DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2018

Tenancy Model Selection Guidelines

FREDDY AASBERG PIPIRS

PATRIK SVENSSON

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Authors

Freddy Aasberg Pipirs <pipirs@kth.se>
Information and Communication Technology
KTH Royal Institute of Technology

Patrik Svensson <patsven@kth.se>
Information and Communication Technology
KTH Royal Institute of Technology

Place for Project

Stockholm, Sweden

Examiner

Mira Miroslawa Kajko Mattsson
KTH Royal Institute of Technology

Supervisor

Johan Montelius
KTH Royal Institute of Technology

Abstract

Software as a Service (SaaS) is a subset of cloud services where a vendor provides software as a service to customers. The SaaS application is installed on the SaaS provider's servers, and is often accessed via the web browser. In the context of SaaS, a customer is called *tenant*, which often is an organization that is accessing the SaaS application, but it could also be a single individual. A SaaS application can be classified into *tenancy models*. A tenancy model describes how a tenant's data is mapped to the storage on the server-side of the SaaS application.

By doing a research, the authors have drawn the conclusion that there is a lack of guidance for selecting tenancy models. The purpose of this thesis is to provide guidance for selecting tenancy models. The short-term-goal is to create a tenancy selection guide. The long-term-goal is to provide researchers and students with research material. This thesis provides a guidance model for selection of tenancy models. The model is called *Tenancy Model Selection Guidelines* (TMSG).

TMSG was evaluated by interviewing two professionals from the software industry. The criteria used for evaluating TMSG were *Interviewee credibility*, *Syntactic correctness*, *Semantic correctness*, *Usefulness* and *Model flexibility*. In the interviews, both of the interviewees said that TMSG was in need of further refinements. Still they were positive to the achieved result.

Keywords

SaaS, Cloud, Tenancy Pattern, Tenancy model, Software Engineering

Abstract

Software as a Service (SaaS) är en delmängd av molntjänster där en tjänsteleverantör tillgodoser mjukvara som en tjänst åt kunder. SaaS-applikationen installeras på SaaS-leverantörens servrar, och åtkomsten till applikationen sker oftast via webbläsaren. I sammanhanget av SaaS kallas en kund för *tenant*, vilket oftast består av en organisation, eller i vissa fall enbart av en användare. En SaaS-applikation kan delas in i *tenancy-modeller*. En tenancy-modell beskriver hur en tenant:s data är associerad till lagringsutrymmet på SaaS-leverantörens server.

Efter att ha gjort en förstudie kunde författarna dra slutsatsen att det råder guidningsbrist för val av *tenancy-modeller*. Syftet med denna tes är att tillgodose vägledning för val av *tenancy-modeller*. Kortsiktsmålet är att skapa en guide för val av *tenancy-modeller*. Långsiktsmålet är att tillgodose forskare och studenter med forskningsmaterial. Denna tes tillgodoser en modell för guidning av val för *tenancy-modeller*. Namnet på denna guide är textitTenancy Model Selection Guidelines (TMSG).

TMSG utvärderades genom intervjuer med två personer som jobbar inom mjukvaru-branschen. Kriterierna som användes vid utvärderingen av TMSG var följande: *Trovärdighet hos den intervjuade personen*, *Syntaktisk korrekthet*, *Semantisk korrekthet*, *Användbarhet* och *Modellens flexibilitet*. I båda intervjuerna ansåg de medverkande att TMSG behöver ytterligare finslipning, och de var båda positiva till det uppnådda resultatet.

Nyckelord

SaaS, Moln, Tenancy Pattern, Tenancy model, Mjukvaruutveckling

Acknowledgement

Firstly, we would like to thank our supervisor associate professor Johan Montelius at KTH. He has provided us with great knowledge and guidance about the theoretical parts of this thesis.

Secondly, we would like to thank our examiner associate professor Mira Miroslawa Kajko Mattsson. She has helped us with the theoretical parts of this thesis. She has given us great input of how to create a thesis of high quality.

We would also like to thank our supervisor at CRM Treasury Systems AB, Daniel Svensson. He has provided us with great guidance and help throughout the implementation of the tenancy models, and great knowledge about software engineering.

Contents

1 Introduction

Cloud services are widely-spread today, and statistics points out that the usage will continue to grow.[1] It is also pointed out to become more important within the areas of machine learning, artificial intelligence, and internet of things in a few years.[2] There is a certain type of service within Cloud Services that is called *Software as a Service* (SaaS), where a vendor provides software as a service to the tenants (customers). The SaaS application is run on the vendor's server and the tenants can access it through the Internet.[3] When implementing a SaaS application, a tenancy model must be chosen. The tenancy model decides how a tenant's data is mapped to the storage.[4] At the end of 2016, SaaS represented 25% of the enterprise software market. By 2021 it is expected to represent 40%[5].

1.1 Background

Businesses has utilized the Internet to create new types of services that has not been available before, one of those services is *Software as a Service* (SaaS). A SaaS application is run on the SaaS provider's (vendor) own servers. SaaS allows users to connect to a SaaS application through the Internet, as shown in Figure 1. The SaaS application is often accessed directly in the web browser, but can also be accessed with a dedicated software client. Organizations, which includes users, rent the use of the SaaS application. [3]

A SaaS application can be classified into *Tenancy Models*. A tenancy model describes how a customer's data is mapped to the database on the server-side of the SaaS application. For example, customers can share database or have their own dedicated database. In the context of *SaaS*, a customer is often a organization that is accessing the SaaS application, but it could also be a single individual.[4]

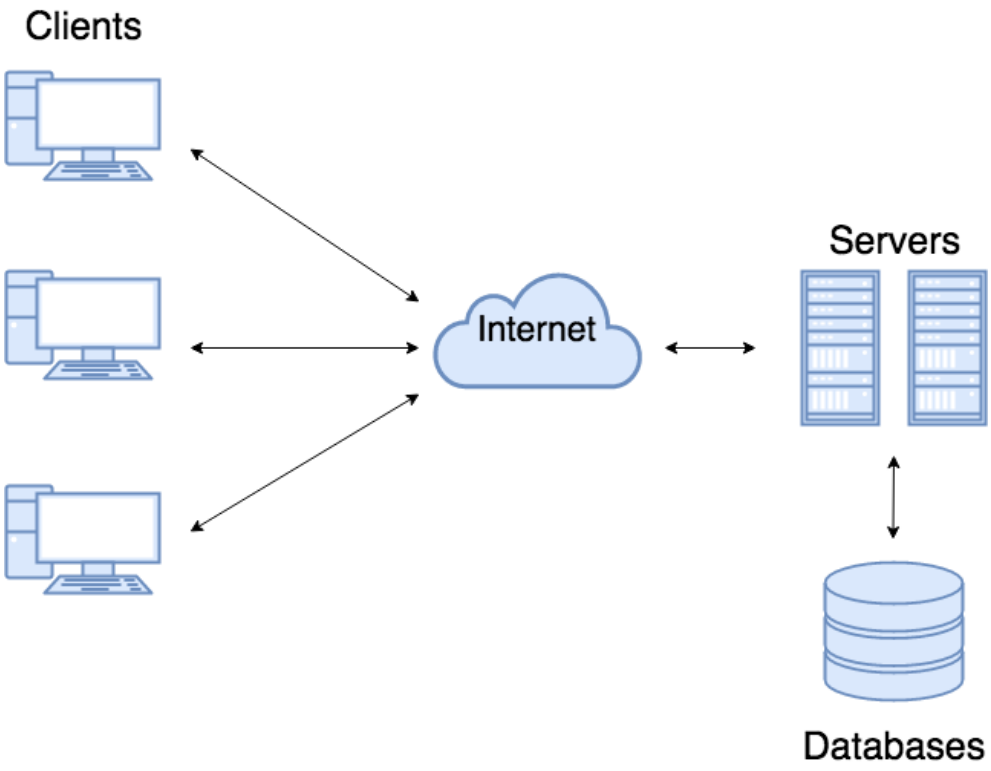

Figure 1: Technical Architecture of a SaaS Application

1.2 Problem

Today, there is a lack of guidance for selecting tenancy models. The decision to select the tenancy model that benefits the SaaS provider's requirements and needs, can be a tedious and difficult task. The change of an already implemented tenancy model to another could cost both money and time. After research on the Internet by the authors, the conclusion as the first meaning states can be drawn. Furthermore, statistics shows that the usage of cloud services and SaaS will continue to grow the upcoming years[1]. This tells that the demand of guidance for selecting tenancy models will probably continue to grow the upcoming years.

1.3 Research Question

The research question that this thesis has defined as a guidance is:

What steps should a guidance model consist of to provide guidance when selecting a tenancy model?

A research question's purpose is to help writers to focus on the right things in their research. This is done by providing a path through the research and writing process.[6]

1.4 Purpose

The purpose of this thesis is to create guidance for selecting tenancy models for SaaS applications. This thesis is aimed to be used by businesses as a support for selecting the right tenancy model that benefit their requirements and needs.

1.5 Goal

The short-term goal of this thesis is to provide businesses with a model of guidelines for selecting a tenancy model that benefit their requirements. The model is called *TMSG*, which is an acronym for *Tenancy Model Selection Guidelines*. The long-term goal is to provide researcher with research material for further study within the area of selecting tenancy models.

1.5.1 Benefits, Ethics and Sustainability

Today, the privacy and protection of data has become more important than ever before. One example of what kind of actions that are taken to protect data and privacy is the *General Data Protection Regulation* (GDPR), which is a law that applies to all members of the *European Union*[7]. One of the requirements that TMSG are taking into consideration is *isolation*, which are about protection of data and privacy. Therefore, we have to write the *isolation* part of this report with extra caution.

In this study, we are constrained to a set of development technologies that are provided by our stakeholder. Therefore, we are enforced to be as independent of the technologies in the report as we possibly can, and ensure that the report is not biased toward the constraint-technologies. This is to ensure that the target group of the report is as broad as possible.

1.6 Methodology

The methodology of a degree project is a fundamental cornerstone for being able to steer the project in the right direction and to reach a correct result. A method is a set of actions or a process for ensuring a result of high quality in a degree project. [8]

When deciding upon methods for a project, the first step is to categorize the project into one of the main methods, *quantitative methodology* or *qualitative methodology*. In quantitative research, experiments are used to measure variables, which can be used to verify or falsify theories and hypotheses. Qualitative research is more about understanding a certain meaning, opinions and behavior to reach hypotheses and theories, or to develop computer systems, artifacts and innovations. [8]

This report is using both a *qualitative-* and a *quantitative research method* as complement for reaching the goal of this study. Hence, *triangulation* is used in the study. An investigation has been made on the implementations of the three different tenancy models and research literature has been used, collected from the literature study. These investigations and research has been used to draw conclusions, hence, it is a qualitative study. Load tests is performed and used together with statistics, which can be classified as a quantitative study. Due to the formulation of theories based on observations of patterns in this study, this study is a *inductive approach*.

Literature study, implementation of tenancy models and load testing was used to collect data. Interviews was used to evaluate the results of the study.

1.7 Stakeholders

The main stakeholder of this thesis is CRM Treasury Systems. CRM Treasury Systems was founded in Stockholm, Sweden in the year of 1984. CRM Treasury Systems is a financial technology company that offer their customers a SaaS-based software solution. The offered SaaS-application is used to manage *treasuring*, which can further be divided into the subparts: *Control*, *Analysis* and *Risk management*.[9] The problem which this report is aimed to solve originates as a proposal from the company CRM Treasury Systems. Other stakeholders of this thesis are the researchers and students that will use it as research material.

1.8 Scope and Limitations

CRM Treasury System AB:s main programming language is C#, which run on the .NET framework. Our supervisor at CRM Treasury Systems is a C# and .NET developer. Therefore, it is a natural decision to choose C# and .NET as our technologies to implement the tenancy models. CRM Treasury Systems has recently begun to host their SaaS applications in Azure, and because of the comfort in hosting applications in Azure, we have decided to use it as our hosting platform. Azure SQL Database is the database technology that is used to implement the tenancy models. The databases are hosted in Microsoft Azure. The load tests are created and run in Visual Studio Enterprise 2017. The RAM usage for each application is adjusted to reflect the usage and resource demands of CRM Treasury Systems current SaaS application. The guidance of TMSG is limited to a set of requirements that is used to compare the tenancy models: *isolation*, *tenancy-cost*, *performance*, *development* and *maintainability*. TMSG is limited to three following tenancy models:

- *single tenant application, single database*
- *multi tenant application, database-per-tenant*
- *multi tenant application, sharded database*

1.9 Outline

The disposition of this report is structured in the following way:

2. *Technical Background.* In this chapter, technical background concerning concepts of tenancy models, SaaS, elastic pools, eDTU and a comparison from Microsoft are given. Finally, the chapter will be tied together with the related work that we have done with the technical concepts.

3. *Research Strategy*. In this chapter, a presentation how this report was conducted is given. The *research phases*, *applied research methods*, *research instruments* and *quality assurance* are presented in this chapter.

4. *Work Process.* In this chapter, the conducted work of this study is presented. The work for each research phase is presented. A presentation of the conducted work of implementing the tenancy models and load testing is given in this chapter.

5. *Comparison Results*. In this chapter, the comparison that has been conducted in this study is presented. This chapter is closed with a summary of the comparison.

6. *Model Review.* In this chapter, the review of TMSG model is presented. The review of the model was conducted by interviewing two professionals from the software industry. The answers from the interviews are presented and evaluated in this chapter.

7. *Discussion.* In this chapter, a discussion is held based on the results in *Chapter 5*, research phases and quality assurance.

8. *Conclusion*. In this chapter, we present conclusions based on the work of this this report. This chapter is presenting the future work that can be used by other researchers to continue our work. Furthermore, recommendation for a tenancy model will be provided for our stakeholder.

2 Technical Background

This chapter gives a theoretical background for this study. *Section 2.1* presents a theoretical description of tenancy models and SaaS. *Section 2.2* presents the tenancy model of *single tenant application, single database*. *Section 2.3* presents the tenancy model of *multi tenant application, database-per-tenant*. *Section 2.4* presents the tenancy model of *multi tenant application, sharded database*. In *Section 2.5*, the concept of elastic pools is described. eDTU is explained in *Section 2.6*. A comparison from Microsoft was helpful in this study, it is presented in *Section 2.7* and is provided in Appendix A. Last in this chapter, *Section 2.8* presents how all the other sections in this chapter are used in the rest of this study.

2.1 Introduction to Tenancy models & SaaS

SaaS is a way providing a complete software service over the Internet, that is rented by the customers from the SaaS provider. In the context of SaaS, the correct terminology of a *customer* is the word *tenant* [1]. Hereinafter, we will refer to customer as a tenant. All of the underlying hardware, middleware, infrastructure, back-end software and application-data are located and provided by the SaaS provider's data center. [3][11] As shown in Figure 2, it describes how a SaaS-solution is used from a high level perspective, where tenants represent the organizations or private entities that are renting the SaaS. Users represents someone that is using the SaaS from inside a tenant, for example, employees from the organization that rents the SaaS.

