REPORT

Group Brukerfeil:
Gorm-Erik Aarsheim, Vegard Alvsaker, Ingve Fosse, Marius Johansen Holen, Espen Sivertsen Oftedal, Kevin Archival Smørdal

Course code	IS-304
Course name	Bachelor Thesis in Information Systems
Course responsible	Hallgeir Nilsen
Deadline	May 20th, 2020
Number of words (whole document)	18839
Project/product title	Report IS-304

We confirm that we do not cite or otherwise use other people's work without this being stated and that all references are listed.	Yes X	No

Can the report be used for teaching purposes?	Yes X	No

We confirm that everyone in the group has contributed to the report	Yes X	No

Report IS-304

University of Agder
Institute of Information Technology
Group 4, Brukerfeil

Preface

This report has been written as a part of the course IS-304 Bachelor Thesis in Information Systems. The report will discuss and document how the bachelor project was executed, central decisions made, and reflect on the team's outcome of the project.

Thanks to Merethe Sjøberg, for allowing us to conduct our bachelor project at Sikri, acting as the product owner during the course of the project, and following up on executed work with crucial and honest feedback.

Thanks to Geir Inge Hausvik, for excellent feedback and input.

Thanks to Sikri and their employees, for dedicating their time and effort in facilitating a work environment and providing technical advice.

Kristiansand, University of Agder, Spring of 2020

The authors of this report are:

Gorm-Erik Aarsheim:	gormea17@uia.no
Vegard Alvsaker:	vegaal17@uia.no
Ingve Fosse:	ingvef17@uia.no
Marius Johansen Holen:	marijh17@uia.no
Espen Sivertsen Oftedal:	espeno17@uia.no
Kevin Archival Smørdal:	kevins16@uia.no

Summary

As a part of the bachelor programme in IT and Information systems, the group Brukerfeil has conducted their semester-long bachelor project at Sikri AS. Sikri is a software company that produces a case processing and archive system named Elements. Users of Elements are case workers in municipal institutions, amongst others. The product, named Enode, is a visual presentation of the status of incoming and outgoing messages expedited through the Elements system. The purpose of Enode is to be used as a tool for Sikri and their customers as a means of tracing failed messages as well as verifying that messages have been successfully expedited. A representative from Sikri has acted as the product owner for this project. Together with the product owner and technical staff at Sikri, the concept and scope of the project was developed.

The team was looking for a challenging project which could develop the team's technical understanding and skills. The team specifically selected this project because it provided the opportunity to work with modern technologies in developing a larger web application. The product consists of a backend and a frontend. The backend was developed using ASP.NET Core, and the frontend was developed in React with TypeScript.

Throughout the project, the Scrum framework has been utilized. In order to maintain progress and to sort out problems, the team has iteratively reflected on what has made the sprints go well and what could have been improved. Marius Johansen Holen acted as the team's Scrum master throughout the project and this has helped ensure continuous progress. Azure DevOps was used as a Scrum board for managing the time in relation to the scope.

The team has encountered a multitude of challenges throughout the course of the project. The project scope has gone through moderate changes as it was defined based on assumptions which turned out to not be correct once further investigation was performed. For example, it was not possible to retrieve all the data that was assumed possible early on in the project. In addition, the Corona virus has become an obstacle in allowing the team to physically work together. Due to an agile development methodology, the team has been able to create quality in the product despite the changes that have occurred.

By the end of the semester, the team had successfully managed to develop a functional web application. Overall quality has been ensured by carrying out acceptance tests along with the product owner, ensuring that requirements have been fulfilled. Technical quality has been ensured by properly unit-testing the application. The team has gained valuable experience and knowledge by conducting this project, both in terms of technical experience as well as experience in development processes and project management.

Table of contents

Table of figures

1. Introduction

The group Brukerfeil has during the spring of 2020 conducted their bachelor project at Sikri AS. The team consists of six members with different backgrounds and skill sets, making the team diverse and resilient. The project started in January and finished in the middle of May.

Sikri was formed in January 2020, when it was separated from the IT-company EVRY. The company has approximately 100 employees, out of which 30 are software developers. Elements is a case management and archiving system which is developed and maintained by Sikri. It is widely used by public agencies, municipalities and organizations across Norway. In total Sikri serves more than 400 customers in Norway and Sweden.

The team was tasked with developing a web application meant to function as a monitoring tool for Sikri and their customers. The product gives a visual presentation of the status of messages that have been expedited in and out of Elements to see if messages have been expedited successfully or if they have failed to send.

The agile methodology Scrum has been used as the project management framework. The team has followed processes and principles from the framework throughout the course of the project, implementing an iterative work process. The project has gone through an analysis phase, execution phase and lastly a closure phase, preparing the product to be handed over to Sikri. The decisions and approaches that the team has carried out will be described in later chapters.

The main objective of this report is to document Brukerfeil's bachelor project and process. It details the product and its specifications, in addition to what has been done in order to ensure a high level of quality in the end product. The report describes the different challenges encountered, and as this is a bachelor project, it also details the experience gained from the project. Finally, the report serves as documentation for the Enode system for the benefit of Sikri.

2. Product

The product, named Enode, is a visual presentation of the status of incoming and outgoing messages expedited through the Elements system. The team developed Enode over the course of seven sprints in addition to one pre-sprint (16 weeks in total). Enode consists of a backend to fetch data and a frontend to display the information to the user. It was developed by the team upon request from Sikri AS. In this chapter we are going to detail the product further and run through a demo of Enode as it stands today.

2.1 Product Description

Enode is a standalone system to be used in relation with Elements. Elements is a case management and archive system which is used for managing cases and applications. Users of the system are case workers in municipal institutions. Case workers can send and receive messages regarding cases and applications. A building application is an example of such an application, as this needs to be processed by a case worker. Difi is the governmental digitalization agency and a part of its responsibility is storing public cases and applications. Elements exchanges messages regarding cases and applications with Difi. Both incoming and outgoing messages can originate from either an Elements user, another organization's case management and archive system or an individual person. After the message is sent it is handled by Difi before being passed to the recipient. Sikri discovered a demand for additional functionality for tracking the status of messages expedited between Elements and Difi, and how failed deliveries of messages could be detected. The purpose of Enode is to display the status of a message when being transferred between Elements and Difi.

The demand for additional functionality stems from Elements users and Sikri employees wanting an easy way of tracking the status of messages. The aim of Enode is to improve how to detect failed deliveries of messages. Data is added and modified continuously as cases are processed. The system acts as a status dashboard for digital messages transferred between Elements and Difi. The list of messages can be filtered by its latest status. It can also be sorted in ascending or descending order by creation date, date of last update, status or by its ID. By using the search fields, one can search for a specific message by its ID, or filter the list to only show messages to or from a desired sender or receiver. The user can also select a specific message to show a timeline of all its statuses. The system has been deployed using container technology, meaning it is running in an environment where the application and its dependencies are isolated from other processes (DeMuro, 2019). Enode is hosted by Sikri in a Kubernetes cluster, which ensures continuous deployment and scalability.

The application consists of a frontend and a backend. The backend fetches data from the Difi API and the Elements API. The Difi API is an API that interacts with the governmental database for case archiving. It is an open API and is accessible by Elements and other systems. The Elements API is publicly restricted and used to interact with a database used only by the customers of Sikri. It is used for storing information about a case and for archiving cases that are finished. The frontend fetches data from the backend and displays it as a list of messages with their corresponding statuses.

2.2 Product Demo

The product demo was created as an informational video for readers of this report. It was also done in order for Sikri to show to their customers, and for the benefit of other interested parties. The demo displays a typical user's use case for Enode. It clarifies how the system retrieves information, which describes the manual processes solved by the system.
Brukerfeil's demo for Enode can be found here: https://youtu.be/ZOkuQkkvBK0

3. Project Decisions

Over the course of this project, several important decisions were made. Some decisions were made by the team alone, while decisions with more impact on the product design and functionality were influenced by the product owner. This chapter will describe and discuss the decisions taken regarding development methodologies, technologies and other administrative matters such as time management.

3.1 Project Management

To ensure a collaborative workflow within the team, it was important to set a line of tools to use for different needs. These decisions were decided early in the project and the tools were used by the entire team throughout the project. The methodology used in this project was Scrum, an agile method framework to manage the project.

3.1.1 Scrum

The team desired to use an agile method for development. The main reason the team chose to utilize the Scrum framework is that it implements iterative development processes, providing the opportunity to effectively handle change (Drumond, 2018). In previous projects the team has utilized the project management framework and been satisfied with the processes it provided. Scrum was chosen over e.g. Kanban, because the team appreciated the structures it could bring into the project, both in terms of roles, routines and meetings. It also implements a clearly defined scope, which does not necessarily come with other project management frameworks (Rehkopf, 2018 b). Kanban is usually applied if the project scope is unclear and requires continuous evaluation and incremental changes to the system (Hofmann, 2020 b, p. 38-39). A Scrum project consists of a series of sprints, which are short, time-bound periods, during which a team works on a set of defined tasks. For this project the sprints lasted for two weeks.

Scrum master is a designated role in Scrum. Marius Johansen Holen was assigned this role. He had the responsibility of arranging and leading Scrum events, and for making sure it added value to the team. He was also responsible for removing impediments in order for the team to maintain progress. The Scrum master took care of administrative tasks such as planning for meetings and communicating with the stakeholders (Scrum.org, 2017). By doing this, the team members could focus on the tasks which they were assigned.

3.1.2 Management Tools & Version Control

In the planning phase of the project, the team decided to use Azure DevOps as the project management tool. The team had preexisting experience with the tool and knew of the benefits that its comprehensive functionality provided. Sikri and several other renowned IT-companies utilize Azure DevOps, which was another reason why the team decided to use the tool. Azure DevOps is designed specifically for managing system development projects and is developed and maintained by Microsoft (Microsoft, 2019 b).

The tool provided a Scrum board for breaking user stories into multiple features and tasks in a tree-based breakdown structure. Tasks were then distributed amongst team members, and estimated the time it would take to finish the tasks. Time-estimation was done in plenary to let members voice their opinions. Azure DevOps enabled dividing the project into sprints, where tasks could be linked to each sprint. The tool generated statistics such as burndown charts (Appendix, Picture 1-1,5, Sprint 1-6 Burndown Chart, p. 34-39), which displayed the total amount of work remaining each day of a sprint with a descending curve (Rehkopf, 2018 a).

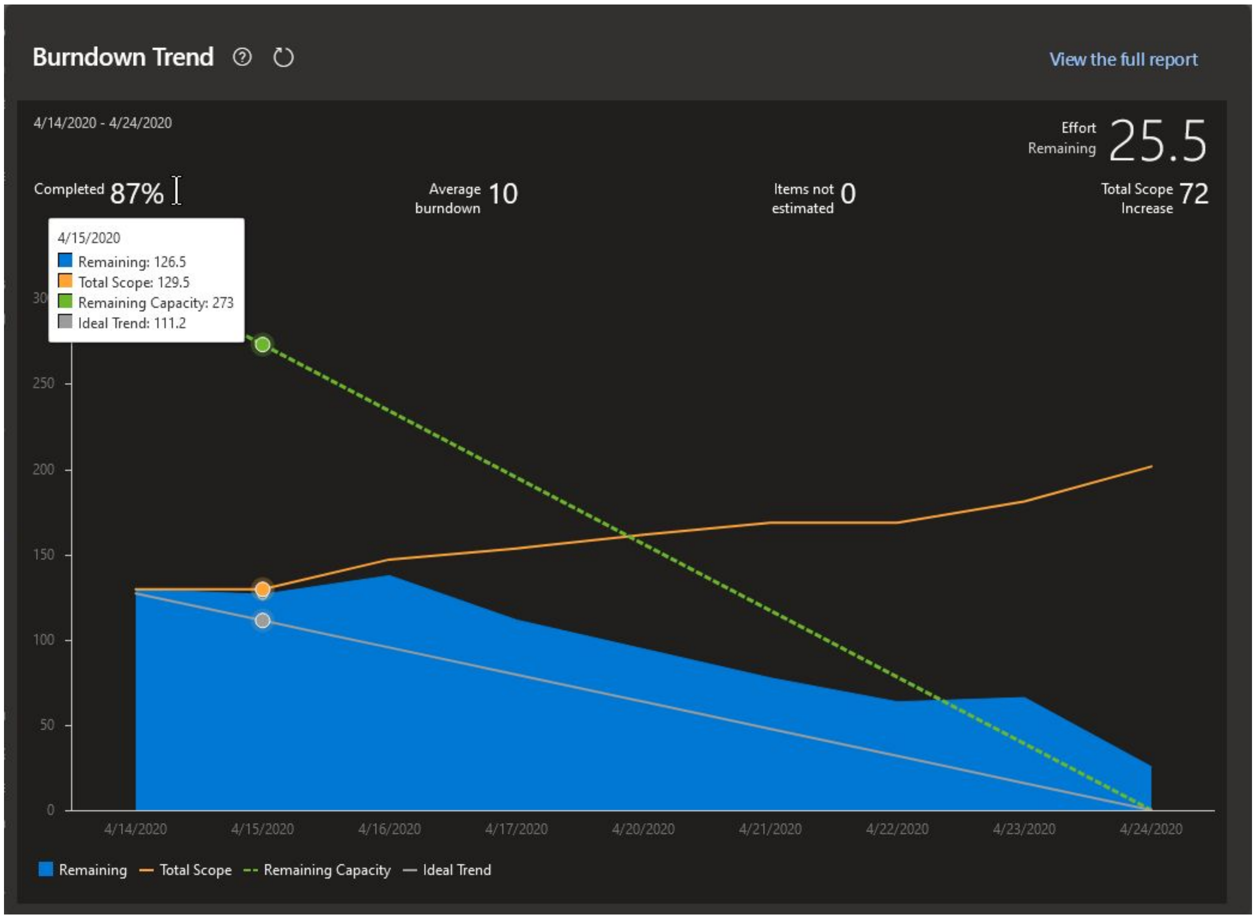


Figure 1, Sprint 6 Burndown chart

This came in handy when estimating how much time a task would take to implement in later sprints. Burndown charts also served as documentation of the workflow. Azure DevOps implemented the three important steps for successful planning: allowing the team to create a work breakdown structure, arranging the tasks in a prioritized order, and communicating the plan amongst the team members (Hofmann, 2020 a, p. 8-9).

Azure DevOps is a complete management tool in that it integrated several different elements from the development process, which is one of the main reasons why the team decided to use this tool. Azure DevOps also made it possible to integrate a Git repository as a part of the project management tool, where the team's code was stored and used for version control. This made browsing code from the repository possible directly in the web browser. In addition, work items in the scrum board could be linked with code and pull requests in the repository. Further, DevOps made it easier to create build- and deployment pipelines, resulting in the possibility of automatically deploying the application to a website when a sprint had ended.

To attain cohesive project management, the team had one Azure DevOps board for the systems development, and a separate board for administrative tasks such as reports, assignments and meetings, see (Appendix, Picture 1.6, Azure DevOps Enode Administrator Backlog, p. 40). This way the team could separate tasks that were directly related to the development of the product from tasks regarding writing reports and assignments.

3.2 Technology

The main technologies used during this project are: ASP.NET Core, React.js and Docker. These technologies are all open source and are commonly used in web development. React was used in the frontend, ASP.NET Core in the backend, and Docker was used to support deployment of the application. The relevant technologies will be described in more detail in this chapter.

3.2.1 ASP.NET Core

ASP.NET Core is an open-source web framework, developed and maintained by Microsoft (Microsoft, 2019 a). The framework uses C# and is a popular framework for developing web applications. Some members had experience using this framework and were aware of its qualities. The team chose to use this framework because it is a modern and popular framework widely used by IT-companies also including Sikri. It was beneficial and recommended by Sikri to use the same technologies that Elements uses, in order for smoother integration and to make it easier for Sikri to further develop and maintain the application. This also allowed the team to receive technical advice from professionals at Sikri, if needed.

3.2.2 React & TypeScript

For the frontend of the application the team decided to use React with TypeScript. React is a modern and popular JavaScript library, widely used by IT-companies for creating frontend applications. The team decided to include TypeScript as this offers type-checking for JavaScript at compile-time, increasing efficiency by minimizing type errors. Elements' frontend is developed using React, which makes for a more seamless transition when handing the application over to Sikri. Two members had some experience using the library, making it possible to apply good practices from the start, as well as quickly getting the rest of the team up to speed with the development.

3.2.3 Docker

The team used Docker for containerizing the application. Containerization is the process of building an application into lightweight, executable packages called containers (docker.com, 2018). Containers can be run independently and includes everything needed to run the application. Containerizing the application allowed the team to deploy the application to a website, hosted by Sikri, without being inside of Sikri's environment. Deploying the application to a website was part of the project scope, and a necessity to successfully complete the project.

3.2.4 Code Standards and Folder Structures

Already from the start of the project, the team wanted to incorporate clear standards and structures into the development of the product. This made the code more readable, thus making code reviews easier to perform. Also, the standard documents acted as a rulebook, making it easy to flag code snippets which did not adhere. Both the frontend and backend had logical and cohesive folder structures, ultimately making navigating files easier. The "Prettier" Visual Studio Code extension created a standardized format on the code, providing better readability of the code. The team also incorporated a standard naming convention of Git branches. Branch names had the following naming convention; "backend/feature/fetchMessages", specifying backend or frontend, what type of task was to be worked on, and the actual task that would be implemented. This convention described the branch's task, and also created a folder structure in the Git repository, enabling a better structure and navigation.

3.3 Time Management

In order to manage the time in the project, the team developed a schedule with estimates for when the larger activities should be completed. These activities represented project milestones and acted as sprint goals. The Minimum Viable Product (MVP) is the minimum amount of features the system needs to function as intended and to attract early adopters (ProductPlan, 2020 a). It consists of features prioritized as must-have, according to the MoSCoW-model (ProductPlan, 2020 b). The team aimed to complete the most important activities first, and then work on activities down the prioritization list. Managing the schedule this way put pressure on the team to complete activities within the deadlines that had been set, which ultimately created a more efficient workflow.

Another means of time management on a smaller scale, was time estimation of product backlog items. Estimating how long tasks from the backlog would take to implement gave an indication of how many hours of work went into a single sprint. The team could adjust the work amount according to how many hours the team had available in a sprint. Estimating how long tasks would take to implement was however a difficult task as the team started off with little experience with time-estimation. As the team got more experience with the processes and development, a better intuition was developed, resulting in more accurate time estimations. The team worked from 09:00 to 16:00 every weekday throughout the project. This created a good work routine, and also made planning easier, as the team knew how much time was at disposal each sprint.

3.4 Work Location

To be able to be as productive as possible, the team wanted an office space where good work routines, collaboration and communication could be developed. Sikri was able to provide an efficient workspace with good utilities for conducting the project. Because of this, it was desired by the team to use these as much as possible.

Due to the outbreak of the coronavirus, the offices of Sikri had to close down. The team therefore had to adjust their work environment, and start working from home. The choice of working from home, without no longer meeting in person, was a strategic decision to lower the risk of getting sick and possibly impacting the project. In the beginning the team saw this as a challenge, believing that productivity would be affected. Working from home did not really have an impact on individual productivity. However, when the team was situated at the Sikri offices, there was a lower bar for asking questions, or having a chat with any of the professionals.

4. Project Execution

This chapter describes the different activities the team performed between the start and the end of the project. The project lasted for 8 sprints, including one pre-sprint. During this time, the team performed the project planning, the development of the product, and quality assurance.

4.1 Analysis (pre-sprint)

The team started off the analysis-phase with an introduction to Sikri and a demonstration of Elements. The team learned Sikri's expectations towards the product and started planning for how the team would conduct the project. The pre-sprint marked the project's start and lasted for two weeks. During this time, the team laid the foundations for the project.

4.1.1 Product Scope

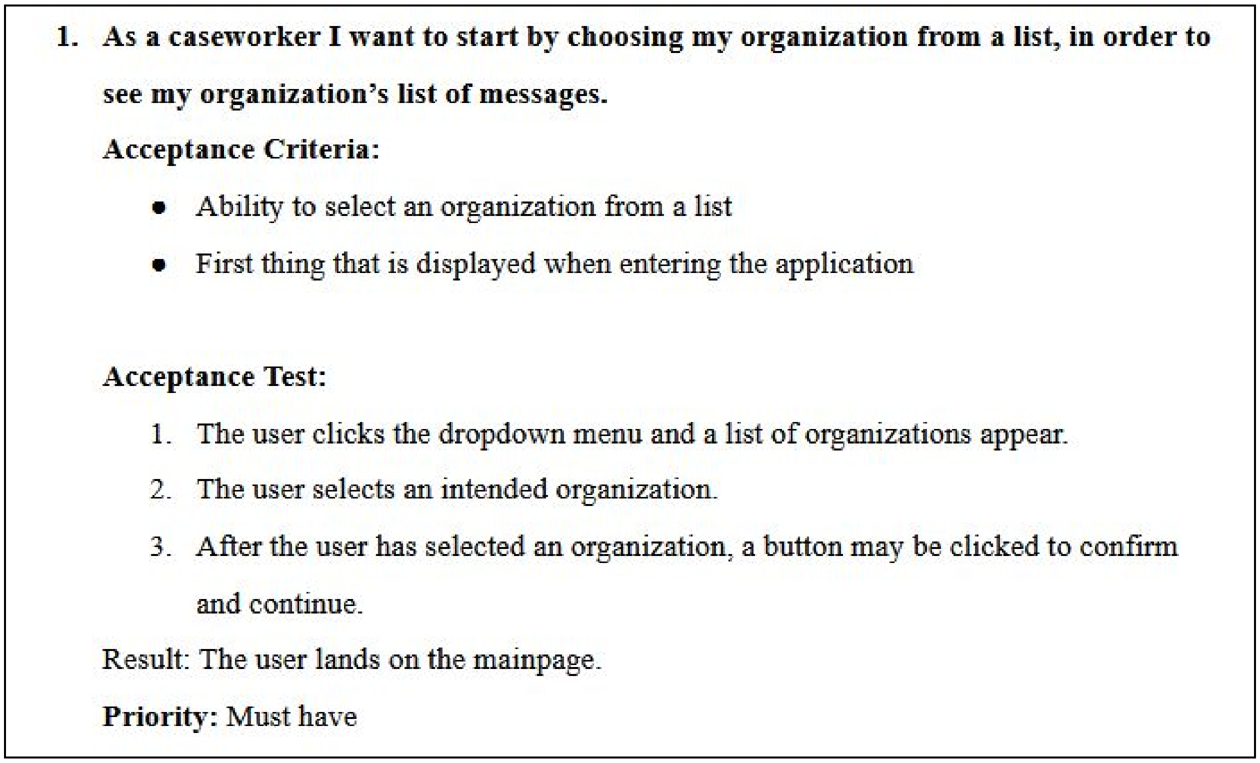
The concept of the product was a suggestion from Sikri, as this represented functionality that they would like to have in Elements. After the concept of the product had been established, the team, together with the product owner, began developing the project scope. Sikri's representative Merethe Sjøberg had the role of product owner and became the project's primary stakeholder. The scope was made out of a set of requirements which defined the functionality that would be implemented into the product in order for it to be functioning. The requirements would later be written into user stories, representing functionality from a user's perspective.

