universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„A Configuration Approach for Database Audit Systems "

verfasst von / submitted by

Nedim Rifatbegovic BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2021 / Vienna, 2021

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 926
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Wirtschaftsinformatik
Betreut von / Supervisor:	Univ.-Prof. Dipl.-Ing. Dr. Dr. Gerald Quirchmayr
Mitbetreut von / Co-Supervisor:	

Table of Contents

Table of Contents 2
Abstract 4
1. Introduction 6
2. Related Work 9
3. Research Approach 19
4. Requirements Analysis 22
5. A Model for the Support of Database Audits 28
6. Implementation Approach and Prototype 42
7. Scenario-Based Testing of the Prototypes Functionality and Usability 82
8. Discussion of the Results 127
9. Conclusion 130
References 133
Figures 135

Acknowledgments

Many thanks to my brother, Kenan, who supported me by giving me advice on the improvement of certain descriptions and sentences in the thesis, and I hope I can also help in situations where you need my input! Most of all, I would like to express my thanks to my parents who have made it possible for me to pursue my studies and have equipped me with the necessary confidence and skills.

The proof reading carried out by Andrada, was essential for smoothing out the language of the thesis, especially for getting rid of many typing errors I would never have found myself.

I would like to take the opportunity to thank my colleagues who gave me a lot of very valuable feedback on the prototype.

My sincere gratitude to my supervisor, Prof. Quirchmayr, for his advice, and his guidance through this research.

Abstract

To evaluate the core systems holding the most valuable assets of a company, routine IT audits are carried out to evaluate their ability to protect these assets. This thesis proposes a model for auditing relational databases, which aims to fulfill multiple defined controls, but also some of the characteristics defined by the COBIT framework. The COBIT framework has also been the guideline for the controls implemented with the proposed model. A prototype has been developed, which consists of a TypeScript backend and a modern web interface made with React. Scenario-based testing of the prototype's functionality and usability has been done, and the discussion of the major results has been shown. Finally, the thesis concludes with the most important findings and the prototype results, offering a few possibilities for further work on the model and the prototype.

Keywords: Database, Audit, React, MariaDB, COBIT

Kurzfassung

Um die Kernsysteme, die die wertvollsten Vermögenswerte eines Unternehmens enthalten, zu bewerten, werden routinemäßige IT-Audits durchgeführt, um ihre Fähigkeit zu bewerten, diese Vermögenswerte zu schützen. In dieser Arbeit wird ein Modell für die Prüfung von relationalen Datenbanken vorgeschlagen, das darauf abzielt, mehrere definierte Kontrollen, aber auch einige der vom COBIT-Framework definierten Merkmale zu erfüllen. Das COBIT-Framework war auch die Leitlinie für die Kontrollen, die mit dem vorgeschlagenen Modell implementiert wurden. Es wurde ein Prototyp entwickelt, der aus einem TypeScript-Backend und einer modernen, mit React erstellten Weboberfläche besteht. Es wurden Szenario-basierte Tests der Funktionalität und der Benutzerfreundlichkeit des Prototyps durchgeführt und die wichtigsten Ergebnisse wurden diskutiert. Schließlich schließt die Arbeit mit den wichtigsten Erkenntnissen und den Ergebnissen des Prototyps und bietet einige Möglichkeiten für die weitere Arbeit an dem Modell und dem Prototyp.

Stichwörter: Datenbank, Audit, React, MariaDB, COBIT

1. Introduction

The goal of this research was to develop a model and implement a prototype that would be able to audit databases and generate reports, showing potential risks, for internal and external auditors. In my experience working as an IT auditor, I noticed that there isn´t much focus on auditing databases. After discussing this topic with some CISA (Certified Information Systems Auditor) certified auditors, it came to my knowledge that there is an interest for these types of systems, and that the main focus of IT auditors has been auditing the application layer in companies, but also that the database audits are becoming more important each year. The model proposed by this thesis focuses on relational databases (MariaDB) since it was one of the most used databases in the companies where I have been auditing. A prerequisite for this model is that the audited company must use a ticket system to document the changes and issues regarding the database (Atlassian Jira will be used for the prototype since it is an industry-standard and one of the most used ticket systems).

Before going any further into details of the research and development that has been done, the importance of audits and database audits needs to be explained first. As described on the official ITIL Page [1], IT audits focus on making sure that the IT services are aligned with the business requirements of the organization. During this process, the auditor examines processes and past occurrences that may have not been done as defined in the organization`s goal. An auditor can be an employee of an organization (internal auditor), but an auditor can also come from a reputable company specialized in auditing. Audits make it possible to detect risk factors and check if the audited organizations are following defined controls which help to minimize or avoid these risks. The framework which has been the guideline for this research is COBIT - Control Objectives for Information Technologies [2], which is one of the most used frameworks, to the best of my knowledge. As mentioned in the paper [3], the most important components of an information system which is being audited are Operating Systems Audit, Internet Servers Audit, Database Audit (which this thesis is focusing on), Data Centre and Disaster Recovery Audit, Audit of Applications, Internal Network (LAN) Audit and External Network (WLAN) & Mobile Device Audit. Each of these fields needs to be audited with different technical means (both software and hardware can be audited). Database audits have been chosen as the research field of this thesis, for multiple reasons. In my experience as an IT auditor, I had noticed that this field does

not have many software solutions which could audit databases. As the author in the documentation [4] describes, in the year 2008 there have been already 285 million records that had been compromised. Over 70% of the data breaches had been caused by external sources, and from the internal sources 20% have been done by insiders, and 32% by business partners. The top exploited vulnerability according to the previously mentioned document, has been unauthorized access via default accounts. The average total cost per incident has been around 6.65 million dollars. Over 90% of the stolen records have been linked to organized crime. Some of the database vulnerabilities that get exploited are default accounts and passwords, easily guessed passwords, missing patches, excessive privileges, etc. But there are also external threats like for example SQL-injections and weak or non-existent audit controls. Many of these breaches can be easily avoided by regular audits and by following best practices and frameworks (for example controls defined by COBIT). The thesis will focus on relational databases since they have been used most often in my experience working as an IT auditor. The relational database used for the prototype and testing with test data has been MariaDB [5]. Databases are collections of related files. As described in the mentioned article, most databases today are relational databases. They are called relational because they deal with tables of data related to a common field. That means that they save the data and enable access to it by relating to one another. In these types of databases, each row in the table contains a unique ID, also known as the key.

To control the handling of errors, updates, restores, and backups a ticket system will be needed. Most companies nowadays are using ticket systems that allow their IT support to organize, document, and handle different tasks. These tickets are also important for forensic purposes since they document the flow of certain actions, show who and when executed them and they can also show the communication regarding certain issues. The ticket system used for this thesis is Atlassian Jira, which is one of the standards in the industry and one of the most used ticket systems.

For this research, a test environment in a Windows 10 virtual machine has been set up, where a test relational database (MariaDB) and a ticket system (Atlassian Jira) have been installed and filled with data. The prototype is using the latest technologies, where the backend is written in TypeScript (using TypeORM as an ORM mapper) and the frontend is using React with TypeScript (where styled-components and react-

bootstrap are being used for the UI to make it accessible on multiple devices, like for example smartphones, tablets, and computers).

The thesis consists of nine parts. The introduction has shown a general overview of the research field. Followed by the related work chapter, where the COBIT controls, and similar research and approaches are discussed. After the related work chapter, the research approach will be described, where the plans and procedures of the research will be explained. In the requirements analysis chapter, the requirements will be presented, and the specific COBIT controls will be described in detail. After that, the proposed model will be described, showing the approach used for fulfilling the previously described requirements and defining which answers should the prototype provide. There will be a big focus on the implementation of the prototype, where the whole architecture of the solution will be explained, all the implemented scenarios, and the technical solution for the addressed controls. This chapter is followed by a chapter about the scenario-based testing of the functionality and usability of the prototype, where the application testing will be described, but also multiple interviews will be shown, where few auditors and a web designer have graded the user interface and user experience, and answered a few questions on the functionality and usability of the solution. The results of the research and the interviews will be discussed. Finally, the thesis will be concluded with a short overview of the research and a description of possible future work on the model and prototype.

2. Related Work

To the best of my knowledge, there is no such software solution for auditing databases like the prototype presented in this thesis. There are few commercial products, but they are specialized in specific products and their reports are not generalized [3], but a short overview of the commercial solutions will be given later in this chapter. During the research, I noticed that there aren´t many scientific papers on this topic. Out of the few which I found, most of them have been following the same guideline, the COBIT framework, which has been also the basis for the implemented controls of the prototype presented in this thesis.

COBIT (Control Objectives for Information and Related Technology) is a framework created by ISACA (Information Systems Audit and Control Association) for IT governance and management. Information and technology are the most valuable assets of many enterprises as described in [2] . Successful companies recognize their importance and use them to maximize their value. But with that come increased regulatory compliance and increased associated risks. The management of IT-controls related risks in the core of IT governance. As stated in COBIT [2], IT governance is the responsibility of executives and the board of directors. Their responsibilities are leadership, organizing structures, and processes that ensure that the IT supports the enterprise's objectives. The IT governance also integrates good practices, that the enterprise needs to fully take advantage of to maximize the benefits provided by the information. COBIT provides a collection of best practices over this domain. These practices are the results of a collection made by experts in this field and help the enterprise maximize their IT-enabled investments, but also avoid potential risks. COBIT considers the business needs and makes links to the business requirements. It organizes IT activities into an accepted process model. COBIT is divided into four domains and 34 processes. The framework defines control objectives, which provide a set of requirements that need to be considered by the management to ensure that the controls are being effective. Implementing the COBIT framework over their IT gives the enterprise multiple benefits [2]:

- Better alignment, based on a business focus
- An overview, understandable to management, of what the IT does

- Clear ownership and responsibilities based on process orientation - which is also important for forensics
- General acceptability with third parties and regulators – which is also why the prototype presented in this thesis considers the use cases for internal, but also external auditors
- Shared understanding amongst all stakeholders, based on a common language
- Fulfillment of the COSO requirements for the IT control environment (COSO is a generally accepted internal control framework for enterprises, while COBIT is the generally accepted internal control framework for IT).

This thesis is also following the COBIT 4.1 version since I have the most experience using this version during the audits that I have done. The COBIT 5 version has been also considered, but to the best of my knowledge, no major changes have been noticed, and the controls which have been implemented in the presented prototype have not been changed. As described in COBIT, top management is increasingly realizing the significant impact that information can have on the success of their enterprise. So, the management often demands to know how the information in the enterprise is being managed, if it is likely that it will achieve the defined objectives. The controls implemented by the presented prototype will be discussed in detail in the chapter on requirements analysis, and the chapter about the model of the solution.

As previously mentioned, COBIT has been the guideline for most of the papers that I have found during my research. The paper [3], also aims to identify techniques that should help perform IT database audits. An interesting summary has been presented, by summarizing the issues related to the national and international IT audit regulations. The author emphasizes the importance of intellectual production (services, research, and development, competition, etc.), and that this production is using information (which is also its most precious asset). As stated in this paper, the daily activities of enterprises, but also individuals, are dominated by the electronic processing of information. Its crucial components are hardware resources, software resources, human resources, and data. The economic activities are orientating towards processes that rely on the information (for example financial transactions, stock exchange, auctions, establishing prices, etc.). But the information technologies are rapidly mutating. The companies need to be able to adapt and auditing this system has become a necessity, to make sure that the operations of the enterprise are

following the defined goals. As the author states, the database audit must provide the management with relevant reports regarding the operating and the protection of the data within their organization. As described in the paper, information audits cannot be included within other types of audits, because information audits have specific objectives, that have specific procedures, and needed tools. As described in this paper [3], COBIT defines seven important characteristics of information, those are:

- Availability – the information must be available at any time during the decision process
- Integrity – the content and accuracy of the data must be following the rules and expectations of the organization
- Compliance – the logical structure of information and its concrete values must reflect the actual level of processes it characterizes
- Reliability – the information must relate to the specific decision-making process which is being served
- Efficiency – the information must be provided with the lowest consumption of resources
- Effectiveness – the information must be relevant, accurate, and timely provided for decision making
- Confidentiality – the information must be provided only to users to whom they are intended to be delivered.

The model proposed by this thesis covers the availability, compliance, reliability, and confidentiality of information. The details of how these characteristics are fulfilled will be explained later in the chapter about the model of the solution. As the author mentions in [3], the ISO 27000 specific standard, in the EU, prescribes three characteristics defined in COBIT. But the EU standard allows the auditor to audit other analytical characteristics that are defined by him. The author also focuses on auditing databases, due to their importance for enterprises. The paper lists some commercial solutions, those are:

- Cross-Platform Audit – auditing information systems with multiple database engines
- DB Audit and Security 360 – auditing Oracle, SQL Server, DB2, Sybase, and MySQL database servers

- Azure SQL Database Auditing basics – auditing functions, data access, database scheme, and sub-scheme changes
- Idera`s SQL compliance manager and ApexSQL Tools ApexSQL Audit – auditing SQL server databases.

The problem that has been identified by this analysis is that the audit reports of these software products are not uniform, and in most cases have different meanings or content. The authors define multiple assumptions while developing their method. Those assumptions are:

- Information characteristics like previously defined by COBIT are important elements for the analysis of IT audit
- The author states that those seven characteristics have different degrees of importance for the final analysis
- The paper also defines 21 steps, that are used for database auditing, where the proposed method aims to verify the seven characteristics
- The last assumption is that the designed method must be able to be generalized, simple to use and that the results generated by the audit need to be clear for the decision-makers.

To display the results, the authors are using the Balanced Scorecard, where they define five levels of assessment grades, from 1 to 5 for each evaluation process. Those values are 1 – improperly, 2-accordingly, 3-medium, 4-high, and 5-very high. The author states that these grades should be generated automatically for each process, concerning the approach degree of information characteristics. This research uses a similar approach, where the balanced scorecard has been modified. More details on how the balanced scorecard is being generated and formatted will be shown in the chapter about the model proposed by the thesis. Some of the processes that the authors propose (and that are also being implemented in this thesis) are:

- Verifying that the database is running under a software version that is being supported by the vendor
- Checking for easily guessed passwords
- Verifying that the database permissions are granted or revoked for specific levels of user groups

- Reviewing database permissions granted to individuals instead of groups of roles (this thesis implements a similar control, where user groups are checked, as are the users. The presented prototype is checking the title (position in the enterprise) of every user, and if is it corresponding to the rights that a person with that title should have. For example, that a trainee is not having admin rights. But more details on the controls will be presented in the chapter about the solution of the thesis).

Another important aspect is the chain of custody for forensic purposes as described in [6]. The authors state that during forensic database investigations, audit records become a crucial evidential element (for example when certain events can be attributed to insider activity). Also, during my experiences, while doing IT audits, or ISAE 3402 Type one and Type two reports, I noticed that auditing companies put a great emphasis on the chain of custody. For example, that each action is documented, and can be traced (for example who and when has executed which task). The authors of the mentioned paper propose a proactive approach. They consider multiple aspects which they cover with their proposed method, for example, role segregation, evidence provenance, event timelines, and causality. Their method collects and preserves database audit records, which can be used as digital evidence. They implemented triggers which store the data in a vector-clock-based timeline. As the authors mention, database forensics allows the investigation of malicious activities (which have been performed by employees or insiders). But it can be difficult to consider audit records as legally relevant because if there is a lack of accountability and forensic features, that would enable the malicious insiders to hide their activities. Like in the example presented in [6], there have been unauthorized payments that were made in a public institution in Ecuador, which used privileged system credentials, to make those orders look legitimate. In this case, even the data that was collected was in the end inadequate. For this reason, the authors proposed a method that would establish an unbroken accountability trail – also known as Chain of Custody. This is a difficult task because the reactive approach can´t always properly analyze the users (insider/employee) actions. The reactive database forensics approach consists of bottom-up methods, but these may not be appropriate for investigating databases, because the investigation is relying on the knowledge and expertise of the person investigating the database. The other approach (which the authors took) is a proactive

approach, which is top-down oriented. It bases on the premise that databases were designed with forensic ready features, which means that they already have specific triggers which are needed for the auditing of the activities executed by the enterprise's insiders. With those triggers' records can be generated and preserved to get a clearer picture of the activities. As the authors state, digital forensics has been considered as a scientific approach with the purpose of identifying, collecting, validating, preserving, and analyzing the digital evidence. The authors define four principles of the chain of custody approach. Those are:

- No action (evidence) which is executed by any user can be changed
- If the original data is required, the evidence must also be provided, explaining the relevance and implications of such data
- All events must be generated, collected, and preserved (so that – best-case – an independent third party can access and examine the collected data, to make conclusions)
- The person responsible for the investigation is also responsible for ensuring that the application follows these principles.

As stated by the authors, accountability and forensics are very important for each database investigation, because the digital evidence needs to explain each occurrence of data manipulation language (DML) events (where insertions, deletions, and updates have been executed). To achieve this, the chain of custody must be initiated in parallel with each database operation. The authors state that role segregation, evidence provenance, event timelines, and causality as the chain of custody requirements need to be defined and collected to ensure that the environment is forensically ready. The authors state in [6], that there needs to be a clear functional separation, to prevent changes in audit records. It is also stated that the administrator role should not be the one in charge of managing the audit functions, instead, the enterprise needs a forensic role who should be in charge, to avoid the violation of the administrative functions (for example disabling the audit mechanisms, etc.). Function segregation is needed to prevent a user from having administrator and forensics permissions, at the same time. After each event, the method proposed in [6], is collecting six different attributes. Those are:

- What audit record has been generated (defined with a record identifier – Id)

- When the audit record was generated (real / hardware clock timestamp)
- Why the audit record was generated (a type of event – insert, update, or delete)
- Who the actor is (user-identifier)
- What the DB actor's role is (the authors defined three types – DBadmin, DBforensic, and DBuser)
- Where the DML event has been triggered (collecting the user's IP address).

As previously stated, the audit records need to be generated (every time a DML event is triggered, an entry for the audit records is also being generated), collected (after each DML event, the defined six attributes are being saved in the newly generated audit record), and preserved (so that in case of forensic investigations the data can be analyzed). For example, if a user triggers an insert call, where the data needs to be inserted into the database, a DML event is triggered. For that event, a corresponding audit record is being generated and the 6 defined attributes are being saved in the audit records. This makes it possible to construct a timeline of events and to explain their interactions, which can be used then for later investigations. To achieve the role segregation, the authors implemented a transactional and a forensics database. So, the user with the administrative rights is responsible for the transactional database, while the user with the forensics rights is responsible for the forensics routines, the generation, collection, and preservation of DML event-related audit records. To test the generation of the data, the authors have generated a synthetic workload produced by a Master Event Generation Server (MeGen) and two Event Generation Clients (CeGen). The MeGen represents a master terminal, which is coordinating the activities to emulate a distributed environment. The CeGen represent two terminals, which produce test DML requests. The authors have implemented triggers and the stored procedures as external forensic routines (with enable/disable permissions which are assigned only to the forensic role user – this prevents any malicious misusage of privileges). The evidence is generated every time a DML request has been sent from the CeGen to the transactional database. After that, the transactional database data tables have corresponding audit tables in the forensic database, in which the generated data is being collected and stored (including the defined descriptive attributes). The authors use a causal table from the forensic database to build an event timeline, which can be then used for analysis. As shown in [6], the authors have generated 720 DML request samples to test their model, and analyze the results. The

authors conclude with the statement that timeliness and causality are mutually related chain of custody requirements, because, as stated, one cannot be explained without the other.

Some functionalities in the solution of this thesis, have been inspired by the approach presented by [6], but instead of creating a separate database, I have used the ticket system Atlassian Jira and saved the log details in tickets (which are saved in their own PostgreSQL database). More details on this part of the solution will be presented later in one of the following chapters.

During my research, I also found an interesting document created by ISACA, on the best practices in database auditing [7]. The author has defined a data security risk and compliance life cycle, which consists of six components. Those are: discover, classify, assess, prioritize, fix, monitor. The stage of discovery produces a database or asset inventory. The classification finds sensitive data and associated requirements. The assessment scans the database for vulnerabilities, misconfigurations/configuration changes, and user entitlements. The prioritization is combining info from the classification and assessment phase and is determining which actions should be executed (what should be fixed, what should be monitored), and in which order to do the tasks. The fixing stage is creating and running fix scripts, applying patches, and creating monitoring policies to implement compensating controls. The monitoring stage is auditing the privileged access (and any access to sensitive data). Monitoring for possible exploits or any unusual and suspicious behavior. The report recommends multiple steps which should be done to secure databases. For example, checking:

- if the configuration has been configured securely
- if the latest patches have been installed (to stay on top of all security alerts)
- if the least privilege principle has been implemented (if the user rights have been given appropriately)
- if multiple levels of security have been implemented (regular scans of the database for vulnerabilities, active database activity monitoring, database intrusion detection, encryption of data)
- if third party database installations have been installed
- if a good password policy has been configured
- if all non-used functionalities have been disabled

The author [7] has also made a list of recommendations for controls that should be done during periodical audits of the database systems:

- Check for object and system permissions (views, procedures, tables, file, folder, registry, etc. permissions)
- Looking if there are new database installations (for example third-party products installed on the database servers, if found they should be secured or removed)
- Search for users with DBA privileges
- Auditing database configuration and settings (if the settings have been changed by a system upgrade or patch, this could open the database to attacks, if they have been changed without an update, that could mean that the database has been already compromised)

The mentioned paper has given a good overview of the complexity of securing a database, and which possible tasks can be done to avoid risks. Concluding the report, the author has also given some short management recommendations, for example, real-time monitoring of the database (database audits and intrusion detection), scanning logs, implementing real-time alerting, and keeping a library of best practices.

The authors of the paper [8] present an interesting framework for database auditing. As stated by the authors, database auditing can help strengthen the security of the database. Their proposed method will log the database activities through analyzing network traffic, executing audit analysis through event correlation, and generating alarms if any anomalies or violations of the defined security regulations have been detected. They claim that their method, compared to native auditing mechanisms in databases, is providing zero impact to the performance of the database or the applications that access the database. They mention also that it is important that enterprises are using third-party auditing components since it complies with the principle of the separation duties. The authors state that database systems are affecting most of the aspects of our lives (bank accounts, medical records, employment records, phone records, etc.). They mention that almost all databases used for these fields are relational. If the databases that are holding this data, are not secure, then it can have a potentially devastating impact on our lives. Enterprises are, like described in the paper, increasingly adopting their database systems as key data management technologies for their operations and decision making, this makes the

security of the databases crucial for the success of the enterprises. Users are getting database access, including DBA, such as application developers, software engineers, marketing, HR, and customer support employees. In case, the company didn't implement measures that are monitoring and auditing these users, the assets of the company are at high risk of being compromised. The solution to avoid the risks is to implement database auditing. Database auditing consists basically of observing a database and of being aware of the actions which are executed by the database users. Especially the users with administration rights, that have access to sensitive data. The authors in [8] state, that the auditing actions are separated from the DBA user, and that enterprises should give these rights to compliance officers or other groups (for example, security or networking operations team), because of the segregation of duties, which is critical for the security of the database. The approach the authors took is to tap into the network of the enterprise with an agent, which will monitor the traffic flowing into and out of the database system. They then extract the information which they need for the audit analysis and create alarms in case of anomalies or violations of the defined security regulations. It is also pointed out that the logging of the data is very important, because with the right logging method, database activities and other related information can be recorded and used for the audit analysis (for example used for detecting if the security policies are being violated – if they are being violated, to provide the evidence, to discover potential attacks on the database, and in case of any damage – to recover the database). Their logging system is based on a passive mode, because of its high efficiency and good extendibility, and is divided into three stages: packet capturing, packet parsing and data storage. Concluding, they remark that this approach has a weakness, in case the database communication has been encrypted, this method will be invalid.

The COBIT framework and the mentioned articles and scientific papers have been an inspiration for the model proposed in this thesis. The requirements that need to be fulfilled by the model will be described in detail in the chapter on requirement analysis. Concluding the related work chapter, to the best of my knowledge, I found no previous method which proposes the same approach as the solution in this thesis does. Besides COBIT, they helped me get a better overview of the database audit infrastructure and which controls are essential while generating a database audit report and analyzing the collected data.

3. Research Approach

The proposed model presented in this thesis follows the research framework suggested by the authors in [9]. The authors state that the framework consists of understanding, executing, and evaluating the information systems research, in the combination of behavioral-science and design-science paradigms. The authors also state that the information systems research is composed of people, (business) organizations, and their technologies (which exist or are planned). The information system research uses the environment (people - their roles, capabilities, and characteristics, organizations - their strategies, structure, and processes, and the technology – infrastructure, applications, development capabilities, architecture) to understand the business needs. Also, a knowledge base is needed for the research, to collect applicable knowledge, consisting of frameworks, models, methods, methodologies, etc. During the research, the business needs and the applicable knowledge need to be evaluated.

The master thesis is be based on a collection of information and knowledge gathered during my years working as an IT auditor and the COBIT framework – which has been my guideline while auditing. The COBIT framework has also been used for the controls implemented in this prototype, and some of the previously mentioned papers and reports, but also some discussions with colleagues who have been or are currently working as IT auditors (two of them are also CISA certified IT, auditors).

As mentioned, at the beginning of my research I talked with IT auditors and got insights into the processes of some of the “big four” companies (big four is the nickname used for describing the four largest accounting firms – Deloitte, Ernst & Young (EY), PricewaterhouseCoopers(PwC), and Klynveld Peat Marwick Geordeler (KPMG)), and in their processes on how they are auditing databases, which aspects are important during the process, and which proof needs to be collected during the audit. I also noticed, during these talks, that their focus on the database audit was low, considering how important this field is. Also, during my time working as an IT auditor, I noticed that the database audit has been done without software designed for database audits, most of the database audit analysis consisted of manual work. The IT auditors whom I talked to, agreed that the database audit process needs to be improved and that a

software solution designed for auditing relational databases would be a good approach to improving database audits.

The guideline for the research has been globally accepted and to the best of my knowledge most used framework – COBIT. After researching which controls are database related, those who could have been fulfilled with such a software solution have been selected. The details on which controls have been selected will be presented in the next chapter. The implementation of these controls will be presented in the chapter showing the model of the proposed solution.

After defining the controls that the model should cover, research on existing methods or similar approaches has been done. During my research, I have used multiple online engines (for example Google Scholar, IEEE Xplore, ScienceDirect, and ResearchGate). I noticed that, to the best of my knowledge, there aren´t many scientific papers covering this field. The papers that have been found, and their proposed methods, have had some good solutions, but also some shortcomings. I tried to learn from both the past shortcomings and the good aspects of these solutions, to create a good model that would fulfill the expected results.