With SaaS, you do not have to install and maintain the application by yourself as a customer, you let it over to the provider instead. This is putting less weight on the customer, since they do not have to manage complex software and hardware. The SaaS provider will manage access, security, availability

[1] *Tenant*. "A *tenant* is a group of users sharing the same view on an application they use. This view includes the data they access, the configuration, the user management, particular functionality and related non-functional properties. Usually the groups are members of different legal entities."[10]

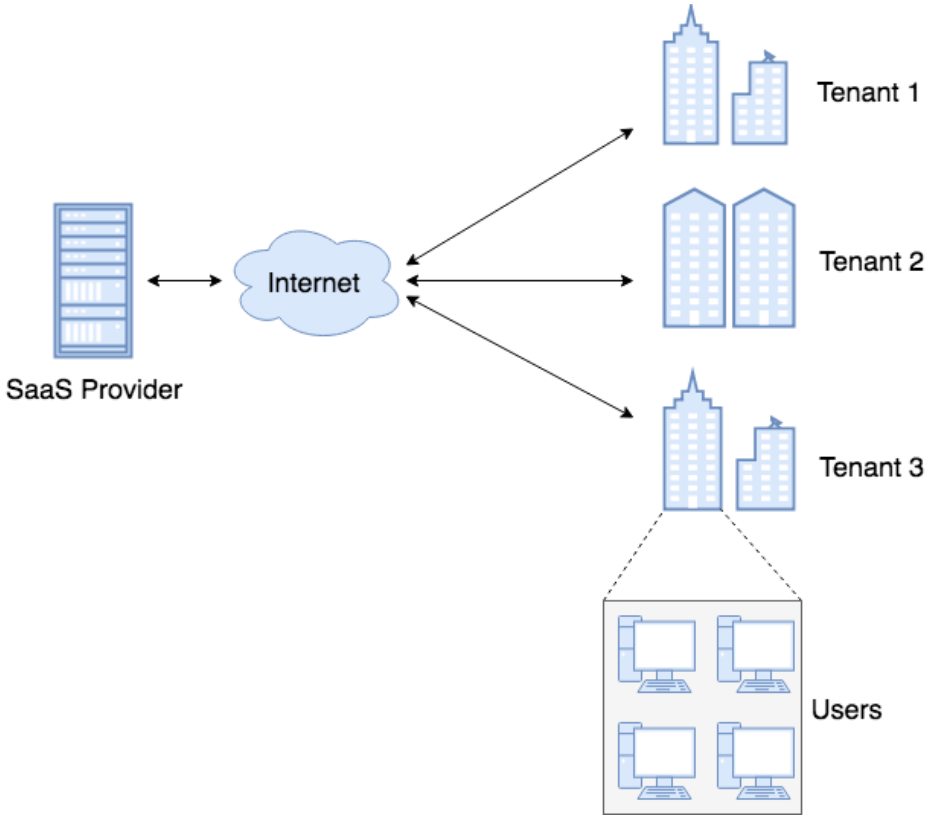

Figure 2: Overview of a SaaS Application

and performance. The provider installs and maintain the SaaS-application while you access it over Internet. [11] The usage of a SaaS solution has benefits compared to applications installed on merely the customer's own machine:

- *Access to advanced software.* Instead of the customer is managing, buying and deploying the needed hardware and software; with SaaS applications it is not needed. With a SaaS application the customer do not need to update, install, maintain any software, hardware or middleware, since it is already assured by the provider of the service.[3]
- *Pay only for what you use.* A SaaS application is automatically scaling up and down depending on the usage. [3]
- *Use free client software.* Almost all SaaS application's parts that run on the client-side, are running directly in the browser. Therefore, there is

no need of downloading and installing specific software to run the SaaS application. [3]

- *Mobilize your workforce easily.* A workforce will be mobilized efficiently, since it is possible for a user to access a SaaS application from anywhere, from any device that is connected to Internet. There is no need to worry to develop applications that runs on different Operating Systems, since it is already assured by the service provider.[3]

- *Access application data from anywhere.* Since the data that the SaaS application uses is stored in the cloud[2], it can be accessed from any machine that has an Internet connection. This will also minimize the risk of data loss in case of broken machine, since it is stored in the cloud.[3]

As mentioned in *Section 1.1*, a tenancy model describes how a tenant's data is mapped to the storage on the server-side of the SaaS application. The choice of tenancy model will not affect the functionality of the application, but it will affect other aspects of the application. Furthermore, the choice of a tenancy model is having a large impact on both design and management of a SaaS application. A decision of changing to a different tenancy model for a SaaS application can be costly in certain cases.[4] This study is focusing on three tenancy models, which are described below in *Section 2.2*, *Section 2.3* and *Section 2.4*.

2.2 Single Tenant Application, Single Database

The *single tenant application, single database* is a tenancy model where each tenant gets a dedicated application installed on the SaaS provider's server, look at Figure 3. All of the dedicated installations of the application have a standalone database connected, which no other installations can access. [4]

[2] *Cloud storage.* is a storage model where the data is stored on remote servers and accessed trough Internet. The storage provider takes care of maintenance, operation and management.

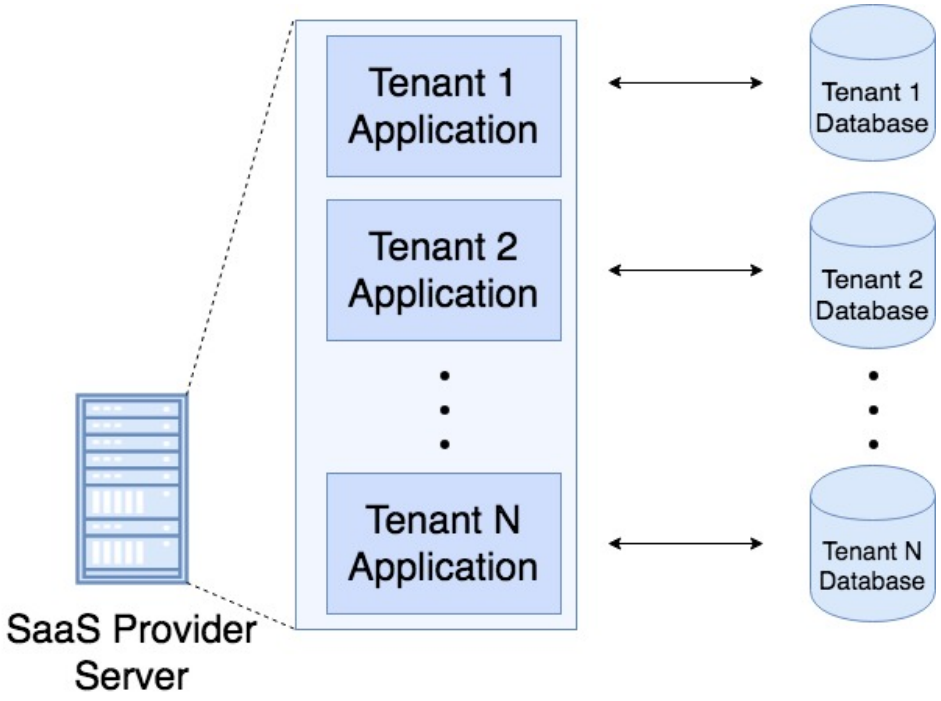

Figure 3: *Single tenant application, single database*. *N* is an integer that represents the number of tenants

Since every database that belongs to a tenant is connected to the dedicated installation to the corresponding tenant, this model provides the greatest isolation.[4] The connection string to the database that belongs to a tenant is hard-coded in the setup in the corresponding application.

2.3 Multi tenant Application, Database-per-tenant

The *multi tenant application, database-per-tenant* is a tenancy model where one single application is installed at the SaaS provider's server, look at Figure 4. For each tenant, a dedicated database is connected to the application. A database belonging to a specific tenant can exclusively be accessed by that tenant. Furthermore, there is need for a *Catalog Database* in *multi tenant application, database-per-tenant*. The catalog database holds essential information about each tenant, like connection string to database and display name.

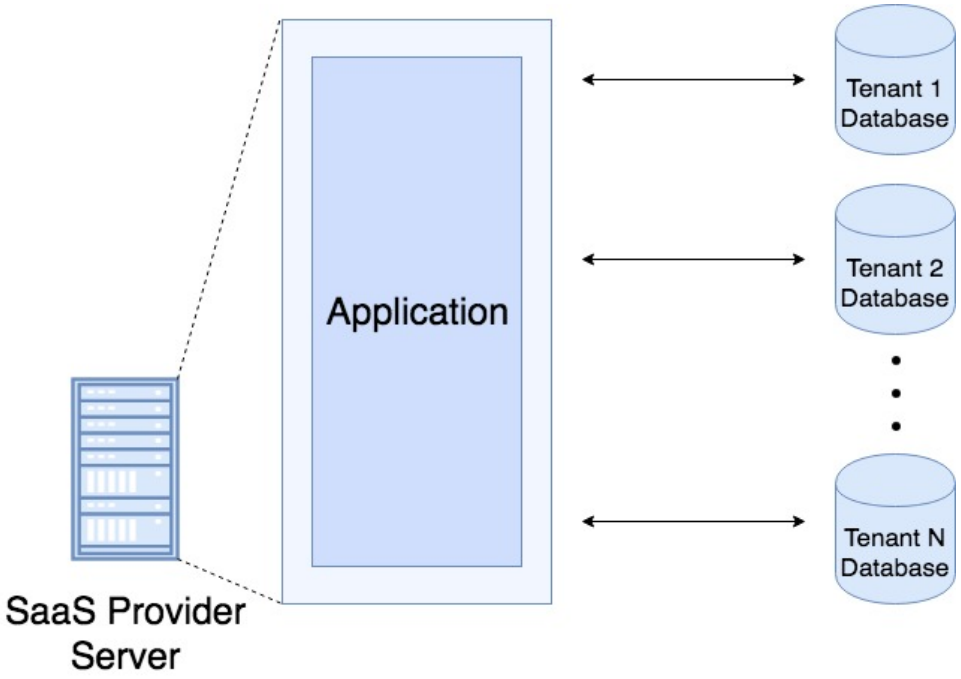

Figure 4: *Multi tenant application, database-per-tenant*. *N* is an integer that represents the number of tenants

2.4 Multi tenant Application, Sharded Database

The *multi tenant application, sharded database* is a tenancy model where one single application is installed at the SaaS provider's server, like in *multi tenant application, database-per-tenant*, look at Figure 5. The difference between *multi tenant application, database-per-tenant* and *multi tenant application, sharded database*, is that an installation of a database can be shared among tenants.[4] A sharded database is when several physical distributed databases is logically acting as one database with the same schema. A "shard" is the physical database, which has the same schema as all of the other shards, but its own distinct subset of data. In *multi tenant application, sharded database*, one tenant's data must be stored in the same shard. When working with shards, a shard key is used to determine which data that should be placed in each shard.[12] In our case, we will map data belonging to a tenant to the shard that holds the data of the tenant, therefore we use a tenant-id (which is unique) on the data as a shard key.

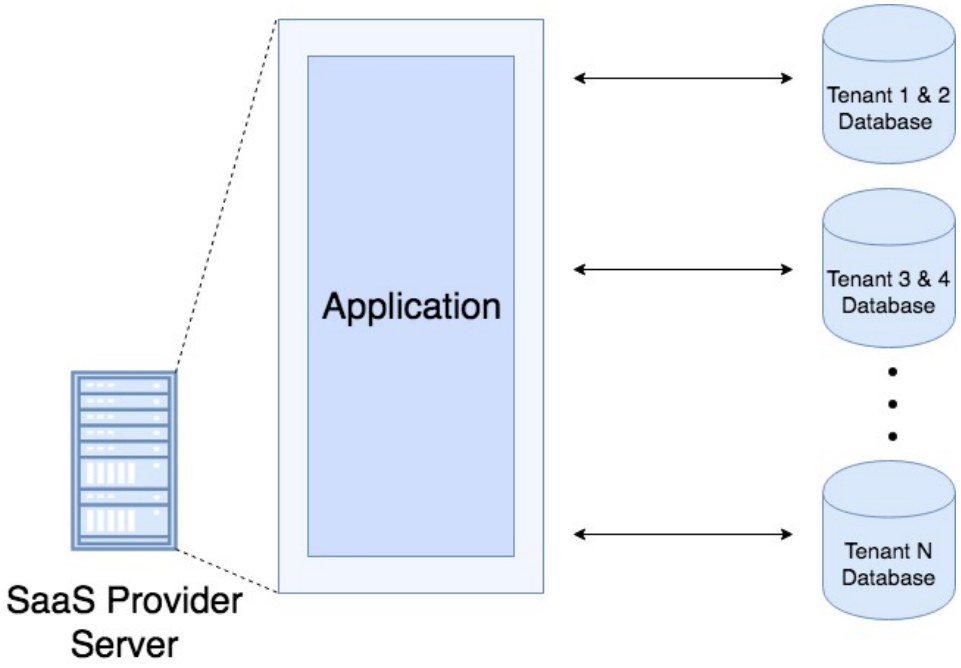

Figure 5: *Multi tenant application, Sharded database. N* is an integer that represents the number of tenants

2.5 Elastic Pools

A common way of providing tenants with data-storage, is to give each tenant a dedicated database that has some resources allocated for utilization, for example: IO- and CPU-time. But every different tenant has a different unpredictable usage pattern. Hence it is hard to predict the resource requirement for each tenant. One way to solve this problem is to use an elastic pool. Elastic pools gives the possibilities to group together databases into a ”pool”. Instead of giving each database a set of resources, you give the pool a set of resources that all the databases in the pool are sharing.

The usage of elastic pools can lower the cost on resources because the databases is sharing the resources and it is unlikely that several databases gets a heavy workload at the same time. [13]

2.6 eDTU

Elastic Database Transaction Unit (eDTU) is Microsoft Azure's way of measuring the resources allocated for an elastic database-pool. eDTU is calculated from a combination of hardware specific resources, more exactly: CPU, I/O and storage.[4].

2.7 Microsoft's Comparison

Microsoft has created a comparison of tenancy models. This comparison is only a supplement in this study. The comparison from Microsoft are based on the following criteria: *Scale*, *Tenant isolation*, *Database cost per tenant*, *Performance monitoring and management*, *Development complexity* and *Operational complexity*. The table of the comparison from Microsoft is provided in *Appendix A*.

2.8 Usage in Study

The concept tenancy models and SaaS are essential to be able to understand this study. The concepts of *Section 2.1*, *Section 2.2*, *Section 2.3* and *Section 2.4* are general for this study, and will be used extensively. All databases that are used in the implementation of the tenancy models, is grouped together in an elastic pool. Furthermore, the performance of an elastic pool is measured in eDTU. The concepts of *Section 2.5* and *Section 2.6* are factors that could affect the result of the performance tests in this study; hence, it is important to understand the concepts to be able to analyse and discuss the results. The comparison from Microsoft in *Section 2.7* was used in our comparison of tenancy models.

3 Research Strategy

This chapter is presenting the research strategy of this thesis. In *Section 3.1*, the research methods that has been used in the study are presented. *Section 3.2* is presenting the phases that was used to reach the goal of this study. *Section 3.3* is presenting the research instruments that was used in the study. *Section 3.4* presents the actions that was taken to assure the quality of this study. In *Section 3.5*, the sampling method used in this study is presented. Finally, *Section 3.6* presents the experiences gained in this study. Look at Figure 6 for an illustration of the research strategy.

