Developing a Data Model and an Architecture for a Dental Implant Quality Registry

Lars Vågenes

Master's thesis in Software Engineering at

Department of Computer science, Electrical engineering and Mathematical sciences,
Western Norway University of Applied Sciences

Department of Informatics,
University of Bergen

Supervisors: Svein-Ivar Lillehaug and Yngve Lamo

May 2022

Western Norway
University of
Applied Sciences

Abstract

Currently there are 61 quality registries related to healthcare in Norway, contributing to better quality in the care that is provided within the various areas of healthcare they cover. None of these registries cover fields within the discipline of dentistry. This snapshot of the situation in Norway is mirrored in what is found globally. Whereas quality registries are considered as important systems supporting good research and development of national policies within most areas of healthcare in most countries – clinical research within dentistry most often is based on smaller data sets, whereas policy making has to be based on systematic review articles.

With scientific knowledge on how to develop quality registries within dental care being absent, this thesis aim at developing an artifact for a dental implant quality registry that can be used within a larger research project to build experience and knowledge in how to develop such a quality registry. First, the needs and requirements for a dental implant quality registry is discussed. Then, an artifact was developed using the design science methodology. By investigating other registries and going through several iterations where the artifact was evaluated by domain experts, the data model was developed. The architecture of existing registries and modern design patterns for web applications were explored and used to design the architecture and the API for the artifact. The availability and reliability of the artifact were tested with success, whereas evaluations were done throughout the development process, with a final evaluation done through a semi-structured interview. Finally, suggestions for future improvements are presented.

Contents

List of Figures

Chapter 1

Introduction

This project is part of a larger ongoing research project between the Norwegian Research Centre AS (NORCE), Department of Clinical Dentistry at the University of Bergen (UiB) and the Department of Computer science, Electrical engineering and Mathematical sciences (IDER) at the Western Norway University of applied sciences (HVL). The goal of this research project is to develop and implement a quality registry for dental implants. There are three more master thesis projects included in this project.

The main goal of this master thesis project is to develop a robust conceptual data model and an architecture for a quality registry for dental implants. Combined, this will provide a basis for implementing an artifact that can be used in the larger joint research project to build experience and knowledge to develop a future quality registry for dental implants. Furthermore, and within the scope of this thesis, the backend component for realizing such a dental implant quality registry will be developed. The data model and architecture will be designed to handle the services that are expected of such a quality registry.

1.1 Motivation

In Norway, every year from 2014 to 2017, more than 14000 dental implants were placed which qualified for a refund from Helfo (The Norwegian Health Economics Administration). This resulted in refunds for around 100 million NOK each of the years (Lie et al., 2019). Lie et al. also speculates that there may be just as many implants set without Helfo refund, and Helfo may only refund half of the cost. This means that the Norwegian dental implant market could be estimated to 400 million NOK spread across 30 000 implants. In Europe the dental implant market is estimated to 5.5-6 million implants each year, while 12-18 million implants are used on patients worldwide (Klinge et al., 2018). Among these implants there are many different designs and layouts, with different qualities and properties. There is no complete overview of all implants used in Norway, however they are all assumed to be of good quality. The information about the quality of these implants generally come from scientific articles based on limited data or systematic overview articles (Klinge et al., 2018).

As mentioned initially, information from the use of dental implants in Norway is lacking. A potential solution for this is to implement a dental implant quality registry, which stores information about all implants set in Norway, as well as information about the procedures used, possible complications and implant removals. Such a system could be used for statistics, analytics and research, with a goal of both improving the quality of dental implants used, and discover faults in implants previously used. Analyzing the data could reveal differences in quality between different implants and clinical procedures, similar to what was accomplished in the hip and knee replacement registry (explored in section 2.2.1), where differences in results were discovered to relate to the cement used (Havelin et al., 2000). Such information could be used to recommend good quality implants at reasonable prices, as well as to guide future development of implants and the clinical procedures performed in the right direction. Research can also be used to measure the quality of different implants and clinical procedures against each other, revealing

which implants and clinical procedures are most beneficial for the patient in question. Furthermore, research can result in manufacturers being pushed to improve their products, and clinics to improve their procedures, raising the bar for each other to always strive for better care, like the diabetes registry explored in section 2.2.2 has contributed to improvements in various ways. Research and annual reports can also be used to discover batches of faulty implants, and information from the dental implant quality registry can reveal who else have implants from the faulty batch, so that the patients can be notified.

In dental implant research, the most prevalent method of collecting data is the randomized controlled trial (RCT) (Klinge et al., 2018). Some common shortcomings with this method are sample sizes that are too small, and insufficient long-term follow-up appointments. These are shortcomings that over time could be combated by establishing a dental implant quality registry. High participation rates in such a registry can also facilitate research that discovers trends, and deviances due to rare side effects.

1.2 Research questions and expected results

The aim of this master thesis project is to develop an artifact for a dental implant quality registry that can be used in the joint research project mentioned initially to build experience and knowledge in how to develop such a quality registry. With:

- Dental informatics lagging behind health informatics in general as goes for work on standardization, interoperability and the collection of data for use in research and for educational purposes (Benoit et al., 2022).
- No dental implant quality registry, nor dental health quality registries, having been developed and nor taken into use yet (this is also an indication on dental informatics lagging behind as a speciality within the broader research field of health informatics).
- The development of a health quality registry being a bit special in that

it will take years of the registry in use (collecting data) to get the acknowledging feedback on whether the right data was collected, and in the right format.

It is of uttermost importance that we do build up experience and foundational knowledge in how to develop the desired quality registry for dental implants before the real job is done. Hence, this and the other three ongoing master thesis projects.

Consequently, the expected result of this master thesis project is an artifact that can contribute in particular to the process of developing a data model and an architecture for a future dental implant quality registry. The main research questions are designed to figure out whether the proposed design (data model and architecture) and the implementation are suited to develop the required artifact. The research questions for this work are defined as follows:

- RQ1: What are the potential needs for a dental implant quality registry?
- RQ2: What requirements should a quality registry for dental implants have?
- RQ3: How should the architecture of a dental implant quality registry be designed to fill the need in research question one, and cover the requirements in research question two?
- RQ4: How should the data model for a quality registry for dental implants be designed and implemented to fill the need and requirements in Research question one and research question two, while supporting the architecture in research question three?
- RQ5: How should the data in the dental implant quality registry be made accessible, while supporting the discoveries from the previous research questions?

1.3 Research method

This project uses design science as the research method. Information systems can be considered a multi-paradigmatic research community. In a paradigmatic research community there tend to be a universal agreement on the research methods used and the phenomena of interest. In a multi paradigmatic research community, on the other hand, the phenomena of interest and methods of investigation may overlap (A. Hevner & Chatterjee, 2010). With design science research, the phenomena can be created, rather than naturally occurring (Kuhn, 1970). Designing, or creating, an artifact that does not exist is therefore at the core of design science. The design science research method is intended to fill the knowledge gap in innovative designs (A. Hevner & Chatterjee, 2010).

1.4 Outline of work

1.4.1 Domain- and data model

Initially, the work will consist of designing a domain model and data model based on the features and services required by a quality registry for dental implants. Some important concerns and considerations are described below.

What to register

To make future use of the data stored in a quality registry, it is important to define and store the right data (parameters and format) so that one can draw conclusions from future research. This will first and foremost include the necessary parameters to identify the type of implant including its production batch, as well as the chosen clinical procedure and its outcome. A list of important categories of attributes are as follows:

- Details about the implant
- Details about the patient
- Time and nature of the procedure

- Reason for procedure (Diagnosis)
- Clinic performing the procedure
- Result and complications from procedure

During a previous pilot study collecting this data from a small selection of clinics in Norway, some concerns were raised regarding the selection of data (Lygre et al., 2020). As the choice of implant is usually based on the state of the patient before the procedure, it is also important to make this clear in the artifact by including the deciding factors.

Also, from the evaluation of the pilot study (Lygre et al., 2020), users reported spending more time registering data than they would like, as some data had to be entered more than once. Especially in cases where more than one implant had been placed in a patient, data about the procedure would have to be entered several times. Furthermore, some of the users responded that it was too time consuming to fill out the form. This also resulted in lacking registrations from some of the units, due to users being unsatisfied with the composition of the form.

When considering parameter to manually enter, it is important to limit the workload wherever it is possible, in order to seamlessly integrate the registration of data into the clinicians workflow. One way to do this is to avoid multiple registrations of data.

Avoid multiple registrations of data

It would be beneficial to have an input model for the data required by a dental implant quality registry integrated with existing Electronic Dental Records (EDR). With this integration, the quality registry could acquire a lot of the information directly from the EHR (Chaussalet & Bos, 2006), in this case represented as an EDR. By doing so, the fields in the registration form that would have been duplicate entries can be filled in beforehand. By acquiring as much information as possible from EDRs, the registration work would be simpler for the users at the different units. This could also allow for

more parameters to be registered without giving the users more unnecessary work.

Obtaining consent for data collection

In Norway, it is required to have the consent from a patient to use any of the patient's data for research. A complaint raised by the research group doing the previous pilot study (Lygre et al., 2020), was that the process to obtain this mandatory consent caused delays as patients would have to read and accept a written consent form for the collection of data. This also resulted in clinics not giving the consent form to the patients until the next checkup after the procedure. By introducing a digital consent form, the patient could get the form before the procedure. This would allow them to read through and accept the form before the procedure is done, or even before the appointment whilst they are in the waiting room. A digital form could also give the patient automated reminders if they have or have not consented to the treatment. Furthermore, the digital consent forms are more likely to be properly understood by both the users and the patients (Tait & Voepel-Lewis, 2015).

Additional features

The artifact can be taken even further, by making it more interactive for the users. Making the artifact more interactive can help improve treatments by making it more accessible to the different healthcare providers. It will also be easier for researchers to access vital data related to their research on the topic. As mentioned in the paragraph about motivation, annual reports can help to improve participating unit‘s treatment plans. In addition to this, one could also keep the artifact continuously updated. This can provide updated statistics at all times, and could even serve as a tool for clinicians to make informed decisions on which implants they should recommend to their patients, facilitating for patient-centred care (Richards et al., 2015) (Stewart, 2001).

1.4.2 Security

As the artifact should be developed so that it is highly available, proper security also has to be implemented. Following the CIA-model of Confidentiality, Integrity and Availability (Samonas & Coss, 2014), the development of the artifact must have additional focus on confidentiality and integrity. Confidentiality and integrity is accomplished through the authentication and encryption implemented on the backend of the registry, further explained in section 4.4, whereas the availability aspect is advised through the services that are proposed for the various users. A general principle of the CIA model is that the more services an application offers, the higher will the demands be for mechanisms to ensure confidentiality and integrity - and vise versa.

Interface for data collection

It is important to develop an intuitive and efficient interface. To facilitate high participation rates, the interface must be designed to help the user register the correct information. It must also be simple enough for all healthcare workers to understand and use. Any information that can be acquired from a different source should not be necessary for the user to enter, as the artifact should be integrated with other systems, where information might already be stored. The program should check data fields to avoid healthcare providers entering invalid or incorrect details in the system. Development of the interface is included in another master thesis project by Elise Fiskeseth (Fiskeseth, 2022).

Structure

As mentioned in an earlier section, integration is an important part of the artifact. For the registry to be properly integrated with EDRs, it should be built with standards for representation of structured data and communication of data in mind. A potential tool is to implement the OpenEHR archetypes approach, and these archetypes should then be carefully implemented with domain knowledge governance (Garde et al., 2007).

1.4.3 Artifact

When the domain- and data models are defined, an artifact will be developed. The artifact will implement the models and make the features and services available for evaluation. The artifact will be developed as a GraphQL API connected to a SQL database. The API will expose the services designed in the domain model through use of appropriate standards.

1.4.4 Evaluation

After design and implementation, the software will be evaluated. This will consist of evaluating if the implementation can solve the tasks given in the domain model, and if the domain model can solve the problems posed above. Based on the evaluation, potential new iterations to combat any weaknesses discovered will be described.

1.4.5 Project plan

The project plan is to build and evaluate an artifact of a registry for the quality of dental implants. A data model, an architecture and an artifact will be developed. The artifact will then be evaluated by a small selection of clinicians and experts. This will be done through several iterations, where improvements will be made, or additions will be added, for each iteration.

1.5 Outline of the thesis

1.5.1 Chapter 1 Introduction

Chapter one introduces the research project and the focus of this master's thesis. It introduces the problem to be solved, and the research questions that will be explored. This is followed by a presentation the research method that will be used and an outline of the work that is done.

1.5.2 Chapter 2 Background

This chapter gives some background information about dental implants and quality registries. A few existing registries are explored, and some benefits they realised are presented. Other work where a quality registry was integrated with an Electronic Health Record (EHR) is also presented.

1.5.3 Chapter 3 Design and method

The third chapter focuses on the design and research method. The research method is further detailed, and explanations of how it is used in this master's thesis is given. The design process is shown, where several iterations to develop the design, and what the iterations achieved is detailed. Finally, the artifact developed is presented, and some of the most significant parts are described.

1.5.4 Chapter 4 Implementation

In this chapter, the implementation of the artifact is described. The frameworks used to build the artifact is shown, and its architecture is presented and explained. Then, the security of the artifact is explained followed by an analysis of its performance, using several types of tests.

1.5.5 Chapter 5 Evaluation

In chapter five the artifact is evaluated. First, the data model, data access and the architecture is briefly evaluated by referring to results from previous chapters. Next, an interview is used as a qualitative evaluation of the artifact. A summary of the results from the interview is given.

1.5.6 Chapter 6 Discussion

A discussion of how this master's thesis answers the research questions posed in chapter one opens the sixth chapter. This is followed by reflections on the

research method as well as the tools and frameworks used. Finally, some of the limitations this master's thesis project faced are briefly explained.

1.5.7 Chapter 7 Conclusion and Further Work

This chapter gives a conclusion to the thesis, followed by suggestions for further work. Earlier chapters have introduced ideas that were outside of the scope of this thesis, and were not implemented in the artifact. These are revisited and proposed for further work.

1.5.8 Appendix

The appendix contains some data and results listed below.

- A list of the recorded parameters with a list of options for each parameter.
- A list of table value relations.
- Results from each performance test.
- A transcript of the the interview.

1.6 Chapter Summary

Information about dental implants in Norway is lacking, which means there is a need for more information. This master's thesis utilizes design science to explore the use of a dental implant quality registry to fill this need, by developing a data model and an architecture, and implementing them in an artifact.

Chapter 2

Background

2.1 Dental implants

Dental implants were invented almost 70 years ago by Swedish orthopedic surgeon Per-Ingvar Brånemark. These implants have since become the standard of care for prosthetic replacement of missing teeth (Bautista, 2019). A dental implant is a fixture surgically placed into the jawbone, where it fuses to the bone over time. This process is called «osseointegration» and is often done with titanium for the best bone integration. The dental implant serves as the tooth's root, and an artificial tooth can be placed on top. This is done to ensure the most amount of function and esthetic is restored for the patient.

There is no general overview of all implants set in Norway, however some are registered when applying for a Helfo refund. Some implants are covered, including rare medical states, anomalies in mouth and jaw, and tumors limited to the mouth and head region (Lovdata, 2014). Among these implants, between 2014 and 2017, more than 14000 dental implants were placed each year, which qualified for a Helfo refund in Norway. This resulted in refunds for around 100 million NOK each year (Lie et al., 2019). In Europe, the dental implant market is estimated to be 5.5-6 million implants a year, while it's assessed to be around 12-18 million implants worldwide (Klinge et al.,

2018).

There are many different designs and layouts among these implants, with different qualities and properties. Some of these properties include size and different grip strengths. As mentioned, there is no complete overview of all implants and treatments used in Norway. It varies from unit to unit what specific implants they use and how they treat their patients, but they are all presumed to be good. The information about the quality of these implants, and treatments, generally come from scientific articles based on limited data or systematic overview articles (Lie et al., 2019).

Figure 2.1: Different dental implants (Klinge et al., 2018).

2.2 Quality registries

A medical quality registry is a registry that collects information on the assessment, treatment, and follow-up of patients within defined groups. The primary goal of a quality registry is to contribute to better the care plan for

the patient and reduce variation in health care services and treatment quality between healthcare facilities (Alter, 1978). A medical quality registry serves as a database that can be used for statistics and research regarding patients' conditions and treatment plans. Through soft regulatory policies, high-quality registries can be made that accentuate healthcare professionals' own improvements in their patient care (Levay, 2016). There are 61 quality registries related to healthcare in Norway (kvalitetsregistre, 2022). A few of them are listed below. They were specifically chosen as there is research based on each of them, which shows several benefits realized by the registries. These benefits are also described below.

2.2.1 Hip and knee replacement quality registry

By looking at information collected in the hip and knee replacement registry from 1987 to 2000 (Havelin et al., 2000), a study could derive several useful conclusions that could benefit future treatment. The study revealed, among other findings, that the choice of cement had a great impact on the durability of an implant. The study also showed a higher tendency for revisions in uncemented prostheses. This was, however, not statistically significant. This would also suggest that one can safely choose the cheaper cemented prosthesis over the uncemented options, without it having any negative consequences for the patient (Havelin et al., 2000).

2.2.2 The Norwegian diabetic registry for adults

The Norwegian diabetic registry sends out annual reports on the status of a unit's patients' hba1c values, compared to a mean value of all diabetic patients in the country. Through these results, all units who score below the average in any field will know which areas they have to improve on whilst pushing the average forward. This will secure better treatment, and also make sure patients get better care (Cooper et al., 2013) (kvalitetsregistre, 2019).

2.2.3 Cleft lip and palate registry

A study used data from the Norwegian cleft lip and palate registry, "Norsk Kvalitetsregister for leppe-kjeve-ganespalte", to compile information on the types slit found over time (Krogstadmo & Vatnaland, 2016). The registry started recording children who were born with cleft, lip, and palate slits in 2011 (kvalitetsregistre, 2021a). The study could then use the registry's latest report, in 2016, to use statistics from a large set of data, improving the accuracy of the data presented.

2.2.4 National quality registry for hand surgery

The Swedish national quality registry for hand surgery implemented a voluntary follow-up workflow where question forms were given to participants before the surgery and at different times after the surgery to record the short and long-term effects of the procedure. A review of the registry proposes that the patient registered data can be used to compare treatment from the patient's perspective and better inform the patient prior to the surgery (Arner, 2016).

