TARTU UNIVERSITY
FACULTY OF SOCIAL SCIENCES

NARVA COLLEGE
STUDY PROGRAM “INFORMATION SYSTEMS DEVELOPMENT “

Igor Tjantov

DEVELOPING ANDROID AND WEB APPLICATIONS FOR SENDING INVOICES

Diploma Project

Supervisor: Andre Sääsk, M.Sc.

NARVA 2022

Olen koostanud töö̈ iseseisvalt. Kõik töö̈ koostamisel kasutatud teiste autorite tööd, põhimõttelised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

.....

Töö̈ autori allkiri ja kuupäev

Non-exclusive license to reproduce thesis

I, Igor Tjantov (date of birth: 06.03.1987),

1. herewith grant the University of Tartu a free permit (non-exclusive license) to:

1.1. reproduce, for the purpose of preservation and making available to the public, including for additional to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright.

"Developing android and web applications for sending invoices" supervised by M.Sc. Andre Sääsk,

2. I am aware of the fact that the author retains the right referred to in point 1.

3. This is to certify that granting the non-exclusive license does not infringe the intellectual property rights or rights arising from the Personal Data Act.

Narva, 10.06.2022

Contents

LIST OF TERMS AND ABBREVIATIONS

C# – object-oriented programming language.

APP – mobile application.

WEB – internet space.

DB – database.

SQL – structured query language.

MYSQL – relational database management system.

ASP.NET – Microsoft web framework.

MVC – ’model-view-controller’ is software design pattern.

AWS – Amazon web services

AWS RDS – Amazon relational database service.

API – application programming interface.

ID – identifier.

AZURE – Microsoft cloud service.

IDE – integrated development environment.

1. INTRODUCTION

1.1 PROBLEM

In many small businesses that rent out various premises, there is a problem with how the tenant billing work is not optimally arranged. Billing is for rent and utilities (if any.). Often, an individual employee records the data from the meters and passes it to the accountant, who manually edits the invoice template in Excel and sends it to the tenant. Sometimes in small enterprises, the owner of the enterprise himself performs this task, since he does not have other employees or a full-time accountant. There are already various solutions on the Estonian market that allow to issue invoices, but in them this procedure takes time, additional training may be required to use and extra functions that are not necessary. These same businesses want an application that simplifies this process and, with minimal worker intervention, generates an invoice, sends it, and stores it in a database.

This project is an order from a customer who rents out premises and wants to simplify the billing procedure as much as possible.

1.2 SOLUTION

The solution to the problem is to study existing solutions, determine the specific wishes of the customer and develop an application that will be easier and faster to work with, unlike those that are already on the market.

1.3 PROJECT GOAL

The goal of the project is to create a smartphone application that can quickly create and send an invoice to a tenant. Create a web application that also allows the user to create and send invoices, as well as download them and manage the necessary tenants' data. Create a database that will store the necessary information for the application to work, as well as the history of invoices already issued. Applications should be as simple as possible and require minimal intervention in the work.

1.4 PROJECT TASKS

To achieve this goal, we need to complete the following tasks:

- Conduct research to find out what billing applications already exist.

- Determine the pros and cons of existing applications.
- Determine the tools necessary for application development.
- Study the tools necessary for application development.
- Develop a database for the application.
- Develop a web application.
- Develop a mobile application.
- Develop an API that will connect web application and mobile application.
- Test applications.

1.5 OUTLINE

The first chapter is an introduction. In the first chapter, the author describes the existing problem and proposes a solution.

The second chapter is an overview of existing solutions, identification of their disadvantages.

In the third chapter, the author lists the technologies and tools used in the work and explains why these technologies were chosen for the task. The author describes the system development process.

In the fourth chapter, the author will list the functionality implemented in the system and the functionality that the author was unable to implement.

In the fifth chapter, the author will list the functionality implemented in the system and the functionality that the author was unable to implement. Also, the author describes the future development of the application.

2. EXISTING SOLUTIONS

In this chapter the author overviews existing solutions in Estonia, making identification of their disadvantages and comparison with the author's idea.

There are many mobile billing applications, but they are all foreign and not represented in Estonia. Their principle is the same as that of all Estonian billing applications – manual billing with the addition of a lot of data.

There are several web applications of the same type in Estonia.

The most common billing applications:

- Merit Aktiva
- Isolta
- Minuarved.ee

2.1 Merit Aktiva

Merit Aktiva is the most popular app in Estonia. It is intended for entrepreneurs and accountants. As advertised by Merit, it is easy to learn, simple to use, and its capabilities are sufficient for the most demanding professionals. Thanks to cloud technology, it is very easy to share work between an accountant and other specialists. Merit Aktiva is a comprehensive solution for enterprise accounting.

Advantages:

- Web application with cloud storage
- User-friendly interface
- Multilingual interface
- Huge functionality:
    - Sales and purchase invoices
    - Financial accounting
    - Stock records
    - sending VAT reports
    - and so on (total 30 functions).

Disadvantages:

- Overloaded interface
- Training for use is needed
- Creating invoices takes time

After a simple registration, the user gets to the main page (**Figure 1**) of the application, which contains many tables. To issue an invoice, you need to click on the "invoices" button in the "sales" menu. On the new page, the user needs to click on the "New invoice" button and on the invoice page (**Figure 2**) that opens, manually add all the account data. Customer data is saved and automatically added with a new invoice, but the service for which the invoice is issued must be added manually each time.

After the invoice has been created and saved, the user can download it by clicking on the "PDF" button or send it by e-mail by clicking on the "MAIL" button on the invoice overview page (**Figure 3**).