3.1 Research Methods

During this study, research method has been applied to facilitate the process of reaching the goals of the thesis. The research method that was applied was: *Quantitative Research*, *Qualitative Research*, *Triangulation*, *Induc-*

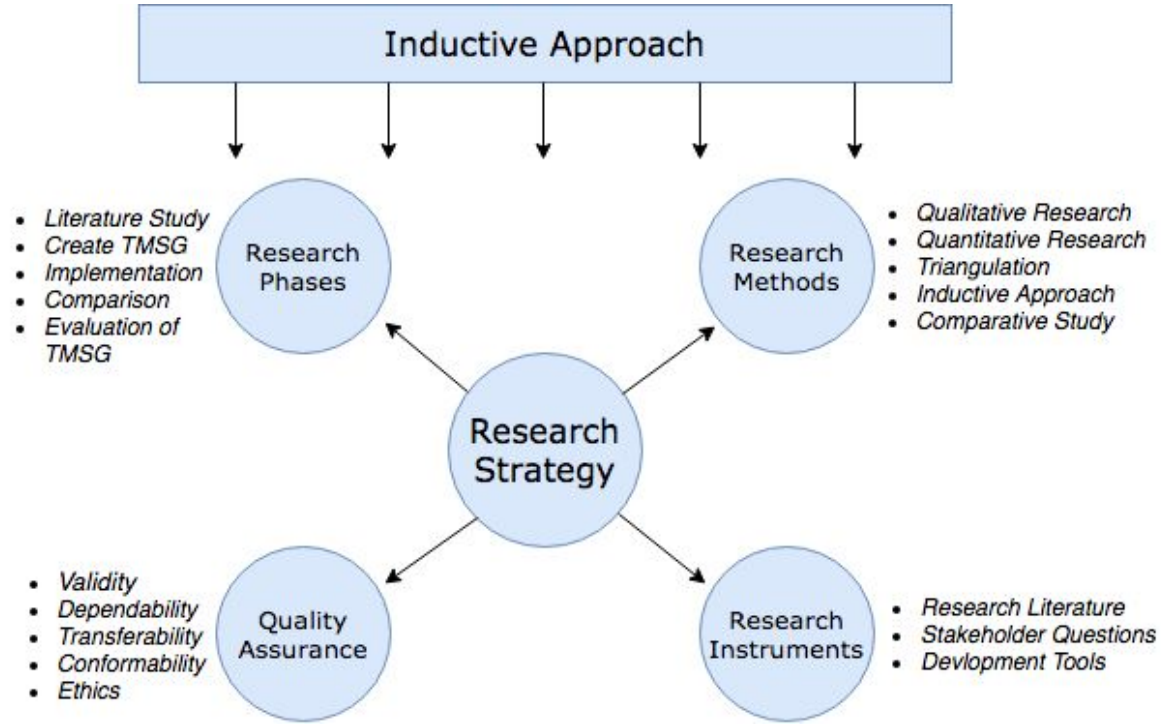

Figure 6: Depiction of *Research Strategy*

tive Approach and *Comparative Study* The following subsections are explaining how each of the applied research method was used throughout this study.

3.1.1 Quantitative Research

Quantitative research method is used to verify and falsify theories and hypotheses by tests and experiments of measuring variables. Quantitative research's usage of tests, experiments and measuring of variables can also be used together with functionality of computer systems. A quantitative research is used together with a hypothesis that must be numerically measurable. To use this method a large set of data is needed.[8]

In this study, the load tests that was performed in the *comparison*-phase is the activity that makes this study a quantitative research. The load tests are using *time* as a measurement of the average response time for a request to a SaaS application.

3.1.2 Qualitative Research

Qualitative research method is about understanding meanings, opinions and behaviors to reach hypotheses and theories, or to develop computer systems, artifacts and innovations. Compared to quantitative research, this method is using a smaller set of data to reach a sufficient result that is reliable. The collection of data in this method often continues until saturation of data is reached. [8]

In this study, qualitative research has been used as the literature study and implementation of tenancy models has worked as a data input to the comparison, where hypotheses and theories have been made. Furthermore, the literature study has been used to create the the TMSG model.

3.1.3 Triangulation

By combining a *qualitative research* and *quantitative research* methods as a complement to get a complete view of the research area and situation. This is called *triangulation*, it is used to ensure the correctness of the result by increasing the validity and credibility. Even though the triangulation is using several methods, they are often used by applying one method at the time.[8]

We have applied triangulation by using both quantitative research and qualitative research. Triangulation is applied in the *comparison*-phase to ensure the correctness of the output of the phase, which is a partition of the results in this report. The quantitative research and qualitative research has been applied one at the time, hence, independently of each other.

3.1.4 Inductive Approach

The inductive approach is when theories and propositions are formulated with alternative explanations from patterns and observations. An inductive approach goes together well with development of artifacts. Data is often collected with methods that belongs to qualitative research. The data is analysed to reach understanding for a phenomena and to get new aspects of the phenomena.[8]

This study has been conducted with an inductive approach. After been conducting the phases of this study it is possible to draw theories and propositions. The data that had been analysed are from the literature study, implementation of tenancy models and load tests.

3.1.5 Comparative Study

In a comparative study, there are two or more cases that are examined and compared. The cases that are compared have to be similar in some aspect. The cases that are compared also have to differ in some way, otherwise, the comparative study will not output any value. When a comparative study is

performed, it is necessary to decide which interesting aspects, properties or attributes that is noted and record the cases that are compared. Comparative studies are often divided into *descriptive comparison* and *normative comparison*. Descriptive comparison is describing and sometimes explaining the invariance of the cases. Normative comparison differs from descriptive comparison in the way that it is evaluating the cases based on criteria, and the aim is to point out the best case. [14]

Comparative study was used for this thesis to compare the tenancy models. The type of comparative study that is used is normative comparison, due to the evaluative characteristics of the comparison among the tenancy models. The comparative study was performed in the *comparison*-phase.

3.1.6 Alternative Research Methods

The reason why qualitative research was not considered as the research method for the load test, was mainly because of its measurable nature. In qualitative research uses statistics and measurement of variables to verify or falsify a hypotheses or theories. Since the load tests are measured in the variable of time and statistics was used to analyse the data, qualitative research was a natural choice. Furthermore, large amount of data for a qualitative research was accessible by performing the load tests with Visual Studio.

When it comes to studies that concerns understanding meanings, opinions and behaviors, quantitative research do not provide enough understanding and interpretation of the context of the subject. [15] Hence, a quantitative research was not suitable for the work regarding TMSG model, implementation and comparison (excluding load tests).

3.2 Research Phases

One of the cornerstones for the research strategy is the *research phases*. The research phases are the steps that is performed to reach the goal of this

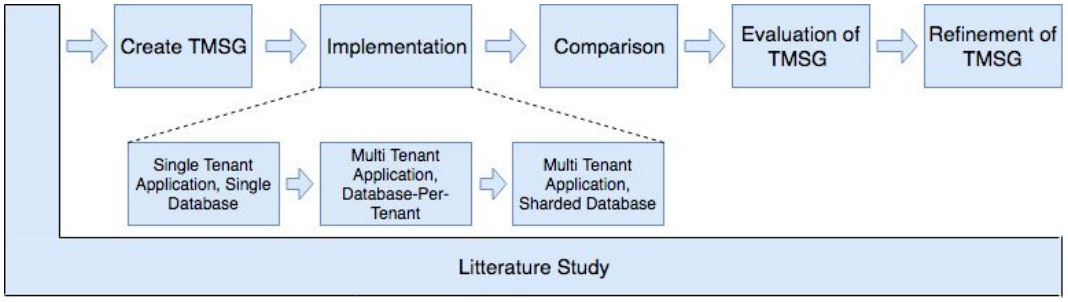

Figure 7: This image depicts the method phases of this thesis

thesis. This section is describing the research phases that were conducted in this study. The phases are conducted in the following sequence: *Create TMSG*, *Implementation*, *Comparison*, *Evaluation of TMSG*, *Refinement of TMSG*. Furthermore, the *Implementation*-phase includes thee sub phases in the following sequence: (1) *single tenant application, single database*, (2) *multi tenant application, database-per-tenant*, (3) *multi tenant application, sharded database*. A quick summary of the phases are shown in Figure 7.

3.2.1 Literature Study

This study was commenced with a literature study. The purpose with a literature study is to supply us with sufficient knowledge mostly in the area of SaaS, selection of computer systems, tenancy models and in the areas around that concerns tenancy models, to be able to follow through this method and to reach a thesis achievement. The literature study also had the purpose to give us an understanding of the problem, and to find similar studies to ours that can be used to reach the goal of the thesis. The literature study took place continuously throughout this study, from beginning to end. But more weight where be put on it in the beginning of the study.

The information and knowledge that has been obtained in the literature study has been collected from databases that *Kungliga Tekniska Högskolan Library* (KTHB) was subscribing to, IEEE [16], ACM [17], and Scopus [18].

Articles in the following areas were read: *Multi-tenancy*, *Single-tenancy*, *Databases*, *Tenancy models*, *SaaS*, none of which were older than five years. Furthermore, a thesis within *selection of computer systems* was read. Other more general search channels such as Google was used to collect documentation, SaaS solutions, tenancy models, selection of computer systems, books and information from companies that makes cloud services.

3.2.2 Create TMSG

This phase is where the TMSG model was to be created. This phase identifies the steps that would be needed to take by a business to select the right tenancy model.

The output from this phase was a selection guide that has defined steps which are recommended take in a selection of a tenancy model, that can be presented with a figure, as shown in Figure 8. Before this study was commenced we already knew that a comparison of tenancy models would needed to be summarized; this summary was created in the *comparison*-phase within this study. All the steps of the TMSG model, which are the output of this phase, are completed and defined after this phase; except for the *comparison*-phase of the TMSG model.

3.2.3 Implementation

In the *implementation*-phase is the phase when we implemented the three tenancy models. The *implementation*-phase consists of the following sub-phases, where each sub-phase represents the implementation of the corresponding tenancy model:

- Single tenant application, single database
- Multi tenant application, database-per-tenant
- Multi tenant application, sharded database

We begun with implementing the *single tenant application, single database*. Afterwards of implementing *single tenant application, single database* the implementation was modified to a *multi tenant application, database-per-tenant*. Finally, the *multi tenant application, database-per-tenant* was modified to the tenancy model of *multi tenant application, sharded database*.

We have received continuously help and guidance from our supervisor at CRM Treasury Systems, Daniel Svensson. Daniel has provided us with great help in our development of the three applications that was developed during this study, and how to perform load tests on them. Daniel has helped us with the practical parts of this thesis. During the study, we have had continuous contact with Daniel during the *implementation*-phase; this was to ensure that the practical parts of this thesis was proceeding in the right direction.

3.2.4 Comparison

To support the selection of tenancy models in TMSG, a comparison among the tenancy models of *single tenant application, single database*, *multi tenant application, database-per-tenant* and *multi tenant application, sharded database* was conducted. The comparison among the three tenancy models are based on five different criterion, *isolation*, *tenancy-cost*, *performance*, *development* and *maintainability*. The following paragraphs explains how to evaluate a tenancy model based on a certain criterion. Each criterion, except for *performance*, has associated questions that is used by the author to evaluate the criterion.

Isolation. The isolation was evaluated by a literature study in combination with the experience gained by implementing the three different tenancy models. There are two questions defined for isolation:

- *Is it possible for one tenant to read or overwrite another tenant's data?*

- *Is it possible to guarantee that a tenant is reading or writing to the correct database?*

Tenancy-cost. The three questions belonging to tenancy-cost are listen below. The first question is reached by calculating the increase of RAM when adding a tenant. The first question is reached by calculating the increase of disk when adding a tenant. The third question is reached by converting the increase of RAM and disk from the first goal into a monetary cost.

- *What is the cost of adding a tenant when the cost is measured in RAM?*
- *What is the cost of adding a tenant when the cost is measured in disk?*
- *What is the cost of adding a tenant when the cost is measured in monetary terms?*

Performance. The test and evaluation of the performance was performed by hosting all the applications in Azure, and applying a load test on them one by one. No questions were defined to performance due to the reason that the criterion will be analysed with statistics. The load tests were performed by loading specific pages on the applications that implements one of the different tenancy models with *Hypertext Transfer Protocol* (HTTP) GET requests. There are four different tests that are applied to each of the tenancy models, which are the following:

1. *Home page with cache*
2. *Home page without cache*
3. *Course page with cache*
4. *Course page without cache*

The access of the ”Home page” is meant to GET a website with very little database access or no database access and load it into the browser. The access of the ”Course page” is meant to GET a website that is dependent of a larger set of data to retrieve from a database. The ”with cache” or ”without

cache" decides if the data from the database will be cached or not. Every test was run with a load of 2, 4, 8, 16, 32 and 64 simultaneously virtual users. The virtual users loaded the application with HTTP-requests constantly during 2 minutes for each test.

Development. The development criterion was evaluated by implementing the three different tenancy models. The implementation gave enough experience to evaluate both development criterion for the three different tenancy models. To help evaluating the development criterion, we have defined two questions:

- *What is the development complexity of implementing the tenancy models?*
- *What is the continuous development work during the life cycle of the tenancy model?*
  - What is the complexity of changing queries?
  - What is the complexity of changing schema?

Maintainability. The maintainability was evaluated by implementing the tenancy models, and seeing how they were operated in Azure. There is only question defined for maintainability:

- *What is the complexity of updating an application to a newer version?*

3.2.5 Evaluation of TMSG

In this phase, the criteria on how to evaluate the TMSG model has been defined. The literature study gave a result of a similar study to ours, *Select Database (SeDB) - A Database Selection Process Model*[15], where a process model that helps to select the right database had been the result of the thesis. Due to the similarities in of the theses, and the positive outcome of the thesis, we have adopted some of the criteria for evaluation. The criteria

were evaluated with interviews with professionals from the software industry. One possibility would be to evaluate TMSG by comparing it to a similar model that has the same purpose. But since no similar model to TMSG was found during the literature study it was not feasible. The TMSG model was evaluated with the following evaluation criteria:

1. *Interviewee credibility*. When interviewing people about a certain subject, it is important that the interviewee is trustworthy. A credible interviewee is one that has the quality of being trustworthy and believable. This evaluation criterion is to ensure the trustworthiness of the answers. [15]

2. *Syntactic correctness*. When talking about syntactic correctness, we refer to the logical phases of TMSG. This evaluation criteria ensures that all the relevant phases are included, no unnecessary phases are included and that the sequence of TMSG is correct. [15]

3. *Semantic correctness*. When using TMSG it is important that the model is accurate described and explained. If TMSG is semantic correct, the user of the model should understand the relevancy and applicability of its phases. This criterion ensures that TMSG can be understood by a user. [15]

4. *Usefulness*. Usefulness means that something is applicable and of practical use. When using TMSG it is important that the model is useful to the user. This evaluation criterion ensures that TMSG is useful and practical. [15]

5. *Model flexibility*. Flexibility is about the ability of adapting and adjusting to external changes. When using TMSG it is important that the model is adaptable, adjustable and reusable. This evaluation criterion ensures that TMSG's general usability and to make sure it can be transfered to various contexts. [15]

3.2.6 Refinement of TSMG

This phase is refining the TSMG model that was defined in the phase of *Create TMSG*. This phase uses the feedback from the *Evaluation of TMSG*-phase to refine the TMSG model.

3.3 Research Instruments

Research instruments are used to collect, measure and analyse data that is related to thesis[19]. The following subsections explains the research instruments has been provided as an input for the research strategy of this thesis.

3.3.1 Research Literature

This study was commenced with a literature study. The research literature that was found provided an input to all the phases in the study. A notable finding in this study was a comparison of tenancy models from Microsoft. The comparison from Microsoft can be found in Appendix A.

3.3.2 Stakeholder Questions

Some of the comparison criteria definitions, and questions of this study was influenced by a list of questions and requirement provided by CRM Treasury Systems. The list included several questions and requirements regarding selection of tenancy models that CRM Treasury Systems wanted to be answered:

Requirements

- Tenant A should not be able to read or write tenant B's data.

Isolation

- Is it possible to assure that the tenant connects to the right database?

Tenancy-cost

- Are there any differences in RAM usage among the tenancy models, provided the same number of active tenants?
- Are there any differences in disk usage among the tenancy models, provided the same number of active tenants?

Development

- How complex is the continuous development?

Maintainability

- How do to solve updates, to newer versions, in the different tenancy models?

3.3.3 Development Tools

The *development tools* that was used as research instruments were the following: *programming language C#*, *.NET Framework*, *Azure* and *Visual studio enterprise 2017*. These development tools were used in the *implementation* phase of this study. C#, .NET Framework and Visual studio enterprise 2017 were used to implement three different tenancy models. Azure was used to host the implemented SaaS applications. Furthermore, Visual studio enterprise 2017 was also used to run load tests.