2.3 Other work

A proof-of-concept architecture for a network-based Learning Health System has been developed to integrate an existing registry with several Electronic Health Record (EHR) systems. The project sought to automate a series of reports and demonstrate the use of observational registry data for comparative effectiveness research (Marsolo et al., 2015). Within the project, they worked with three leading EHR systems to create data collection forms within the EHR systems. Figure 2.2 presents the architecture used in the article. This architecture has a lot in common with what the dental implant quality registry should accomplish. A weakness with the approach taken in the Marsolo project is the requirement for each registry to work with each EHR vendor for integration. The aim of the dental implant quality registry, however, is

to serve as a standalone artifact. As a standalone artifact it can provide direct data entry through a user interface, as well as an API for external entities such as EDRs to integrate with. The artifact will record data that is not normally registered in the EDRs, but can have meaningful information about the implants and procedures. The architecture is explained in section 4.2.

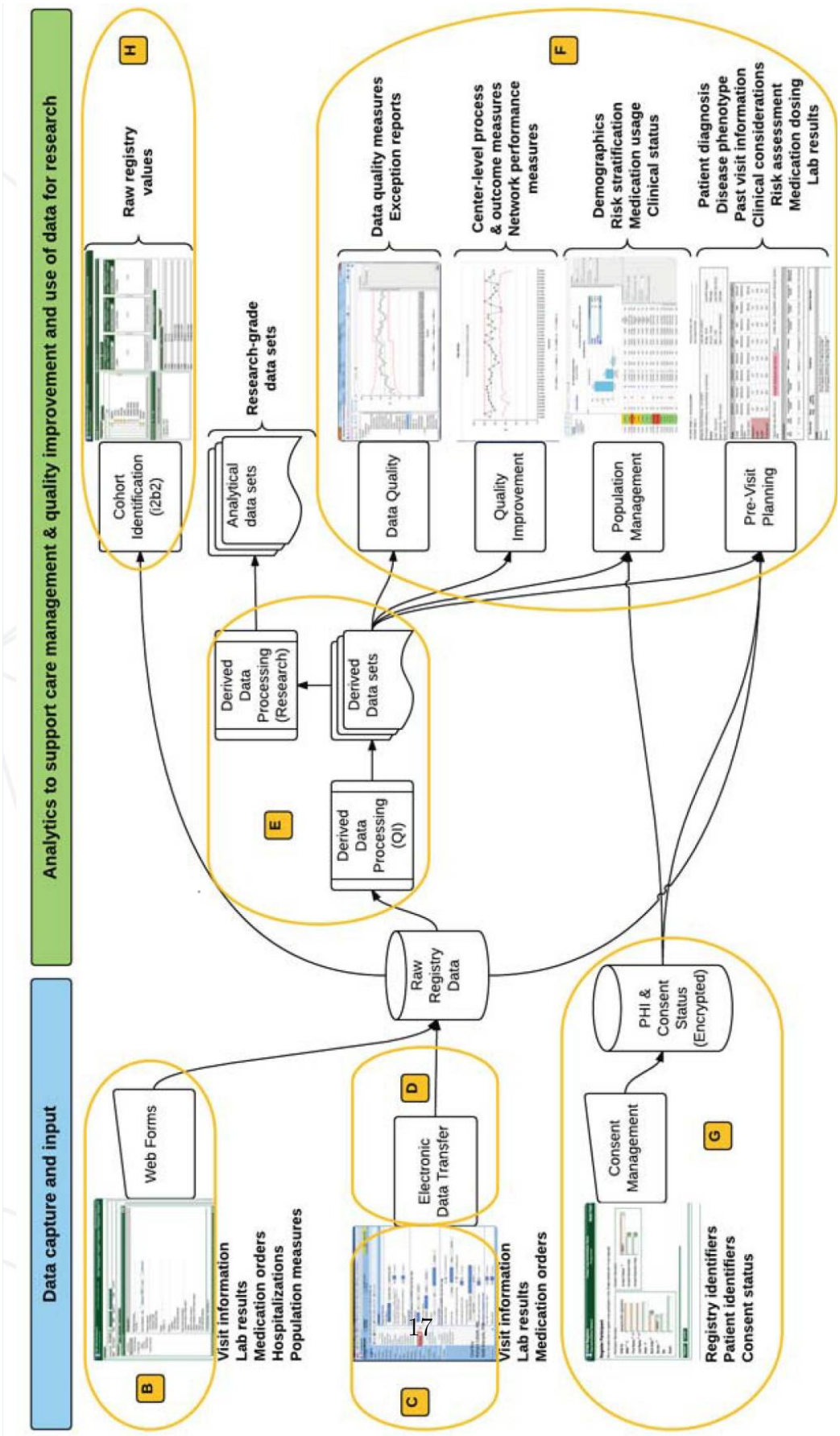

Figure 2.2: Functional Architecture of the Learning Health System presented (Marsolo et al., 2015).

Chapter 3

Design and method

This section will describe the process of designing, as well as the development of an artifact. The artifact and some use cases will also be presented. Firstly, the project will be connected to the research method used, and an overview of how this research method is accomplished is given.

3.1 Research method

As mentioned in an earlier section, design science is used as the research method in this project. The design science process can be divided into six objectives, which will be elaborated on below (Pfeffers et al., 2006).

3.1.1 Problem identification and motivation

The problem identification is intended to define and justify the value of a solution. This is done to motivate the researchers and audience to pursue the solution and accept the results. It also helps the audience to understand the reasoning for the researchers' understanding of the problem. For the dental implant quality registry, this is done in the Motivation section of the introduction.

3.1.2 Objectives of a solution

At this point, the objectives of a solution must be inferred. These can be quantitative and/or qualitative objectives that are rationally inferred from the problem specification. In the section 2.3, it is shown that no currently available implementation of a system for dental implants exists. There are, however, several other registries available within other domains, but these show some limitations. The solution and its objectives are presented later in this section.

3.1.3 Design and development

This means creating the artifact, which includes determining the artifact's desired functionality and designing its architecture. The pre-evaluation artifact is described in section 3.5, wheras the iterations leading up to the pre-evaluation artifact are described in section 3.2.2. In addition to this, the whole implementation, its data model, and architecture are described in chapter 4.

3.1.4 Demonstration

Demonstration of the artifacts' efficacy to solve the problem. This can be thorough use in experiments, simulation, case studies, proof, or other appropriate activities. For the dental implant quality registry, the optimal way to demonstrate its efficacy would be to gather data for an extended time period and evaluate the gathered data. However, this would be too time-consuming as an initial demonstration, and instead, the artifact has been presented to some clinicians and experts who have given feedback through several iterations. The iterations are described in 3.2.2, and the artifact is also evaluated in chapter 5.

3.1.5 Evaluation

The evaluation involves observing and measuring the efficacy of the artifact to support a solution to the problem. This means comparing the objectives of a solution to the observed results from the use of the artifact in a demonstration. There have been many smaller evaluations for each iteration, primarily focusing on the data model. The evaluation is described in chapter 5.

3.1.6 Communication

The sixth activity is to communicate the problem and its importance. This means explaining the artifact, its utility, novelty, and design. For the backend of the dental implant quality registry, that is done through this thesis. Other parts, such as the frontend interface, are described in a different master's thesis by Elise Fiskeseth (Fiskeseth, 2022).

3.2 Design process

During the design and development process, an iterative approach was taken. For each iteration, a meeting was held, where feedback was given. Based on the feedback, the design and implementation were adjusted by going back to step three on the model described in section 3.1. Various researchers with different backgrounds were present at the meetings. Who attended the meetings varied from meeting to meeting, but many were present at all meetings. Firstly, there were master's students who were tasked with designing and implementing different parts of the dental implant quality registry artifact. This was primarily the backend and the frontend used by clinics to register implants into the registry. The student's supervisors were also present, who have experience in health informatics. Several experts from NORCE and Odontology/UiB were also present. Some with expertise in dental care, and others with experience from other quality registries. In addition to these researchers, some other clinicians have also been present. As the users of the dental implant quality registry, the clinicians' feedback

was essential. At the meetings, the state of the artifact was usually presented first. This was done by showing the frontend of the artifact. Initially, different models were also shown; however, these did not result in as high-quality feedback as the frontend did. With that in mind, the frontend was the focus of all the following meetings. Through the frontend, the artifact was demonstrated to evaluate the data model, the structure and experience of the interface, and the services provided. The meetings were held with a varying frequency, with some meetings held one week apart and others up to a month.

3.2.1 Project goals, requirements and solution objectives

The project aims to design and create an artifact that can serve as a quality registry for dental implants, which can be used to collect data on implants used in Norway to make statistical analysis on the data. The scope of this master thesis project is limited to the backend of the artifact, with a focus on the development of the data model and architecture, as well as a structured API to access the stored data. The system requires a database to store collected data, and an application that receives and manages implant-, procedure- and patient reported data. This system needs to be available at all times over the internet, both for data entry by clinicians and researchers who want to access reports and statistics.

3.2.2 Iterations

A new iteration of the artifact was developed, demonstrated, and evaluated between each meeting. Every meeting resulted in feedback that was used to improve some aspects of the artifact. Primarily the improvements were of the following types:

- Layout and structure of the interface
- Recorded parameters and their options

- The way recorded parameters should relate to other data

The feedback received in each meeting demanded different amounts of work, and some were more specific to the frontend layout without requiring significant alterations of the data model. For that reason, the iterations described below will not correspond 1:1 with meetings but will be compiled to iterations where notable changes to the data model were made.

First iteration

For the first iteration, the parameters recorded for the pilot study (Lygre et al., 2020) were implemented in a data model. The pilot study used a formfor data input which is shown in figure 3.1. Based on this form, an initial data model was developed.

Second iteration

From a meeting where some participants of the pilot study were present, some concerns with the form used in the pilot were raised. Most prominent was the fact that in cases where more than one implant would be placed, information that is common for each implant in the procedure had to be entered several times. Another concern was with the order in which information would be entered. Based on this, an initial suggestion for the data model was suggested. This model is presented in figure 3.2. In this model, there are two main tables: "Insertion" which is for each procedure where implants are inserted, and "Removal" which is for each procedure where implants are removed. In each of these procedures, there can be several implants. Therefore, these implants are in a separate table, with reference to an insertion or removal. This structure is intended to allow separation between data that is specific for each implant and common for procedures where several implants are set. This means that the data model supports for a better API to tackle the problem of re-entering data for each implant.

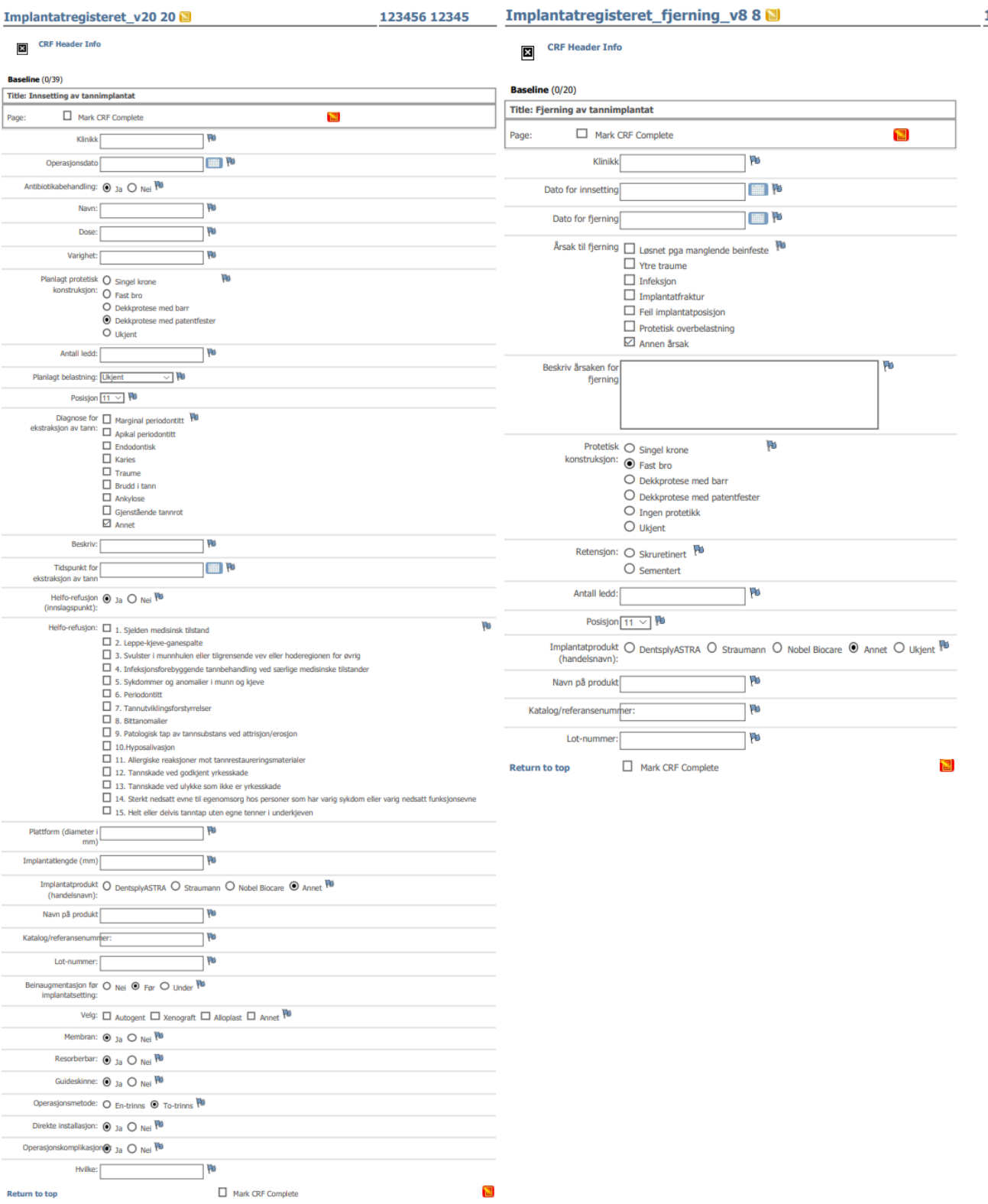

Figure 3.1: Form used in the pilot study (Lygre et al., 2020).

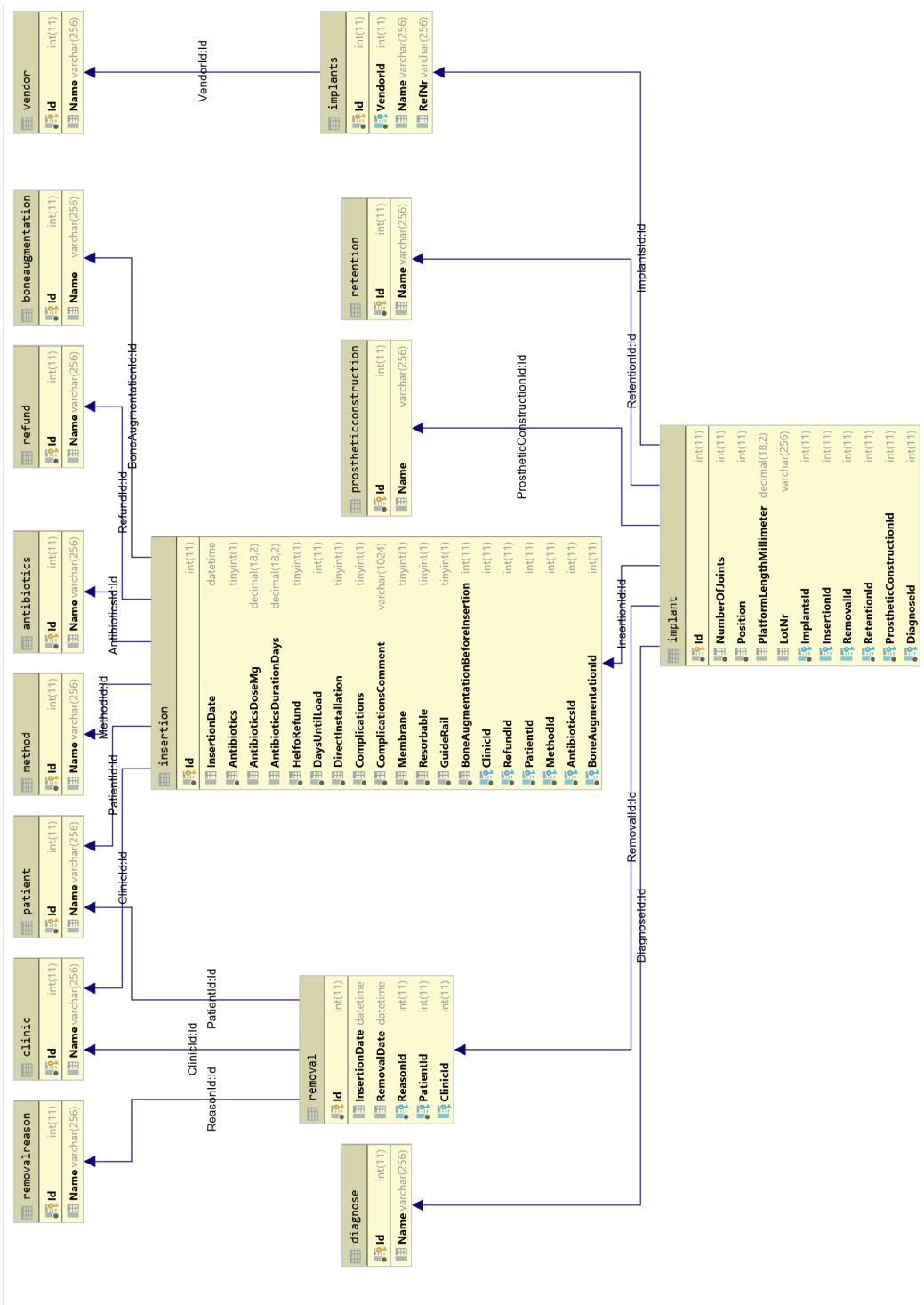

Figure 3.2: Initial data model suggestion.

Third iteration

To not be a burden on the clinician, it is important to keep the data entry process into the artifact as short and simple as possible. At the same time, all important parameters have to be included to provide for meaningful research on the registered data. For the third iteration, some changes were made on which data to store. Fields such as what kind of refund, if any, patients receive from Helfo can be retrieved from Helfo in cases where those are relevant for research, and will therefore be removed from the data model. Previously unrecorded values about the patient, such as the bone density and volume were added. This is done because clinicians may choose different implants depending on the patient's condition. They are recorded with the Lekholm/Zarb Index. The Lekholm/Zarb Index evaluates the bone quality and quantity (Albrektsson & Wennerberg, 2005). Other values were also altered and can be seen in the data model (figure 3.5).