4.1.2 Specification

A specification was developed after gathering enough information from the product owner, see (Appendix, Technical specification, p. 50). The specification details elements such as the users of the system, the system goal, technologies and MVP. The document includes functional requirements which are the functional features in the system. Non-functional requirements such as usability and reliability are also described. The specification helped all members of the team in understanding the goal of the product, and the technologies which were going to be utilized.

4.1.3 User Stories

Based on the functional requirements, the team developed user stories and prioritized them together with the product owner by utilizing the MoSCoW-model; a prioritization technique used for managing system requirements (ProductPlan, 2020 b). MoSCoW is used to identify which features are most important to implement, and which features are less important for the system to function. The features defined as "must-haves" formed the MVP as these were vital for the system to function. This was a quite thorough and comprehensive process which set the foundation for the rest of the project. With experience from previous system development projects, the team was aware that the user stories would be broken into several smaller tasks and would become the base of the work to come. Here is an example of a user story which was developed:



1. **As a caseworker I want to start by choosing my organization from a list, in order to see my organization's list of messages.**

   Acceptance Criteria:

   - Ability to select an organization from a list
   - First thing that is displayed when entering the application

   Acceptance Test:

   1. The user clicks the dropdown menu and a list of organizations appear.
   2. The user selects an intended organization.
   3. After the user has selected an organization, a button may be clicked to confirm and continue.

   Result: The user lands on the mainpage.

   Priority: Must have

Figure 2, User story 1

4.2 Design

The team decided on a design that conforms with Elements' design, see (Appendix, Picture 4, Elements landing page, p. 85). This made sense as the product is meant to be used as a complimentary piece of software in conjunction with Elements. It was therefore important to carry on the brand of Elements in Enode. Styling was done to be familiar to the users of Elements and therefore improving the usability of the application (Daily Life Science, 2013, 1:04).

4.2.1 Sketches

The team developed sketches collectively on a whiteboard, see Picture X. The team determined the layout of the application, which would be central to the rest of the project. The usages of these sketches helped greatly with envisioning and clarifying the design and layout of the application. It also enabled the members of the team to give feedback to each other, and to receive feedback from the product owner. (Vasilakopoulou, 2017, p. 24).

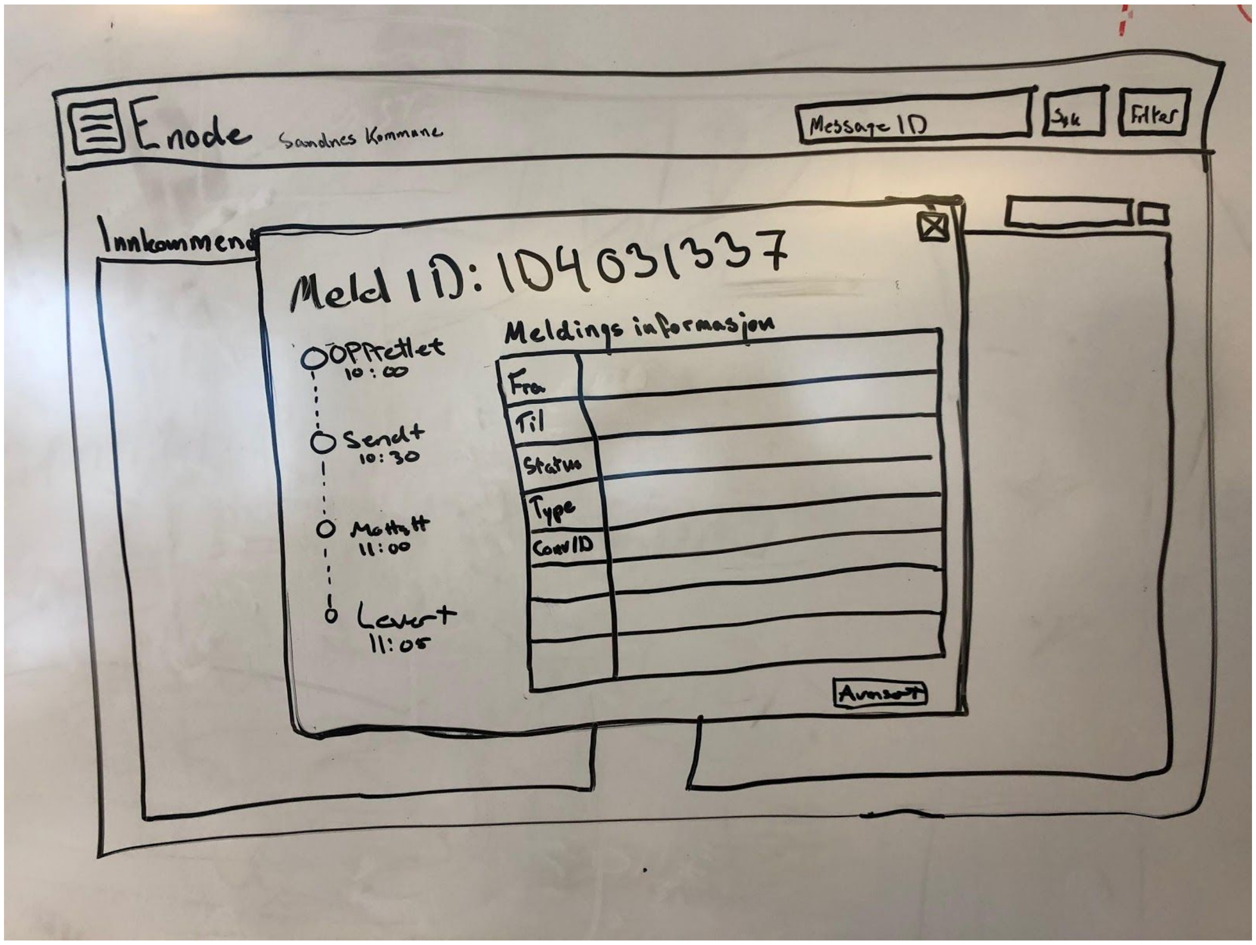


Figure 3, Sketch of Selected message

4.2.2 Prototypes

The prototypes were developed iteratively, where one user story or feature was prototyped at a time. The prototyped feature was then shown to the product owner and the rest of the team, to receive feedback. By doing this, the prototype could be adjusted continuously to conform with the received feedback. Multiple prototype versions were also developed in order to envision and reflect on different solutions, see (Appendix, Picture 3-3.7, Prototype filter 1-7, p. 61-68). This iterative process was performed until the design aligned with the wishes of the product owner.

4.3 Technical Analysis

This chapter describes activities carried out during the technical analysis. It describes the application's architecture, structural analysis and explains how data is fetched, modeled and displayed in the application.

4.3.1 System Architecture

A sketch of the system architecture was developed to create a clearer picture for both the team and the product owner regarding how the components in the system are tied together. (Appendix, Picture 2, Application architecture, p. 60). This helped with visualizing the system on a higher level. At the same time it also aided the individual team members with seeing which part of the system they would be working on.

Backend

When designing the architecture for the backend service the team focused on creating an application with a high degree of cohesion and decoupling, as these are good practices in system architecture (Nilsen, 2020, p. 8, 9). This resulted in creating sub-projects for each logical part, which gave a clear understanding of which projects were coupled.
These parts were:

- API - controller classes which handle HTTP requests and responses to and from the frontend, and also configuration retrieval
- Services - classes which handle manipulation or validation of some data
- Repositories - classes which fetch and serialize data from external APIs
- Common - classes which are used by several of the projects (e.g. data models, interfaces)
- Tests - classes which performs unit tests on all the above projects

Each project would normally have a reference to the "Common" project, where the classes in the project would use the interfaces (located in "Common") and not the actual implementation. This made testing easier, because class dependencies could be mocked by their interfaces, instead of using their actual implementation. In practice, this meant that unit tests results would not be affected by dependencies, because these were set up to behave identical for each test run.

Frontend

The frontend architecture (*Figure 4, Frontend Architecture*) is better explained in a tree structure. Using React, the team tried to design features into components which would be reusable and enable the state to be kept in the top-level. This meant fetching data (messages) in the top-level component, saving it in a state and passing the data to the child components. Some components ended up being quite large due to both extensive logic and UI rendering. This was resolved by extracting the logic into one component, and the UI rendering into another component. The UI component would then be a child component of the logic parent. This is visualized by blue headers for logic (container) components and green headers for UI rendering components. The dotted lines in the figure represent components that are rendered conditionally, such as “Filterbox” or “HamburgerMenu”.

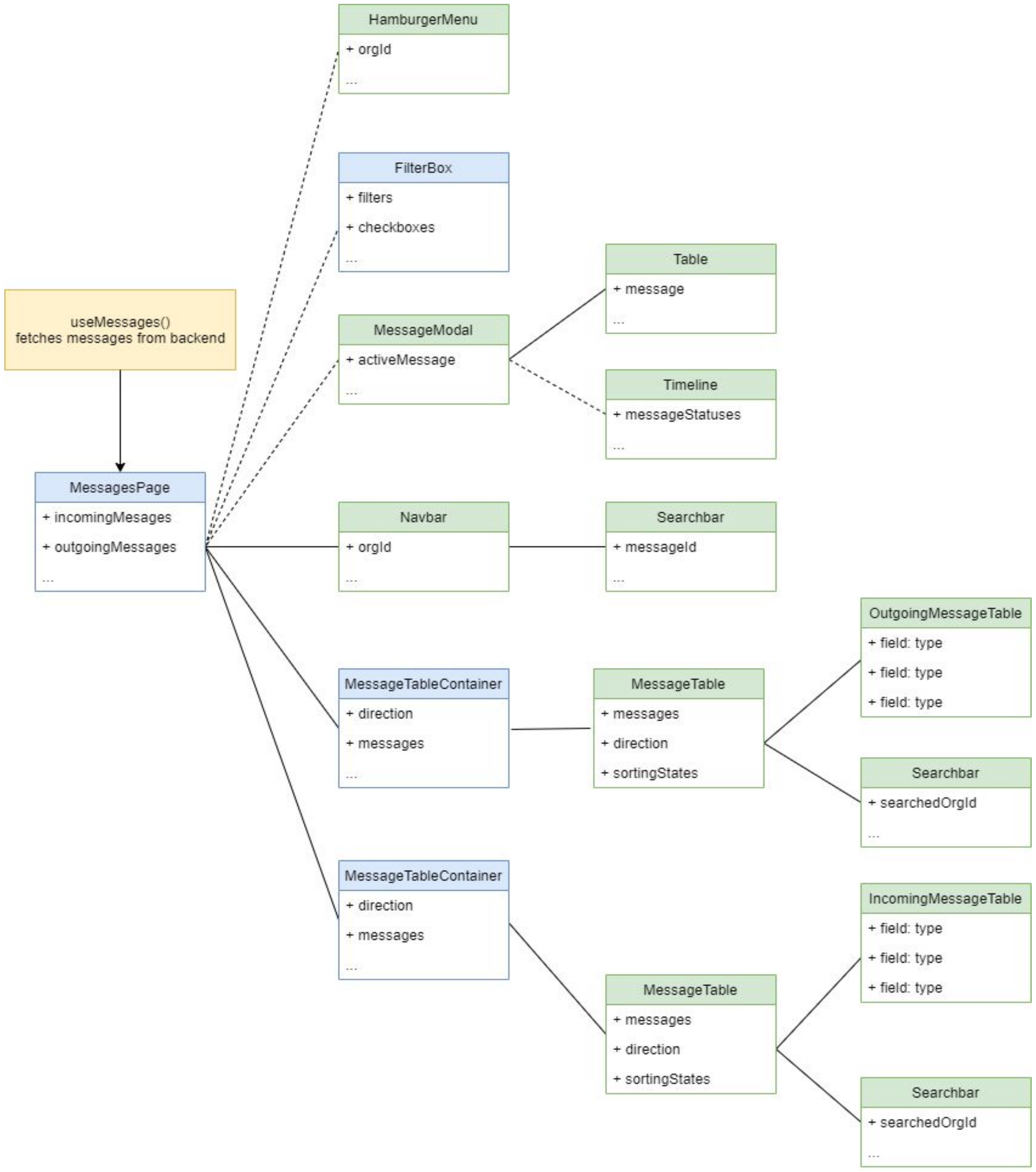


Figure 4, Frontend Architecture

4.3.2 Dataflow

To solve the problem of displaying statuses from two data sources, the team needed to figure out how the data should flow from the APIs to the backend. A message is present in both APIs (Elements API and Difi API) and matches through a 'conversation-ID'. Confusingly, a conversation-ID does not represent a conversation, but a message. A message represents a letter being sent from a sender to one or more receivers, on a "one time" basis. While conversations on the other hand, are the thread of messages between the sender and the receiver. A conversation can "contain" one or more messages.

To match messages between these two sources, it is not as simple as just fetching the n newest messages from both APIs, because they were never guaranteed to match. The team discussed several ways of doing this, from fetching a batch of messages from each API, to fetching a batch from one of the API and consequently individually fetch the matching counterpart in the other API. The latter method would result in very many HTTP requests, and the first method gave insufficient results. The team therefore landed on a hybrid method, where a batch from each API was fetched and attempted matched. Those messages that did not match, would then be individually fetched from the other API, and then matched. This is visualized in the model below. It is also worth mentioning that outgoing messages are based on the Elements API and incoming messages are based on the Difi API. This means that if an outgoing message is found in the Difi API, but not in the Elements API, the message will not be shown in Enode's outgoing message list. In incoming messages, it is opposite. This was a requirement from the Product Owner.

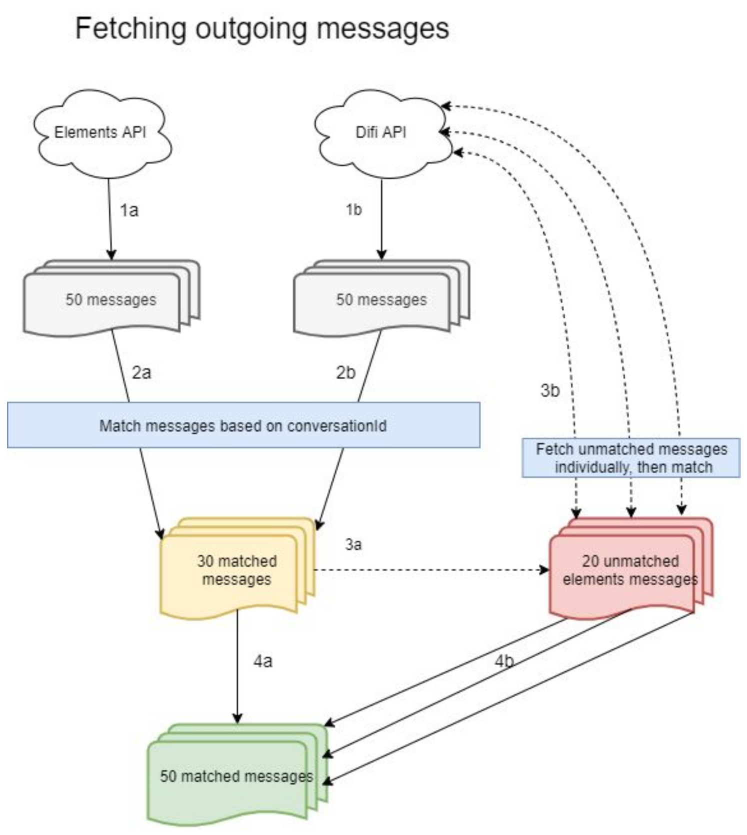

Fetching outgoing messages
Elements API
Difi API
1a
1b
50 messages
50 messages
2a
2b
3b
Match messages based on conversationId
Fetch unmatched messages individually, then match
30 matched messages
3a
20 unmatched elements messages
4a
4b
50 matched messages

Figure 5, Fetching outgoing messages

4.4 Risk Analysis

A risk analysis is the process of assessing the likelihood of an adverse event occurring (Hayes, 2019). A project risk is an uncertain event that could affect "the magic triangle", meaning it could influence scope, schedule and/or cost (Hofmann, 2020 c. p. 10). Risk analysis should be conducted early on in a project. The risk assessment is an ongoing process, and should be revised continuously (Hofmann, 2020 c, p. 30). The risk assessment and issue log can be found on pages 41 and 43 in the appendix, respectively.

4.4.1 Risk Register

During the pre-sprint, the team assessed potential risks and risk sources to the project. New risks have been added to the risk register and other risks have been reevaluated as time has passed. After having identified potential risks, a qualitative risk analysis was performed. This is the process of further analyzing and prioritizing individual risks by assessing their likelihood of occurring and their potential impact.(Hofmann, 2020 c, p. 19). Immediately after, a quantitative risk analysis was performed, where the team assessed the combined effect of identified risks (Hofmann, 2020 c, p. 10). Each risk's probability and impact were evaluated with a number from one to five, where one represents the lowest score and five the highest. By multiplying the values, the risk score was calculated. A risk response would then have to be decided. Performing the risk analysis proved to be useful as the team was made aware of potential risks and threats to the project. Being aware of this meant the members could act proactively to avoid these risks occurring or minimizing their impact on the project.

4.4.2 Issue Log

The issue log was the collection of the project's incidents. It was updated when risks in the risk assessment occurred. The exception to this was when issues arose from unidentified risks. In those cases, both the risk and issue were added to the issue log. From the issue log one can tell which risk it concerns, which date the issue occurred and when it was closed. The issue includes a risk score and a comment detailing what happened and what was done to handle the issue. The log was a useful tool when encountering previously assessed issues, as the team could respond quickly and appropriately.

4.5 Development

The development process is an important topic to discuss as this describes how the product was developed. This chapter goes into more detail about this process. It will describe topics regarding work routines, distribution of work, and implementation of features in the development.

4.5.1 Work Routines

Sprint planning

The overall work routine during the development was quite consistent. Throughout the project the whole team worked from 09:00 until 16:00 every weekday. After the Coronavirus outbreak the routines changed slightly, as the team started working from home. Scrum was utilized during the whole project and the team followed a two-week sprint layout. The first Monday of each sprint was spent on planning the sprint. During planning, the team decided which features would be worked on. The team broke the features down into tasks and scheduled them. Then tasks were assigned to the members which suited them best. Time estimation was done in plenary to let team members voice their opinions. Finally, each member wrote a description of what their task would solve and 'definition of done' criteria, which gave every team member clear and accomplishable tasks that they would focus on during the sprints. These criteria usually came from user stories, but they also contained technical criteria such as which files should be created or what programming package to use.

Implementation
The tasks would be assigned to the relevant person in Azure DevOps, and the 'definition of done' criteria was used to ensure that the intended task was completed. In addition to meeting the criteria, the team members would have to create unit tests for the code developed. Git was used as version control. When a feature/task was worked on, a branch would be created for said task. When a team member completed a task, they would first confirm that the acceptance criteria was met, then save their work to the relevant branch, and finally merge with the dev branch. This process is described in Appendix, Git Guide, p. 49. After this process was completed, a merge request would be created via DevOps and team members who were relevant to the task were marked as reviewer(s). Features were never implemented into the dev-branch before at least one other team member had carried out a review of the code and approved it (Appendix, Git guide, p. 49). These were strict routines that were carried out every time new code was to be implemented into the system. These routines incorporated quality assurance into the team's workflow and routines, and as well increased the code quality.

Sprint conclusion
When approaching the conclusion of the current sprint, the team performed a sprint retrospective. This provided the team with an opportunity for reflection and a look at what could be improved for the next sprint. Lastly, the dev-branch would be merged with the master-branch after ensuring that all unit tests were passing and that the application's functions were working as intended. Early next sprint (usually on the first Monday) the team would perform a sprint review with the product owner. This was in order to ensure that the product was developed as intended, according to schedule.

4.5.2 Work Distribution
Early on the team decided that all members would contribute to all stages of development. This ensured that everyone got a similar learning outcome, including both technical and practical aspects.

One team member functioned as the Scrum master, while another was responsible for taking notes during meetings. All members worked on the reports and the development tasks. Some members focused more on backend- or frontend tasks, which resulted in them gaining a deeper knowledge of their area and enabling them to guide other members. Other members would do both backend- and frontend tasks depending on what tasks needed to be conducted. This was valuable for the team, as workforce resources could be moved easily between frontend and backend, depending on which work remained. This enabled a high progression throughout the project.