3.4 Quality Assurance

In the research strategy, *quality assurance* is the component that validates and verify the research material. If a qualitative research with an inductive approach is used, it is necessary to apply and discuss the following aspects, *validity*, *dependability*, *confirmability*, *transferability* and *ethics*. [8] These aspects are presented in the following subsections.

3.4.1 Validity

In qualitative research, validity is about making sure that the research has been conducted accordingly to the existing rules. The respondent should be able to validate and confirm that the results are correctly understood.[8]

3.4.2 Dependability

Dependability is the process of judging the correctness of the conclusion, by using auditing. [8] In quantitative and qualitative research the dependability can be supported by the grade of repeatability.[15]

3.4.3 Transferability

Transferability is to create rich descriptions. These descriptions can become a database for other researchers.[8]

3.4.4 Confirmability

Confirmability is about confirming that the research has been performed in good faith without personal assessments that have affected the results of the thesis.[8] In qualitative studies, confirmability corresponds to objectivity and refers to the degree the results can be confirmed by others[15].

3.4.5 Ethics

Ethics is the moral principles in planning, conducting and reporting results of a research study. Ethics is covering protection of participants, maintenance of privacy, avoiding coercion and to have consent in written form, and treating material with confidentiality. [8]

3.5 Sampling Method

The selection of interviewees for evaluation of TMSG was based on *convenience sampling*. Convenience sampling means that the population that the

sample is take from, are conveniently available. In our sampling, two interviewees was chosen from CRM Treasury Systems because of the availability, hence it is convenience sampling. Other sampling methods was not considered due to time restrictions.

3.6 Experiences Gained

This study has provided much experience. In the beginning of the study we planned to use more criteria, and also larger scope of the criteria used in TMSG. We realised that the scope was to wide by reading literature and advices from professionals. The time to implementing the tenancy models extended the initial expected time for the implementation. This unexpected event has provided us with more experience in estimating a required time for development. Lastly, we encountered problems with calculated cost for adding a tenant for a specific tenancy model. We tried to solve this problem for a long time, but at last we gave up. This has given us the experience that it is sometimes accepted to take a step back when encountered with a problem that can not be solved.

4 Work Process

In this chapter, the work process of this study is presented. In *Section 4.1* the conduction of work in the phase of creation of TMSG is presented. The implementation work of the tenancy models is presented in *Section 4.2*, *Section 4.3*, *Section 4.4*, and *Section 4.5*. In *Section 4.6*, the work of the comparison phase is presented. Due to the comprehensive information regarding the work of load testing, it is explained in *Section 4.7*. The work of the evaluation phase, which is about the interviews, is presented in *Section 4.8*. Finally, the last section of this chapter is *Section 4.9*, where the refinements of the TMSG model, based on the interviews, is presented.

4.1 Creation of TMSG

In this subsection, the work of creating the TMSG model is presented. The work of creating the TMSG model was supported by research literature that was provided from the literature study-phase. During the literature study another similar study was found, which provided support in this phase, *Select Database (SeDB) - A Database Selection Process Model* [15]. The TMSG model consists of four consecutive phases that is aimed to help businesses selecting a tenancy model. The TMSG model is built up with the following four phases: *analyse business*, *transform into criteria*, *select tenancy model* and *evaluation of selection*, this is illustrated in Figure 8.

The TMSG model is based on the assumption that the business that uses it has already decided that they want to implement a tenancy model. Hence, the question that TMSG model has capability to answer is: *Which tenancy model would benefit the business the most, assuming that one will be implemented?*

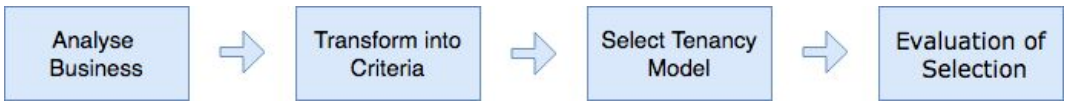

Figure 8: Overview of TMSG phases

4.1.1 Analyse Business

This step is used to analyse the business that is applying the TMSG model. This analysis is used to get insights and understanding of all the systems that are affected and interconnected by the SaaS application, and understanding the needs of the business. Since all businesses are different, one tenancy model that suits one business might not suit another business. If this step is done properly, the chances of selecting the right tenancy model is higher.

By reading *Select Database (SeDB) - A Database Selection Process Model* [15], knowledge was gained in how a life cycle of a system can cause ripple effects in systems that surrounds and interconnects with it. Since the SaaS application that belongs to a business is not living an isolated life, for example tenants are using the system, developers are changing the system; it is important to detect and analyse the systems that are affected by the SaaS application. The TMSG model is applied in the first phase of *tenancy model selection* in a SaaS application's life cycle, as is shown in Figure 9. The systems that are affected by a change of a SaaS application (by changing tenancy model) are called *echo system*. Even though TMSG model is playing a small part of the life cycle of a SaaS application, it is affecting the selection of a tenancy model. The outcome of phase one is affecting the succeeding phases. 9, shows the life cycle of a SaaS application with the surrounding echo system it is affecting. The small arrows in the echo system points out different aspects that a SaaS application could affect. Yet, it is important to understand that there could be more or less aspects in the echo system that a SaaS application affects, the Figure 9 is just an example.

A value proposition offered by a business is the set of benefits and values it promises to deliver to a customers to satisfy their needs. All customers can be divided into marketing segments.[20] In our case the tenants acts as the role of a consumer. By using the TMSG there is a risk of changing the value

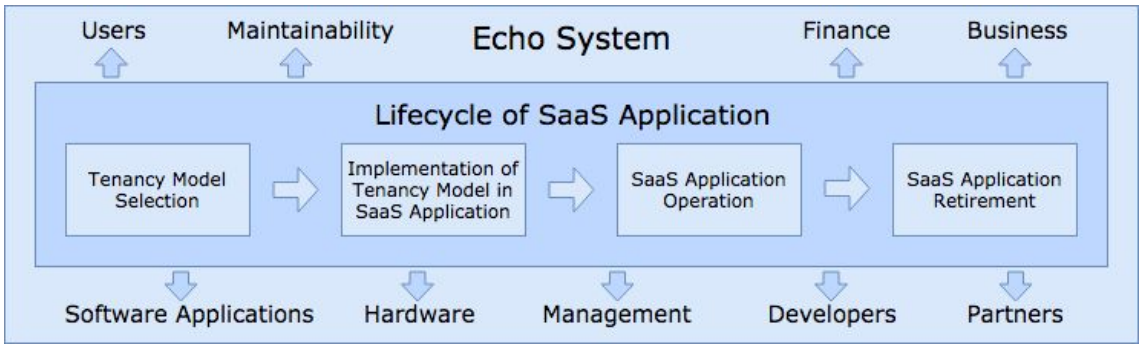

Figure 9: Overview of life cycle and echo system of a SaaS application

proposition of the business that is using the model (if it is not conducted correctly), assuming that they already have a SaaS application and wants to change tenancy model, look at Figure 10. It is necessary that the intended marketing segment and customers is analysed and fully understood. The profile of an intended marketing segment or customer should contain the following:

- **Who** - relevant details of the marketing segment or customers that includes demographics, roles and responsibilities.[21]
- **Where** - The type of business and organizations where the marketing segment or customers can be found. [21]
- **What** - the behavior that is required to achieve the desired value. This desired value could for example be: certain performance of the SaaS application, protection of data or an user interface that's easy to understand.[21]
- **Why** - The problems that is related to the solution that the value proposition is offering.[21]
- **How** - A detailed, expected use of the product offering. This could for example be a sequence of events that is involved in the use of the product offering.[21]

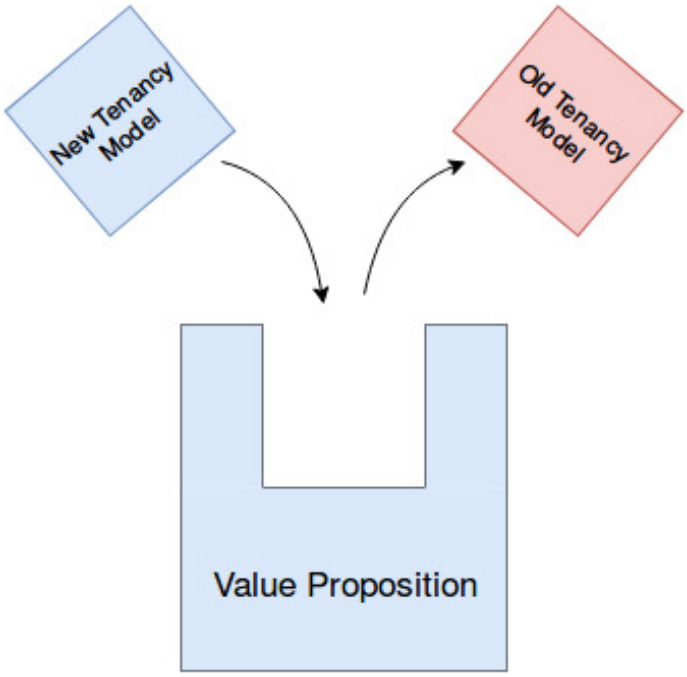

Figure 10: Illustration of value proposition change

To do an analyse like the one above can give much information about the marketing segment and customers, which will help to select the right tenancy model. An analyse like this strongly relates to the *Users* and *Business* in the echo system, shown in Figure 9.

There's aspects of the echo system that is affecting the internal parts of the business that are important to analyse, like *maintainability*, *finance*, *business*, *management* and *developers*. For example, in the finance aspect, the company might be in a financial crisis, which is reasonable to take into consideration if the company want to implement a new tenancy model.

4.1.2 Transform into criteria

This phase is transforming the analysis from the previous phase into graded criteria. This transformation of the input from the previous phase into criteria is making the selection of a tenancy model in the next phase *select tenancy model* feasible. Hence, the output from this phase are graded criteria

which are based on the needs of the user of TMSG model, which are used to compare the tenancy models in the next phase.

Based on the analysis from the previous phase, the user of the TMSG needs to use it to evaluate each of the criteria in the list below, one by one. Based on the evaluation of each criteria, the user of TMSG should be able to determine the appropriate relevance-level of the criterion for the business. The criteria in the following list are providing a description of how the user should interpret each of them. Each criterion is also followed with questions that can be useful to consider for the user. Some of these questions are taken or rewritten from *Subsection 3.2.4*.

Isolation. This criterion is about how protected the tenant's data is. The data that the SaaS application is storing can be sensitive, therefore it is necessary to assess the protection level that is needed for it.

- *In what extent does the tenant's data need to be protected?*
- *In what extent do we need assurance that a tenant is reading or writing to the correct database?*
- *In what extent do we need assurance that one tenant cannot read or overwrite another tenant's data?*

Tenancy-cost. This criterion is about financial restrictions of the business that is providing the SaaS application. The addition of a tenant to a SaaS application has a cost. This price increase comes from the additional hardware that might be needed.

- *In what extent are the financial constraints for adding a tenant?*

Performance. This criterion is about the performance of delivering content to the tenants. When it comes to the definition of performance, it is measured in time.

Hence, performance is about delivering content from the SaaS application to the tenant as fast as possible.

- *How important is the performance to able to satisfy the need of the tenant?*

Development. This criterion is both about the complexity of developing the SaaS application and the complexity of continuous development during the application's life cycle.

- *Are there constraints on the development complexity of implementing the SaaS application?*
- *Are there constraints on the continuously development work during the life cycle of the SaaS application?*

Maintainability. This criterion is about the maintainability of the SaaS application. Maintainability for software can be costly and time-consuming. In this case, the maintainability is limited to updating a SaaS application to newer versions.

- *Are there any constraints/limits on how many updates of SaaS applications that are imposed?*

4.1.3 Select Tenancy Model

In this phase, the selection of a tenancy model is conducted. Based on the input from the previous phase, that includes analysed criteria, the selection of a tenancy model is feasible. If the previous phases have not been conducted correctly, the chances of selecting the right tenancy model for the business could possibly be inaccurate.

The selection of a tenancy model is done by comparing the criteria from the previous phase against the comparison that this thesis has produced. The comparison is provided in *Appendix C*.

4.1.4 Evaluation of Selection

This phase is to assure that the correct tenancy model has been selected. If the previous phases of the TMSG model has been misconducted, there is a risk that the selected tenancy model does not suit the business. Since the TMSG model is applied before the actual implementation of the tenancy model, it is highly important to ensure that the right tenancy model is selected. The implication of selecting a tenancy model that is not suiting the business might be loss of money, time and trust from tenants.

Since all the previous phases to this point has been conducted one-by-one with no evaluation of the current state of the TMSG model during the process; the phases of the model could have been performed incorrectly or insufficient. This might lead to delusion that following these phases of the model always leads to the best result. Therefore, it is highly important to rewind and evaluate the previous phases before implementing the selected tenancy model. The following questions can be useful to evaluate if the right tenancy model has been selected:

- *Is the selected tenancy model reasonable for our business?*
- *Is there a risk that the previous phases were misconducted?*

4.2 Implementation of Tenancy Models

The implementation is based on three steps, which is presented in *Subsection 3.2.3*. The *single tenant application, single database* is based on an example SaaS application from Microsoft, *Getting Started with Entity Framework 6 Code First using MVC 5* [22]. The decision of choosing an already existing SaaS application was mainly because of the time limit set for development, which was set to five weeks. The reason why *single tenant application, single database* was selected as the foundation of all the three steps in the implementation was because CRM Treasury Systems is using the tenancy model by the time of writing this thesis. Furthermore, it was a natural choice

of tenancy model to begin with, since CRM Treasury Systems were to consider changing tenancy model if it is beneficial. A hyperlink to the repository with all the source code used in this study is provided in *Appendix D*.

4.3 Single tenant Application, Single Database

Our first task was to get familiar with the example application. The application from Microsoft is a fictional example of a university SaaS application. The example application includes the following functionality:

- *Student administration.*
- *Editing existing courses.*
- *Creating new courses.*
- *Instructor assignments.*

4.3.1 Home page

This is the page where tenants are entering the site. The name of the university is static data on the page, hence it is not retrieving any data from a database. Because *single tenant application, single database* is dedicated for one tenant, which implies that there is less dynamic data and complexity in the code to make it work for a specific tenant (which is needed when tenants are sharing SaaS application) .

4.3.2 Course page

This page has more functionality, the courses that are listed are retrieved from a database. This data is retrieved from the database each time a tenant makes a request from the page, if it is not cached. The course page has more complexity compared to home page, because it needs to receive data from the database or cache. The connection string, containing data on how to connect to a tenant's database is hardcoded in each SaaS application that implements *single tenant application, single database*.

4.3.3 Caching of the course page

A small cache was already in the example application in the course page. This cache was caching data from the database for one minute.

4.4 Multi tenant application, database-per-tenant

The second part of development was to implement a *multi tenant application, database-per-tenant* SaaS application. This implementation of tenancy model was built on the previous implementation of *single tenant application, single database*. Since tenants shared a SaaS application, this was implemented with extra caution regarding data isolation between tenants. Key areas that were highlighted was cache implementation, since it was storing tenant data. Also, implementation of the catalog database including connection strings for each tenant specific database, this is further explained in *Subsection 4.5.3*. Since we had a hard deadline, our main focus was to implement the multi tenant support on the home page and course page of the SaaS application. Our approach to distinguish which tenant a request is belonging to was done by using a unique query string in the URL for each request. This query string id was then used as a tenant-id.

4.4.1 Home page

Since the tenants shared the SaaS application, the home page had to be dynamic depending on which tenant that was accessing it. Depending on which tenant that used the application their specific university name was shown.

4.4.2 Course page

The correct connection string for a tenant is provided via a catalog database, so we could avoid hardcoded metadata in the application. The back-end functionality now fetched a tenant-id from the query string for each request to the page, and obtained a connection string from the catalog database. Finally, opened a connection with the correct database of the tenant. This func-

tionality was built with the purpose to extend the already implemented *single tenant application, single database*, without making any direct changes to the database calls made from the code.