Fourth iteration

Several new parameters have to be recorded correctly with the new data model. In this iteration, the selectable options were discussed for each question. Some values were suggested by different attendees of the meeting, and the optimal options were chosen based on how much information they added and how they would be experienced by clinicians. Some information may be difficult to know accurately, and too many options could lead to confusion. With this in mind, options were chosen in order to cover all common scenarios with as few options as possible. Among other things, this meant considering the use of collective terms and limiting parameters such as stability to only record 1. Good stability, and 2. Reduced stability. Consistency in options was also discussed, which led to all options that describe a time span to be measured in weeks. A list of final options for each question can be found in appendix 8.1.

Fifth iteration

Values recorded in the pilot study showed that almost 40% of the included patients had two or more implants. There was even one recorded patient with 12 implants (Lygre et al., 2020). In cases where several teeth are missing or being replaced, one or more bridges can be used (InformedHealth.org, 2006). One or more implants can fasten one bridge. This means that one procedure can be done for one or more crowns or bridges, and each bridge can have one or more implants. A table for bridges is therefore added to the data model and can be seen in the final data model shown in figure 3.5. The bridges are added as "planned prosthetic constructions" as they are set in a different procedure than the implants. In some rare cases, they can be different from the prosthetic construction eventually used. As it is just a planned prosthetic construction, the table just stores its start and end positions, along with a value to determine if it is a plate prosthetic or a normal bridge. It is a crown if the start and end positions are the same.

Sixth iteration

The full structure of the data model did not come across as easily by showing the frontend interface, so a table was used to discuss and determine how often each parameter in the artifact should be recorded for each implant. The table was split into three columns: "Implant", "Prosthetic construction" and "Procedure". Each row was then filled with parameters that needed to be entered once for each instance of that column. Figure 3.3 shows the agreed upon table. This required several changes to the data model, which primarily meant moving fields from the "Insertion" table into the "Implant" table. The table mentioned above can also be seen in appendix 8.2.

Seventh iteration

For this iteration, the antibiotics field was evaluated to be insufficient. At the meeting for this iteration, the clinician present presented cases where antibiotics were used pre-, per-, and post- procedure. The updated data

Figure 3.3: Table of relation.

model after the sixth iteration supports only one of these cases at a time. To support several types of antibiotics use, the data model was altered with an antibiotics table which has a many-to-many relationship with the procedure. By doing this, an instance of antibiotics can be used to represent each use of antibiotics. The new table is shown in figure 3.4.

3.3 Description of artifact - The dental implant quality registry

The artifact, as seen from the outside, is a GraphQL API. The API is split into queries and mutations, where queries are used to request data, and mutations are used to add or alter data. The primary queries and mutations will be listed below. These are structured as graphs, meaning that the API consumer has a lot of control and can request only the data they need. For example, the options query listed below can either be used once to request all the data needed for the entire insertion and removal forms, or only the necessary information can be queried at the point where it is needed. The full documentation can be found at /ui/playground.

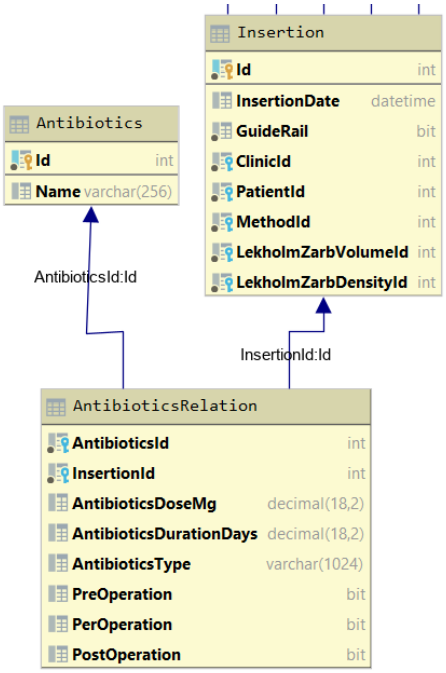

Figure 3.4: The new antibiotics relational table.

3.3.1 Queries

The queries are reached by sending an HTTP POST request to /graphql. The request must contain a query variable that contains the GraphQL query.

Options

The ”options” query is used to get all options for the parameters recorded in the artifact. These are structured as an object with an id and a name. The name is a text shown by the frontend, and the id is stored with the registered implants.

Insertions

The insertion query lists procedures where implants have been set, with its attached implants and planned prosthetic constructions accessible as a graph.

Removals

The removal query lists procedures where implants have been removed, along with the implants that were removed.

PatientInfo

This query allows access to a patient and contains a graph for all implants and graphs for only inserted implants and removed implants. It also contains a graph for insertion procedures and removal procedures. Finally, it also contains a graph of all teeth positions with a history of implants placed in those positions.

3.3.2 Mutations

The mutations are sent as an HTTP POST request, much like the queries. In this case, the query variable will start with "mutation", and the mutation will follow using GraphQL syntax. Variables are also added within an object called variables, as regular JSON variables, objects, or lists.

CreateInsertion

The "createInsertion" mutation is used to add a procedure where implants are set into the artifact. The mutation is sent with a variable containing the details for the procedure and a list of all prosthetic constructions and implants together with their respective details.

CreateRemoval

The createRemoval mutation is used to add a procedure where implants are removed. This mutation will contain a variable with details for the procedure and a list of implants, including their details.

3.4 Data Model

The final data model is shown in figure 3.5. This is the underlying structure that is accessible through the GraphQL API. The data model was developed over several iterations, with evaluations, which are described in section 3.2.2.

3.5 Description of artifact - Question Forms

In addition to recording data on the implants and the clinical procedures, the short- and long-term effects experienced by the patients would also be beneficial to record. This was discussed in some of the early meetings concerning the iterations of the artifact. Therefore the GraphQL API contains a series of queries and mutations that create, manage and get question forms that can be sent to patients. Unfortunately, implementation of this feature on the frontend has not been done, as it had to be set outside the scope of priority within the master thesis project time frame. As a consequence, it has neither been included in the iterations where experts evaluated the artifact. Instead, the implementation of the question forms has been made very flexible and can be managed by an administrator. The question forms are intended to be sent to patients at intervals decided by artifact administrators, and are left for future work. The question forms are structured so that a form can be made, and questions can be added to the forms, then these forms can be sent to patients, and their responses can be recorded in the artifact.

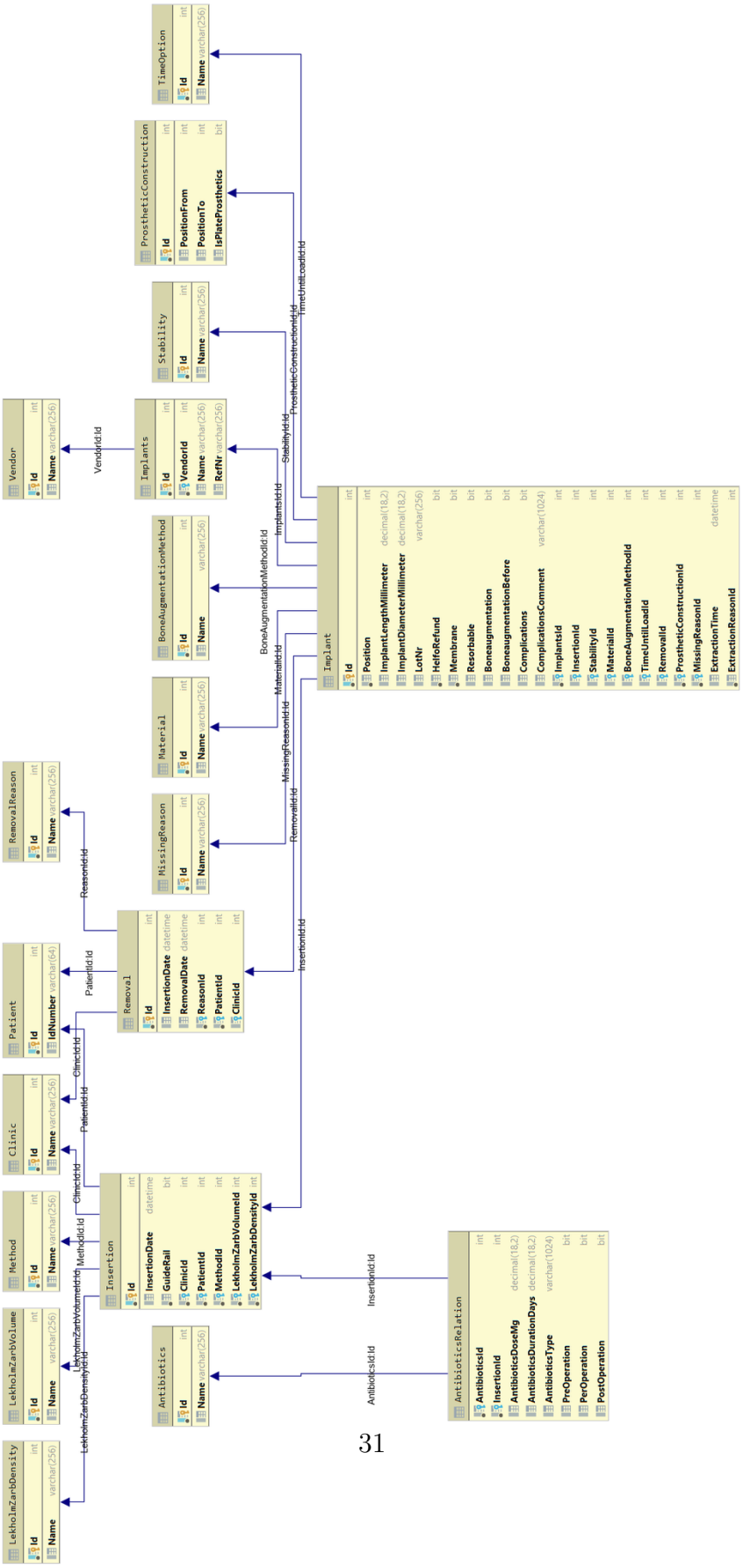

Figure 3.5: The data model.

3.5.1 The question form

Any number of forms can be created. For each form, any number of questions can be added. The type of answer can be controlled by the creator and can accept number answers or multiple choice answers for the questions. The question can also be set to accept several options as answers. The number of questions can be limited to an upper and lower limit. Free text answers are not supported to discourage questions that give little or difficultly measurable answers, however, this can be added in the future if required.

3.5.2 Mutations and queries for question forms

Create form

A form can be created using the "createQuestionForm" mutation. This creates an empty form with a name and a description. Some control variables can be set, such as "retryCount", "sendAfterDays", and "retryAfterDays". These variables are intended to be used to manage the automatic sending of question forms to patients.

Create question

The "createFormQuestion" mutation creates a question for a form. This will contain the question and a description of the question. It can also set the question to allow more than one answer. If the question takes a number as its answer, then one option has to be made for the question.

Create option

The "createQuestionOption" mutation can be used to add options to a question. For questions with a number answer, the "isNumber" value must be true, and a lower and upper limit can be set. If the answer is multiple choice, then as many options as you like can be added.

Let a patient answer the question form

"sendQuestionForm" as a mutation that creates an instance of a question form. It will connect a form with a patient and an implant. When this instance is made, the form can be sent to the patient, and the answer can be recorded using the "answerQuestion" mutation. This creates an answer instance for each option that the patient answers.

Get forms

The "questionForms" query can be used to get all the question forms. It can be accessed as a graph, which means that the forms will contain a list of their questions, and the questions will contain a list of their options.

Chapter 4

Implementation

To make an easily accessible artifact, the application was implemented as a web application. This allows for cross-platform access through any device that has a modern web browser. It also facilitates the use of modern frameworks and well-established protocols to interface with the application. The programming language used is C#.

This chapter explains what tools and frameworks were used to implement the artifact. Explanations and models of how the artifact was implemented is also shown, and results from tests of the artifacts performance is shown.

4.1 Framework

The backend of the application was implemented using the ASP .NET 5 framework, which is the latest stable release from Microsoft at the time of writing (Microsoft, 2021a). .NET is a platform made up of tools, libraries, and programming languages for the development of a wide variety of applications. ASP.NET is a set of tools and libraries specifically aimed at making web applications and is an extension of the .NET platform (Microsoft, 2021c).

4.1.1 Data store

The backend uses a Microsoft SQL Server as its data store. A SQL data store was chosen because the data collected by the artifact are all related, and thus a relational database fits well. An object mapper called Dapper was used to simplify the mapping of database rows to C# objects. Dapper automates the mapping of objects while still letting the developer write the SQL queries. It also adds some security as it provides protection against SQL Injection attacks (Dapper, 2021).

4.1.2 GraphQL API

The backend of the artifact can be accessed through a GraphQL API. GraphQL is a query language for APIs. It provides a graph of all the data in the API and gives the client the ability only to query the data that they want (GraphQL, 2021). For the artifact, GraphQL-dotNET was implemented. GraphQL-dotNET is a library that implements a GraphQL API in .NET. It uses an HTTP POST request to define a query or mutation. This query accesses predefined query objects that define how data should be retrieved.

GraphQL Playground

GraphQL Playground was also implemented. GraphQL playground provides a powerful GUI that clients can use to write queries directly and access complete documentation for the API. The playground also provides auto-completion as queries are written, which also makes it an excellent tool for API consumers.

The documentation is shown in figure 4.2.

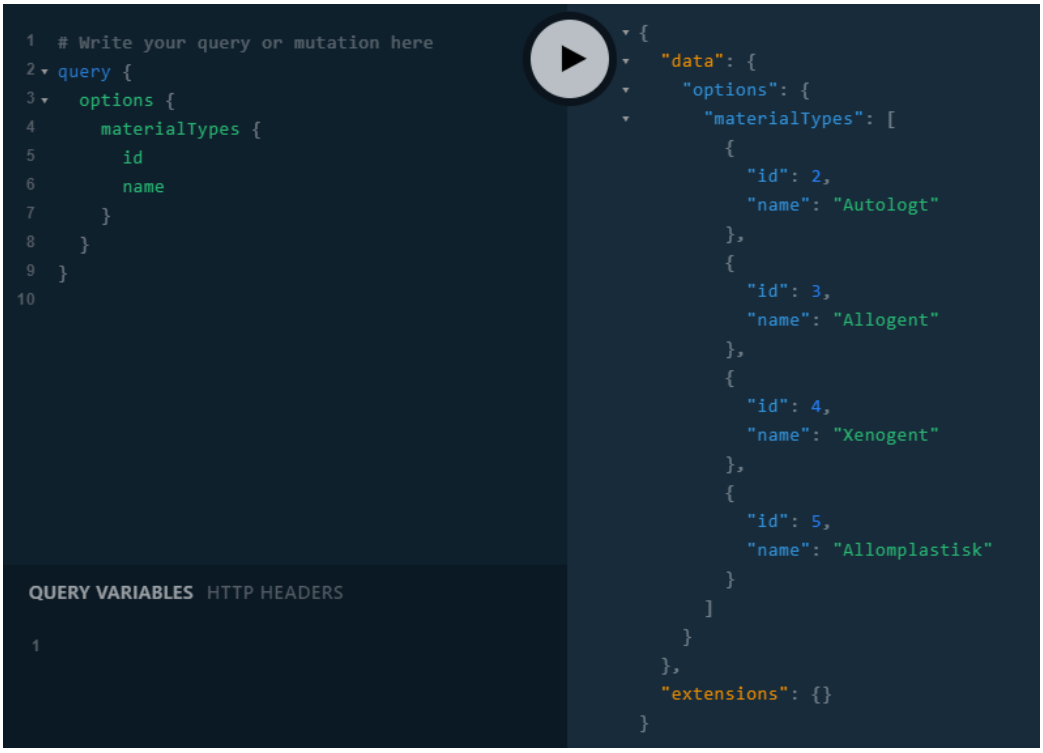

Figure 4.1: GUI to write queries directly to the API.

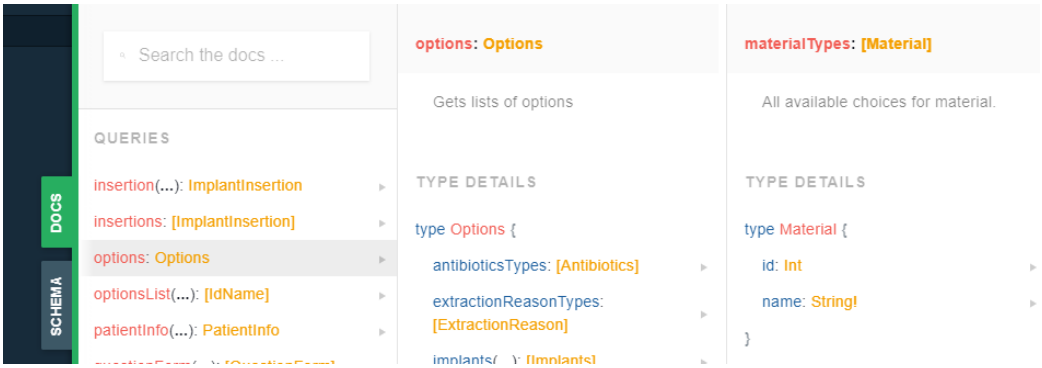

Figure 4.2: API documentation provided in the GraphQL playground.

4.2 Application architecture

The artifact is implemented using the MVC pattern (Deacon, 2009). The view is implemented as a React view, where the controller is the GraphQL API, and the model is stored in the database. This thesis will limit its scope to the core data collection functionality marked with blue in figure 4.5.

This consists of a backend API connected to a database and a view. The functionality focuses on the dental clinics that will enter data from implant cases into the system. Furthermore, figure 4.5 shows a suggestion of other use cases that would be beneficial to implement.

4.2.1 Software architecture framework

The development of the architecture loosely follows the Architectural Development Model (ADM) from The Open Group Architecture Framework (TOGAF). The framework is good for implementing vast systems in large companies (Reselman, 2020). It is well accepted within the community of TOGAF users that the framework does not have to be followed strictly, as many of its recommendations are not always found applicable (Kotusev, 2016). With this in mind, the architecture of the artifact is developed while going through the eight phases of the TOGAF framework shown in figure 4.3. Although different teams would be responsible for different phases of development in a large system, the architecture of the artifact is developed in iterations where domain experts are regularly consulted. During the scope of this thesis, the benefits of TOGAF are not realized but can instead be further explored in future work.

4.2.2 Software architecture pattern

The Model-View-Controller (MVC) architecture was conceptualized in the '70s and is still very relevant. It is intended as a mechanism to separate the presentation from the application (Deacon, 2009). The MVC-pattern is shown in figure 4.4, and the architecture of the artifact is shown in figure 4.5. MVC provides a natural separation for the artifact, as the artifact will require several presentation applications to cover the entire domain, as shown in figure 4.5.

