

Resultat:

Dokumenter / questionnaires som skal brukes til brukertesting ble definert.

Dato: 04.05.18**Hvor: Datalab på OsloMet (p35)****Hvem: Thomas L., Bjørn, Lars, Thomas G. og Kim****Mål: Hva skal
gjøres/Har blitt
gjort?****Hvordan:****Hvorfor:**

- Skrive rapport
- Siste sprint playback med IBM og kunde
- Utføre brukertester

Resultat:

10.2 Flowcharts and wireframes

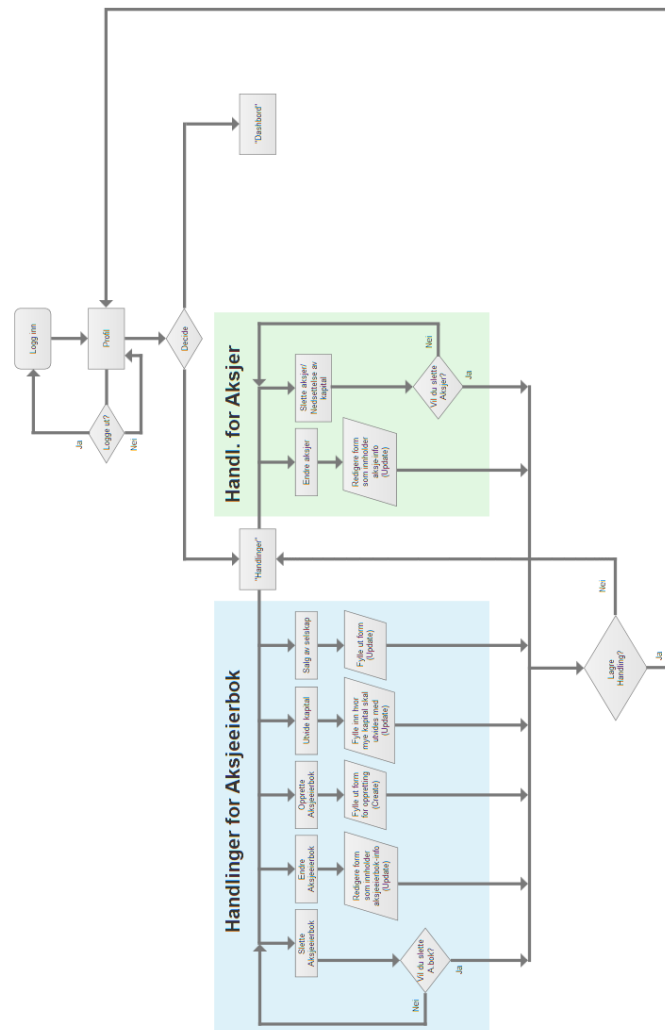


Figure 44: Flowchart for companies

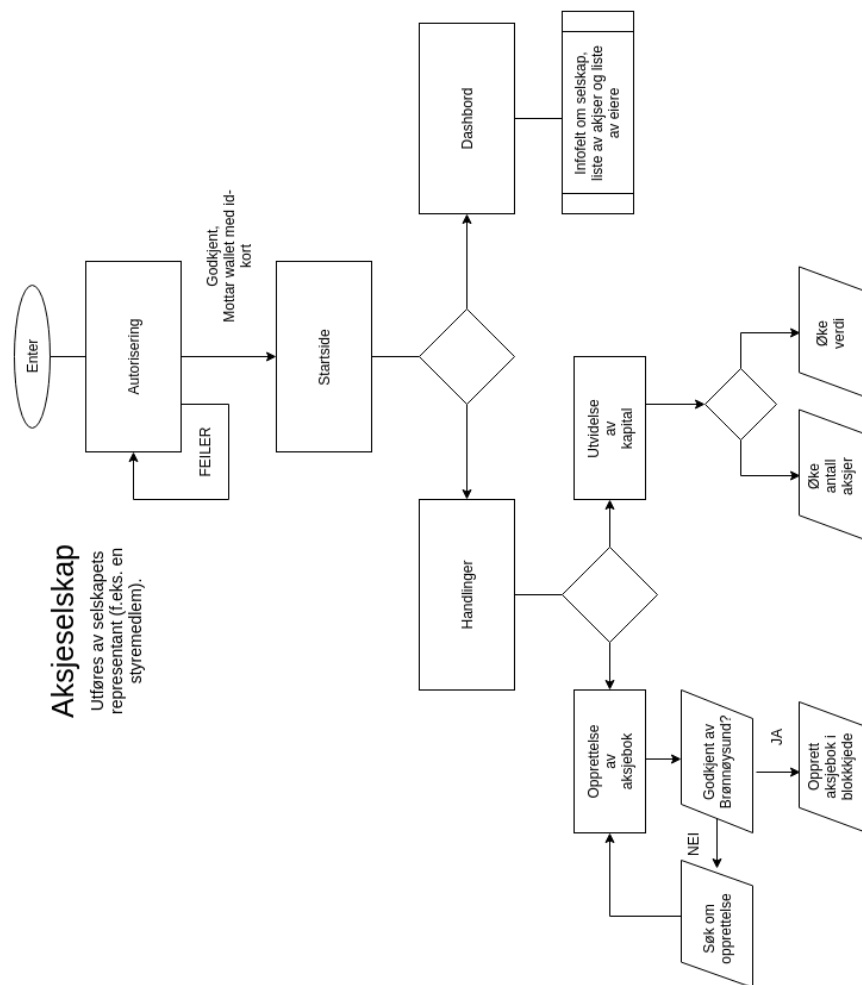


Figure 45: Revised Flowchart for companies

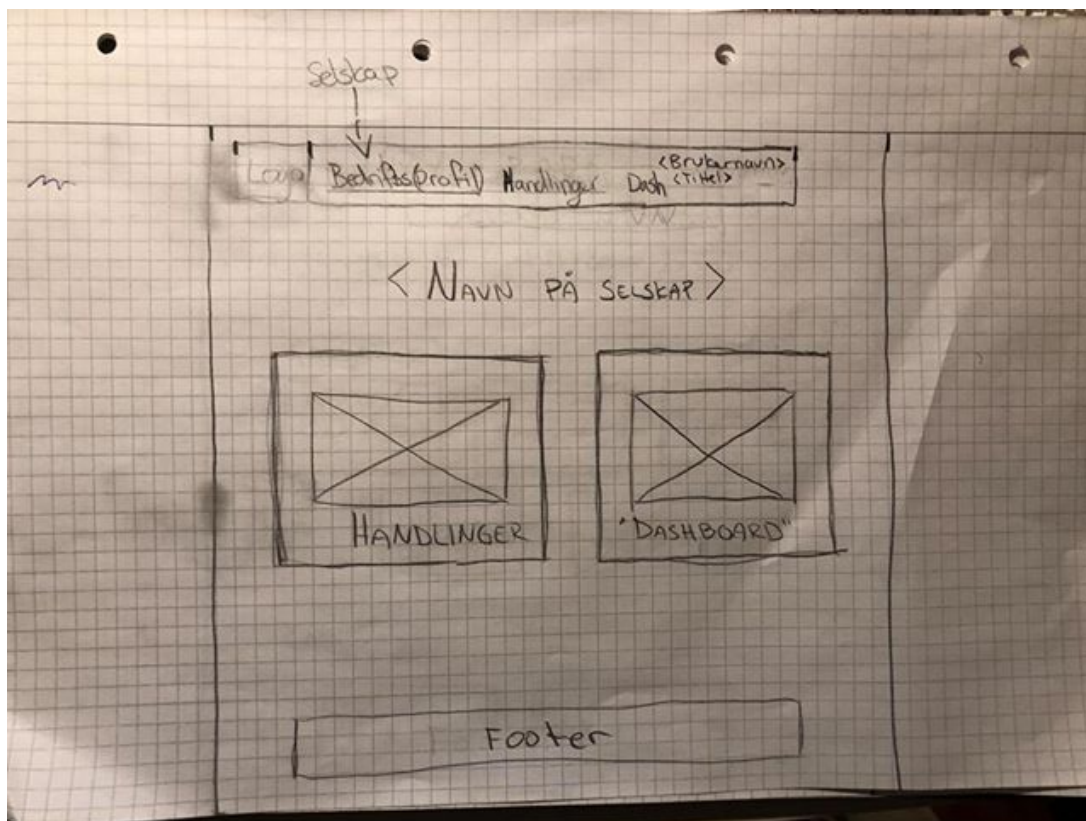


Figure 46: Low-fi wireframe of landing page

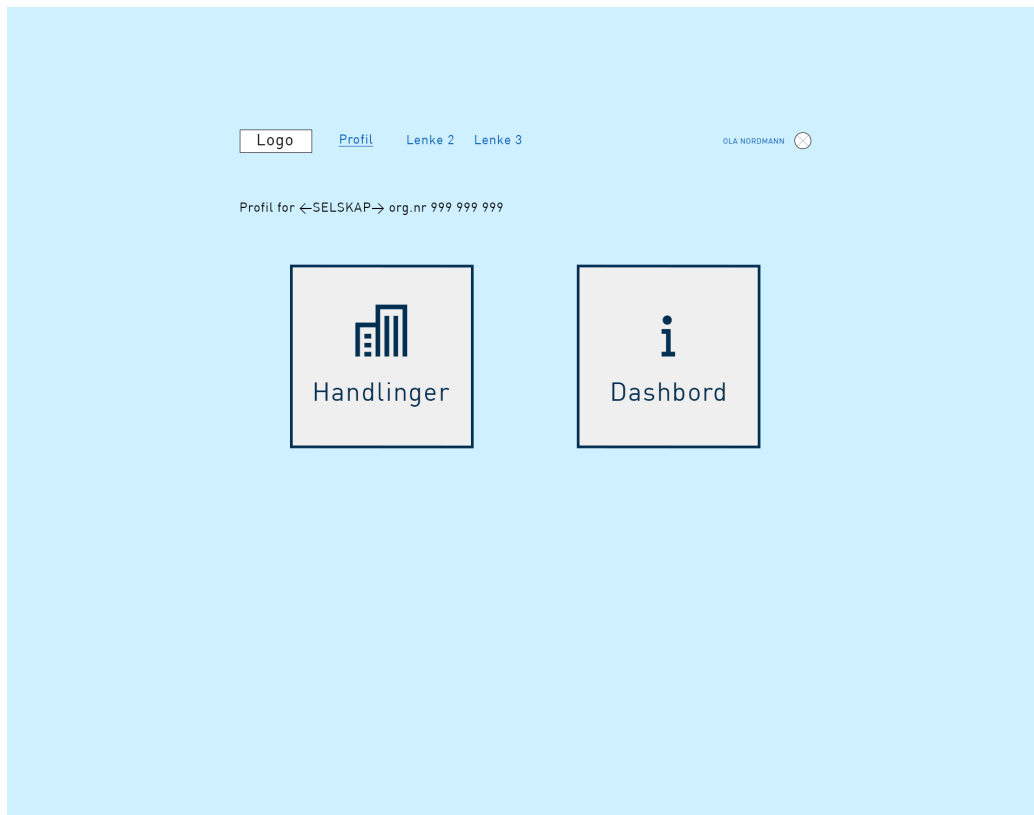


Figure 47: High-fi wireframe of landing page

Velkommen til din bedrift side!