After defining the controls that should be implemented, the infrastructure and technology to fulfill these controls needed to be defined.

First, the type of database that will be audited needed to be chosen. Since I had the most experience with auditing relational databases, and since the research proved, like described in the previous chapter, that the relational databases are most commonly used [3], [5], [7], [8]. I decided to use relational databases.

Also, the operating system environment needed to be chosen. After a short research, a clear decision has been made, the test environment will use Microsoft Windows since it is the most used OS [10] globally. For test purposes a virtual machine has been set up, using Windows 10.

The programming language TypeScript has been chosen for both backend and frontend since I had the most experience working with JavaScript and TypeScript. For the backend, TypeScript has been chosen with Node.js and TypeORM as an ORM (object-relational mapper). During the search for the best frontend framework, I was trying to decide between two of the most popular ones (which have been using

TypeScript), I ended up choosing React, since it´s, according to the article [11] newer and more popular than Angular. Also, in my current position, I am a frontend developer working with React, and I already had plenty of experience with this technology. Research has been done after that, to find the best libraries that will be needed to create a good user interface and user experience for the auditors. A decision has been made to make the application usable on multiple different devices (pc / laptop, smartphone, tablet) so that it would deliver an easier experience for the auditors. After the development process, scenario-based testing of the usability and functionality of the prototype has been done with multiple IT auditors and a frontend web designer (the details of this research have been presented in the upcoming chapter about testing the prototype).

After the research was concluded, the requirements have been clearly defined and the requirements analysis will be presented in the next chapter.

4. Requirements Analysis

The requirements have been researched as described in the previous chapters. During the analysis of the requirements, multiple questions have arisen. The model needs to answer multiple questions:

What does the stakeholder need, and who is this model addressing?

- The stakeholders/users whom this model is addressing are internal and external IT auditors. Since the internal IT auditor is auditing only his enterprise, and the external auditors have multiple clients whom they are auditing, the two user types needed to be separated. To moderate the application an application administrator is needed.
- The model needed to take into consideration the requirements and business needs of both types of users.

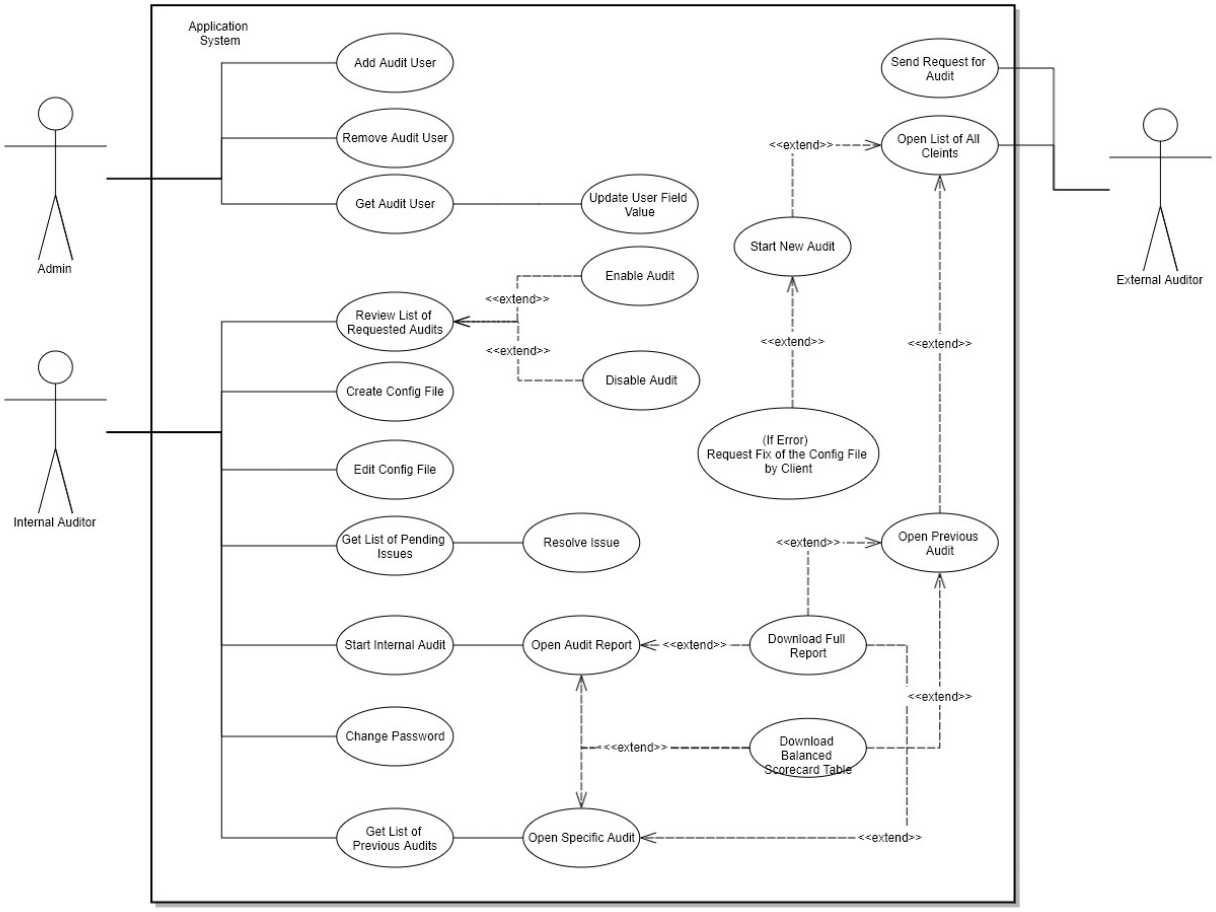

Figure 1: Use Case Diagram – Showing the Three Different User Types

- The use case diagram [Figure 1], is showing the use case for the three different types of users, and actions that need to be implemented for each of them. The first user that needs to be implemented is the administrator (Admin) of the application. He needs to have a few administrative functionalities: to add new auditors to the system (internal or external), to remove users from the system (if requested), and to update specific attributes (if requested). To keep the list of the auditors private each internal auditor needs to have a unique ID, which he can then give to an external auditor (for example via email). For that reason, the internal auditor needs to also have the possibility to view the list of all requested external audits and to enable or disable them. To do the database audit remotely a configuration file is needed for the enterprise's database and ticket system (the need for the ticket system will be described later in this chapter). In case of changes in the enterprise's database or ticket system, the internal auditor needs to also have the possibility to update specific attributes. In case something goes wrong while the external auditor is auditing the enterprise's database, the external auditor has the option to send an error notification to the internal auditor. For that reason, the internal auditor needs to have the functionality to get the list of all error notifications or as it is called in the use case “Pending Issues”, and after he fixed the issues in his configuration file or database, he needs to set the status of the issue to resolve. The internal auditor needs to have the option to audit the enterprise's database. After the audit has been done, the internal auditor needs to download the report of the audit showing its possible risks and he needs to have the ability to download the proofs that have been collected during the database audit. The internal auditor needs to have the ability to review all the previous audits of his database and to download the reports and proof of them. The external auditor needs the functionality to add his clients with their unique ID as previously mentioned. After adding clients, he needs to see the list of all the requests he sent. It is only after the internal auditor accepts his request, then he can open the internal auditors’ details. After opening the details, he needs to have the options to see all the previous database audits that he has done for the specific internal auditor and download the old reports and proofs. He needs to start a new database audit (and after that download the report and proof). In case the audit fails, he needs to have the option to send an error notification to the internal auditor, like

previously mentioned. Both, the internal and external auditors need the possibility to change their passwords.

What are the core software requirements?

- After the described research, the conclusion has been brought that the solution needed to address the audit of relational databases. Due to the flexibility, a web application was chosen as a good solution, this would enable the external auditors to remotely audit the enterprise's database.

What should the software architecture look like?

- After defining the software requirements which the model needed to answer, a software architecture needed to be created, so that it would answer the requirements. The architecture is being shown below in the deployment diagram [Figure 2]. The left side of the deployment diagram is showing the infrastructure for the test enterprise called “Client” – which represents the internal auditors’ database and ticket system - in the diagram. The clients` architecture needs to consist of a relational database, and a ticket system to document different types of logs (errors, changes, recoveries, backups) and to establish a chain of custody for these actions. The right side of the diagram represents the architecture of the application. The applications need a frontend, which needs to use a modern framework, making it possible for the UI to scale to multiple different devices. A backend is needed to process previously defined tasks and to manage the application. Also, a database is needed to save the different users (admin, internal auditor, external auditor), the audits, reports, and the interactions between the different users.

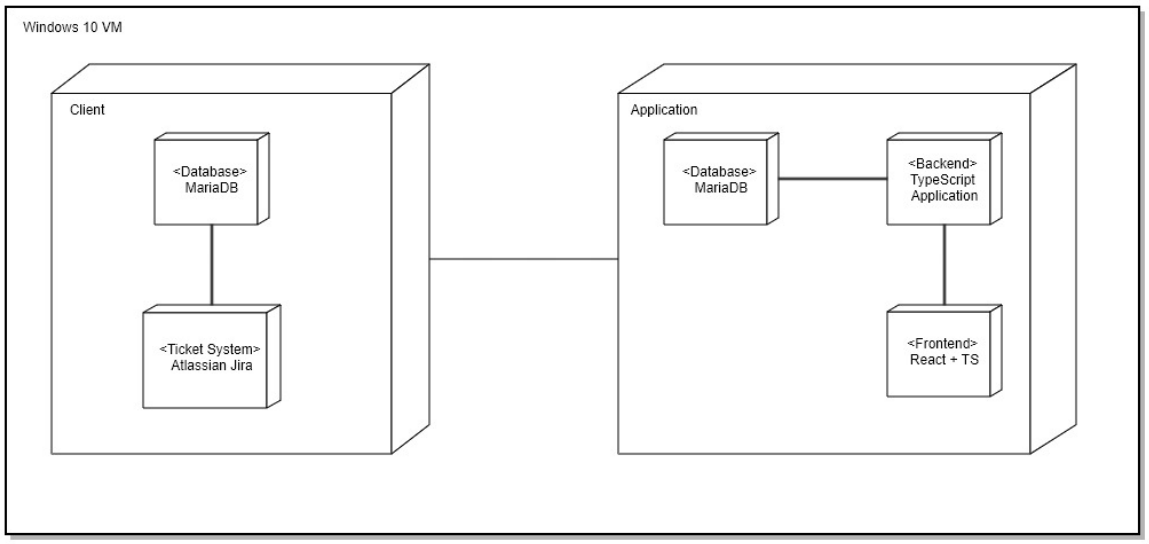

Figure 2: Deployment Diagram – Showing the Configuration of the Prototype and the Test Infrastructure

Which controls needed to be answered?

- As mentioned before, COBIT has been the guideline for this research and the controls that need to be implemented by this model. But there have also been some controls that could also be resolved with this model, as described by [7]. The controls that this model needs to implement are:
    - Checking if the database is running on a supported database software version? [7]
    - COBIT PO2.3 [2]: Checking if user groups have been defined?
    - Checking if the principle of least privilege is implemented, and searching for users with administrator privileges? [7]
    - COBIT DS5 [2] & EU Standard (as described in [12]): Checking the password policy?
    - COBIT AC4 [2]: Checking if unexpected interruptions in data processing are being handled?
    - COBIT DS11.5 [2]: Checking if the procedures for backup and restoration of the database have been implemented, followed, and documented?
    - COBIT AI6.1, AI6.2 [2]: Checking if changes are following a defined process and if they are being documented?

Which characteristics defined by COBIT can be answered by this model?

- As described in [2] and [3], COBIT defines seven important characteristics. The proposed model fulfills four of the described seven. These are availability, compliance, reliability, and confidentiality. The documentations from the logs and tickets system also partly fulfill the integrity characteristic, but integrity has not been added since the prototype is not connected to the applications accessing the database of the enterprise, so the integrity of the incoming data could not be guaranteed.

How can the chain of custody be established?

- Inspired by the paper [6], a chain of custody also needed to be implemented. It is also important that there exists a trusted third party (which the proposed model represents), as described in the paper. To enable the chain of custody, a ticket system needs to be connected with the logs of the database, where all the needed data will be saved (for example who executed, which task, and when).

How can the potential risks be presented?

- The results of the database audit needed to be shown to the auditors. The solution of this model was inspired by the paper [3]. The results of the controls will be shown in a modified balanced scorecard, which consists of the information on what controls have what risk level for each of the four previously mentioned characteristics defined by COBIT.

What should the report contain?

- The report needs to contain the results of the audit, so the auditor needs the possibility to download the generated balanced scorecard. If more details are needed, the auditor needs the option to download the full report containing all proof collected during the database audit, including the ticket collected from the ticket system (assigned person, summary, comments, and status of the ticket are needed to fulfill the chain of custody).

This has been an overview of all the requirements that need to be fulfilled by the proposed model. In the next chapter, the solutions proposed by the model for these

requirements will be presented. In the chapter after that, the implementation of the prototype will be presented, and the technical solution for the proposed model.

5. A Model for the Support of Database Audits

The requirements that this model needs to fulfill have been described in the previous chapter. In this chapter a detailed description of the processes will be given, with which this model aims to fulfill all the defined requirements. The processes will be described in the same order as the requirements, to make them easier to follow for the readers.

The overall architecture of the model is shown in [Figure 3]. The model consists of three different interfaces (one for each of the user types – application administrator, internal auditor, and external auditor). From the interfaces, API calls are sent to the backend of the application, which handles the incoming requests, processes the data, and responds. The application backend is handling the communication with the database of the model, and with the clients (internal auditors) ticket system and database. The technical implementation of the components and details on their communication has been described in the following chapter, also showing the list of all the APIs that have been provided by the backend of the model.

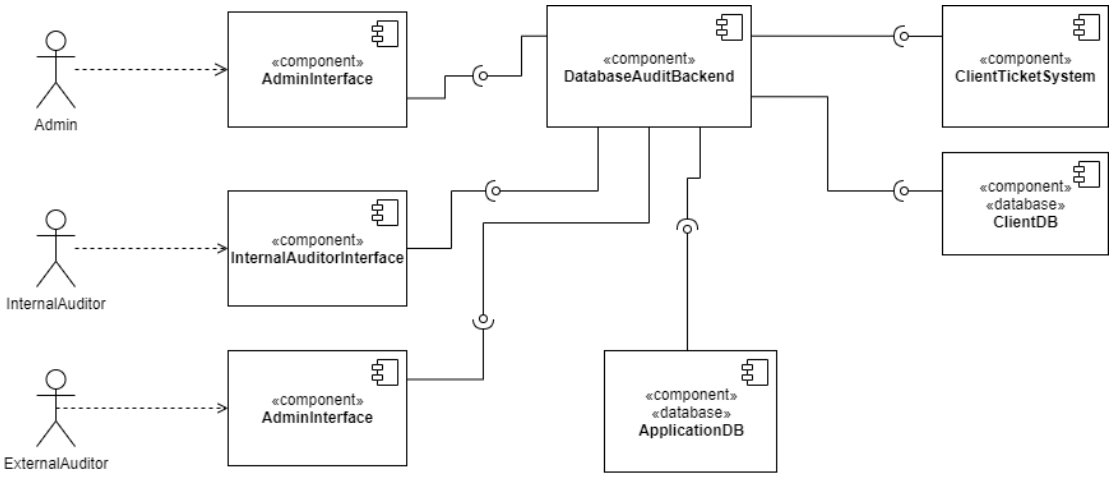

Figure 3: Component Diagram - Presenting the Architecture of the Model

What does the stakeholder need, and who is this model addressing?

- In the previous chapter, it has been defined that three users are needed (administrator, internal auditor, and external auditor), and what functionalities do they need to have.
- To get a better overview of the processes this model proposes, multiple BPMN diagrams will be shown.

- The first BPMN diagram [Figure 4], is representing the administrator's swim lane. After the administrator (“Admin”) logs in, he can choose one of three different actions he can execute. The first one is adding new users, which consists of entering the auditor’s credentials and choosing the auditor type (internal or external). After the request has been sent, the auditor needs to check if the process was a success, if yes, he needs to notify the added user per email, that his account was created and to send him the initial password. In case the user has been already saved in the database the process ends here. Since the email is a unique value, the same user cannot be added multiple times.

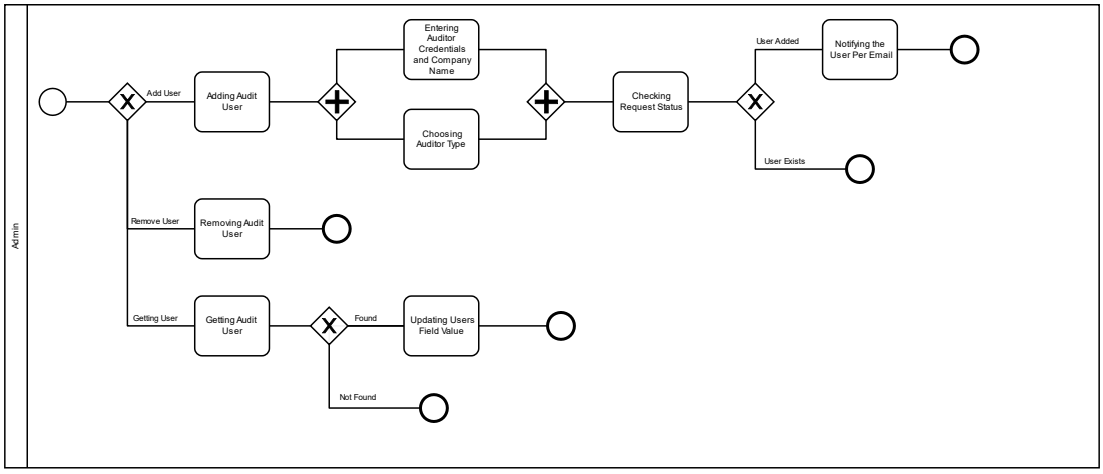

Figure 4: BPMN Diagram – Administrator Processes

- Because of the complexity of the processes, not all functionalities of the internal and external auditor will be shown in the same diagram. The following diagram [Figure 5], is showing the internal auditor handling the adding and updating of the configuration file. After the user makes the call to action, the backend is checking if a config file has been added, if yes, the internal auditor can choose the attribute he wants to update and enter the new value, or in the case of the private key, he can upload a new private key needed for the ticket system. In the case that no config file has been added, the internal auditor needs to fill out the requested fields and submit the config file.

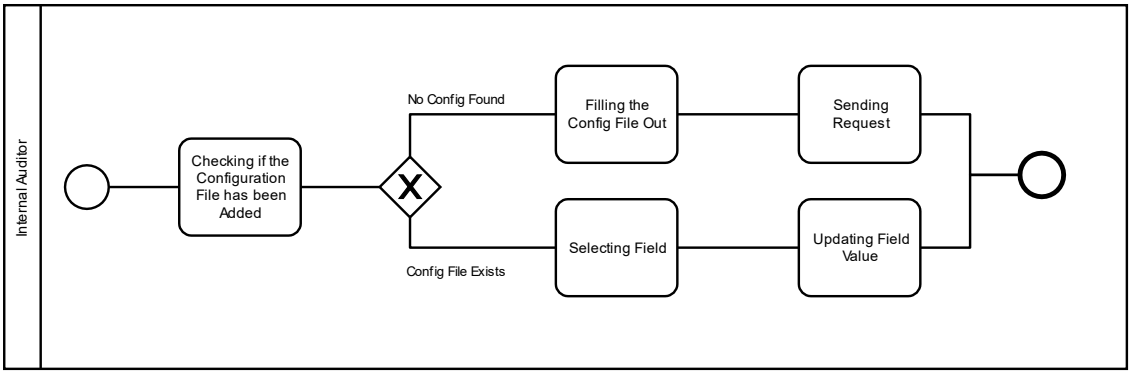

Figure 5: BPMN Diagram – Internal Auditor, Configuration Handling

- After adding the configuration file, the internal auditor can start auditing his database. As shown below in [Figure 6], after the internal auditor starts the internal audit of the enterprise's database, he can decide if he wants to download the files or not. In the case that he doesn`t need the balanced scorecard or the proof collected during the database audit, the process ends there for him. In case he wants to do download one or both, he can download them as PDFs (in separate files), after that, the database audit process ends there for him.

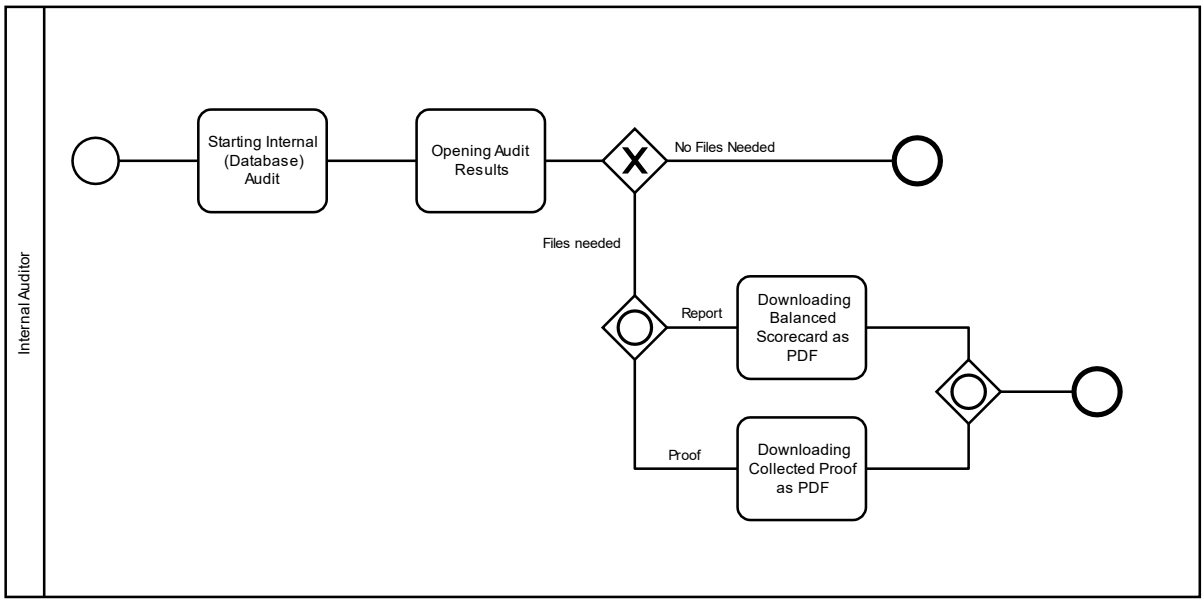

Figure 6: BPMN Diagram – Internal Auditor, Auditing Database

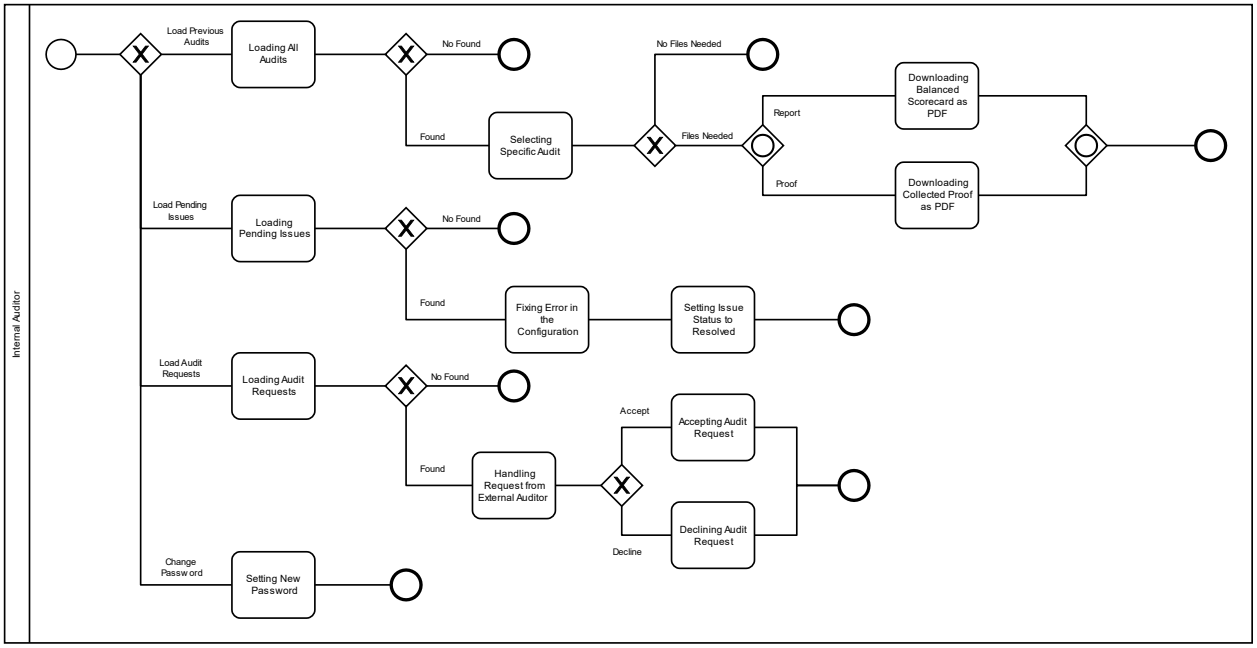

Figure 7: BPMN Diagram – Internal Auditor, Additional Functionalities

- The diagram [Figure 7], is representing the other functionalities that the internal auditor possesses. The internal auditor can request all the audits of his database to be loaded (both internal and external). If none are found the process ends there. In the case he already has some audits, he can select a specific one, where the Balanced Scorecard will open, and he can decide if he wants to download the Balanced Scorecard, or the proof collection (or both in separate files), as PDF. If an external auditor wants to audit the database of the enterprise, he first needs to send an audit request with the internal auditor's unique ID. The internal auditor can decide if he wants to accept or decline the request. In the case he accepts the audit request, the external auditor gains access to the database and he can then start the audit, in case he declines, the external auditor does not get the access and the process ends after that. In the case an external auditor gets an error during the database audit of the internal auditor, he can send him an issue request, so that the internal auditor gets informed that he needs to adjust the config file and fix the issue. The internal auditor can load all the issued requests. If there are none, the process ends there. In the case that there are some requests, he first needs to fix the issue, and then, and only then can he set the status of the request as resolved. The internal auditor can also change his password.