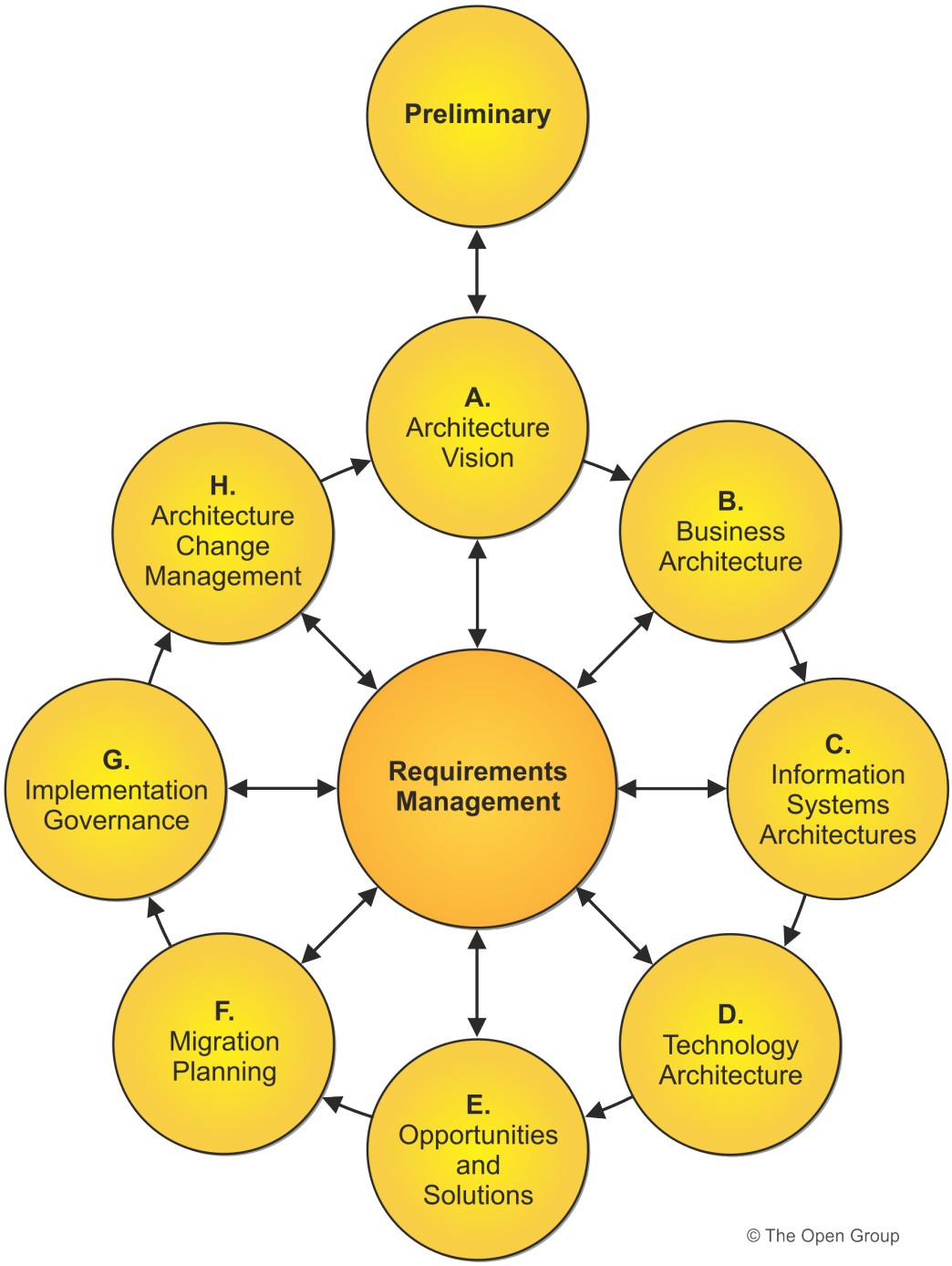

Figure 4.3: The phases of the Architectural Development Model. Figure from The Open Group (TOGAF, 2021).

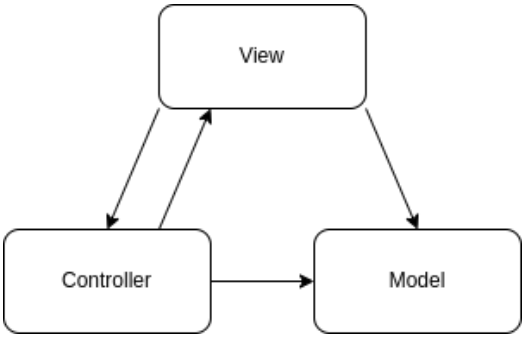

Figure 4.4: Model-View-Controller pattern.

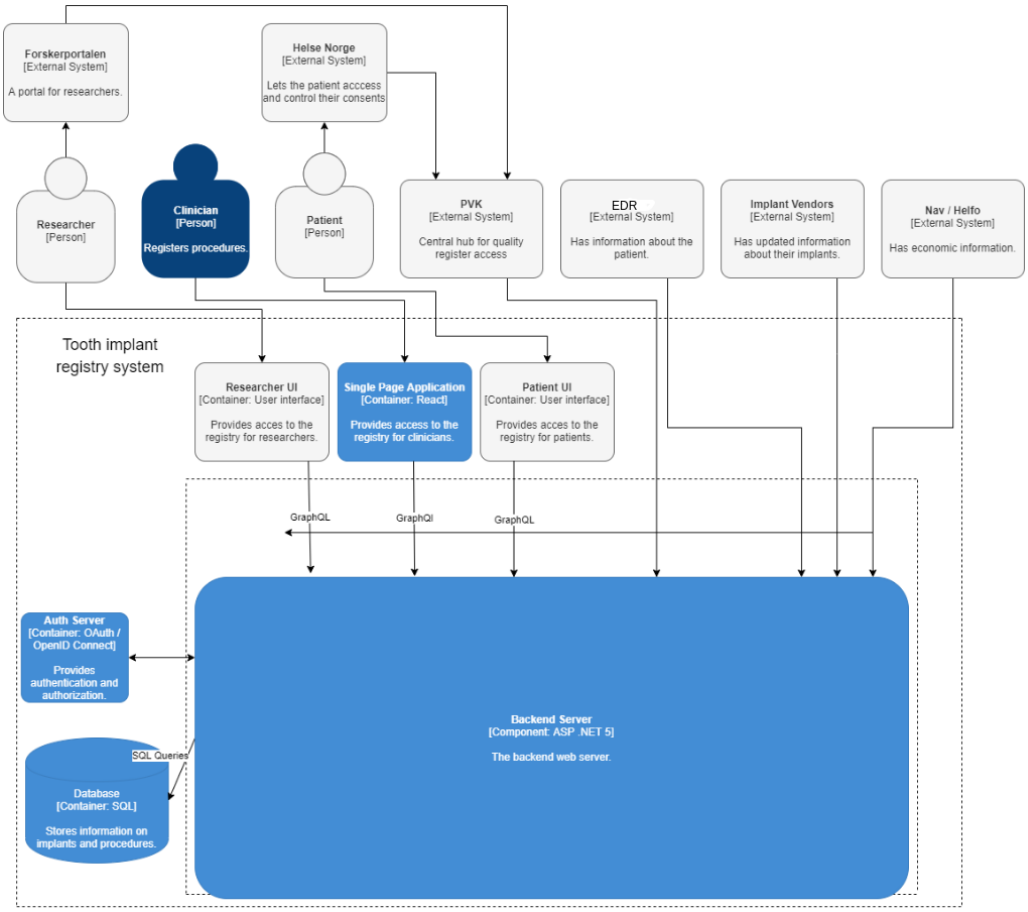

Figure 4.5: General architecture suggestion.

4.2.3 Clinicians

Clinicians enter data into the artifact after performing a procedure that involves dental implants. This is done through a view developed by Elise Fiskeseth (Fiskeseth, 2022), documented in her master's thesis (Fiskeseth, 2022). The data entered by the clinicians provide the foundation for the artifact.

4.2.4 Patients

As described in the strategy and execution plan from the national service environment for medical quality registries (kvalitetsregistre, 2020), there is a need for increased patient and user participation, both for reporting of patient reported outcome (Enden et al., 2018) data, and access to their own data and new registry services. For this artifact, that means that the patients should also have access to the artifact, both for patient reported outcome services related to their cases as well as analytics and educational services to support patient-centered care (Richards et al., 2015). This can be supported through a separate view for patients or through Helse Norge. By providing access through Helse Norge, they can aggregate access to several registries from one place.

4.2.5 Researchers

A view for researchers should be developed so that reports and statistics can be made available. Direct access to the API should also be made available. This can be done through the GraphQL playground. Similar as for patients, access for researchers can also be accomplished through a common portal. Unit (Direktoratet for IKT og fellestjenester i høyere utdanning og forskning) discusses a portal for researches (Bjerde et al., 2019). A platform called "Helseanalyseplattformen" (E-Helse, 2022) is in development, and work to integrate with their solutions should be considered in future work.

4.2.6 Electronic Dental Record Systems

A lot of the data relevant for research in the artifact may already exist in an Electronic Dental Record (EDR). This may result in clinicians having to enter the same data into both the EDR and the artifact. To avoid this, the artifact should obtain data from EDRs.

4.2.7 Implant Vendors

In order to register an implant into the artifact, three things have to be entered: The vendor, a LOT number, and a reference number. Entering these values can be tedious, and automation is possible. By acquiring a catalog of implants from the vendors, a full list of implants can be searched and even display information and statistics. Furthermore, barcodes or QR codes and can be linked to the artifact and fully automate data entry for the implants identifying information.

4.2.8 Other systems

Access to even more systems such as NAV and Helfo can provide even more information, and research on how treatment affects the patient can be done more easily.

4.3 Backend architecture

The backend of the artifact is implemented based on the layered model and primarily consists of three layers, separated by interfaces using inverse dependencies. As shown in figure 4.6, the data access layer is at the bottom. This is an abstraction for the data access. The data access layer provides methods for retrieving and manipulating data. The next layer is the domain layer. This is where the business logic happens. For the artifact, this primarily means that data are being mapped and validated. This is also where services revolving around the domain logic will reside. On the top layer, there is a web API. This is implemented as the GraphQL API mentioned earlier.

4.3.1 Using the Layered Model

The layered model approach requires the application to be split into several layers, providing an abstraction to the lower layer. The project is split into three layers: presentation, domain, and data source. This is modeled after the concept as presented by Martin Fowler in his book "Patterns of Enterprise Application Architecture" (Fowler, 2012). Some of the main advantages achieved by the layers are abstractions, separation of concerns, and modularity. The abstractions allow higher-level layers to implement their functionality without knowledge of the lower layers' functionality. The separation of concerns and modularity of the artifact allows for changes to parts of the system without affecting or requiring changes to the rest of the artifact.

Weakness with the layered model

A weakness of the layered model is the domain's dependency on the data access layer. This means that the data layer is at the heart of the artifact instead of the domain. At the heart of most software is the domain-related problems that the artifact is intended to solve, which encourages a domain-driven design process (Evans & Evans, 2004). With the domain at the heart of the software, an architecture where the domain has no dependencies may present a better solution. A Clean architecture, which builds on the hexagonal architecture invented by Alistair Cockburn, would achieve this (Martin et al., 2018). The primary purpose of the artifact, is storage of data, while the domain of the artifact facilitates management and access to the data. For that reason, the layered model was chosen.

4.3.2 Data model

The data model represents the data stored about implants, the patients, and the clinical procedures where implants are inserted or removed. The data model has gone through many iterations, where the registered variables have been evaluated by experts and clinicians who suggested new parameters, the necessity of existing parameters, as well as answer options.

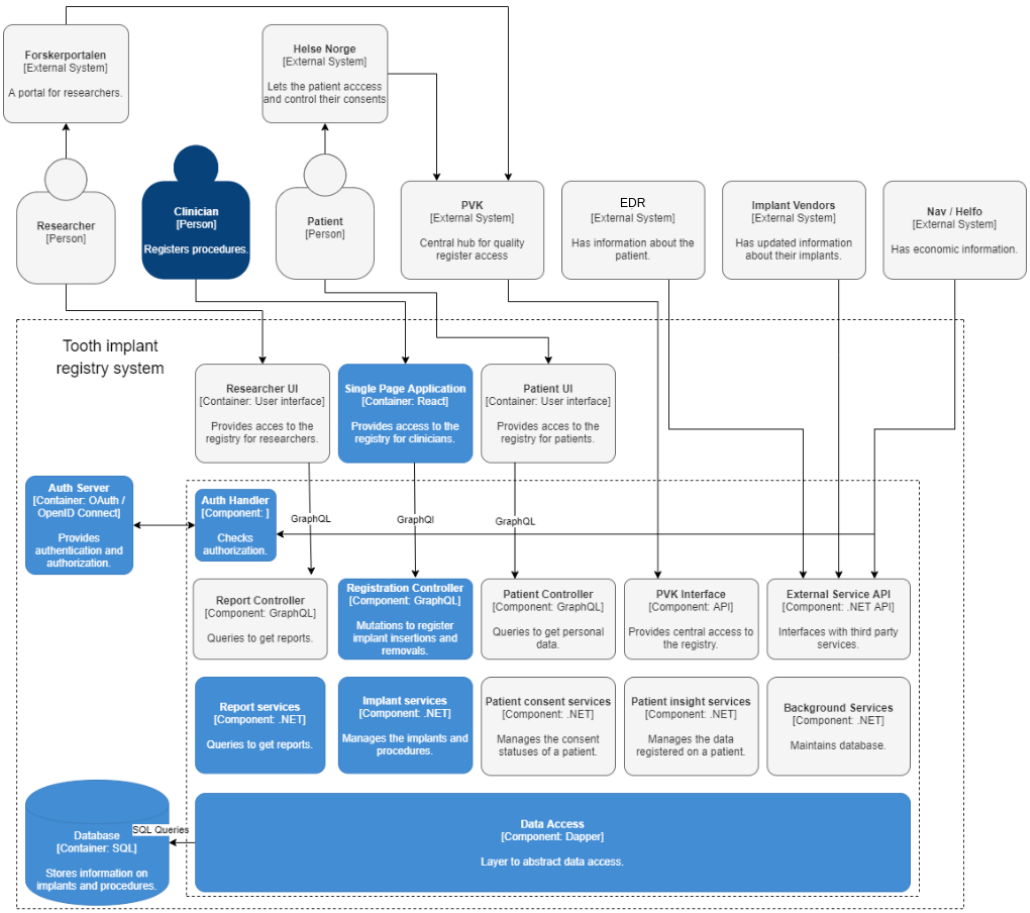

Figure 4.6: General architecture suggestion including backend architecture.

Implants, Insertion and Removal

The data model is shown in figure 3.5. There are mainly three interesting tables: Insertion, Removal, and Implant. The implant table holds information on all inserted and removed implants, which is the artifact's centerpiece. All of the implants are connected to either an insertion or removal. The Insertion table holds information about each procedure. There may be several implants set during one procedure, which means that there is a one-to-many relationship between insertions and implants. The same goes for the Removal table. The Removal table represents a procedure where one or more implants are removed and stores some values common for all implants.

Planned Prosthetic construction

Prosthetic construction represents a planned construction that will be mounted on top of the implant. The planned prosthetic construction may be relevant for the choice of implants but could, in some cases, be different from the construction used. There can be more than one implant inserted for each planned prosthetic construction, and there can even be implants inserted for several planned prosthetic constructions in the procedure. A table for prosthetic construction is related to the insertion, and its relation to the implants is by position.

Options

Several smaller tables are shown in the data model diagram in figure 3.5, which only stores a name and an id. These tables list options for each parameter. For example Material will list the following items: "Autologt", "Allogent", "Xenogent", "Allomplastisk". The id will then be stored in its related table, which in this case is the Implant table.

Vendors and implant types

Although implants are stored with a reference number and LOT number, tables have been added for vendors and implant types. In these tables, a list of each vendor's catalog of implants can be stored. This way, the artifact can provide a search through implants and even some auto-completion when entering the reference number. More importantly, the addition of values to these tables can provide verification that the entered implant is correct.

4.4 Security

4.4.1 Authorization

In order to access resources on the backend, a user has to be authorized. Resources in the GraphQL API are specified with which access rights are required to access them and checked against the user. The backend implements a JWT (Json Web Token) solution to authorize users. JWT is an open standard for a self-contained way of transmitting information between two parties securely (IETF, 2021). When a JWT Token has a valid claim, the requested resource is returned.

Implementation

Using the GraphQL-dotNET implementation, a require-permission function call can be added to any field on graph objects or queries. This is simply added to any field as shown in figure 4.7. The use of authorization was added by adding middleware to the GraphQL API. Through this middleware, the use of JWT was also added. JWT (Json Web Tokens) are an IEFT standard containing an encrypted payload of the authorization information (IETF, 2021).

```
Field<OptionsGraphType>(
    name: "Options", description: "Gets lists of options",
    resolve: context => new OptionsGraphType(data)
).RequirePermission("nameOfPermission");
```

Figure 4.7: Require Permission.

4.4.2 Authentication

Authentication to the artifact is intended to be done using a secure third-party login called ID-porten. ID-porten is a common log-in solution that requires an electronic ID. The electronic ID verifies that users are who they claim to be. ID-porten is operated by the Norwegian Digitalisation Agency

(Digdir, 2021). This authentication will be managed by an authorization server, which then provides a JWT to be used with the artifact backend. The authorization server will be further explained and implemented by Steffen Andrè Grønmo in his Master's thesis (Grønmo, 2021).

4.5 Performance

Beyond the ability to quickly provide input forms and receive data, the artifact does not require a lot of performance, but it should still be able to serve many users at the same time. The .NET 5 framework is used to develop the artifact, which means that quite significant performance and scalability are possible. Stephen Toub has performed some benchmarks showing the performance of .NET 5 as well as its improvements from previous versions (Toub, 2020).

4.5.1 Concurrency

In each query and each graph object in the GraphQL API, the fields have been implemented asynchronously. This means that only the requested data is being retrieved from the database for each query to the API. This requires a good separation of functionality in the business logic.

4.5.2 Performance Testing

As part of the evaluation of the artifact, it has been tested using several different tests to determine its performance. These tests are executed using the open-source K6 load testing tool. K6 is a free load testing tool that makes performance tests and lets the user define tests using Javascript (K6, 2021a).

Hardware

The tests were run on a local machine with the following specifications:

- Processor: Intel(R) Core(TM) i9-9900KF CPU @ 3.60GHz
- Ram: 16 GB DDR4 2666MHz
- Graphics: MSI GeForce RTX 3090 SUPRIM X
- OS: Microsoft Windows 10 Education
- OS Version: 10.0.19042 N/A Build 19042

The SQL server is hosted on Heroku, and is likely to be the limiting factor in these tests, as it is a rather cheap option with the following specs available:

- DTU: 5
- vCore: 0
- Memory Size: 0 Bytes
- Maximum Storage: 2 GB
- Maximum Concurrent Sessions: 300
- Location: North-East Europe

With this in mind, the tests will represent the low-end of results one can expect from database hosting options.