Handleringer

Oversikt

Søk etter innhold

starte og drive bedrift

hjelp og kontakt

Brønnøysundregistrene, 8910 Brønnøysund. Org.nr. 974 760 673 [Om Altinn](#) [Driftsmeldinger](#) [Personvern og cookies](#)

Figure 48: Current design of the company landing page

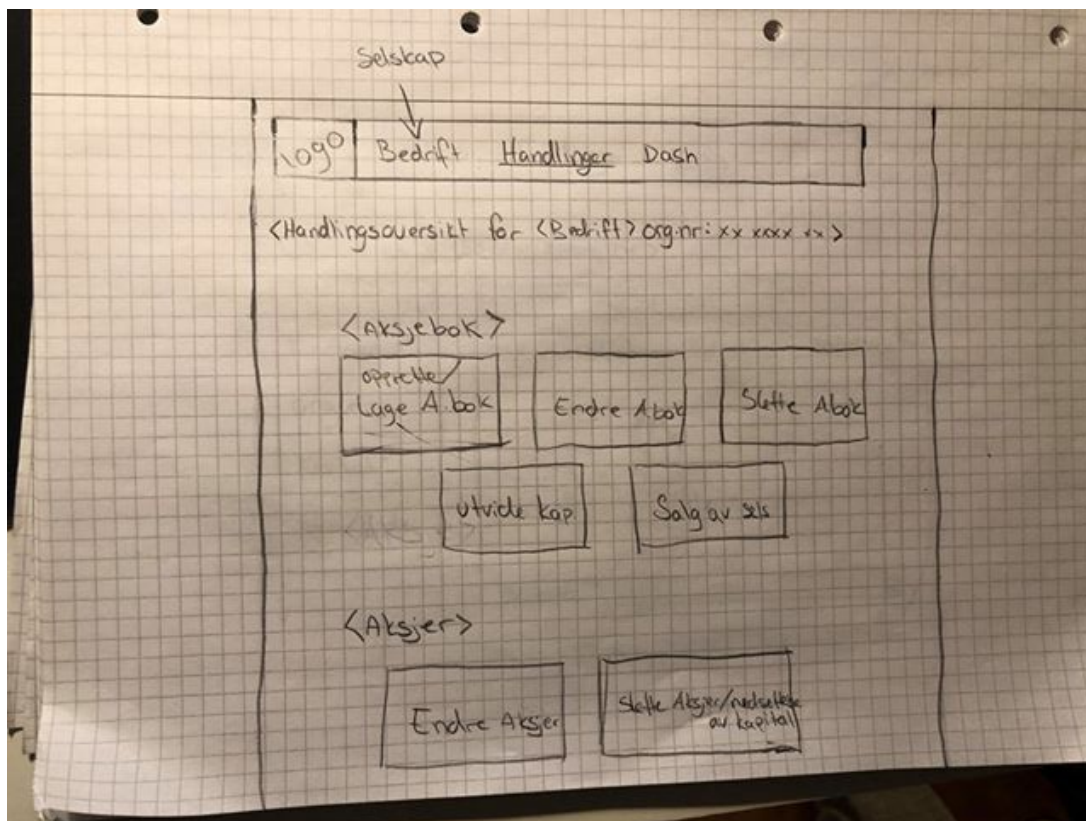


Figure 49: Low-fi wireframe of company actions

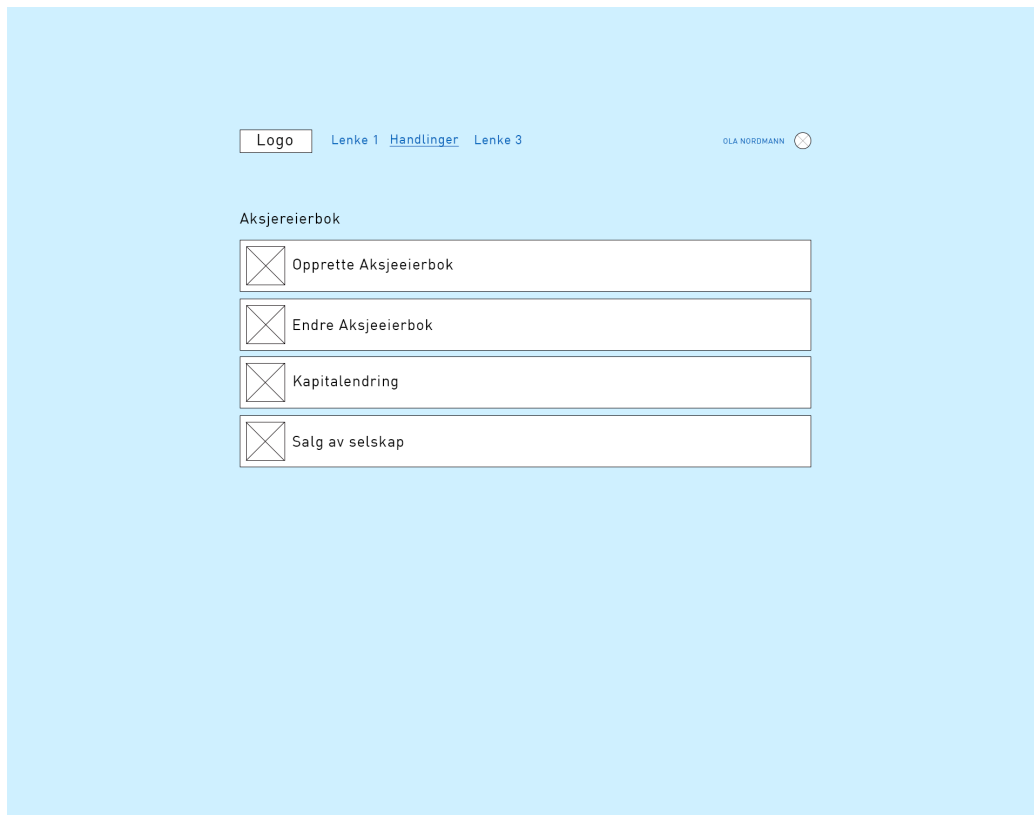


Figure 50: High-fi wireframe of company actions

Her kommer det dynamisk tekst??