4.4.3 Catalog database

Since each tenant need their own database, we implemented a catalog database which holds metadata about tenants. We built the back-end functionality with a separate class, where the class collects URL-data and uses the query string to collect the tenant-id for the request. The tenant-id is used to retrieve the tenant's corresponding connection-string to their own database. The connection-string will only be provided if the user is authorized.

4.4.4 Caching

Once we got the different databases working, we noticed that tenant-specific data where shared among tenants. A new cache was implemented using tenant-id together with a cache-key, thus solving the information leak. It is implemented on both home and course page storing connection-strings, university name and course details.

4.5 Multi Tenant Application, Sharded Database

To implement the tenancy model of *multi tenant application, sharded database*, we had to further develop our previous developed SaaS application, *multi tenant application, database-per-tenant*.

4.5.1 Home page

Nothing new was implemented on the home page, since it already supports multi-tenancy from the previous implemented SaaS application.

4.5.2 Course page

The course page supported multi-tenancy from the previous implemented SaaS application. But we further developed the back end to support sharded databases.

4.5.3 Database

SQL-RLS (Row-level security) was implemented in the database-layer via a security policy. The security policy is applied to a specific *table* in the database. Each time a request was made to the database from the SaaS application to a table in the database, the corresponding security policy of the table was triggered. The security policy is linked together with a *predicate function* that was applied to each row of the table that has the corresponding security policy. This was ensuring that every request from the application to the database was checked by the database-layer, so that if the tenant was authorized it would receive correct data.

We changed the schema for the course table, which the course page retrieves the courses from, by adding a tenant-id column, therefore each row was bound with a tenant-id. When using the security policy, it would filter out the rows that does not have the same tenant-id as the tenant that was retrieving data from the database.

4.5.4 Caching

No further development was needed for the cache at this part.

4.6 Comparison

During the whole development period, we continuously reviewed our work and compared the models based on different criteria, *isolation*, *tenancy-cost*, *performance*, *development* and *maintainability*. These criteria are further explained in *Section 3.2.4*.

4.7 Load Testing

This section will present how the work of perform load tests was conducted. A description of the machine used for load testing and the resources allocated on Azure is found in the beginning of this section. The setup for the SaaS applications used in the load test are presented last in this section.

4.7.1 Technical Background

Each of the three different tenancy models were deployed to Azure on an application server. A SQL elastic pool was allocated in Azure to provide persistent storage for the SaaS applications. All the SQL databases that was connected to the three different tenancy models were placed in the shared SQL elastic pool. Only one of the three tenancy models implementations were run on the application server at a time. This ensured that the other tenancy model's databases could not affect the database performance of the currently load tested tenancy model. The resources that were allocated on Azure for the tests are:

Server
Name: P2V2
CPU: 2 cores
RAM: 7GB
Disk: 250GB
Virtual Machine: Dv2-series compute
Location: Netherlands

Database
Type: SQL Elastic Pool
Pricing Tier: Production
Performance: 100 eDTU:s
Capacity: 100GB
Location: Netherlands

The machine that ran the load tests was a local computer at CRM Treasury Systems office in Värtahamnen, Stockholm. The machine run the load tests in *Visual Studio Enterprise 2017*.

4.7.2 Load Test of Tenancy Models

Subsection 3.2.4 describes that all three different SaaS applications of the tenancy models where loaded with four different HTTP load tests, *Home page with cache*, *Home page without cache*, *Course page with cache*, *Course page without cache*, and that the four different tests was performed with six different loads of simultaneously virtual users.

All the load tests that were performed was URL-based. The load tests that was used in visual studio, was a constant load of simultaneously requests from virtual users, during a span of two minutes. All the SaaS applications are tested one by one, first *single tenant application, single database*, followed by *multi tenant application, database-per-tenant* and finally *multi tenant application, sharded database*. Each SaaS application was load tested in the following way:

1. Home page with cache.
2. Home page without cache
3. Course page with cache
4. Course page without cache

All four load tests presented above has a corresponding URL for each of the implemented tenancy models. Each test was run six times with a different load of virtual users. The load tests had the following number of virtual users: 2, 4, 8, 16, 32, 64. The load tests where run with both cache and without cache. All the test was using a HTTP GET method. The load tests environment consisted of a two-tenant setup. This means that each test for a specific tenancy model, the corresponding SaaS application setup on the

application server was customized for two tenants. All the load tests created in Visual Studio were designed to make requests from both tenants, evenly distributed on both.

For the *single tenant application, single database* this meant that the setup was two application instances on the application server with one database for each instance. The *multi tenant application, database-per-tenant* load tests consisted of one shared application instance, one database for each tenant and one catalog database. The *multi tenant application, sharded database* consisted of one application, one sharded database and one catalog database. Hence, the sharded database is shared by two tenants.

4.8 Evaluation of TMSG

The evaluation of the TMSG model were performed with interviews. The interviews was made with Andreas Söderqvist and Daniel Svensson; both of them are employed by CRM Treasury Systems.All the interview questions were based on questions from a similar study to ours *Select Database (SeDB) - A Database Selection Process Model* [15]. The question of *IC 5* below, was the only question that was added. The answers from the interviewee are provided in *Appendix B*. The questions are presented in the following sub-sections presented below:

4.8.1 Interviewee Credibility

- **IC 1**: What is your profession?
- **IC 2**: What roles have you had?
- **IC 3**: For how long have you had those roles?
- **IC 4**: Have you ever participated in any tenancy model selection project?
- **IC 5**: What is your education?

4.8.2 Syntactic Correctness

- **SeC 1**: Are the phases relevant to be part of the TMSG model?
- **SeC 2**: Are the phases applicable?

4.8.3 Semantic Correctness

- **SyC 1**: Are there any redundant or unnecessary phases in the TMSG model?
- **SyC 2**: Are there any important phases related to selecting a tenancy model that are not addressed in the TMSG model?
- **SyC 3**: What do you think about the proposed sequence of phases in the TMSG model?

4.8.4 Usefulness

- **F 1**: With minor adaptations, is it possible to use the TMSG model in various tenancy model selection projects?

4.8.5 Model Flexibility

- **U 1**: Does the TMSG model guide you in selecting a tenancy model?
- **U 2**: Would you consider using TMSG model for selecting a tenancy model?

4.9 Refinement of TMSG

In this section, the work of refining the TMSG model is presented. All the refinements of TMSG model are based on the feedback that was received from the conducted interviews in the previous phase of *Evaluation of TMSG*. The refinements of TMSG model are presented in corresponding phase that the refinements were done in the model.

4.9.1 Analyse Business

In *Subsection 4.2.1* we are highlighting the importance of understanding the customers and marketing segment. But, both of the interviewees said that it is at least as important to understand the inner state of the business that provides the SaaS application.

One of the interviewee mentions that we need to be clear and be more careful with the usage of the word ”value proposition” in *Subsection 4.1.1*. We are possibly not clear with what we want to convey in the paragraph where ”value proposition” is mentioned. We want to convey that by changing tenancy model there is a risk that the value proposition will not be the same, which might imply that the value proposition does not fulfill the customer’s needs anymore.

Both of the interviewees emphasizes the importance of analyse the inner state of the business in the phase of *Analyse Business* (at the moment of interviewing, we are highlighting the external parts of the business more, like the paragraph of identifying customers and marketing segments). Daniel from the second interview says that it is very important to understand the internal processes of the business and the product that is offered, examples are: IT operations, development and how invoices are handled. He adds that it is important to understand what type of data that the business is handling. The handling of a certain type of data can be governed by laws and rules, like *Payment Card Industries* (PCI). Hence, it is important to see the risks of a software solution, from the provider’s perspective; this can vary depending in what type of business area that you are doing business within.

4.9.2 Transform into Criteria

In this phase, we are introducing a grading system to make the grading of the criteria more feasible for a business.

The following is a grading system that is used in the comparison:

- **Very High** - When the criterion is of very high concern.
- **High** - When the criterion is of high concern.
- **Medium** - When the criterion is of medium concern.
- **Low** - When the criterion is of low concern.
- **Very Low** - When the criterion if of very low concern.

The advantage of the grading system is its simplicity, and it is probably wise since it is not evaluated in this study. When using the TMSG model, the user will go through all the criteria and grade them depending on their needs. Each criterion is set to the one of the following grades: *very high*, *high*, *medium*, *low*, *very low*.

One of the interviewee would like to add and replace some questions for the criteria. For the isolation criterion, the question of *"in what extent does the tenants' data need to be protected?"* are replaced with: *"what are the effects if a tenant is seeing another tenant's data?"* and *"what are the risks that someone would try to hack the system (for example IT-infrastructure or SaaS application)?"*. These questions are more guiding and helpful if a business wants to decide what level of isolation they need. For the *tenancy-cost* criterion, the questions of *"in what extent are the financial constraints of the continuous cost for a tenant?"* and *"in what extent are the financial constraints of the maintainability cost for a tenant?"* are added.

4.9.3 Select Tenancy Model

The previous phase of *Transform into Criteria* introduced a grading system. The input into this phase is now a grading of each criterion based on the grading system presented in *Subsection 4.1.2*. A summary of the comparison, based on the grading system is presented in *Appendix C*. The summary

of the comparison is created with the knowledge gained from comparison-phase in the method, and Microsoft's comparison. This makes the comparison of criteria more feasible, which implies that the chances of selecting the right tenancy model is higher. The results from the comparison-phase of the method presented in *Section 5.2* can be used for a more detailed comparison. in Furthermore, one of the interviewees highlights the importance to be able to show the advantages and disadvantages of the tenancy models. The provided summary of comparison is clearly showing the advantages and disadvantages of the tenancy models.

4.9.4 Evaluation of Selection

Both of the interviewees pointed out that it would be suitable to create a *Proof of Concept* (POC) in this phase. A POC would give a better estimate if the selected tenancy model is sufficient to fulfill the grading of the criteria, hence minimizing the risk of implementing wrong tenancy model.

5 Comparison Results

The results of the comparison-phase are based on the research literature, experience gained from implementing the tenancy models and performing load tests. A summary of the comparison is presented in *Appendix C*. In *Section 5.1* the results regarding isolation are presented. The result of tenancy-cost is presented in *Section 5.2*. In *Section 5.3* the performance results are presented. Further on in *Section 5.4* results regarding the development are presented. The comparison results regarding maintainability are presented in *Section 5.5*. Lastly an summary of the comparison are presented in *Section 5.6*.

5.1 Isolation

The evaluation of isolation is based on the questions defined in *Subsection 3.2.4*.

5.1.1 Single tenant application, single database.

There is no possibility for a tenant to read or write another tenant's data, since each tenant's SaaS application is its own installation that is dedicated for the tenant. This implies that there is no possibilities to write or read from a tenant's dedicated database, since they are separated by both the SaaS application and physically by the separate databases[23].

It is possible to guarantee that a tenant is reading and writing to the correct database, since there is one dedicated installation of the SaaS application per tenant and the databases are physically separated between tenants.

5.1.2 Multi tenant application, database-per-tenant.

The application is shared between tenants, which could be a potential risk of leaking data between tenants if it is not correctly implemented. If the setup of the application is correctly implemented, this should not let tenants read or overwrite another tenant's data. Each tenant has their own dedicated database, which implies that the databases also is physically isolated. Precautions are needed on the application-layer where metadata associates a database with a specific tenant, this is to ensure that the correct tenant is accessing its own data. If the setup is correct, this prevents any tenant from accidentally or maliciously accessing other tenant's data[23] and its possible to guarantee that a tenant is writing or reading from the correct database.

5.1.3 Multi tenant application, sharded database.

The SaaS application and database is shared between tenants, this means that tenant isolation is sacrificed[4]. There is a potential risk of leaking data between tenants if it is not correctly implemented. If the setup of the SaaS application is correct, it is possible to guarantee that a tenant is writing or reading from the correct database.

Extra security functionality has to be added to the database layer, securing that each tenant only reads and write their own data. In our application, we solved it by adding SQL Row-level security.

5.2 Tenancy-cost

We uploaded 10 installations of a *single tenant application, single database* SaaS applications to the application server in Azure. There were no possibility to retrieve the RAM or disk usage.
Hence, the lack of data from RAM and disk usage implied that we had no further possibility to conclude any results regarding the cost measured in monetary terms.

5.3 Performance

Below follows the results of load testing the SaaS applications. The graphs shows the different response time for each application with varying setups.

Figure 11 presents how the average response time differs between *multi tenant application, database-per-tenant* and *multi tenant application, sharded database* applications on the *Index page without cache*. The *Single tenant application, single database* SaaS application has no dynamic information retrieved from the database displayed on its index page, therefore it was excluded from this test. The graphs show that there is almost no difference between the two models, except on the last test with 64 virtual users where the *multi tenant application, sharded database* is slower.

Figure 12 shows how the average response time differs for the Index page with cache. The three different tenancy models *single tenant application, single database*, *multi tenant application, database-per-tenant* and *multi tenant application, sharded database* are all tested at this point.

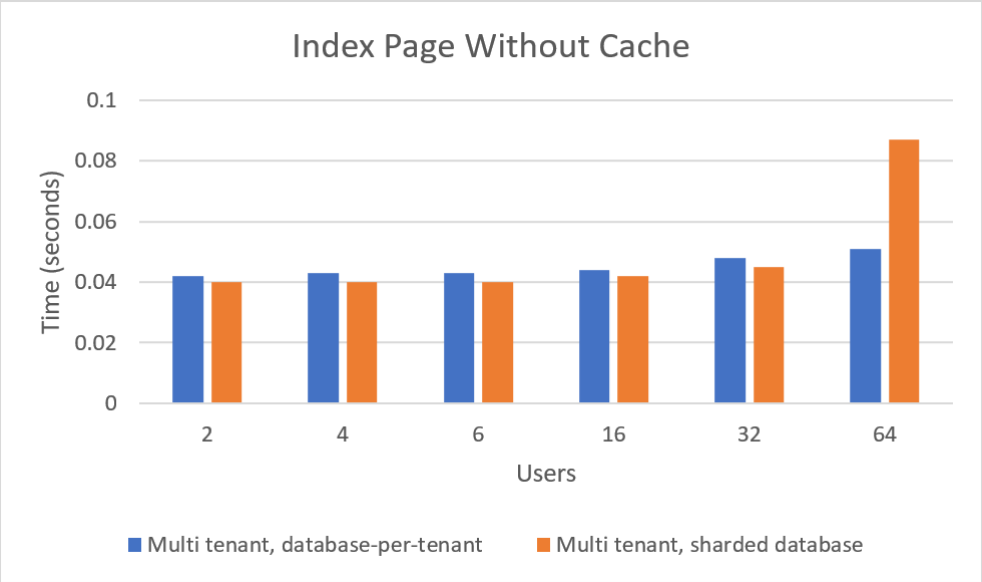

Figure 11: Average response time (seconds) for the index page without caching

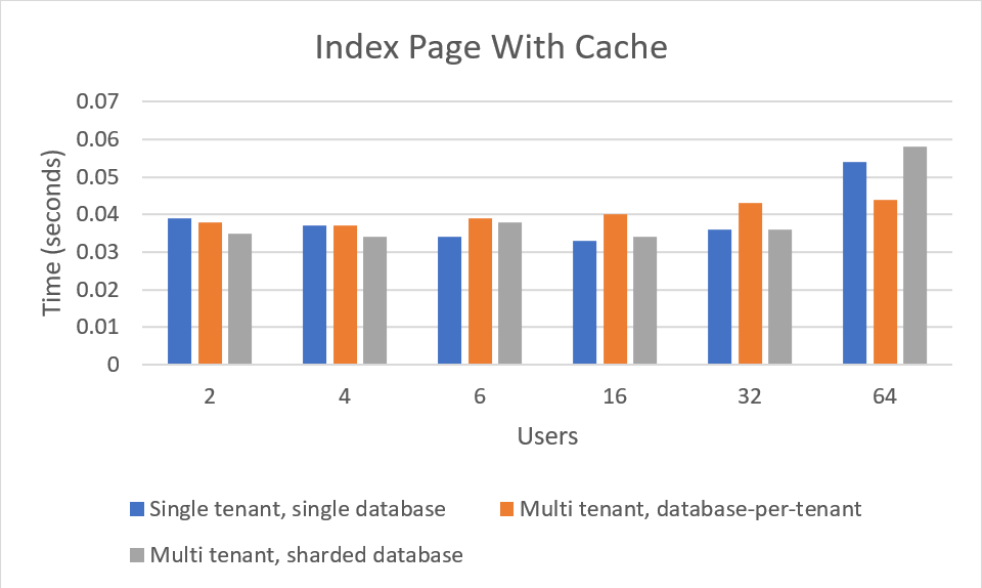

Figure 12: Average response time (seconds) for the index page with caching

The figure shows that there is no remarkable differences in response time among the tenancy models.