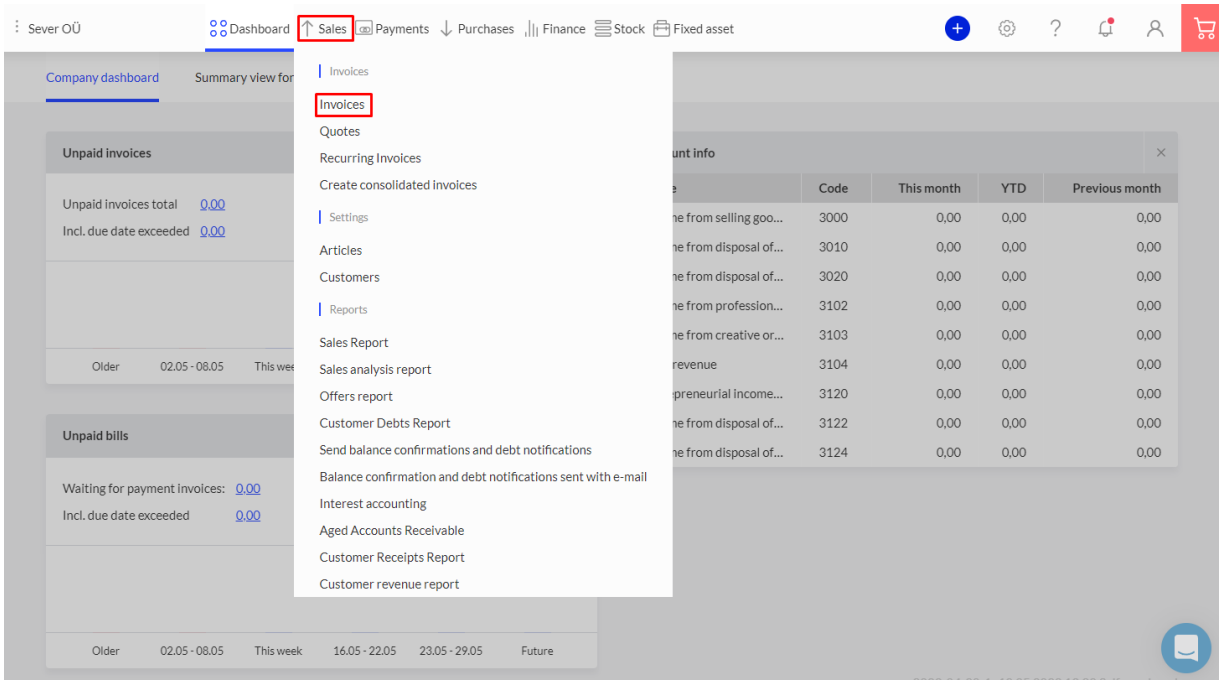

Figure 1. *Merit Aktiva main page (Source: Author)*

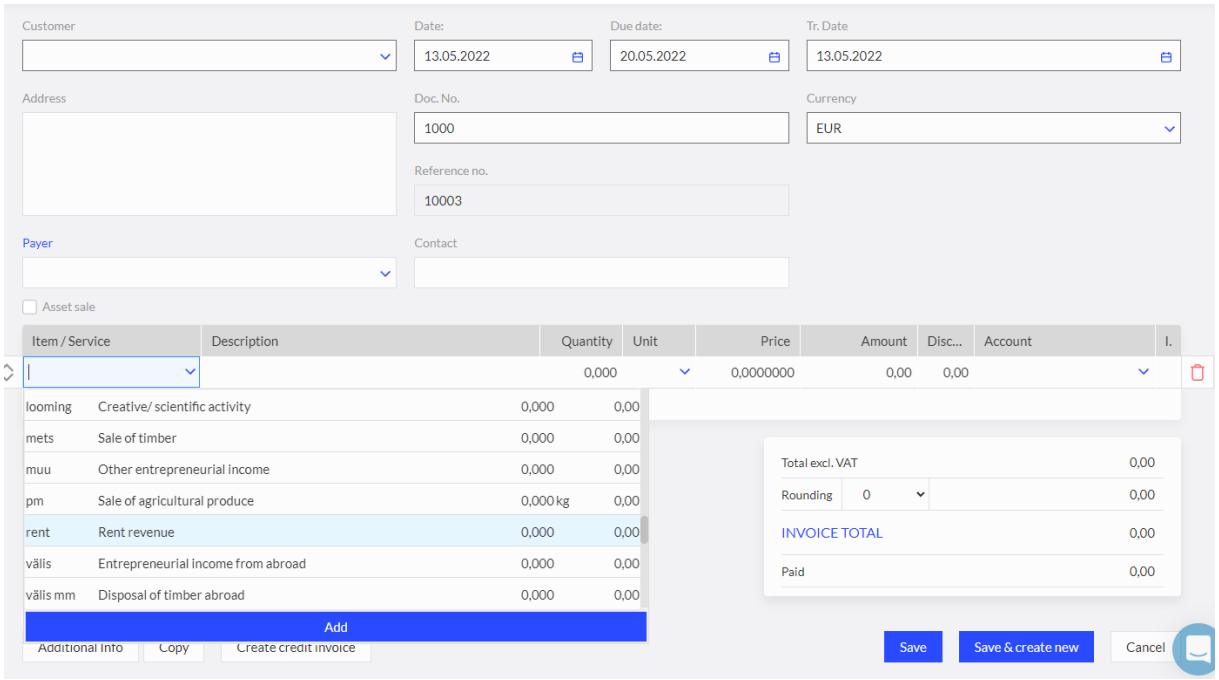

Figure 2. *Merit Aktiva invoice page (Source: Author)*

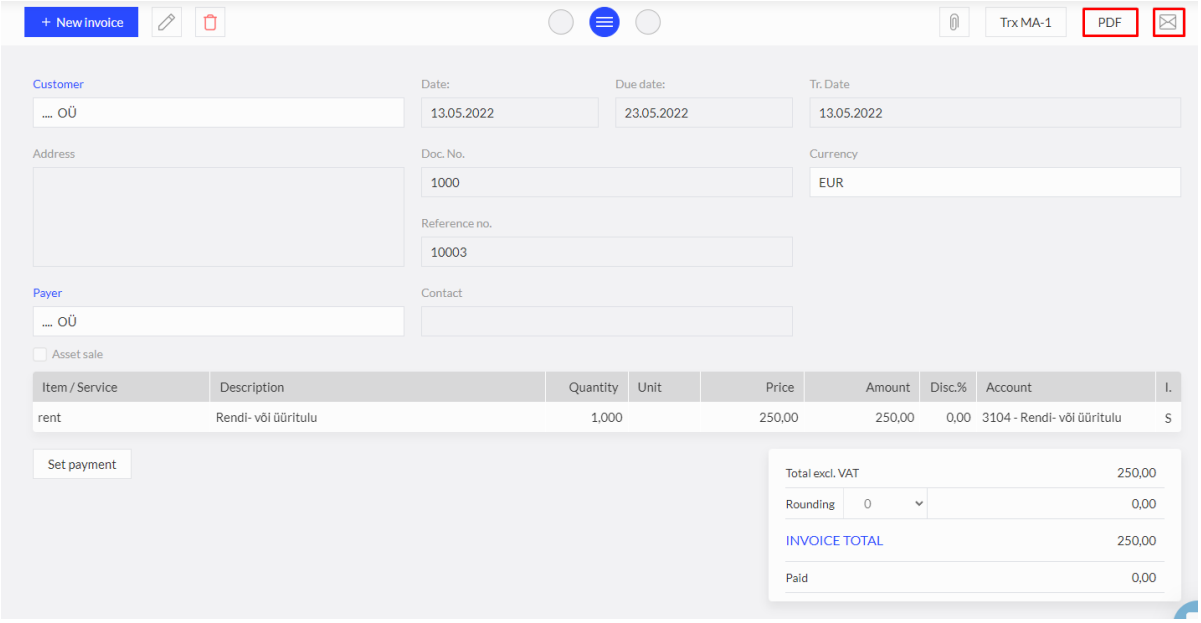

Figure 3. *Merit Aktiva invoice* overview *page (Source: Author)*

Result:

Merit Aktiva is a very powerful application for solving all accounting tasks. It has many functions that are not needed in the requirements of the project customer. The billing procedure takes time and becomes complicated if there are several tenants. Sending an invoice is also done manually (by pressing a separate button).

2.2 Isolta

Isolta is an invoicing-only web application. In this application, you can create an invoice and send it, create a list of all clients, a list of services, view invoice history and download a history report.

Advantages:

- Web application with cloud storage
- User-friendly interface
- Multilingual interface
- Easy to use

Disadvantages:

- Creating invoices takes time
- Can't create utility invoice

An invoice can be easily created by clicking on the "New Invoice" button on the main page (**Figure 4**). On the invoice page (**Figure 5**), need to enter the tenant and service details manually if they have not been created in advance. If the service (rent) was created earlier, then when creating an invoice, it still needs to add additional data, for example, the month for which tenant need to pay the rent for the premises. But creating an informative utility invoice in this application is not possible, since in such an invoice need to indicate the old and new meter readings.

After the user can download invoice by clicking on the "Create a PDF and print" button or send it by e-mail by clicking on the "Send via email service" button on the invoice overview page.