- The actions which can be executed by the external auditor are shown below [Figure 8]. The external auditor can change his password. To audit the database of a client (internal auditor), the external auditor needs to send the internal auditor an audit request by entering the internal auditor`s unique ID. He gets this ID sent by the internal auditor (via email for example). After that, the process ends. The external auditor can only wait now until the internal auditor accepts his request. Only after the internal auditor has accepted his request is when he can start the database audit. The external auditor can load all his audit requests, and the ones who accepted his request can be opened for more details. After selecting the specific client, the external auditor can start the database audit. He can decide if he wants to download the balanced scorecard and/or the proof collection or not, and after that, the process ends. The external auditor can load all the previous audits he has generated for each specific client. In the case he gets an error during the database audit, he can send the internal auditor an issue request. After that, the process ends, and he waits until the issue is fixed. After that, he can start a new database audit.

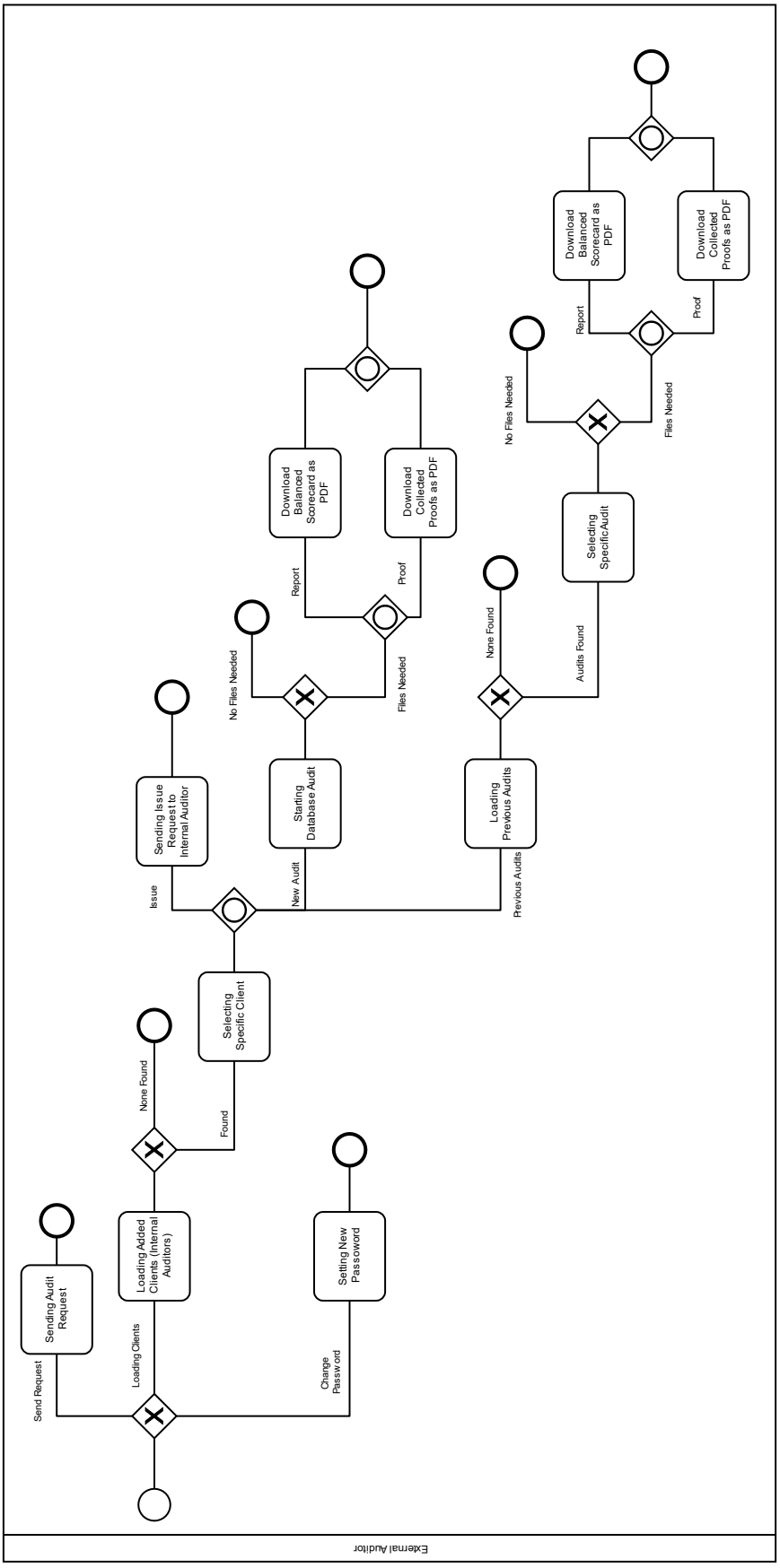

Figure 6: BPMN Diagram – External Auditor Actions

What are the software requirements & what will the software architecture look like?

- As defined in the previous chapter, a web application needed to be implemented. The shown deployment diagram [Figure 2], is giving a good overview of the software architecture that has been implemented. To test the model, a test setup has been created, containing a Windows 10 virtual machine in which a test relational database has been created and filled with dummy data to test the model. The data has been generated by an online data generator [13]. A class diagram will be shown in the next chapter where a good overview of the test database will be shown, but also the database of the application. The prototype consists of a frontend, using React and TypeScript, and a backend written also in TypeScript, using TypeORM as an ORM to connect to the database. To make further work possible with the project, and to make it possible to connect to other databases, not only MariaDB, but the project is also using native libraries for each database type. Since it is a prototype, only the connection to MariaDB needed to be established.

Which controls needed to be answered?

- In the previous chapter, the controls that need to be implemented have been defined. For each control, a BPMN diagram will be shown, describing the approach of the model, and how the model aims to fulfill the required controls.
- Checking if the database is running on a supported database software version? [7]

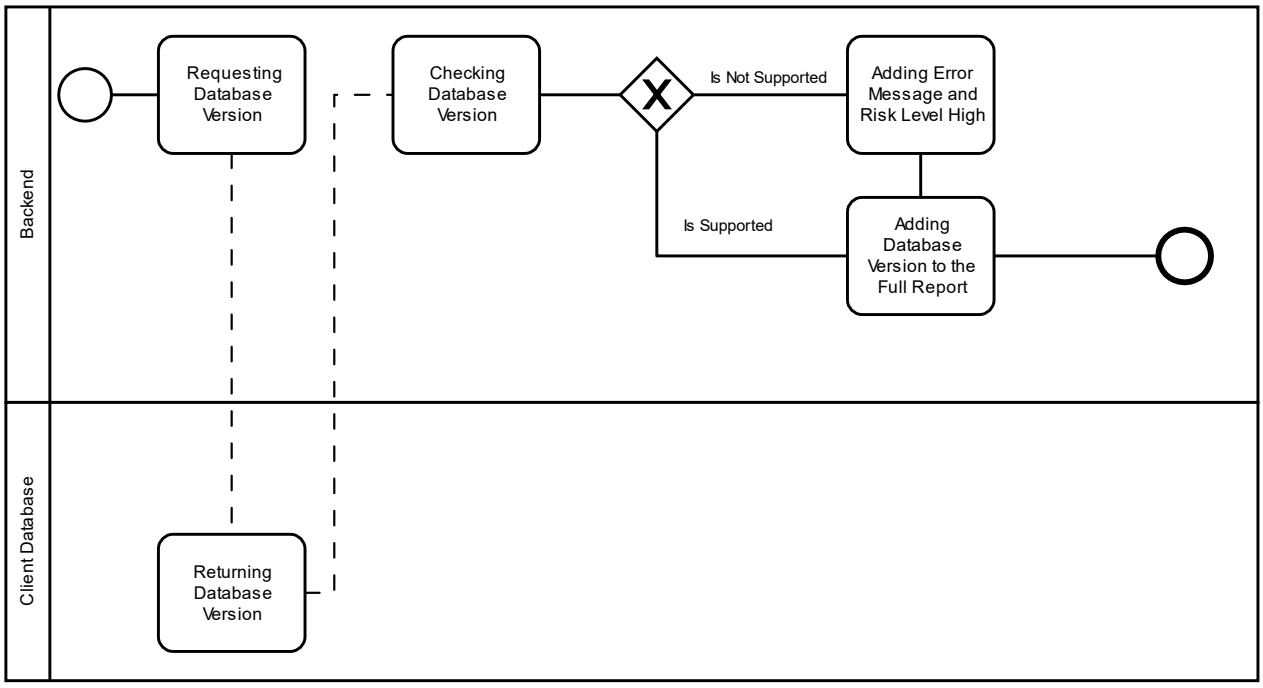

Figure 9: BPMN Diagram – Database Version Control

- The BPMN diagram [Figure 9] is showing the process of controlling the database version. The application is sending a query request to the client's database. The database responds with its version. After the application backend receives the version, it checks if the received version is being supported, if yes, only the database version is being added to the full report (proof), which can be downloaded, if not, an error message is being added with the risk level being set as high, and after that, the database version is also being added to the report.

- COBIT PO2.3 [2]: Checking if user groups have been defined?
  - The BPMN diagram [Figure 10] is showing the process of controlling the user groups (if they have been defined by the client). The application is sending a query request to the client's database. After the application receives the response, it checks if user groups have been found. In the case the database contains no user groups, an error message is added to the full report (proof) and the risk level for this control is being set as high. If the user groups are found, they are being added to the full report (proof).

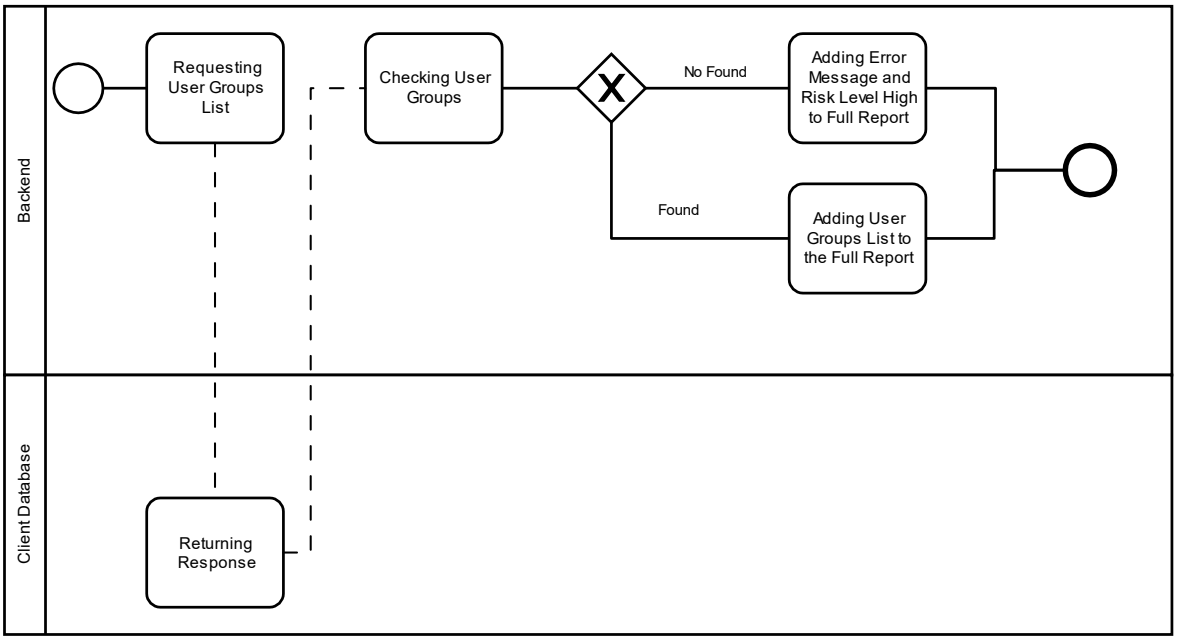

Figure 10: BPMN Diagram – COBIT PO2.3 Control

- Checking if the principle of least privilege is implemented and searching for users with administrator privileges? [7]
    - The following BPMN diagram [Figure 11], is showing both controls (checking if the principle of least privilege is implemented and searching for users with administrator privileges) since they are connected in this model. The application is sending a request to the client's database, where it awaits the list of all users in the database. After the users are loaded, the application is checking each user, if they have an attribute showing to which user group the user has been added and if his title (seniority level) has been added. If not, the ID of the user is being added to the full report (proof), also an error message where the risk level is being set as high. If it is added, the application checks if the user has been added to a user group corresponding to his title. For example, if a user has administrator user rights, but his title is below a specific title, that user will be seen as a risk level – high, and his id and the corresponding message will be added to the full report (proof). If a user has been added to the user group corresponding to the senior title, but his level is below the senior level, he will be seen as a risk level – medium, and also his details will be added to the full report (proof). In

the case a user has been added to a user group corresponding to the title trainee or junior level, he will be seen as low risk.

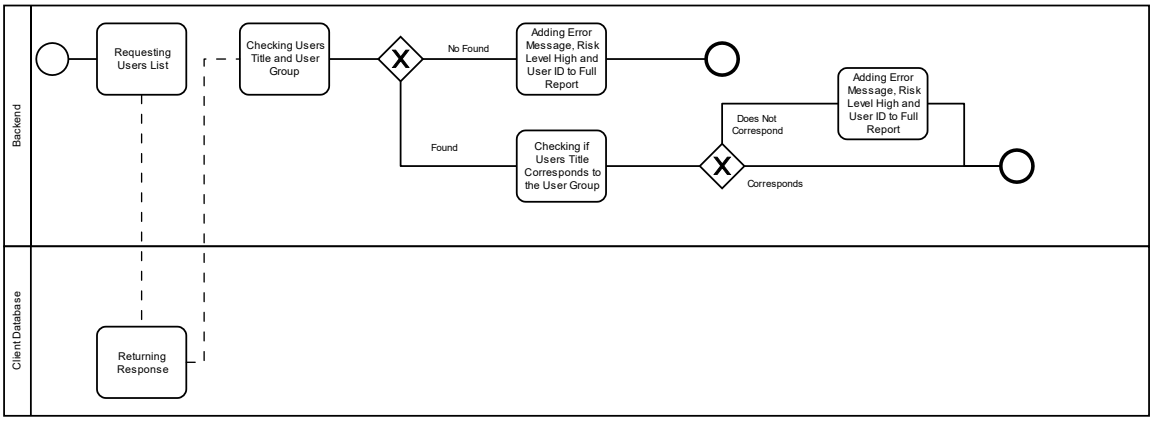

Figure 11: BPMN Diagram – Least Privilege & Administrator User Rights Control

- COBIT DS5 [2] & EU Standard (as described in [12]): Checking the password policy?
    - The BPMN diagram [Figure 12] is showing the process of the password check. The password policy that is being controlled, has been defined in the previous chapter. The process starts with the request of the application to get all users from the client's database. After the users are loaded, the application checks if each user has a password. If some of them do not have a password, those users will be rated with a high risk, and their user IDs, risk level, and corresponding message are being added to the full report (proof). If the user has a password, it will be checked for the minimum requirements. Multiple checks are being done for each password. The password needs to have a minimum length of 8 characters, if it does not contain at least 8 characters it will be rated with a high-risk level. After that, the password is being checked again. If it contains the user's first or last name, it will be rated with a low-risk level. The password also needs to fulfill at least three of the following four requirements. The requirements are that the password needs to have at least one uppercase letter, at least one lowercase letter, at least one number, and at least one special character. If the user does not fulfill at least three of those four, it is being rated with a high-risk level. The password is also being checked for national special characters, if it contains them, then it is being rated with a medium risk level.

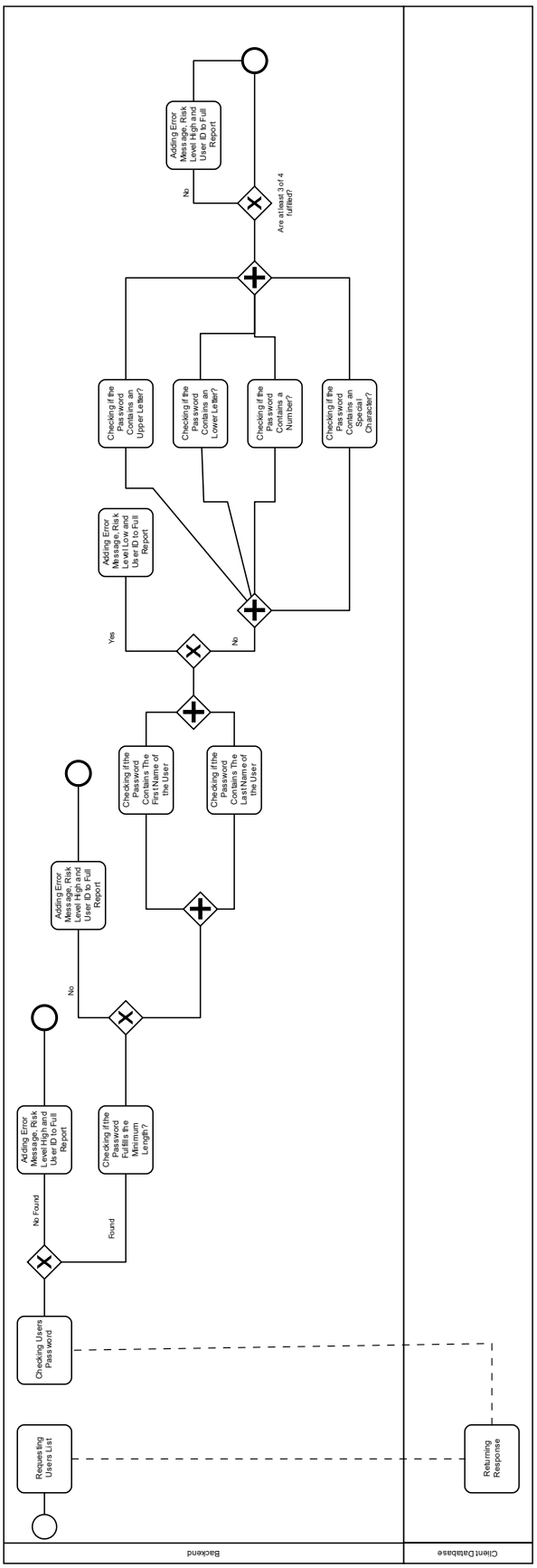

Figure 12: BPMN Diagram – COBIT DS5

- COBIT AC4 [2]: Checking if unexpected interruptions in data processing are being handled? COBIT DS11.5 [2]: Checking if the procedures for backup and restoration of the database have been implemented, followed, and documented? COBIT AI6.1, AI6.2 [2]: Checking if changes are following a defined process and if they are being documented?
    - The controls related to the interruptions, backup, restoration, and changes are represented in the same BPMN diagram [Figure 13] since they are all controlled during the same process. The application is sending a request to the client's ticket system, loading all the tickets for each of the four different types of projects. The application is assuming that all logs have been automatically added to the database. For that reason, after the tickets are loaded, the application is checking if each log entry contains a corresponding ticket. In the case a log entry is not documented in the ticket system, it is rated with a high-risk level. The other tickets, that are corresponding to the log entries are being added to the full report (proof), including their details (assignee, ticket status, comments, issue description) and the log ID. With this approach the chain of custody is being fulfilled since the data is provided, who executed a specific task, when did he do it, and what did he do. It is also important to mention that the tickets are saved in a different database than the companies database that is being audited. By default, the ticket system Jira has its own PostgreSQL database, so the database admin does not have access to it (he does have access, only in the case that he is also being added as the admin for the ticket system).

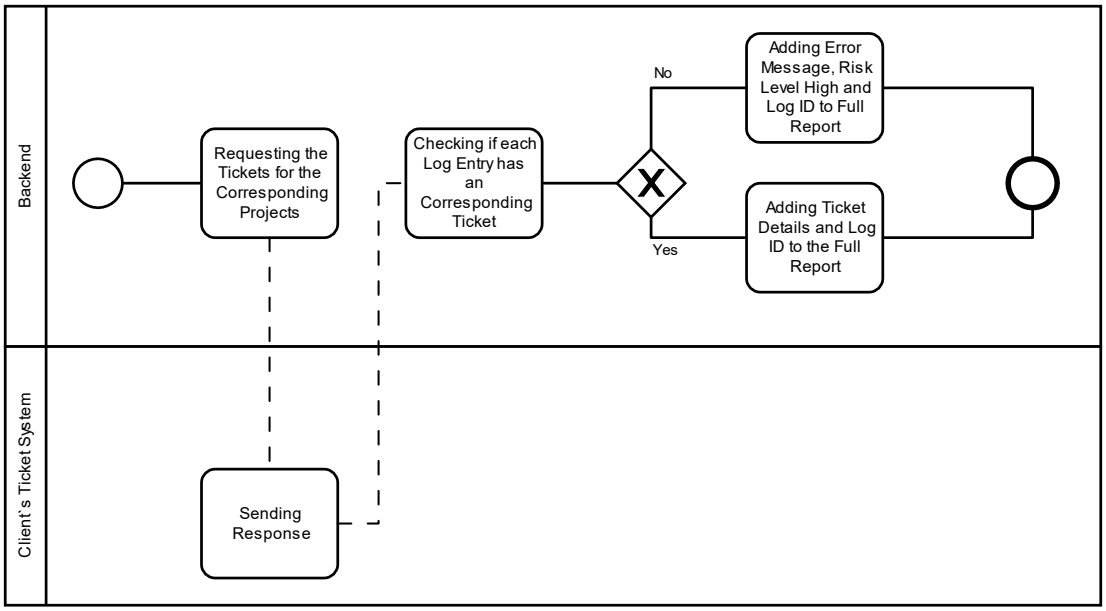

Figure 13: BPMN Diagram – COBIT AC4, DS11.5, AI6.1, AI6.2

Which characteristics defined by COBIT can be answered by this model?

- As described in the previous chapter the model is fulfilling four of the 7 by COBIT defined characteristics. These characteristics are availability, compliance, reliability, and confidentiality. Availability is being proved by all seven controls. Compliance is being proved during the password check (COBIT DS5), and during the control of the interruptions, backups, restorations, and changes (COBIT AC4, DS11.5, AI6.1, and AI6.2). The reliability characteristic is being proved during the database version control, and during the control of the interruptions, backups, restorations, and changes (COBIT AC4, DS11.5, AI6.1, and AI6.2). The confidentiality isn´t being proved only during the database version control; the other six controls are proving this characteristic.

How can the chain of custody be established?

- As previously mentioned, the chain of custody will be checked during the last control of the interruptions, backups, restorations, and changes (COBIT AC4, DS11.5, AI6.1, and AI6.2). The ticket details are containing the needed data and are being saved in a different database than the one being audited.

How can the potential risks be presented?

- During the controls, the risk levels are being tracked, but also the corresponding proof for each control. After the audit has been completed, the results of the analysis are presented to the auditor in the form of a modified balanced scorecard, where the rows are representing each of the controls, and the columns are representing the four characteristics defined by COBIT. The risks are graded from “OK” – no risk detected, to “HIGH” risk. The risks are also being shown in different colors, for example, “OK” is represented in green, and “HIGH” is represented in red – which should mean alert. Besides those two, the other risk levels are “LOW” and “MID” – meaning medium risk level.

What should the report contain?

- The auditor can download the balanced scorecard as a short version of the report, in the case that he only needs to present the fields where the risks have been found, but the auditor can also download a full report (proof collection), where all the proofs that have been collected during each of the controls are being presented. Containing for example all the risk levels, and to which user or control they are related, and all of the ticket details from the ticket system.

6. Implementation Approach and Prototype

In the previous two chapters, the requirements have been defined and the proposed model has been described. This chapter gives a detailed overview of the prototype implementation (how the testing environment has been set up, how the controls are implemented, and how the test database is being audited).

To make further work on the thesis a more accessible development environment has been set up in a Windows 10 virtual machine, and the source code and the test data that has been imported in the test database have been uploaded to a git repository. For the development, the editor Visual Studio Code has been used. The architecture of the prototype has been previously shown in [Figure 2], but now a more detailed overview will be given.

First, the test database and the test ticket system have been installed and filled with test data. The database that has been used is the relational database MariaDB (version 10.5.5). The ticket system that is being used is the Atlassian Jira Software (version 8.5.8). The class diagram [Figure 14], is giving an overview of the classes and their attributes for the test database, but also the database that the prototype is using. The test database, “Client”, is representing the enterprise's database that is being audited. The classes which are needed for the audit are the user groups, users, and logs. The user groups need to have the user group id. The other attributes are used for a better description such as CRUD (create, read, update, and delete) rights and group name. The user class contains multiple needed attributes, those are the user ID, the passwords, the user group ID – the user group to which he has been added, and the title, which represents the seniority level in the enterprise. The other fields like username and email are also used for a better description and more realistic testing. The third class which is needed is the logs. The logs contain multiple attributes which are needed for the database audit, such as log id, project key (showing the project key of the ticket system), the ticket key (the ticket from the ticket system where the log has been documented), and the type as a descriptive attribute, showing if the log entry is an error, change, backup or restore. The attribute names are being added to the config file by the internal auditor and can have different names. This has been done, under the assumption that not all enterprises would name these classes with

the same name. The logs are also being used under the assumption that the enterprise has implemented its triggers which are logging the events.