Test query

In order to simulate a clinician opening a form for registration of a procedure, a query that retrieves all data required to fill a complete form is used. The query consists of all parameters representing possible options for all fields recorded by the artifact. The request will concurrently get twelve parameters, which all return a list of results. This means that each request does a fair amount of work on the backend. The data retrieved by each query is shown in appendix 8.1. The query is shown below.

```
{
  options {
    antibioticsTypes{id, name}
    extractionReasonTypes{id, name}
    lekholmZarbVolumeTypes{id, name}
    lekholmZarbDensityTypes{id, name}
    materialTypes{id, name}
    methodTypes{id, name}
    missingReasonTypes{id, name}
    prostheticConstructionTypes{id, positionFrom, positionTo,
        numberOfJoints, isPlateProsthetics}
    removalReasons{id, name}
    stabilityTypes{id, name}
    timeOptionTypes{id, name}
    vendors{id, name}
  }
}
```

Load Testing

The primary goal with load testing is to assess the performance of the system with a focus on concurrent requests and users (K6, 2021b). For this test, load testing is done using one request per second for each user and one request per 10 seconds for each user. This is done over an extended period. Results from the load test is shown in figure 4.8. The average, median, and ninety-fifth percentile are shown for each test, from left to right. The graph shows that the artifacts' performance degrades as it approaches 400 users. In figure 4.9 we can see that the performance degrades for some requests, increasing the average and ninety-fifth percentile, while the median remains very similar.

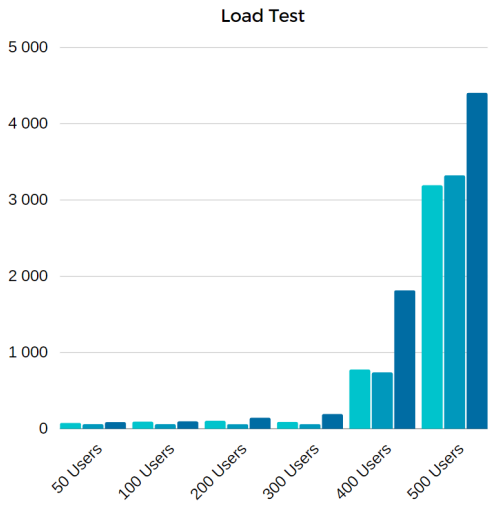

Figure 4.8: Load test for 50-500 users.

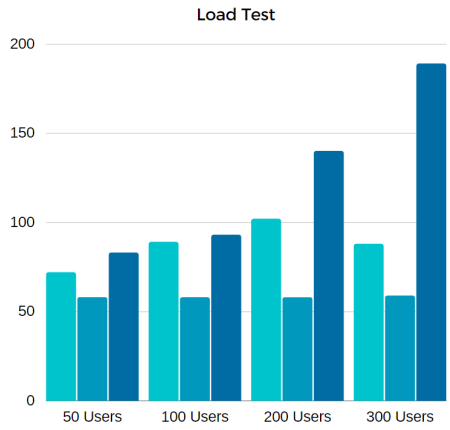

Figure 4.9: Load test for 50-300 users.

Smoke Test

A smoke test is a basic test designed to verify that an API works, as a sanity check after code changes (K6, 2021c). Running smoke test can be very useful for API methods that are used a lot, as well as for API methods that are required to be very responsive. In figure 4.10 there is a snippet of a detailed log output from a smoke test, using one user to repeatedly check a query for one minute.

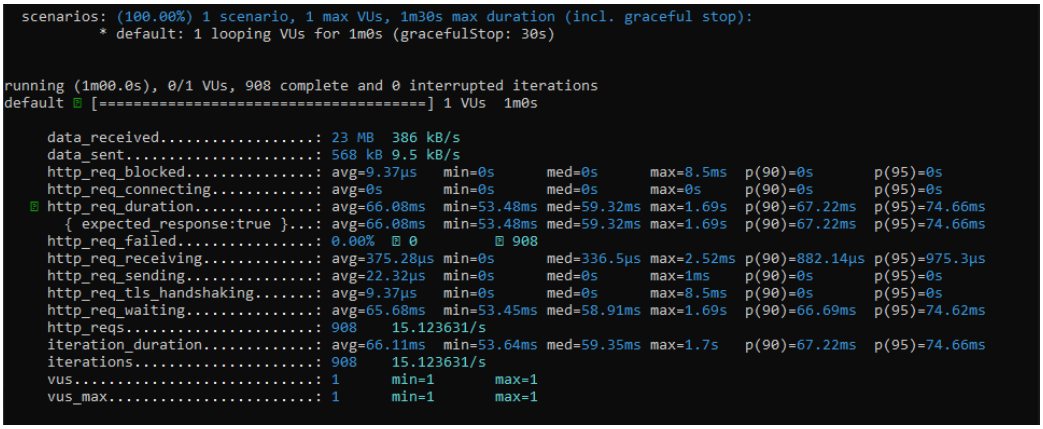

Figure 4.10: Output from a smoke test using the K6 load testing tool.

Soak Test

A soak test is a test running for a longer time, aimed at discovering performance and reliability issues for a system that is under pressure for an extended period of time (K6, 2021d). The soak test was run for ten hours, using 250 concurrent users, making the combined query shown earlier in this section once per ten seconds. The test sent nearly nine hundred thousand requests, where every single request was successful. The average response time was 81.28 milliseconds, the median response time was 59.31 milliseconds, and the ninety-fifth percentile was 158.44 milliseconds. Complete output can be found in appendix 8.3.

Stress Test

A stress test is intended to test a system under extreme conditions, where the system is put under a load that is higher than its normal use and higher than the system is expected to handle (K6, 2021e). K6 lists four qualities the stress is intended to test.

- How your system will behave under extreme conditions.
- What the maximum capacity of your system is in terms of users or throughput.

- The breaking point of your system and its failure mode.
- If your system will recover without manual intervention after the stress test is over.

The result from the stress test can be seen in figure 4.11. Based on the stress tests, the system is able to sustain a very high amount of requests without failing, and instead responding with increased latency. When the system is pushed past its breaking point, as shown with a test using 1500 users, which can be found in appendix 8.3, the system will fail some requests but recover to normal operation quickly after it is no longer overwhelmed.

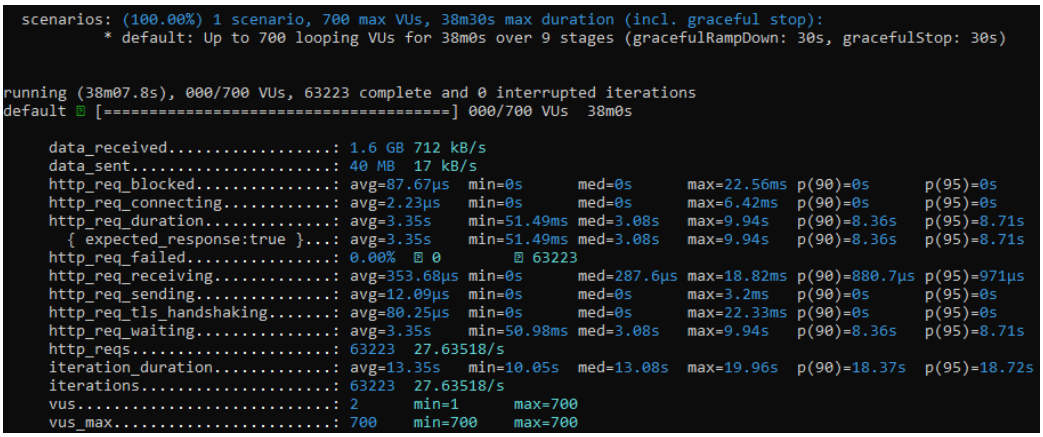

Figure 4.11: Output from a stress test using the k6 load testing tool.

4.5.3 Performance test summary

The performance tests indicate that more than 300 users can use the system concurrently, even with a low-tier database subscription. Furthermore, the stress tests show that the system can handle over 700 users for a while with significantly higher delays but without failing. It also recovers nicely after periods with too many requests, causing some to time out. The soak test also shows that the system can handle 250 concurrent users for long time periods without failing. Complete results from the tests can be found in appendix 8.2.

Chapter 5

Evaluation

5.1 Data Model

During the meetings after each of the iterations explained in section 3.2.2, the state of the artifact was evaluated as part of a lean development. The artifact was evaluated by domain experts. The domain experts mainly evaluated the content of the data model through the data-entry frontend, developed by Elise Fiskeseth (Fiskeseth, 2022). The relations of the data model were also formulated in a way that was intended to be more easily understandable for clinicians, shown in appendix 8.2. This was presented as the table shown in figure 3.3, so that the domain experts could evaluate the relations in the data model as well. These evaluations were used to gradually develop and evaluate the data model shown in figure 3.5, until the clinicians were satisfied that the data accurately recorded the necessary data.

5.2 Data Access

An important task of this project, as described by research question five, is the availability of the artifact. To evaluate the availability of the artifact, a series of load tests were done. The load tests are described in section 4.5.2, and the results of all tests are shown in appendix 8.3. The performance

tests indicate that the artifact can handle a lot of concurrent users, and despite delays, is able to recover from large bursts of concurrent users. The performance tests are summarized in section 4.5.3.

5.3 Architecture

As described in research question three, a large part of this project is the development of an architecture for a dental implant quality registry artifact. While the architecture proposed in section 4.2 is quite extensive, only a part of it, as marked with blue in figure 4.6, is implemented. For that reason, unimplemented parts of the architecture is not directly evaluated. The implemented parts of the architecture is shown to be effective by the results in the previous sections. The benefits of the architecture is also further explained as the third research question is answered in section 6.1.3.

5.4 Interview

As a qualitative evaluation of the artifact, a dental expert who had no previous involvement with the artifact was interviewed, using a semi structured interview. The interview started with some questions to determine the interviewees background, then an introduction to the artifact was given to the interviewee, where feedback was recorded with the interviewees consent. Then the interviewee got some tasks to solve, using the artifact. Finally, some questions were supposed to be asked, however there was not enough time to get through all, which lead the final few questions to be answered in an email. The interview was held in Norwegian and a transcript can be found in section 8.4. The responses to the interview are summarized below.

5.4.1 Semi structured interview

This master's thesis is structured to fit within the research type design science, which is a type of qualitative research methodology. The interview contained questions related to the artifact, that the interviewee could answer

in any way they liked. This type of interview is of the type semi structured, which contains questions regarding a topic that the researchers have set, but it does not require specific answers. A semi structured interview is therefore open ended, with an informal tone (Longhurst, 2003). When questions are open ended, they open for answers and viewpoints that the interviewer might not have thought of beforehand, and this improves the research (Raworth et al., 2012). For these reasons, a semi structured interview was determined to give the most beneficial feedback within the time constraints of the project. It would have been more beneficial to interview several dental expert, however it was too difficult to find experienced clinicians willing to participate in the evaluation.

5.4.2 The interviewee

The interviewee has 20 years of experience setting 500-600 implants each year, and has experience with three different implant systems. She also has experience leading dentist education and lecturing for dentists and implant vendors. She has some experience with other research from other registries, and thinks a quality registry for dental implants is important. Especially because implant vendors are focusing sales of implants to dentist without implant expertise. She also think that a dental implant quality registry will lead to some exciting research, and make sure bad implants are rooted out more quickly.

5.4.3 The current process

Implants are primarily chosen by the dentist who chooses the crown or bride that is placed on top of the implant, however sometimes the implant surgeon chooses the implant themselves. The implants used are largely chosen based on previous experience, where the condition of the patient is considered, and the most important parameter is the planned prosthetic construction to be mounted on top of the implants. Some dentists only use one implant system, and will always use that. Statistics and information is scarcely available, and

vendors keep their own statistics for themselves while Master's this projects are used to gather information about recently used implants.

5.4.4 The dental implant quality registry

The artifact was easy to understand, and can be an important tool to ensure high quality treatment. It is preferable to integrate the artifact as much as possible, to reduce time spent logging in to and switching systems, as long as the data gets to where it is supposed to. Historic information such as reason for loss of teeth, some diagnoses and if the patient is a smoker can be useful to record and present in the system. The interviewee also thinks that the more specific information the artifact records, the better, as it makes it easier to make connections, and had some feedback to change the wording and recorded parameters. She also thinks that some information such as refunds may be unnecessary to include, while reason for removal and reason for missing teeth often will be excessive. She also missed some more information about the healing distance in a one or two step operation. Filling in information in the artifact can be naturally grouped with entries in the journal, and should easily become a habit, as long as the artifact is easily available.

5.5 Summary

The evaluation is split in two parts. Firstly, the data model, architecture and data access was evaluated based on the iterations in section 3.2.2, and then by a series of tests summarized in section 4.5.3. Then the artifact was evaluated using a semi structured interview. The key points from the evaluation are listed below.

- The data model records most of the necessary data.
- The data is accessible.
- The architecture facilitates the implemented functionality in the artifact well .

- A dental implant registry can be an important tool to ensure high quality treatment.
- The artifact can be integrated into the workflow easily.
- The artifact should be integrated as much as possible with other relevant systems.
- The artifact could still have some values changed a little to separate types of procedures

Chapter 6

Discussion

This chapter will discuss the research method and research questions, followed by some reflections on frameworks used to develop the artifact. Design science research should provide contributions to the knowledge base, in addition to solving problems in the artifact domain (Pfeffers et al., 2006) (A. R. Hevner et al., 2004). This chapter will recap both the contributions to the knowledge base and the artifact domain, achieved by use of the design science research process.

6.1 Research question answers

As mentioned at the start of this thesis, the project aims to develop a dental implant quality registry artifact that can be used in the joint research project. As part of the process to achieve this, the research questions are answered below. The answers provide contributions to the knowledge base through the information discovered during the iterations in the design process.

6.1.1 RQ1: What are the potential needs for a dental implant quality registry?

A dental implant quality registry can provide the necessary data for research to reveal weaknesses in procedures and implants that are used. Data from the hip and knee replacement quality registry discussed in section 2.2.1, revealed differences in durability based on the type of cement used. Similar quality assurance can be provided for dental implants through the data collected by a dental implant quality registry. As mentioned in section 1.1, information on dental implants in Norway is lacking. A quality registry for dental implants can serve as the source of this information. Therefore the needs for a dental implant quality registry, is to provide the information that is currently lacking.

6.1.2 RQ2: What requirements should a quality registry for dental implants have?

In chapter 2 some existing quality registries, and their benefits, are explored. The requirements of the quality registry for dental implants will attempt to adapt the benefits of these registries and suggest improvements. Primarily information about the implant and its prosthetic construction will be recorded.

Hip and knee replacement registry

In section 2.2.1 the hip and knee replacement registry's discoveries show the importance of recording the method and equipment used in the procedure, in addition to the implant itself. This is reflected in the data model found in section 4.3.2, where values such as the procedure method and bone treatment are recorded.

Cleft lip and palate registry

Section 2.2.3 describes a study taking advantage of reports produced by the Norwegian cleft lip and palate registry, implying the value of annual reports. Although annual reports are valuable, they will always be slightly outdated.

A requirement for the dental implant registry will be continuous reporting, providing up-to-date access to the database through an API. Continuous reporting should provide the benefits of the annual report with always up-to-date information and a framework to query data more easily and manage authorization.

The Norwegian diabetic registry for adults

The Norwegian diabetic registry for adults discussed in section 2.2.2 provided a way for different units to monitor their results in comparison to the mean. And in doing so, improved the care for patients by pushing the units that were lagging behind to catch up, which in turn improved the average results. The same effect will be facilitated in the dental implant quality registry by providing real-time access to results for all reporting units. By doing so in real-time, the units will have faster feedback on the effect of their actions.

National quality registry for hand surgery

The Swedish national registry for hand surgery introduced the benefits of the patient-reported outcome, shown in section 2.2.4. The patient-reported outcome will be included in the dental implant quality registry. This should be implemented by adding support for the use of question forms that can be sent to the patients to collect their answers.

6.1.3 RQ3: How should the architecture of a dental implant quality registry be designed to fill the need in research question one, and cover the requirements in research question two?

State of the art

In section 2.3 a similar quality registry is explored. The project integrated a registry with EHR systems and developed a learning health system. Based on that experience, they developed a proof-of-concept architecture. The architecture of that registry is shown in figure 2.2. This architecture served

as a guideline for the implementation of the system architecture of the dental implant quality registry artifact. In section 4.3 the architecture of the artifact is explained. Relevant architectural patterns are explored, and the choice of architecture is justified.

Improvements and contributions

The implementation of the system architecture mentioned above had a weakness. It had to implement a data registration form for each EHR system to be used with the registry, which could lead to lower coverage of patients, as EHR systems that have not implemented a data entry form won't enter their patient data into the registry. The architecture of the dental implant quality registry, shown in figure 4.5, seeks to address this problem. By implementing the artifact as a standalone application, the system will be available for all clinics who should enter data. Furthermore, the quality registry provides an API that allows EDR systems to build their own data entry forms, that can interface with the quality registry. With this approach, the artifact will serve as an extension of the EDRs, providing additional data. The artifact will also be continuously available, which makes the most up-to-date data available at all times, rather than just through yearly reports. It also allows researchers to create their own graphs and reports with up-to-date information at all times. The architecture is further explained in section 4.2, and should fill the needs discovered in research question one and cover the requirements discovered in research question 2.

This project

Only the most important parts of the architecture were implemented in the artifact within the scope of this project, marked with blue in figure 4.6. The rest of the architecture can be further explored and implemented in further work.

6.1.4 RQ4: How should the data model for a quality registry for dental implants be designed and implemented to fill the need and requirements in Research question one and research question two, while supporting the architecture in research question three?

The data model has been iteratively developed over several rounds of feedback from domain experts. The data model is at the heart of the artifact, as the main purpose of the artifact is to store data about the quality of implants and procedures over time. As discovered while answering research question two, storing some additional data about the treatment was highly beneficial for the Hip and knee replacement registry. The National quality registry for hand surgery also implemented retrieval of patient-reported outcome data, which they saw benefits from. The National quality registry for hand surgery is further explored in section 2.2.4. Recording these data variables was also reinforced by the opinions of the domain experts consulted during the iterations of the data model, shown in section 3.2.2. Based on that, information about the procedure and state of the patient is included in the data model shown in figure 3.5. A question form system has also been implemented to capture patient recorded data and is explained in section 3.5.1. The data model is implemented in a SQL database at the core of the artifact shown in the architecture in figure 4.6.