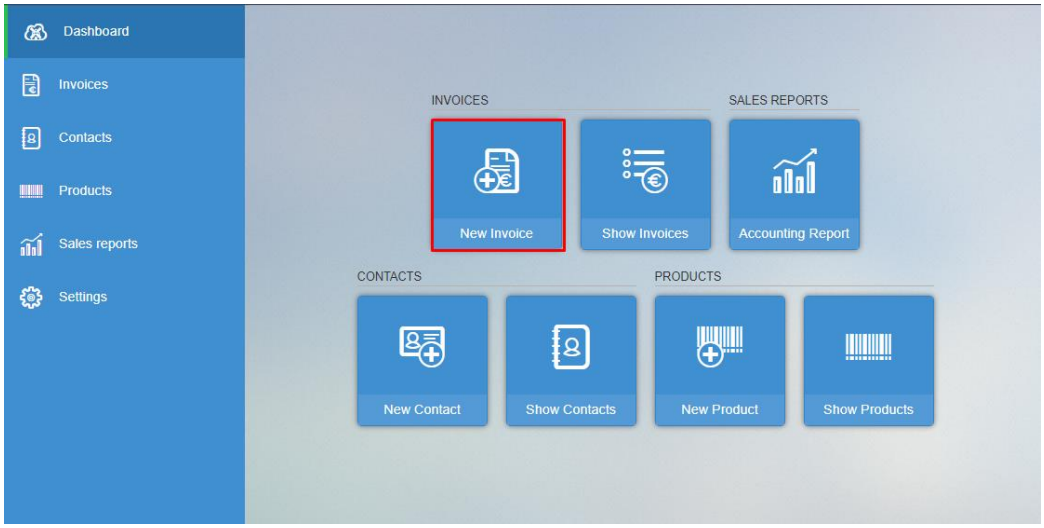

Figure 4. *Isolta main page (Source: Author)*

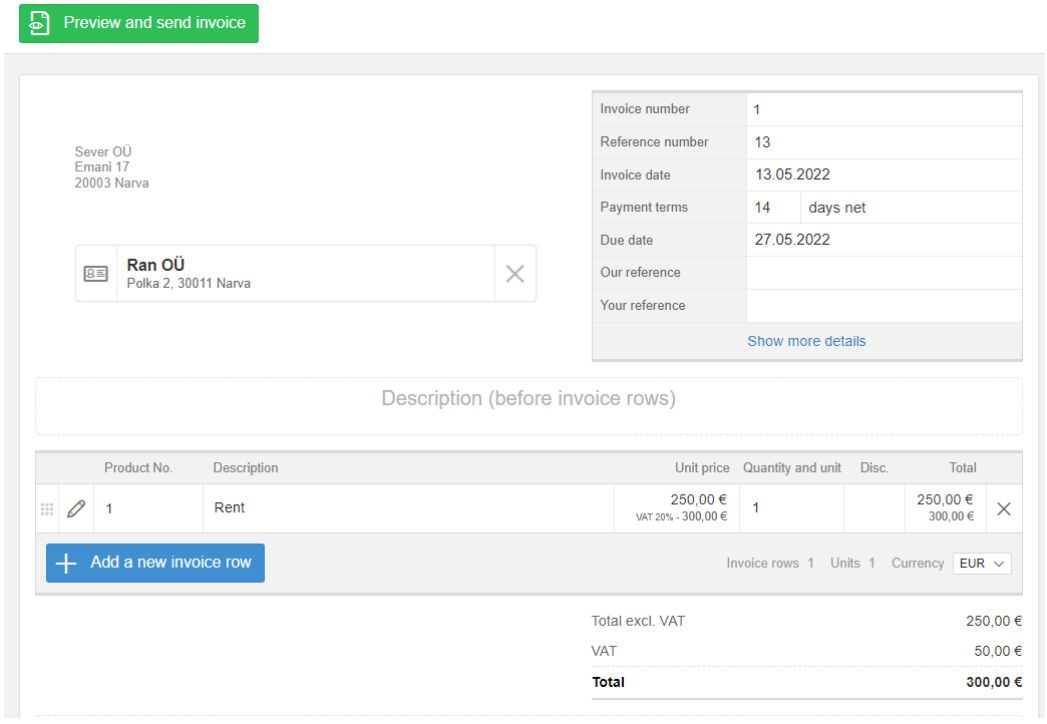

Figure 5. *Isolta invoice page (Source: Author)*

Result:

Isolta is a very simple application for creating and sending invoices. But still the billing procedure takes time and becomes complicated if there are several tenants. Main problem that creating an informative utility invoice in this application is not possible. Also, Isolta don’t have smartphone version, and this obliges the user to be at the computer.

2.3 Minuarved.ee

Minuarved is an invoice creating only web application. In this application, you can create an invoice, create a list of clients, a list of services, view invoice history. Application almost the same as Isolta.

Advantages:

- Web application with cloud storage
- User-friendly interface
- Easy to use
- Has mobile version

Disadvantages:

- Creating invoices takes time
- Dark (black) theme
- One-language interface
- Can't create utility invoice
- Invoice can’t be sent via email

An invoice can be easily created by clicking on the "Lisa uus arve" button on the main page (**Figure 6**). On the invoice page (**Figure 7**), need to enter the tenant and service details manually if they have not been created in advance. If the service (rent) was created earlier, then when creating an invoice, it still needs to add additional data, for example, the month for which tenant need to pay the rent for the premises. But creating an informative utility invoice in this application is not possible, since in such an invoice need to indicate the old and new meter readings.

After the user can download invoice by clicking on the "Lae alla PDF" button on the invoice overview page (**Figure 8**).

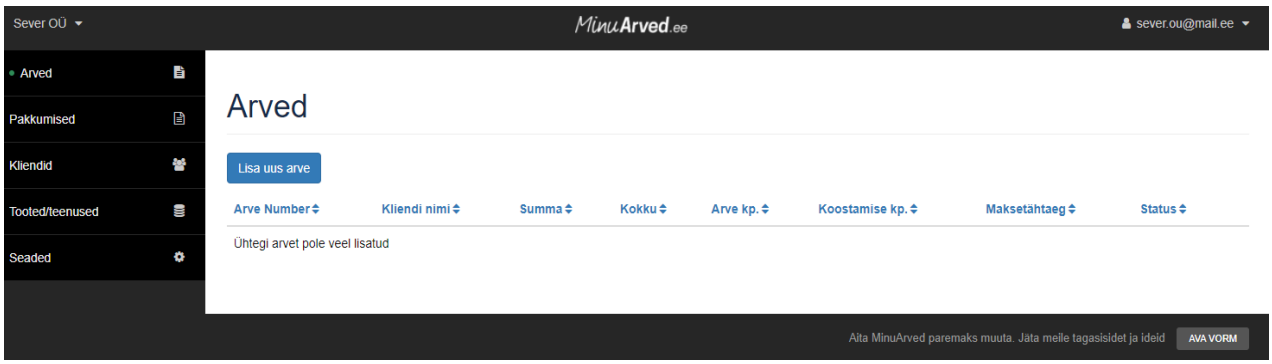

Figure 6. *Minuarved main page (Source: Author)*

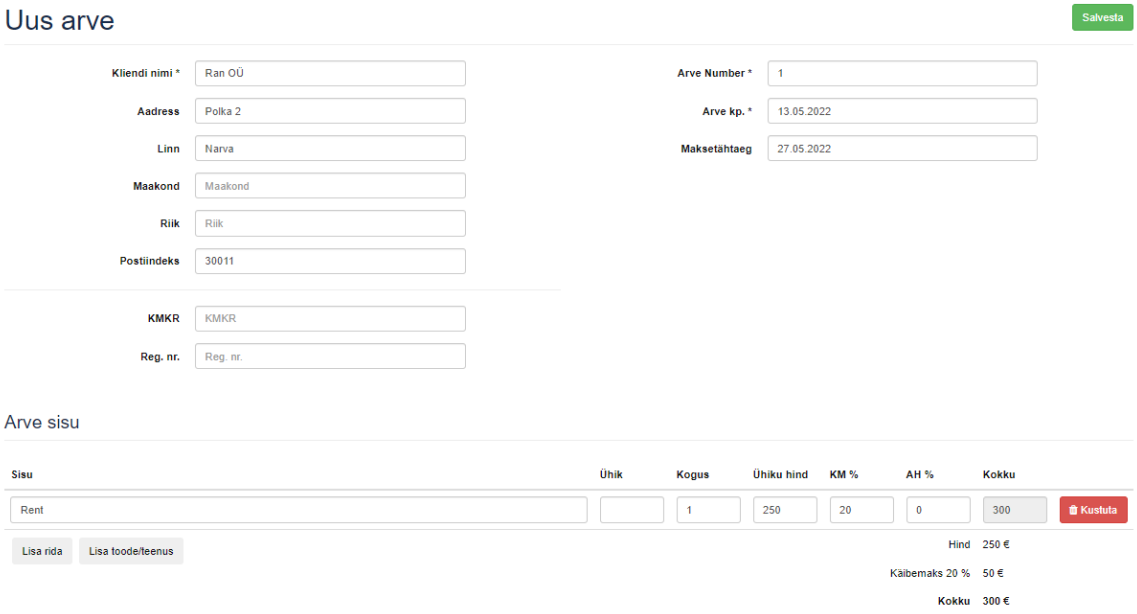

Figure 7. *Minuarved invoice page (Source: Author)*

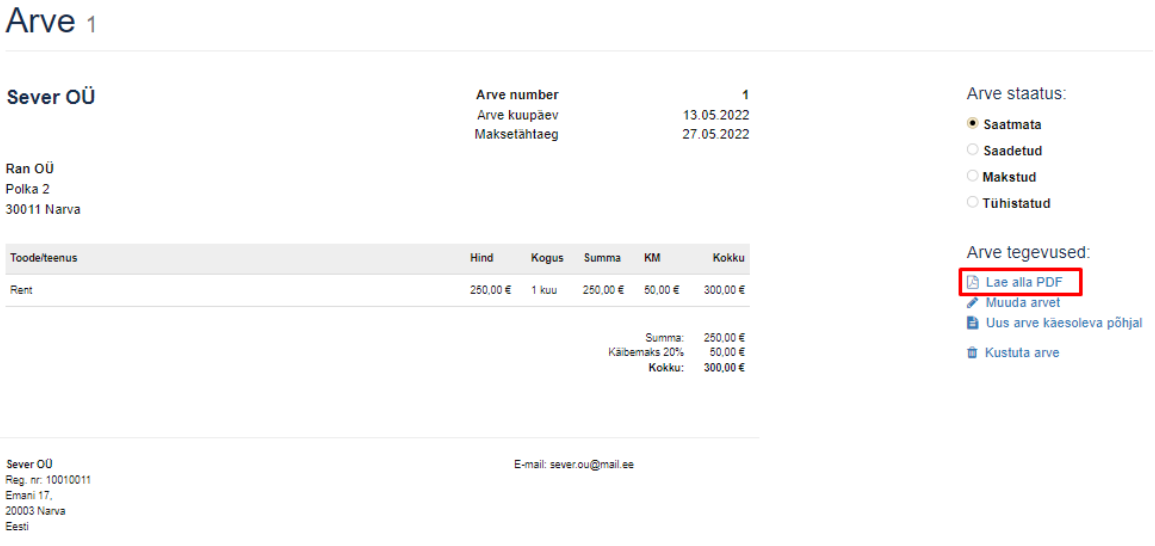

Figure 8. *Minuarved invoice overview page (Source: Author)*

Result:

The Minuarved application is very similar to Isolta in terms of how it works. A simple interface and a minimal set of functions allows to create an invoice but does not have the ability to send it by e-mail. There is also a problem with creating utility bills. The creation of each invoice requires an additional explanation (for example, the month of rent), which takes time.