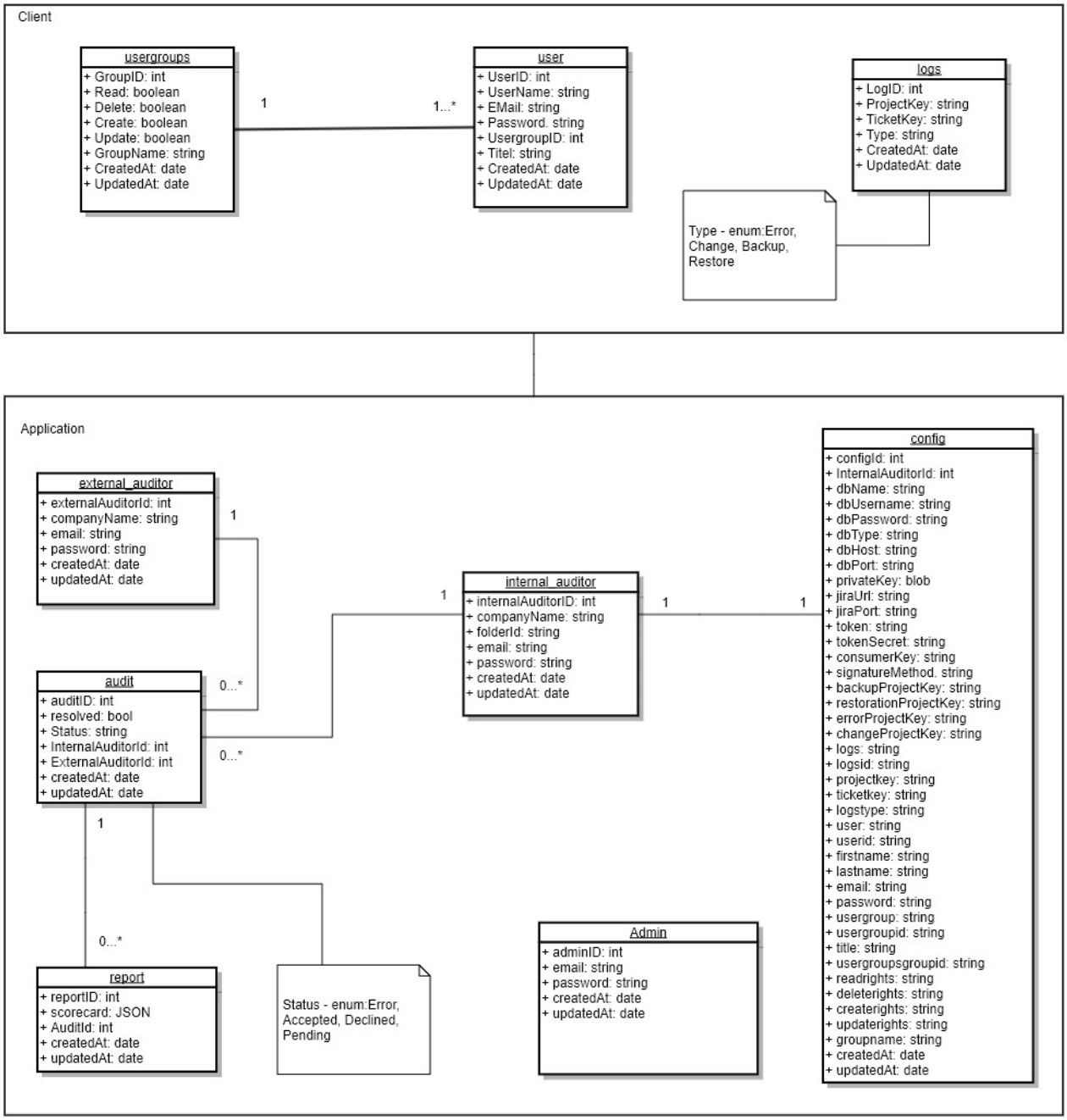

Figure 14: Class Diagram – Classes of the Application and Test Database

The test ticket system has been filled with test tickets for each of the log types, and with multiple test users, for example, the executive who is approving changes, the senior developer who is handling the error messages, and a database user, who is being used as a default user for generating comments, etc. The screenshot [Figure

15] is showing a test ticket from Atlassian Jira, showing a change request, where the senior developer requests to rename a class, but his change is being denied by its executive. The ticket is showing the creation date of the ticket, who is the person handling the issue (the assignee), the status of the ticket, the description of the issue, and the communication which is being shown in the comments.

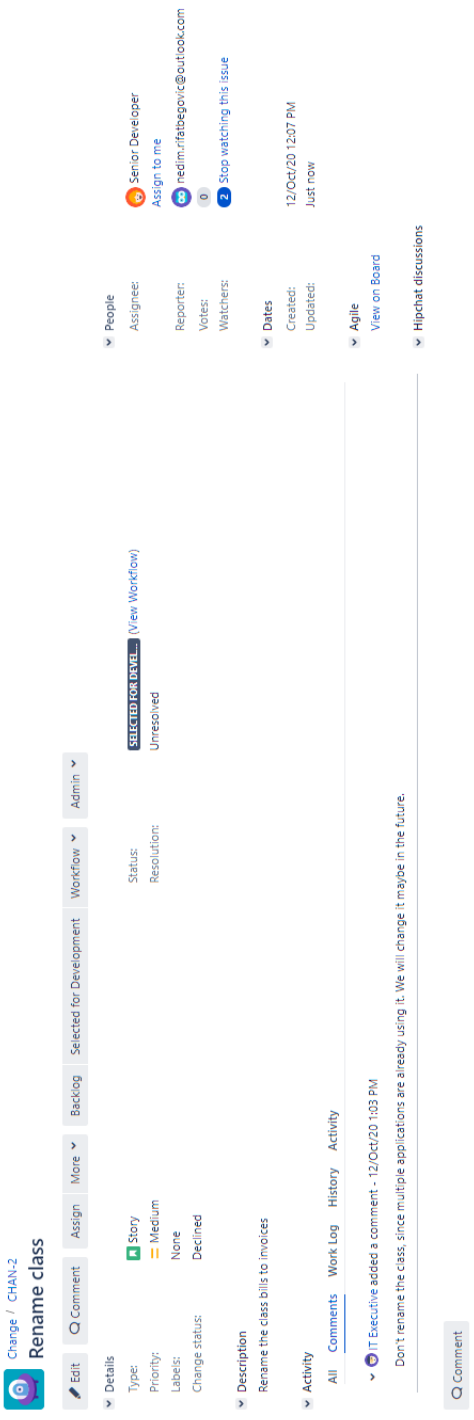

Figure 15: Test Ticket from Atlassian Jira

The previously shown class diagram [Figure 14], is also showing the classes and attributes from the prototype (under the label “Application”). The prototype is also using a MariaDB database and contains six different classes. The admin class represents the application administrator who is the moderator of the application and has the functionality to add and remove external and internal auditors, and to update their attributes when needed. For example, to change the company name, an internal auditor needs to request this change by the admin, because of possible conflicts, if the internal auditor changes the company, he needs to get a new account with a new email, so that he can get a new unique id (which the external auditors need, to send him an audit request). The admin contains the attribute admin ID, email, and password with which he can log in to the application.

The internal auditor class is representing the internal auditor of an enterprise, and he is connected to the config class, representing the configuration file that he needs to fill out at the beginning, and to the audits, which are generated by him, or by the external auditor – after he approved their request. The internal auditor has one config file, but he can have multiple different audits. He has an internal auditor id, company name – which is being shown to the external auditors and in reports, folder id – which is representing the unique id – which the external auditor needs, to send him an audit request, and his credentials (email and password) which he needs to login into the application.

The config class represents the configuration file filled out by the internal auditor. The config file has multiple attributes, but they must be entered only once and will be used for all future audits. The config file is identified by his config id, and the foreign key representing the internal auditor's id. To audit the enterprise's database, the following attributes need to be filled out:

- dbName – the name of the database
- dbUsername – username created for the application (needs enough rights to see the tables and query the fields)
- dbPassword – password for the created user
- dbType – for the prototype, the default value is MariaDB, but this field has been added to enable future development and can be used to select different databases

- dbHost – host (link) needed to access the database
- privateKey - private key needed for the authentication to the ticket system (the private key is being uploaded as a .pem file, and needs to be generated by the enterprise)
- jiraUrl – is the link needed to access the enterprise's ticket system
- jiraPort – is the port number needed to access the ticket system
- token – the token needs to be generated by the ticket system admin so that the application can access the tickets via oAuth (for this and for the following ticket system authentication an OAuth dance needs to be done so that the access is given for a specific user. After the process is done, the needed credentials are provided)
- tokenSecret – is the token secret needed to access the ticket system
- consumerKey – is the consumer key needed to access the ticket system
- signatureMethod – the default signature method for the Jira OAuth is RSA, but this field has been added in the case if other ticket systems are using different signature methods
- backupProjectKey – is the project key where the enterprise is documenting their backup logs
- restorationProjectKey – is the project key where the enterprise is documenting their restoration logs
- errorProjectKey – is the project key where the enterprise is documenting their error logs
- changeProjectKey - is the project key where the enterprise is documenting their change logs
- logs – defines the class name for logs in the enterprise's database
- logsid - defines the attribute name for the log id
- projectkey – defined the attribute name for the project key
- ticketkey – defines the attribute name for the ticket key
- logstype – defines the attribute name for the logs type
- user – defines the class name for users in the enterprise's database
- userid – defines the attribute name for the user-id
- firstname – defines the attribute name for the user`s first name
- lastname – defines the attribute name for the user`s last name

- email – defines the attribute name for the user`s email
- password – defines the attribute name for the user`s password
- usergroupid – defines the attribute name for the user group to which the user has been added
- title – defines the attribute name of the user`s title
- usergroup – defines the class name for user groups in the enterprise's database
- usergroupsgroupid – defines the attribute name for the user groups group id
- readrights – defines the attribute name for the read rights
- deleterights – defines the attribute name for the delete rights
- updaterights – defines the attribute name for the update rights
- createrights – defines the attribute name for the create rights
- groupname – defines the attribute name for the group name

The audit represents the audits that are being generated by the internal and external auditors. Each audit is being related to an internal auditor, but only external audits are being related to internal and external auditors. An audit can also have multiple reports since internal reports are always added to the same audit, while external audits are created for each external auditor. The audit class can be identified by its id and the foreign keys for the internal auditors and external auditor's id. The audit has the attribute status – representing if it is pending, accepted, or declined (internal audits are by default accepted), and the resolved attribute, which is used when an external auditor sends an issue request (that the audit process failed). After the internal auditor has fixed the issue, he sets this attribute to true, meaning that the issue has been resolved.

The external auditor is identified by his external auditor id. He also contains the attributes company name and his credentials (email and password) which he is using to log in to the application. He is only connected to the audit. After the external auditor has entered the internal auditor`s unique id, an audit is being created, connecting them both (in the case that it is a valid id).

The report class represents the reports that are being generated during a database audit and which are being identified by the report id and the foreign key audit id. The generated balanced scorecard and the collected proof are being saved in the attribute scorecard formatted as a JSON. That attribute contains two main branches:

- “scorecardTable” – containing the formatted balanced scorecards, and the audit results
- “balancedScorecards”, containing the proof which has been collected during the database audit.

The details of which data is being collected will be presented later in this chapter.

To generate the tokens needed for the authentication to Jira, Atlassian has provided online documentation [14], where the steps have been described. So, to generate the tokens, a Jira user needs to be created, which will be used by the application to access the projects and tickets. For the so-called OAuth dance, the private key and the Jira URL are needed. The code below is showing the needed parameters.

```
1.  const keyfile = pathtoPrivateKey;
2.
3.  let privateKeyData = fs.readFileSync(keyfile, "utf-8");
4.  var REQUEST_TOKEN_URL = jiraurl + "/plugins/servlet/oauth/request-token";
5.  var ACCESS_TOKEN_URL = jiraurl + "/plugins/servlet/oauth/access-token";
6.  var AUTHORIZE_TOKEN_URL =
7.    jiraurl + "/plugins/servlet/oauth/authorize?oauth_token=";
8.  var OAUTH_VERSION = "1.0";
9.  var HASH_VERSION = "RSA-SHA1";
```

After the request has been sent to the ticket system, the administrator needs to approve it, and after that, the needed tokens are returned. The OAuth dance code is provided in the previously mentioned Atlassian Jira documentation and can be found in different programming languages, for example, JavaScript, Java, Python, etc.

Since the prototype consists of two parts, a backend, and a frontend. Firstly, a short overview of the backend will be given. After that, the implementation of the controls will be shown. The controls will be followed by the description of the frontend implementation, showing the user stories for each user type and the implemented user interface.

The code is written using TypeScript (version 4.2.3), NodeJS (version 14.15.1), npm (version 6.14.8) and React (version 17.0.1) for the frontend. The backend is a TypeScript application, using the NodeJS library “express” [15] – a web application framework, to make HTTP calls possible. To enable the communication between the backend and frontend, multiple APIs have been implemented. An overview of each

call will be given (which inputs are expected, and what output is being generated). The backend is currently running only on localhost for test purposes, under port 5000.

The “/login” API is expecting a JSON as input, with the attributes email, password, and type – showing which type of user is requesting to login (admin, internal or external). The response is containing a Boolean value, showing if the has entered the correct credentials.

The “/loadinternal” API is called by the admin user and is expecting the internal auditor`s email and type to load his data from the database. The response is the user details of the specific internal user.

The “/loadexternal” API is called by the admin user and is expecting the external auditor`s email and type to load his data from the database. The response is the user details of the specific external user.

The “/updateinternal” API is called by the admin user and is expecting the internal auditor`s email (which attribute needs to be updated), and the attribute name and new value for the attribute.

The “/updatexternal” API is called by the admin user and is expecting the external auditor`s email (which attribute needs to be updated), and also the attribute name and new value for the attribute.

The “/internal” API is called by the admin and is expecting the internal auditor`s email, password, and company name as input. The backend is checking if the user already exists, if not, the internal auditor is created, and the API responds with the internal auditor`s unique ID.

The “/external” API is called by the admin and is expecting the external auditor`s email, password, and company name as input. The backend is checking if the user already exists, if not, the external auditor is created.

The “/deleteuser” API is called by the admin and is expecting the type of user that needs to be removed from the database and his email. The backend is searching for the user, and if found, removes him from the database. The API is responding with a Boolean value if the user has been removed or not.

The “/internalpassword” API is called by the internal auditor and is expecting his email and the new password. After the request has been processed, the API responds with a Boolean value showing if the request has been successful.

The “/newconfig” API is called by the internal auditor and is expecting a JSON containing the config attributes as defined previously in this chapter. The private key is being expected in a separate field since it is being sent as a base64 encrypted string and is being saved into the database as a Blob (file). After the request has been processed, the API responds with a Boolean value showing if the request has been successful.

The “/checkconfig” API is called each time an internal auditor opens the configuration page. The auditor`s email is requested, to check if the auditor has already added a configuration file. Depending on the response, the internal auditor will get a different page to see.

The “/updateconfig” API is called by the internal auditor. The expected input attributes are the internal auditor`s email, which attributes need to be updated, and the new value of the attribute. A Boolean value is sent back as a response, showing if the attribute update was a success.

The “/getexternalaudits” API is called by the internal auditor and is expecting as input the internal auditor’s email. The backend is searching then for all external audit requests. If non is found a message is returned that none have been found, or if requests have been found a JSON array is returned, where each element is having the attributes audit id, the email of the external auditor, and the current status of the request (pending, declined, or accepted).

The “/changeauditstatus” API is called by the internal auditor and is expecting the selected audit id and the new audit status (accepted or declined) which needs to be set. This is used only for external audits, which the internal auditor needs to accept or decline. The response which is given is the default response that is generated by TypeORM if the request has been successful or not.

The “/internalauditrequest” API is called by the internal auditor and is expecting as input the email of the internal auditor. The backend is checking if an internal audit has been created, if yes, then a new audit report is being generated and added to the

found audit. If not, an audit is added to the internal auditor, and after that, a new report is being generated. The details on database audit and report generation will be given later in this chapter.

The “/unresolvedrequests” API is called by the internal auditor, and his email is the expected input. The backend is searching for audits of the internal auditor, where there has been an issue request sent by the external auditor (where the resolved attribute has been set to false). The API is responding with a JSON array consisting of the audit`s id where an issue request has been detected and the external auditor´s email and id.

The “/allreports” API is called by the internal auditor and is expecting the auditor’s email as input. The backend is searching for all audit reports which have been generated for the specific internal auditor`s database. The response of this API is a JSON array containing the elements report (which is the JSON containing the balanced scorecard table and the collected proof), the date when the report has been generated, its id, and the company name for which the report has been generated.

The “/externalpassword” API is called by the external auditor and is expecting his email and the new password. After the request has been processed, the API responds with a Boolean value showing if the request has been successful.

The “/adduniqueiduser” API is called by the external auditor and represents his audit request to a specific internal auditor. The API is expecting the external auditor`s email and the internal auditor`s unique id, to create a new audit connecting them both. The response is a message telling the external auditor if the request has been successful or not.

The “/getallclientsexternaluser” API is called by the external auditor and is expecting his email as input. The backend is then loading all of his added clients (internal auditor`s), and sending the list formatted as a JSON array.

The “/externalauditrequest” API is called by the external auditor after his audit request has been approved by the internal auditor. This API is expecting the audit id and the external auditor`s email as input.

The “/allexternalreports” API is called by the external auditor and is expecting the audit it. The backend is then loading all reports generated by this external auditor for a specific audit and returning them formatted as a JSON array.

The “/updateresolvedaudit” API is called by the external auditor and is expecting the audit id and action type. This API is used after an audit has failed and is being sent to the internal auditor as an issue request so that the internal auditor gets notified that something is wrong with his configuration.

Besides the described app file, the backend consists of three different parts:

- the entities, which are the entities defined with TypeORM (defining the classes and their attributes)
- the queries, which are implemented functions used for querying the application's database, and for generating the report
- the client folder, where the queries are saved, which are needed for accessing the enterprise's database and ticket system.

The entities are classes that are mapped to the corresponding database tables, as described in [16]. The entities consist of columns and relations, and an entity always consists of a primary column representing the primary key (for example the id).

```
1.   import { BaseEntity, Column, Entity, PrimaryGeneratedColumn } from "typeorm";
2.  import { Field, ID, ObjectType } from "type-graphql";
3.
4.  @ObjectType()
5.  @Entity()
6.  export class Admin extends BaseEntity {
7.    @Field(() => ID)
8.    @PrimaryGeneratedColumn()
9.    adminId: number;
10.
11.   @Column({ type: "timestamp" })
12.   createdAt: Date;
13.
14.   @Column({ type: "timestamp" })
15.   updatedAt: Date;
16.
17.   @Column({ type: "text", unique: true })
18.   email: string;
19.
20.   @Column({ length: 250 })
21.   password: string;
22. }
```

This is a simple example of an entity that has been created. The Class Admin is defined like previously shown in the class diagram, with the “adminId” as the primary key, and the defined attributes like email and password. The attributes “createdAt” and “updatedAt” are default attributes, needed to track when the database entry has been created or updated. As shown in the code, the email is in this case a unique attribute, since there can´t be two users with the same email. In the code example shown below, the entity for the internal auditor is being shown. The interesting thing, in this case, are the many to many, and the many to one method. They are creating relationship tables where the connection between different classes is defined. While implementing controls, these methods have been important, since they enable an easier search for related classes (for example when wanting to find the audits of a specific internal auditor, etc.).

```
1.  import {
2.    BaseEntity,
3.    Column,
4.    Entity,
5.    JoinTable,
6.    ManyToMany,
7.    OneToMany,
8.    PrimaryGeneratedColumn,
9.  } from "typeorm";
10. import { Field, ID, ObjectType } from "type-graphql";
11.
12. import { Audit } from "./Audit";
13. import { Config } from "./Config";
14.
15. @ObjectType()
16. @Entity()
17. export class InternalAuditor extends BaseEntity {
18.   @Field(() => ID)
19.   @PrimaryGeneratedColumn()
20.   internalAuditorId: number;
21.
22.   @Column({ type: "timestamp" })
23.   createdAt: Date;
24.
25.   @Column({ type: "timestamp" })
26.   updatedAt: Date;
27.
28.   @Column({ type: "text", unique: true })
29.   email: string;
30.
31.   @Column({ length: 250 })
32.   password: string;
33.
34.   @Column({ type: "text" })
35.   companyName: string;
36.
37.   @Column({ type: "text", unique: true })
```

```
38.   folderId: string;
39.
40.   @OneToMany(() => Config, (config) => config.internalAuditor)
41.   configs: Config[];
42.
43.   @ManyToMany((type) => Audit, (audit) => audit.internalAuditors)
44.   audits: Audit[];
45. }
```

The queries part of the backend is consisting of simple queries needed to get, update, or delete data from the application`s database, but also the query part contains the functions implemented for the defined COBIT controls. An example of a simple query using TypeORM is shown below.

The code example is showing the removal of an internal auditor from the database. As input value, the internal auditor`s email is provided, and if a user with the specific email is found, he is being removed from the database. Depending on the result, a Boolean value is returned.

```
1.  const removeInternal = async (email: string) => {
2.    const connection = getConnection();
3.
4.    console.log("__START__");
5.    try {
6.      await connection
7.        .createQueryBuilder()
8.        .delete()
9.        .from(InternalAuditor)
10.       .where("email = :email", { email: email })
11.       .execute();
12.     console.log("__END__");
13.     return true;
14.   } catch (error) {
15.     console.log("__FAIL__");
16.     return false;
17.   }
18. };
```

The details for each implemented COBIT control will be shown later in this chapter.

The third part of the backend is the client queries. To make it easier to query different types of databases and to make the application more flexible, the native npm libraries are being used for the enterprise's database instead of TypeORM. In this prototype only the queries for the relational database MariaDB have been implemented, using the mariadb library [17]. After loading the needed input data from the configuration added by the internal auditor, the enterprise's database can be accessed. A simple

example is shown below. The code example is showing a simple query, loading all the logs from the enterprise's database.

```
1.  // * Get Logs
2.  export async function getLogs(data: IDBConnection, logs: string) {
3.    let conn: any;
4.    try {
5.      conn = await getDBConnection(data);
6.      const rows: any[] | undefined = await conn.query("SELECT * FROM " + logs);
7.
8.      if (rows !== undefined || rows !== null) {
9.        conn.end();
10.       return rows;
11.     } else {
12.       conn.end();
13.       return undefined;
14.     }
15.   } catch (err) {
16.     conn.end();
17.     throw err;
18.   }
19. }
```

After giving the overview of the backend structure, the implemented controls are shown. Each of the defined controls from the model chapter and the requirements analysis chapter will be shown, with a detailed description of the implementation and important code parts. Now only the backend part will be shown, with details of the data fetching and of the database analysis, the user interface, and the results that are shown to the end-user (the internal and external auditor) will be shown later in this chapter when describing and showing the defined user stories.

Like previously described, the database audits are triggered with an API call by the internal or by the accepted external auditor. For example, if an internal auditor calls the API "/internalauditrequest", the database audit is started. The “internalReport” function is called, from the queries part of the backend. First, the internal auditor`s email is checked, and then whether there´s an internal audit already created.

```
1.  // * Get internal auditors ids (added to the audit)
2.      const internalAuditorPreload = await auditRepository
3.        .createQueryBuilder("audit")
4.        .leftJoinAndSelect(
5.          "audit.internalAuditors",
6.          "internal_auditor",
7.          "internal_auditor.internalAuditorId = :internalAuditorId",
8.          { internalAuditorId: auditorid }
9.        )
10.       .getMany();
11.
```

```
    // * Get audits without external auditors (find internal audit - if any)
    for (let i: number = 0; i < internalAuditorPreload.length; i++) {
      if (internalAuditorPreload[i].externalAuditors === undefined) {
        // * If yes return audit id
        return internalAuditorPreload[i].auditId;
      }

      if (internalAuditorPreload[i].externalAuditors.length === 0) {
        // * If yes return audit id
        return internalAuditorPreload[i].auditId;
      }
    }

    return auditorid;
```

This code part is showing the searching for the specific internal auditor. A loop is triggered to find if there are any of his audits, that have no external auditor. If there is an audit without an external auditor, that means that this is the internal audit, and the newly requested database audit report can be added to it. In the case that there is no such audit, that means that this is the first internal audit, and a new audit will be created. After the internal audit has been created (or found), the controls are triggered with the function called “generateReport”.

Before starting the audit, the needed config data is loaded from the application`s database (the enterprise`s database host, port, user, password, and database type). The last attribute is important in the case of future work on the prototype so that it can be defined which type of database is being audited (MariaDB or something else).

The first control that is being called is the check if the database is running under a supported software version and is being called with the “getDBVersion” function.

```
  conn = await getDBConnection(data);
    const rows = await conn.query("SELECT VERSION()");
    let replay: IDBVersion = {
      version: "",
      status: false,
    };
    // * List of supported versions: https://mariadb.com/kb/en/mariadb-server/ (10.2 until 2022)
    if (rows[0]["VERSION()"] !== undefined || rows[0]["VERSION()"] !== null) {
      const dbfullversion: string = rows[0]["VERSION()"];
      const dbVersionStr: string = dbfullversion[3];
      var dbVersionNumber: number = +dbVersionStr;

      // ! 1 => Version is up to date
      if (dbVersionNumber > mdbconfiguration.latestSupportedVersion) {
        conn.end();
        replay = {
          version: dbfullversion,
          status: true,
```

```
        };
        return replay;
      } else {
        // ! 0 => Version is not up to date
        conn.end();
        replay = {
          version: dbfullversion,
          status: false,
        };
        return replay;
      }
    } else {
      conn.end();
      return undefined;
    }
```

As shown in the code, the connection to the enterprise`s database is established, and the database software version is loaded with a simple SQL query. Since, there is no API enabling to load the list of the supported versions for MariaDB, to the best of my knowledge, a configuration file has been created, containing the last supported version. The currently last supported version will be supported until 10.02.2022 as defined in the official MariaDB documentation [18]. In the case of an older version of the database software being found, a false value is returned, and the risk level for this control is set to high. If the found version is supported, the status that is being returned is true, and the status for this control in the balanced scorecard will be set to “OK” since no risk has been detected. In both cases, the type of the database and the found version are returned and later added to the full report (proof).

The second control that has been implemented is the COBIT PO2.3 control, where the application is searching for user groups in the enterprise`s database. This control is called after the database version check has been completed, with the “getUserGroups” function.

```
  data: IDBConnection,
  usergroups: string,
  groupname: string
) {
  let conn: any;
  try {
    conn = await getDBConnection(data);
    const rows: any = await conn.query("SELECT * FROM " + usergroups);

    if (rows === undefined) {
      conn.end();
      return undefined;
    } else {
      conn.end();
      let groupnames: IUserGroupsNames[] = [];
```

```
16.       for (let i: number = 0; i < rows.length; i++) {
17.         let grname = {
18.           GroupName: rows[i][groupname],
19.         };
20.         groupnames.push(grname);
21.       }
22.       return groupnames;
23.     }
24.   } catch (err) {
25.     conn.end();
26.     // throw err;
27.     return undefined;
28.   }
```

A SQL query is done, searching for all defined user groups in the enterprise`s database. If no user groups are found, undefined is returned, and this control`s risk level is set to high. If yes, the user groups are returned as an array for further analysis, since they are needed for the next control.