6.1.5 RQ5: How should the data in the dental implant quality registry be made accessible while supporting the discoveries from the previous research questions?

As shown in section 2.2 a common way of making data from registries available is through annual reports, which, although valuable, can be inefficient. As mentioned as a requirement in section 6.1.2, an improvement on annual reporting can be continuous reporting. By doing so, the artifact will be available on the internet, similar to a Software as a Service (SaS) application (Sun

et al., 2007). The artifact will be hosted as a web application, making it available over the internet. A GraphQL API was implemented as the interface to the artifact. GraphQL provides a structured way of querying data while also implementing authorization. The GraphQL API is explained in section 4.1.2, and the authorization, as well as other security and performance concerns, are explained in section 4.4.

6.1.6 Contributions

As shown while answering the research questions above, contributions to the knowledge base was made by first revealing the needs for a dental implant quality registry, then determining its requirements based on the achievements of registries in other domains. This thesis also designed an architecture that supports those discoveries, while also taking advantage of modern concepts in software architecture and web services. A data model has been developed based on iterative feedback from domain experts, and is designed to fill needs mentioned. Finally, building on the architecture and data model, it is shown how accessibility to a dental implant quality registry can be realised in a way that is structured and always up-to-date. In addition, this master's thesis project has contributed an artifact implementing the data model, architecture and accessibility as a web service, which will be used and expanded on in future research.

6.2 Reflections

6.2.1 Design science

Design science served as a useful framework for this project, as it allowed for the development of an artifact. Design science provided good guidelines for how to structure the development process and how to iterate the design and implementation of the artifact. It was also useful in the presentation of results in this thesis. Drawbacks from the use of design science for this project were mostly related to limitations on the iterations, as the evaluation

of each iteration was held inconsistently at times. This was partly due to the COVID-19 pandemic and the difficulty of finding times where all parties could attend a meeting. This caused varying workloads in relation to the time between each iteration.

6.2.2 Tools and framework

The choice of tools and frameworks used in this project was a good fit and accomplished their intended tasks. There were some choices that could have been slightly better but would also bring their own drawbacks.

.NET 5

The .NET 5 framework was used in the development of the backend and has provided a really solid foundation for the artifact. The framework has extensive documentation (Microsoft, 2021b), and provides support for the other tools and frameworks used. The framework provided everything needed to create a web server that can host an API. It also has tools that provide secure methods of cryptography and authorization.

GraphQL

GraphQL provided a good, powerful API on top of .NET 5. As mentioned in section 4.1.2, this was accomplished using a library called "GraphQL-dotNET", which extended the functionality of the .NET 5 API with GraphQL support. GraphQL-dotNET also required a unique implementation of authorization. It also implemented the GraphQL API in a way that caused some of its use cases to not be properly analyzed by static code analysis. And depended on its own error messages instead. This caused parts of the implementation to be more difficult than the traditional API used in .NET 5 applications.

Microsoft SQL Server

Microsoft SQL Server was chosen as the database for this project; however, this can be changed fairly easily in the data access layer of the backend, as most SQL implementations are very similar. Access from the backend to the SQL server was done through dapper, which lets you write the SQL code and map the result into objects. This reduced the workload when changes to the data model had to be made. Due to many changes in the structure of data, using an Object-Relation mapper like Entity Framework could have been preferable to save even more time on code changes.

6.3 Limitations

This project has had some limitations due to its dependency on iterating on the design and development of an artifact while being dependent on meetings for evaluation. Meetings were sparsely held during a portion of the project, while frequently in another. This was partially due to the COVID-19 pandemic and several parties struggling to find times where everyone was available to meet.

6.4 Communication

In design science, as described by Pfeffers et al., 2006, communication is the sixth and final step. This step is intended to to communicate the the problem, the artifact, the artifacts utility, the rigor of the artifacts design and the artifacts effectiveness to researchers (Pfeffers et al., 2006). Through answering the the research questions, this thesis communicates the aforementioned qualities. The research questions build upon one another by first showing the problem of lacking information about dental implants. They then go on to exploring the requirements for the artifact before looking at the architecture, data model and accessibility of the artifact. While doing so, the utility of the artifact is investigated by looking at similar artifacts in other domains in chapter 2, and then further shown as the artifact was evaluated in section

5. The rigor of the artifacts design is shown by the implementation of its architecture in section 4.2, and then verified through a series of load tests summarised in section 4.5.3. The artifacts effectiveness to researchers is evaluated in section 5.4, and will be further shown in future work in the joint research project mention in the introduction of this thesis. The master's thesis project by Oddmund Huseby (Huseby, 2022) utilizes the data model from this thesis for prototyping of dental implant registry analytics services, making use of a synthetic populated database, and further communicates the artifacts effectiveness to researchers. In this way the data model has been further tested and evaluated through the use of future services. This testing resulted in a minor change in the data model, only.

Chapter 7

Conclusion and further work

7.1 Conclusion

There is room for improvements in the quality of dental implants and treatments. And with such improvements, a lot of money can be saved both for individuals and the state. By creating a quality registry for the quality of dental implants, such improvements could be made, like they were with the quality registries mentioned earlier. As described in the evaluation, the methods used in this project have proven effective in a test scenario. It was also developed in close cooperation with domain experts, resulting in a well-evaluated data model.

7.2 Further work

As mentioned initially, this project is a part of a larger research project. The scope of this master thesis project is limited to the implementation of the backend of an artifact representing the dental implant quality registry. In section 4.2 a series of unimplemented additions to the system are suggested. These suggestions can be explored in further work. The master's thesis by Oddmund Huseby (Huseby, 2022), mentioned in section 6.4, utilizes this artifact in further work to better explore its effectiveness to researchers.

7.2.1 Patients

As mentioned in section 4.2.4, patients should be able to interact with the artifact. Patients should be able to view and manage data registered about them. The artifact should also record patient-reported outcome data, as explained in the answers to the research questions. Services supporting question forms to record patient-reported outcome data are implemented in the artifact. However, it still requires a view where data can be entered. The artifact system also needs a view and/or an integration that allows patients to access the artifact.

7.2.2 Researchers

Researchers can access the artifact through the GraphQL API. In addition to these statistical models, pre-made reports or graphQL queries should be provided. These can be made available through a view or an integration as suggested in section 4.2.5.

7.2.3 EDR systems and other systems

In section 4.2.6 and section 4.2.8 integration with third-party systems are suggested. These can provide data that doesn't need to be entered into the artifact. With third-party integration, more information related to implants, the procedure, the state of the patient, and the outcome of the procedure on the patient over time.

7.2.4 Implant Vendors

Integration with implant vendors should be implemented to identify all implants easily. In section 4.2.7 the benefits of integration with implant vendors are described.

7.2.5 Using Standards

In addition to the general modules suggested in the architecture, some additional work can be done. Integration with other health systems can be simplified by adding common Learning Health System standards.

FHIR

Health Level Seven Internationals' (HL7) Fast Healthcare Interoperability Resources (FHIR) is a standard that describes resources and API for exchanging EDR data. It can be combined with GraphQL to create an FHIR API over GraphQL (HL7, 2021). The artifact can be extended with FHIR to provide a standard API to integrate with EDRs.

7.2.6 Quality registry qualification

Further work to accomplish quality registry status in Norway must be done. Following the steps described by the National service environment for medical Quality Registries, this can be done (kvalitetsregistre, 2021b). Work must also be done to ensure that all quality registry requirements for establishment and operation are covered (kvalitetsregistre, 2021c).

7.2.7 Further Work on recorded parameters

Although several iterations with several experts have been done to determine the optimal parameters to record in the artifact, the evaluation interview in section 5.4 revealed some minor disagreement on the structure and recorded parameters. More interviews could be held, or a new pilot test of the artifact can be conducted to narrow down the best parameters to record in the artifact.

7.2.8 Integration with "Helseanalyseplattformen"

A platform called "Helseanalyseplattformen" is beeing developed (E-Helse, 2022), and integration with the platform should be considered in future work.

Bibliography

Albrektsson, T., & Wennerberg, A. (2005). The impact of oral implants-past and future, 1966-2042. *J Can Dent Assoc*, *71*(5), 327.

Alter, M. (1978). Medical registers. *Advances in neurology*, *19*, 121–139.

Arner, M. (2016). Developing a national quality registry for hand surgery: Challenges and opportunities. *EFORT open reviews*, *1*(4), 100–106.

Bautista, D. S. (2019). Dental Implants Procedure, Types, Problems, and Cost. https://www.medicinenet.com/dental_implants/article.htm. (accessed 12.05.2022)

Benoit, B., Frédéric, B., & Jean-Charles, D. (2022). Current state of dental informatics in the field of health information systems: A scoping review. *BMC oral health*, *22*(1), 1–17.

Bjerde, K. W., Thomassen, G., & Eide, H. A. (2019). forvaltning av forskningsdata forskerportal og samordnede lagringstjenester. *UNIT*. https://www.unit.no/sites/default/files/media/filer/2019/10/Sak%2035-19%20Forvaltning%20av%20forskningsdata%20-%20Mandat%20for%20konseptfasen-v0.02.pdf. (accessed 12.05.2022)

Chaussalet, T., & Bos, L. (2006). The ICMCC second conference on "Medical and Care Compunetics". https://doi.org/10.1016/S1386-5056(06)00188-2

Cooper, J. G., Thue, G., Claudi, T., Løvaas, K., Carlsen, S., & Sandberg, S. (2013). The Norwegian diabetes register for adults - an overview of the first years. *Norsk Epidemiologi*. https://doi.org/10.5324/nje.v23i1.1599

Dapper. (2021). *Dapper orm*. https://dapper-tutorial.net/ (accessed: 01.06.2021)

Deacon, J. (2009). Model-View-Controller (MVC) Architecture. *John Deacon Computer Systems Development, Consulting Training*. https://www.rareparts.com/pdf/MVC.pdf. (accessed 12.05.2022)

Digdir, N. D. A. (2021). *Id-porten*. https://eid.difi.no/en/id-porten (accessed: 12.05.2022)

E-Helse, D. f. (2022). *Helseanalyseplattformen*. https://www.ehelse.no/programmer/helsedataprogrammet/helseanalyseplattformen%20#Helseanalyseplattformen%20oppsummert (accessed: 07.05.2022)

Enden, T., Bernklev, T., Jelsness-Jørgensen, L.-P., & Amdal, C. D. (2018). Pasientene kjenner best egen helse. *Tidsskrift for Den norske legeforening*.

Evans, E., & Evans, E. J. (2004). *Domain-driven design: Tackling complexity in the heart of software*. Addison-Wesley Professional.

Fiskeseth, E. (2022). Developing a user interface and integrated workflow data collection for a dental implant quality register (in submission). *Master thesis in Software Engineering, Department of Informatics, University of Bergen, Norway*.

Fowler, M. (2012). *Patterns of enterprise application architecture: Pattern enterprise application architecture*. Addison-Wesley.

Garde, S., Knaup, P., Hovenga, E., & Heard, S. (2007). Domain knowledge governance for openehr archetypes. *Methods of Information in Medicine*, *46*, 332–343.

GraphQL, T. G. F. (2021). *A query language for your api*. https://graphql.org/ (accessed: 01.06.2021)

Grønmo, S. (2021). National registry for dental implants - integration. *Master thesis in Software Engineering, Department of Informatics, University of Bergen, Norway*.

Havelin, L. I., Engesæter, L. B., Espehaug, B., Furnes, O., Lie, S. A., & Vollset, S. E. (2000). The norwegian arthroplasty register: 11 years and 73,000 arthroplasties. *Acta Orthopaedica Scandinavica*, *71*(4), 337–353.

Hevner, A., & Chatterjee, S. (2010). Design science research in information systems. *Design research in information systems* (pp. 9–22). Springer.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 75–105.

HL7, H. L. S. I. (2021). *Using graphql with fhir*. https://build.fhir.org/graphql.html (accessed: 05.12.2021)

Huseby, O. (2022). Use of synthetic health data in prototyping for developing dental implant registry services (in submission). *Master thesis in Software Engineering, Department of Informatics, University of Bergen, Norway*.

IETF. (2021). *Json web token (jwt)*. https://datatracker.ietf.org/doc/html/rfc7519 (accessed: 27.11.2021)

InformedHealth.org. (2006). Cologne, germany: Institute for quality and efficiency in health care (iqwig). *Prostate cancer: Overview*.

K6. (2021a). *K6 documentation*. https://k6.io/docs/ (accessed: 11.12.2021)

K6. (2021b). *Load testing*. https://k6.io/docs/test-types/load-testing/ (accessed: 11.12.2021)

K6. (2021c). *Smoke testing*. https://k6.io/docs/test-types/smoke-testing/ (accessed: 11.12.2021)

K6. (2021d). *Soak testing*. https://k6.io/docs/test-types/soak-testing/ (accessed: 11.12.2021)

K6. (2021e). *Stress testing*. https://k6.io/docs/test-types/stress-testing/ (accessed: 11.12.2021)

Klinge, B., Lundström, M., Rosén, M., Bertl, K., Klinge, A., & Stavropoulos, A. (2018). Dental implant quality register—a possible tool to further improve implant treatment and outcome. *Clinical Oral Implants Research*, *29*, 145–151.

Kotusev, S. (2016). Enterprise architecture is not TOGAF. *The Chartered Institute for IT*. https://www.bcs.org/articles-opinion-and-research/enterprise-architecture-is-not-togaf/. (accessed 12.05.2022)

Krogstadmo, S., & Vatnaland, J. E. (2016). *Beintransplantasjon hos pasienter med leppe-kjeve-ganespalte. en litteraturstudie om beintransplantasjoner hos lkg-pasienter i fortid, nåtid og fremtid* (Master's thesis). The University of Bergen.

Kuhn, T. S. (1970). *The structure of scientific revolutions* (Vol. 111). Chicago University of Chicago Press.

kvalitetsregistre, N. s. f. m. (2019). Norsk diabetesregister for voksne. https://www.kvalitetsregistre.no/registers/364/resultater (accessed 12.05.2022)

kvalitetsregistre, N. s. f. m. (2020). nasjonale medisinske kvalitetsregistre status 2020. *https://www.kvalitetsregistre.no/*. https://www.kvalitetsregistre.no/sites/default/files/jan20_statusrapport_2020_web.pdf. (accessed 12.05.2022)

kvalitetsregistre, N. s. f. m. (2021a). Dental Implants Procedure, Types, Problems, and Cost. https://www.kvalitetsregistre.no/register/barn-andre/norsk-kvalitetsregister-leppe-kjeve-ganespalte (accessed 12.05.2022)

kvalitetsregistre, N. s. f. m. (2021b). *Etablering av kvalitetsregistre*. https://www.kvalitetsregistre.no/etablering-av-kvalitetsregistre (accessed: 06.12.2021)

kvalitetsregistre, N. s. f. m. (2021c). *Krav til drift og rapporteringe*. https://www.kvalitetsregistre.no/krav-til-medisinske-kvalitetsregistre (accessed: 06.12.2021)

kvalitetsregistre, N. s. f. m. (2022). Registeroversikt. https://www.kvalitetsregistre.no/registeroversikt (accessed 12.05.2022)

Levay, C. (2016). Policies to foster quality improvement registries: Lessons from the swedish case. *Journal of internal medicine*, *279*(2), 160–172.

Lie, S., Lygre, G., Reichhelm, I., Eggum, E., Bull, V., & Gjengedal, H. (2019). Data fra helfo og norsk pasientskadeerstatning gir liten informasjon om kvalitet og omfang av behandling med tannimplantater i norge. *Nor Tannlegeforen Tid*, *129*, 776–82.

Longhurst, R. (2003). Semi-structured interviews and focus groups. *Key methods in geography*, *3*(2), 143–156.

Lovdata. (2014). 1.stønadsberettiget undersøkelse og behandling. https://lovdata.no/dokument/SF/forskrift/2014-12-16-1702 (accessed 12.05.2022)

Lygre, G. B., Gjengedal, H., Lie, S. A., Wake, J. D., Reichhelm, I., & Berge, T. L. L. (2020). Prospektiv registrering av implantatretinert oral re-

habilitering. *Rapport til Helsedirektoratet*. https://norceresearch.s3.amazonaws.com/rapport-til-Helsedirektoratet_Pilotprosjekt_Prospektiv-registrering-av-implantatretinert-oral-rehabilitering_25nov2020.pdf

Marsolo, K., Margolis, P. A., Forrest, C. B., Colletti, R. B., & Hutton, J. J. (2015). A digital architecture for a network-based learning health system: Integrating chronic care management, quality improvement, and research. *eGEMs*, *3*(1).

Martin, R. C., Grenning, J., Brown, S., Henney, K., & Gorman, J. (2018). *Clean architecture: A craftsman's guide to software structure and design*. Prentice Hall.

Microsoft. (2021a). *.net and .net core support policy*. https://dotnet.microsoft.com/platform/support/policy/dotnet-core (accessed: 01.06.2021)

Microsoft. (2021b). *.net documentation*. https://docs.microsoft.com/en-us/dotnet/ (accessed: 27.11.2021)

Microsoft. (2021c). *Asp.net*. https://dotnet.microsoft.com/apps/aspnet (accessed: 01.06.2021)

Pfeffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V., & Bragge, J. (2006). The design science research process: A model for producing and presenting information systems research. *Proceedings of the First International Conference on Design Science Research in Information Systems and Technology (DESRIST 2006), Claremont, CA, USA*, 83–106.

Raworth, K., Sweetman, C., Narayan, S., Rowlands, J., & Hopkins, A. (2012). *Conducting semi-structured interviews*. Oxfam.

Reselman, B. (2020). TOGAF and the history of Enterprise Architecture. *red hat*. https://www.redhat.com/architect/togaf. (accessed 12.05.2022)

Richards, T., Coulter, A., & Wicks, P. (2015). Time to deliver patient centred care. *Bmj*, *350*.

Samonas, S., & Coss, D. (2014). The cia strikes back: Redefining confidentiality, integrity and availability in security. *Journal of Information System Security*, *10*(3).

Stewart, M. (2001). Towards a global definition of patient centred care: The patient should be the judge of patient centred care. *Bmj*, *322*(7284), 444–445.

Sun, W., Zhang, K., Chen, S.-K., Zhang, X., & Liang, H. (2007). *Software as a service: An integration perspective* (B. J. Krämer, K.-J. Lin, & P. Narasimhan, Eds.). Springer Berlin Heidelberg.

Tait, A. R., & Voepel-Lewis, T. (2015). Digital multimedia: A new approach for informed consent? *Jama*, *313*(5), 463–464.