2.4 Outcome of the research

After studying the most common applications for creating invoices, it became clear to the author that all of them do not meet the requirements of the customer and cannot solve the problem considered in this project to the full extent.

All existing solutions have the same main disadvantages:

- It is impossible to create a sufficiently informative invoice, which is especially important for utility costs.
- Based on the needs of the customer, manual editing is required when creating new invoice. This takes time if multiple tenants need to be billed.

3. DEVELOPMENT

In this chapter, the author describes the application development process, main features of the developed application also will be described. Also, the author of the project lists the technologies and tools used to solve the problem.

All data presented in the form of business names, addresses, numbers, and email addresses are completely fictitious. Any coincidence with the real data is completely random. Email addresses are test temporary addresses of the author of the project. And after the completion of the project, this data will be closed and deleted.

3.1 TECHNOLOGIES AND TOOLS USED IN THIS PROJECT

3.1.1 C#

C# is one of the most popular programming languages. The author chose this language for the graduation project because he knows it more than others and feels confident when working with it. The author also wants to develop his knowledge of this language and specialize in it in the future.

3.1.2 ASP.NET MVC

ASP.NET MVC is a feature-rich platform for building web applications and APIs using the Model-View-Controller design framework. It combines well both the frontend and the backend part of the application. The author has good experience with this platform.

3.1.3 Xamarin

Xamarin is a framework for cross-platform mobile application development (iOS, Android, Windows Phone) using the C# language. Since the author of the project wanted to create both a web application and an android application in the same programming language and in the same environment, he chose Xamarin. Also, if there is a need to release a version of the program for iOS, then this version will be written on this platform. Also, on Xamarin, can be use the .NET libraries familiar to the author.

3.1.4 Microsoft Visual Studio 2019

Since the author is developing an application on ASP.NET platforms (ASP.NET MVC and Xamarin), he uses the original development environment Microsoft Visual Studio.

3.1.5 MySQL

The author has experience with three different databases – MySQL, MSSQL and PostgreSQL. Since all these databases do not have big differences in structure and syntax, and for this project there is no need for a specific database, the author chose the MySQL database because he has been working with it for a

long time. The author prefers the database because it is free and does not require large hardware capacities, so it works faster on weak servers.

3.1.6 MySQL Workbench

MySQL Workbench is a graphical tool for working with MySQL servers and databases. This is used by the project author to work with a database on a remote server.

3.1.7 EPPlus

EPPlus is a .NET Framework/.NET Core library for managing XML spreadsheets. It has no dependencies to any other library such as Microsoft Excel, so Microsoft Office does not need to be installed, what makes this library preferable to the Microsoft Office Interop library built into the development environment. EPPlus can create, read, and modify XML workbooks (xlsx and xlsm), that are used in the project.

3.1.8 Spire.XLS

Spire.XLS for .NET is a Excel .NET API that can be used to create, read, write, convert, and print Excel files in any type of .NET (C#, ASP.NET, .NET Core, .NET 5.0, Xamarin) application. (https://www.e-iceblue.com/Introduce/excel-for-net-introduce.html#.YmWjOOrP2Uk, 2022).

The author uses this library to convert the edited XML file to PDF format.

3.1.9 Amazon Web Service

To deploy applications and store the database, the author considered Amazon or Azure cloud services. Since these services are identical and offer the same features, the author worked with both, and he decided to stay on the Amazon service, since there are more favorable conditions financially.

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud. On this service author holds database for the project.

Amazon Elastic Compute Cloud (Amazon EC2) is one of the Amazon Web Services that allows the user to rent a virtual server. The author uses a Windows server to deploy his ASP.NET project using Web Deploy. Web Deploy IIS web deployment tool from Microsoft that simplifies deployment of Web applications and Web sites.

3.2 STRUCTURE OF THE SYSTEM

3.2.1 Main Components

Considering other similar programs that issue invoices and which are presented in Estonia, it is possible to accurately determine the advantages of the application of the author of the project. Features that will radically set this app apart from others. The main advantage and feature that the author needs to achieve is the maximum ease of use and the minimum of actions that are required from the user.

In the application, it should be easy for the user to manage data from the database. This data can be added, changed or deleted. The application must be available on different devices and at any time, which requires its installation in the cloud. The application should be able to create a full-fledged commercial invoice at the touch of just one button and send it by e-mail to the tenant and the accountant of the user's company.

3.2.2 Use Cases

The diagram shows the use cases that are available to the user. When working with a web application, the user can:

- Manage data about tenants.
- Check billing history.
- Edit prices.
- Send invoices to tenants.

In all cases, the web application communicates with the database and makes the necessary changes.

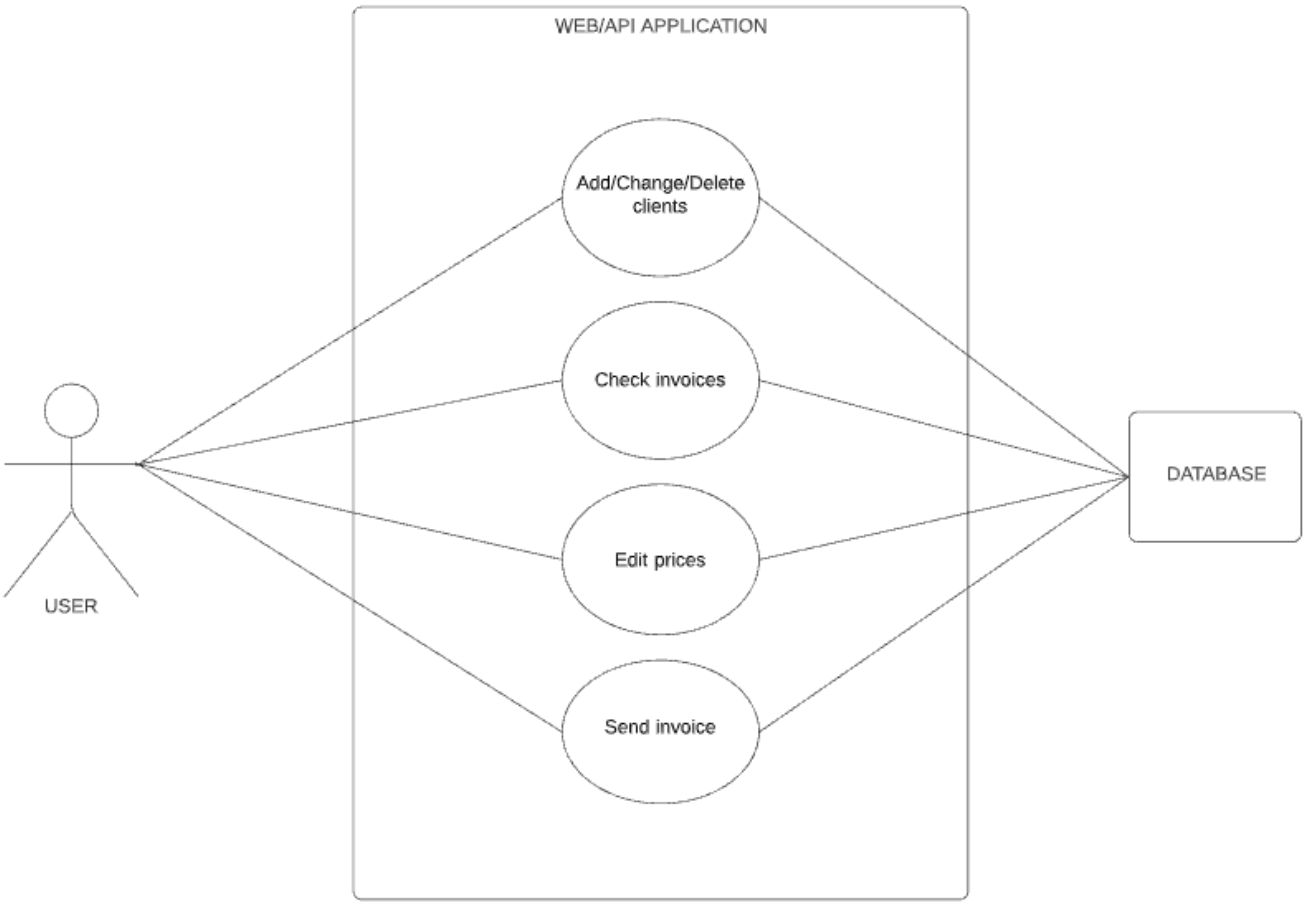

Figure 9. *Use Case diagram web application (Source: author)*

When working with an android application, the user can:

- Receive needed data about tenants.
- Send invoices to tenants.