4.6 Quality Assurance

From the start of the project, the team agreed that the quality of the product was to be kept to a high standard. This was a goal for the team, as the team wanted to fulfill the product owner's requirements and develop the system that had been envisioned by Sikri. The product also needed to implement high quality as it was potentially going to be deployed to the users of Elements. The team had some ideas of how to ensure a high-quality product. Code standards, folder structures, acceptance tests and unit tests were items of value, which the team agreed were to be performed from the start. These would help the team achieve quality by implementing a uniform code standard, and application structure, testing that the application performed as expected, and that the features fulfilled their requirement definitions. Ensuring the use of these could also greatly aid in the transfer of the product to Sikri, making the transition easier.

Acceptance tests are often meant to reflect the use of a feature in the real world, and the result of that use (Omland, 2006, p. 3). The user stories and their respective acceptance tests were developed together with the product owner. These describe the actions the user performs and the results from performing them. The acceptance tests were evaluated by a binary; pass or failure (Agile Alliance, 2012). It was important for the team to align with the product owner during the planning phase of the project, as the tasks which the system is going to fulfill is a determining factor of the quality in the product. A document discussing quality can be found in the Appendix, together with the Git guide. (Appendix, Quality of end product, p. 45 , Git guide, p. 49).

4.6.1 Unit Testing

Unit tests are tests which are performed on each unit of the program. A unit can be described as a class, method or module, often performing a single task, with few inputs and outputs. (Nilsen, 2020, p. 9). The program is built in this way, in order to create a high level of cohesion and loose coupling (Nilsen, 2020, p. 9). Unit tests are written to test these simple units, by observing the results, and comparing them against the expected result. Unit tests were run on each component in the backend to ensure the method performed as expected, and that the method could be used in the product. Tests were required to pass before a feature could be completed and merged into the dev-branch. This was a general rule for all team members, to emphasize the quality of the end product. In the frontend, unit tests were performed on data fetching functions and other utility functions. User interface components were not tested until quite late in the process, because these would change too frequently to be prioritized.

4.6.2 Acceptance Tests

During the run of the project, the team wrote user acceptance tests for the features, to ensure that the features would be implemented as intended. The acceptance tests described a set of user actions, and the correct results to those actions. Each one of the user stories had at least one corresponding acceptance test. When carrying out acceptance tests, a team member would perform the described actions, and the result was evaluated, see (Factory Acceptance Test 1, 2, p. 55-58). Two meetings were held, as not all tests passed during the first run. The product owner was present at these meetings, to oversee, and grade the tests with either a pass or not passed. This aligned with the agile framework, Scrum, as the team had the opportunity to go back on these features and complete them. Factory Acceptance Test 1 and 2 can be found in the Appendix (Factory Acceptance Test 1, 2, p. 55-58).

5. Reflection

Throughout the project we have faced a number of challenges. In this chapter we will reflect on different topics regarding how the project has progressed and what we have learned and experienced through the course of the semester.

5.1 Changes and Challenges

With little to no experience working with Sikri's systems, APIs, and the program Elements, we had to plan and analyze the task ahead based on input from Sikri employees. This made planning more challenging. We managed this by creating user stories based on information provided by Sikri employees together with the product owner. At this stage, the development phase could begin by breaking down every user story into backlog items, in order for the us to implement them.

One of the larger challenges in the project was the issue of not getting access to the relevant resources that the project was to be based upon. Getting access to credentials and the API endpoints from Sikri took a long time, ultimately preventing us from making progress. As a result of not getting access to the resources, we did not know which data could be used in the system. In turn, this also resulted in us not being able to analyze and structure the dataflow, as well as which data should be presented in the frontend.

After receiving access to external systems, we discovered that there had been a misconception regarding what information could be retrieved from the APIs. We had previously thought that it was possible to gather more information regarding the location of a specific message. As a result, one of the planned features was a timeline containing the location and status of a message, see (Appendix, Picture 3, Prototype selected message, p. 61). We later realized that this feature could not be implemented because of a lack of information. The driver for this scope-change was internal, meaning that the change had to be made because of an internal oversight made by the team during the planning phase (Hausvik, 2020, p.7-9).

Another central challenge the team had to overcome was the overall lack of experience. We were all relatively new to systems development, as well as unfamiliar with many of the tools utilized. We managed to overcome these challenges mainly by being patient and recognizing that things were going to take more time, as we had to learn the tools and get a feel for the workflow. Helping factors with these issues were Azure DevOps that assisted in planning and estimating time, in addition to the team being able to ask anyone at Sikri for advice.

During the development phase, the coronavirus pandemic broke out. This resulted in the team and the rest of the Sikri employees working from home. In the beginning we saw this as a challenge, and expected productivity to be affected. Working from home did not really have an impact on individual productivity. However, when the team was situated at the Sikri offices, there was a lower bar for asking questions, or having a chat with any of the professionals. This change happened because of the COVID-19 crisis, and was recognized as an external driver for change (Hausvik, 2020, p.7-9).

5.2 Learning Outcomes

Considering that most of the team members had little to no experience with working in a software development project, the amount of experience and knowledge we gained from this project was substantial. There were many aspects of the project that helped us grow, both as developers and as coworkers. This chapter will mainly focus on both technical skills and cooperation skills, as well as a self-evaluation for each team member.

5.2.1 Technical Skills

Throughout the project, the team members have gained valuable experience on the technical side of systems development. The members have enhanced their technical skills both in frontend and backend development. Most of the team members had no previous experience with the technologies utilized. This meant that the team members had to learn backend technologies like the Docker, ASP.NET Core framework, C#, Moq and Xunit for testing, and API related frameworks like ODATA and REST. Frontend technologies included React, TypeScript, HTML/CSS and Jest for testing. Besides the back- and frontend technologies, most team members were new to the tools utilized: Visual Studio, Visual Studio Code, Azure DevOps and Git.

The team has also improved their architectural understanding of web applications, as the entire application was designed from the ground up, by us. The team has made important fundamental decisions regarding dataflow in the application. This has taught the team how data flows in a web application and how data is accessed between frontend and backend. The team has learned how APIs work, both by requesting data from external APIs, in addition to developing one for the product solution. These are all valuable skills and experiences, which the members will make good use of in the future.

5.2.2 Cooperation Skills

The team has experienced firsthand how important communication and cooperation is in a systems development project. Thankfully, the members were highly cooperative throughout the project, both towards each other, and the stakeholders. Good cooperation made for more productive discussions, as members brought up their own opinions and views. Cooperation and communication have also been important in the development of the product. When members were working independently on implementing features, it was important to properly establish what was going to be implemented, and how the code was going to behave and perform. Through discussions and working closely in the development and processes, the team gradually became better at communicating and cooperating effectively. These are valuable experiences and skills that members can apply to any future work and projects.

5.2.3 Self Evaluation

This report was developed equally by all members of Brukerfeil. This sub-chapter contains the personal experiences from our bachelor project. During the project's planning and development phase, our roles differed only slightly as every member wrote code and developed equally. The only dedicated, non-development roles were inhabited by Espen, who wrote the minutes during meetings, and Marius who acted as the Scrum master.

Gorm-Erik Aarsheim

My contributions in this project have been of both a technical and non-technical sort. On the technical side, I have contributed by implementing functionality. I have also written unit-tests to ensure that the implemented functionality is performing as expected. I believe that I established a basis for how tests should be performed in the backend, making it easier for the rest of the team to follow these conventions. I have also connected the application to external services for storing configurations which was a tedious and difficult, but important process. In addition, I have had the main responsibility of setting up a deployment pipeline, which builds the application into lightweight executable packages called Docker containers, which can be deployed to a website. On the non-technical side, I have contributed to written assignments, reports and logs throughout the semester, always trying to maintain progress. I have also actively attended discussions regarding both the product development, planning and project processes, in order for the team to find good approaches and solutions. I believe that my contributions in discussions have been important for bringing up different views and arguments for finding solutions to the team's best interests.

Vegard Alvsaker

My role during this project has been technical. As I had experience with both the frontend and backend technologies we used, I was often in positions where I taught and gave advice to other team members. I contributed to the application structure, both backend and frontend, where I focused on making the application scalable and testable. I heavily advocated for using TypeScript in the frontend, which highlights code errors before the application is run. This created an extra quality layer. Most of my code was written in the frontend and was mainly in the data fetching domain or in bigger problems, such as how data should flow through the components. In addition to acquiring a great matter of technical skills, I have learnt how to pass knowledge to others in a way which they can understand, and also how to cope with rapid and frequent changes in an agile project. I have experienced the reality in how difficult it is to execute everything in a plan, which highlights the importance of an agile method.

Ingve Fosse
I was active during discussions, trying my best to contribute towards a satisfying outcome of the project. I think it is important that each team member voices their opinion even though it might reveal a disagreement about a certain aspect of the project. Most of the drawings of the application were my contributions, including wireframes, hand drawn prototypes etc. The digital prototypes created in Adobe XD were created by me as a means to test different solutions. The prototypes were also used to decide on the aesthetics of the application. I had never used a UX/UI designing tool before, however, I was familiar with the Adobe-Suite. This allowed me to quickly pick up Adobe XD and start prototyping. We presented some solutions and received input on them from the product owner. I think this was a good solution. My code contributions mainly included large parts of the backend code, including data models for the messages, methods for retrieving the messages from the API's and a service to merge them. Lastly, I spent a lot of time unit-testing the backend code by utilizing Xuint and Moq. The coding aspect of the project was by far the most demanding for me. I had no previous experience with unit-testing, ASP.NET, C#, React, TypeScript or any of the tools we utilized except Git. The learning curve was steep, but I received support from the team when I required it. I picked it up reasonably fast and I think I was useful to the team.

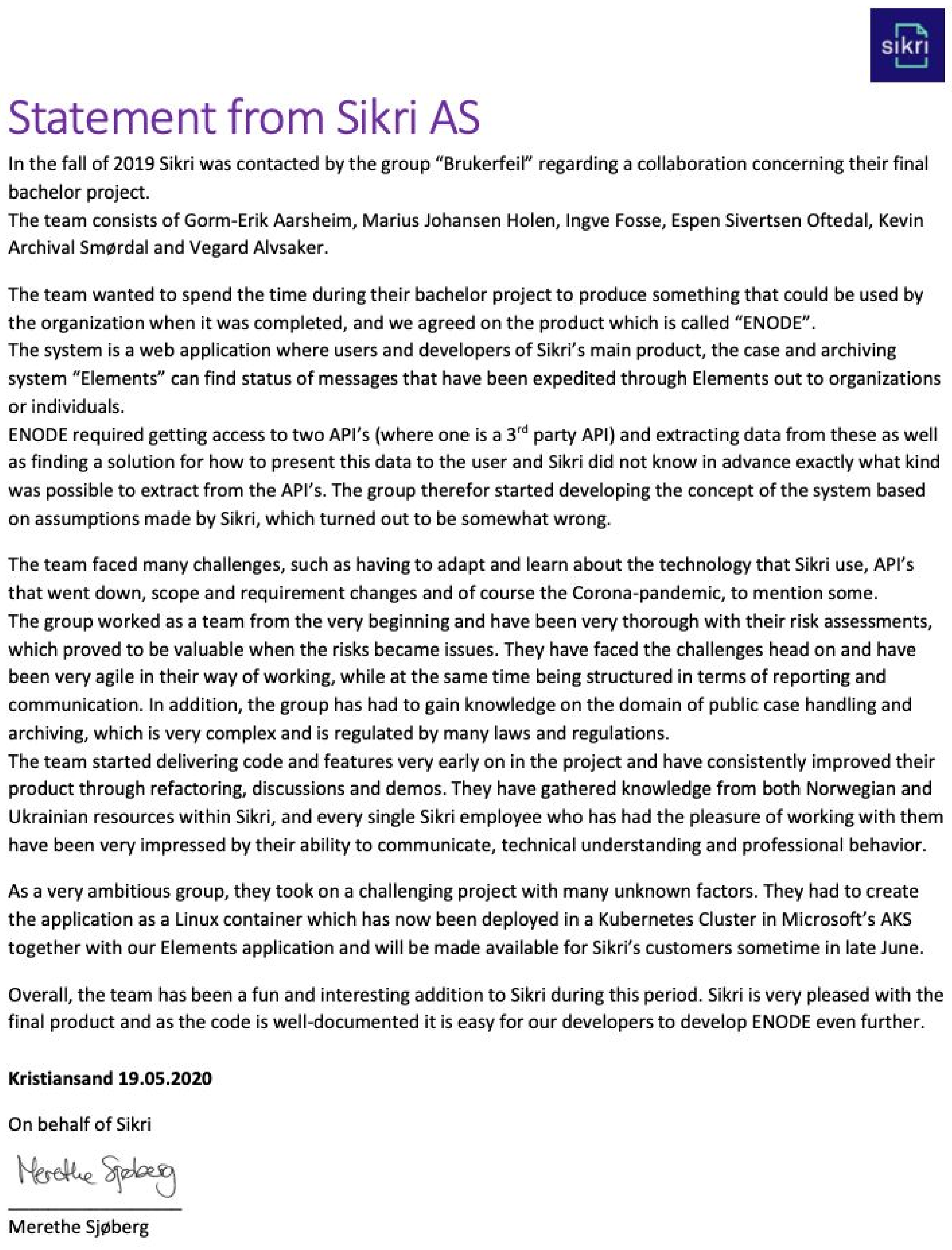
Marius Johansen Holen
My goal for this project was to learn a lot and to gain valuable experience from contributing to a system development project. During the project I have acted as the Scrum Master. The role has meant scheduling meetings, leading sprint planning meetings and being the team's spokesman during meetings with stakeholders. I have tried to eliminate obstacles to the project, making sure the team can maintain focus and progress towards the goal. Besides being the project Scrum master, I have also been working along with the team with both technical and non-technical tasks. At the start of the project I worked on the backend, but gradually changed focus more towards the frontend. By doing this, I have gained a good understanding of different aspects of developing a web application. I have learned to use both C# and React, giving me a great learning outcome. For the planning and assignments, I have contributed with discussion to how it should be conducted and to the writing of reports and assignments.

Espen Sivertsen Oftedal
My contributions largely revolve around documenting, as well as backend and later frontend development. At the start, I felt motivated and ready to start working and learning something new. I was the main person documenting the day-to-day discussions. There was a lot of documentation performed, especially during the earlier stages of the project. I started of working on the backend, using C#. At the same time, I was documenting meetings, daily routines and decisions. After a while, motivation fell to some degree as I was performing some of the same tasks each day, and this became repetitive. Therefore, it was for my own benefit when I later in the project went over to frontend development. Here I had something new to learn and work on in the form of React with TypeScript. Working on different aspects of the project has been of great benefit to myself, as I have learned more than if I had not switched over. Some time was spent on "getting into" these new subjects. It would probably be more time efficient for the overall project if members of the team worked on the same areas of the system throughout the entirety of the project. However, I am happy this did not happen as my learning outcome was increased.

Kevin Archival Smørdal
For this project I had a goal to learn something new and experience something challenging from a real systems development project. I worked the entire project on the frontend part of the project where all the data was fetched from the backend and presented accordingly. Starting off early in the project, we used pair-programming to create a basis for our project with folder- and file structures. Throughout the project, I contributed by creating new components such as the front page, organization list, message modal and the hamburger menu/dropdown to handle the data from the APIs. I also contributed by refactoring messages and the status style for each status that had to be done due to the scope changing consecutively. A lot of time was spent learning these new languages and tools, though I ended up mostly doing the styling of the application. Creating most of the styling with HTML/CSS according to the prototype, I also contributed to other team members' work on styling.

5.3 Statement from Client

This is the statement the group received from the supervisor at Sikri, Merethe Sjøberg who served as the product owner as well as Brukerfeil's main contact person from Sikri.



Statement from Sikri AS

In the fall of 2019 Sikri was contacted by the group "Brukerfeil" regarding a collaboration concerning their final bachelor project.
The team consists of Gorm-Erik Aarsheim, Marius Johansen Holen, Ingve Fosse, Espen Sivertsen Oftedal, Kevin Archival Smørdal and Vegard Alvsaker.

The team wanted to spend the time during their bachelor project to produce something that could be used by the organization when it was completed, and we agreed on the product which is called "ENODE".
The system is a web application where users and developers of Sikri's main product, the case and archiving system "Elements" can find status of messages that have been expedited through Elements out to organizations or individuals.
ENODE required getting access to two API's (where one is a 3rd party API) and extracting data from these as well as finding a solution for how to present this data to the user and Sikri did not know in advance exactly what kind was possible to extract from the API's. The group therefor started developing the concept of the system based on assumptions made by Sikri, which turned out to be somewhat wrong.

The team faced many challenges, such as having to adapt and learn about the technology that Sikri use, API's that went down, scope and requirement changes and of course the Corona-pandemic, to mention some.
The group worked as a team from the very beginning and have been very thorough with their risk assessments, which proved to be valuable when the risks became issues. They have faced the challenges head on and have been very agile in their way of working, while at the same time being structured in terms of reporting and communication. In addition, the group has had to gain knowledge on the domain of public case handling and archiving, which is very complex and is regulated by many laws and regulations.
The team started delivering code and features very early on in the project and have consistently improved their product through refactoring, discussions and demos. They have gathered knowledge from both Norwegian and Ukrainian resources within Sikri, and every single Sikri employee who has had the pleasure of working with them have been very impressed by their ability to communicate, technical understanding and professional behavior.

As a very ambitious group, they took on a challenging project with many unknown factors. They had to create the application as a Linux container which has now been deployed in a Kubernetes Cluster in Microsoft's AKS together with our Elements application and will be made available for Sikri's customers sometime in late June.

Overall, the team has been a fun and interesting addition to Sikri during this period. Sikri is very pleased with the final product and as the code is well-documented it is easy for our developers to develop ENODE even further.

Kristiansand 19.05.2020

On behalf of Sikri

Merethe Sjøberg

Merethe Sjøberg

Figure 6, statement from Sikri AS

6. Conclusion

The team conducted their bachelor project at Sikri AS, a company producing a case processing and archive system called Elements. The team was tasked with developing a system for displaying the status of messages expedited in and out of their system. Throughout the project the team has used Scrum as a project management framework. Together with a Sikri employee acting as product owner, a set of requirements and project specification was developed, which defined the project scope.

The project end product is the fully operational web application Enode. A system which allows its users to select their organization and view the statuses of messages sent to and from their organization. It collects information from two sources, and combines them in order to accurately represent the status of the messages.

During the development of the product the team faced several challenges. The most prominent challenges the team experienced was the lack of access to resources required for the application to function. Also, the team had little to no documentation and knowledge on said resources. The team handled this by staging work independently of these resources. Even through our best efforts to mitigate the impact of these challenges, it would come to delay the progression of the development, impacting both the progression and project scope. Finally, the lack of experience regarding the technical aspects surrounding the project proved to be a challenge but was overcome by the team's persistence.

Through this project the team has gained valuable experience by working on a realistic systems development project. The team has experienced the different phases and processes that go into a systems development project. This serves as valuable experience as the members will know how to work in an agile environment and know how to respond to changes occurring both in the scope as well as outside of the project. The value of proper planning is an important experience that the team has made, as this sets the foundation for the work to be conducted. The team has also gained more technical experience by working with modern technologies through the majority of a semester.

Together, the team members have developed a fully functioning web application. The team believes the system is of high quality as it successfully implements the requirements defined by the product owner. The project has had a great educational impact on the team, gaining both technical and non-technical experience from a realistic systems development project. These experiences will be valuable for future projects.

References

Agile Alliance. (2012). Acceptance Testing.
Fetched from https://www.agilealliance.org/glossary/acceptance

Daily Life Science. (2013, 13. October). What is Quality? - Quality Meaning in real terms.
Fetched from https://www.youtube.com/watch?v=ZpFqnefTGA8

DeMuro, J. (2019, 18. December). What is Container Technology?
Fetched from https://www.techradar.com/news/what-is-container-technology

Docker.com. (2018, 8. August). What is a Container?
Fetched from https://www.docker.com/resources/what-container

Drumond, C. (2018, 13. December). What is Scrum?
Fetched from https://www.atlassian.com/agile/scrum

Hausvik, G. I. (2020, 4. April) Lecture 9: Change Management.
Presented at IS-305 Managing digitization projects, University of Agder.

Hayes, A. (2019, 7. October) Risk Analysis.
Fetched from https://www.investopedia.com/terms/r/risk-analysis.asp

Hofmann, S. (2020 a, 15. January). Lecture 2: Project scope, time, and budget management.
Presented at IS-305 Managing digitization projects, University of Agder.

Hofmann, S. (2020 b, 5. February). Lecture 4:Agile approaches in (IT) project management.
Presented at IS-305 Managing digitization projects, University of Agder.

Hofmann, S. (2020 c, March). Lecture 6: Risk management.
Presented at IS-305 Managing digitization projects, University of Agder.

Microsoft. (2019 a, 5. February). What is ASP.NET?
Fetched from https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet-core

Microsoft. (2019 b, 9. September). DevOps solutions.
Fetched from https://azure.microsoft.com/en-us/solutions/devops/

Nilsen, H. (2020, March). Overview of SW quality issues.
Presented at IS-304 Bachelor Thesis in Information Systems, University of Agder.

Omland, Egeland. H. (2006). Systems Analysis and Design (SAD).
Presented at IS-200 Bachelor Thesis in Information Systems, University of Agder.