The next control is checking if the principle of least privilege is implemented, and search for users with administrator user rights. To controls this, the loaded user group is needed, and the users' list. Before this control can begin, the function “getUsers” is fetched, to load all users from the enterprise`s database. This control starts after that with calling the “checkUserGroupsStatus” function. As input variables, this function expects the arrays of the user groups and users, and the needed data defined in the config. The code shown below is part of the function where the control is done. The function is checking the title (seniority level) of the selected user and comparing it to its current user group. For example, if a user is a junior developer, but has admin rights, this user will be marked as a high-risk level. This is done for each type of user group; the only difference is that the risk level is different for different types of users. It is also important to mention that for this control multiple possible types have been added. In the case, that an enterprise is using different names or different languages, it needs to be expanded in the code. The function is returning an array, where each element contains the type of the error, its risk level, and the user id of the user where the risk has been detected. This has been done with the IDs because of the EU GDPR law so that there are no privacy issues since the report can be also generated by external auditors.

```
1.   // * As defined in the example
2.         // * 1 === Executive / Admin
3.         // * 2 === Senior
4.         // * 3 === Junior / Trainee
```

```
5.          if (users[i][usersUserGroupID] === 1) {
6.            if (
7.              users[i][usersTitle] === "Executive" ||
8.              users[i][usersTitle] === "executive" ||
9.              users[i][usersTitle] === "admin" ||
10.             users[i][usersTitle] === "Admin"
11.           ) {
12.             // Do nothing, it is ok!
13.           } else {
14.             let newError: IError = {
15.               type: ErrorWrongUserGroup,
16.               description:
17.                 "User has wrong user group. Current title: " +
18.                 users[i][usersTitle] +
19.                 " but has been added as admin!",
20.               level: errorLevel.HIGH,
21.               userid: users[i][usersUserID],
22.             };
23.             errors.push(newError);
24.           }
25.         } else if (users[i][usersUserGroupID] === 2) {
26.           if (
27.             users[i][usersTitle] === "Senior Developer" ||
28.             users[i][usersTitle] === "Senior" ||
29.             users[i][usersTitle] === "Senior Associate" ||
30.             users[i][usersTitle] === "senior" ||
31.             users[i][usersTitle] === "senior developer" ||
32.             users[i][usersTitle] === "senior associate"
33.           ) {
34.             // Do nothing, it is ok!
35.           } else {
36.             let newError: IError = {
37.               type: ErrorWrongUserGroup,
38.               description:
39.                 "User has wrong user group. Current title: " +
40.                 users[i][usersTitle] +
41.                 " but has been added as senior!",
42.               level: errorLevel.MID,
43.               userid: users[i][usersUserID],
44.             };
45.             errors.push(newError);
46.           }
47.         } else if (users[i][usersUserGroupID] === 3) {
48.           if (
49.             users[i][usersTitle] === "Junior Developer" ||
50.             users[i][usersTitle] === "Junior" ||
51.             users[i][usersTitle] === "Junior Associate" ||
52.             users[i][usersTitle] === "junior" ||
53.             users[i][usersTitle] === "junior developer" ||
54.             users[i][usersTitle] === "junior associate" ||
55.             users[i][usersTitle] === "Trainee" ||
56.             users[i][usersTitle] === "trainee"
57.           ) {
58.             // Do nothing, it is ok!
59.           } else {
60.             let newError: IError = {
61.               type: ErrorWrongUserGroup,
62.               description:
63.                 "User has wrong user group. Current title: " +
64.                 users[i][usersTitle] +
```

```
65.                 " but has been added as junior / trainee!",
66.               level: errorLevel.LOW,
67.               userid: users[i][usersUserID],
68.             };
69.             errors.push(newError);
70.           }
71.         }
72.       } else {
73.         let newError: IError = {
74.           type: ErrorUserGroupNotFound,
75.           level: errorLevel.HIGH,
76.           userid: users[i][usersUserID],
77.         };
78.         errors.push(newError);
79.       }
```

After the database audit has finished, the report is saved in the database via the “addReport” function, where the expected input variables are the audit id and the generated report formatted as a JSON.

The following control that has been implemented is the COBIT DS5 control, where the passwords of the users are checked if they fulfil the required password policy. The control begins after the function “checkPassword” is called. The function requires the user list and the needed config data. The check is done following the previously defined password policy [12]. The passwords need to be between 8 and 16 characters long. It is also not allowed for the password to contain the first or last name of the user. The password is not allowed to contain characters that are only found on specific national keyboards. The password needs to also fulfil at least three of these four requirements:

- At least one standard uppercase character (A – Z)
- At least one standard lowercase character (a – z)
- At least one number (0 – 9)
- At least one symbol among the following: ! % - _ + = [ ] { } : , . ? < > ( ) ;

First, the function checks if the selected user has a password, if not he is classified with the risk level high. After that, the function “passwordPolicyCheck” is called, where the defined password controls are done.

The length is the first characteristic that is being controlled and is done as shown in the code shown below.

```
1.   // * Check minimal password length
2.     if (password.length < 8) {
3.       obj.result = false;
4.       obj.error = "The password is not long enough!";
5.       obj.level = errorLevel.HIGH;
6.       obj.userid = userid;
7.       return obj;
8.     }
9.     if (password.length > 16) {
10.      obj.result = false;
11.      obj.error = "The password is too long!";
12.      obj.level = errorLevel.LOW;
13.      obj.userid = userid;
14.      return obj;
15.    }
```

The password`s length is checked if it is higher than the minimum length (8). If it isn´t, the password is classified with a high-risk level. Also, the user`s id and the error message are added, and the Boolean attribute result, which is showing if the password fulfills the minimum requirements. After the minimum length is checked, the maximum length is controlled. If the password is longer than the maximum length (16), it is rated with low risk.

After the password has passed the length check, the next check is the control if the password contains the user`s first name or last name. The code shown below is the part of the function where this check is done. If the user contains either the first name or the last name, the password is rated with low risk.

```
1.    // * Check if the password contains the firstname
2.     if (firstname !== undefined && firstname !== null) {
3.       if (password.includes(firstname)) {
4.         obj.result = false;
5.         obj.error = "The password contains firstname!";
6.         obj.level = errorLevel.LOW;
7.         obj.userid = userid;
8.         return obj;
9.       }
10.    }
11.
12.  // * Check if the password contains the firstname
13.    if (firstname !== undefined && firstname !== null) {
14.      if (password.includes(firstname)) {
15.        obj.result = false;
16.        obj.error = "The password contains firstname!";
17.        obj.level = errorLevel.LOW;
18.        obj.userid = userid;
19.        return obj;
20.      }
21.    }
```

The control of the national special characters and the four requirements are done in the same loop. At the beginning of the function, the requirements are defined as shown below.

```
1.    // * Defining the counters
2.    var numUpper: number = 0;
3.    var numLower: number = 0;
4.    var numNums: number = 0;
5.    var numSpecials: number = 0;
```

Before the loop, the counters are set to zero for each of the four requirements. The loop is going through each character of the selected password, as shown below, and is checking if there is at least one number, at least one uppercase standard character, at least one special character from the defined list, and at least one lowercase standard character. During the loop, at the same time, the password is being checked for any special national characters. If one national special character is found, the password is rated with the “MID” risk level and added to the reply array. If no national special character has been found, the counters are controlled, and if not, enough requirements are fulfilled and the id is rated with a high-level risk and added to the reply array.

```
1.    // * Defining the counters
2.    var numUpper: number = 0;
3.    var numLower: number = 0;
4.    var numNums: number = 0;
5.    var numSpecials: number = 0;
6.    for (var i = 0; i < password.length; i++) {
7.      if (anUpperCase.test(password[i])) numUpper++;
8.      else if (aLowerCase.test(password[i])) numLower++;
9.      else if (aNumber.test(password[i])) numNums++;
10.     else if (aSpecial.test(password[i])) numSpecials++;
11.     else if (aParticularKeyboard.test(password[i])) {
12.       obj.result = false;
13.       obj.error = "Password contains special national character!";
14.       obj.level = errorLevel.MID;
15.       obj.userid = userid;
16.       return obj;
17.     }
18.   }
19. // * Checking if the password has the right format
20.   let counter: number = 0;
21.   if (numUpper >= 2) {
22.     counter = counter + 1;
23.   }
24.   if (numLower >= 2) {
25.     counter = counter + 1;
26.   }
27.   if (numNums >= 2) {
```

```
    counter = counter + 1;
  }
  if (numSpecials >= 2) {
    counter = counter + 1;
  }

  if (counter > 2) {
    return obj;
  } else {
    obj.result = false;
    if (counter === 1) {
      obj.error =
        "Only " + counter + " requirement has been fulfilled! (Min is 3 / 4)";
    } else {
      obj.error = counter + " requirements have been fulfilled! (Min is 3 / 4)";
    }
    obj.level = errorLevel.HIGH;
    obj.userid = userid;
    return obj;
  }
```

The following four controls (COBIT AC4, DS11.5, AI6.1, and AI 6.2) are done during the same function call since all four types of events (interruption, backup, restoration, and change) are documented in the ticket system. First, the logs are loaded from the enterprise`s database with a simple SQL query. After the logs have arrived, the needed data is loaded from the config and formatted as needed for the authentication to the ticket system. After the data is prepared, the function “checkTSTickets” is called and is expecting the logs and the prepared data as input. The function is going through each log entry and looking if it has been documented in the ticket system. In the case that no corresponding Jira ticket has been found, the log entry is rated with a corresponding risk level.

```
if (tsTicket["fields"] === undefined) {
        let errorlevelmsg: string = errorLevel.LOW;
        if (logs[i][logsProjectkey] === errorProjectKey) {
          errorlevelmsg = errorLevel.HIGH;
          errortypemsg = "INTERUPTION";
        } else if (
          logs[i][logsProjectkey] === restorationProjectKey ||
          logs[i][logsProjectkey] === backupProjectKey
        ) {
          errorlevelmsg = errorLevel.MID;
          errortypemsg = "BACKUP/RESTORATION";
        } else if (logs[i][logsProjectkey] === changeProjectKey) {
          errorlevelmsg = errorLevel.MID;
          errortypemsg = "CHANGES";
        }

        const errormsg: ITicketSystemReply = {
          logid: logs[i][logsId],
          level: errorlevelmsg,
```

```
20.         errordescription: "TICKET NOT DOCUMENTED",
21.         errortype: errortypemsg,
22.       };
23.       response.push(errormsg);
24.     }
```

As shown in the code, in case of interruptions that have not been documented and handled, the log entry is rated with a high-risk level. In the case that no backup/restoration/changes have been found, the selected log entry is rated with a "MID" level risk.

In the case the ticket has been found, the needed data is formatted as a JSON and added to the reply JSON array, containing the log id, the comments found in the ticket, the ticket status, the ticket assignee, and the ticket description. The code below shows a part of the function where the data is collected and formatted as needed.

```
1.    const fields = tsTicket["fields"];
2.       // * Check if ticket has comments
3.       if (fields["comment"].total === 0) {
4.         // * No Comments have been found
5.       } else {
6.         for (let i = 0; i < fields["comment"]["comments"].length; i++) {
7.           let commentInput: ITSComments = {
8.             text: fields["comment"]["comments"][i]["body"],
9.             author: fields["comment"]["comments"][i]["author"]["name"],
10.          };
11.          commentsArray.push(commentInput);
12.        }
13.      }
14.      // * Check status of the ticket
15.      const ticketStatus = fields.status.name;
16.      // * Check the assignee of the ticket
17.      let assignee: undefined | string = undefined;
18.      if (
19.        fields.assignee === null ||
20.        fields.assignee === undefined ||
21.        fields.assignee.name === null
22.      ) {
23.        assignee = undefined;
24.      } else {
25.        assignee = fields.assignee.name;
26.      }
27.
28.      // * Get ticket description
29.      let ticketdescription = tsTicket["fields"]["description"];
30.
31.      const errormsg: ITicketSystemReply = {
32.        logid: logs[i][logsId],
33.        ticketComments: commentsArray,
34.        ticketStatus: ticketStatus,
35.        assignee: assignee,
36.        ticketDescription: ticketdescription,
37.      };
```

```
38.         response.push(errormsg);
```

After the controls have finished, the data is prepared and correctly formatted. The data for the full report (proof) is saved in a JSON object where the data collected during the controls is saved. The code below shows the structure of the collected proof.

```
1.      let balancedScorecards: IBalancedScorecard = {
2.        dbversion: dbVersionJSON,
3.        usergroups: userGroupsJSON,
4.        usergroupscheck: userGroupsReply,
5.        passwords: userspasswordsJSON,
6.        users: usersJSON,
7.        ticketsystem: TSTickets,
8.      };
```

Each of the nodes represents the collection of the results for the specific control. This is the data that will be shown to the user under proof download. After that, the data for the balanced scorecard table is being formatted in the function “prepareTabledata”. The functions are expecting the previously formatted JSON object as input. It is checking then for each control type if any risk levels have been defined, and in the case, they have been found formats the data accordingly. Below an example for the preparation of the database version control is shown, where the function is searching for the database version risk level, if none are found, the risk level is set to “OK”. Since each control is fulfilling multiple characteristics as defined by COBIT, and as previously described, the Boolean values are set to true for the characteristics this control fulfills. For example, in this case, the control can only fulfill the availability and the reliability. As will be shown later in this chapter, the risk level will be shown only for these two characteristics, since the other two can never be fulfilled with this control. The other controls are following the same method of preparing the balanced scorecard table input.

```
1.    if (balancedScorecards.dbversion["level"] !== undefined) {
2.      databaseInput = {
3.        cobit: {
4.          Availability: true,
5.          Compliance: false,
6.          Reliability: true,
7.          Confidentality: false,
8.        },
9.        value: balancedScorecards.dbversion["level"],
10.     };
11.   } else {
12.     databaseInput = {
13.       cobit: {
```

```
        Availability: true,
        Compliance: false,
        Reliability: true,
        Confidentality: false,
      },
      value: "OK",
    };
  }
```

After all of the inputs have been correctly prepared, they are formatted into a JSON, as shown below and then returned.

```
const scorecardTable: IScorecardTable = {
    dbversion: databaseInput,
    userrights: usergroupsInput,
    userrightscheck: usergroupscheckInput,
    password: passwordInput,
    interuptions: INTERUPTIONInput,
    backuprestoration: BACKUPRESTORATIONInput,
    changes: CHANGESInput,
  };
```

After the balanced scorecards table structure has been prepared and is returned. The JSON objects are then combined into one JSON object where each of them is a node. They are then again returned to the initial function, where they are as such inserted into the database for the specific audit id.

Before replying to the API call and sending the data to the frontend, the report date is loaded and the company name for which this audit has been done is loaded. Later these two additional attributes will be used to show these values to the user in the user interface and are then going to be added into the balanced scorecard table and the full report (proof), for more detailed documentation.

Now that the overview of the backend structure and the implementation of the controls have been shown, the frontend architecture will also be presented, followed by the different user stories which have been implemented.

The front end is written using React. The structure is separated into five different parts. The front end consists of the “App” folder, where the App Router can be found, which is responsible for the routing through the application. The next are the assets, which contain the images used in the user interface, the interfaces used through the application, and the login check, which is used to check if a logged-in user can access the URL which he is trying to access. For example, an internal auditor cannot access the pages of the external auditor. The following part is the components. React

components are reusable parts of code, which can be independently called on any page. A simple component example is shown below, showing the banner picture which is used throughout the whole application. Important to notice here is that the application is using react-bootstrap as the style foundation, which makes the application resizable and usable on multiple different devices.

```
<Styles>
  <Jumbotron fluid className="jumbo">
    <Container>
      <h1>Database Audit</h1>
      <p>Welcome!</p>
    </Container>
  </Jumbotron>
</Styles>
```

Also, for the styling of the components, the library styled-components are used. Styled-components enable global styles, and also make it possible to write the code and CSS style at the same time, making the conditional rendering of the components easier. The components have the same structure, they consist of the functional component itself, a resource file, where the texts used for the component are saved. This makes it also easier for future development, to build the multi-language functionality. Each component that makes API calls, has an API folder, where the functions handling the API calls can be found.

The next part is the pages, which are collections of components that are shown to the end-user. They are divided into admin pages, external pages, internal pages, and also pages which all of them use (the error page, the login failed page, and the home page which is shown before the login page to each of them). The code of the external auditor`s home page is shown below, as an example of how the pages are written.

```
<React.Fragment>
 {userState.email !== undefined ? (
   <React.Fragment>
     <div>
       <h3>Add new client</h3>
       <EAddClients email={userState.email} />
     </div>
     <div>
       <h3>Get all clients</h3>
       <EClientsList email={userState.email} />
     </div>
   </React.Fragment>
 ) : (
   <React.Fragment>
     <p>External user could not be found!</p>
```

```
16.         </React.Fragment>
17.       )}
18.       <Row>
19.         <Logout />
20.       </Row>
21.     </React.Fragment>
```

The “EAddClients” and “EClientsList” are components that need the property email, to render the components. The next part of the frontend application is the redux folder. Since this application is not using cookies but states instead. For that reason, the application is using React Redux as a state-manager with Redux-Saga as a middleware. Redux is structured into actions, saga, and reducer, which are then all combined in a store, and provided as such over the whole application. The code below is showing an example action from the application where a call to action has been made, for the user to login and to be set in the Redux store.

```
1.  export const setUser = (props: IUserCall) => {
2.    return {
3.      type: SET_USER,
4.      user: props,
5.    };
6.  };
```

After the action has been called it is sent to the reducer with the type “SET_USER”. But, because the application is using Redux-Saga, the message on the way to the reducer is intercepted. The Sagas make the synchronization of the API calls and the error handling easier. The code below is showing an example saga, where the generator “fetchUserAsyncWatcher” is collecting all “SET_USER” actions and sending them to the generator “fetchUserAsync”. The Sagas then handle the synchronization of the API call and send the result to the reducer with a new type (success or fail depending on the results).

```
1.  function* fetchUserAsync(action: IUserAction) {
2.    try {
3.      const status: boolean = yield connectUser(action.user);
4.      let user: IUserState = {
5.        email: action.user.email,
6.        password: action.user.password,
7.        type: action.user.type,
8.        status: status,
9.      };
10.     yield put({ type: SET_USER_SUCCESS, user: user });
11.   } catch (error) {
12.     yield put({ type: SET_USER_FAIL, user: undefined });
13.   }
14. }
```

```
15.
16. export function* fetchUserAsyncWatcher() {
17.   yield takeEvery(SET_USER, fetchUserAsync);
18. }
```

After the new type and value have been sent to the reducer, the reducer checks the incoming type and depending on the value of the type sets the new state value, as shown in the code below.

```
1.  const ExistingUserReducer = (
2.    state: ExistingUserDetails = initialState,
3.    action: IExistingUser
4.  ) => {
5.    switch (action.type) {
6.      case GET_USER_SUCCESS:
7.        return action.existinguser;
8.      case GET_USER_FAIL:
9.        return undefined;
10.     default:
11.       return state;
12.   }
13. };
```

To make the development with Redux easier, there is an extension for Google Chrome [19], which shows the flow of the action calls and how the states in the store are changing, as shown in [Figure 16].

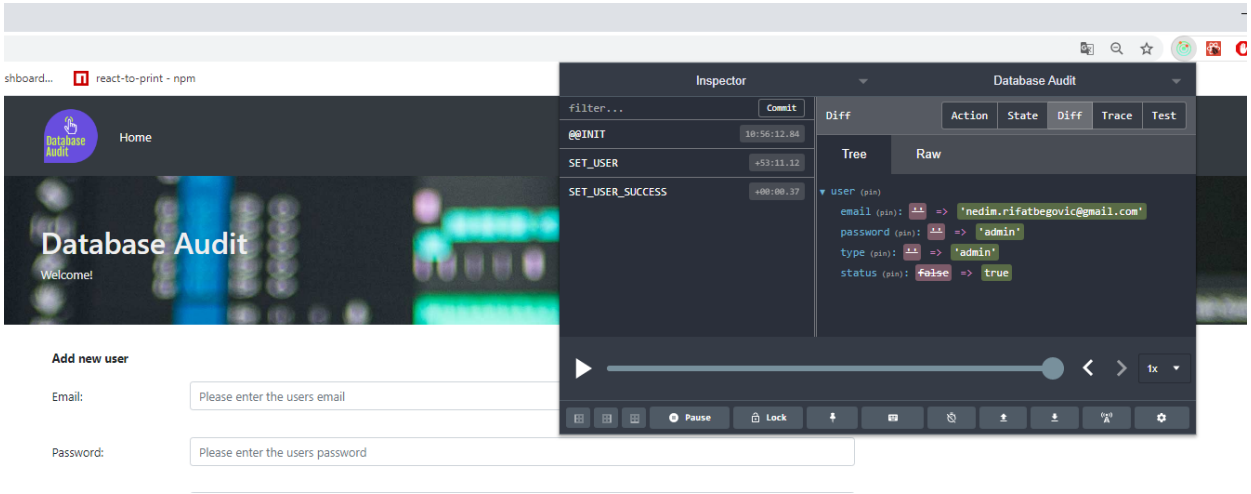

Figure 16: Redux DevTools Example

Since the overview of the frontend application has been given, now the user stories for each user type will be presented, showing the user interfaces for each page, and how the results from the backend have been presented (balanced scorecard, proof, etc.).

After the start of the application, before the users` can log in, a home page is loaded, giving a short description of the project (a short overview of the prototypes` architecture, which controls` aims the model to fulfill).

The user can then navigate to his login page (depending on the user type – admin, internal auditor, or external auditor). [Figure 17] shows the home page of the prototype.

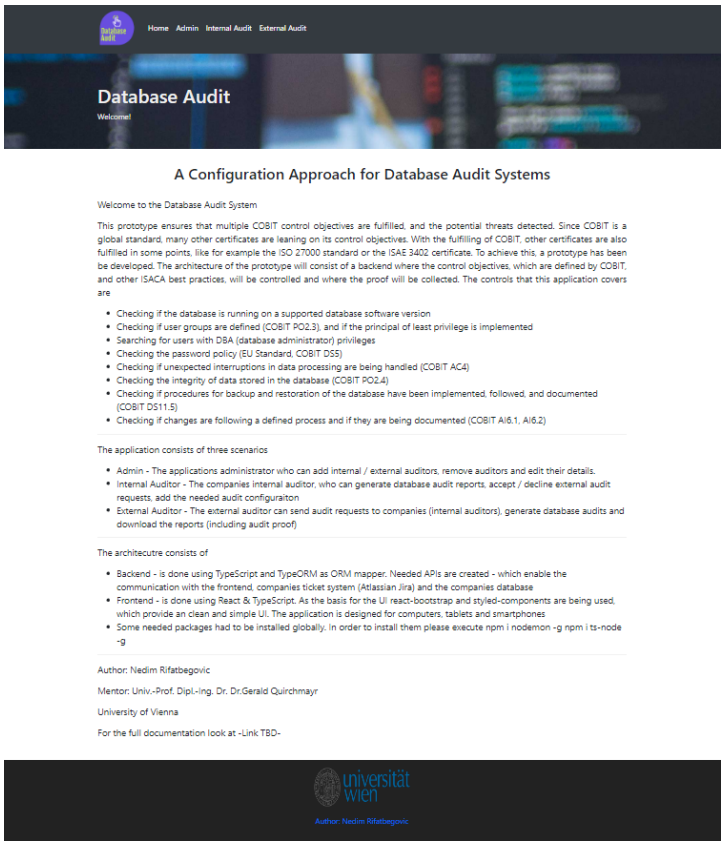

Figure 17: Home Page of the Prototype

Each user gets the same looking form during the login, where he needs to enter his credentials (email and password) to login into the application.

The first user story that is shown is the application administrator`s (admin) user story. After the admin has logged in, his home page opens, where the admin can add new users, get details of existing users, or remove a user. The home page view will be shown under [Figure 18] for the PC version, [Figure 19] for the iPad version, and [Figure 20] for the iPhone version of the page.

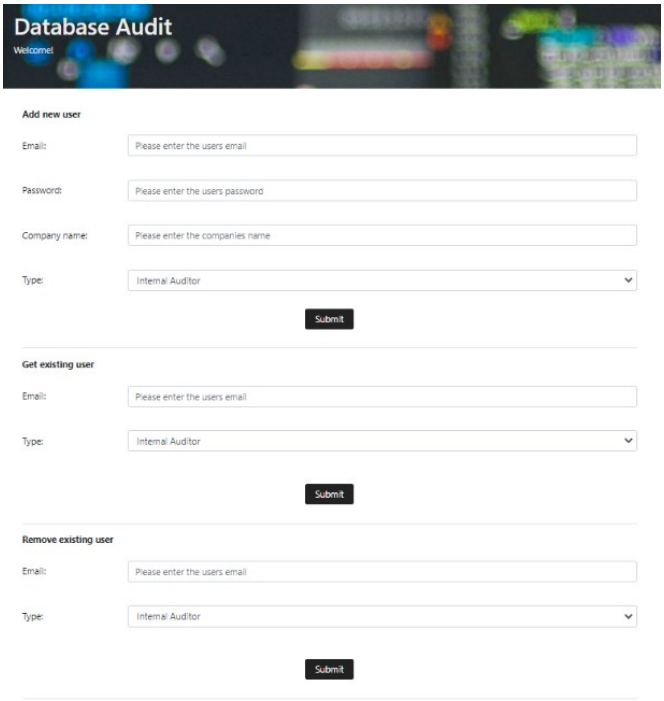

Figure 18: Admin Home Page – PC Version

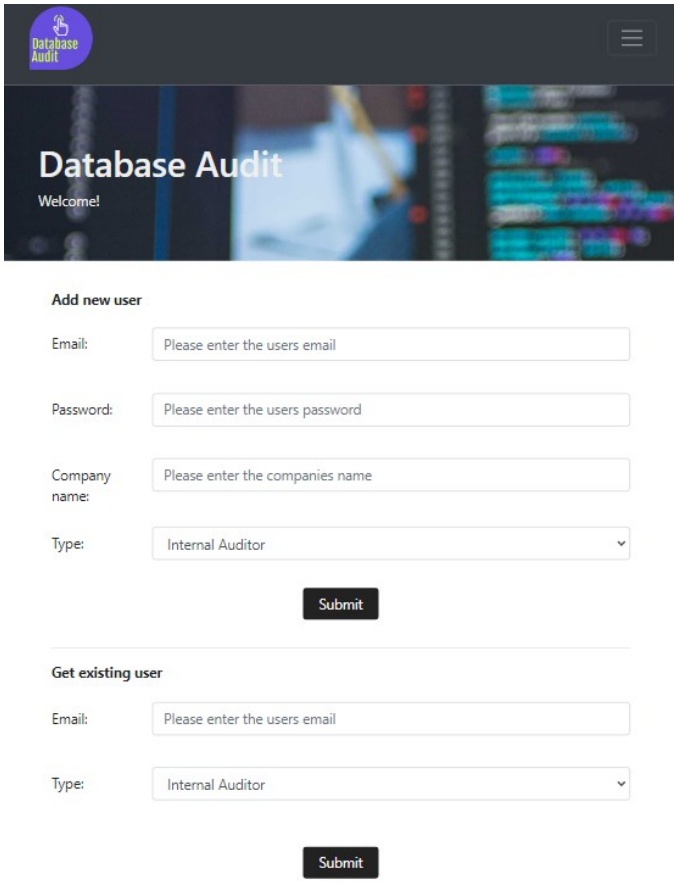

Figure 19: Admin Home Page – iPad Version

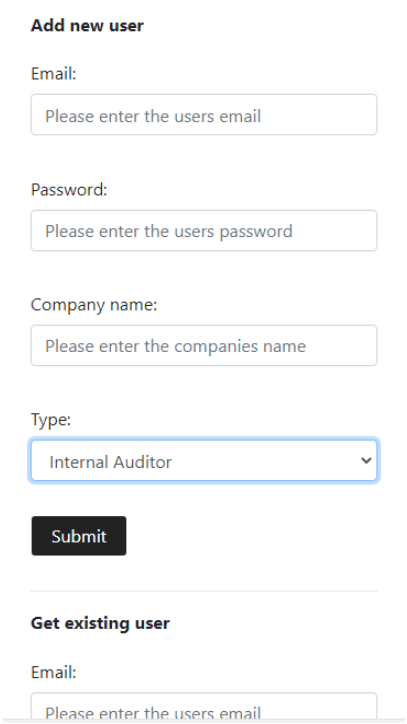

Figure 20: Admin Home Page – iPhone Version

As the screenshots show, the page can be used on multiple different devices. In the case the admin wants to add a new user, he needs to enter the user`s email, initial password – which the added user can change as soon as he logs in for the first time, the user`s company name, and the type – if he is an internal or external auditor. The field type has been added for the cases where the same person is working as an external auditor, but also as an internal auditor for the enterprise he is employed at. This way he can be added as an internal auditor for the own company and as an external auditor when he needs to audit other companies.