[Registrer nytt aksjeselskap](#)

[Utvid kapital](#)

[Forespørsler aksjeoverdragelse](#)

Søk etter innhold

[starte og drive bedrift](#)

[hjelp og kontakt](#)

Brønnøysundregistrene, 8910 Brønnøysund. Org.nr. 974 760 673 [Om Altinn](#) [Driftsmeldinger](#) [Personvern og cookies](#)

Figure 51: Current design of the company actions

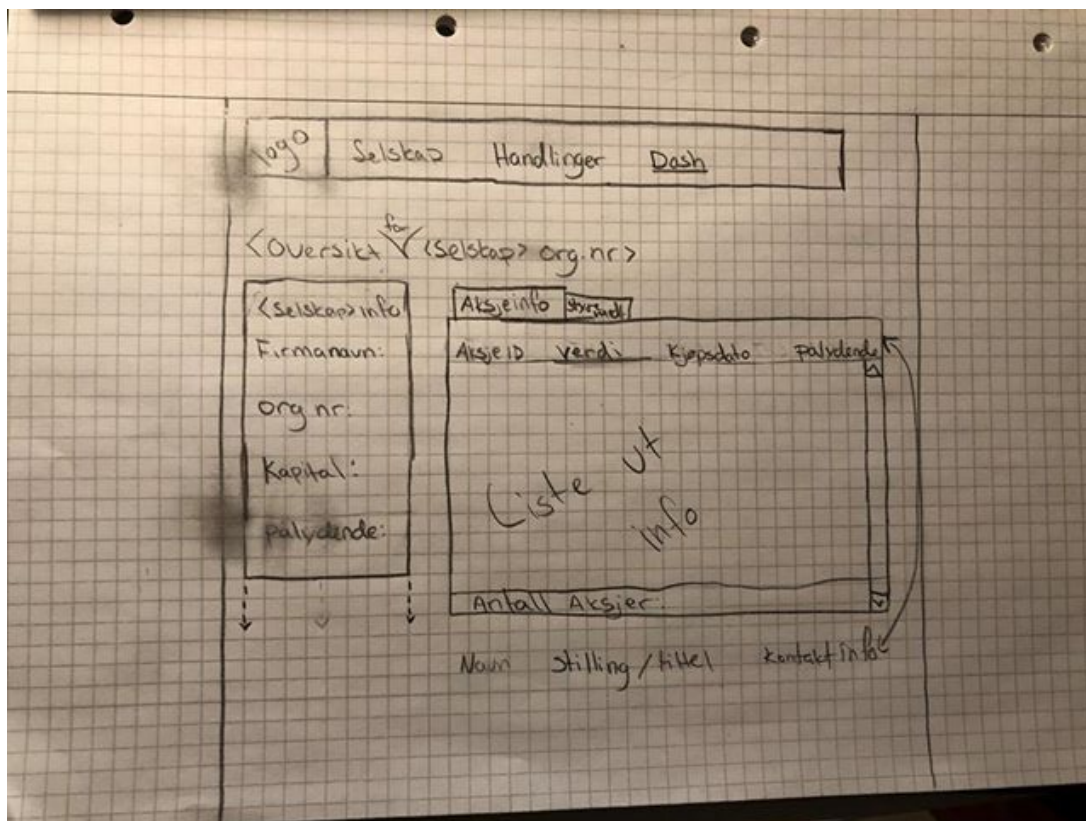


Figure 52: Low-fi wireframe of company dashboard

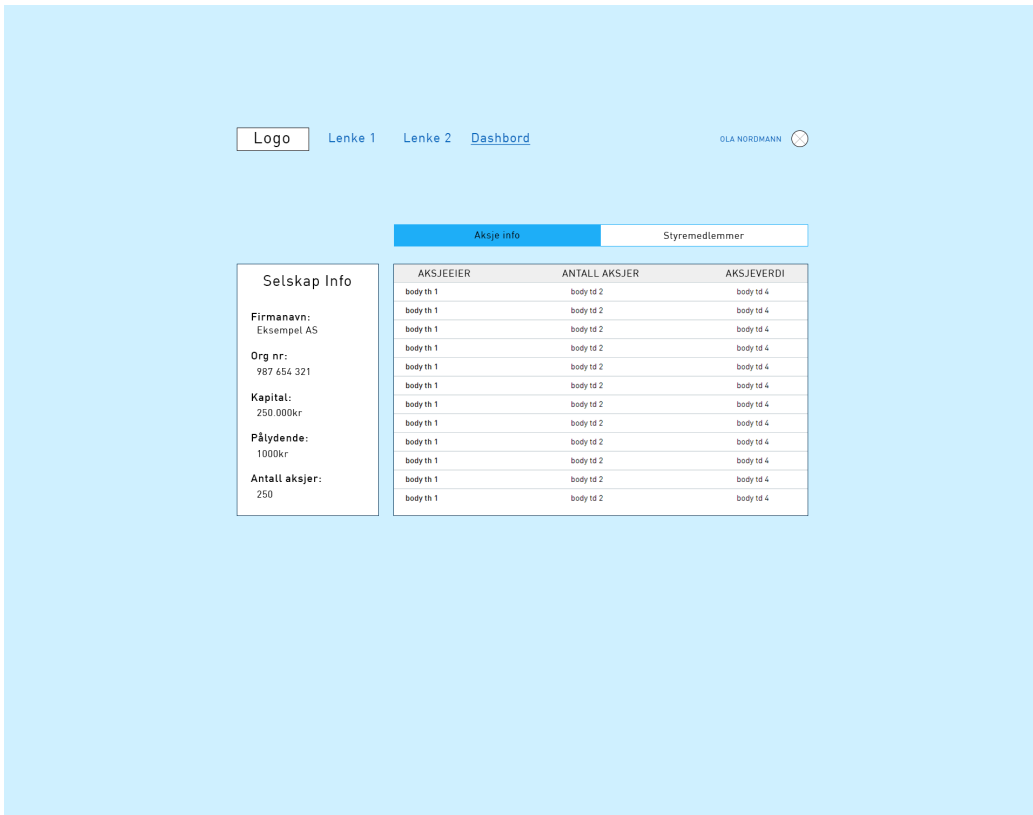


Figure 53: High-fi wireframe of company dashboard

Din firmaoversikt

Din aksjebok

ORGNR	KAPITAL	PÅLYDENDE	ANTALL AKSJER
123456789	80000	80	1000

Liste over aksjonærer

Søk etter innhold

starte og drive bedrift

hjelp og kontakt

Brønnøysundregistrene, 8910 Brønnøysund. Org.nr. 974 760 673 [Om Altinn](#) [Driftsmeldinger](#) [Personvern og cookies](#)

Figure 54: Current design of the company dashboard

10.3 Results user testing

Here we will show the test results from our UX testing. Complete with pre and post questionnaire and observational log.