ProductPlan. (2020 a). Minimum Viable Product (MVP).
Fetched from https://www.productplan.com/glossary/minimum-viable-product/

ProductPlan. (2020 b). MoSCoW Prioritization.
Fetched from https://www.productplan.com/glossary/moscow-prioritization/

Rehkopf, M. (2018 a, 28. February). Learn how to use burndown charts in Jira Software.
Fetched from https://www.atlassian.com/agile/tutorials/burndown-charts

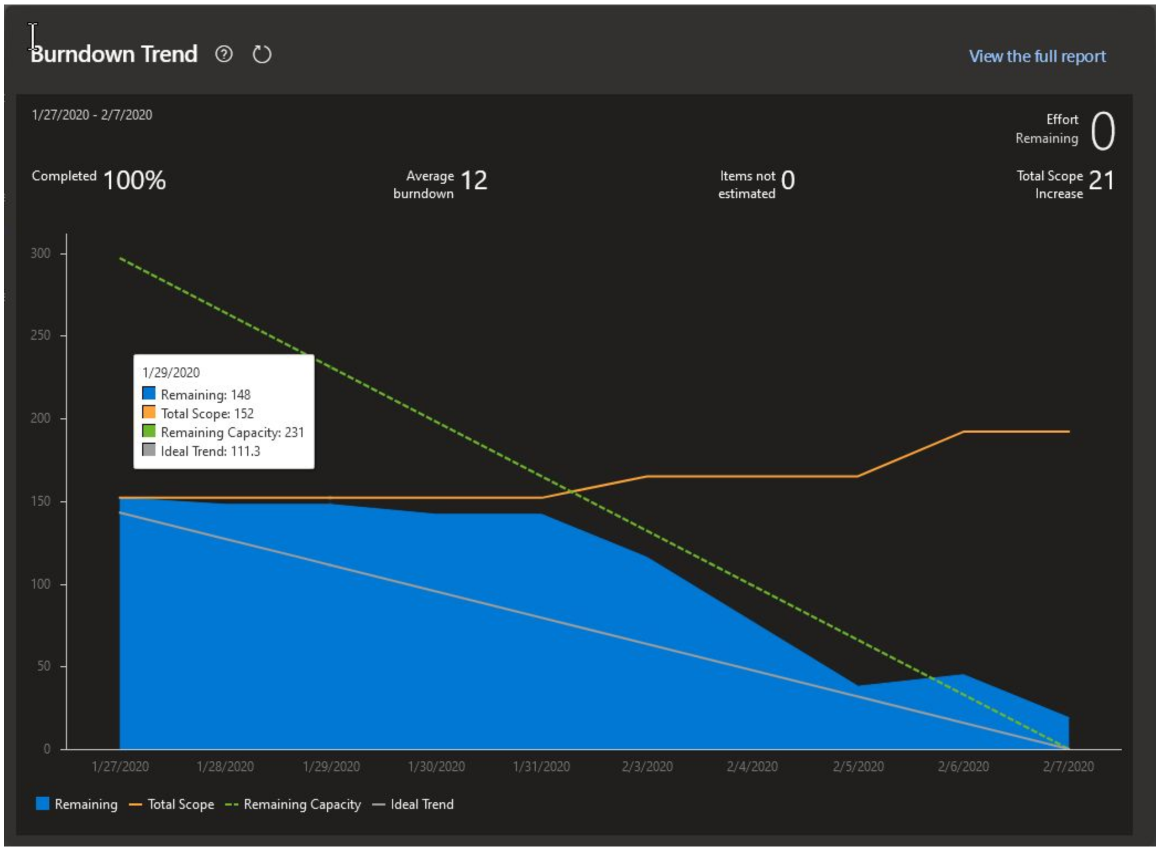
Rehkopf, M. (2018 b, 4. May). Kanban vs. Scrum.
Fetched from https://www.atlassian.com/agile/kanban/kanban-vs-scrum

Scrum.org. (2017, 28. March). What is a Scrum Master?
Fetched from https://www.scrum.org/resources/what-is-a-scrum-master

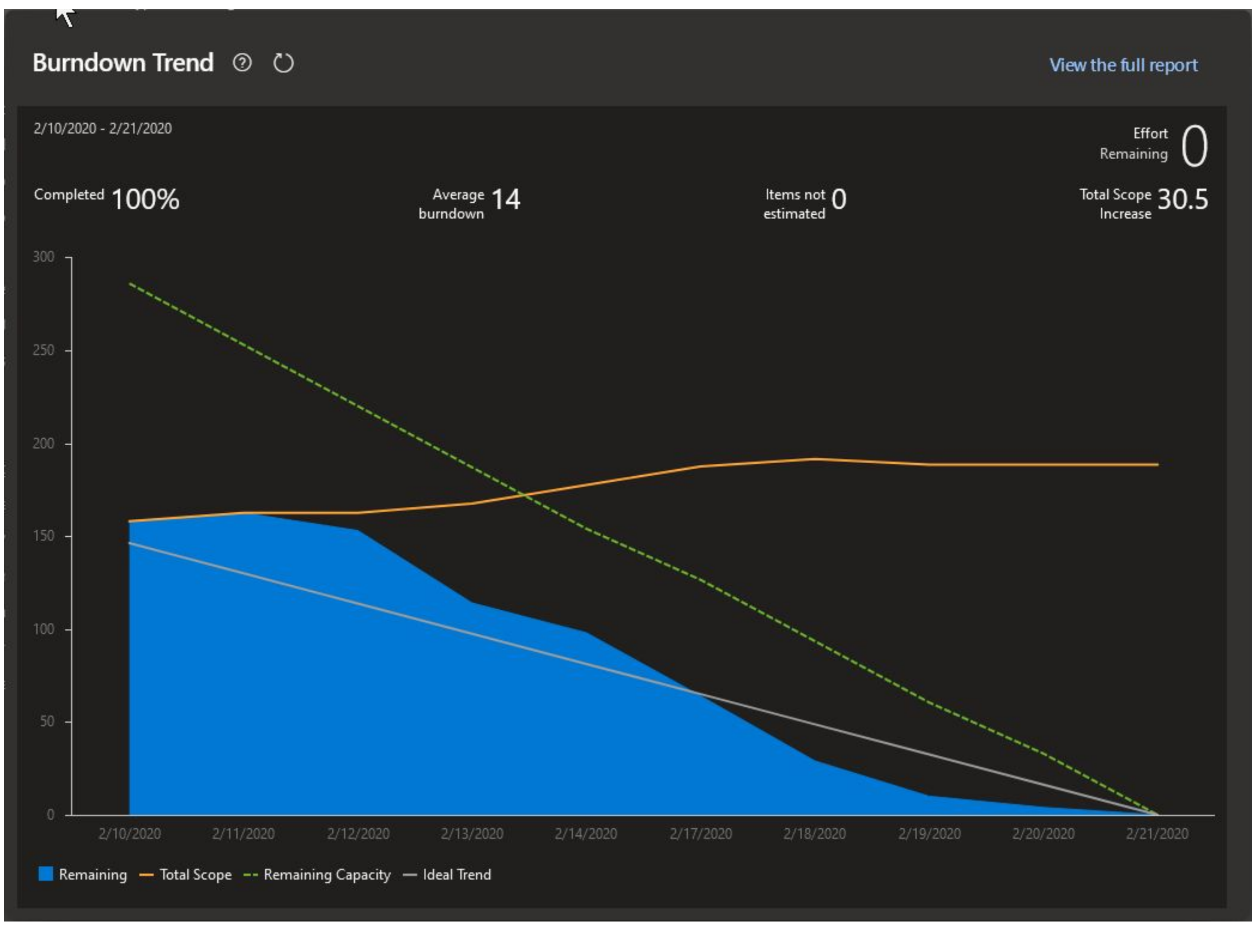
Vasilakopoulou, X. (2017, 15. November). User Interfaces IS-104, Revisionlecture.
Presented at IS-104 Bachelor Thesis in Information Systems, University of Agder.