In the case the admin wants to get the details of an existing auditor, he needs to enter the auditor`s email and type – if he is an external or internal auditor.

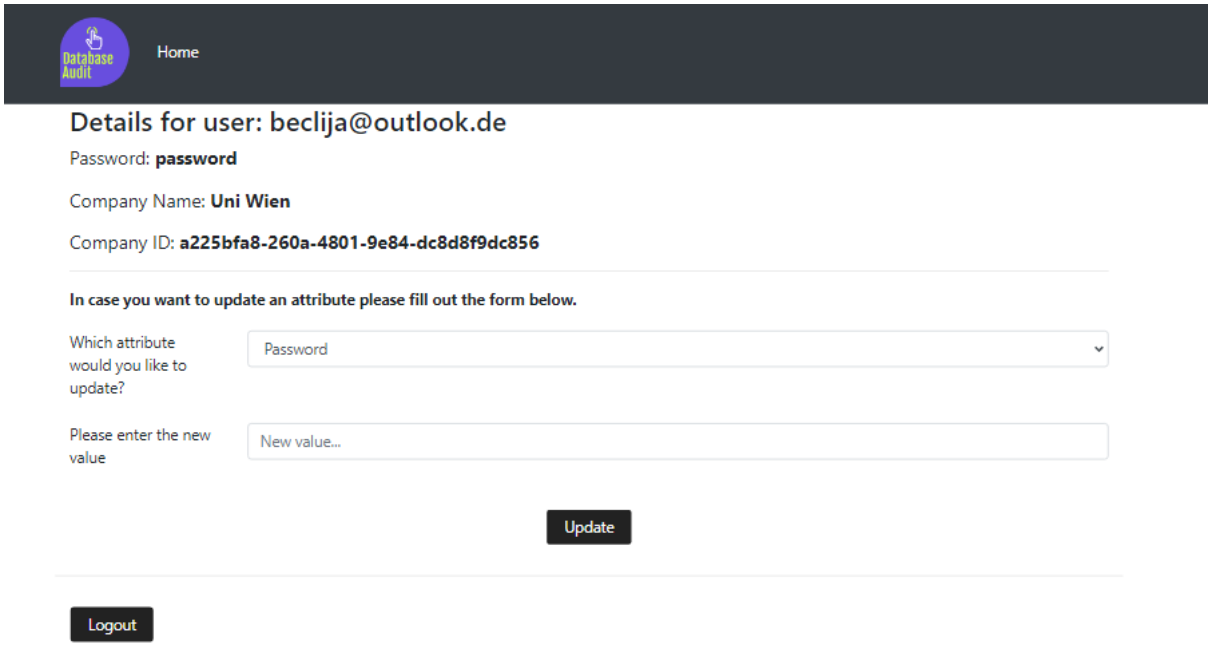

Figure 21: Admin User Details Page

As [Figure 21] shows, after the user has been found, the admin is forwarded to the user details page. Where the auditor`s details can be seen, and the admin can update the auditor`s attributes, in the case he forgets his password, or if he changes the company, he is currently employed at. The admin can also remove auditors when needed, by entering the user`s email and type.

The internal auditor`s user story starts after the internal auditor has logged in to the application. After the login he is forwarded to a home page, where he can audit his database (generate a report), to load the pending requests for the needed configuration (which represent the issue requests sent by external auditors` in the case they encountered an issue during their audit), to load the overview of all external audit requests, and to load an overview of all previous audits` of his database. [Figure 22] is showing the home page before any of these functionalities have been called.

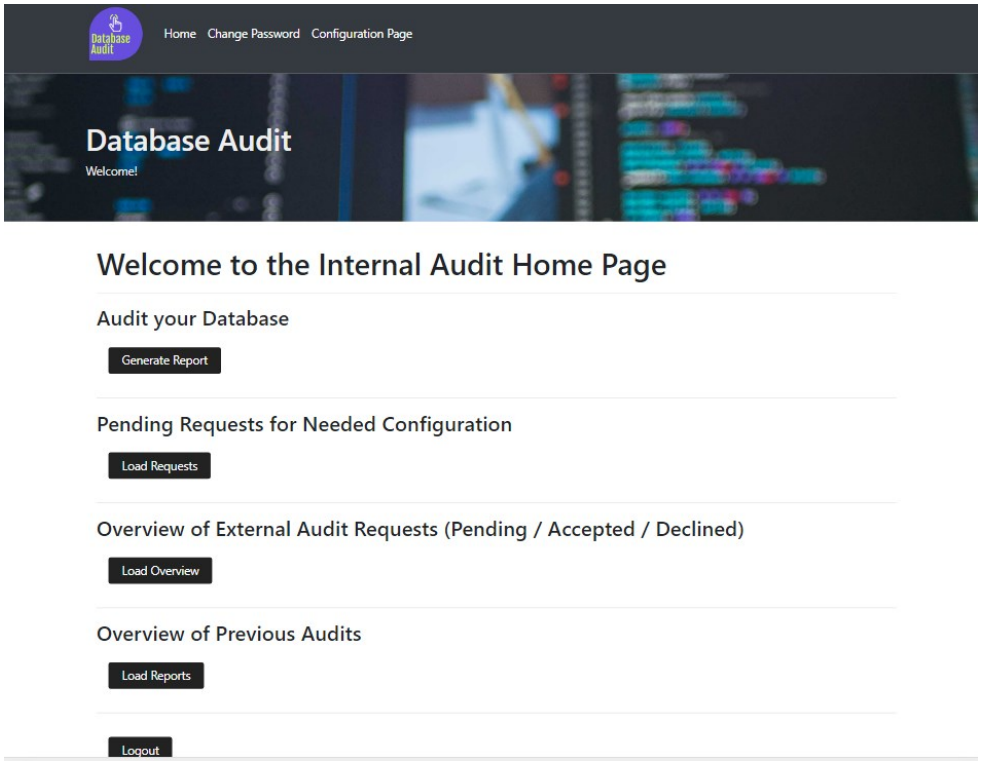

Figure 22: Internal Auditor – Home Page

After the internal auditor has generated the report, the balanced scorecard is shown as can be seen in [Figure 23]. The balanced scorecard consists of the characteristics that are being fulfilled with this model, as columns, and of the different controls that have been analysed during the database audit, like rows. The results are then shown, showing the detected risk levels, in different colours for an improved user experience.

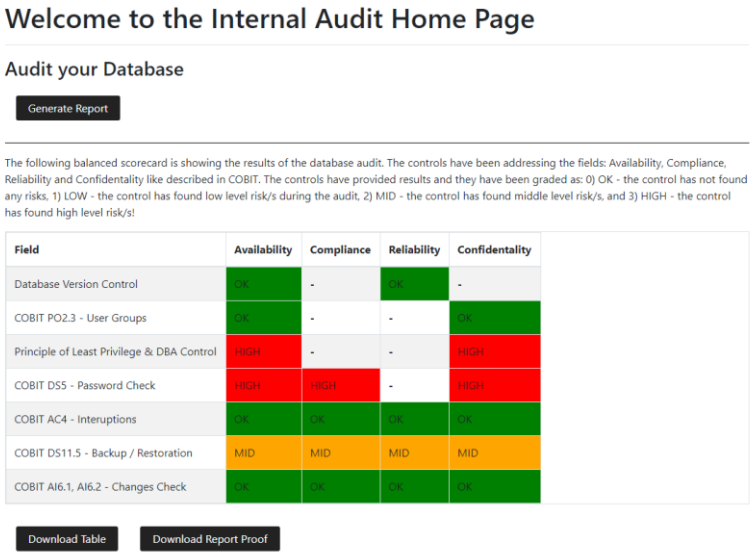

Figure 23: Internal Auditor – Generated Report – PC Version

The following two figures [Figure 24], [Figure 25] are showing the generated balanced scorecard also on iPad and iPhone.

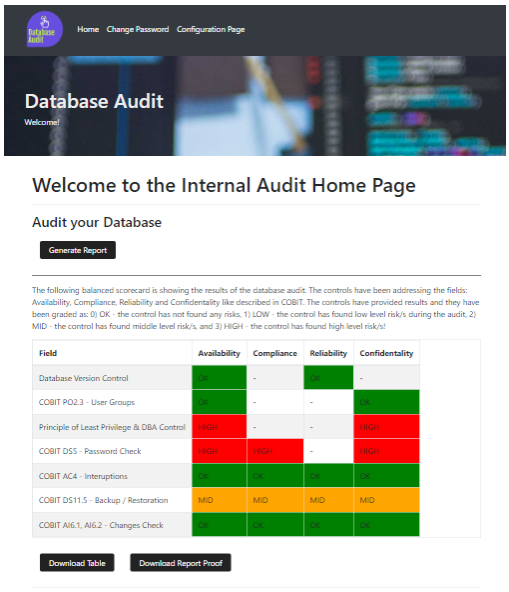

Figure 24: Internal Auditor – Generated Report – iPad Version

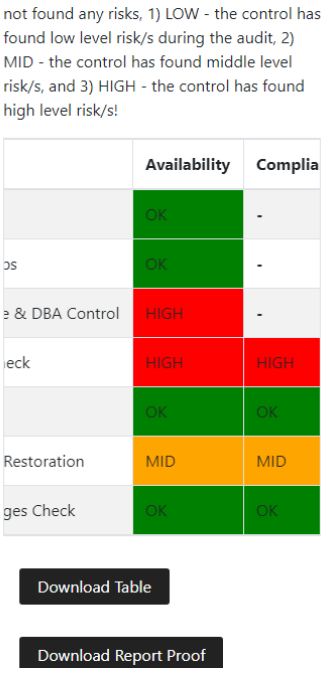

Figure 25: Internal Auditor – Generated Report – iPhone Version

The internal auditor can execute the “Download Table” action, where the balanced scorecard will be downloaded as a PDF file, or to execute the “Download Report Proof” action, where the collected proof during the database audit will be downloaded as a

PDF file. The PDF version of the balanced scorecard is shown in [Figure 26], while the collected proof is shown in [Figure 27].

Figure 26: Balanced Scorecard – PDF Version

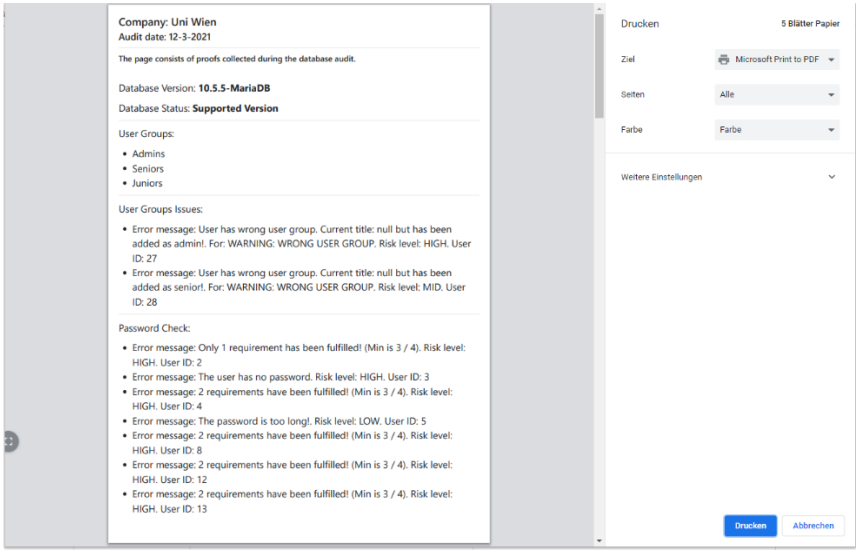

Figure 27: Collected Database Audit Proof – PDF Version

As shown in the previous two figures, both documents contain the name of the company which database has been audited and the date of the audit. The proof document contains the collected data during each of the controls. For example, which database software version has been found, and if it is supported, which user groups have been found (if any), etc.

The internal auditor can also load all issue requests. Only the unresolved ones will be shown, with the audit id for which the request has been sent, and with the external auditor`s email, so that the internal auditor can contact him after the issue has been resolved. After the issue has been resolved he can update the request, which will set the resolved status of the audit to true (which means the issue is fixed).

Also, the requests of all external auditors can be loaded. The loaded table contains the audit id, the email of the external auditor, and the current audit status. New audit requests are labelled with the status “Pending” and depending on the internal auditor`s choice they can be set to “Accepted” or “Declined”.

The overview of previous audits` can also be loaded, where the internal auditor gets a table containing the report date, when the audit has been executed, and the report ids for each database audit report which can be selected. This field has been added since each audit can have multiple reports generated for it. The user can then select a specific audit report, and the data will be loaded from the backend and presented to the internal auditor with the balanced scorecard, and the possibility to download the balanced scorecard as a PDF file, and to download the collected proof for the specific report, also as a PDF file.

These three functionalities have been shown below [Figure 28].

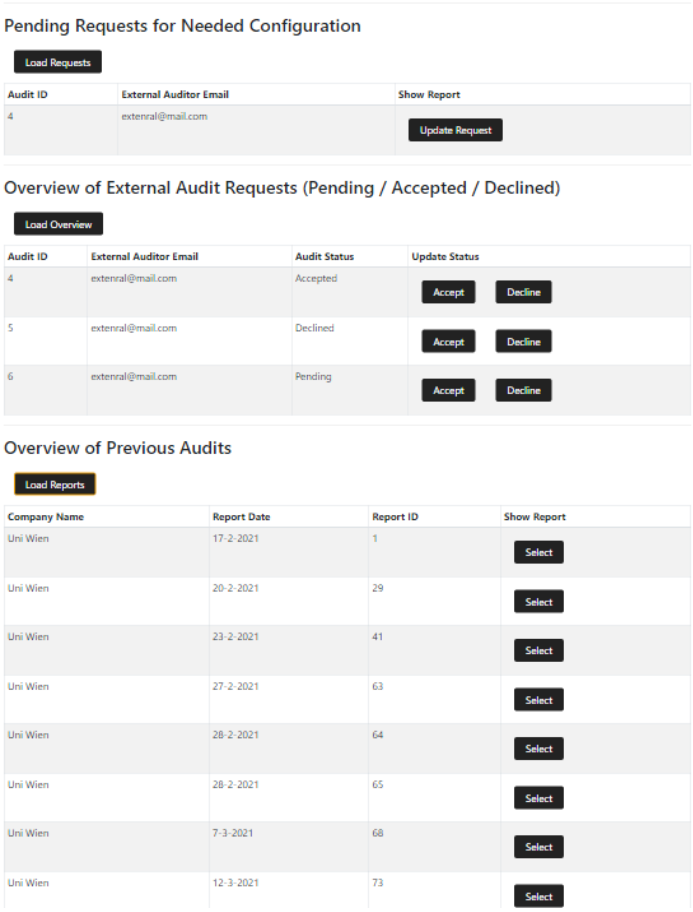

Figure 28: Internal Auditor – Loaded Data from the Backend

To audit the database, the internal auditor first needs to add the needed configuration data. The fields that need to be added have been described in the model chapter. After the internal auditor has navigated to this page, in the navigation bar, an API call is sent to the backend, to check if there already exists a configuration for the internal auditor or not. In the case that no configuration has been found, the [Figure 29] is shown to the auditor, where he needs to upload the private key needed for the ticket system authentication and to enter the rest of the needed data. In the case a configuration has been found, the [Figure 30] is shown to the user, and he gets an overview of the currently added data. He can also select an attribute to update it with a new value.

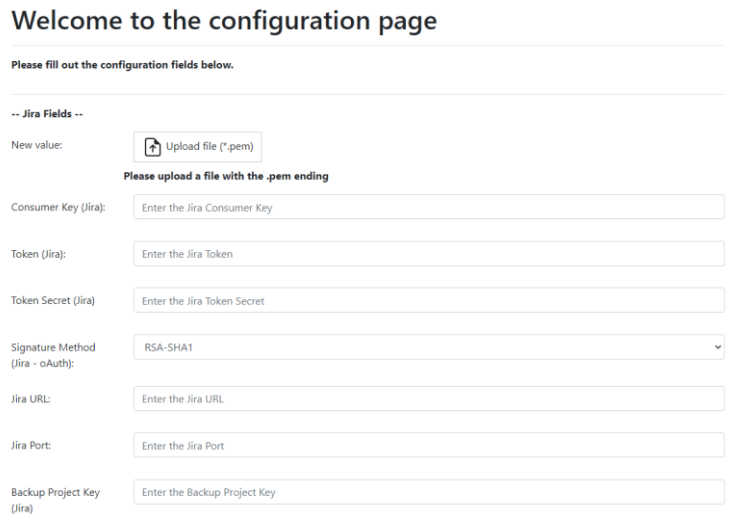

Figure 29: Internal Auditor – Config Page – Filling out the needed Data

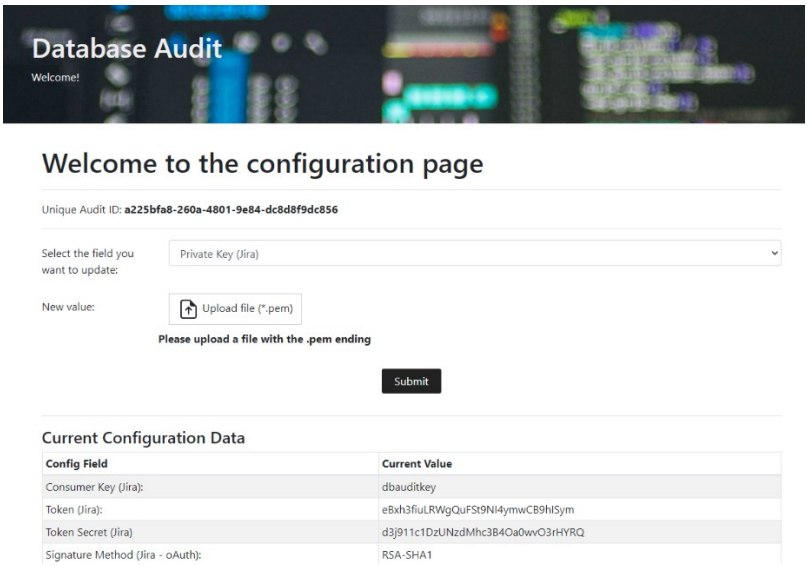

Figure 30: Internal Auditor – Config Page – Updating a specific Config Attribute

The internal auditor has the functionality to change his password, after navigating to the change password page. As shown in [Figure 31], the user can enter the new password into the input field, and the password will be updated.

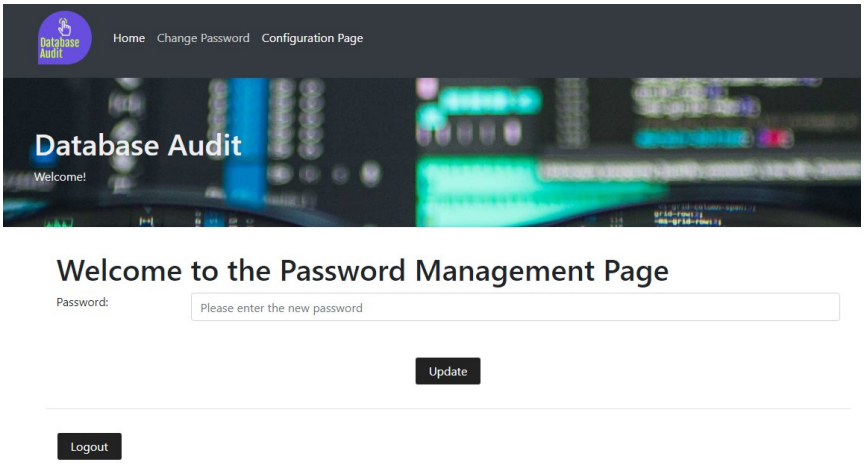

Figure 31: Internal Auditor – Changing Password

The external auditor user story also starts after the auditor has logged in to the application. As shown in [Figure 32], a home page is loaded, where the external auditor can send audit requests to internal auditors or load his list of already added clients. He is sending the audit request by adding the internal auditor`s unique id into the input field.

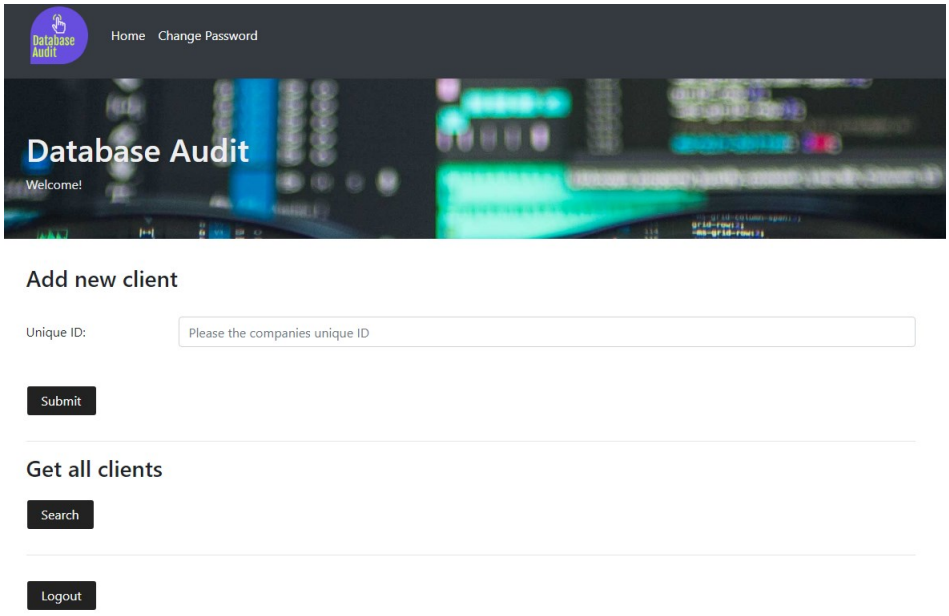

Figure 32: External Auditor – Home Page

After the external auditor has loaded all clients, a table is shown, which can be seen in [Figure 33], where the external auditor gets an overview of all his audit requests, of the company name which he wants to audit, their unique id, the status if they already accepted or declined his request (or if it is still pending), and the date the request has been sent.

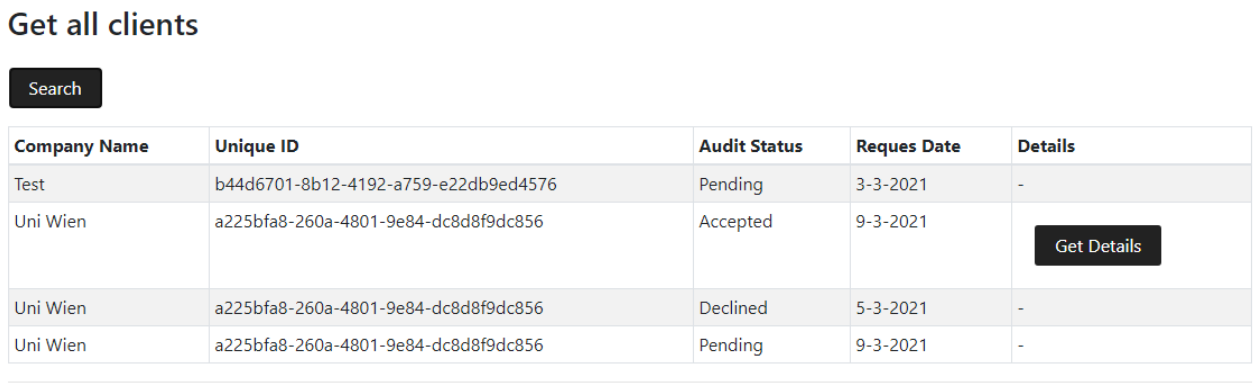

Figure 33: External Auditor – List of Loaded Audit Requests

The external auditor can only open the audits which have been accepted by the internal auditor. After the external auditor opens the specific audit details, he can generate a new audit report (to audit the database of the enterprise), load the previously generated audits for this audit (that have been generated by him), or submit a request (in the case the audit has failed) – an issue request will be sent to the internal auditor, so he is notified that there exists an issue with the configuration provided by him. This page is shown in [Figure 34].