Subject one:

Pre-test questionnaire results:

Gender:	Female
Age group:	39-48
Eyesight:	Nearsighted
Color blindness:	No
Handedness:	Left
Mobility impairment:	No
Comfort level with computers:	Strongly like
Observational protocol:	

Questions	Observations
Does the subject seem relaxed/at ease in the situation?	Focused and a little stressed
Does the subject appear to be used to using computers?	The subject seems familiar. Is able to locate the goal of the task reasonably quick.
Does the subject ask a lot of questions? If so, specify if questions are regarding the solution or the given instructions.	Several questions during the test on which data to fill into the different fields. Some instruction was given.
Does the subject talk during the test? If so, what is the test subject saying?	Very little outside the questions.
What kind of emotions does the subject express?	Positive about the test, other than that nothing out of the ordinary.
What kind of body language does the subject have during the test?	A little tense, but curious and interested. Seems ready to solve problems.
Does the subject spend much time trying to solve each task?	Total time spent was around average, but the subject asked a lot of questions that seemed to speed up the process. The subject also expresses that not everything is as self explanatory as she would liked it to be.
Was there any specific point throughout the test that the test subject seemed to struggle more than expected?	Not particularly. Filling in the business info took the longest however.
Did the subject complete the test? If not, please specify why not, and how far the subject got.	Completed the business part, however the solution crashed midways which caused the subject to not have sufficient time to finish the stock owner part

Post-test questionnaire

Questions	Answers
What did you like about the product?	Simple and easy to get a overview of.
What did you not like about the product?	There is no indication that the application is loading something.
What could have been done differently, and how?	
Did the product have a natural and logic flow to it? If not, what should be done to achieve this?	The subject wanted a way to get to the approval page easier, maybe a link in the navigational bar.
Does the product lack any important elements from your point of view? If so, please specify.	
What is your overall evaluation of the product?	Very good.
Any additional comments?	Would have liked to be able to see the previous value after changing it.

Subject two:

Pre-test questionnaire results:

Gender:	Male
Age group:	39-48
Eyesight:	Nearsighted
Color blindness:	No
Handedness:	Right
Mobility impairment:	No
Comfort level with computers:	Strongly like

Observational protocol:

Questions	Observations
Does the subject seem relaxed/at ease in the situation?	Yes. The subject seems curious and interested.
Does the subject appear to be used to using computers?	Yes. The subject is very exploratory and focuses on what the application lets him do.
Does the subject ask a lot of questions? If so, specify if questions are regarding the solution or the given instructions.	Very few questions asked, the subject seems very independent and understands how he should proceed.
Does the subject talk during the test? If so, what is the test subject saying?	Found a problem in some fields that we had missed. Otherwise not much was said.
What kind of emotions does the subject express?	Positive and in a good mood. Energetic.
What kind of body language does the subject have during the test?	Confident. Seems to like being a tester.
Does the subject spend much time trying to solve each task?	Subject is very thorough and spends time to explore the application fully.
Was there any specific point throughout the test that the test subject seemed to struggle more than expected?	No. Nothing worth mentioning.
Did the subject complete the test? If not, please specify why not, and how far the subject got.	Yes.

Post-test questionnaire

Questions	Answers
What did you like about the product?	Has the potential to be a good solution.
What did you not like about the product?	Some functionality does not work like in real life. Naming/help text.
What could have been done differently, and how?	Written use case for the test.
Did the product have a natural and logic flow to it? If not, what should be done to achieve this?	Would have more logic/connection woven together. (Capital and shareholder register at the same time).
Does the product lack any important elements from your point of view? If so, please specify.	Unfinished.
What is your overall evaluation of the product?	Can be used to explain the technical solution.
Any additional comments?	Lacking logic concerning adding stocks. He would have wanted the option to add stocks and increase capital in one action. Wants to be able to decide who owns what shares. Lack of custom errors. The listing of stock owners is clunky.

Subject three:

Pre-test questionnaire results:

Gender:	Male
Age group:	39-48
Eyesight:	Nearsighted
Color blindness:	No
Handedness:	Right
Mobility impairment:	No
Comfort level with computers:	Like

Observational protocol:

Questions	Observations
Does the subject seem relaxed/at ease in the situation?	Yes. The subject seems comfortable, curious and confident.
Does the subject appear to be used to using computers?	Yes. The subject seems confident and used to working with, or using a computer.
Does the subject ask a lot of questions? If so, specify if questions are regarding the solution or the given instructions.	Some questions were asked. Mostly connected to the instructions that were given.
Does the subject talk during the test? If so, what is the test subject saying?	A little bit. The subject seemed a little unconsecrated at times.
What kind of emotions does the subject express?	Normal. Positive and curious. Energetic.
What kind of body language does the subject have during the test?	Normal. Positive and curious
Does the subject spend much time trying to solve each task?	About average time solving the tasks.
Was there any specific point throughout the test that the test subject seemed to struggle more than expected?	No. But the subject did ask questions regarding the instructions during the test. Which indicates some insecurity of how things worked.
Did the subject complete the test? If not, please specify why not, and how far the subject got.	Yes.