Appendix

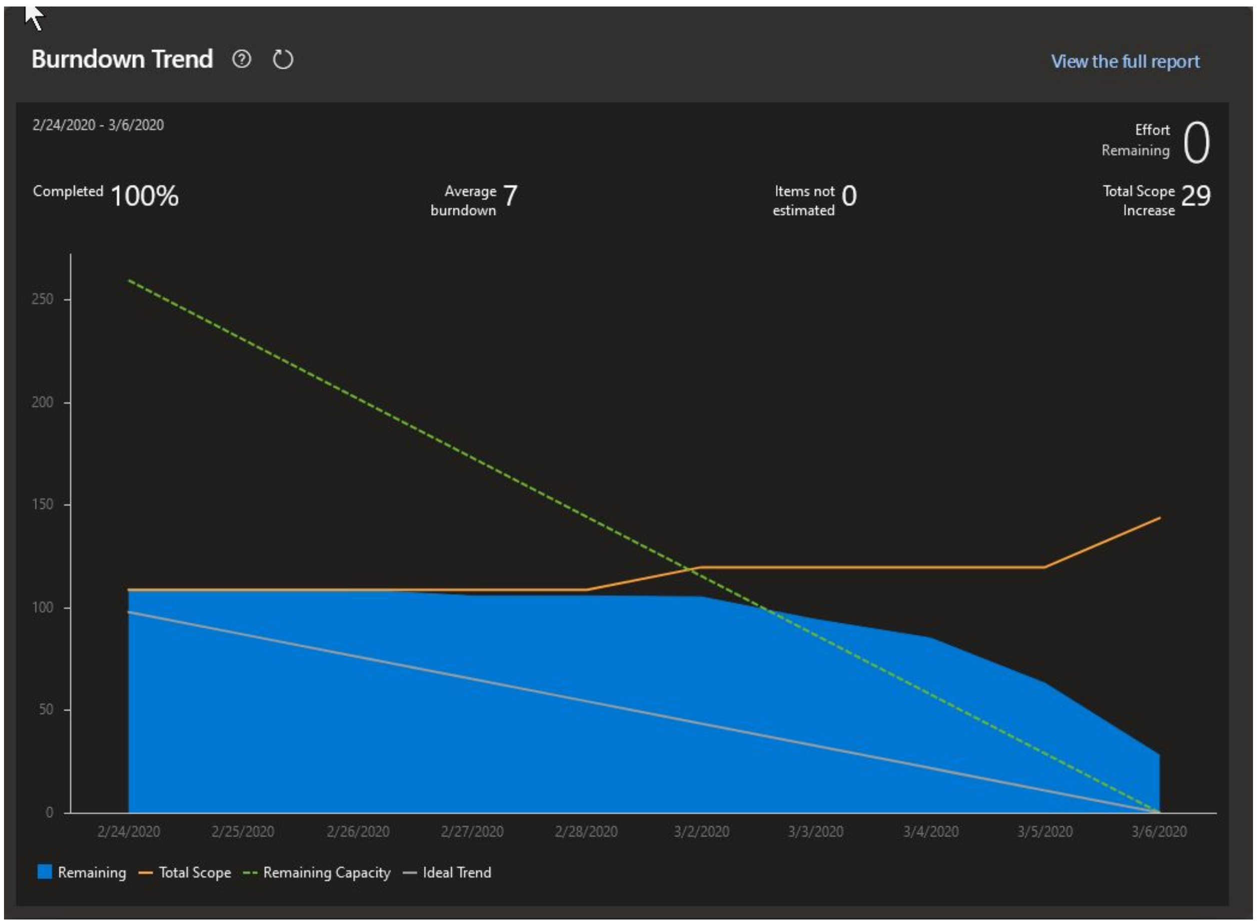
Picture 1, Sprint 1 Burndown Chart



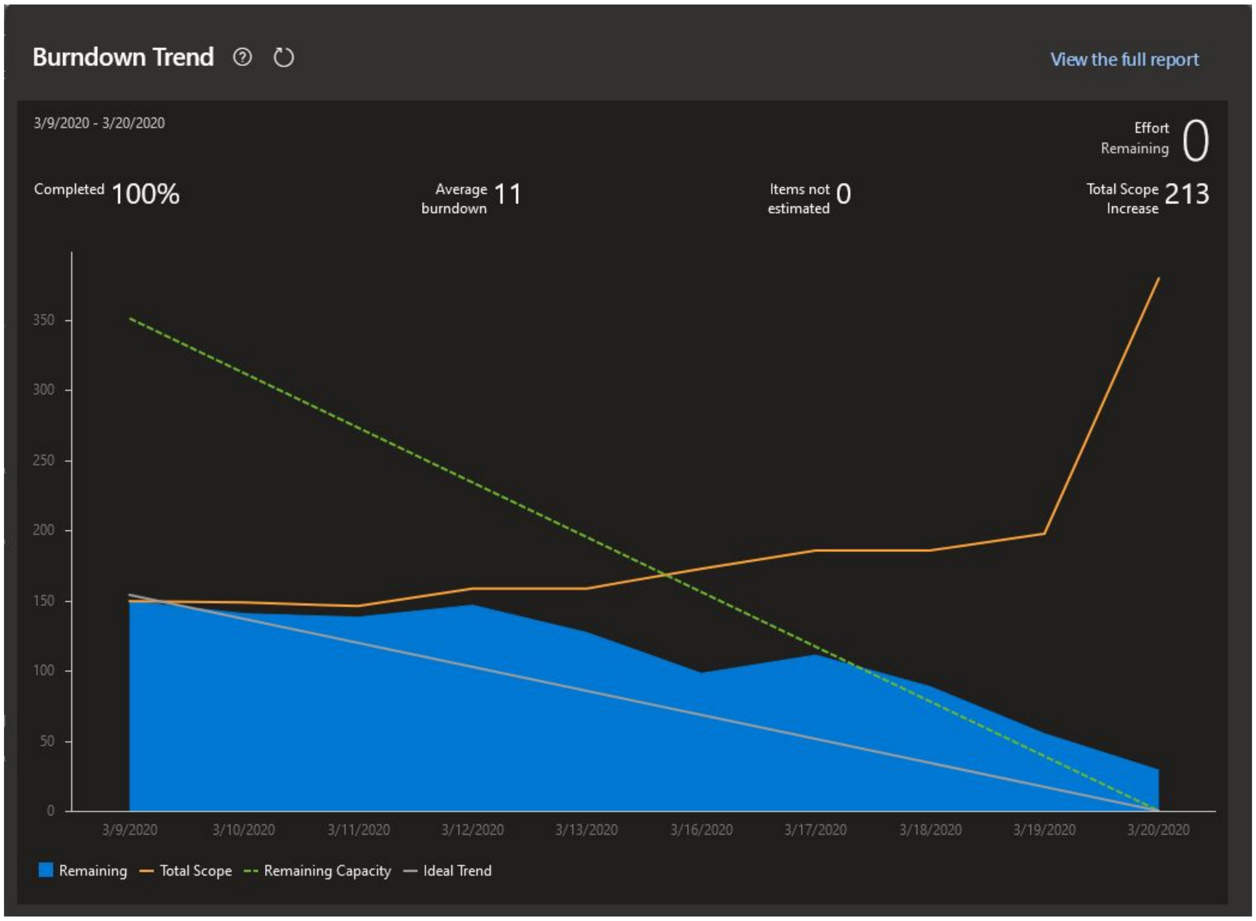
Picture 1.1, Sprint 2 Burndown Chart



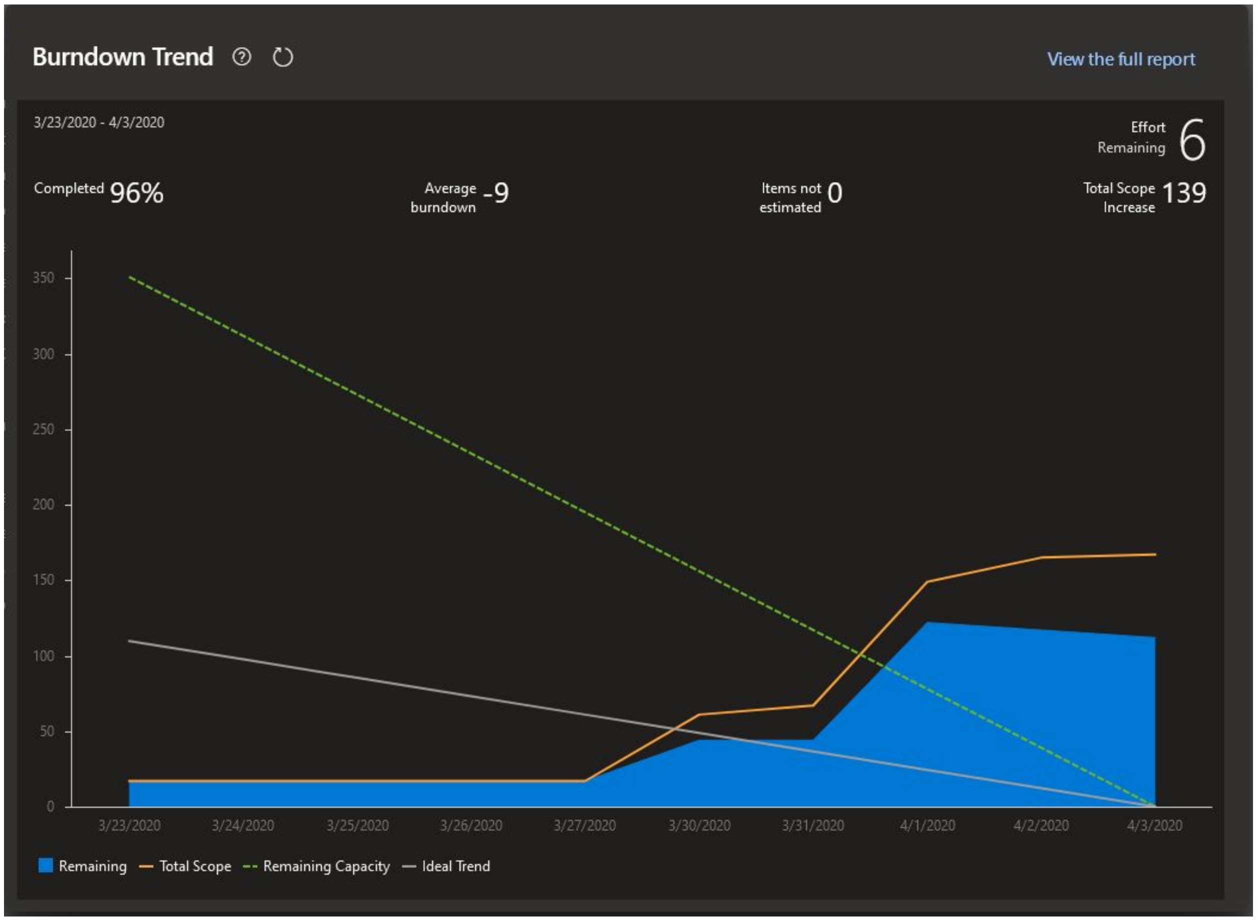
Picture 1.2, Sprint 3 Burndown Chart



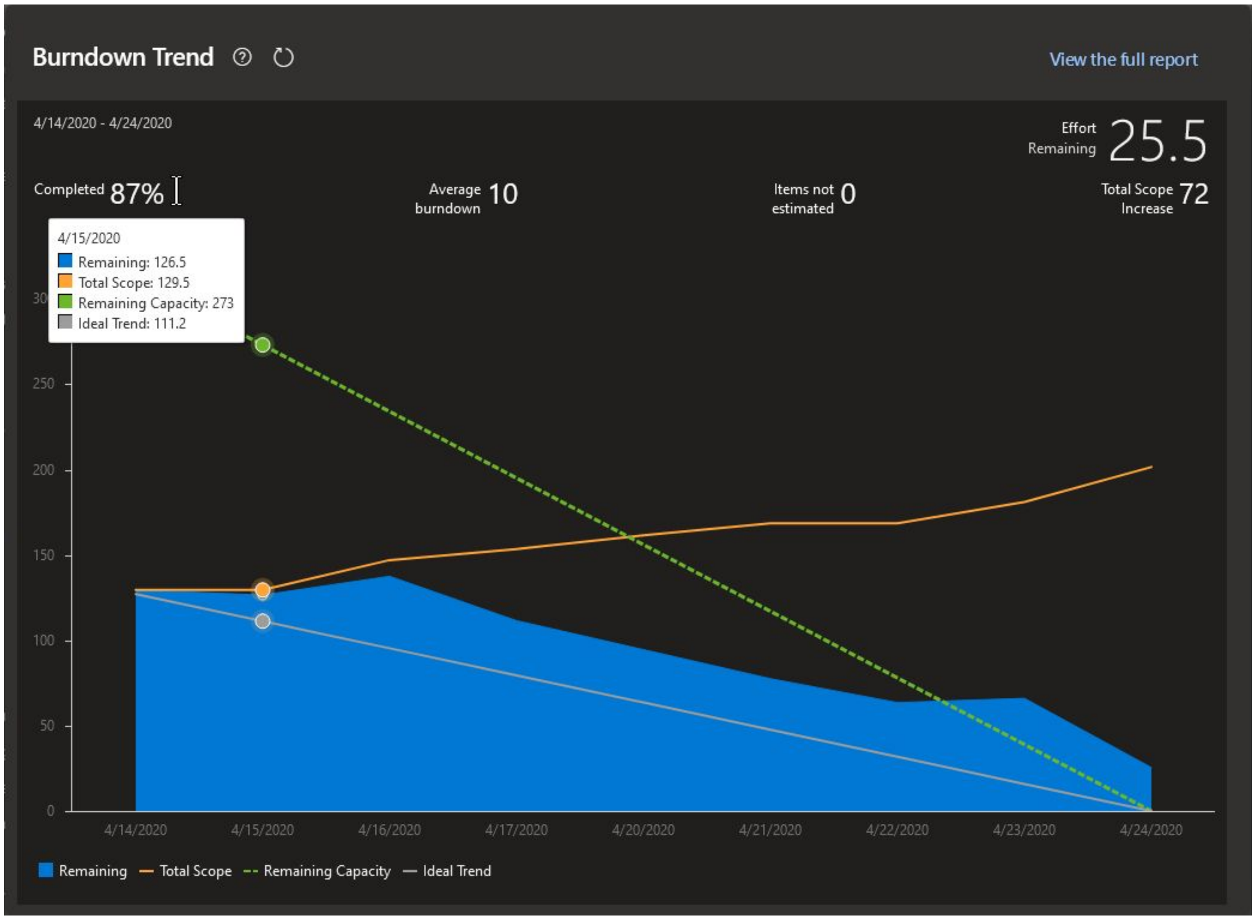
Picture 1.3, Sprint 4 Burndown Chart



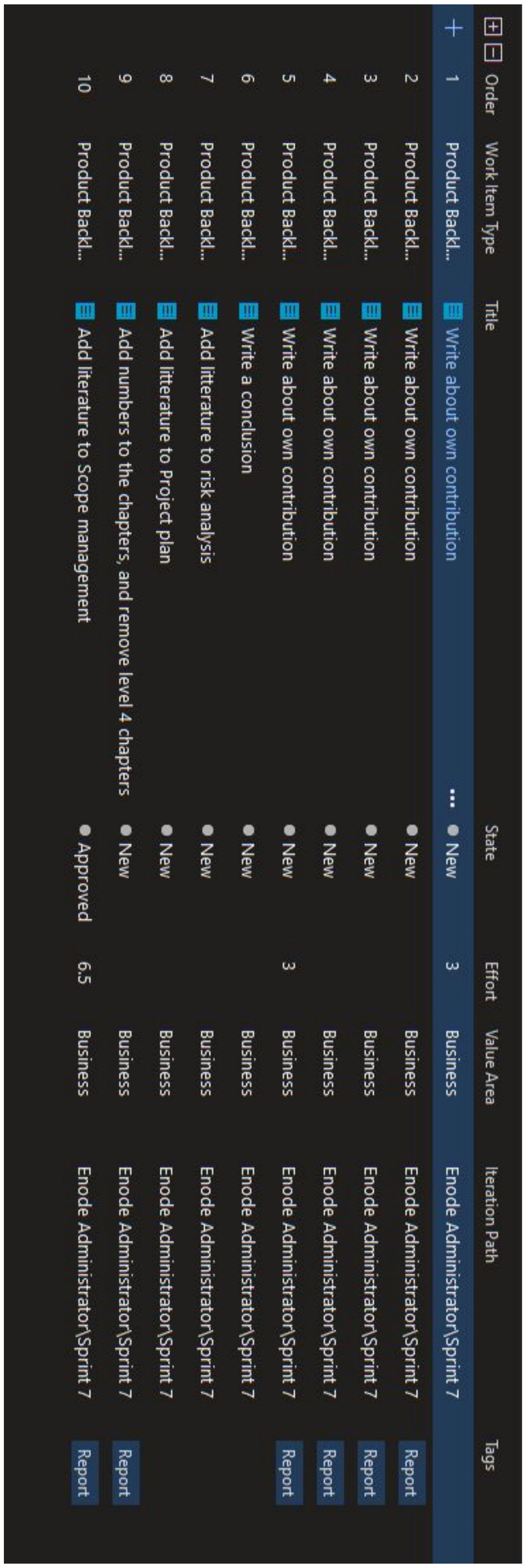
Picture 1.4, Sprint 5 Burndown Chart



Picture 1.5, Sprint 6 Burndown Chart



Picture 1.6, Azure DevOps Enode Administrator Backlog



Order	Work Item Type	Title	State	Effort	Value Area	Iteration Path	Tags
1	Product Backl...	Write about own contribution	● New	3	Business	Enode Administrator\Sprint 7	
2	Product Backl...	Write about own contribution	● New		Business	Enode Administrator\Sprint 7	Report
3	Product Backl...	Write about own contribution	● New		Business	Enode Administrator\Sprint 7	Report
4	Product Backl...	Write about own contribution	● New		Business	Enode Administrator\Sprint 7	Report
5	Product Backl...	Write about own contribution	● New	3	Business	Enode Administrator\Sprint 7	Report
6	Product Backl...	Write a conclusion	● New		Business	Enode Administrator\Sprint 7	
7	Product Backl...	Add litterature to risk analysis	● New		Business	Enode Administrator\Sprint 7	
8	Product Backl...	Add litterature to Project plan	● New		Business	Enode Administrator\Sprint 7	
9	Product Backl...	Add numbers to the chapters, and remove level 4 chapters	● New		Business	Enode Administrator\Sprint 7	Report
10	Product Backl...	Add literature to Scope management	● Approved	6.5	Business	Enode Administrator\Sprint 7	Report

Figure 1. Risk assessment

#	Risk	Probability	Impact	Risk score	Risk response	Comment
1	No ability to gather data because of server-issues	4	3	12	Transfer	Wait until the host has made the data available again. Contact the host, to make sure that they are aware of the issue.
2	Insufficient information from API(s)	3	4	12	Mitigate	Remove/rework feature. Should the scope of the project be affected, it will have to be reviewed and reworked with the product owner.
3	Lack of workspace after pandemic or other natural disaster	5	2	10	Mitigate	The members of the team will work from home, using digital tools for communication.
4	Getting stuck because of lack of experience	4	2	8	Mitigate	Acquire the required knowledge from team members, Sikri personnel or UiA staff.
5	Progress is too slow to reach goals	2	4	8	Avoid	Stick to the planned time frame. Adjust workflow accordingly. The schedule should ensure sufficient progress.
6	Unable to receive technical advice	2	4	8	Mitigate	Ask other people for help. This can be other personnel at Sikri or UiA. If none can help we should seek answers on the internet or temporarily work around the technical issue at hand.
7	A team member gets sick, or is elsewise unavailable.	3	2	6	Mitigate	The team members workload will have to be distributed among the rest of the team.
8	Project coordinator/product owner unavailable for a long period of time	2	3	6	Mitigate	The team will talk to other personnel at Sikri that might be able to inform us. If not, the team will have to make our own assumptions and base our decisions on these.
9	Network failure	3	2	6	Mitigate	Move physical location, mobile network. UiA team rooms can be utilized.
10	Project is cancelled due to external reasons	1	5	5	Accept	Inform UiA, get council from Geir/other UiA-personnel
11	A global pandemic breaks out	1	5	5	Mitigate	Mitigate risk by following advice from health organizations and other institutions.

12	Team member quits the team	1	4	4	Accept	Inform UiA and Sikri product owner. Consider changing the project scope.
13	Machine failure	3	1	3	Mitigate	Store all work online. Save often. Commit and push code at regular intervals.
14	License in Visual Studio expires	1	3	3	Mitigate	If Visual Studio is installed on a new computer, recover the product key from azure.com/education-overview.
15	Unable to receive support from UiA/ Geir	1	2	2	Mitigate	Reach out through mail, if there is no response, visit at UiA-office or contact at a lecture.

Figure 2. Issue log

#	Issue	Risk score	Open date	Close date	Comment
1	Network failure	6	27.01.2020	12.03.2020	The team had continuous problems with the wireless network at the Sikri office. The issue was resolved by waiting or restarting the computer. Several team members were affected.
2	Network failure	6	07.02.2020	10.02.2020	The network in the Sikri office went down. The following monday, when the team returned, everything was in order.
3	No ability to gather data because of server-issues	12	19.02.2020	21.02.2020	Test API from Difi was down. Makes front end testing very difficult as no data is available. Team got in contact with people who could fix the problem.
4	Unable to receive technical advice	8	20.01.2020	18.02.2020	Sikri employees in Norway were not able to assist with technical information regarding the Elements API. The team was put in contact with professionals from Ukraine.
5	Team member get sick, or is elsewise unavailable	6	02.03.2020	04.03.2020	One member became sick 1½ days and unable to participate in team meetings.
6	Getting stuck because of lack of experience	8	05.03.2020	15.03.2020	A member was stuck for a longer period because of inexperience with setting up Elements configuration manager. Continuous consulting with one of the Sikri Employees.
7	A global pandemic breaks out	5	11.03.2020	N/A	The Corona virus was officially declared a global pandemic, the team responded by awaiting instructions or advice on what to do (Wednesday). Social distancing was adviced a day later, the Sikri offices closed two days later (Friday).
8	Lack of workspace after pandemic or other natural disaster	2	12.03.2020	N/A	In order to stop the spread of disease the Sikri offices were closed until further notice. From the 12th, the team realized that work would have to be conducted from home due to the COVID-19 outbreak. Deciding to work from home starting the following day.
9	No ability to gather data because of server-issues	12	02.04.2020	02.04.2020	Sikri's configuration server was unstable and at several times inaccessible. This hindered members from performing tasks. Tasks that did not require configuration server-data was instead performed while this was an ongoing issue. Team contacted the Sikri staff.
10	Unable to use visual studio due to license expiration	11	14.04.2020	14.04.2020	Due to installation on a new workstation, the product key was not automatically registered despite linking the student account to software. Recovered product key from azure.com/student-preview and provided visual studio with said key.

11	No ability to gather data because of server-issues	12	16.04.2020	16.04.2020	Sikri’s Difi-server was not reachable when the team started working in the morning. This hindered the team from doing work, as it was not possible to test the application. Team contacted the Sikri staff.
12	No ability to gather data because of server-issues	12	20.04.2020	20.04.2020	Sikri’s Elements-server was not reachable when the team started working in the morning. This hindered the team from conducting work, as it was not possible to test the application. Team contacted the Sikri staff.

Quality of end product

From the start of the project the team agreed that quality of the product was to be kept to a high standard. The team had some ideas of how to ensure a high-quality product. Code standards, folder structures, acceptance tests and unit tests were items of value, which the team agreed were to be performed from the start. Acceptance tests are often meant to reflect the usage of a feature in the real world, and the result of that usage (Omland, 2006, p. 3). After the user stories were developed, the team wrote acceptance tests. These describe the actions the user performs and the results from performing them. They were evaluated by a binary; pass or failure (Agile Alliance, 2012).

Unit tests are tests which are performed on each unit of the program. A unit can be described as a simple piece of the overarching program, often performing a single task, with few inputs and outputs. The program is built in this way, in order to create a high level of cohesion (Nilsen, 2020, p. 9). Unit tests are written to test these simple units, by observing the results, and comparing them against the expected result. In order to get a better understanding of quality in the product, the team consulted with the product owner. Some points were brought up, especially those of acceptance tests and further development and maintenance of the system.

Definition of quality

Quality in software development is different for each project. There are multiple factors which determine the quality of the product. Mainly, the quality of a product is derived from the task(s) which is fulfilled (Nilsen, 2020, p. 3). It is therefore imperative for the quality of a software product to clearly define the task(s) which one is trying to solve together with the customer or product owner. Then create some acceptance criteria for the tasks, so that one can be certain of when the task is fulfilled (Daily Life Science, 2013, 0:47). It is at this stage the user stories are developed. The team worked hard, together with the product owner, during the planning of the project to develop the user stories, and their corresponding acceptance criteria. This was done to gather a clear picture of what the product was going to deliver to the user. The acceptance criteria were reworked multiple times over, to ensure the team had a good foundation to create a quality product from.

The team decided on a design that conforms with Elements' design. This made sense as the product is meant to be used as a complimentary piece of software in conjunction with Elements. Another bonus to the team was that the design language was therefore already in place, and the system would be somewhat familiar to the users, see (Appendix, Picture 2.1 Prototype landing page, p. 57). Styling was done to be familiar to the users of Elements (Daily Life Science, 2013, 1:04). The team worked hard to conform to the prototype after it was finished and approved by the product owner.

By the information given from the product owner, the team now had a clear picture of what the performance of the system should be. This, also an important part of the product's quality, would have to be kept in mind during development. The team's product is aiming to align itself with Elements, and its uptime, even though the product developed by the team is not as crucial to the end user as Elements is. The second performance figure to keep in mind during the development was that of response times. A figure of 5 seconds or less was brought up by the product owner. The team worked to keep the response times of the product under the given timeframe. Error handling was also something the team kept in mind under development. The system should handle errors and give a response(s) to the user when an error occurs, as it can help ensure prevention of system errors in the system itself, and tell the user what went wrong and where in the case of an error, which in turn could help the user make decisions and creates a sense of quality (Daily Life Science, 2013, 1:04).

How to ensure quality in a product

The team followed several different methods and ideologies to ensure the continued development of quality in the product. This includes the creation of routines for writing and implementing features, writing documentation on how to perform relevant tasks, and the use of tools and risk management.

The iron triangle

“The iron triangle” depicts the balance between the four elements of Time, Cost, Quality and Features. In system development, something must give compared to the others (Omland, 2006, p. 6, 7). In the case of the development of the team's product, Enode, it was mainly the features which had to be decreased. During the project, cost was something which the team did not have to think about, however, the team's available time had to be balanced. This to ensure that quality in the product was kept to a high level.

Code standards, folder structure and testing

Code standards and folder structures were developed by the team to make the task of continuing someone else's work and to make further development easier when it’s eventually handed over to Sikri. Having a standard layout will also assist the product owner in the future, if they wish to continue development on the product. “Prettier” is a tool which can be installed as an extension in the IDE Visual Studio Code. “Prettier” allowed the team to keep a consistent format and structure on their code, as it formatted code upon saving. This allowed the team to be consistent when working on the frontend of the product. Unit tests were written for all methods used in the backend from the start of the project. Unit tests were run on each component to ensure the method performed as expected, and that the method could be used in the product.

Documentation

Documentation was performed during the course of the project, first and foremost in the form of a log or diary. The log was written to help the team keep track of decisions which were made during the course of the project. As the team got further into the development of the product, the log proved itself useful multiple times, as uncertainties arose around decisions which had been made. Technical documentation was also developed at the start of the project. This included development guidelines, guidelines for use of version control (Appendix, Git Guide, p. 54), in addition to a sprint plan. When developing a larger system including multiple people a structure of how to write, validate and implement code is vital, and this helped the team greatly in the consistency of their product. Documentation is also crucial for the further development of the product, as the team will not be performing the task of maintaining, or further developing the product.

Involving the product owner

For the product and the development to better align with the wishes of the product owner, the team and the product owner agreed that both parties should participate in status meetings. Status meetings were at first held at the end of each sprint. Later, during the COVID-19 pandemic, this was altered to one meeting every week to preserve the quality and continuous incremental improvements to the product. This benefitted both the team and the product owner.

Acceptance tests were written for each user story. These are central to the system, as all features are developed from their corresponding user story. The team worked closely with the product owner when developing the acceptance tests in order to ensure alignment between the two parties.

Security and risk assessment

During the planning phase of the project, security was kept close in mind as there was uncertainty about the data to be handled and the GDPR implications this could have. It was decided that sensitive and personal information would not be handled by the product. Neither should any data be stored by the system. Early on in the project the team decided on developing a risk assessment, risk matrix, and issue log. The value of these became clear, as all stakeholders can view the risks which the team sees themselves facing, and what to do if a risk is realized. The risk assessment and incident log proved to be useful during the COVID-19 pandemic.

Recommendations

Discuss within the team and create clear and realistic goals from the start of the project. Estimating tasks and time at the start of a project can prove troublesome as a result from lack of information and knowledge. Nevertheless, defining a common goal for the project can be of great value and a tool to keep the team on track later on. Roughly estimating time can also be of help early on. All projects have a time budget or a deadline which has to be kept in mind and estimating time early on can help create a picture of how much time the team has at their disposal.

Important decisions and meetings should be documented. The task of documenting can seem mundane and not worth the time it takes when performed on a daily basis. However, the team experienced on several occasions that the log was a useful tool. When utilized in the right way it may prevent supposition, misunderstandings, extra investigative work, and superfluous discussions.

Git Guide

Man er klar til å pushe når:

- funksjonene i koden matcher brukerhistorien(e).
- alle tester er laget + passert. Dette gjelder både positive og negative tester.
- koden ser fin ut: ingen ut-kommentert kode, fornuftige variabelnavn, ingen store mellomrom/tomrom i koden.
- Alle tester (All tests) passerer – slik at man vet at man ikke har laget nye feil en plass.

Lage egen branch:

- git checkout -b branchName (følg naming standard)
- git add -A
- git commit -a
- git push (kan være du må sette upstream: git push -u origin branchName)

Pushe til egen branch:

- git branch – sjekk at du er i din egen branch.
- git add .
- git commit -m “ lur text”
- git push
- create pull request à koden skal inn til code review NB: Husk å sette opp en reviewer!!

Reviewer checklist:

- variabelnavn i henhold til standard
- unit tester på plass
- fint formatert kode

Technical specification

By group Brukerfeil

13.01.2020

Brief Overview

The group is going to develop a system to be used in relation with the application Elements. The system is a case and archive system used for application processing amongst other things. Sikri wants the group to develop a stand-alone extension to Elements in order to be able to track the status of mail which has not been delivered properly or has failed somewhere in the current system.

The users of our system:
Archivists
System owners
System administrators
Sikri AS employees (who?)
Department leaders
Case managers
Assistants?

Application Specification

The application the team is going to develop will act as a status dashboard for digital post sent in Elements. The application will show four different states of a sent digital letter: sent, received, read and/or failed. This will be collected from the APIs of Elements and DIFI. The application may be used by end users in order to view statuses of digital post. In addition, the application may be used by the developers for troubleshooting, and as a measure of debugging the path of the post that is failing.

Goals

The group aims to develop a reliable application that is able to present accurate information in a user friendly manner. The application will have a focus on message failures, making it clear to the user what, and where a message has failed.

The group will also strive to deliver a product which has clean code and invites to further development.

Minimum Viable Product (MVP)

The MVP consists of the features proposed by the user stories prioritized as “Must have”.
The product should at least include a list of incoming and outgoing messages for an organization. The list is sorted by most recent date of creation and each message has a status with a corresponding icon. Each message can be clicked so that the user can gather information about what integration points the message has visited. This information should be presented in a timeline, displaying the integration points the message has been through. A color and icon representing the status of the message will be displayed together with each integration point.

Functional Requirements

Function	User Story	Explanation
Select organization	#1	User should be able to select an organization from a dropdown menu
Display list of organization’s sent and received messages	#2 #3	User should get a list of messages depending on the organization they choose listed by most recent at the top
Message in color depending on status	#4	User will be able to see the statuses of the messages in the list
Display timeline of nodes, with	#5	User will see a timeline of the nodes

statuses on nodes		that the messages has passed through in transmission, and the statuses of the nodes
Display status of received and sent messages of all organizations	#6	User will see sent and received messages of all organizations, along with their statuses
Search in messages	#7 #8	User should be able to search for organizationID/personID and messageID
Filter messages	#9 #10 #11	User should be able to filter messages by status, incoming and outgoing and message type
Sorting messages	#12 #13 #14 #15	User should be able to sort messages (ascending/descending) by creation date and time, last update, sender/receiver and status
Statistical view of messages	#16 #17	Display statistics of message statuses and message types
Detailed information about receipt	#18	Display more detailed information about the receipt for each node in the timeline

Non-Functional Requirements

Usability

The product should be easily usable, requiring little effort to be properly used. This should be achieved by having a structured and organized user interface, using good design principles.

Integrity

The product requires a high level of integrity, ensuring an accurate status between the relevant parties. This is to ensure that the data retrieved will not be altered in our application, in order to maintain consistency.

Confidentiality

The status of the messages should have a high level of confidentiality, however public information from DIFI could be accessible to all authorized users. Regarding messages that are not public the message statuses should be restricted to the relevant people.

Reliability

The application the team is developing, is to be used in close relation to Sikri's Elements. The uptime requirements should be 90% or above. As the system is not quite as crucial as Elements itself is for the users.

Performance

Response-time should be kept as low as possible. We will aim for a response time of less than five second, as long as the data from the API's are located in the back end of the system. Response time will depend, to some degree, on elements outside the scope of the system locally. As data is transferred between API's, from different locations both geographically and logistically over the Internet. To lower perceived response time of the system some form of loading system could be implemented. This to make the system responsive to user input and lower the perceived wait time for the user.

Technologies

Kubernetes - Deploy application in a container cluster

React (with TypeScript) - To develop frontend for the application.

SASS?

DevOps - To plan and manage the project

Git - DevOps repository - To control and version source code

[ASP].NET Core (OData) - Backend app to serve frontend with data Used as the current application is in the process of being re-written as micro services, running on containers in conjunction with Linux.

Factory Acceptance Test 1

1. Choose an organization from a list
MUST HAVE
Description: As a user I want to start by choosing my organization from a list
In order to see my organization's list of messages
Status: Tror denne er akseptert. Note(Må implementere redirect hvis man endrer orgId i URL til ugyldig orgId). Godkjent

2. List of organization's sent and received messages
MUST HAVE
Description:As a caseworker, I want to see lists of my organization's sent and received messages, in order to quickly access information regarding messages.
Status: Viser to lister. Godkjent

3. See most recent messages at the top of the list
MUST HAVE
Description: As a caseworker I want to see the most recent messages at the top of the list, in order to quickly access information on the most recent messages.
Status: Meldinger sortert på lastUpdated ikke creationDateAndTime. Godkjent

4. Each message in list is shown with a colored background
MUST HAVE
Description: As a caseworker I want each message shown with a color~~ed icon~~ correlating to the message's status.
Status: Viser vel ikke fargene i Elements? Samme farger som i DIFI. Godkjent

~~**5. Show all nodes a message has traveled through**~~
~~**MUST HAVE**~~
~~Description: As a caseworker I want to see all nodes a message has traveled through, in order to better understand where a message has failed.~~

~~**6. Show the status of all sent and received messages in the system**~~
~~**SHOULD HAVE**~~
~~Description: As a Sikri-employee, I want to be able to see the status of all sent and received messages in the system, in order to support our customers.~~

7. Search in sent and received messages by sender/receiver
SHOULD HAVE

Description: As a caseworker, I want to search in sent and received messages by organizationId/ personId, in order to quickly find a specific message.
Status: Ikke godkjent. Gjør i tillegg live-søk mens man skriver

8. Search messages by messageID
SHOULD HAVE
Description: As a caseworker I want to search messages by messageID, in order to quickly find specific messages.
Status: Resultat vises i message-modal. Kun én melding derav ikke sortert. Ikke godkjent

9. Filter messages by status
SHOULD HAVE
Description: As a caseworker, I want to filter messages by status, in order to quickly find intended messages.
Status: Akseptert

10. Filter messages by incoming/outgoing
SHOULD HAVE
Description: As a caseworker I want to filter messages by incoming and/or outgoing, in order to quickly find intended messages.
Status: Allerede filtrert i to separate tabeller. Godkjent

11. Filter messages by message type
SHOULD HAVE
Description: As a caseworker, I want to filter messages by message type, in order to quickly find intended messages.
Status: Fjernet i dag da alle meldinger er av samme type (Digitalt/ DPO)

12. Sort messages by creation date and time
COULD HAVE
Description: As a caseworker, I want to sort messages by creation date and time, in order to find recent and old messages.
Status: Akseptert hvis vi rekker å merge. Ikke godkjent

13. Sort messages by last update
COULD HAVE
Description: As a caseworker, I want to sort messages by Last update, in order to quickly find intended messages.