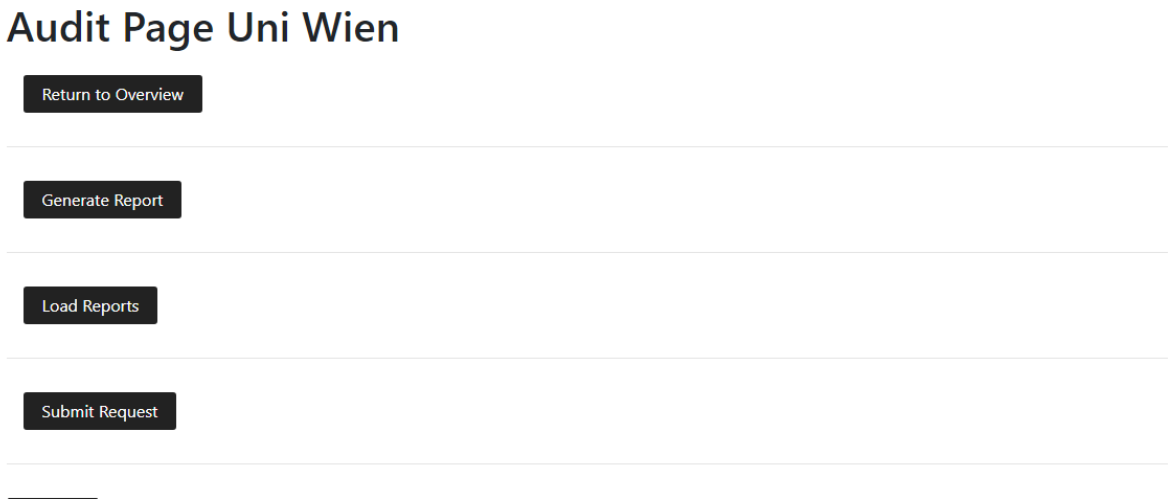

Figure 34: External Auditor – Audit Details Page

The view of this page after the audit has been done, and the list of previous reports has been loaded is shown below in [Figure 35].

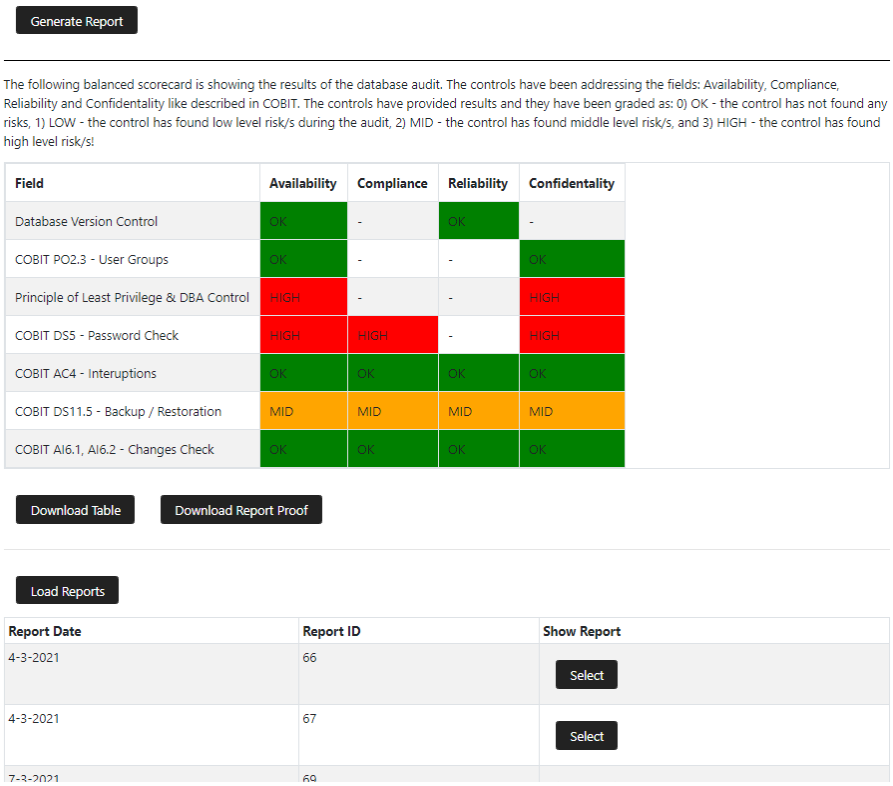

Figure 35: External Auditor – Audit Details Page – Data Loaded

The external auditor can download the balanced scorecard for each of these reports and download the collected proof as shown in the use case of the internal auditor. He can also change the password, the same way as presented in the use case of the internal auditor.

The implementation of the prototype has been presented, showing the architecture of the application (both backend and frontend), how the controls have been implemented, and the different user stories for the three different user types. In the next chapter, the scenario-based testing of the prototype will be presented, which also shows a few questionaries of multiple IT auditors and a web designer on the functionality and usability of the presented prototype.

7. Scenario-Based Testing of the Prototypes Functionality and Usability

The presented prototype has been tested with a scenario-based approach, a questionnaire has been sent to multiple auditors and a web design to test the usability of the prototype and to get feedback from potential end-users.

The database and the ticket system have been filled out with test data. Most of the data has been generated with an online data generator [13]. It is important to mention that only the test database has been filled with the generated data, while all the data from the prototype`s database, has been generated during the testing of the prototype.

The test database has been called “dbauditcompanyexample”, and contains three different classes (“logs”, “user”, and “usergroups” class). Three different user groups have been added (Admins, Seniors, and Juniors). After that, 28 users have been added, with the attributes, user id, first name, last name, email, user group id, and title (seniority level). In the fields, intentional mistakes have been added, like for example some users with no password, passwords not fulfilling the minimum password requirements, users with no title, to test if the controls are detecting the issues and risk factors. Also, 26 logs have been added, with the attributes log id, project key, ticket key, and type (Error, Change, Backup, Restoration). Here a ticket has been added with no ticket key, to test if the implemented controls would detect the risk. The code shown below is showing how the logs have been filled with SQL commands.

```
1. insert into logs (LogID, ProjectKey, TicketKey, Type, CreatedAt, UpdatedAt) values (6,
   'CHAN', 'CHAN-1', 'Change', '2019-12-17 17:39:53', '2020-01-02 06:35:01');
2. insert into logs (LogID, ProjectKey, TicketKey, Type, CreatedAt, UpdatedAt) values (7,
   'CHAN', 'CHAN-2', 'Change', '2019-12-11 19:08:36', '2020-10-21 06:57:06');
3. insert into logs (LogID, ProjectKey, TicketKey, Type, CreatedAt, UpdatedAt) values (8,
   'CHAN', 'CHAN-3', 'Change', '2019-11-23 01:12:58', '2020-09-08 02:23:18');
4. insert into logs (LogID, ProjectKey, TicketKey, Type, CreatedAt, UpdatedAt) values (9,
   'CHAN', 'CHAN-4', 'Change', '2020-07-25 18:00:00', '2020-03-23 10:49:14');
5. insert into logs (LogID, ProjectKey, TicketKey, Type, CreatedAt, UpdatedAt) values (10,
   'CHAN', 'CHAN-5', 'Change', '2020-08-03 21:10:10', '2020-02-02 18:47:01');
6. insert into logs (LogID, ProjectKey, TicketKey, Type, CreatedAt, UpdatedAt) values (11,
   'CHAN', 'CHAN-6', 'Change', '2020-01-27 03:43:07', '2020-12-11 21:28:05');
```

To test the frontend, the user stories have been gone through multiple times, trying out each functionality and trying to provoke errors, with for example wrong inputs, trying to open pages with the wrong access, etc. After everything has been

implemented and tested, a questionnaire has been created, including the screenshots of each page and each functionality, and has been sent to five different testers (four of them have been auditors, and one a web designer).

For each page 9 or 10 questions have been asked (depending on the specific page). Some of the questions have been open questions, and some have been questions where the tester needed to grade the specific page (1 – lowest, 10 – highest). Below the questionaries are shown, and the results of the testing and the research will be presented in the next chapter on the discussion of the results.

Auditor – D. Cengic

Full Name	D. Cengic
Experience with Audits (have you been working, or do you currently work as an IT Auditor)? If yes, in which company (optional)?	Experience: 3+ years Company: Deloitte Austria
Below the questions, you can see multiple screenshots from a database audit software. Please, look at them and answer the following questions/grade the user interface and usability from 1 to 10. The application consists of three use cases: 1. Application Administrator Page– for admins of the application 2. Internal Auditor Page 3. External Auditor Page – auditors employed by external companies, specialized in auditing Open Questions: Please answer the questions with 2-3 sentences 1-10: 1 – Lowest,10 – Highest	
Admin Page	
Admin Home Page	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Since there are 3 different functionalities, all important for the functional database audit, I would use all of them, based on the use case and/or my needs.
How satisfied are you with the	10

available features? (1-10)	
How likely are you to refer to this web application? Why or why not? (open question)	I would very likely recommend the application since it has a clean design and fulfills the functionality it is intended.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC, since in most cases I would access the presentation from my work device (laptop).
How do you like the interface on the PC? (1-10)	8
How do you like the interface on the iPad? (1-10)	9
How do you like the interface on the iPhone? (1-10)	7
Any other remarks? (open question)	No.
Admin User Details	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	The page is relatively self-explanatory. I would use all of the presented features since they represent the basic user administration functionality, regarding the captured and stored user data.
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would very likely recommend the application since it has a clean design and fulfills the functionality it is intended.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC, since in most cases I would access the presentation from my work device (laptop).
How do you like the interface on the PC? (1-10)	8
How do you like the interface on the iPad? (1-10)	9

How do you like the interface on the iPhone? (1-10)	7
Any other remarks? (open question)	No.
Internal Page Home	
Do you like the interface, is the page easy to understand? (1-10)	8
Which features of the page would you use, and are there any options that are missing? (open question)	Overview of the previous audits.
How satisfied are you with the available features? (1-10)	9
How likely are you to refer to this web application? Why or why not? (open question)	I would very likely recommend the application since it has a clean design and fulfills the functionality it is intended.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC, since in most cases I would access the presentation from my work device (laptop).
How do you like the interface on the PC? (1-10)	8
How do you like the interface on the iPad? (1-10)	9
How do you like the interface on the iPhone? (1-10)	7
Any other remarks? (open question)	No.
Internal Page Report	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Generate and download the report. Information about control risk mitigation would be helpful, especially if it is covered through some other tested control.

How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would very likely recommend the application since it has a clean design and fulfills the functionality it is intended. It also visualizes and helps to get the daily job effectively done.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC, since in most cases I would access the presentation from my work device (laptop).
How do you like the interface on the PC? (1-10)	10
How do you like the interface on the iPad? (1-10)	10
How do you like the interface on the iPhone? (1-10)	10
Any other remarks? (open question)	No.
Internal Requests	
Do you like the interface, is the page easy to understand? (1-10)	9
Which features of the page would you use, and are there any options that are missing? (open question)	The page is relatively self-explanatory as well. I would use all of the presented features.
How satisfied are you with the available features? (1-10)	9
How likely are you to refer to this web application? Why or why not? (open question)	I would very likely recommend the application since it has a clean design and fulfills the functionality it is intended. It also visualizes and helps to get the daily job effectively done.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC, since in most cases I would access the presentation from my work device (laptop).
How do you like the interface on the PC? (1-10)	8

How do you like the interface on the iPad? (1-10)	8
How do you like the interface on the iPhone? (1-10)	8
Any other remarks? (open question)	No.
Internal Page – Save Table	
Do you like the interface, is the page easy to understand? (1-10)	9
Which features of the page would you use, and are there any options that are missing? (open question)	Export as PDF would be mostly user functionality, however, export as an excel document would be more useful, since in such case data would be reusable.
How satisfied are you with the available features? (1-10)	7
How likely are you to refer to this web application? Why or why not? (open question)	I would very likely recommend the application since it has a clean design and fulfills the functionality it is intended. It also visualizes and helps to get the daily job effectively done.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC, since in most cases I would access the presentation from my work device (laptop).
Any other remarks? (open question)	No.
Internal Page – Save Report	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Export as PDF would be mostly user functionality.
How satisfied are you with the available features? (1-10)	10

How likely are you to refer to this web application? Why or why not? (open question)	I would very likely recommend the application since it has a clean design and fulfills the functionality it is intended. It also visualizes and helps to get the daily job effectively done.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC, since in most cases I would access the presentation from my work device (laptop).
Any other remarks? (open question)	No.
Internal Config Page – First Configuration Input	
Do you like the interface, is the page easy to understand? (1-10)	7
Which features of the page would you use, and are there any options that are missing? (open question)	The page is relatively self-explanatory as well. I would use all of the presented features.
How satisfied are you with the available features? (1-10)	9
How likely are you to refer to this web application? Why or why not? (open question)	I would very likely recommend the application since it has a clean design and fulfills the functionality it is intended. It also visualizes and helps to get the daily job effectively done.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC, since in most cases I would access the presentation from my work device (laptop).
Any other remarks? (open question)	No.
Internal Config Page – Configuration Update	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Upload file function.

How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would very likely recommend the application since it has a clean design and fulfills the functionality it is intended. It also visualizes and helps to get the daily job effectively done.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC, since in most cases I would access the presentation from my work device (laptop).
Any other remarks? (open question)	No.

External Home Page	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	The page is relatively self-explanatory as well. I would use all of the presented features.
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would very likely recommend the application since it has a clean design and fulfills the functionality it is intended. It also visualizes and helps to get the daily job effectively done.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC, since in most cases I would access the presentation from my work device (laptop).
Any other remarks? (open question)	No.
External Home Page – All Clients	
Do you like the interface, is the page easy to understand? (1-10)	9
Which features of the page would you	Search functionality. Option to set personalized filters would be useful.

use, and are there any options that are missing? (open question)	
How satisfied are you with the available features? (1-10)	8
How likely are you to refer to this web application? Why or why not? (open question)	I would very likely recommend the application since it has a clean design and fulfills the functionality it is intended. It also visualizes and helps to get the daily job effectively done.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC, since in most cases I would access the presentation from my work device (laptop).
Any other remarks? (open question)	No.
External Home Page – Client Details	
Do you like the interface, is the page easy to understand? (1-10)	9
Which features of the page would you use, and are there any options that are missing? (open question)	All four functions.
How satisfied are you with the available features? (1-10)	8
How likely are you to refer to this web application? Why or why not? (open question)	I would very likely recommend the application since it has a clean design and fulfills the functionality it is intended. It also visualizes and helps to get the daily job effectively done.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC, since in most cases I would access the presentation from my work device (laptop).
Any other remarks? (open question)	No.
External Home Page – Client Details	Same as previous.
Do you like the interface, is the page easy to understand? (1-10)	9

Which features of the page would you use, and are there any options that are missing? (open question)	All four functions.
How satisfied are you with the available features? (1-10)	8
How likely are you to refer to this web application? Why or why not? (open question)	I would very likely recommend the application since it has a clean design and fulfills the functionality it is intended. It also visualizes and helps to get the daily job effectively done.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC, since in most cases I would access the presentation from my work device (laptop).
Any other remarks? (open question)	No.
External Home Page – Client Details Generated Data	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Generate and download the report. Information about control risk mitigation would be helpful, especially if it is covered through some other tested control.
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would very likely recommend the application since it has a clean design and fulfills the functionality it is intended. It also visualizes and helps to get the daily job effectively done.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC, since in most cases I would access the presentation from my work device (laptop).
Any other remarks? (open question)	No.

Auditor – B. Chissico

Full Name	B. Chissico
Experience with Audits (have you been working, or do you currently work as an IT Auditor)? If yes, in which company (optional)?	BDO Austria, +2 years' experience
Below the questions, you can see multiple screenshots from a database audit software. Please, look at them and answer the following questions/grade the user interface and usability from 1 to 10. The application consists of three use cases: 4. Application Administrator Page– for admins of the application 5. Internal Auditor Page 6. External Auditor Page – auditors employed by external companies, specialized in auditing Open Questions: Please answer the questions with 2-3 sentences 1-10: 1 – Lowest,10 – Highest	
Admin Page	
Admin Home Page	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	One Open Question: What does "get User" do? I assume it lets the admin edit an existing user. Maybe the naming could be changed to make it more explicit.
How satisfied are you with the available features? (1-10)	8
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend the web application. The reasoning is that such an application will tremendously increase the communication between different stakeholders who are involved in the auditing process.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	Hard to answer, I'd use it primarily on PC. Although, there might be specific use cases where it would be necessary or an added benefit to having it as a mobile version as well.
How do you like the interface on the PC? (1-10)	10

How do you like the interface on the iPad? (1-10)	10
How do you like the interface on the iPhone? (1-10)	10
Any other remarks? (open question)	The 10 points are due to the simplicity of the interface.
Admin User Details	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Features that I would use: Generate Report, Overview previous reports Missing features: None spotted
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend the web application. The reasoning is that such an application will tremendously increase the communication between different stakeholders who are involved in the auditing process.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC
How do you like the interface on the PC? (1-10)	10
How do you like the interface on the iPad? (1-10)	10
How do you like the interface on the iPhone? (1-10)	10
Any other remarks? (open question)	-
Internal Page Home	
Do you like the interface, is the page easy to understand? (1-10)	8
Which features of the page would you use, and are there any options that are	Generate Report, Get the Previous Report

missing? (open question)	
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend the web application. The reasoning is that such an application will tremendously increase the communication between different stakeholders who are involved in the auditing process.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC
How do you like the interface on the PC? (1-10)	10
How do you like the interface on the iPad? (1-10)	10
How do you like the interface on the iPhone? (1-10)	10
Any other remarks? (open question)	-
Internal Page Report	
Do you like the interface, is the page easy to understand? (1-10)	7
Which features of the page would you use, and are there any options that are missing? (open question)	Generate Report
How satisfied are you with the available features? (1-10)	8
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend the web application. The reasoning is that such an application will tremendously increase the communication between different stakeholders who are involved in the auditing process.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC

How do you like the interface on the PC? (1-10)	8
How do you like the interface on the iPad? (1-10)	8
How do you like the interface on the iPhone? (1-10)	8
Any other remarks? (open question)	A „quality of life“ feature that I find missing is that you have to use the print function of the browser to generate the report in PDF. A “Download as PDF/Excel/CSV” would be an easy improvement that could go way.
Internal Requests	
Do you like the interface, is the page easy to understand? (1-10)	8
Which features of the page would you use, and are there any options that are missing? (open question)	Features I’d use: All of them
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend the web application. The reasoning is that such an application will tremendously increase the communication between different stakeholders who are involved in the auditing process.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC
How do you like the interface on the PC? (1-10)	10
How do you like the interface on the iPad? (1-10)	10
How do you like the interface on the iPhone? (1-10)	10
Any other remarks? (open question)	
Internal Page – Save Table	
Do you like the interface, is the page	7

easy to understand? (1-10)	
Which features of the page would you use, and are there any options that are missing? (open question)	
How satisfied are you with the available features? (1-10)	7
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend the web application. The reasoning is that such an application will tremendously increase the communication between different stakeholders who are involved in the auditing process.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC
Any other remarks? (open question)	I would like to see a “download as PDF”- Button for the reports rather than having to go via print as pdf.
Internal Page – Save Report	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	
How satisfied are you with the available features? (1-10)	
How likely are you to refer to this web application? Why or why not? (open question)	
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC
Any other remarks? (open question)	

Internal Config Page – First Configuration Input	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	All of them
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend the web application. The reasoning is that such an application will tremendously increase the communication between different stakeholders who are involved in the auditing process.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC
Any other remarks? (open question)	-
Internal Config Page – Configuration Update	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	All of them
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend the web application. The reasoning is that such an application will tremendously increase the communication between different stakeholders who are involved in the auditing process.
On which device would you use the application and	PC

why? (PC / Laptop, iPad or iPhone)	
Any other remarks? (open question)	

External Home Page	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	All of them
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend the web application. The reasoning is that such an application will tremendously increase the communication between different stakeholders who are involved in the auditing process.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC
Any other remarks? (open question)	
External Home Page – All Clients	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	All of them
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or	I would recommend the web application. The reasoning is that such an application will tremendously increase the communication between different stakeholders who are involved in the auditing process.

why not? (open question)	
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC
Any other remarks? (open question)	
External Home Page – Client Details	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	All of them
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend the web application. The reasoning is that such an application will tremendously increase the communication between different stakeholders who are involved in the auditing process.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC
Any other remarks? (open question)	
External Home Page – Client Details	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	All of them
How satisfied are you with the available features? (1-10)	10

How likely are you to refer to this web application? Why or why not? (open question)	I would recommend the web application. The reasoning is that such an application will tremendously increase the communication between different stakeholders who are involved in the auditing process.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC
Any other remarks? (open question)	
External Home Page – Client Details Generated Data	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	All of them
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend the web application. The reasoning is that such an application will tremendously increase the communication between different stakeholders who are involved in the auditing process.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC
Any other remarks? (open question)	

Auditor – S. Muratspahic

Full Name	S. Muratspahic
Experience with Audits (have you been working, or do you currently work as an IT Auditor)? If yes, in which company (optional)?	BDO, +5 years' experience

Below the questions, you can see multiple screenshots from a database audit software. Please, look at them and answer the following questions/grade the user interface and usability from 1 to 10.

The application consists of three use cases:

7. Application Administrator Page– for admins of the application
8. Internal Auditor Page
9. External Auditor Page – auditors employed by external companies, specialized in auditing

Open Questions: Please answer the questions with 2-3 sentences
1-10: 1 – Lowest,10 – Highest

Admin Page	
Admin Home Page	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Everything is available
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would refer because of the usability
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	Every application -> PC most
How do you like the interface on the PC? (1-10)	10
How do you like the interface on the iPad? (1-10)	10
How do you like the interface on the iPhone? (1-10)	10
Any other remarks? (open question)	No
Admin User Details	
Do you like the interface, is the page	10

easy to understand? (1-10)	
Which features of the page would you use, and are there any options that are missing? (open question)	Everything is available
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would refer because of the usability
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	Every application -> PC most
How do you like the interface on the PC? (1-10)	10
How do you like the interface on the iPad? (1-10)	10
How do you like the interface on the iPhone? (1-10)	10
Any other remarks? (open question)	No
Internal Page Home	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Everything is available
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would refer because of the usability

On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	Every application -> PC most
How do you like the interface on the PC? (1-10)	10
How do you like the interface on the iPad? (1-10)	10
How do you like the interface on the iPhone? (1-10)	10
Any other remarks? (open question)	No
Internal Page Report	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Every feature-> nothing is missing
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	Usability is very good – simple to handle
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	Every – PC most
How do you like the interface on the PC? (1-10)	10
How do you like the interface on the iPad? (1-10)	10
How do you like the interface on the iPhone? (1-10)	10
Any other remarks? (open question)	No

Internal Requests	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Nothing is missing
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	Very much – an overview is very good, information from previous audits is useful
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	Every but pc most
How do you like the interface on the PC? (1-10)	10
How do you like the interface on the iPad? (1-10)	10
How do you like the interface on the iPhone? (1-10)	10
Any other remarks? (open question)	NO
Internal Page – Save Table	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Everything is available
How satisfied are you with the available features? (1-10)	10

How likely are you to refer to this web application? Why or why not? (open question)	Yes - usability
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	Every but PC most
Any other remarks? (open question)	No
Internal Page – Save Report	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Every feature – good overview
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	yes
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC most – but others too
Any other remarks? (open question)	No
Internal Config Page – First Configuration Input	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Every feature

How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	yes
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC most
Any other remarks? (open question)	No
Internal Config Page – Configuration Update	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	every
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	yes
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC most
Any other remarks? (open question)	No

External Home Page	
Do you like the interface, is the page easy to understand? (1-10)	10

Which features of the page would you use, and are there any options that are missing? (open question)	every
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	yes
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC most
Any other remarks? (open question)	No
External Home Page – All Clients	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	every
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	yes
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC most
Any other remarks? (open question)	No
External Home Page – Client Details	
Do you like the interface, is the page	10

easy to understand? (1-10)	
Which features of the page would you use, and are there any options that are missing? (open question)	every
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	yes
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC most
Any other remarks? (open question)	No
External Home Page – Client Details	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	every
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	Yes - usability
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC most
Any other remarks? (open question)	No

External Home Page – Client Details Generated Data	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	every
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	Yes - usability
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC most
Any other remarks? (open question)	No

7.4 Auditor – I. Hadzovic

Full Name	I. Hadzovic
Experience with Audits (have you been working, or do you currently work as an IT Auditor)? If yes, in which company (optional)?	Experience:2 years Company: PwC Sweden
Below the questions, you can see multiple screenshots from a database audit software. Please, look at them and answer the following questions/grade the user interface and usability from 1 to 10. The application consists of three use cases: 10. Application Administrator Page– for admins of the application 11. Internal Auditor Page 12. External Auditor Page – auditors employed by external companies, specialized in auditing Open Questions: Please answer the questions with 2-3 sentences 1-10: 1 – Lowest,10 – Highest	

Admin Page	
Admin Home Page	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	I would use all of them, based on the needs in each specific case.
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would highly recommend it since it has a great design, is easy to use, and has great functions/features.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC/Laptop since I mainly use a work device to access it.
How do you like the interface on the PC? (1-10)	9
How do you like the interface on the iPad? (1-10)	8
How do you like the interface on the iPhone? (1-10)	8
Any other remarks? (open question)	No.
Admin User Details	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	I would use all of the features since they present basic user administration functionality.
How satisfied are you with the available features? (1-10)	10

How likely are you to refer to this web application? Why or why not? (open question)	I would highly recommend it since it has a great design, great functions/features for what it is intended/used.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC/Laptop since I mainly use a work device to access it.
How do you like the interface on the PC? (1-10)	9
How do you like the interface on the iPad? (1-10)	8
How do you like the interface on the iPhone? (1-10)	8
Any other remarks? (open question)	No.
Internal Page Home	
Do you like the interface, is the page easy to understand? (1-10)	9
Which features of the page would you use, and are there any options that are missing? (open question)	Overview of previous audits.
How satisfied are you with the available features? (1-10)	9
How likely are you to refer to this web application? Why or why not? (open question)	I would highly recommend it since it has a great design, great functions/features for what it is intended/used.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC/Laptop since I mainly use a work device to access it.
How do you like the interface on the PC? (1-10)	10
How do you like the interface on the iPad? (1-10)	9

How do you like the interface on the iPhone? (1-10)	9
Any other remarks? (open question)	No.
Internal Page Report	
Do you like the interface, is the page easy to understand? (1-10)	9
Which features of the page would you use, and are there any options that are missing? (open question)	Generate and download the report.
How satisfied are you with the available features? (1-10)	9
How likely are you to refer to this web application? Why or why not? (open question)	I would highly recommend it since it has a great design, great functions/features for what it is intended/used. It also has a visualization feature which is great and helpful.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC/Laptop since I mainly use a work device to access it.
How do you like the interface on the PC? (1-10)	9
How do you like the interface on the iPad? (1-10)	9
How do you like the interface on the iPhone? (1-10)	9
Any other remarks? (open question)	No.
Internal Requests	
Do you like the interface, is the page easy to understand? (1-10)	9
Which features of the page would you use, and are there any options that are	I would use all of them, and I specifically like the overview of previous audits.

missing? (open question)	
How satisfied are you with the available features? (1-10)	9
How likely are you to refer to this web application? Why or why not? (open question)	I would highly recommend it since it has a great design, great functions/features for what it is intended/used.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC/Laptop since I mainly use a work device to access it.
How do you like the interface on the PC? (1-10)	9
How do you like the interface on the iPad? (1-10)	9
How do you like the interface on the iPhone? (1-10)	9
Any other remarks? (open question)	No.
Internal Page – Save Table	
Do you like the interface, is the page easy to understand? (1-10)	9
Which features of the page would you use, and are there any options that are missing? (open question)	The available feature to export as PDF is really good, although it would be great if you would have an option to save the table in excel format.
How satisfied are you with the available features? (1-10)	9
How likely are you to refer to this web application? Why or why not? (open question)	I would highly recommend it since it has a great design, great functions/features for what it is intended/used.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC/Laptop since I mainly use a work device to access it.