TOGAF, T. O. G. (2021). *The togaf® standard, version 9.2 overview*. https://www.opengroup.org/togaf (accessed: 04.12.2021)

Toub, S. (2020). Performance Improvements in .NET 5. *Microsoft Devblogs*. https://devblogs.microsoft.com/dotnet/performance-improvements-in-net-5/. (accessed: 12.05.2022)

Chapter 8

Appendix

8.1 Data options for pre-evaluation artifact

8.1.1 RemovalReason

- Løsnet pga. manglende beinfeste
- Ytre traume
- Infeksjon
- Implantatfraktur
- Feil implantatposisjon
- Protetisk overbelastning
- Annen årsak

8.1.2 Stability

- Redusert stabilitet
- God stabilitet

8.1.3 Antibiotics

- Annet
- Penicillin
- Metronidazole
- Clindamycin
- Erythromycin

8.1.4 ExtractionReason

- Ukjent / Annet
- Karies
- Periodontitt

8.1.5 MissingReason

- Ukjent
- Agenisi
- Traume
- Ekstraksjon

8.1.6 Method

- En trinns
- To trinns

8.1.7 Timeoption

- Under 1 dag
- Under 1 uke

- Inntil 8 uker
- Over 8 uker

8.1.8 Material

- Autologt
- Allogent
- Xenogent
- Allomplastisk

8.1.9 BoneAugmentationMethod

- Bløtvevsoppbygning
- Alveolar prosess
- Beinoppbygning i bihule
- Annen hardvevsoppbygning

8.1.10 LekholmZarbDensity

- Type 1: Helt homogent kompakt bein
- Type 2: Tykt lag med kompakt bein som omgir en kjerne av tett trabekulært bein
- Type 3: Tynt lag med kompakt bein som omgir en kjerne av tett trabekulært bein
- Type 4: Tynt lag med kompakt bein som omgir en kjerne av trabekulært bein med lav tetthet

8.1.11 LekholmZarbDensity

- A

- B
- C
- D
- E

8.2 Table of value relations

8.2.1 1 Gang per implantat

- Implantattype
- LotNr
- Stabilitet score
- Tid til belastning
- Implantatets diameter
- Implantatets lengde
- Posisjon
- Membran + Resorberbar
- Beinaugmentasjon (metode/materiale og før/under)
- Komplikasjoner
- Helfo refusjon
- Årsak til manglende tann
- Ekstraksjonsgrunn og tid

8.2.2 1 Gang per operasjon

- Antibiotika (type, dose, varighet + før/under/etter)
- Klinikk

- Guideskinne
- Innsettelsesdato
- Operasjonsmetode
- Pasient
- Lekholm zarb (tetthet, volum: 1-4)

8.2.3 1 Gang per planlagt konstruksjon / bro

- Posisjon Fra
- Posisjon Til
- Antall ledd

8.3 Result of all performance Tests

8.3.1 Load Tests

All users spam requests, for increasing amount of users until 50 users:

```
1     data_received ..................: 1.0 GB 967 kB/s
2     data_sent ......................: 26 MB  24 kB/s
3     http_req_blocked ...............: avg=99.34 s    min=0s
        med=0s       max=22.31ms p(90)=0s       p(95)=0s
4     http_req_connecting ............: avg=2.67 s     min=0s
        med=0s       max=1.14ms  p(90)=0s       p(95)=0s
5       http_req_duration ..............: avg=6.58s     min=53.58ms
        med=5.78s    max=14.42s  p(90)=13.18s   p(95)=13.72s
6       { expected_response:true }...: avg=6.58s     min=53.58ms
          med=5.78s    max=14.42s  p(90)=13.18s   p(95)=13.72s
7     http_req_failed ................: 0.00%       0
        40328
8     http_req_receiving .............: avg=382.38 s   min=0s
        med=342.7 s  max=14ms    p(90)=898.03 s  p(95)=979.8 s
```

```
9       http_req_sending ..............: avg=20.11 s    min=0s
          med=0s       max=13.99ms p(90)=0s       p(95)=0s
10      http_req_tls_handshaking ......: avg=90.62 s    min=0s
          med=0s       max=21.26ms p(90)=0s       p(95)=0s
11      http_req_waiting ..............: avg=6.58s     min=53.5ms
          med=5.78s    max=14.41s  p(90)=13.18s    p(95)=13.72s
12      http_reqs .....................: 40328   37.339064/s
13      iteration_duration ............: avg=6.59s     min=53.58ms
          med=5.78s    max=14.42s  p(90)=13.18s    p(95)=13.72s
14      iterations ....................: 40328   37.339064/s
15      vus ...........................: 1       min=1          max=500
16      vus_max .......................: 500     min=500        max=500
```

100 users spam requests

```
1       data_received .................: 811 MB 966 kB/s
2       data_sent .....................: 20 MB  23 kB/s
3       http_req_blocked ..............: avg=25.7 s     min=0s
          med=0s       max=14.48ms p(90)=0s       p(95)=0s
4       http_req_connecting ...........: avg=650ns     min=0s
          med=0s       max=1.07ms  p(90)=0s       p(95)=0s
5         http_req_duration ...........: avg=2.29s     min=53.25ms
          med=2.59s    max=3.29s   p(90)=2.88s    p(95)=2.93s
6         { expected_response:true }...: avg=2.29s     min=53.25ms
            med=2.59s    max=3.29s   p(90)=2.88s    p(95)=2.93s
7       http_req_failed ...............: 0.00%       0
          31410
8       http_req_receiving ............: avg=383.94 s   min=0s
          med=360.1 s  max=4.39ms  p(90)=884.01 s  p(95)=965.8 s
9       http_req_sending ..............: avg=14.19 s    min=0s
          med=0s       max=2.76ms  p(90)=0s       p(95)=0s
10      http_req_tls_handshaking ......: avg=23.76 s    min=0s
          med=0s       max=13.96ms p(90)=0s       p(95)=0s
11      http_req_waiting ..............: avg=2.29s     min=53.13ms
          med=2.59s    max=3.29s   p(90)=2.88s    p(95)=2.93s
12      http_reqs .....................: 31410   37.390273/s
13      iteration_duration ............: avg=2.29s     min=53.25ms
          med=2.59s    max=3.29s   p(90)=2.88s    p(95)=2.93s
14      iterations ....................: 31410   37.390273/s
```

```
15      vus ..............................: 1       min=1          max=100
16      vus_max ..........................: 100     min=100        max=100
```

One request each second with 50 users

```
1  data_received ..................: 685 MB 814 kB/s
2  data_sent ......................: 17 MB  20 kB/s
3  http_req_blocked ...............: avg=15.48 s    min=0s       med=0
       s        max=11.99ms p(90)=0s       p(95)=0s
4  http_req_connecting ............: avg=388ns     min=0s       med=0s
           max=1ms      p(90)=0s       p(95)=0s
5      http_req_duration ..............: avg=347.75ms min=51.85ms
       med=371.46ms max=3.08s    p(90)=529.97ms p(95)=572.56ms
6  { expected_response:true }...: avg=347.75ms min=51.85ms med
       =371.46ms max=3.08s    p(90)=529.97ms p(95)=572.56ms
7  http_req_failed ................: 0.00%       0                26659
8  http_req_receiving .............: avg=370.45 s   min=0s       med
       =334.4 s    max=3.34ms  p(90)=884 s      p(95)=968.1 s
9  http_req_sending ...............: avg=7.75 s     min=0s       med=0
       s        max=3.33ms  p(90)=0s       p(95)=0s
10 http_req_tls_handshaking .......: avg=14.29 s    min=0s       med=0
       s        max=10.11ms p(90)=0s       p(95)=0s
11 http_req_waiting ...............: avg=347.37ms min=51.67ms med
       =371.18ms max=3.08s    p(90)=529.65ms p(95)=572.11ms
12 http_reqs ......................: 26659   31.697664/s
13 iteration_duration .............: avg=1.35s     min=1.05s    med
       =1.37s     max=4.1s     p(90)=1.53s     p(95)=1.57s
14 iterations .....................: 26659   31.697664/s
15 vus ............................: 1       min=1          max=50
16 vus_max ........................: 50      min=50         max=50
```

One request each second with 100 users.

```
1       data_received ..................: 587 MB 978 kB/s
2       data_sent ......................: 14 MB  24 kB/s
3       http_req_blocked ...............: avg=34.8 s     min=0s
            med=0s       max=11.92ms p(90)=0s     p(95)=0s
4       http_req_connecting ............: avg=762ns     min=0s
            med=0s       max=1ms      p(90)=0s     p(95)=0s
```

```
5       http_req_duration ..............: avg=1.36s      min=52.62ms
        med=1.54s    max=2.14s    p(90)=1.84s p(95)=1.91s
6       { expected_response:true }...: avg=1.36s      min=52.62ms
           med=1.54s    max=2.14s    p(90)=1.84s p(95)=1.91s
7     http_req_failed ................: 0.00%        0
        22771
8     http_req_receiving .............: avg=365.85 s   min=0s
        med=323.9 s   max=4.5ms    p(90)=878 s   p(95)=965.05 s
9     http_req_sending ...............: avg=9.47 s     min=0s
        med=0s        max=1.25ms   p(90)=0s      p(95)=0s
10    http_req_tls_handshaking .......: avg=32.39 s    min=0s
        med=0s        max=11.92ms p(90)=0s      p(95)=0s
11    http_req_waiting ...............: avg=1.36s      min=52.19ms
        med=1.54s    max=2.14s    p(90)=1.84s p(95)=1.9s
12    http_reqs ......................: 22771   37.936748/s
13    iteration_duration .............: avg=2.37s      min=1.05s
        med=2.55s    max=3.14s    p(90)=2.85s p(95)=2.91s
14    iterations .....................: 22771   37.936748/s
15    vus ............................: 1        min=1          max=100
16    vus_max ........................: 100      min=100        max=100
```

One rquest per 10 seconds for 50 users

```
1     data_received ..................: 31 MB  102 kB/s
2     data_sent ......................: 797 kB 2.6 kB/s
3     http_req_blocked ...............: avg=324.52 s   min=0s
        med=0s        max=10.53ms p(90)=0s         p(95)=0s
4     http_req_connecting ............: avg=6.95 s     min=0s
        med=0s        max=578.5 s   p(90)=0s         p(95)=0s
5       http_req_duration ..............: avg=71.5ms     min=53.7ms
        med=58.26ms max=1.68s    p(90)=67.45ms  p(95)=83.16ms
6       { expected_response:true }...: avg=71.5ms     min=53.7ms
           med=58.26ms max=1.68s    p(90)=67.45ms  p(95)=83.16ms
7     http_req_failed ................: 0.00%        0
        1219
8     http_req_receiving .............: avg=394.77 s   min=0s
        med=376.9 s   max=1.33ms   p(90)=898.84 s   p(95)=967.74 s
9     http_req_sending ...............: avg=17.54 s    min=0s
        med=0s        max=1ms      p(90)=0s         p(95)=0s
```

```
10      http_req_tls_handshaking .......: avg=300.07 s   min=0s
            med=0s       max=9.97ms  p(90)=0s       p(95)=0s
11      http_req_waiting ...............: avg=71.09ms  min=52.74ms
            med=57.86ms max=1.68s   p(90)=66.65ms  p(95)=82.63ms
12      http_reqs ......................: 1219    3.982334/s
13      iteration_duration .............: avg=10.07s   min=10.05s
            med=10.06s  max=11.68s  p(90)=10.07s   p(95)=10.09s
14      iterations .....................: 1219    3.982334/s
15      vus ............................: 1       min=1       max=50
16      vus_max ........................: 50      min=50      max=50
```

One rquest per 10 seconds for 100 users

```
1       data_received ..................: 62 MB  203 kB/s
2       data_sent ......................: 1.6 MB 5.2 kB/s
3       http_req_blocked ...............: avg=322.91 s   min=0s
            med=0s       max=11.07ms p(90)=0s       p(95)=0s
4       http_req_connecting ............: avg=6.25 s     min=0s
            med=0s       max=1.57ms  p(90)=0s       p(95)=0s
5         http_req_duration ..............: avg=89.85ms  min=52.83ms
          med=58.21ms  max=3.03s   p(90)=75.32ms p(95)=93.29ms
6         { expected_response:true }...: avg=89.85ms  min=52.83ms
              med=58.21ms  max=3.03s   p(90)=75.32ms p(95)=93.29ms
7       http_req_failed ................: 0.00%       0
            2431
8       http_req_receiving .............: avg=371.36 s   min=0s
            med=329.29 s   max=4.07ms  p(90)=886.2 s   p(95)=955.55 s
9       http_req_sending ...............: avg=21.51 s    min=0s
            med=0s       max=1ms     p(90)=0s       p(95)=0s
10      http_req_tls_handshaking .......: avg=298.8 s    min=0s
            med=0s       max=10.55ms p(90)=0s       p(95)=0s
11      http_req_waiting ...............: avg=89.46ms  min=52.34ms
            med=57.86ms  max=3.03s   p(90)=75.06ms p(95)=93.05ms
12      http_reqs ......................: 2431    7.90308/s
13      iteration_duration .............: avg=10.09s   min=10.05s
            med=10.06s   max=13.05s  p(90)=10.08s  p(95)=10.1s
14      iterations .....................: 2431    7.90308/s
15      vus ............................: 2       min=2       max=100
16      vus_max ........................: 100     min=100     max=100
```

One request per 10 seconds for 200 users

```
1     data_received ..................: 276 MB 456 kB/s
2     data_sent ......................: 6.9 MB 11 kB/s
3     http_req_blocked ...............: avg=148.34 s   min=0s
        med=0s       max=33.89ms p(90)=0s       p(95)=0s
4     http_req_connecting ............: avg=2.34 s     min=0s
        med=0s       max=1.45ms  p(90)=0s       p(95)=0s
5       http_req_duration ..............: avg=102.04ms min=51.37ms
        med=58.59ms max=7.43s    p(90)=106.26ms p(95)=139.78ms
6       { expected_response:true }...: avg=102.04ms min=51.37ms
          med=58.59ms max=7.43s    p(90)=106.26ms p(95)=139.78ms
7     http_req_failed ................: 0.00%      0
        10781
8     http_req_receiving .............: avg=418.96 s   min=0s
        med=416.5 s  max=5.1ms   p(90)=900.3 s    p(95)=977.2 s
9     http_req_sending ...............: avg=10.99 s    min=0s
        med=0s       max=1.03ms  p(90)=0s       p(95)=0s
10    http_req_tls_handshaking .......: avg=137 s      min=0s
        med=0s       max=32.72ms p(90)=0s       p(95)=0s
11    http_req_waiting ...............: avg=101.61ms min=51.13ms
        med=58.18ms max=7.43s    p(90)=105.7ms  p(95)=139.21ms
12    http_reqs ......................: 10781  17.782292/s
13    iteration_duration .............: avg=10.1s    min=10.05s
        med=10.06s  max=17.45s  p(90)=10.11s   p(95)=10.14s
14    iterations .....................: 10781  17.782292/s
15    vus ............................: 2      min=2      max=200
16    vus_max ........................: 200    min=200    max=200
```

One request per 10 seconds for 300 users

```
1     data_received ..................: 415 MB 683 kB/s
2     data_sent ......................: 10 MB  17 kB/s
3     http_req_blocked ...............: avg=145.68 s   min=0s
        med=0s       max=16.41ms p(90)=0s       p(95)=0s
4     http_req_connecting ............: avg=3.21 s     min=0s
        med=0s       max=1.01ms  p(90)=0s       p(95)=0s
5       http_req_duration ..............: avg=88.03ms  min=52.41ms
        med=59.3ms   max=1.9s    p(90)=126.22ms p(95)=189.35ms
```

```
6       { expected_response:true }...: avg=88.03ms  min=52.41ms
          med=59.3ms  max=1.9s    p(90)=126.22ms p(95)=189.35ms
7     http_req_failed ..............: 0.00%      0
        16193
8     http_req_receiving ...........: avg=355.87 s  min=0s
        med=295.7 s  max=7.88ms  p(90)=871.2 s   p(95)=961.42 s
9     http_req_sending .............: avg=15.15 s   min=0s
        med=0s       max=1.04ms  p(90)=0s       p(95)=0s
10    http_req_tls_handshaking .....: avg=135.91 s  min=0s
        med=0s       max=16.41ms p(90)=0s       p(95)=0s
11    http_req_waiting .............: avg=87.66ms  min=51.89ms
        med=58.94ms max=1.9s    p(90)=125.44ms p(95)=188.65ms
12    http_reqs ....................: 16193  26.633951/s
13    iteration_duration ...........: avg=10.09s   min=10.05s
        med=10.06s  max=11.93s  p(90)=10.13s   p(95)=10.19s
14    iterations ...................: 16193  26.633951/s
15    vus ..........................: 5      min=5        max=300
16    vus_max ......................: 300    min=300      max=300
```

One request per 10 seconds for 400 users

```
1     data_received ................: 521 MB 858 kB/s
2     data_sent ....................: 13 MB  21 kB/s
3     http_req_blocked .............: avg=152.63 s  min=0s
        med=0s       max=17.07ms p(90)=0s       p(95)=0s
4     http_req_connecting ..........: avg=2.29 s   min=0s
        med=0s       max=7.47ms  p(90)=0s       p(95)=0s
5       http_req_duration ..........: avg=774.78ms min=52.22ms
        med=738.87ms max=3.93s   p(90)=1.64s   p(95)=1.81s
6       { expected_response:true }...: avg=774.78ms min=52.22ms
          med=738.87ms max=3.93s   p(90)=1.64s   p(95)=1.81s
7     http_req_failed ..............: 0.00%      0
        20231
8     http_req_receiving ...........: avg=367.34 s  min=0s
        med=323.5 s  max=9.66ms  p(90)=878.7 s   p(95)=964.95 s
9     http_req_sending .............: avg=14.53 s   min=0s
        med=0s       max=4.53ms  p(90)=0s       p(95)=0s
10    http_req_tls_handshaking .....: avg=143.27 s  min=0s
        med=0s       max=17.07ms p(90)=0s       p(95)=0s
```

```
11      http_req_waiting ..... : avg=774.4ms  min=51.9ms
            med=738.21ms max=3.93s   p(90)=1.64s   p(95)=1.81s
12      http_reqs ..... : 20231  33.324741/s
13      iteration_duration ..... : avg=10.78s   min=10.05s
            med=10.74s   max=13.94s  p(90)=11.65s  p(95)=11.82s
14      iterations ..... : 20231  33.324741/s
15      vus ..... : 1        min=1        max=400
16      vus_max ..... : 400      min=400      max=400
```