Status: Akseptert hvis vi rekker å merge. Ikke godkjent

14. Sort messages by sender or receiver

COULD HAVE

Description: As a caseworker I want to sort messages by sender or receiver, in order to find the intended sender/receiver

Status: Akseptert hvis vi rekker å merge. Ikke godkjent

15. Sort messages by status

COULD HAVE

Description: As a caseworker I want to sort messages by status, in order to more easily find messages I want to view.

Status: Akseptert hvis vi rekker å merge. Ikke godkjent

16. Statistics of message statuses

COULD HAVE

Description: As a caseworker, I want to view statistics over message statuses, in order to analyze trends.

Status: Not implemented

17. Statistics of message type

COULD HAVE

Description: As a caseworker I want to see statistics over message types, in order to better understand the types of messages that are being sent.

Status: Not implemented

18. View additional information

COULD HAVE

Description: As a system administrator, I want to be able to view additional information about the receipt of a message, in order to further troubleshoot.

Status: Not implemented

19. Login feature

COULD HAVE

Description: As a caseworker, I want a login feature in the system. So that the system recognizes me, and which organization I am from automatically.

Status: Not implemented

Factory Acceptance Test 2

<u>7. Search in sent and received messages by sender/receiver</u>
SHOULD HAVE
Description: As a caseworker, I want to search in sent and received messages by organizationId/ personId, in order to quickly find a specific message.
Status: Godkjent

<u>8. Search messages by messageID</u>
SHOULD HAVE
Description: As a caseworker I want to search messages by messageID, in order to quickly find specific messages.
Status: Resultat vises i message-modal. Kun én melding derav ikke sortert, Godkjent

<u>12. Sort messages by creation date and time</u>
COULD HAVE
Description: As a caseworker, I want to sort messages by creation date and time, in order to find recent and old messages.
Status: Akseptert hvis vi rekker å merge, , Godkjent

<u>13. Sort messages by last update</u>
COULD HAVE
Description: As a caseworker, I want to sort messages by Last update, in order to quickly find intended messages.
Status: Akseptert hvis vi rekker å merge, Godkjent

<u>14. Sort messages by sender or receiver</u>
COULD HAVE
Description: As a caseworker I want to sort messages by sender or receiver, in order to find the intended sender/receiver
Status: Akseptert hvis vi rekker å merge, Godkjent

<u>15. Sort messages by status</u>
COULD HAVE
Description: As a caseworker I want to sort messages by status, in order to more easily find messages I want to view.
Status: Akseptert hvis vi rekker å merge, Godkjent

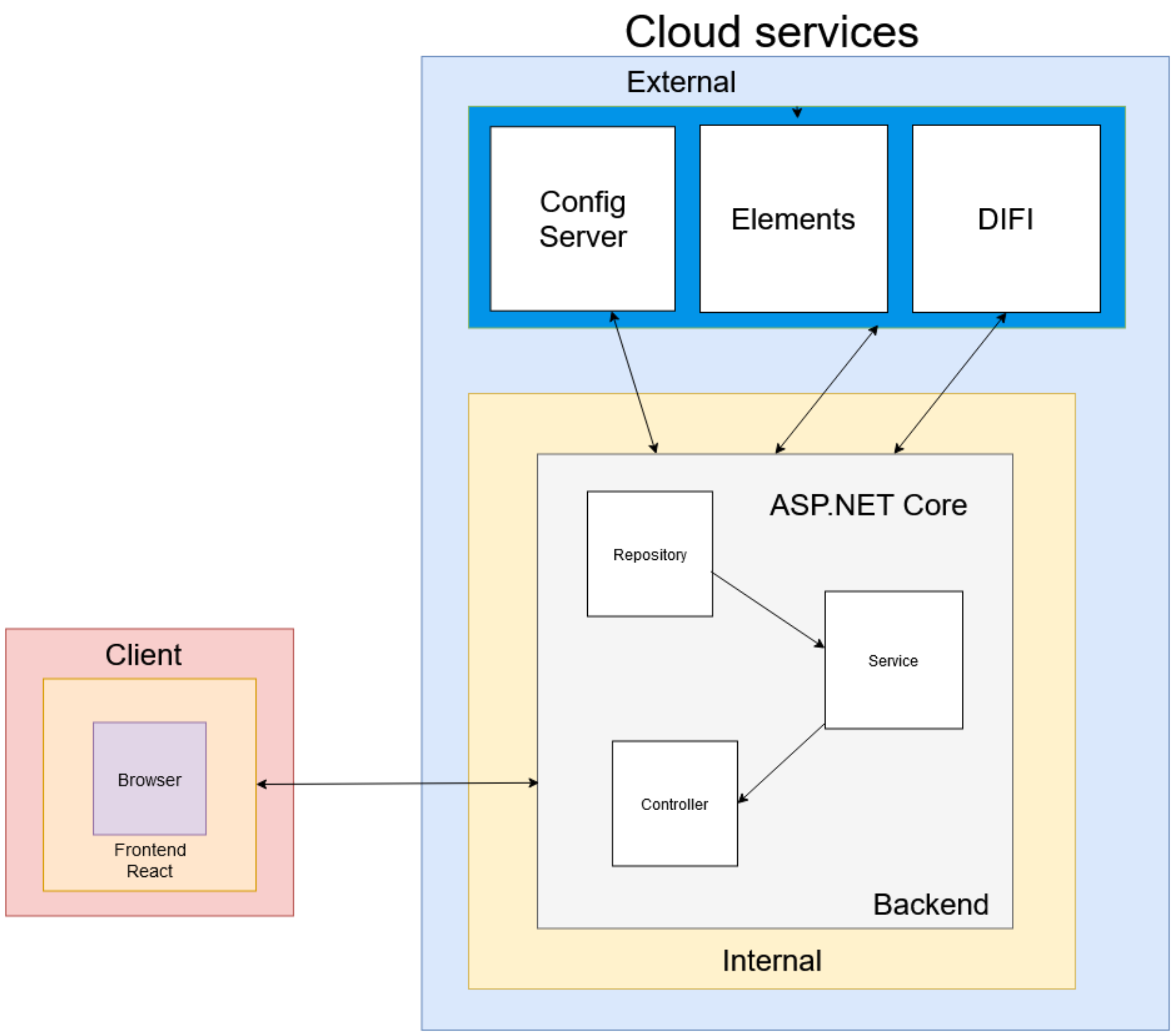
16. Statisticcs of message statuses
COULD HAVE
Description: As a caseworker, I want to view statistics over message statuses, in order to analyze trends.
Status: Not implemented

17. Statistics of message type
COULD HAVE
Description: As a caseworker I want to see statistics over message types, in order to better understand the types of messages that are being sent.
Status: Not implemented

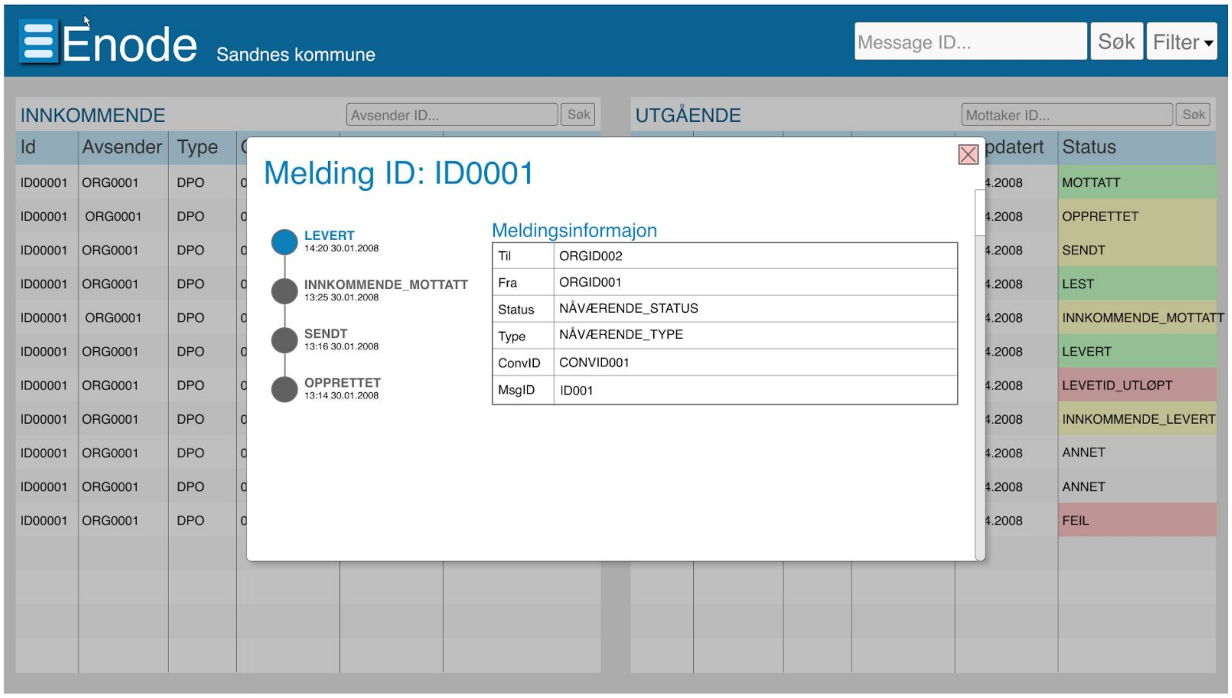
18. View additional information
COULD HAVE
Description: As a system administrator, I want to be able to view additional information about the receipt of a message, in order to further troubleshoot.
Status: Not implemented

19. Login feature
COULD HAVE
Description: As a caseworker, I want a login feature in the system. So that the system recognizes me, and which organization I am from automatically.
Status: Not implemented

Picture 2, Application architecture



Picture 3, Prototype selected message



Melding ID: ID0001

LEVERT
14:20 30.01.2008

INNKOMMENDE_MOTTATT
13:25 30.01.2008

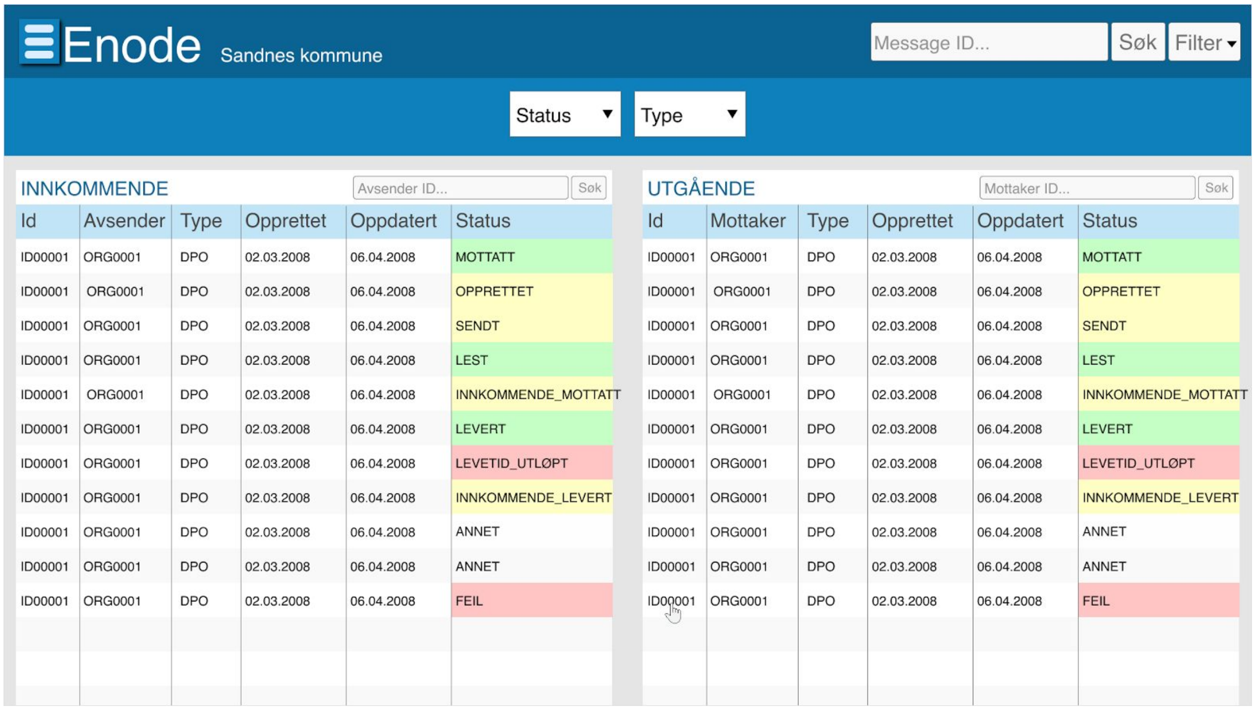
SENDT
13:16 30.01.2008

OPPRETTET
13:14 30.01.2008

Meldingsinformajon

Til	ORGID002
Fra	ORGID001
Status	NÅVÆRENDE_STATUS
Type	NÅVÆRENDE_TYPE
ConvID	CONVID001
MsgID	ID001

Picture 3.1, Prototype filter 1



INNKOMMENDE

Avsender ID... Søk

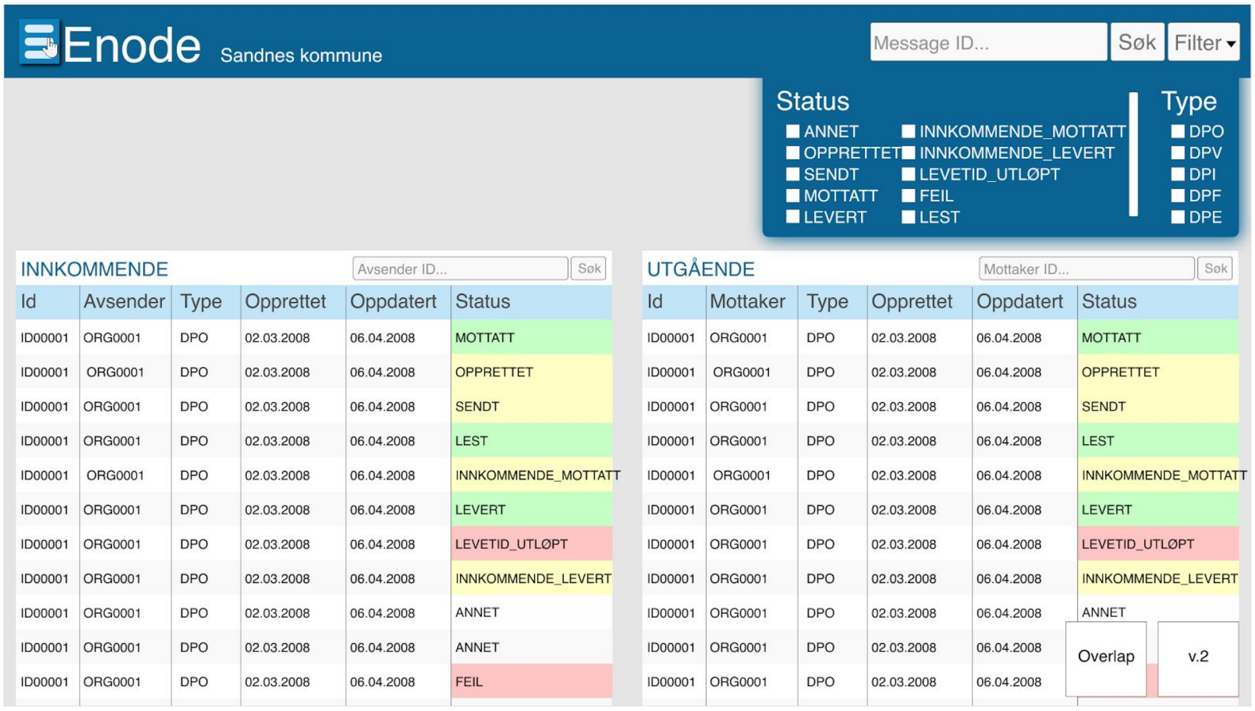
Id	Avsender	Type	Opprettet	Oppdatert	Status
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	OPPRETTET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	SENDT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEST
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVETID_UTLØPT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	FEIL

UTGÅENDE

Mottaker ID... Søk

Id	Mottaker	Type	Opprettet	Oppdatert	Status
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	OPPRETTET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	SENDT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEST
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVETID_UTLØPT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	FEIL

Picture 3.2, Prototype filter 2



Enode Sandnes kommune

Message ID... Søk Filter

Status

☐ ANNET	☐ INNKOMMENDE_MOTTATT
☐ OPPRETTET	☐ INNKOMMENDE_LEVERT
☐ SENDT	☐ LEVETID_UTLØPT
☐ MOTTATT	☐ FEIL
☐ LEVERT	☐ LEST

Type

☐ DPO
☐ DPV
☐ DPI
☐ DPF
☐ DPE

INNKOMMENDE Avsender ID... Søk

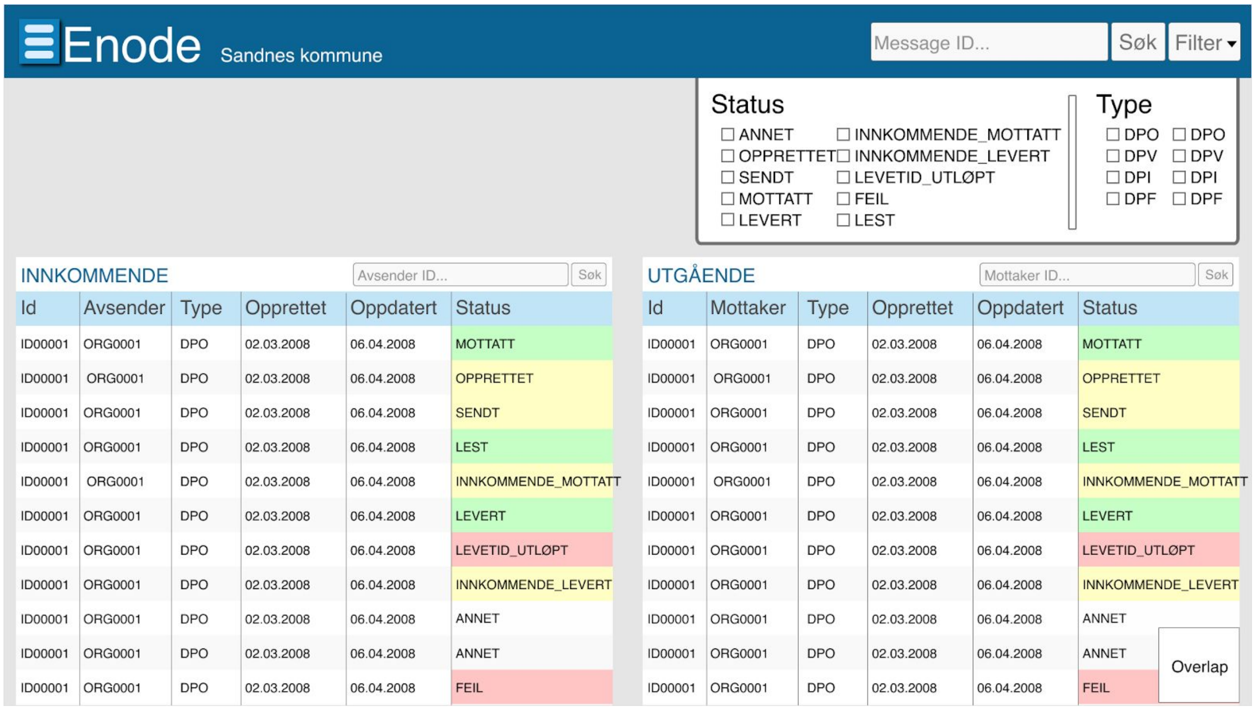
Id	Avsender	Type	Opprettet	Oppdatert	Status
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	OPPRETTET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	SENDT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEST
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVETID_UTLØPT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	FEIL

UTGÅENDE Mottaker ID... Søk

Id	Mottaker	Type	Opprettet	Oppdatert	Status
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	OPPRETTET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	SENDT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEST
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVETID_UTLØPT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	

Overlap v.2

Picture 3.3, Prototype filter 3



Enode Sandnes kommune

Message ID... Søk Filter ▾

Status

☐ ANNET	☐ INNKOMMENDE_MOTTATT
☐ OPPRETTET	☐ INNKOMMENDE_LEVERT
☐ SENDT	☐ LEVETID_UTLØPT
☐ MOTTATT	☐ FEIL
☐ LEVERT	☐ LEST

Type

☐ DPO	☐ DPO
☐ DPV	☐ DPV
☐ DPI	☐ DPI
☐ DPF	☐ DPF

INNKOMMENDE

Avsender ID... Søk

Id	Avsender	Type	Opprettet	Oppdatert	Status
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	OPPRETTET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	SENDT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEST
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVETID_UTLØPT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	FEIL