Figure 13 shows the average response time for Course page without cache, the different models does not differ much except for the two tests with 32 and 64 virtual users. This is where the two tenancy models of *multi tenant application, database-per-tenant* and *multi tenant application, sharded database* shows worse performance than *single tenant application, single database* model.

Figure 14 illustrates average response time for Course page with cache. The results are similar compared to Figure 13 where the cache is deactivated. The response time is faster but still *multi tenant application, database-per-tenant and multi tenant application, sharded database* are slower compared to *single tenant application, single database*.

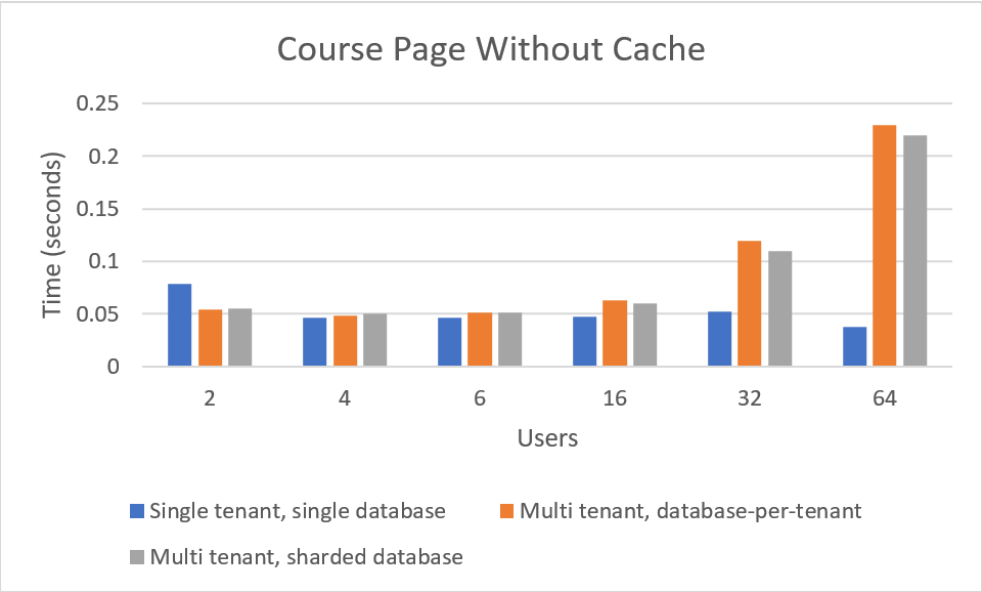

Figure 13: Average response time (seconds) for the course page without caching

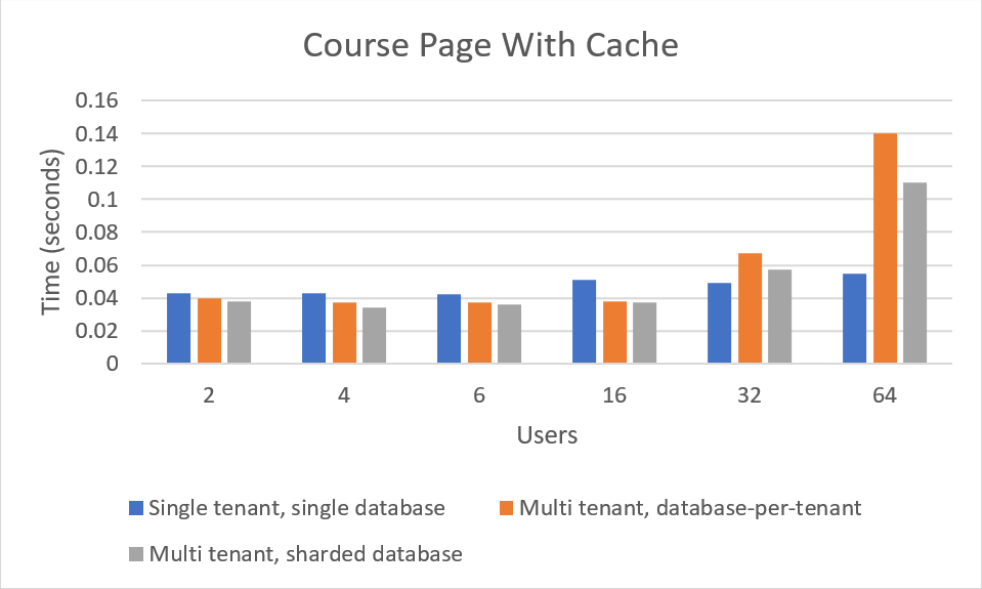

Figure 14: Average response time (seconds) for the course page with caching

5.4 Development

The evaluation of development is based on the questions defined in *Subsection 3.2.4*.

5.4.1 Single tenant application, single database.

This tenancy model is the simplest to develop. Since the SaaS application is dedicated for a single tenant and all tenant specific functionality can be hardcoded into the SaaS application. The database that the tenant is using, is dedicate for the tenant, hence there is no need to handle the functionality of differentiate tenant's databases or data in the databases.

If changes of the queries that is made from the SaaS application are needed, the changes are made directly in the application-layer of the SaaS application. Since each tenant has their own dedicated application, it can be complex to change all running SaaS application for each tenant. To change the database schema, the changes are made directly in the database that is connected to the SaaS application. Since each tenant has its own dedicated database, it can be complex to change all database schema for each tenant.

5.4.2 Multi tenant application, database-per-tenant.

More complex development than the *single tenant application, single database*. Compared to *single tenant application, single database*, identification of the specific tenant, catalog database and functionality to retrieve the connection string belonging to a tenant's database had to be implemented.

If changes of the queries that are made from the SaaS application is needed, the changes are made directly in the application-layer of the SaaS application. Since all tenants are sharing the same SaaS application, the changes are only made in one SaaS application, hence the complexity is lower than in *single tenant application, single database*. If changes to the database schema are needed, the changes are made directly in the database. Chang-

ing the schema can be complex, since all tenants has their own database, and changes for each tenant is needed.

5.4.3 Multi tenant application, sharded database.

This tenancy model was the most complex to implement, compared to the other three. Compared to *multi tenant application, database-per-tenant*, this tenancy model also needed changes in the database-layer to be implemented. The changes in the database imposes adding a shard key in the database schema and adding RLS. RLS entails adding *predicate functions* and *security policies*.

If changes to the queries that are made from the SaaS application is needed, the changes are made directly in the application-layer of the SaaS application. Since all tenants are sharing the same SaaS application, the changes are only made in one SaaS application. The RLS might impose restrictions of what types of queries that is allowed to the database, hence RLS have to be changed. The complexity of changing a query could be more complex than *multi tenant application, database-per-tenant*. If changes to the database schema is needed, the changes are made directly in the database. Changing the schema can be complex, since the tenancy model uses RLS. Since the database is sharded, which means that tenants can share a database; this can lower the complexity of updating the schema used by databases.

5.5 Maintainability

The complexity of updating is measured in how many updates that are needed when the SaaS application needs to be updated to a new version. The evaluation of maintainability is based on the question defined in *Subsection 3.2.4*.

5.5.1 Single tenant application, single database.

If the SaaS application or the database needs an update, there will be *n* amounts of updates, where *n* is the number of tenants, for either the applications or databases.

5.5.2 Multi tenant application, database-per-tenant.

The *multi tenant application, database-per-tenant* tenancy model only needs one update for the SaaS application, since the application is shared among tenants. Updating the databases would result in *n* amounts of updates, where *n* is the number of tenants.

5.5.3 Multi tenant application, sharded database.

The *multi tenant application, sharded database* tenancy model only needs one update for the SaaS application, since the application is shared among tenants. The number of updates that is needed for the databases can be in between 1 and *n*, where *n* is the number of tenants. Because of the flexibility that the sharded database is offering, it is possible for all tenants to share one database, while it is also possible for each tenant to have its own database. This model has the possibility to offer the smallest number of updates in a SaaS application (with database included).

5.6 Summary of Comparison

This *summary of comparison* is derived from our own results in this study, and a comparison made by Microsoft presented in *Appendix A*. The Microsoft comparison is only used when it is applicable, since our criteria and their criteria are not defined the same.

5.6.1 Isolation

This grading is based on the results of *Subsection 5.1*. Based on isolation, we have given the tenancy models the following grades:

- *Single Tenant Application, Single Database* - **Very High**
- *Multi Tenant Application, Database-per-tenant* - **High**
- *Multi Tenant Application, Sharded Database* - **Low**

5.6.2 Tenancy-cost

We did not reach a result in our research regarding tenancy-cost. In Microsoft's comparison, they have provided a comparison for database-cost-per-tenant. Based on tenancy-cost, Microsoft has graded the tenancy models the following:

- *Single Tenant Application, Single Database* - **High**
- *Multi Tenant Application, Database-per-tenant* - **Low**
- *Multi Tenant Application, Sharded Database* - **Very Low**

5.6.3 Performance

In the load tests, the performance difference among *multi tenant application, database-per-tenant* and *multi tenant application, sharded database* are varying, but if the graphs in *Subsection 5.3* are aggregated, it looks like they have the same performance. When requesting the course page, the *single tenant application, single database* performed much better than the other two tenancy models. Microsoft has not provided a comparison for performance. Based on performance, we have given the tenancy models the following grades:

- *Single Tenant Application, Single Database* - **High**
- *Multi Tenant Application, Database-per-tenant* - **Low**
- *Multi Tenant Application, Sharded Database* - **Low**

5.6.4 Development

In Microsoft's comparison for development they have defined development complexity as: *changes to schema* and *changes to schema*, which concerns continuous development complexity. We have included implementation development complexity in our comparison as well, which is presented in *Subsection 5.4*. In the comparison we have weighted this together to create a grading among the tenancy models. The grade *Very Low* should be interpreted as high development complexity, and *Very High* as very low development complexity. We have given them the following grades:

- *Single Tenant Application, Single Database* - **Low**
- *Multi Tenant Application, Database-per-tenant* - **Medium**
- *Multi Tenant Application, Sharded Database* - **High**

5.6.5 Maintainability

This grading is based on the results of *Subsection 5.5. Multi tenant application, sharded database*, can vary between *very low* and *medium* because of the flexibility on how many tenants' data that can be stored in a single database. Microsoft's definition of maintainability was wider than our definition, hence it was hard to apply. The grade *Very Low* should be interpreted as high maintainability complexity, and *Very High* as very low maintainability complexity. We have given them the following grades:

- *Single Tenant Application, Single Database* - **Very Low**
- *Multi Tenant Application, Database-per-tenant* - **Medium**
- *Multi Tenant Application, Sharded Database* - **Medium - Very High**

6 Model Results

In this chapter, the review of TMSG is presented. The review was conducted by interviewing two employees at CRM Treasury Systems. *Section 6.1* presents a detailed description of the interviews. In *Section 6.2* a conclusion of the interviews is presented.

6.1 Interview Results

In this section, a detailed description of the interviews are presented. The presentation of the following subsections are based on the evaluation criteria. The first interview was made with Andreas Söderqvist. The second interview was made with Daniel Svensson.

6.1.1 Interviewee Credibility

The first interviewee holds a bachelor of science in engineering within computer science. He has worked as *Chief Technology Officer* (CTO) for 3 years, 11 years as developer, 1.5 years as Scrum Master and 1.5 years as Agile Coach. At the moment he holds the title of CTO. He has participated in a tenancy selection before, even though it was not an active choice.

The second interviewee holds a master of science in engineering within computer science. He has worked as software engineer since 2011; during that time he has also had roles like project leader for 2 years, software architect. Furthermore, as a software engineer he has had roles that are focusing on security and IT-infrastructure. He has never participated in a selection of a tenancy model.

6.1.2 Semantic Correctness

The first interviewee said that all phases are relevant. But he thinks that we should be more clear about how the transition from *Analyse Business*-phase to *Select Tenancy Model*-phase is conducted. He said that all the

phases are applicable. But he would like to see some kind of grading system in *Transform into Criteria*-phase. This can possibly make *Select Tenancy Model*-phase more feasible. He also mentions that it is very important to understand the importance of analyse the inner state of the company (that is selecting tenancy model). At the moment of interviewing there is less emphasis on the analyse of inner state of the selecting company; at the moment of writing it is slightly improved, but not sufficient. At the moment of the interview we are using the word of ”value proposition”, in the presentation of TMSG, in a vague way. Lastly, the first interviewee thinks that it would be wise to implement a *Proof Of Concept* (POC) of the selected tenancy model before implementing it as a complete SaaS-application.

The second interviewee said that the all phases in TMSG are relevant. He said that all phases, with some modifications, are applicable. The second interviewee mentions the same thing as the first interviewee, it is very important to understand the importance of analysing the the inner state of the business in the selecting company. He said that it is very important to understand the internal processes of the business and the product that is offered. The interviewee added that it is important for the selecting company to understand what type of data they are handling; handling of certain types of data can be governed by laws and rules. It is important to understand the risks of a solution from the providers perspective; this can vary depending on what type of business area the company is doing business. The interviewee also like to add some questions in *Transform into Criteria*-phase. He would like to add the following questions to the isolation criterion: *”What are the effects if a tenant a is seeing another tenant’s data?”* and *”What are the risks that someone would try to hack the system?”*. These two question would replace the first question of *”In what extent does the tenants data need to be protected?”*; they are suitable, both more descriptive and guiding. He mentions that it would be interesting to add the question of *”What is the continuous cost for a tenant?”* for the criterion of tenant-cost. Another interesting feature the interviewee would like to see is how the cost of main-

tainability is affected when adding a tenant; this would go under tenant-cost. He adds that it is important to see the advantages and disadvantages of the chosen tenancy model. Lastly, just like the first interviewee says, he thinks that the idea of implementing a POC in *Evaluation of Selection* could give a better estimation if the selected tenancy model is sufficient to fulfill the criteria of the business.

6.1.3 Syntactic Correctness

When conducting the first interview, the interviewee said that there are no redundant or unnecessary phases in the TMSG model. He also thinks that there is no important phases related to selecting a tenancy model that are not addressed in TMSG model. Furthermore, he thinks that the proposed sequence of phases in the TMSG model is reasonable.

The second interviewee thinks that there are no redundant or unnecessary phases in TMSH model. He also says that there are no important phases related to selecting a tenancy model that are not addressed in TMSG model. Furthermore, for the current sequence of phases, he purposed that it would be adequate to go back from *Evaluation of Selection*-phase to *Transform into Criteria*-phase to compare the selected tenancy model with the criteria that was crystallized in that phase.

6.1.4 Model Flexibility

The first interviewee said that if TMSG is going to be used with minor adaptations in various tenancy model selection projects, the criteria in *Transform into Criteria*-phase must be further developed.

The second interviewee says that TMSG could be used in various tenancy model selection project with some minor adaptations; if a few questions are added and customized to the *Transform into Criteria*-phase.