Post-test questionnaire

Questions	Answers
What did you like about the product?	That the flow worked well.
What did you not like about the product?	No authentication. GUI errors. Comments on the solution being small.
What could have been done differently, and how?	Written use case for the test.
Did the product have a natural and logic flow to it? If not, what should be done to achieve this?	Yes.
Does the product lack any important elements from your point of view? If so, please specify.	
What is your overall evaluation of the product?	OK. As a bachelor thesis; good!
Any additional comments?	

10.4 Step-by-step guide system tests

System tests step-by-step

Test 1A-CreateStockbook

1. Open the public web application
2. Choose to log on as a business user
3. Go to "Handler" tab
4. Click the button labeled "Registrer nytt aksjeselskap"
5. Enter '123456789' in the input field labeled "Orgnr"
6. Enter '10' in the input field labeled "Antall aksjer"
7. Enter '500' in the input field labeled "Kapital"
8. Enter '4321' in the input field labeled "Eier"
9. Enter '10' in the input field labeled "Antall Aksjer" (Next to "Eier")
10. Click the button labeled "Registrer søknad"

Test 1B-ApproveStockBook

1. Open the web application developed for Brønnøysundregistrene
2. Go to "Søknader" tab
3. Click button labeled "Søknader opprettelse av aksjebok"
4. Click the button labeled "Godkjenn" for the entry with firm ID '123456789'

Test 1C-DenyStockBook

Prerequisite: Rerun of test 1A

1. Open the web application developed for Brønnøysundregistrene
2. Go to "Søknader" tab
3. Click button labeled "Søknader opprettelse av aksjebok"
4. Click the button labeled "Avslå" for the entry with firm ID '123456789'

Test 2A-IncreaseCapital (amount)

1. Open the public web application
2. Choose to log on as a business user
3. Go to "Handler" tab
4. Click the button labeled "Utvid kapital"
5. Click the button labeled "Legg til flere aksjer"
6. Enter '123456789' in the input field labeled "Orgnr"
7. Enter '10' in the input field labeled "Antall nye aksjer"
8. Click the button labeled "Lever søknad"

Test 2B-ApproveIncreaseCapital (amount)

1. Open the web application developed for Brønnøysundregistrene
2. Click the button labeled "Søknader"
3. Click the button labeled "Søknader utvide kapital"
4. Click the button labeled "Godkjenn" for the entry with firm ID '123456789'

Test 2C-DenyIncreaseCapital (amount)

Prerequisite: Rerun of test 2A

1. Open the web application developed for Brønnøysundregistrene
2. Click the button labeled "Søknader"
3. Click the button labeled "Søknader utvide kapital"
4. Click the button labeled "Avslå" for the entry with firm ID '123456789'

Test 3A-IncreaseCapital (value)

1. Open the public web application
2. Choose to log on as a business user
3. Go to "Handler" tab
4. Click the button labeled "Utvid kapital"
5. Enter '123456789' in the input field labeled "Orgnr"
6. Enter '1000' in the input field labeled "Ny kapital"
7. Click the button "Lever søknad"

Test 3B-ApproveIncreaseCapital (value)

1. Open the web application developed for Brønnøysundregistrene
2. Click the button labeled "Søknader"
3. Click the button labeled "Søknader utvid kapital"
4. Click the button labeled "Godkjenn" for the entry with firm ID '123456789'

Test 3C-DenyIncreaseCapital (value)

Prerequisite: Rerun of test 3A

1. Open the web application developed for Brønnøysundregistrene
2. Click the button labeled "Søknader"
3. Click the button labeled "Søknader utvid kapital"
4. Click the button labeled "Avslå" for the entry with firm ID '123456789'

Test 4A-BuyStock

1. Open the public web application
2. Choose to log in as a private user
3. Go to "Aksjemarked"
4. Open the entry labeled 'Orgnr: 123456789'
5. Press the button labeled 'Kjøp aksjer' for the entry with owner ID '4321'
6. Enter '10' in the input field labeled "Antall aksjer"
7. Enter '80' in the input field labeled "Bud"
8. Click the button labeled 'Gi bud'

Test 4B-ApproveBuyStockSeller

1. Open the public web application
2. Choose to log on as a private user
3. Go to "tilbud"
4. Click the button labeled 'Godta' for the entry with buyer ID '1234' and firm ID '123456789'

Test 4C-ApproveBuyStockBusiness

1. Open the public web application
2. Choose to log on as a business user
3. Go to "Handleringer"
4. Click 'Forespørsler aksjeoverdragelse'
5. Click the button labeled 'Godta' for the entry with buyer ID '1234' and seller ID '4321'

Test 4D-DenyBuyStockSeller

Prerequisite: Rerun of test 4A

1. Open the public web application
2. Choose to log on as a private user
3. Go to "tilbud"
4. Click the button labeled 'Avslå' for the entry with buyer ID '1234' and firm ID '123456789'

Test 4E-DenyBuyStockBusiness

Prerequisite: Rerun of test 4A and 4B

1. Open the public web application
2. Choose to log on as a business user
3. Go to "Handleringer"
4. Click 'Forespørsler aksjeoverdragelse'
5. Click the button labeled 'Avslå' for the entry with buyer ID '1234' and seller ID '4321'

10.5 Step-by-step guide unit tests

Unit tests step-by-step

Create Company By Bronnoysund

This is a test of the transaction CreateCompany performed by Bronnoysund

Arrange

Input for 'useIdentity()'

Identity: "TestFirma"

Input for new transaction:

Create transaction RegisterFirm with values:

capital: 200000

numberOfStock: 2

distribution: [1,1]

firmIdentifier: "332211"

newStockOwners: ["1234", "4321"]

Input for 'useIdentity()':

Identity: "Bronnoysund"

Input for new transaction:

Create transaction CreateCompany with values:

IdToTransactionWithEvent: id from transaction RegisterFirm

response: ACCEPTED

Act

Submit transaction RegisterFirm

Submit transaction CreateCompany

Assert

Expected to find register of shareholders with values:

id: 332211

capital: 200000

numberOfStock: 2

firmIdentifier: "332211"

Expected to find stock with values:

id: 1

owner: "1234"

value: 100000

registerOfShareholders: "332211"

Expected to find stock with values:

id: 2

owner: "4321"

value: 100000

registerOfShareholders: "332211"

RegisterChangeOnFirm by stock owner

This is a test of the transaction RegisterChangeOnFirm performed by stock owner Frank.