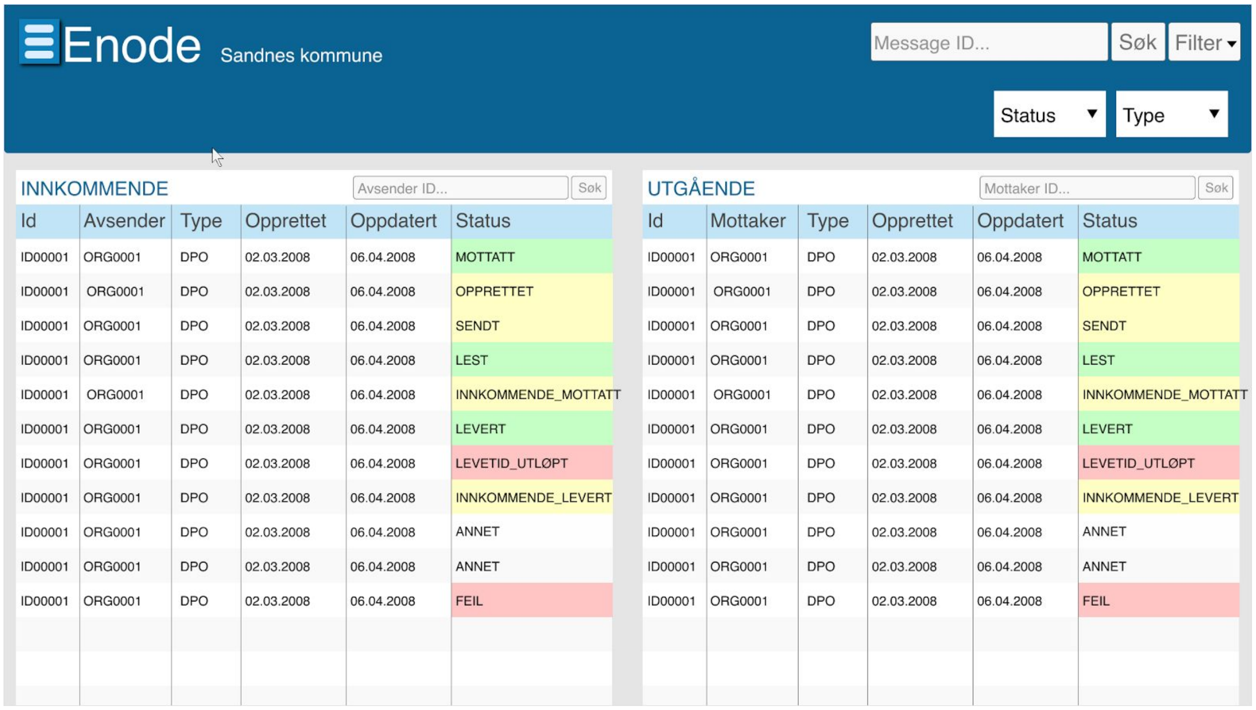
UTGÅENDE

Mottaker ID... Søk

Id	Mottaker	Type	Opprettet	Oppdatert	Status
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	OPPRETTET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	SENDT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEST
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVETID_UTLØPT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	FEIL

Overlap

Picture 3.4, Prototype filter 4



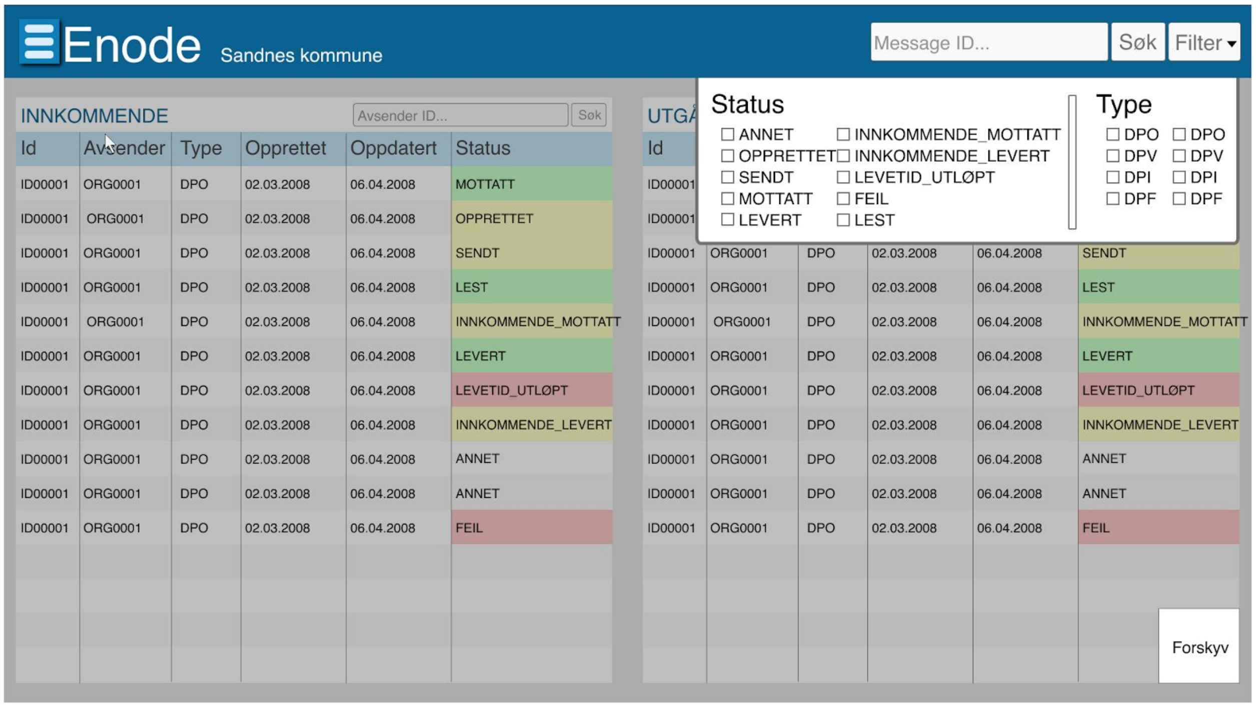
INNKOMMENDE

Id	Avsender	Type	Opprettet	Oppdatert	Status
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	OPPRETTET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	SENDT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEST
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVETID_UTLØPT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	FEIL

UTGÅENDE

Id	Mottaker	Type	Opprettet	Oppdatert	Status
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	OPPRETTET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	SENDT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEST
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVETID_UTLØPT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	FEIL

Picture 3.5, Prototype filter 5



Status

☐ ANNET	☐ INNKOMMENDE_MOTTATT
☐ OPPRETTET	☐ INNKOMMENDE_LEVERT
☐ SENDT	☐ LEVETID_UTLØPT
☐ MOTTATT	☐ FEIL
☐ LEVERT	☐ LEST

Type

☐ DPO	☐ DPO
☐ DPV	☐ DPV
☐ DPI	☐ DPI
☐ DPF	☐ DPF

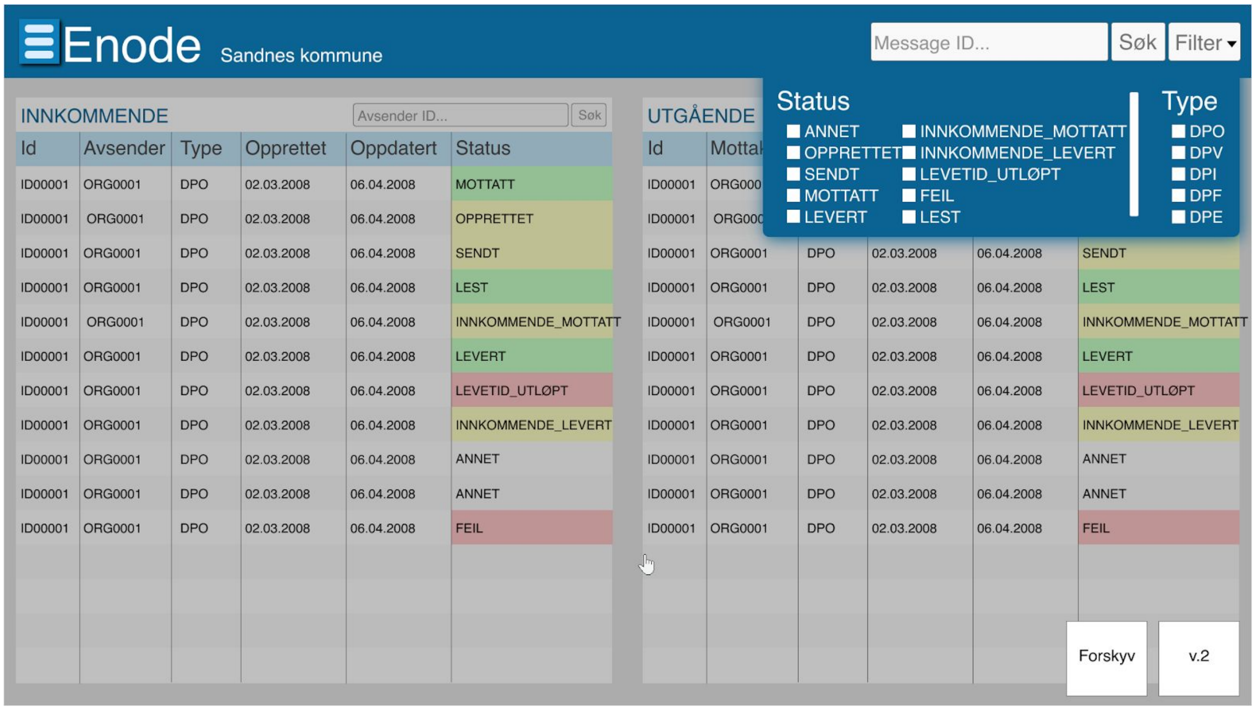
INNKOMMENDE

Id	Avsender	Type	Opprettet	Oppdatert	Status
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	OPPRETTET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	SENDT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEST
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVETID_UTLØPT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	FEIL

UTGÅ

Id					
ID00001					
ID00001					
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	SENDT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEST
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVETID_UTLØPT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	FEIL

Picture 3.6, Prototype filter 6



Enode Sandnes kommune

Message ID... Søk Filter ▾

Status
- ☐ ANNET
- ☐ OPPRETTET
- ☐ SENDT
- ☐ MOTTATT
- ☐ LEVERT
- ☐ INNKOMMENDE_MOTTATT
- ☐ INNKOMMENDE_LEVERT
- ☐ LEVETID_UTLØPT
- ☐ FEIL
- ☐ LEST

Type
- ☐ DPO
- ☐ DPV
- ☐ DPI
- ☐ DPF
- ☐ DPE

INNKOMMENDE Avsender ID... Søk

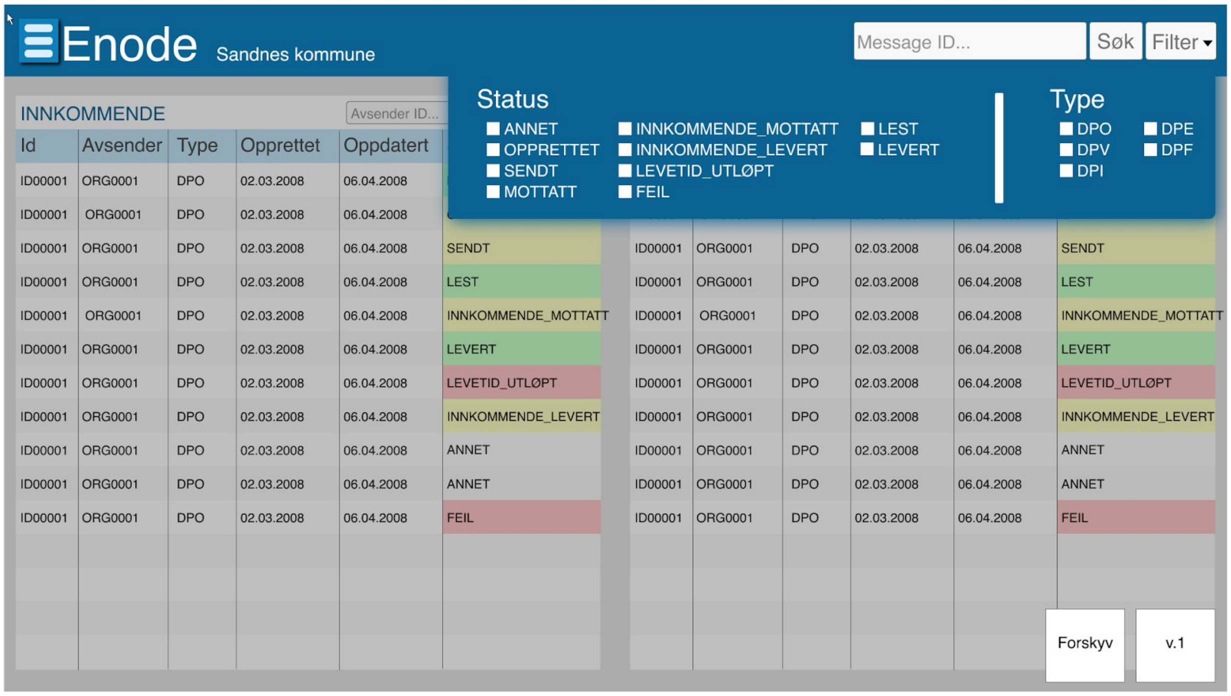
Id	Avsender	Type	Opprettet	Oppdatert	Status
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	OPPRETTET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	SENDT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEST
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVETID_UTLØPT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	FEIL

UTGÅENDE

Id	Mottak				
ID00001	ORG000				
ID00001	ORG000				
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	SENDT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEST
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVETID_UTLØPT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	FEIL

Forskyv v.2

Picture 3.7, Prototype filter 7



Status

☐ ANNET ☐ INNKOMMENDE_MOTTATT ☐ LEST
☐ OPPRETTET ☐ INNKOMMENDE_LEVERT ☐ LEVERT
☐ SENDT ☐ LEVETID_UTLØPT
☐ MOTTATT ☐ FEIL

Type

☐ DPO ☐ DPE
☐ DPV ☐ DPF
☐ DPI

Id	Avsender	Type	Opprettet	Oppdatert	
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	SENDT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEST
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVETID_UTLØPT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	FEIL

ID00001	ORG0001	DPO	02.03.2008	06.04.2008	SENDT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEST
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_MOTTATT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	LEVETID_UTLØPT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	INNKOMMENDE_LEVERT
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	ANNET
ID00001	ORG0001	DPO	02.03.2008	06.04.2008	FEIL

Steering Committee Meeting #1

Sted:

Kjøita Park 21, 4630 Kristiansand S 19. Feb. 2020

Deltakere

Navn	**Roller**
Marius	SCRUM master
Merethe	Produkteier
Geir Inge	Veileder
Espen	Referat taker

Agenda

Prosjekt plan og status
Intro om produktet. En web app for å vise status som går innom Elements sak og arkiv
Melding – hører til saker som blir opprettet for kommuner, private og mange andre.
Systemet skal kunne «følge» meldinger som går mellom. For å se feil, og hvor den er innom
Hvordan? Konfig server – org. Som benytter El. Db med saker – kobler opp mot Elements og DIFI API -
Ark. Strukt. Rerpo- henter API data, Service behandler. Controller-----

Plan og status:

Sprint 2 nå. Ferdig med 1. MÅL HER
Pre Sprint: Sette oss inn i tek og tools. Enkelt, mye som er gjort med C#, og tools som DevOps boards VS. Jira. Valgte DevOps.
Hinder i **sprint 1**– koble til API, ingen tilgang med autentisering fikk ikke koble til API
Sprint 2- MVP, MUST have, bestemt hvordan UI skal se ut. Første utkast til prototype.

Sprinter – burndown charts

Burndown chart Sprint 1:

Litt til å komme i gang med C#, og sette oss inn i API'er og Unit testing. US – brytes ned i PBIer, som
Tidsestimat 150 timer, mindre enn taket. 5.5 timer per person, per dag. Ca. Halvparten av NOE ble estimert, vanskelig å estimere tid i begynnelsen.

Sprint 2

Nesten fulgt Ideell trend. Total scope litt opp, men ikke så mye som i første sprint.
Estimerte for flere timer enn i Sprint 1. Blir bedre med mer erfaring. Fungert bra, og vi er nesten i mål med PBI'er to dager før slutten av sprinten.

Liten demo

Benytter test API fra DIFI. Screen av hvordan listen ser ut som det er nå.
Forside, ingen logging, men det kan komme etter hvert. Velger Org. Og bekrefter. Skal da få to lister med Inn og Ut fra den org. Som er valg. Ikon og farge, sammen med status i teks.
Prototype er laget. Velger org. Skal kunne søke på ID- på melange, samt sender og mottaker.
Filter som type melding, om det er Kommune, privatpersoner eller statuser.
Typer er DIFI definerte. (DPO, osv.)

Utfordringer og problemer

Tilgang til API tok lengre enn tenkt**.** Brukte test API, men her er det litt begrenset med data, som hindrer oss litt. Kommer nok til å resultere i reforming av kode. Kom oss «rundt» med å bygge et skjelett på hvordan det skal være-

Diskutert hvordan ting skal gjøres.

Tidsestimering – mye nytt, ingen særlig erfaring med dette, men vi blir bedre og bedre etter hvert som sprinter kommer og går.

Hvordan dele opp US. Inn i PBIer eller tasker. Vanskelig å si hva som kommer til å skje to uker fram.
Bratt læringskurve når det kommer til teknologier og verktøy.
Sliter en del med Wifi, der vi ikke får koblet til og dette ut til tider.
Mye tid på disk. Og synsing mellom medlemmene i gruppen.

Videre

Mandag sprint planning. Se over hvordan sprint 2 gikk.
Se på hva som må forandres i det som er gjort ettersom ny data er tilgjengelig.

Planlegge bedre med ny informasjon. Refaktorere kode.
Koble opp Elements API. Etter hvert config server, slik at vi kan hente ut skikkelige Organisasjoner.
Se på filtrering og sortering.
Litt mer layout, ta inspirasjon fra Elements.
Prøver å få bygget prosjektet i Kubernetes, slik at det kommer i sky.
Begynne på prosjektrapport.

Diskusjon

PBI for ikke funk. Ting?

Skal det estimeres tid til ting som ikke er funksjonelt for produktet. Skal det lages PBI for ting som dette?
Det er en del av prosjektet – Det vi synes at vi skal lage PBI på, lager vi en PBI på. Slik gjøres det i Sikri. eller lage et eget scrum-board, men ikke blande oss inn i for mange systemer. Da blir det fort tull.
Geir er enig, det er en del av prosjektet, og tid må gå til ikke funksjonelle ting.
Spesielt ting som rapporten, da dette kommer til å ta mye tid.
Få fremhevet alt, slik at det ikke er ting vi glemmer.
Sorterte ut, det som ikke er funksjonelt.
Legge det under en annen feature?
Eller lage nytt board – så lenge vi har kontroll på det vi skal gjøre.
Få det med på en eller annen måte. Hjelper i rapport etter hvert, viser kompleksitet og arbeid som må gjøres under et prosjekt.

Estimering

Merethe - Veldige gode chart. Er vi veldig gode på estimering eller safer vi oss selv?
Vi har estimert, ved å dele opp tasker i ting vi er rimelig sikre på at vi kan klare å fullføre. Vi har nok safet litt for at vi ikke skal bomme helt på estimatet. Kan hende at vi ikke er strenge nok at tasker er ferdige? (Definition of done). Estimate er 150 timer av total 280. Kanskje ikke et fullt bilde på hvordan sprinten har vært. Også ettersom at vi ikke har lagt inn ting som møter, og andre ikke funksjonelle ting. Ting som å sette oss inn i tek. Og andre, er ikke representert I charten. Vanskelig å sette PBI på ting som dette.
Geir- ser bra ut, men få kanskje inn det som ikke er funksjon. Slik at alt arbeid er representert i charten.
Merethe – halvparten av tiden er gått til andre ting enn utvikling/koding. Vært interessant og sett ting som også ikke er funksjonelle.

Ikke legg inn etter at PBI er ferdig! Vil vi få en peak i «remaining work». Vi legger inn det som vi har foran oss.

Risiko analyse

Merethe - se de høyest vurderte.

De største til nå har vært ting som å ikke få tilgang til API'er, og hvilke data vi kan hente ut fra API'ene. Som med tidslinje, der vi hadde sett for oss at det skal være en tidslinje over IP som meldinger har vært. Det viser seg gjerne at vi ikke får de dataene vi trenger for å vise tidslinje. Ikke noe som har skjedd enda, men er en risk.

Geir- skal vi jobbe videre med risk, og loggen. Eller tenker vi at det et fast dokument? Den er ikke vært noe om vi ikke bruker dette videre.

Vi logger hindringer etter hvert i risk loggen. Dette har vi ikke vært gode på så langt.API er nede? Legg ting som dette inn. Merethe - Dokumentere vi koden?

Ikke så langt, nei. Vi er ganske klare på at mye må refaktoreres, som en følge har vi ikke dokumentert da vi vet at ting må skrives om. Etter hvert som vi får riktig data inn, skal koden skrives «som den blir» og dokumentering må på plass. Det er levende dokument. holde det konstant oppdatert, og legg inn risker som vi møter på som ikke er lagt inn i risikoanalysen

Merethe og Geir: ser bra ut, og begge er imponerte over hvor lang vi har kommet. Og vi ligger bra an så langt. Godt at vi har en løsning der vi kan sitte på plassen og få hjelp om det er noe.

Geir- skal det integreres med Elements? Skal man kunne trykke på et element i listen og komme inn i Elements med den meldingen som er valgt.

Tanken er at det gjerne kan integreres senere, men dette er ikke en del av scopet.

Avgjørelser som blir tatt

HUSK: ha med Risikoanalyse og logg med neste SCM møte.