The adding and customization of the questions will help to make TMSG model more general and flexible.

6.1.5 Usefulness

The first interviewee believes that the TMSG model is guiding for selection of tenancy model. Furthermore, he would consider using TMSG for selecting a tenancy model.

The second interviewee thinks that the TMSG model is guiding in a selection of a tenancy model, provided that the model is of good quality (comparison was not shown). Furthermore, he would consider using TMSG in a selection of a tenancy model.

6.2 Evaluation of Result

The interviewees' credibility is of high quality. Both of the interviewees hold a degree from university within computer science, and they have many years of experience from the software industry. The model's phases are relevant, but there should be more weight on how to conduct *Transform into Criteria*-phase, one proposal is to introduce a grading system. The phases are applicable with some modifications. Both of the interviewees agrees on that *Analyse Business* has to further be improved. One useful idea that the interviewees came up with was to implement a POC in the last phase. Furthermore, one of the interviewees added that some of the questions in *Transform into Criteria* must be refined. Both thinks that there is no redundant, unnecessary or missing phases in TMSG. One interviewee thinks that it would be adequate compare the chosen tenancy model with the crystallized criteria from *Transform into Criteria*-phase. After the interviews, we can conclude that the model is not flexible and general enough; both of the interviewees agree on that *Transform into Criteria*-phase must be refined. Both of the interviewees believes that TMSG is guiding and would consider using it in a selection of a tenancy model.

7 Discussion

In this chapter, a discussion is presented based on the comparison results in *Chapter 5*, research phases and quality assurance. In *Section 7.1* a discussion concerning the creation of the TMSG model is given. *Section 7.2* is about the implementation phase. In *Section 7.3* a discussion about the comparisons are given. In *Section 7.4* a discussion about the evaluation of TMSG model is given. *Section 7.5* discuss the refinements of the TMSG model. Last in this chapter, *Section 7.6* is presented, where a discussion on quality assurance is given.

7.1 Create TMSG

The *Literature Study* phase provided an input to the *Create TMSG* phase, and made the phase feasible. During the literature study, we realized that the area of providing guidance for selecting computer related systems is a vast area of study. This revelation made us realize that the result of this study will probably not provide a full guidance model for business. Instead, the study will provide a knowledge base for future researchers to continue the development and refinement of the TMSG model. Either way, during the *Create TMSG* phase, based on the research literature, we tried to cover as much of the necessary steps that was needed for a selection of a tenancy model. We hope that the steps of TMSG that was missed in the *Create TMSG* phase, was covered in the *Refinement of TMSG* phase, with the interviews as input. If we had more time, a more extensive literature study would have been conducted, and probably improved the initial TMSG model created in *Create TMSG* phase.

7.2 Implementation

The output of the *Implementation* phase of the study was an implementation of three different tenancy models, *single tenant application, single database*, *multi tenant application, database-per-tenant* and *multi tenant*

application, sharded database. One important thing that should be discussed here is the correctness of the implementations. Is it possible that we implemented the tenancy models incorrect, and this error has affected the outcome of the upcoming phases of the study. During the development of the tenancy models we were provided with continuous feedback from our supervisor at CRM Treasury Systems, to guide us through the development. Hence, the risk that the implemented tenancy model are incorrect is small.

7.3 Comparison

In the comparison-phase, several comparisons were made, which was based on five criteria. This section discusses the result from the *comparison*-phase, where each criterion is discussed in its own subsection.

7.3.1 Isolation

During the development, we continuously read about the different types of tenancy models and their impact on isolation. From our understanding, the sharded database with RLS is favorable to companies where data only should be visible to certain users. We have not investigated how secure the RLS is when it comes to separating different tenant's data and therefore this could be further explored. During this work, there has not been any study about SQL-injections or exploits regarding the databases. Both the *multi tenant application, database-per-tenant* and *single tenant application, single database* are very isolated, which are achieved by having separate databases. One advantage of *single tenant application, single database* is that it is possible run each installation of the SaaS application on different physical servers, which would increase the isolation.

7.3.2 Tenancy-cost

As presented in *Section 6.2*, it was not possible to measure any RAM or disk usage. The reason why it was not possible to reach this report's goals for tenancy-cost, was because of unrealistic RAM usage measures in Azure.

When hosting the *single tenant application, single database* on a local machine at CRM Treasury Systems office, the usage of RAM was around 200 MB for the SaaS application, this was measured in the debugging tools provided by Visual studio 2017. In Azure, the RAM usage was around 50% of the application server's total amount of RAM, which was 1.75GB, when not running any applications on it. The usage of RAM on the application server is probably mainly from the operating system. When hosting and running 10 SaaS applications of the tenancy model of *single tenant application, single database*, the percentage of RAM usage was still around 50%. We also tested to scale up the application server to 3.5GB of RAM, while running 10 SaaS applications of *single tenant application, single database*. When not running any SaaS applications on the scaled-up application server, the RAM usage was around 20%. The RAM usage for the scaled-up server did barely change when 10 SaaS applications was run at the same time.

The reason why this is occurring could be because of that the operating system on the application server is making optimizations. One example of optimization could be to put memory taken up by the running applications in the swap on the disk. We also tried to find out if there were any possibilities to turn off the swapping, but with no success.

Another problem that complicated the measurement of RAM is the usage of dynamic-link library (DLL) files. DLL:s are files that are loaded into a running program, which can also be shared with other running programs. This prevented us from making the assumption that the RAM usage will be proportional with the number of tenants.

Because of the complications of measuring the RAM usage, none of the other tenancy-cost goals was considered. Fortunately, Microsoft's comparison provided some input to this criterion, which were used in the summary of the comparison.

7.3.3 Performance

The machine that was used to load test the SaaS applications in Azure, was located at CRM Treasury Systems office in Värtahamnen, Stockholm. The load testing of the three different tenancy models, could have been affected by the network supplied via CRM Treasury Systems. Since there were other employees using the network at the same time as the load testing was performed. Also, a few days prior to the load testing, the Internet service provider had some issues with delivering their service to CRM Treasury Systems. Finally, it is possible that Azure is experiencing high workload on their servers at the moment of usage, which could have affected the result.

We have chosen not to use POST-operation for the load test. This decision was made together with CRM Treasury systems, since their SaaS application mainly uses GET-operations.

Index page without cache. Up to 32 virtual users the two tenancy models of *multi tenant application, database-per-tenant.* and *multi tenant application, sharded database* perform equally well. The result of 64 virtual users is significant, the *multi tenant application, sharded database* is processing a request at almost the double amount of time as *multi tenant application, database-per-tenant.* This is probably a result from the issue connected to the Internet service provider.

Index page with cache. For this result, there is not much to discuss. The average request time is almost the same up to 32 virtual users. The average request time increases something at 64 virtual users compared to the lower levels of virtual users for *single tenant application, single database* and *multi tenant application, sharded database.*

Course page without cache. Up to 16 virtual users the performance of all three tenancy models are almost the same. At 32 virtual users, the average

response time for *multi tenant application, database-per-tenant* and *multi tenant application, sharded database* has more than double compared to *single tenant application, single database*. At 64 virtual users, the average response time for *multi tenant application, database-per-tenant* and *multi tenant application, sharded database* are more than five times more than *single tenant application, single database*. The *multi tenant application, database-per-tenant* and *multi tenant application, sharded database* is following the same average response time pattern in the result, and *single tenant application, single database* is not following the pattern. A possible explanation for this could be that both *multi tenant application, database-per-tenant* and *multi tenant application, sharded database*, needs to retrieve a connection string for each tenant's database, and that they are sharing an application. This could possibly signify that the multi tenant application is the bottleneck or that the shared catalog database is. Also, a very significant result is the average response time for a *single tenant application, single database* is better as more virtual users is added.

Course page with cache. Up to 32 virtual users the average request time is almost the same. At 64 virtual users, the average response time are almost doubled for *multi tenant application, database-per-tenant* and *multi tenant application, sharded database* compared to *single tenant application, single database*. This is following the same pattern as the same load test but without cache. One possible reason for the significant average response time for 64 virtual users could be because that the multi tenant application is the bottleneck, since the tenants are sharing it. Since the result from the catalog database is cached, it is unlikely that it is a bottleneck.

7.3.4 Development

Our implementation of the SaaS applications was made in C#, .NET and Azure SQL Database. Still, it is possible to implement a SaaS applications in several other languages, which opens up a discussion of how difficult it is

to implement the tenancy models in other languages. Since C# is a C-looking language that is implementing the paradigms of object-orientation, imperative and also has some flavors of functional programming. These paradigms are often implemented by other common programming languages today, like *Java* or *C++*. It is also easy to see that the structure of a program that is written in programming languages like C#, Java, C and C++ are similar to each other; especially if they are compared to declarative programming languages, like *Haskell*, *Erlang* or *SQL*. With this as an argument, it is almost certain that the complexity of the implementation of the tenancy models are the same for common languages like Java, C++ or C compared to our programming language of implementation, C#.

Like mentioned in the paragraph above, the complexity of the implementation of the tenancy model may vary from programming language to another. The discussion and arguments in the paragraph above holds for the result of *complexity of changing the queries*, since the queries are implemented in the application-layer of the SaaS application, where the programming languages are used. The result of *complexity of changing the queries*, should be valid for languages that is similar to C#.

The complexity of changing the database schema is simple for the *single tenant application, single database* and *multi tenant application, database-per-tenant*, because the way of alter the schema in those tenancy models are by executing the ”ALTER TABLE”, ”CREATE TABLE”, ”DROP TABLE” or any other SQL-commando that are changing the schema. This functionality of changing a schema exists in all different SQL databases. The concept of RLS is limited to the SQL database of Azure SQL Database. This is limiting the result of *complexity of changing the schema* for the *multi tenant application, sharded application* to Azure SQL Database. No research has been made to see if there is equal technology as RLS in other SQL databases.

7.3.5 Maintainability

The same SaaS application can be installed at different data-centers, which are located at different geographical locations. This can improve performance when a customer connects to their SaaS application and the data center that host the application is physically closer to the customer. This would lead to more SaaS applications hosted, which could lead to more maintenance-work. Our result for *maintainability* has not taken this into consideration. The result is based on the assumption that the SaaS application is run at one single geographical location.

7.3.6 Summary of Comparison

The summary of comparison are based on the results of the comparison in *Section 5.2* and the comparison provided from Microsoft. The results of our own comparison has not yet been evaluated, which is a potential validity threat. Furthermore, the grading of each criterion for each tenancy, except for tenancy-cost, is made by ourselves and needs to be evaluated as well. The criterion of tenancy-cost was entirely graded after Microsoft's comparison. The gradings among the tenancy models should be interpreted as grading relatively to each other, since it is hard to define what a "low isolation" actually is, because it depends on who you ask.

7.4 Evaluation of TMSG

The *Evaluation of TMSG* phase gave some good feedback on the TMSG model that we defined in *Create TMSG* phase. As expected in *Create TMSG* phase, this phase would provide feedback and provide input to the succeeding phase to cover up the flaws in the TMSG model. Since only industrial professionals was interviewed, it would have been interesting to interview academic professionals as well. We think that the industrial and academic professionals have different aspects of the result, which could have been valuable input. This could have probably increased the validity of the study. Fur-

thermore, we only interviewed industrial professionals from CRM Treasury Systems. Since, many of the criteria and belonging questions in the *Transform into Criteria* phase was provided from CRM Treasury Systems. This could have affected the interviewees answers regarding if the TMSG model is ”good enough”, since their needs might be satisfied. One of the interviewees also said that we need to add and improve questions belonging to criteria in *Transform into Criteria* to make it more general - this might confirm our presented theory above.

7.5 TMSG Results

In this section, a discussion based on the model results in *Chapter 6*. The subsections in this section are divided into the evaluation criteria of TMSG model.

7.5.1 Interviewee Credibility

Both of the interviewees has a university degree within relevant field, and many years of experience from the software industry. Hence, we can conclude that the credibility of the interviewees was of high quality. Even though, none of the interviewees had any experience from selection of a tenancy model, this could possibly have jeopardize the credibility of the interviewees.

7.5.2 Semantic Correctness

The *Refinement of TMSG* phase was based on the feedback from the interviewees in the *Evaluation of TMSG* phase. Almost all feedback was used to improve the TMSG model, but maybe the most important feedback from the interviewees, which was that it is more important to understand the inner state of the business of the SaaS provider than the tenant’s business, was not used to improve the TMSG model to the extent that we wanted. This feedback was the most important, still, it was the one that was hardest to remediate. In an ideal case, we would have wanted to give a step-by-step guidance

on how to analyse the inner state of the company business to get a better understanding of it. Unfortunately, this would require a new literature study, and due to time restrictions it was not possible.

7.5.3 Syntactic Correctness

Both of the interviewees came to the same conclusion that there was no redundant or unnecessary phases in the TMSG model. And that the phases addressed covers the most important part of selecting a new tenancy model. The second interviewee purposed that it would be adequate to go back from *Evaluation of Selection*-phase to *Transform into Criteria*-phase, which we also believe is a necessary step if the extra redundancy is needed.

7.5.4 Model Flexibility

The interviewees said that TMSG need to be modified for it to be flexible enough. To be more specific, both of them points out that *Transform into Criteria*-phase must be improved to improve the flexibility. The questions in the phase was improved to be more flexible by receiving advises from one of the interviewee. The best way to make sure that the model is flexible enough is to apply TMSG in a project, and evaluate the outcome. Otherwise, more interviews could be conducted in the future to receive more feedback to improve the flexibility.

7.5.5 Usefulness

The interviewees said that they would consider using the TMSG model in an selection of tenancy model, and that the model provides guidance for the selection.

7.6 Quality Assurance

In this section, the quality of the study is discussed. It will be discussed from the aspects of *validity*, *dependability*, *transferability* and *conformability*.

7.6.1 Validity

To provide validity to this report, we evaluated the result by interviewing professionals from the software industry. One threat to validity is fact that both of the interviewees was from CRM Treasury Systems. Due to the reason that many of the criteria for the TMSG model in the *Transform into Criteria* phase were taken from CRM Treasury Systems requirements, presumably the TMSG model is probably satisfying the needs. The comparison that is included in the TMSG model were not evaluated.

7.6.2 Dependability

The dependability of judging the correctness of the conclusion can be hard if it wants to be replicated, due to the fact that this is to a large extent a qualitative research. It is hard to replicate qualitative research because the research process can vary. The load testing of this study is the only element of quantitative research of this study, which can easily be replicated.

7.6.3 Transferability

All of the work in the method phases in this study has been documented as much that is possible. The steps belonging to the implementation of the tenancy models are not provided in detail in this report due to space and time restrictions. But generally, we believe that this thesis has been documented well enough to be transferred.

7.6.4 Conformability

The result was evaluated with interviews with professionals from the software industry. There is a record holding the interviews in *Appendix B*. These

answers are transferred from the spoken words of the interviewees, to a written form by the authors. This might increase the risk of that the answers were misunderstood by the authors. To prevent this, the answers to the questions could have been sent back to the interviewees for confirmation; unfortunately, this was not done.

8 Conclusion

In the introduction to this study we defined our research question as follows: ”How can guidance for selecting tenancy model be provided?”. This question has been answered in this study by providing a guidance model for selecting tenancy models, which is called *TSMG*. The TMSG model consists of four sequential phases in the following order: *Analyse business*, *Transform into Criteria*, *Select Tenancy Model* and *Evaluation of Selection*.

The purpose of this study was to provide guidance for selecting tenancy models. The short-term goal was to provide a guidance model for selecting tenancy models. Both of the purpose and short-term goal has been reached by providing the TMSG model. The long-term goal of providing researchers with research material for further study has also been reached. The TMSG model provides many of the essential needs for a tenancy selection project, but there is still room for improvements by future researchers, look in *Section 7.4* for information about future work. In this study, both quantitative and qualitative methodologies with an inductive approach has been used to reach the goals. Hence, triangulation was used to reach the goals. Furthermore, a comparative study was used to compare three different tenancy models.