Any other remarks? (open question)	No.
Internal Page – Save Report	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Export report as PDF.
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would highly recommend it since it has a great design, great functions/features for what it is intended/used.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC/Laptop since I mainly use a work device to access it.
Any other remarks? (open question)	No.
Internal Config Page – First Configuration Input	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Page is very easy to use and I would use all of the available features.
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would highly recommend it since it has a great design, great functions/features for what it is intended/used.
On which device would you use the	PC/Laptop since I mainly use a work device to access it.

application and why? (PC / Laptop, iPad or iPhone)	
Any other remarks? (open question)	No.
Internal Config Page – Configuration Update	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Upload file function.
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would highly recommend it since it has a great design, great functions/features for what it is intended/used.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC/Laptop since I mainly use a work device to access it.
Any other remarks? (open question)	No.

External Home Page	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Page is very easy to use and I would use all of the available features.
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web	I would highly recommend it since it has a great design, great functions/features for what it is intended/used.

application? Why or why not? (open question)	
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC/Laptop since I mainly use a work device to access it.
Any other remarks? (open question)	No.
External Home Page – All Clients	
Do you like the interface, is the page easy to understand? (1-10)	9
Which features of the page would you use, and are there any options that are missing? (open question)	Search function for all clients.
How satisfied are you with the available features? (1-10)	9
How likely are you to refer to this web application? Why or why not? (open question)	I would highly recommend it since it has a great design, great functions/features for what it is intended/used.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC/Laptop since I mainly use a work device to access it.
Any other remarks? (open question)	No.
External Home Page – Client Details	
Do you like the interface, is the page easy to understand? (1-10)	9
Which features of the page would you use, and are there any options that are missing? (open question)	I would use all of the available features.
How satisfied are you with the available features? (1-10)	8

How likely are you to refer to this web application? Why or why not? (open question)	I would highly recommend it since it has a great design, great functions/features for what it is intended/used.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC/Laptop since I mainly use a work device to access it.
Any other remarks? (open question)	No.
External Home Page – Client Details	
Do you like the interface, is the page easy to understand? (1-10)	9
Which features of the page would you use, and are there any options that are missing? (open question)	I would use all of the available features.
How satisfied are you with the available features? (1-10)	8
How likely are you to refer to this web application? Why or why not? (open question)	I would highly recommend it since it has a great design, great functions/features for what it is intended/used.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC/Laptop since I mainly use a work device to access it.
Any other remarks? (open question)	No.
External Home Page – Client Details Generated Data	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Generate and download the report.

How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would highly recommend it since it has a great design, great functions/features for what it is intended/used. It also has a visualization feature which is great and helpful.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	PC/Laptop since I mainly use a work device to access it.
Any other remarks? (open question)	No.

7.5 Web Designer – R. Titz

Full Name	R. Titz
Experience with Audits (have you been working, or do you currently work as an IT Auditor)? If yes, in which company (optional)?	No (+2 years as a web designer, at Camelot GmbH)
Below the questions, you can see multiple screenshots from a database audit software. Please, look at them and answer the following questions/grade the user interface and usability from 1 to 10. The application consists of three use cases: 13. Application Administrator Page– for admins of the application 14. Internal Auditor Page 15. External Auditor Page – auditors employed by external companies, specialized in auditing Open Questions: Please answer the questions with 2-3 sentences 1-10: 1 – Lowest,10 – Highest	
Admin Page	
Admin Home Page	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are	All necessary input fields are there to get the use case done.

missing? (open question)	
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend this Application to every company which manages its databases as it unveils possible risks for the database.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	I would prefer a PC or Laptop, as it is the easiest to switch between input fields with a keyboard.
How do you like the interface on the PC? (1-10)	9
How do you like the interface on the iPad? (1-10)	9
How do you like the interface on the iPhone? (1-10)	8
Any other remarks? (open question)	
Admin User Details	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	I'd like to list all users I am allowed to see at once.
How satisfied are you with the available features? (1-10)	8
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend this Application to every company which manages its databases as it unveils possible risks for the database.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	I would prefer a PC or Laptop, as it is the easiest to switch between input fields with a keyboard.

How do you like the interface on the PC? (1-10)	9
How do you like the interface on the iPad? (1-10)	9
How do you like the interface on the iPhone? (1-10)	8
Any other remarks? (open question)	
Internal Page Home	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Every feature is easily accessible as there was also made use of color-coding.
How satisfied are you with the available features? (1-10)	9
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend this Application to every company which manages its databases as it unveils possible risks for the database.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	I would prefer a PC or Laptop, as it makes switching back and forth between the features the easiest.
How do you like the interface on the PC? (1-10)	9
How do you like the interface on the iPad? (1-10)	9
How do you like the interface on the iPhone? (1-10)	8
Any other remarks? (open question)	
Internal Page Report	
Do you like the interface, is the page easy to understand? (1-10)	10

Which features of the page would you use, and are there any options that are missing? (open question)	I don’t see any features missing. Every feature is easily accessible via a button.
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend this Application to every company which manages its databases as it unveils possible risks for the database.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	I would prefer a PC or Laptop, as there is the most screen real estate to plow through the information provided.
How do you like the interface on the PC? (1-10)	9
How do you like the interface on the iPad? (1-10)	9
How do you like the interface on the iPhone? (1-10)	5
Any other remarks? (open question)	As of its nature, it is hard to develop truly responsive tables.
Internal Requests	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	All features are there.
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend this Application to every company which manages its databases as it unveils possible risks for the database.
On which device would you use the	I would prefer a PC or Laptop, as there is the most screen real estate to plow through the information provided.

application and why? (PC / Laptop, iPad or iPhone)	
How do you like the interface on the PC? (1-10)	9
How do you like the interface on the iPad? (1-10)	9
How do you like the interface on the iPhone? (1-10)	5
Any other remarks? (open question)	As of its nature, it is hard to develop truly responsive tables.
Internal Page – Save Table	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	The color-coded table is well structured and gives instant oversight.
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend this Application to every company which manages its databases as it unveils possible risks for the database.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	I would prefer a PC or Laptop, as there is the most screen real estate to plow through the information provided.
Any other remarks? (open question)	
Internal Page – Save Report	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	The logs in the report are well structured.

How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend this Application to every company which manages its databases as it unveils possible risks for the database.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	I would prefer a PC or Laptop, as there is the most screen real estate to plow through the information provided.
Any other remarks? (open question)	The report could make use of tabulation, which makes it easier to scan through the document.
Internal Config Page – First Configuration Input	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Every necessary value is there.
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend this Application to every company which manages its databases as it unveils possible risks for the database.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	I would prefer a PC or Laptop, as there is the most screen real estate to plow through the information provided.
Any other remarks? (open question)	
Internal Config Page – Configuration Update	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you	I like to see the present configuration right down the feature to update them.

use, and are there any options that are missing? (open question)	
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend this Application to every company which manages its databases as it unveils possible risks for the database.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	I would prefer a PC or Laptop, as there is the most screen real estate to plow through the information provided.
Any other remarks? (open question)	

External Home Page	
Do you like the interface, is the page easy to understand? (1-10)	9
Which features of the page would you use, and are there any options that are missing? (open question)	I would like to see a subset of the clients right at the start of the page.
How satisfied are you with the available features? (1-10)	9
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend this Application to every company which manages its databases as it unveils possible risks for the database.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	I would prefer a PC or Laptop, as there is the most screen real estate to plow through the information provided.
Any other remarks? (open question)	The logout button would preferably belong at the top right corner.
External Home Page – All Clients	
Do you like the interface, is the page	9

easy to understand? (1-10)	
Which features of the page would you use, and are there any options that are missing? (open question)	The table is easy to comprehend.
How satisfied are you with the available features? (1-10)	9
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend this Application to every company which manages its databases as it unveils possible risks for the database.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	I would prefer a PC or Laptop, as there is the most screen real estate to plow through the information provided.
Any other remarks? (open question)	The Details tab doesn't need too much margin so each row with the details button could be narrower.
External Home Page – Client Details	
Do you like the interface, is the page easy to understand? (1-10)	9
Which features of the page would you use, and are there any options that are missing? (open question)	There are all features I need on a detail page. There would even be enough space to have a paragraph next to it describing a little more details.
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend this Application to every company which manages its own databases as it unveils possible risks for the database.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	I would prefer a PC or Laptop, as there is the most screen real estate to plow through the information provided.
Any other remarks? (open question)	

External Home Page – Client Details Generated Data	
Do you like the interface, is the page easy to understand? (1-10)	10
Which features of the page would you use, and are there any options that are missing? (open question)	Every feature is easily accessible as there was also made use of color-coding.
How satisfied are you with the available features? (1-10)	10
How likely are you to refer to this web application? Why or why not? (open question)	I would recommend this Application to every company which manages its own databases as it unveils possible risks for the database.
On which device would you use the application and why? (PC / Laptop, iPad or iPhone)	I would prefer a PC or Laptop, as there is the most screen real estate to plow through the information provided.
Any other remarks? (open question)	

The testing of the prototype and the questionaries have been presented. The results of the testing and the research will be discussed in the next chapter.

8. Discussion of the Results

The results of the proposed model have been presented, including the implemented prototype and the testing that has been done. During the requirements analysis, multiple requirements have been defined, which the proposed model aims to fulfil.

The prototype has been able to audit the database and fulfil the defined controls. The first control which has been implemented is checking if the tested database has a supported software version. The control is currently implemented for the relational database MariaDB. To enable this functionality for another database software the prototype needs to be expanded for the last supported version of the database software.

The check of the least privilege implementation and the search for users with administrator rights have also been successful. The prototype can detect user groups and using the users` titles to detect if they have appropriate rights – for example if a trainee or junior seniority level employee has administrator user rights, that will be documented as a risk.

The password policy check was also able to detect the previously defined risk factors. If the passwords in the database are not encrypted the control will be able to detect the defined risks.

The detection of unexpected interruptions, backups & restorations of the database, and changes have been successful. If the logs have been correctly added to the database the prototype will be able to track them, providing a chain of custody for these controls, with the help of the data which has been collected in the ticket system. The prototype has been tested with the Atlassian Jira ticket system. The possibility to implement the communication with other ticket systems is also possible if they provide the needed APIs which the prototype needs to collect the tickets saved in the ticket system.

The model has also addressed multiple characteristics as they have been defined by COBIT. The fulfilled characteristics have been availability, compliance, reliability, and confidentiality. The other characteristics can also be addressed in future work on the model, in the case the prototype gains access to the applications which are using the enterprise`s database. For example, to prove the integrity of the database, on the

output of an application sending data to the database is needed, so that it can be compared with the data saved in the database.

The chain of custody has also been addressed, with documenting the log events in the ticket system. The prototype has been able to load which person was responsible for certain actions, which state they are currently in when did the action start, and the communication regarding a specific event, from the comments field in the ticket system.

The prototype has been able to detect potential risks and to present them to the auditors using a modified version of the balanced scorecards, where the results for each control have been shown. The prototype was also able to collect the needed proof for each of these controls and to provide them to the auditor in the form of a PDF file.

As described in the previous chapter, scenario-based testing of the prototype has been done, and the detected bugs have been fixed. A questionnaire has been done with multiple auditors and a web designer, where the functionality and the usability of the prototype have been tested.

The questionnaire started with the admin home page. The first question was if the testers liked the interface and if it was easy to understand. From five testers all five have rated the page with a 10, meaning that it was well done and easy to understand. The second question regarding this page was regarding the features of the page and which of the provided features they would use. The testers answered that all the needed functionalities on this page have been provided. There was only one suggestion regarding the naming of the “Get User” functionality, that it should be renamed in the future work, for a clearer name describing the action (so that it is easier for the end-user to understand that the details of a specific user are being loaded and that they can be edited by the admin). The testers also answered the next question, that they would recommend the application, because of the clean and simple design, and because it would make the communication between the different users easier and faster. On the question on which device, they would use the application, interestingly all testers have answered on the pc/laptop. The user interface for the pc version of the application has been also graded with the highest grade of the three versions, with a

9.2. The grades and comments for the admin user details page have been similar, with mostly the same grades.

The internal auditor home page has been graded with a grade above 9, and no improvements have been recommended. Comment has been made regarding the balanced scorecard, saying that the colors have been used well for presenting the different types of issues. Also, interesting for this page was that it got the lowest grades for the iPad version (a tester has given the grade 5) since the visibility of the balanced scorecard is not as clear as on the pc version. The download functionality of the balanced scorecard has gotten multiple suggestions, where 3 out of the 5 testers suggested an improved download mechanism. They recommended that the application should in the future enable Excel and CSV formatted downloads of the balanced scorecards, which would also improve the reusability. Regarding the configuration page, only one remark has been made, that the testers would use this page only on the PC version of the prototype, since many fields need to be filled out, and data has to be uploaded, which would be complicated on a smartphone or tablet.

The external auditor pages have gotten higher grades than the internal auditor pages (but with the same remarks regarding the download of the files). The functionality to list all the sent audit requests has been mentioned as very useful.

The results of the testing have shown that the prototype fulfilled the defined requirements. There is still room for future work, with enabling the database audit for other ticket systems and database software. The questionnaire also showed that the user interface for the tablet version should be improved and that the descriptions in a few cases could be more detailed.

In the next chapter, the major conclusions will be summarized.

9. Conclusion

The thesis has presented a configuration approach for database systems. The goal of the research was to propose a model that would enable the database audit for relational databases and to implement a working prototype.

The research of the related work has shown that there is no software solution, to the best of my knowledge, which would be able to audit relational databases as proposed. There have been mentions of a few commercial solutions, but they can only audit a specific database software. Also, the COBIT standard has been identified as the guideline for this research since it's a global standard in this field. Also, a few other research papers have been mentioned as interesting approaches for database auditing.

An overview of the research approach has also been given, presenting the motivation behind the research in this field, and the steps which have been done during the research.

A requirements analysis has been done, where the controls which the proposed model aimed to fulfill have been shown. Besides the COBIT defined controls, a few other best practices have been implemented, for example, the check of the database software – if it is running under a supported version, or the aim to implement a chain of custody. Also, the required software architecture has been defined, which was needed to implement the required controls and to provide a good user interface and user experience for the internal and external IT auditors. A use case diagram has been presented, showing the different types of users that need to be implemented, and the functionalities which they need.

The user types which have been defined are the application administrator (admin), the internal auditor, and the external auditor. The admin needed to be able to handle the creation, editing, and removal of the two other types of users. The internal auditor needed to be able to add the defined configuration, which would enable the database audit. Also, to edit the configuration in case of changes. After the config had been added, the internal auditor needed to be able to audit his database. The internal auditor needed the functionality to download the balanced scorecard and the collected proof during the database audit. He needed the possibility to have an overview of all previously done audits and to get an overview of the audit requests sent by external

auditors. He also needed the possibility to change his password. In the case of issues during database audits, the external auditors needed the possibility to notify him about the issue, which he needs to fix in such cases. For that reason, he needed the functionality to get a list of all pending issue requests. The external auditor needed the functionality to send audit requests to the internal auditor. He needed to see the list of all the sent audit requests and to audit their databases. It was also important that he can get the list of all the audits which he has done previously and notify the internal auditor in the case of issues during the database audit.

The need for a flexible web application has been identified, and a deployment diagram has been shown, describing the needed architecture which would enable the development of the prototype. A test enterprise infrastructure needed to be created including a database and a ticket system, filled with test data. The application needed to consist of a backend that would handle the defined controls, a frontend, which would be resizable and usable on multiple different devices, and a database where the data would be saved.

A model has been proposed which was aiming to fulfill the defined controls and characteristics. For each control BPMN diagrams have been shown, describing all possible cases that the prototype should cover. A way on establishing the chain of custody has also been defined. The structure of the balanced scorecard has been described, mentioning the possible risk levels. The need for a full report (proof) has also been described, and which content it should contain.

After the requirements, and the model aiming to fulfill these, have been defined, the prototype has been developed. The development of the prototype consisted of multiple steps. First, the test architecture needed to be set up, including filling the test database and ticket system with test data.

The backend has been developed using TypeScript and Node, with a list of APIs which have enabled communication with the frontend. A class diagram has been shown, documenting the structure of the test database, also showing the classes and attributes of the prototype database. Communication with the example database and ticket has been established, with which the controls have been tested. A frontend has also been created, using React and TypeScript. For the style basis, react-bootstrap has been used, to make the application usable on multiple different devices. The user

stories for each of the three different users have been shown, showing the database audit generation, the result formatted as a balanced scorecard, and the possibility to download the balanced scorecard and the proof collected during the database audit. After the prototype has been implemented, scenario-based testing has been done.

All the controls have been tested with test data from the example database and ticket system. Also, all the pages which have been implemented have been tested multiple times, for possible bugs. After that, a questionnaire has been sent to multiple auditors and a web designer to test the usability and functionality of the application.

After the testing has concluded, a discussion of the results of the research and the testing has been presented. The testing showed that the controls have provided the expected results. An interesting result was that all of the testers stated that they would use the application mostly on the pc/laptop. A few recommendations have been given for possible improvements in the future.

The research has provided a model for auditing relational databases and has been proved on the test example, auditing a MariaDB database. The model has room for further research and development of the prototype. In the future, the database audit of other relational databases could be implemented, also fulfilling the other characteristics, which have been defined by the COBIT framework (for example the integrity, etc.). The test results showed that the tablet version of the user interface could be improved, with also more detailed descriptions of the provided functionalities in some cases.

Solutions such as the one described in this master thesis, can improve the security of an enterprise`s databases and help it avoid possible risks.

References

[1] ITIL, “Internal Audit Report,” 2018. https://www.itil-docs.com/itil-internal-audit-report-template/#:~:text=An ITIL audit focuses on,providing seamless continuous IT services. (accessed Mar. 31, 2021).

[2] *Cobit 4.1*. The IT Governance Institute, 2007.

[3] I. Rus, “Technologies and Methods for Auditing Databases,” *Procedia Econ. Financ.*, vol. 26, no. 15, pp. 991–999, 2015.

[4] J. Paddock, “Database Security & Auditing,” pp. 1–48, 2009.

[5] MariaDB, “Introduction to Relational Databases,” *2016*. https://mariadb.com/kb/en/introduction-to-relational-databases/ (accessed Mar. 31, 2021).

[6] D. A. Flores and A. Jhumka, “Implementing chain of custody requirements in database audit records for forensic purposes,” *Proc. - 16th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. 11th IEEE Int. Conf. Big Data Sci. Eng. 14th IEEE Int. Conf. Embed. Softw. Syst.*, pp. 675–682, 2017.

[7] R. Barnes, “Database Auditing : Best Practices,” 2009.

[8] L. Liu and Q. Huang, “A framework for database auditing,” *ICCIT 2009 - 4th Int. Conf. Comput. Sci. Converg. Inf. Technol.*, pp. 982–986, 2009.

[9] R. A. Hevner, T. S. March, J. Park, and S. Ram, “DESIGN SCIENCE IN INFORMATION SYSTEMS RESEARCH,” *MIS Q.*, vol. 28, no. 1, pp. 75–105, 2004.

[10] Statista, “Global market share held by operating systems,” 2020. https://www.statista.com/statistics/268237/global-market-share-held-by-operating-systems-since-2009/ (accessed Mar. 31, 2021).

[11] M. D. and K. G., “Angular vs. React: What to Choose for Your Web App?,” 2020. https://www.cleveroad.com/blog/angular-vs-react#:~:text=Angular vs.-,React%3A Popularity Among Developers,30.7%25 Angular. (accessed Mar. 31, 2021).

[12] E. U. Institute, “Strong Password Policy,” 2019. https://www.eui.eu/ServicesAndAdmin/ComputingService/PolicyDocuments/StrongPasswordPolicy (accessed Mar. 31, 2021).

[13] Mocharoo, “Realistic Data Generator,” 2020. https://www.mockaroo.com/1920d500 (accessed Mar. 31, 2021).

[14] A. Jira, “JIRA Developer Documentation : JIRA REST API Example - OAuth authentication,” 2019. https://developer.atlassian.com/server/jira/platform/jira-rest-api-example-oauth-authentication-6291692/?_ga=2.233771614.1576268108.1589960996-

216283740.1586275712 (accessed Mar. 31, 2021).

[15] OpenJS Foundation, “Express - NodeJS Web Framework,” 2019. http://expressjs.com/ (accessed Mar. 31, 2021).

[16] TypeORM, “TypeORM Documentation,” 2021. https://typeorm.io/#/entities (accessed Mar. 31, 2021).

[17] “MariaDB NPM Package v2.5.3,” *2021*. https://www.npmjs.com/package/mariadb (accessed Mar. 31, 2021).

[18] “MariaDB Server Releases,” *2021*. https://mariadb.com/kb/en/mariadb-server/ (accessed Mar. 31, 2021).

[19] Remotedevio, “Redux DevTools,” 2018. https://chrome.google.com/webstore/detail/redux-devtools/lmhkpmbekcpmknklioeibfkpmmfibljd?hl=en (accessed Mar. 31, 2021).

Figures

[Figure 1] Rifatbegovic Nedim, “Use Case Diagram – Showing the Three Different User Types”, 2021
[Figure 2] Rifatbegovic Nedim, “Deployment Diagram – Showing the Configuration of the Prototype and the Test Infrastructure”, 2021
[Figure 3] Rifatbegovic Nedim, “Component Diagram - Presenting the Architecture of the Model”, 2021
[Figure 4] Rifatbegovic Nedim, “BPMN Diagram – Administrator Processes”, 2021
[Figure 5] Rifatbegovic Nedim, “BPMN Diagram – Internal Auditor, Configuration Handling”, 2021
[Figure 6] Rifatbegovic Nedim, “BPMN Diagram – Internal Auditor, Auditing Database”, 2021
[Figure 7] Rifatbegovic Nedim, “BPMN Diagram – Internal Auditor, Additional Functionalities”, 2021
[Figure 8] Rifatbegovic Nedim, “BPMN Diagram – External Auditor Actions”, 2021
[Figure 9] Rifatbegovic Nedim, “BPMN Diagram – Database Version Control”, 2021
[Figure 10] Rifatbegovic Nedim, “BPMN Diagram – COBIT PO2.3 Control”, 2021
[Figure 11] Rifatbegovic Nedim, “BPMN Diagram – Least Privilege & Administrator User Rights Control”, 2021
[Figure 12] Rifatbegovic Nedim, “BPMN Diagram – COBIT DS5”, 2021
[Figure 13] Rifatbegovic Nedim, “BPMN Diagram – COBIT AC4, DS11.5, AI6.1, AI6.2”, 2021
[Figure 14] Rifatbegovic Nedim, “Class Diagram – Classes of the Application and Test Database”, 2021
[Figure 15] Rifatbegovic Nedim, “Test Ticket from Atlassian Jira”, 2021
[Figure 16] Rifatbegovic Nedim, “Redux DevTools Example”, 2021
[Figure 17] Rifatbegovic Nedim, “Home Page of the Prototype”, 2021
[Figure 18] Rifatbegovic Nedim, “Admin Home Page – PC Version”, 2021
[Figure 19] Rifatbegovic Nedim, “Admin Home Page – iPad Version”, 2021
[Figure 20] Rifatbegovic Nedim, “Admin Home Page – iPhone Version”, 2021
[Figure 21] Rifatbegovic Nedim, “Admin User Details Page”, 2021
[Figure 22] Rifatbegovic Nedim, “Internal Auditor – Home Page”, 2021
[Figure 23] Rifatbegovic Nedim, “Internal Auditor – Generated Report – PC Version”, 2021
[Figure 24] Rifatbegovic Nedim, “Internal Auditor – Generated Report – iPad Version”, 2021
[Figure 25] Rifatbegovic Nedim, “Internal Auditor – Generated Report – iPhone Version”, 2021
[Figure 26] Rifatbegovic Nedim, “Balanced Scorecard – PDF Version”, 2021
[Figure 27] Rifatbegovic Nedim, “Collected Database Audit Proof – PDF Version”, 2021
[Figure 28] Rifatbegovic Nedim, “Internal Auditor – Loaded Data from the Backend”, 2021
[Figure 29] Rifatbegovic Nedim, “Internal Auditor – Config Page – Filling out the needed Data”, 2021

[Figure 30] Rifatbegovic Nedim, “Internal Auditor – Config Page – Updating a specific Config Attribute”, 2021
[Figure 31] Rifatbegovic Nedim, “Internal Auditor – Changing Password”, 2021
[Figure 32] Rifatbegovic Nedim, “External Auditor – Home Page”, 2021
[Figure 33] Rifatbegovic Nedim, “External Auditor – List of Loaded Audit Requests”, 2021
[Figure 34] Rifatbegovic Nedim, “External Auditor – Audit Details Page”, 2021
[Figure 35] Rifatbegovic Nedim, “External Auditor – Audit Details Page – Data Loaded”, 2021

Schriftliche Versicherung

Ich versichere, dass die Arbeit von mir selbstständig verfasst wurde und dass ich keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Weiterhin wurde diese Arbeit keiner anderen Prüfungsbehörde übergeben.

Wien, 29. März, 2021

Thesis Affirmation

I affirm that this thesis was written by myself without any unauthorized third-party support. All used references and resources are indicated. All quotes and citations are properly referenced. This thesis was never presented in the past in the same or similar form to any examination board.

Vienna, 29. March 2021