In all cases, the android application communicates with the web application which makes the necessary changes in database.

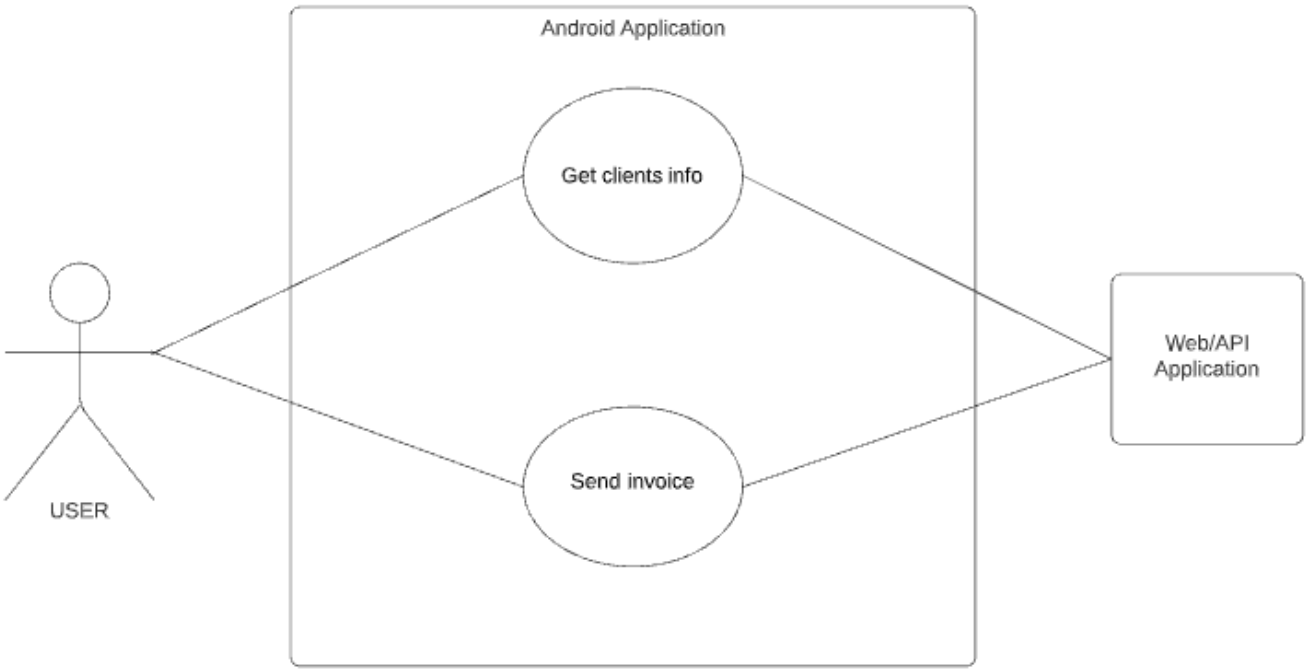

Figure 10. *Use Case diagram android application (Source: author)*

3.2.3 Functional Requirements

A functional requirement describes what a software system should do.

F1. The application should have access restriction through login form.

F2. The application should give the user a choice of what to do.

F3. The application should automatically create invoices.

F4. The application should automatically send invoices.

F5. The application must be able to communicate with API.

F6. The application must be able to communicate with database.

F7. The application should automatically update the data in the database.

F8. The application must be able to access the Internet.

3.2.4 Non-Functional Requirements

Non-functional requirements place constraints on how the system will do so.

N1. The application should work stably.

N2. The application should always be available for operation.

N3. The application should not give out critical errors.

N4. The application should have a simple and intuitive interface.

3.2.5 Dataflow

The diagram (**Figure 11**) shows how data flows between the user, the mobile application, the Web/API application, and the database.

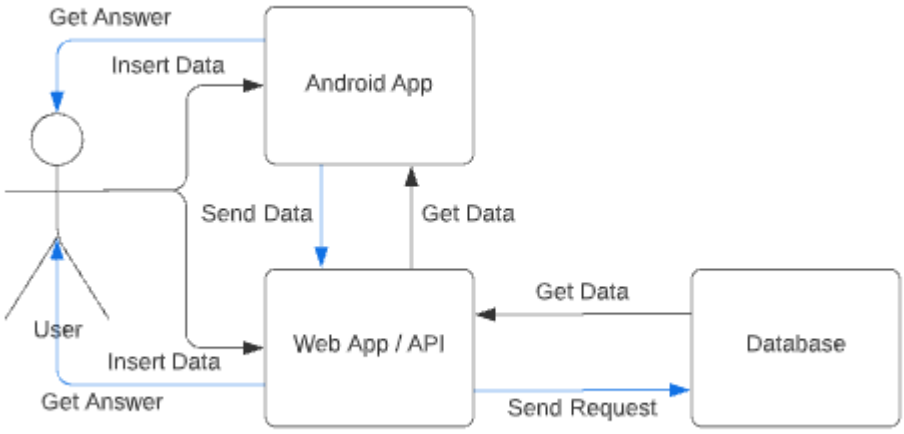

Figure 11. *DataFlow diagram (Source: author)*

In the diagram the user insert data (it can be just a request to send a rental invoice or meter readings) into mobile application or into web/API application. From the mobile application, information is transmitted to the web /API application, processed and the necessary data is transferred from the web /API application to the database. If the procedure completes successfully, the data in the database is saved, then the web /API application informs the user about it, or the mobile application, which in turn informs the user.

3.3 Database

The database contains 6 tables that are not interconnected. Since the project does not need to link data from different tables:

- Client_data
- Company_info

- Invoices
- Last_measurment
- Prices
- Users

Client_data. This table stores data about tenants. This data is key, and it is necessary for the entire operation of the mobile and web application. This data is enough to create and send a rental invoice.

The columns in this table are as follows:

- **ID** – Id of tenant in table.
- **NAME** – Name of tenant.
- **REGNR** – Commercial registration number.
- **ADDRESS** – Address of tenant.
- **EMAIL** – Tenant's email.
- **RENTPRICE** – price for rent.
- **COMMUNAL** – idendificator should the tenant pay for utilities

#	Name	Type
1	ID	int
2	NAME	varchar(80)
3	REGNR	varchar(20)
4	ADDRESS	varchar(255)
5	EMAIL	varchar(80)
6	RENTPRICE	decimal(10,2)
7	COMMUNAL	int

Figure 12. *Client_data (Source: author)*

Company_info. This table stores data about customer's company. This is a key data, and it is necessary for the entire operation of the mobile and web application. This data is enough to create and send a rental invoice.

The columns in this table are as follows:

- **ID** – Id of customer in table.
- **NAME** – Name of customer company.
- **REGNR** – Commercial registration number.
- **ADDRESS** – Address of customer company.

- **TEL** – Customer email.
- **BANK** – Bank where customer has account.
- **BACC** – Number of bank account.
- **EMAIL** – Customer's email.

#	Name	Type
1	ID	int
2	NAME	varchar(30)
3	REGNR	varchar(30)
4	ADDRESS	varchar(100)
5	TEL	varchar(30)
6	BANK	varchar(30)
7	BACC	varchar(50)
8	EMAIL	varchar(50)

Figure 13. *Company_info table (Source: author)*

Invoices. This table stores data about invoices. This data is needed so that the user can understand which invoices have already been issued. The table also stores the name of the invoice and a link to it.

The columns in this table are as follows:

- **ID** – Id of customer in table.
- **TYPE** – Type of invoice (rent or communal).
- **NO** – Invoice number.
- **DATE** – Date when invoices was created.
- **CLIENT** – Tenant's name who received the invoice.
- **PRICE** – Total price of invoice.
- **FILENAME** – Name of the invoice.
- **FILELINK** – Link of the invoice.

#	Name	Type
1	ID	int
2	TYPE	varchar(50)
3	NO	int
4	DATE	date
5	CLIENT	varchar(50)
6	PRICE	decimal(10,2)
7	FILENAME	varchar(80)
8	FILELINK	varchar(500)

Figure 14. *Invoices table (Source: author)*

Last_measurment. This table stores data about measurements. This data is needed to create new utility invoice. Each new invoice updates this data.