One request per 10 seconds for 500 users

```
1       data_received ..... : 3.5 GB 962 kB/s
2       data_sent ..... : 85 MB  23 kB/s
3       http_req_blocked ..... : avg=29.5 s     min=0s
            med=0s       max=49.05ms p(90)=0s      p(95)=0s
4       http_req_connecting ..... : avg=630ns    min=0s
            med=0s       max=2.99ms  p(90)=0s      p(95)=0s
5         http_req_duration ..... : avg=3.19s    min=52.24ms
          med=3.32s    max=10.65s  p(90)=4.23s  p(95)=4.4s
6         { expected_response:true }... : avg=3.19s    min=52.24ms
             med=3.32s    max=10.65s  p(90)=4.23s  p(95)=4.4s
7       http_req_failed ..... : 0.00%      0
            134285
8       http_req_receiving ..... : avg=377.91 s  min=0s
            med=355.6 s  max=5.52ms  p(90)=880 s   p(95)=963 s
9       http_req_sending ..... : avg=7.63 s     min=0s
            med=0s       max=3.05ms  p(90)=0s      p(95)=0s
10      http_req_tls_handshaking ..... : avg=27.47 s    min=0s
            med=0s       max=49.05ms p(90)=0s      p(95)=0s
11      http_req_waiting ..... : avg=3.19s    min=51.79ms
            med=3.32s    max=10.65s  p(90)=4.23s  p(95)=4.4s
12      http_reqs ..... : 134285 37.207444/s
13      iteration_duration ..... : avg=13.2s    min=10.05s
            med=13.32s   max=20.67s  p(90)=14.24s p(95)=14.4s
14      iterations ..... : 134285 37.207444/s
15      vus ..... : 1        min=1        max=500
16      vus_max ..... : 500      min=500      max=500
```

8.3.2 Smoke Test

```
1  One minute hammering an API with one user to verify that it
       works
2
3        data_received ..................: 23 MB  386 kB/s
4        data_sent ......................: 568 kB 9.5 kB/s
5        http_req_blocked ...............: avg=9.37 s     min=0s
             med=0s       max=8.5ms  p(90)=0s       p(95)=0s
6        http_req_connecting ............: avg=0s       min=0s
             med=0s       max=0s     p(90)=0s       p(95)=0s
7          http_req_duration ..............: avg=66.08ms  min=53.48ms
           med=59.32ms max=1.69s  p(90)=67.22ms  p(95)=74.66ms
8          { expected_response:true }...: avg=66.08ms  min=53.48ms
               med=59.32ms max=1.69s  p(90)=67.22ms  p(95)=74.66ms
9        http_req_failed ................: 0.00%       0
             908
10       http_req_receiving .............: avg=375.28 s   min=0s
             med=336.5 s  max=2.52ms p(90)=882.14 s  p(95)=975.3 s
11       http_req_sending ...............: avg=22.32 s    min=0s
             med=0s       max=1ms    p(90)=0s       p(95)=0s
12       http_req_tls_handshaking .......: avg=9.37 s     min=0s
             med=0s       max=8.5ms  p(90)=0s       p(95)=0s
13       http_req_waiting ...............: avg=65.68ms  min=53.45ms
             med=58.91ms max=1.69s  p(90)=66.69ms  p(95)=74.62ms
14       http_reqs ......................: 908     15.123631/s
15       iteration_duration .............: avg=66.11ms  min=53.64ms
             med=59.35ms max=1.7s   p(90)=67.22ms  p(95)=74.66ms
16       iterations .....................: 908     15.123631/s
17       vus ............................: 1       min=1       max=1
18       vus_max ........................: 1       min=1       max=1
19
20 --
```

8.3.3 Stress Test

Stress test gradually increasing in steps up to 500 users, querying the API once every ten seconds, then scaling down to 0 to recover

```
  scenarios: (100.00%) 1 scenario, 700 max VUs, 45m30s max duration (incl. graceful stop):
           * default: Up to 500 looping VUs for 45m0s over 11 stages (gracefulRampDown: 30s, gracefulStop: 30s)


running (45m07.1s), 000/500 VUs, 69240 complete and 0 interrupted iterations
default ╠═══════════════════════════════════════╣ 000/500 VUs  45m0s

     data_received..................: 1.8 GB 657 kB/s
     data_sent......................: 44 MB  16 kB/s
     http_req_blocked...............: avg=59.19 s    min=0s med=0s       max=23.84ms p(90)=0s       p(95)=0s
     http_req_connecting............: avg=1.53 s     min=0s med=0s       max=3.81ms  p(90)=0s       p(95)=0s
     http_req_duration..............: avg=864.27ms min=51.61ms med=95.52ms max=4.55s   p(90)=3.15s    p(95)=3.46s
       { expected_response:true }...: avg=864.27ms min=51.61ms med=95.52ms max=4.55s   p(90)=3.15s    p(95)=3.46s
     http_req_failed................: 0.00%  0 69240
     http_req_receiving.............: avg=359.69 s   min=0s med=301.7 s  max=4.26ms  p(90)=886.5 s  p(95)=973.2 s
     http_req_sending...............: avg=20.52 s    min=0s med=0s       max=7.92ms  p(90)=0s       p(95)=0s
     http_req_tls_handshaking.......: avg=54.36 s    min=0s med=0s       max=23.33ms p(90)=0s       p(95)=0s
     http_req_waiting...............: avg=863.89ms min=51.41ms med=95.12ms max=4.55s   p(90)=3.15s    p(95)=3.46s
     http_reqs......................: 69240  25.57705/s
     iteration_duration.............: avg=10.87s    min=10.05s med=10.1s   max=14.55s  p(90)=13.15s  p(95)=13.46s
     iterations.....................: 69240  25.57705/s
     vus............................: 1      min=1      max=500
     vus_max........................: 500    min=500    max=500


—
```

Stress test gradually increasing in steps up to 700 users, querying the API once every ten seconds, then scaling down to 0 to recover

```
scenarios: (100.00%) 1 scenario, 700 max VUs, 38m30s max duration (incl. graceful stop):
          * default: Up to 700 looping VUs for 38m0s over 9 stages (gracefulRampDown: 30s, gracefulStop: 30s)


running (38m07.8s), 000/700 VUs, 63223 complete and 0 interrupted iterations
default   [==========================================] 000/700 VUs  38m0s

     data_received..................: 1.6 GB 712 kB/s
     data_sent......................: 40 MB  17 kB/s
     http_req_blocked...............: avg=87.67 s    min=0s med=0s       max=22.56ms p(90)=0s       p(95)=0s
     http_req_connecting............: avg=2.23 s     min=0s med=0s       max=6.42ms  p(90)=0s       p(95)=0s
     http_req_duration..............: avg=3.35s      min=51.49ms med=3.08s max=9.94s   p(90)=8.36s    p(95)=8.71s
       { expected_response:true }...: avg=3.35s      min=51.49ms med=3.08s max=9.94s   p(90)=8.36s    p(95)=8.71s
     http_req_failed................: 0.00%      0 63223
     http_req_receiving.............: avg=353.68 s   min=0s med=287.6 s  max=18.82ms p(90)=880.7 s  p(95)=971 s
     http_req_sending...............: avg=12.09 s    min=0s med=0s       max=3.2ms   p(90)=0s       p(95)=0s
     http_req_tls_handshaking.......: avg=80.25 s    min=0s med=0s       max=22.33ms p(90)=0s       p(95)=0s
     http_req_waiting...............: avg=3.35s      min=50.98ms med=3.08s max=9.94s   p(90)=8.36s    p(95)=8.71s
     http_reqs......................: 63223  27.63518/s
     iteration_duration.............: avg=13.35s     min=10.05s med=13.08s max=19.96s  p(90)=18.37s   p(95)=18.72s
     iterations.....................: 63223  27.63518/s
     vus............................: 2      min=1      max=700
     vus_max........................: 700    min=700    max=700
```

```
24
25 ——
```

Stress test gradually increasing in steps up to 1500 users, querying the API once every ten seconds, then scaling down to 0 to recover

```
1  scenarios: (100.00%) 1 scenario, 1500 max VUs, 45m30s max
       duration (incl. graceful stop):
2            * default: Up to 1500 looping VUs for 45m0s over 11
                 stages (gracefulRampDown: 30s, gracefulStop: 30s)
3
4
5  running (45m08.5s), 0000/1500 VUs, 95197 complete and 76
       interrupted iterations
6  default     [========================================] 0000/1500
       VUs  45m0s
7
8        data_received ..................: 2.5 GB 911 kB/s
9        data_sent ......................: 61 MB  22 kB/s
10       http_req_blocked ...............: avg=127.99 s   min=0s
             med=0s       max=87.47ms p(90)=0s       p(95)=0s
11       http_req_connecting ............: avg=2.8 s     min=0s
             med=0s       max=1.02ms  p(90)=0s       p(95)=0s
12       http_req_duration ..............: avg=11.58s    min=51.2ms
             med=9.01s    max=30.47s  p(90)=29.31s   p(95)=30.01s
13         { expected_response:true }...: avg=11.58s    min=51.2ms
               med=9.01s    max=30.47s  p(90)=29.31s   p(95)=30.01s
14       http_req_failed ................: 0.00%      0
             95273
15       http_req_receiving .............: avg=350.98 s   min=0s
             med=285.6 s  max=24.54ms p(90)=873.7 s   p(95)=962.73 s
16       http_req_sending ...............: avg=13.62 s    min=0s
             med=0s       max=8.99ms  p(90)=0s       p(95)=0s
17       http_req_tls_handshaking .......: avg=120.53 s   min=0s
             med=0s       max=87.22ms p(90)=0s       p(95)=0s
18       http_req_waiting ...............: avg=11.58s    min=50.53ms
             med=9.01s    max=30.47s  p(90)=29.31s   p(95)=30.01s
19       http_reqs ......................: 95273  35.175236/s
```

```
20      iteration_duration ..............: avg=21.57s    min=10.05s
            med=19.01s   max=40.47s   p(90)=39.31s   p(95)=40.01s
21      iterations ......................: 95197   35.147176/s
22      vus .............................: 1       min=1         max
            =1500
23      vus_max .........................: 1500    min=1500      max
            =1500
24
25  --
```

8.3.4 Soak Test

Soak test with 250 users sending one request every ten seconds

```
1   scenarios: (100.00%) 1 scenario, 250 max VUs, 10h0m30s max
        duration (incl. graceful stop):
2           * default: Up to 250 looping VUs for 10h0m0s over 3
                stages (gracefulRampDown: 30s, gracefulStop: 30s)
3
4
5   running (10h00m01.9s), 000/250 VUs, 889337 complete and 0
        interrupted iterations
6   default     [========================================] 000/250 VUs
        10h0m0s
7
8       data_received ..................: 23 GB  632 kB/s
9       data_sent ......................: 559 MB 16 kB/s
10      http_req_blocked ...............: avg=2.2 s      min=0s
            med=0s       max=14.73ms p(90)=0s       p(95)=0s
11      http_req_connecting ............: avg=35ns       min=0s
            med=0s       max=4.95ms  p(90)=0s       p(95)=0s
12      http_req_duration ..............: avg=81.28ms   min=0s
            med=59.31ms max=16.74s   p(90)=124.88ms p(95)=158.44ms
13        { expected_response:true }...: avg=81.28ms   min=0s
              med=59.31ms max=16.74s   p(90)=124.88ms p(95)=158.44ms
14      http_req_failed ................: 0.00%       0
            889337
15      http_req_receiving .............: avg=371.51 s   min=0s
            med=331.4 s   max=9.52ms   p(90)=877.2 s    p(95)=957.3 s
```

```
16      http_req_sending ..............: avg=5.51 s      min=0s
            med=0s       max=12.24ms p(90)=0s       p(95)=0s
17      http_req_tls_handshaking ......: avg=2.05 s      min=0s
            med=0s       max=14.22ms p(90)=0s       p(95)=0s
18      http_req_waiting ..............: avg=80.9ms    min=0s
            med=58.97ms max=16.74s   p(90)=124.49ms p(95)=158.07ms
19      http_reqs .....................: 889337 24.702523/s
20      iteration_duration ............: avg=10.08s    min=10.05s
            med=10.06s   max=26.75s   p(90)=10.13s    p(95)=10.16s
21      iterations ....................: 889337 24.702523/s
22      vus ...........................: 1      min=1          max=250
23      vus_max .......................: 250    min=250        max=250
```

8.4 Interview

The interview was done in Norwegian.

Hva er din erfaring med bruk av tannimplantater?

Jeg satt inn mitt første implantat i Mai 2002, så jeg har satt implantater i ca 20 år. Jeg setter ca 500-600 implantater i året. I tillegg leder jeg utdannelsen for tannlegestudenter, og jeg holder forelesninger for tannleger som bruker implantater, og for leverandører av tannimplantater. Vi bruker 3 forskjellige implantatsystemer om hverandre.

Hvordan velger du hvilket implantat du skal bruke i en prosedyre?

Det kommer an på situasjonen. Det er hovedsakelig den som setter skruen på toppen som bestemmer hvilke implantat som skal brukes. Noen kirurger har bare et system, og da bruker dem det. Når paseientene kommer til meg har dem ofte med seg en ordre fra de som skal sette kronen på toppen. Noen ganger får jeg bare henvist en pasient, og da velger jeg selv hvilket implantat jeg setter.

Hvor finner du informasjon om implantater når du skal ta en avgjørelse på hvilket implantat du skal bruke?

Slik som det er nå går det mest på egen erfaring, knytt til hvilke systemer som har best resultater for pasientens utgangspunkt. Det som har mest betydning for valg av implantat er det som skal festes oppå implantatet, om det er krone, bro eller "overdenture".

Hva er din formening om tilgjengeligheten til statistikk og informasjon om kvaliteten til implantater?

Jeg syntes det er veldig lite. Leverandørene har egen statisikk på hva de selger. Vi har masteroppgaver som f.eks ser på implantater som er brukt de siste 5 årene, men vi fører ingen statistiskk fast. Leverandører gir lifstidsgaranti på implantater, og erstatter implantaer som faller ut, noe som gir dem mulighet til å føre statistikk, men den statistikken blir ikke delt siden det regnes som en bedriftshemmelighet da det kan brukes til å sammenligne leverandører mot hverandre.

Hvilken kjennskap har du til (kvalitets-)register?

Jeg har ganske ganske god kjennskap til register, og har jobbet med ortopeder som har drevet med registerforskning i 25 år. Når jeg startet å sette implantater fyllte vi ut noen skriftlige skjemaer som var ment til å brukes i et register, og skjemaene ble sendt til Haukeland, men det ble ikke registrert i noe system, og endte opp med å bli makulert.

Hva er din mening om et register for tannimplantater?

Jeg syntes det er viktig. Det kan gi spennende forskning. Leverandører har begynt å vende seg mot mindre erfarne tannleger, da tannimplantater har blitt mindre nødvendig de siste årene grunnet bedre tilgang til vedlikehold de siste årene. Det gjør at leverandørene er mer pågående og prøver å selge til tannleger uten spesialkompetanse. Det er risikbalet siden man setter noe inn i mennesker, og risikerer infeksjoner ol. Derfor er det viktig med et register.

Før resten av spørsmålene hadde vi en liten introduksjon til systemet, etterfult av en liten test av systemet hvor intervjuobjektet fikk prøve å registrere pasient og implantater. Underveis fikk vi diverse tilbakemeldinger på ordlyd og sammenheng, noe som dekkes i Elise Fiskeseth sin masteroppgave (Fiskeseth, 2022).

Synes du sidene er oversiktlige?

Ja, dei var oversiktelige oglette å forstå.

Opplevde du at det var for mye informasjon på en skjerm?

Nei, synes det var greit.

Synes du knapper og input felt kom tydelig frem?

Ja.

Hva synes du om at registrering av operasjon og implantat er delt i 2 ulike skjema?

Det er greit for min del, men jo færre klikk og jo færre sider gjer det meir brukarvennlig.

Hva synes du om at registreringsskjema er delt opp i mange seksjoner?

Same svar som over.

Hva synes du om at dette registreres i et eget system sammenlignet med å ha det integrert i f.eks. opus dental?

Alt som kan integreres er bra, jo færre innlogginger og brukerkoder ein må huske på, jo bedre. MEN, det er viktig at dataene kommer dit dei skal, helst kostnadsfritt.

Ser du noen verdi i å ha tilgang til pasienthistorikken i dette skjema?

Eg vil gjerne ha tilgang til årsak til tanntap, anamnese og medisinbruk, røyking mm.

Er dette informasjon du har opplevd mangel på tidligere?

I eit register vil alle slike data vere viktige. Jo meir spesifikk ein kan vere, jo lettare kan ein sjå samanhenger.

Ser du noen nytte i bruken av kvalitetsregisteret?

Ja, heilt klart! Det gjer det mulig å sjå om ein type implantat presetere dårligare enn andre, og kan tas ut av markedet tidligare. I tillegg kan ein få mykje data om pasientane, årsak til tanntap, varighet av implantat, om det er forskjell på operatør, om røyking, medisiner påvirkar forløpet og meir.

Hvilke forventninger har du til et kvalitetsregister for tannimplantater?

Eg håpar det kan bli like bra som hofteregisteret, der me kan få mykje god forskning, gjerne fleire phd kandidatar og ny kunnskap.

Hvordan passer registeret inn i din arbeidsflyt?

Så lenge det er lett tilgjengelig, blir det fort ein vane.

Hvilke andre aktiviteter passer data registreringen sammen med?

Journalføring og registrering av implantata.

Hvilke integrasjoner ser du for deg at kunne vært nyttig: Hvilken informasjon mener du systemet kan innhente automatisk, og hva annet kunne du tenke deg at systemet gjorde automatisk?

Tann nr, ellers har eg for lite kunnskap om kva som er mulig allerede....

På bildet (figur 3.3) vises en tabell som indikerer hvordan informasjon kan registreres. Dette skal vise hvor ofte man må registrere informasjon i systemet. Slik det er satt opp nå, kan man i en operasjon sette implantater for en eller flere planlagte konstruksjoner (krone/bro/plate). Kan du si noe om: Hva syntes du om fordelingen av registreringer vist i tabellen? Er det noen variabler som registreres, som du mener gir lite verdi å ha med?

NAV refusjon, er det viktig?

Ekstraksjonsgrunn og årsak til tanntap er litt smør på flesk, bortsett frå ved agenesiar. Men vil gjerne vite når tanntap og kor lenge mellom tap og innsetting.

Er det noen variabler vi ikke registrerer som du mener ville gitt mye verdi?

Ja, om det er 1 eller 2 trinns operasjon, dvs. med tilhelingsdistanse eller dekkskrue, og i så fall når distanseoperasjon?