Achievements from this study are limited to the three different tenancy models of: *single tenant application, single database*, *multi tenant application, database-per-tenant* and *multi tenant application, sharded database*. The criteria that the TMSG model are using is limited to the following: *isolation*, *tenancy-cost*, *performance*, *development* and *maintainability*. During the study, we were also limited to certain technologies like: C#, .NET framework, Azure SQL Database, Azure and Visual Studio Enterprise 2017.

8.1 Recommendations for CRM Treasury systems AB

Our stakeholder, CRM Treasury Systems wanted a comparison among the tenancy models presented in the study. This study is more extensive than merely a comparison, still, a recommendation of tenancy model is provided for our stakeholder. During this development, we have discussed the different types of implementations, and how they would be applicable with *single tenant application, single database*, which is the tenancy model that CRM Treasury Systems is currently using. During our initial work, a few specifications were set up together with CRM Treasury Systems, which they found important. This specification was divided into criteria, which are: *isolation*, *tenancy-cost*, *performance*, *development* and *maintainability*. Among these criteria, isolation is the one that is of the highest concern by CRM Treasury Systems. Furthermore, the *complexity of continuous development* is more important than *complexity of implementation development*.

Our recommendation is to either stay with the *single tenant application, single database* or change to *multi tenant application, database-per-tenant*. If the decision to stay with the current tenancy model is taken, it would mean more maintenance work, better isolation of data, better performance, higher tenancy-cost, continuous development is slightly higher, compared with the alternative. The greatest benefit of continue to use the *single tenant application, single database* for CRM Treasury Systems, is that the isolation of each tenant's data is very high. Furthermore, single tenant databases is typically used in finance applications since there is an high need of data isolation [23]. Another benefit from choosing *single tenant application, single database* is that CRM Treasury Systems won't have to spend resources on implementing *multi tenant application, database-per-tenant*.

The reason why *multi tenant application, sharded database* is not recommended, is because of the low isolation.

8.2 Applications

This thesis can be applied by CRM Treasury Systems for guidance of selecting tenancy models. Also, it is suited for other business that has similar requirements on the SaaS application as CRM Treasury Systems.

8.3 Future Work

During the work, no practical tests has been performed on the tenancy models to test the isolation. All the work regarding isolation has been built on our theoretical knowledge and the experience gained from implementing the tenancy models. Hence, practical tests of the isolation are adequate future work. The only criterion that we did not manage to answer the belonging questions was *tenancy-cost*. The reason why the questions could not be answered was mainly because it was more complicated to measure the RAM usage than we initially thought. A deeper discussion is provided in *Subsection 7.3.2*. The goal of *performance* was reached, and it also provided an exhaustive discussion in *Subsection 7.3.3*. The discussion proposes that the multi tenant application could possibly be a bottleneck for *multi tenant application, database-per-tenant* and *multi tenant application, sharded database*. A future work is to identify if this is a bottleneck. The performed load tests was run from CRM Treasury Systems own network during day and evening time. During the load test that was made during the day, the network was shared between us and employees at CRM Treasury Systems. This could have possibly affected the result of the load tests. Also, the Internet service provider had problem to deliverer Internet a few days earlier. A future work is to ensure the network's performance is predictable when running the load tests.

From the interviews, we received feedback regarding the criteria in *Transform into Criteria*-phase. The interviewee said that the criteria must be further developed to be general enough to be applied in many tenancy selection projects. Therefore, as a future work, it is suitable to further develop the

questions belonging to each criterion, and to see if there are any other criteria that should be added. One of the most important feedback that we got from both interviewees were the importance of analyse the inner state of a business in the *Analyse Business*-phase in TMSG. Right now, we are only highlighting the importance in *Subsection 4.9.1*. In a future work, it is adequate to include a method on how to analyse the inner state of the business to get valuable insights like maintenance-, financial-, development constraints. The TMSG model has only been evaluated theoretically, hence, the model has not been used to select a tenancy model. Furthermore, it has only been evaluated in one iteration, and the refinements has not been evaluated. In the future it would be suitable with a theoretical evaluation based on the refined TMSG model. In this future evaluation, the interviewees are both from the software industry and the academics. After the second iteration of evaluation, it would be appropriate to evaluate the TMSG model by using it in a practical tenancy model selection project.

References

[1] L. Columbus, *Cloud computing market projected to reach $411b by 2020*, https://www.forbes.com/sites/louiscolumbus/2017/10/18/cloud-computing-market-projected-to-reach-411b-by-2020/#55e9d1d378f2, [Online; accessed 28-May-2018], Forbes.

[2] *Current and future trends for cloud architectures*, https://www.capgemini.com/2018/01/current-and-future-trends-for-cloud-architectures/, note =, Capgemini.

[3] *What is saas?* https://azure.microsoft.com/en-au/overview/what-is-saas/, [Online; accessed 26-March-2018], Microsoft Corporation.

[4] *Multi-tenant saas database tenancy patterns*, https://docs.microsoft.com/en-us/azure/sql-database/saas-tenancy-app-design-patterns, [Online; accessed 26-March-2018], Microsoft Corporation.

[5] *Saas is the software model of the future*, Boston Consulting Group.

[6] *How to write a research question*, George Mason University.

[7] *Dataskyddsreformen*, https://www.datainspektionen.se/dataskyddsreformen/, [Online; accessed 20-May-2018], Datainspektionen.

[8] A. Håkansson, *Portal of research methods and methodologies for research projects and degree projects*, https://kth.instructure.com/courses/1585/files/856636/download?wrap=1, [Online; accessed 14-April-2018], Department of Software and Computer Systems, The Royal Institute of Technology, KTH, Kista, Sweden.

[9] *About CRM*, http://www.crm.se/about/about-crm/, [Online; accessed 26-March-2018], CRM Treasury Systems AB.

[10] S. K. Rouven Krebs Christof Momm, *Architectural concerns in multi-tenant saas applications*, https://sdqweb.ipd.kit.edu/publications/pdfs/KrMoKo2012-closer-multitenant-sass.pdf, [Online; accessed 06-April-2018], SAP AG, Karlsruhe Institute of Technology.

[11] *What is software as a service (saas)*, https://www.salesforce.com/saas/, [Online; accessed 23-May-2018], Salesforce.com Inc.

[12] *Sharding pattern*, https://docs.microsoft.com/en-us/azure/architecture/patterns/sharding, [Online; accessed 22-May-2018], Microsoft Corporation.

[13] *Manage multiple sql databases with elastic-pools - azure*, https://docs.microsoft.com/en-us/azure/sql-database/sql-database-elastic-pool, [Online; accessed 23-May-2018], Microsoft Corporation.

[14] U. of Art and D. Helsinki, *Comparative study*, http://www2.uiah.fi/projekti/metodi/172.htm, [Online; accessed 09-June-2018], University of Art and Design Helsinki, 2007.

[15] W. Britts, *Select database (sedb) - a database selection process model*, 2015.

[16] *Iiie xplore digital library*, https://ieeexplore-ieee-org.focus.lib.kth.se/Xplore/home.jsp, [Online; accessed 10-April-2018], IEEE.

[17] *Acm digital library*, https://dl-acm-org.focus.lib.kth.se/dl.cfm?coll=portal&dl=ACM, [Online; accessed 10-April-2018], ACM.

[18] *Scopus*, https://www-scopus-com.focus.lib.kth.se/search/form.uri?display=basic, [Online; accessed 10-April-2018], Elsevier.

[19] *Identifying research instruments*, http://guides.library.duq.edu/researchinstruments, [Online; accessed 07-June-2018], Duquesne University.

[20] P. Kotler, G. Armstrong, S. H. Ang, C. T. Tan, O. H.-M. Yau, and S. M. Leong, *Principle of marketing*.

[21] P. Hudadoff, *The customer value proposition*, http://www.engr.colostate.edu/~marchese/stese/reading2.pdf, Online; accessed 16-June-2018, Applied Product Marketing LLC.

[22] *Getting started with entity framework 6 code first using mvc 5*, https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application, [Online; accessed 14-May-2018], Microsoft Corporation.

[23] O. A. Keshav Gupta Sandeep Kumar, *Data isolation in multi-tenant saas environment*, https://ieeexplore.ieee.org.focus.lib.kth.se/document/7813917/, [Online; accessed 10-April-2018], School of Computing Science and Engineering, Galgotias University, Greater Noida.

Appendices

Appendix - Contents

A Microsoft's Comparison of Tenancy Models

Table 1: A comparison made by Microsoft. Taken from *Multi-tenant SaaS database tenancy patterns* [4]

Measurement	**Standalone App**	**Database-per-tenant**	**Sharded multi-tenant**
Scale	Medium 1-100s	Very high 1-100,000s	Unlimited 1-1,000,000s
Tenant isolation	Very high	High	Low; except for any singleton tenant (that is alone in a MT db).
Database cost per tenant	High; is sized for peaks.	Low; pools used.	Lowest, for small tenants in MT DBs.
Performance monitoring and management	Per-tenant only	Aggregate + per-tenant	Aggregate; although is per-tenant only for singletons
Development complexity	Low	Low	Medium; due to sharding
Operational complexity	Low-High. Individually simple, complex at scale.	Low-Medium. Patterns address complexity at scale.	Low-High. Individual tenant management is complex.

B Interview

In this part of the appendix, the conducted interviews of this study are provided.

B.1 Interview 1

The first interview was conducted with Andreas Söderqvist at CRM Treasury Systems AB.

Question	Answer
IC 1: What is your profession?	*Cheif Technology Officer* (CTO)
IC 2: What roles have you had?	*Developer*, *Scrum Master* and *Agile Coach*
IC 3: For how long have you had those roles?	3 years as CTO. 11 years as a Developer. 1.5 years as Scrum Master. 1.5 years as Agile Coach.
IC 4: Have you ever participated in any tenancy model selection project?	Yes. But it was not a active choice. In this case *single tenant application, single database* was chosen because it is the less complex tenancy model. Also the tenants wanted different versions of the SaaS application.
IC 5: What is your education?	Bachelor of Science in Engineering, Degree Programme in Computer Science.
SeC 1: Are the phases relevant to be part of the TMSG model?	Andreas thinks that all the phases are relevant. He mentions that we should try to be more clear about how the transition from phase *Analyse Business* to the phase *Select Tenancy Model* should be conducted.

SeC 2: Are the phases applicable?	Andreas says that all the phases are applicable, but he missing a few things. One of the things he is missing with the model is some kind of grading measurement in the phase of *Transform into Criteria*. This could possibly make the phase of *Select Tenancy Model* more feasible. He mentions that is very important to understand the importance of analyse the inner state of the business in the phase of *Analyse Business* (at the moment of interviewing, we are highlighting the external parts of the business more, like the paragraph of identifying customers and marketing segments). At the moment of the interview, we are using the word value proposition wildly and vague in the presentation of TMSG; the interviewee says that we need to be more clear in what we mean when using it. Finally, Andreas say that it can be wise to implement a *Proof Of Concept* (POC) of the selected tenancy model before implementing it into the intended SaaS application.
SyC 1: Are there any redundant or unnecessary phases in the TMSG model?	No. But he points out that if the phases previous to the *Evaluation of Selection* phase are done correctly, the final phase will be unnecessary.
SyC 2: Are there any important phases related to selecting a tenancy model that are not addressed in the TMSG model?	No.

SyC 3: What do you think about the proposed sequence of phases in the TMSG model?	The current sequence of phases is reasonable.
F 1: With minor adaptations, is it possible to use the TMSG model in various tenancy model selection projects?	Andreas says that the criteria of *Transform into Criteria* phase must be further developed if the TMSG model is going to be more general and applicable in more selection projects.
U 1: Does the TMSG model guide you in selecting a tenancy model?	He believes that the TMSG model is guiding.
U 2: Would you consider using TMSG model for selecting a tenancy model?	Yes, he would use it.

B.2 Interview 2

The second interview was conducted with Daniel Svensson at CRM Treasury Systems AB.

Question	**Answer**
IC 1: What is your profession?	Software Engineer
IC 2: What roles have you had?	Project leader, software architect and roles that are focusing on security and IT-infrastructure.
IC 3: For how long have you had those roles?	Software engineer since 2011. Project leader for 2 year long time.
IC 4: Have you ever participated in any tenancy model selection project?	No

IC 5: What is your education?	Master of Science in Engineering, Degree Programme in Computer Science.
SeC 1: Are the phases relevant to be part of the TMSG model?	Yes.

SeC 2: Are the phases applicable?	Yes, with some modifications. Daniel mentions the same thing as Andreas from interview 1, it is very important to understand the importance of analyse the inner state of the business in the phase of *Analyse Business* (at the moment of interviewing, we are highlighting the external parts of the business more, like the paragraph of identifying customers and marketing segments). He says that it is very important to understand the internal processes of the business and the product that is offered, examples are: IT operations, development and how invoices are handled. He adds that it is important to understand what type of data that the business are handling. The handling of a certain type of data can be governed by laws and rules, like *Payment Card Industries* (PCI). It is important to see the risks of a solution from the provider's perspective; this can vary a lot depending in what type of business area that you are doing business within. Daniel would like to see some additional questions in *Transform into Criteria* phase. For the isolation questions, he said that the following questions are suitable and more guiding: *"What are the effects of a tenant is seeing another tenant's data?"* and *"What are the risks that someone would try to hack the system?"*, for example: IT-infrastructure or SaaS application.

SeC 2 (Continuation): Are the phases applicable?	These two questions of isolation would replace the first question of *In what extent does the tenants data need to be protected?* in isolation, since they are more descriptive. He says that it would be interesting to add the question of *What is the continuous cost for a tenant?* to the criterion of tenant-cost. Daniel also thinks that it would be nice to see how the cost of maintainability is affected when adding a tenant, this could go under the criterion of tenancy-cost. He adds that it is important to see the advantages and disadvantages of the choosen tenancy model. He says that the idea of having a POC in the *Evaluation of Selection* that Andreas mentioned in interview 1, could give a better estimation if the selected tenancy model is sufficient to fulfill the criteria of the business.
SyC 1: Are there any redundant or unnecessary phases in the TMSG model?	No.
SyC 2: Are there any important phases related to selecting a tenancy model that are not addressed in the TMSG model?	No.
SyC 3: What do you think about the proposed sequence of phases in the TMSG model?	After *Evaluation of Selection* phase, go back to *Transform into Criteria* phase and compare the output from *Evaluation of Selection* with the criteria that was crystallized in that phase.

F 1: With minor adaptations, is it possible to use the TMSG model in various tenancy model selection projects?	Daniel says that it looks quite good in its current state. Questions in *Transform into Criteria* phase must be added and some needs to be customized, to make the TMSG model more general and flexible.
U 1: Does the TMSG model guide you in selecting a tenancy model?	Yes, provided that the *Select Tenancy Model* is of good quality (the comparison was never shown).
U 2: Would you consider using TMSG model for selecting a tenancy model?	Yes, he would considering it. In practice, it is a risk that the first phase is skipped, at least where he is working now. The origin to the risk is that there is a believe that the organization's requirements are already known, hence many requirements are missed.

C Comparison Table

In this part of the appendix the comparison for TMSG model is presented.

Table 4: A comparison of tenancy models

Measurement	**Single Tenant Application, Single Database**	**Multi Tenant Application, Single Database**	**Multi Tenant Application, Sharded Database**
Isolation	Very High	High	Low
Tenancy-cost	High	Low	Very Low
Performance	High	Low	Low
Development	Low	Medium	High
Maintainability	Very Low	Medium	Medium - Very High

D Source Code

The following hyperlink goes to a repository in Github that contains the source code for this study: https://github.com/Patrik-Svensson/Kandidatexamensarbete

TRITA TRITA-EECS-EX-2018:588