The columns in this table are as follows:

- **ID**
- **TANNAME**
- **ENERGY**
- **WATER**
- **LIGHT**
- **SEWER**
- **LASTUPDATE**

#	Name	Type
1	ID	int
2	TANNAME	varchar(50)
3	ENERGY	int
4	WATER	int
5	LIGHT	int
6	SEWER	int
7	LASDUPDATE	date

Figure 15. *Last_measurment table (Source: author)*

Prices. This table stores data about prices for energy, water and water sewer. This data is needed to create new utility invoice.

The columns in this table are as follows:

- **ID**
- **NAME**
- **PRICE**

#	Name	Type
1	ID	int
2	NAME	varchar(50)
3	PRICE	decimal(10,3)

Figure 16. *Prices table (Source: author)*

Users. This table stores data about users, who can use applications and manage data form database. This data is needed for log in and sending emails.

The columns in this table are as follows:

- **ID**
- **NAME – username**
- **ROLE**
- **PASS – user's password**
- **EMAIL – user email**
- **EMAILPSW – password for user email**

#	Name	Type
1	ID	int
2	NAME	varchar(50)
3	ROLE	varchar(50)
4	PASS	varchar(50)
5	EMAIL	varchar(80)
6	EMAILPSW	varchar(50)

Figure 17. *Users table (Source: author)*

3.4 Amazon EC2

The web application with integrated API application is deployed on a virtual machine running Windows operating system in the Amazon cloud service.

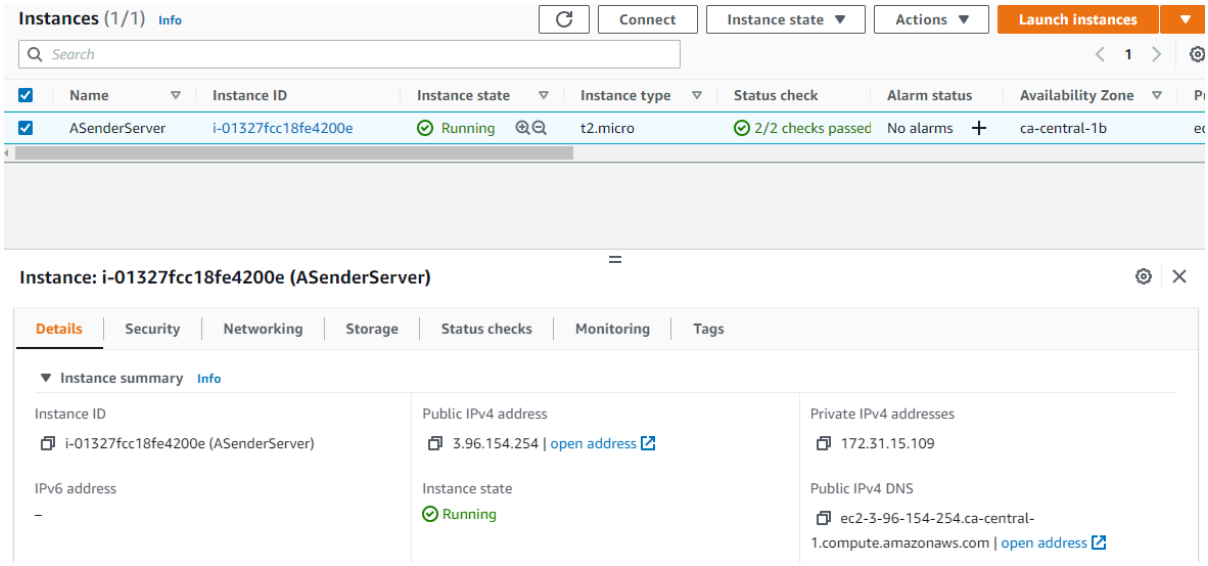

Figure 18. *Amazon EC2 control panel (Source: author)*

4.4 Amazon RDS

The author keeps his MySQL database in the Amazon RDS, the database is in the network and always available.

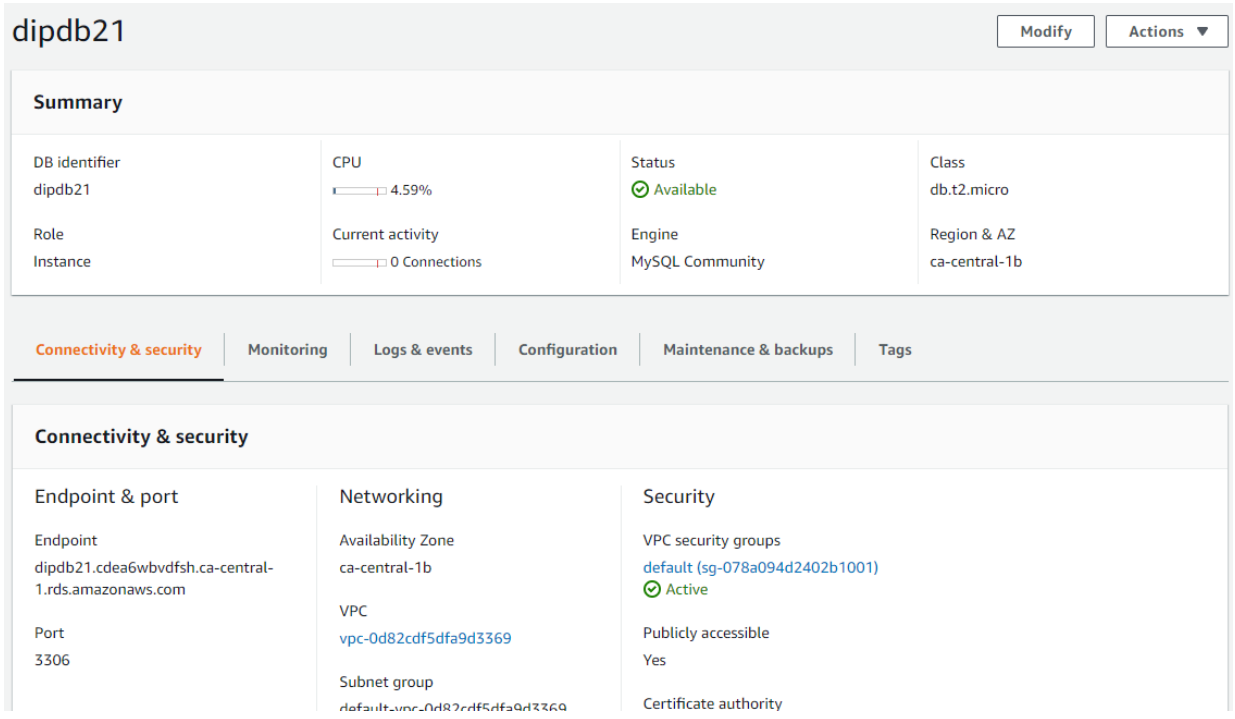

Figure 19. *Amazon RDS control panel (Source: author)*

4.5 Web Application

The application was created in the Microsoft Visual Studio 2019 development environment. Author decided to use C# because he had better experience with it.

The ASP.NET MVC framework was used to create the application. The author had good experience working on this framework, so it was chosen also because of the large amount of information and instructions in the public domain, as well as because the functionality and interface of the web application is not supposed to be complicated.

For the web application to work, is needed a constant and stable Internet connection.

The application images shown in the work are not final and will still be improved.

4.4.1 Login Page

When a user logs into a web application, he gets to the login page (**Figure 20**). On it, the user must enter their authorization data in the "Username" and "Password" fields and then click the "Login" button. If the data is incorrect, the user will see a message about it. Otherwise, it gets to the next page.