Neste treff SCM? Planen er Uke 13 eller 14. Mars-April

Steering Committee Meeting #2

Sted:

MS Teams, Norge 1. Arp. 2020

Deltakere

Navn	**Roller**
Marius	SCRUM master
Merethe	Produkteier
Geir Inge	Veileder
Espen	Referat

Oppsummering av forrige møte
Pre sprint der bruker historier og plan for prosjektet ble utviklet.
Sprint en begynte gruppen på Must have features.
Slutten av sprint 2. Fortsatt med must haves.
Begynte med prototypen.
Utfordringer:
Gruppen hadde ikke tilgang til APIene, der data skal hentes fra
Sprint planning, vanlig å estimere tid. Ikke gode nok til å dele opp PBIer.
Lage eget scrum board. Fungert bra for å skille det som er funksjonelt og det som ikke er.
Bli bedre på å dokumentere koden.
Bli bedre på issue log og risikoanalysen.
Bli bedre på å definere
Agenda
Prosjekt, plan og status

Risikoanalyse:
Gruppen har jobbet mer med risikoanalysen og loggen siden sist styringsmøte.

Vi gikk gjennom i fellesskap og la inn det vi kunne komme på. Det samme ble gjort med loggen. Probability og impact har begge en score på 1-5. Disse ganges, slik at høyest score er 25. For så at risikoen legges vurderes.
Gruppen har blitt bedre på å skrive ned i Issue loggen. Hvordan vi håndtere det sammen med dato, åpne og lukke.

Sprinter – burndown charts
Sprint 3:
Fikk laget skjema for å hente organisasjoner fra config server.
Funksjoner for å hente meldinger fra Elements, ved bruk av data som kommer fra config server.
Frontend- ble det laget prototype.
Unit tester ble laget for noen metoder i frontend.
Innlevering i IS305 ble lagt inn som en ikke funksjonelle PBI i det andre DevOps boardet.
Ikke alle PBIer ble ferdigstilt, da gruppen prøvde å estimere flere timer enn tidligere.

Sprint 4:

Det ble skrevet docker filer for å deploye prosjektet. Dette ble vi ikke helt ferdig med.
Metoder ble skrevet for å kombinerer data fra DIFI og Elements API.
Hamburger og filter meny ble utviklet i stil med prototypen.
Vindu for statuser på meldinger ble utviklet. Her kommer mer informasjon om meldingen. Samt statuser som meldingen har hatt på forskjellige tidspunkter.
Det ble planlagt tid for oppgave i IS305, men den ble ikke gjort denne sprinten.

Sprint 5:
Uke 1 gikk på å skrive første utkast av rapporten, som skal inn i IS305.
Uke 2. Begynte vi på Must have features igjen ettersom rapporten ble ferdigstilt.
Refaktorering og sletting av metoder og kode som ikke brukes lenger ble gjort.
Nye metoder for å hente data fra begge APIer kobles til frontend, slik at vi bruker reelle data.

Demo
Organisasjoner som ligger der nå er enda ikke reelle. Implementering av Gecko og Sikri, som er reelle data jobbes med denne sprinten (5).
Hamburger meny er lagt inn på venstre side. Her kan man gå tilbake til velg org. Statistikk siden som er en should have og derfor ikke ferdigstilt enda.
Navn på den organisasjonen som er valgt vises ikke enda, bare org nummeret. Dette skal vi se på senere.
Søk på Meldings ID i høyre hjørne: Her kan man søke på en ID. Søket gjelder for begge tabeller.
Filtermeny er lagt inn i høyre hjørne: her kan man filtrere på status eller type. Filter gjelder for begge tabeller.

Trykk på melding: Viser informasjon som Til og fra, status, type , convID og MsgID. Samt statuser som meldingen har hatt.
Søk på mottaker ved Org id. Slik at man søke opp meldinger på mottaker eller avsender.

Utfordringer og problemer
Avgjørelser om UI og hvordan det skal se ut. Litt blanda meninger innad i gruppen om hvordan det skal se ut. Derfor ble det laget flere versjoner av ting som filter i prototypen, og deretter avstemning i gruppen.
Config server tok tid da vi ikke helt forstod hva som er mulig. Det ble etterhvert tydelig at det ikke mulig å hente alle organisasjoner. Scopet endres litt. Avklart med produkteier.
Elements og Difi APIene. Ikke detaljert forstått hvordan flyten er. Var ikke åpenbart. Ingen god dokumentasjon har ført til mye testing og feiling. Det var ikke klart hva som var utgående fra Elements API, da propertien IsRecipient måtte være true, og ConversationId måtte være på meldingen. Gruppen har vært avhengig av Kurt hos Sikri. Fikk ikke gjennomgang før forrige uke i sprint 5.
Sprint planning, prøve å bli tydelige på når ting er ferdige. Utfordrende å planlegge timer er enda en utfordring, planleggingsmøter er tunge for gruppen.
Pandemien: Ikke noen store utfordringer, Litt større terskel for å kommunisere med Sikri ansatte,men ikke noen store problemer.
Hjemmekontor - gruppen var til å begynne med bekymret for at vi skulle bli mindre effektive enn når vi sitter sammen på Sikri lokalte. Etterhvert viser det seg at vi jobber effektivt også hjemme. Møter opp kl 9 og sitter dagen ut til kl 16 som før.

Videre
Backend henter ut riktig data. Frontend bruker testdata, det som jobbes med nå er å få data fra backend til frontend, slik at reell data vises. Det ønskes å få dette på plass i løpet av denne uken.
Should haves kommer i sprint 6 og 7, sammen med noen Could haves. Rapport må også jobbes med til 1. mai . Får se hvor langt vi kommer på could haves.
Acceptance tests. Siste uke i sprint 6 om ikke enda senere. Dato er ikke satt enda.

Diskusjon/annet
Tester:
Merethe- vi kan ikke kjøre acceptance tester testen før featuren er ferdig. Planlegg når vi regner med å være klar, ihvertfall med MVPer. For så å sette opp test. Vi kan ta det i slutten av en sprint der er nok det beste.
Geir Inge- Hva skjer med de features om ikke er
Merethe- Prøv å bli ferdig. Se på prioriteten, om vi er bekymret for det som ikke er ferdig må vi ta en avgjørelse, og se om featuren skal være med. Must haves, **skal** være ferdig! Should have - *bør* være ferdige. Diskuter senere og se litt på prio på brukerhistorier. Det beste ville vært om vi

kunne hatt acceptance tests når alle Should haves er ferdige.

Vi skal snakke med Merethe på et senere tidspunkt og se på features.

Spørsmål:

Geir Inge- Hvordan går det med oss? Er vi friske og alt bra.

Ja. gruppa har det bra i det store. Ingen sykdom enda. Ingen store problemer.

Geir Inge- Imponert over det vi har fått til på denne tiden. Hvem er brukeren av applikasjonen?

Saksbehandler i organisasjoner som bruker Elements. De som er kunder av Elements.

Geir Inge- UI, med melding typer, Id osv. Vil bruker forstå disse tingene.

Org Nummer skal byttes ut med Navn på org. Skule litt mer tekniske ting. Avansert skal være et valg for de som vil se dette for å gjøre det mer brukervennlig.
Vi har fått noe info om hvordan vi kan hente ut org navn, ved bruk av Org nummer som hentes. Dette ser vi på etterhvert. MsgID må vi også ta etterhvert, men i utgangspunktet skal det vises som det er nå.

Merethe- Sliter litt med fokus under planleggingen, og planlegging er litt vanskelig for oss. Hva er det som gjør det vanskelig og hvordan komme rundt det.

Lett å miste fokus under en hel dag med planleggingen. Blitt bedre. Ta litt pauser, ta litt luft for å klare å holde fokus.

Merethe- ser vi nytte i det å kunne planlegge annenhver uke?

Ja vi ser nytten, det holder oss på spor. Og det er også derfor gruppen tyner seg gjennom planleggingen i fellesskap. Etterhvert har vi delt opp i PBIer sammen, så fordeler vi dem på gruppen, da den ansvarlige for PBIen skriver ned det som må gjøres. På denne måten forstår den ansvarlige bedre hva som må gjøres, og hva som står i PBIen.

Merethe- Grensesnitt og det at vi har forskjellige syn osv. Hvorfor har vi ikke inkludere Merethe under diskusjonene. Det er en veldig kundespesifikk ting. Misbruk av tid om eier kan komme inn. Forventer å bli involvert mer når det kommer til ting som utseende og funksjonalitet.

Dette skal vi klare. Om det kommer opp i fremtiden, enten det gjelder funksjonalitet, eller UI skal vi huske å få input fra Merethe.

Merethe- Bra issue log og risk assessment. Bra å se at vi bruker det som et levende dokument. Viser at vi har god kontroll.

Kan vi legge ting inn i risk assessment eller skal det bare inn i logg eller begge deler?
Merethe- Det skal inn i begge deler. Det som faktisk inntreffer skal inn i issue log i tillegg.
Geir Inge- Ser vi nytten av loggen og risk assessment?

Ja, det er litt mye arbeid, men det er bra å ha som en plan på hva som skal gjøres når ting inntreffer.

Merethe- Hvordan tester vi?
Unit testing til nå. Unit tester er skrevet i både front og backend.
Merethe- Kjører vi code review?
Det gjøres av noen på gruppen før ting merges hver gang dette skal skje.

Geir Inge- noen praktiske utfordringer når det kommer til å nå mål? Har vi det vi trenge?
Så lenge vi får støtte av Sikri, og de kontaktpersonene vi har tilgjengelige der, så skal det ikke være noe som stopper oss i å komme i mål.

Merethe- **Husk å sende alle brukerhistorier.**

Avgjørelser som ble tatt

Aksjonspunkter og den ansvarlige:

Hva skal gjøres	**Navn ansvarlig**
SCM 3 Ikke noe sener enn 6. Mai	
Prosjekt innlevering 1. Mai	
Hjemmeeksamen 15. Mai	

Steering Committee Meeting #3

Sted:

MS Teams, Norge 6. Mai 2020

Deltakere

Navn	**Roller**
Marius	SCRUM master
Merethe	Produkteier
Geri-Inge	Veileder
Espen	Referat taker

Agenda

- Oppsummering av forrige møte
- Prosjektplan og status
- Sprint for sprint
- Demo
- Utfordringer og problemer
- Prosjektslutt
- Diskusjon

Oppsummering av forrige møte

Forrige Styringsgruppemøte avholdt i slutten av sprint 5 - 01.04.2020

Status på prosjektet da:
- Sprint 3: Koblet opp mot config, kunne hente melding basert på sender/mottaker(søk)
- Sprint 4: Skrevet docker-fil, laget meldings-modal
- Sprint 5: Refaktorert kode, koblet på Elements og DIFI-API for å vise riktig data (Sikri/Gecko)

Utfordringer:
- Avgjørelser om brukergrensesnitt
- Tok tid før vi skjønte oss på Config Server
- Dataflyten i Elements og Difi
- Sprint planning er fortsatt utfordrende
- Pandemien

Aksjonspunkter:

Kjør gjennomgang av akseptansetester sammen med produkteier

Involvere produkteier oftere i beslutninger om brukergrensesnitt/ funksjonalitet

Prosjekt, plan og status

Sprint 6

- Refaktorert kode, sortering av kolonner, styling-endringer, ferdig med MVP

Sprint 7

- Rapport IS-305, gjennomgang akseptansetester, småfiks på søkefunksjoner

Er nå i siste sprint

Risk Analyse

Uklart om vi skulle oppdaterte analysen, det ble klart under forrige møte. Det er lagt inn nytt issue i denne.

Loggen er det lagt inn ting som har skjedd som at vi ikke kan hente data fra API.

Mistet lisens på Visual Studio.

Alle issues har hatt kort varighet .

Sprinter – burndown charts

Sprint 6:

Ferdig med MVP

Planlag for should haves

Laget Unit testing i backend.

Lagt til rette for mer error handling.

Refactor av kode. Ikke vise typer annet en Digitale.

Modal møtte også refakorerers.

Lagt til funksjon for sortering i listen. Standard er nyest til eldst på siste oppdatert. Kan nå sortere på alle kolonner i listene.

Acceptance tester ble gjort 1 omgang. Fikke noen ikke godkjent.

Fikk ikke ferdigstilt alle PBIer. Helst de som ikke ble godkjent i test.

Sprint 7:

Første uke til is 305 rapport.

Ble ferdig med de fleste PBIene.

Gruppa ble litt splittet den andre uken ettersom noen har startet med 304 rapport, noen jobber enda litt med produktet.

Filter på MessageID, error handling for input i URL, og org navn er nesten på plass.

Demo

Viser screen fra forrige SCM og dette i produktet.
Meldinger hentes nå fra "ekte" API data.
Søkebaren og farger er endret litt på. Noen flere statuser er på plass i utgående. Disse er basert på Elements API. ConvId sjekkes opp mot Difi API i et forsøk på å mappe disse. Det er noen meldinger som kommer opp med Ingen data på Difi sin side. Gir mer verdi for bruker da de ser statuser på begge APIer.

Listen kan man scrolle i. Listene kan være ulik lengde, men holder seg da der de skal på siden.

Trykk på melding, da får man det meste av informasjon som kan hentes på den meldingen.

Disse listes og kan scrolles i.

Tidslinje som viser de statusene meldinga har hatt og tiden den har hatt statusen.

Forbedret søk på avsender og mottaker. Real time søk, slik at man kan skrive litt av et Org nummer og det kommer opp med hits i listen.

Utvidet søk på org nummer om man ønsker mer enn det som vises, om det finnes flere meldinger. Listen går tilbake til default.

Sortering på header i listen. Toggler ascending, descending.

Kan søke på meldings ider. Fungerer på samme måte der man får hits på søket live i listen.

Trykk på søk knappen, da kommer meldingen opp i et eget vindu.

Filter: Statuser. Kan velge flere filter på samme tid. Da kommer det meldingen som har disse statusene opp i listen(e).

Utfordringer og problemer

- Config-skjemaene ble ikke lagret i Config-serveren så det måtte legges inn hver morgen
- Deployment (sette appen ut i produksjon)
- Blir ferdig til første akseptansetest
- APIene ikke tilgjengelig i helger, helligdager og etter arbeidstid

Videre

Rapport IS-304 (20.mai)

Lese til eksamen IS-305 (15.mai)

Noe småfiks på web-applikasjonen

Forberede til muntlig eksamen(5.jun)

Diskusjon/annet

Geir Inge: Burndown Sprint 7. Rekker vi å bli ferdige med de 20 timene som er igjen.

Ikke mye som gjenstår av programmeringsoppgaver. Har fått noen småting som vi kan jobbe med når det kommer til produktet, disse er lagt inn. Derfor ser det litt rart ut og ikke er ferdigstilt. En del av dette er bugs og noen småting. Samt deployment der vi står litt fast til vi får noe veiledning fra Sikri.

Geir Inge: Kort om, funksjonene som ikke ble godkjent

6 brukerhistorier som ikke ble ferdige til første FAT. 4 av disse var sorting. 2 søk. Disse ble ikke helt ferdig (sort). Søk traff ikke helt på det som buker historien sa den skulle gjøre.

Geir Inge: Har vi hatt de siste FAT testene?

Hadde en runde i går der vi fikk godkjent de 6 som var igjen. Så alle must haves og Should haves er implementert. De som er igjen er ting som ikke er mulig å implementerer. Fikk input fra produkteier og andre i Sikri som

Geir Inge: Meldinger som vi ikke finner i DIFI api. Åpnet av mottaker men det er ingen data i DIFI. Hva skjer her?

Tingen er at der er test data, slik at vi tror det er data som er manuelt satt inn, ikke reelle data.

Geir Inge: Api som ikke er tilgjengelige i helger osv. Gjelder dette også Difi sitt API?

Ja, det er ikke DIFI sitt API, slik at det ikke er hostet der. Hver instant startes opp for hver org.

Merethe: Ting som test API osv stenges ned for å spare på kost.

Geir Inge: Tidslinje, hele tiden den første som er markert ut? Hvorfor?

Det skal egentlig være siste status som er først i listen. Dette skal vi fikse. Nyeste skal være siste status.

Merethe: Sprint 7 burndown. Har vi tenkt at vi bare hiver inn det som mangler.

PBI er som ikke ble ferdig i sprint 6 ble satt over i sprint 7. Rapporter det det som har vært hovedfokus i starten av sprinter. Det ble derfor ikke planlagt så mye. Dette har gått utover programmering i sprint 7.

Merethe: har egentlig ikke kjør skikkelig planning på programmeringsoppgaver i sprint 7?

Ja.

Merethe: Det er viktig få det med at rapporten har kludrett till burndown chart.

Geir Inge: Hvordan har det fungert med hjemmekontor?

Vi føler at det har gått veldig bra. Var noen bekymringer til å begynne med, men etterhvert så har det vist seg at det går bra. Alle er klar til kl 9 og sitter til kl 16. Vi har klart å opprettholde samme intensitet på arbeidet. Daily, review og retrospekt er alle ting som vi hart fortsatt med.

Merethe: Skal jeg skrive noe til rapporten

Ja. Husk! Enten Video eller tekst.

Tips til muntlig.

Sensor til at alle skal ta ordet. Bård.

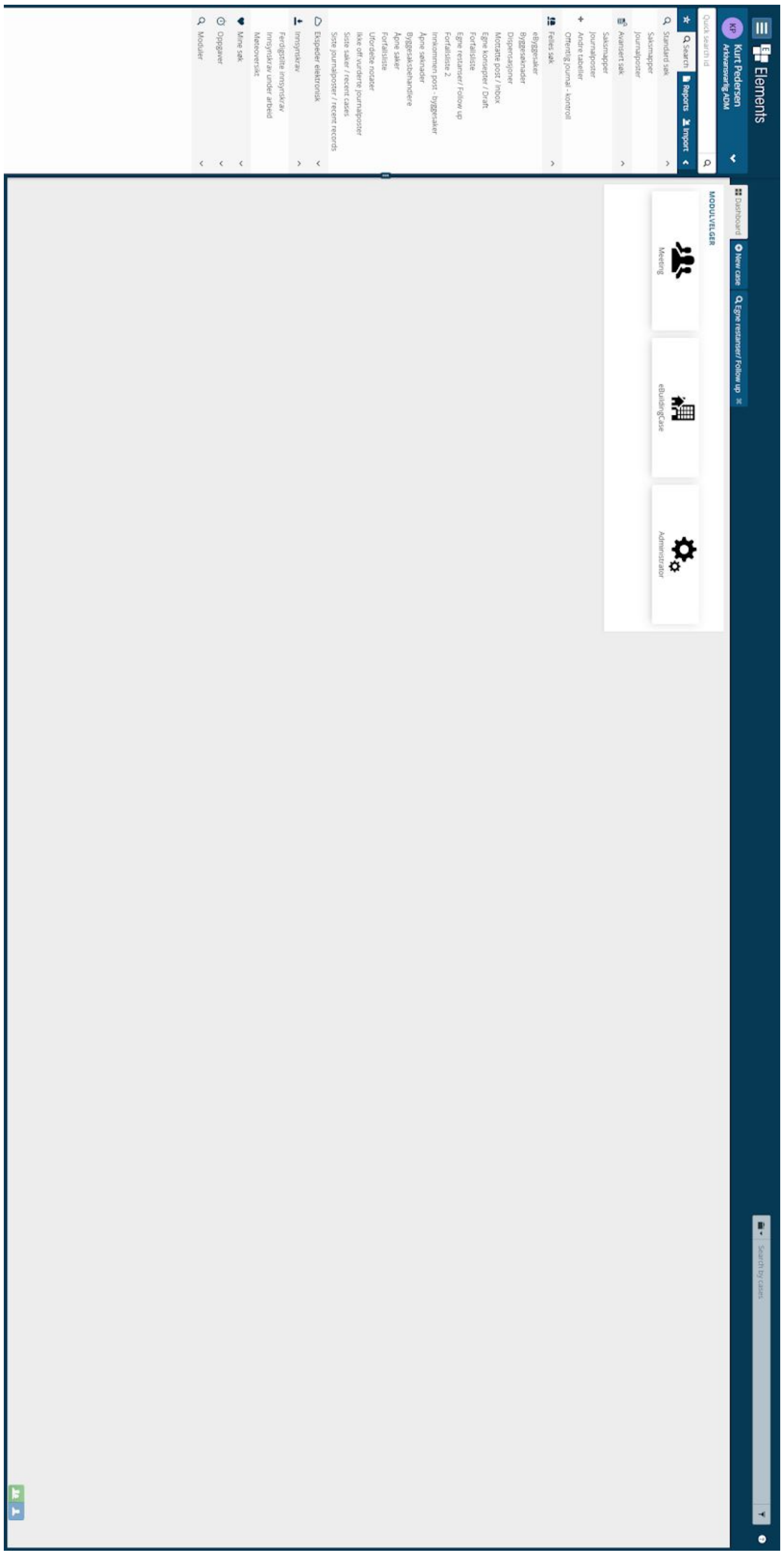
Snakk like mye hver. Alt som mangler i rapporten må dekkes.

Avgjørelser som ble tatt

Aksjonspunkter og den ansvarlige:

Hva skal gjøres	Navn ansvarlig

Picture 4, Elements landing page


Elements
Kurt Pedersen
Arkivansvarlig ADM
Quick search id
Search
Reports
Import
Standard søk
Saksmapper
Journalposter
Avansert søk
Saksmapper
Journalposter
Andre tabeller
Offentlig journal - kontroll
Felles søk
eByggesaker
Byggesøknader
Dispensasjoner
Mottatte post / Inbox
Egne konsepter / Draft
Forfallsliste
Egne restanser/ Follow up
Forfallsliste 2
Innkommen post - byggesaker
Åpne søknader
Byggesaksbehandlere
Åpne saker
Forfallsliste
Ufordelte notater
Ikke off.vurderte journalposter
Siste saker / recent cases
Siste journalposter / recent records
Ekspeder elektronisk
Innsynskrav
Ferdigstilte innsynskrav
Innsynskrav under arbeid
Møteoversikt
Mine søk
Oppgaver
Moduler
Dashboard
New case
Egne restanser/ Follow up
MODULVELGER
Meeting
eBuildingCase
Administrator
Search by cases