Arrange

Input for 'useIdentity()'

Identity: "Frank"

Input for new transaction

Create transaction RegisterChangeOnFirm with values:

capitalChange: "CHANGEVALUE"

newCapital: 1000

amountOfNewStocks: 20

shareholderRegistryID: "332211"

Act

Submit transaction RegisterChangeOnFirm

Assert

Expected transaction to be rejected with message '/does not have .* access to resource/'

Expected events to have length of 0.

registerChangeOnFirm by firm

This is a test of the transaction registerChangeOnFirm transaction performed by firm TestFirma.

Arrange

Input for 'useIdentity()'

Identity: "TestFirma"

Input for new transaction

Create transaction RegisterChangeOnFirm with values:

capitalChange: "CHANGEVALUE"

newCapital: 1000

amountOfNewStocks: 20

shareholderRegistryID: "332211"

Act

Submit transaction RegisterChangeOnFirm

Assert

Expected events to have length of 1.
Expected event 0 to have values:
capitalChange: "CHANGEVALUE"
newCapital: 1000
amountOfNewStocks: 20
shareholderRegistryID: "332211"

RegisterFirm by firm

This is a test of the transaction RegisterFirm by firm TestFirma.

Arrange

Input for 'useIdentity()'

Identity: "TestFirma"

Input for new transaction

Create transaction RegisterFirm with values:
capital: 200000
numberOfStock: 10
distribution: [1,1]
firmIdentifier: "332211"
newStockOwners: ["1234", "4321"]

Act

Submit transaction RegisterFirm

Assert

Expected transaction to be rejected with message 'Du har enten delt ut for mange eller for få aksjer i forhold til hva du ønsker å opprette!'
Expected events to have length of 0.

RegisterFirm by firm

This is a test of the transaction RegisterFirm performed by firm TestFirma.

Arrange

Input for 'useIdentity()'

Identity: "TestFirma"

Input for new transaction

Create transaction RegisterFirm with values:

capital: 200000

numberOfStock: 10

distribution: [5,5]

firmIdentifier: "332211"

newStockOwners: ["1234"]

Act

Submit transaction RegisterFirm

Assert

Expected transaction to be rejected with message 'Du har delt ut aksjer til for få eller for mange nye eiere i forhold til hvor mange personer du har skrevet inn som eier av nye aksjer!'

Expected events to have length of 0.

registerFirmByTestfirma

RegisterFirm transaction ..

Arrange

Input for 'useIdentity()'

Identity: "TestFirma"

Input for new transaction

Create transaction RegisterFirm with values:

capital: 200000

numberOfStock: 2

distribution: [1,1]

firmIdentifier: "332211"

newStockOwners: ["1234", "4321"]

Act

Submit transaction RegisterFirm

Assert

Expected events to have length of 1.

Expected event 0 to have values:

capital: 200000

numberOfStock: 2

distribution:[1,1]

firm: "332211"

newStockOwners: ["1234", "4321"]

Expected BRREG to have value receivedEstablishFirmRequest of length 1 with values:

response: "PENDING"

IdToTransactionWithEvent: id from event created in transactionRegisterFirm

Expected firm with id "332211" to have value

receivedEstablishFirmRequest of length 1 with values:

response: "PENDING"

IdToTransactionWithEvent: id from event created in transaction

RegisterFirm

ExpandCapital by Bronnoysund

This is a test of the transaction ExpandCapital performed by Bronnoysund.

Arrange

Input for 'useIdentity()'

Identity: "ACME"

Input for new transaction

Create transaction RegisterChangeOnFirm with values:

capitalChange: "CHANGEVALUE"

newCapital: 1000

amountOfNewStocks: 20

shareholderRegistryID: "112233"

Input for 'useIdentity()'

Identity: "Bronnoysund"

Input for new transaction

Create transaction ExpandCapital with values:

IdToTransactionWithEvent: id from transaction RegisterChangeOnFirm

response: "ACCEPTED"

Act

Submit transaction RegisterChangeOnFirm

Submit transaction ExpandCapital

Assert

Expected firm with id "112233" to have changeOnFirmRequest of length 1
with values:

response: "ACCEPTED"

Expected BRREG to have value receivedEstablishFirmRequest of empty

Expected firm with id "112233" to have values:

capital: 1000

numberOfShares: 2

Expected stock with id "1" to have values:

value: 500

registerOfShareholders: "112233"

Expected stock with id "2" to have values:

value: 500

registerOfShareholders: "112233"

ExpandCapital by Bronnoysund

This is a test of the transaction ExpandCapital performed by Bronnoysund

..

Arrange

Input for 'useIdentity()'

Identity: "ACME"

Input for new transaction

Create transaction RegisterChangeOnFirm with values:

capitalChange: "CHANGEVALUE"

newCapital: 1000

amountOfNewStocks: 20

shareholderRegistryID: "112233"

Input for 'useIdentity()'

Identity: "Bronnoysund"

Input for new transaction

Create transaction ExpandCapital with values:
IdToTransactionWithEvent: id from event created in transaction
RegisterChangeOnFirm
response: "REJECTED"

Act

Submit transaction RegisterChangeOnFirm
Submit transaction ExpandCapital

Assert

Expected firm with id "112233" to have value changeOnFirmRequest of
length 1 with values:
response: "REJECTED"

Expected BRREG to have value receivedChangeOnFirmRequest of empty

ExpandCapital by Bronnoysund

This is a test of the transaction ExpandCapital performed by Bronnoysund.