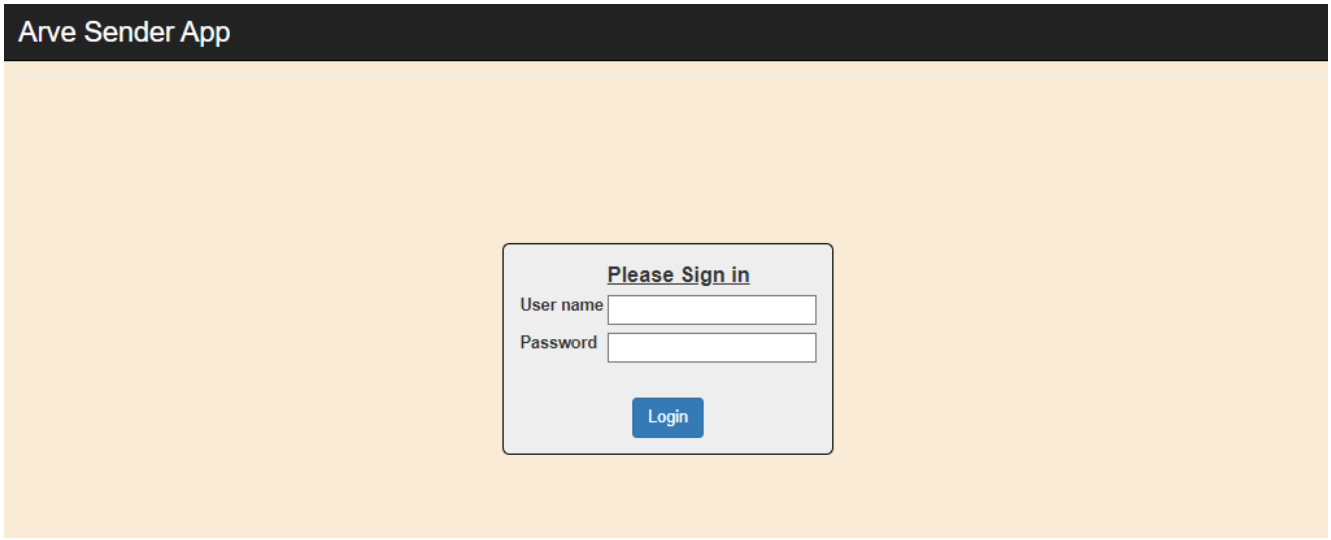

Figure 20. *Login page (Source: author)*

4.4.2 Home Page

On the next page, which is called the "Home" page (**Figure 21**), the user will see several elements with brief information about what can be done in this web application. The user chooses what he wants to do and presses the appropriate button.

- On this page, users choose what to do next:
- He can simply leave the page and end the session by clicking on the "Log out" button.

- Can get information about their tenants. Change them, delete them, or add new ones.
- Can view information about utility prices and change them.
- Can view the history of sent invoices and download them.
- Can create and send an invoice by clicking on the "Send" button.

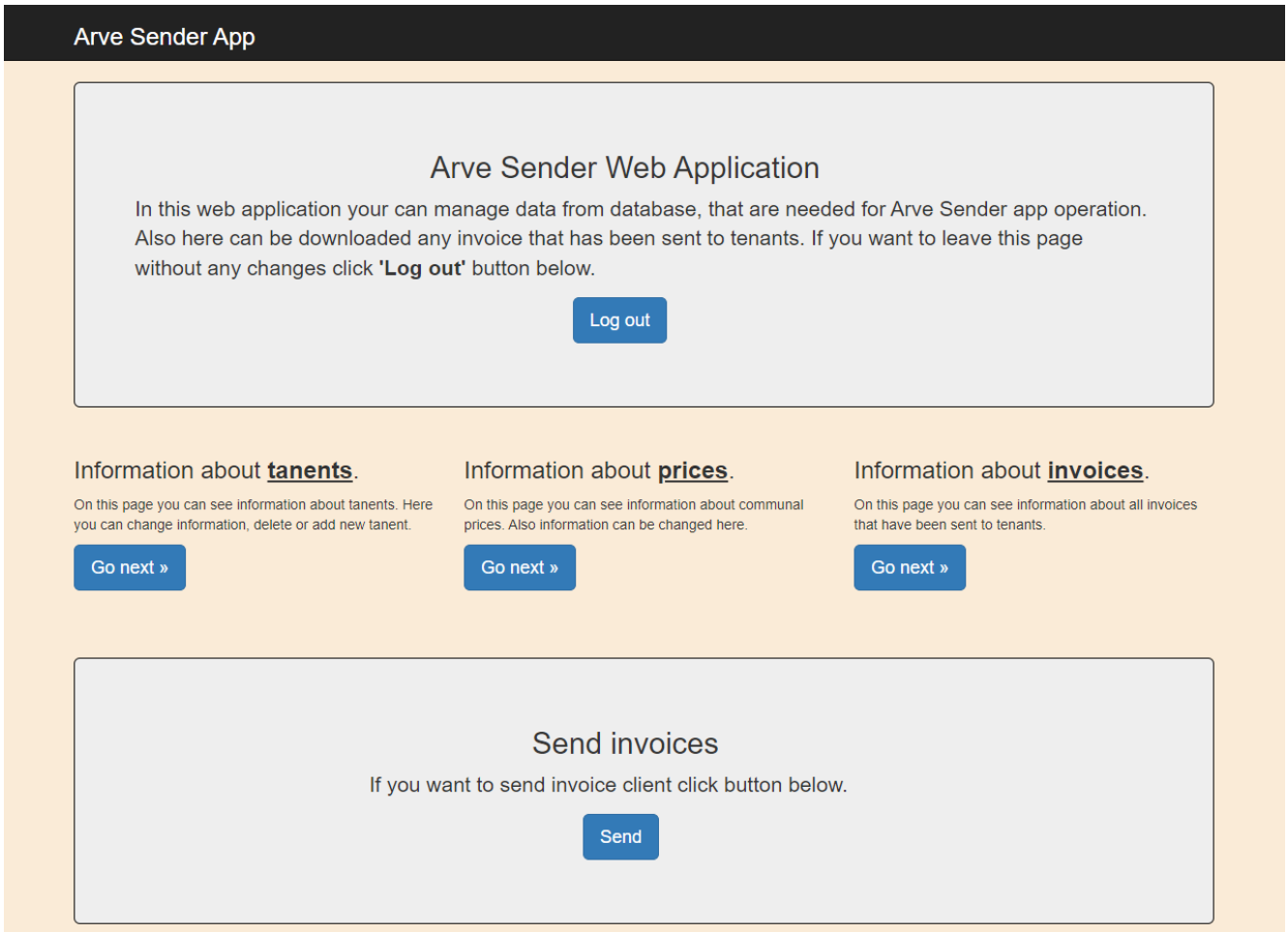

Figure 21. *Home page (Source: author)*

4.4.3 Clients Page

On the clients' page (**Figure 22**), the user sees which tenants he currently has. The data that is displayed:

- Name
- Registration Number
- Address
- Email
- RentPrice

On this page user can manage clients' data: change them, delete them, or add new ones. The user only needs to click the corresponding button or the "back" button or does not want to do anything.