Arrange

Input for 'useIdentity()'

Identity: "ACME"

Input for new transaction

Create transaction RegisterChangeOnFirm with values:
capitalChange: "CHANGEAMOUNT"
newCapital: 1000
amountOfNewStock: 20
shareholderRegistryID: "112233"

Input for 'useIdentity()'

Identity: "Bronnoysund"

Input for new transaction

Create transaction ExpandCapital with values:

IdToTransactionWithEvent: id from event created in transaction

RegisterChangeOnFirm

response: "ACCEPTED"

Act

Submit transaction RegisterChangeOnFirm

Submit transaction ExpandCapital

Assert

Expected transaction to be rejected with message 'Must be option
CHANGEVALUE'

AddStocks by Bronnoysund

This is a test of the transaction AddStocks performed by Bronnoysund ..

Arrange**Input for 'useIdentity()'**

Identity: "ACME"

Input for new transaction

Create transaction RegisterChangeOnFirm with values:

capitalChange: "CHANGEVALUE"

newCapital: 1000

amountOfNewStocks: 20

shareholderRegistryID: "112233"

Input for 'useIdentity()'

Identity: "Bronnoysund"

Input for new transaction

Create transaction AddStocks with values:

IdToTransactionWithEvent: id from event created in transaction

RegisterChangeOnFirm

response: "ACCEPTED"

Act

Submit transaction RegisterChangeOnFirm

Submit transaction AddStocks

Assert

Expected transaction to be rejected with message 'Must be option
CHANGEAMOUNT'

AddStocks by Bronnoysund

This is a test of the transaction AddStocks performed by Bronnoysund.

Arrange**Input for 'useIdentity()'**

Identity: "ACME"

Input for new transaction

Create transaction RegisterChangeOnFirm with values:

capitalChange: "CHANGEAMOUNT"

newCapital: 1000

amountOfNewStocks: 2

shareholderRegistryID: "112233"

Input for 'useIdentity()'

Identity: "Bronnoysund"

Input for new transaction

Create transaction AddStocks with values:
IdToTransactionWithEvent: id from event created in transaction
RegisterChangeOnFirm
response: "REJECTED"

Act

Submit transaction RegisterChangeOnFirm
Submit transaction AddStocks

Assert

Expected firm with id "112233" to have changeOnFirmRequest of length 1
with values:
response: "REJECTED"

Expected BRREG to have value receivedChangeOnFirmRequest of empty

RequestPurchase by stock owner

This is a test of the transaction requestPurchase performed by stock owner
Eva.

Arrange

Input for 'useIdentity()'

Identity: "Eva"

Input for new transaction

Create transaction requestPurchase with values:
bid: 60000
quantity: 1
customer: "4321"
stockOwner: "1234"
registerOfShareholders: "112233"

Act

Submit transaction requestPurchase

Assert

Expected events to have length of 1.

Expected event 0 to have values:

bid: 60000

quantity: 1

customer: "4321"

stockOwner: "1234"

registerOfShareholders: "112233"

Expected stock owner with id "1234" to have value

receivedPurchaseRequests of length 1 with values:

response: "PENDING"

IdToTransactionWithEvent: id from event created in transaction

requestPurchase

Expected stock owner with id "1234" to have value pendingRequests of length 1 with values:

response: "PENDING"

IdToTransactionWithEvent: id from event created in transaction

requestPurchase

respondToPurchaseRequest by stock owner

This is a test of the transaction respondToPurchaseRequest performed by stock owner Eva.

Arrange

Input for 'useIdentity()'

Identity: "Eva"

Input for new transaction

Create transaction requestPurchase with values:

bid: 60000

quantity: 1

customer: "4321"

stockOwner: "1234"

registerOfShareholders: "112233"

Input for 'useIdentity()'

Identity: "Frank"

Input for new transaction

Create transaction respondToPurchaseRequest with values:

transID: id from event created in transaction requestPurchase

response: "ACCEPTED"

Act

Submit transaction requestPurchase

Submit transaction respondToPurchaseRequest

Assert

Expected firm with id "112233" to have value awaitingStockPurchase of length 1 with values:

IdToTransactionWithEvent: transID from transaction

respondToPurchaseRequest

response: "PENDING"

RespondToPurchaseRequest by stock owner

This is a test of the transaction respondToPurchaseRequest performed by stock owner Frank ..

Arrange

Input for 'useIdentity()'

Identity: "Eva"

Input for new transaction

Create transaction requestPurchase with values:

bid: 60000

quantity: 1

customer: "4321"

stockOwner: "1234"

registerOfShareholders: "112233"

Input for 'useIdentity()'

Identity: "Frank"

Input for new transaction

Create transaction respondToPurchaseRequest with values:

transID: id from event created in transaction requestPurchase

response: "REJECTED"

Act

Submit transaction requestPurchase

Submit transaction respondToPurchaseRequest

Assert

Expected stock owner Frank to have value receivedPurchaseRequests of empty.

Expected stock owner Eva to have value pendingRequests of length 1 with values:

response: "REJECTED"

SaleOfStock by firm

This is a test of the transaction SaleOfStock performed by firm ACME

Arrange

Input for 'useIdentity()'

Identity: "Eva"

Input for new transaction

Create transaction requestPurchase with values:

bid: 60000

quantity: 1

customer: "4321"

stockOwner: "1234"

registerOfShareholders: "112233"

Input for 'useIdentity()'

Identity: "Frank"

Input for new transaction

Create transaction respondToPurchaseRequest with values:

transID: id from event created in transaction requestPurchase

response: "ACCEPTED"

Input for 'useIdentity()'

Identity: "ACME"

Input for new transaction

Create transaction SaleOfStock with values:

transID: id from transaction requestPurchase

response: "ACCEPTED"

Act

Submit transaction requestPurchase

Submit transaction respondToPurchaseRequest

Submit transaction SaleOfStock

Assert

Expected stock with id "1" to have values:

owner: "4321"

value: 60000