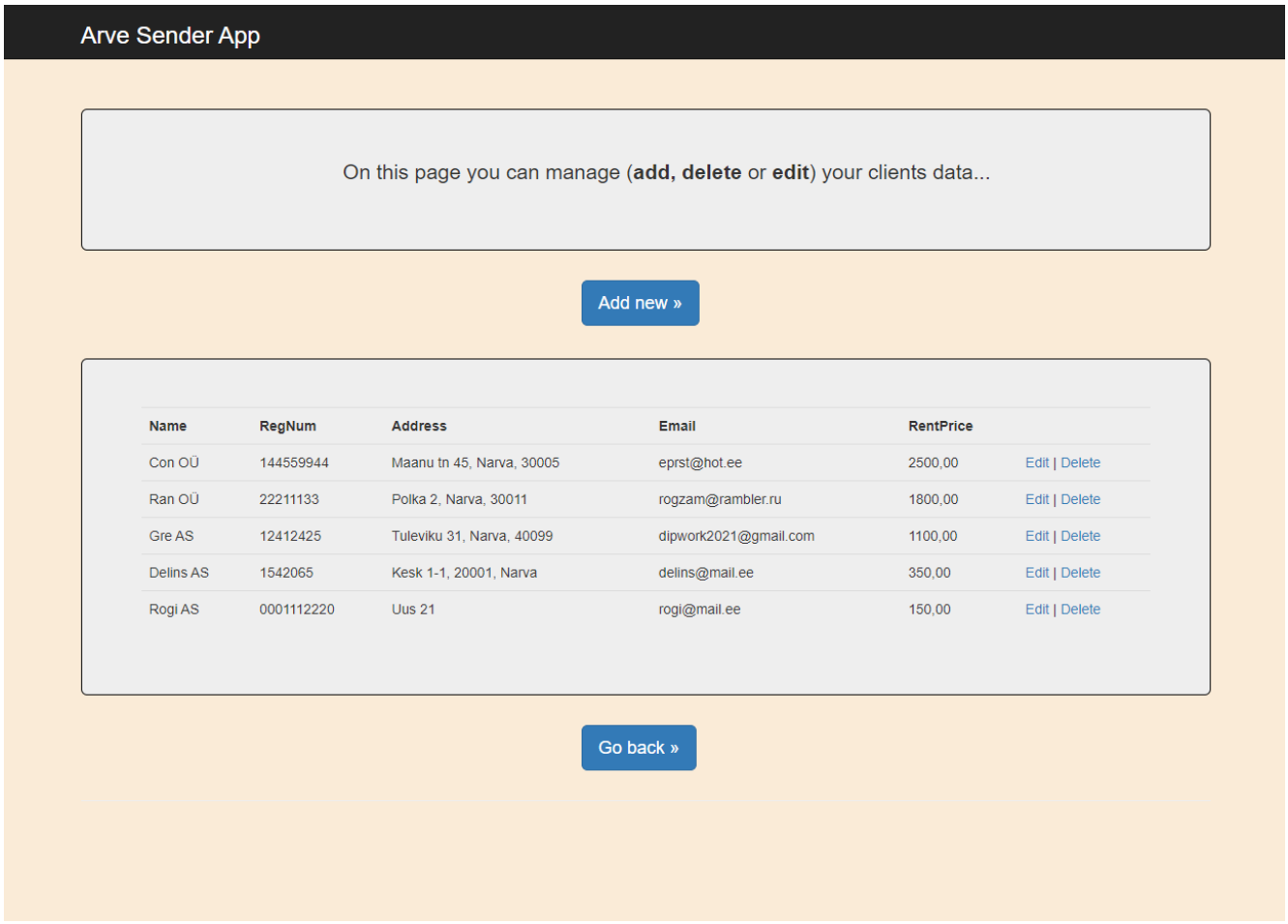

Figure 22. *Clients page (Source: author)*

4.4.4 Add new client Page

After clicking the "Add new" button, the user gets to the page of adding a new tenant (**Figure 23**). After filling in all the fields and clicking on the "Send" button, the new tenant will be added to the database. If errors occur, the user will be directed to the error page.

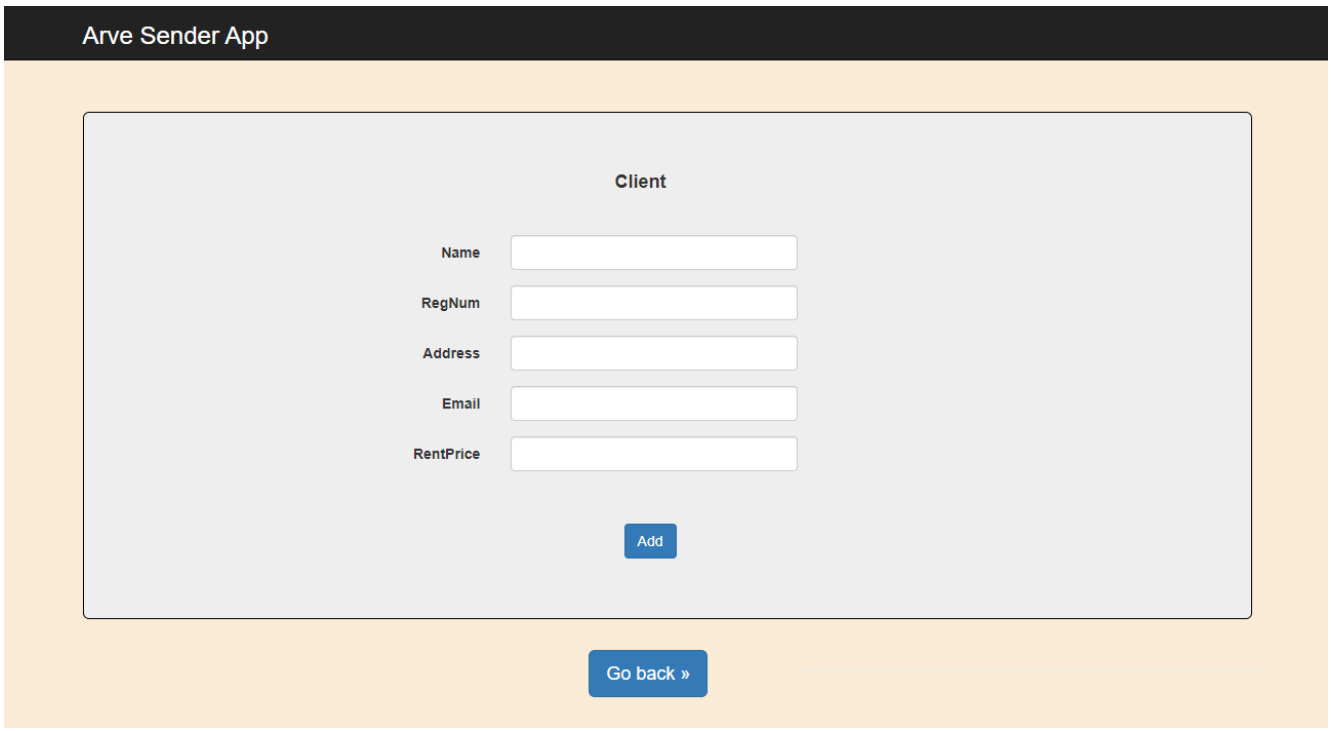

Figure 23. *Clients Add page (Source: author)*

4.4.5 Edit Client Page

After clicking the "Edit" button, the user gets to the page of editing tenant (**Figure 24**). After filling in all the fields and clicking on the "Save" button, the updated tenant will be updated in the database. If errors occur, the user will be directed to the error page.

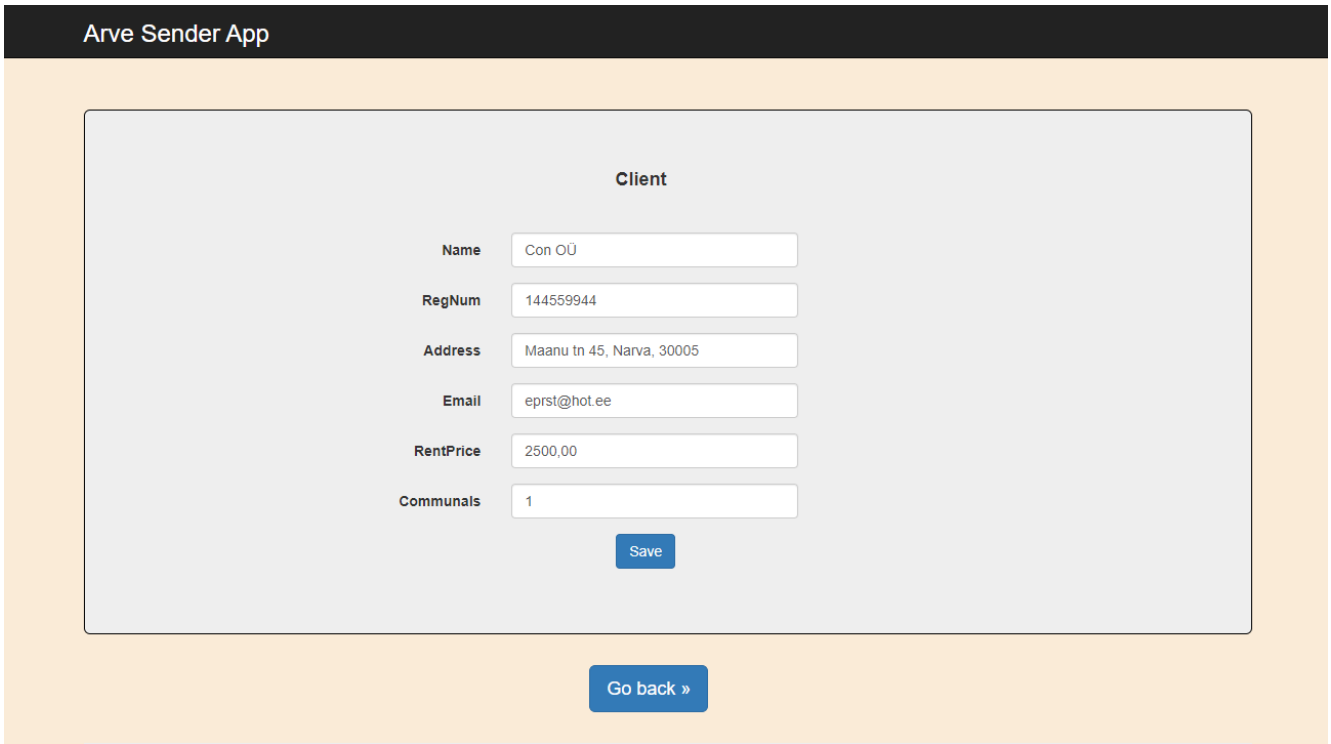

Figure 24. *Clients edit page (Source: author)*

4.4.6 Prices Page

If the user wants to change the prices for utilities, then on the "Home" page he selects the appropriate button and gets to the price page (**Figure 25**), from where he can go to the price change page.

After clicking the "Edit" button, the user gets to the page of editing price (**Figure 26**). After filling in all the fields and clicking on the "Save" button, the updated price data will be updated in the database. If errors occur, the user will be directed to the error page.

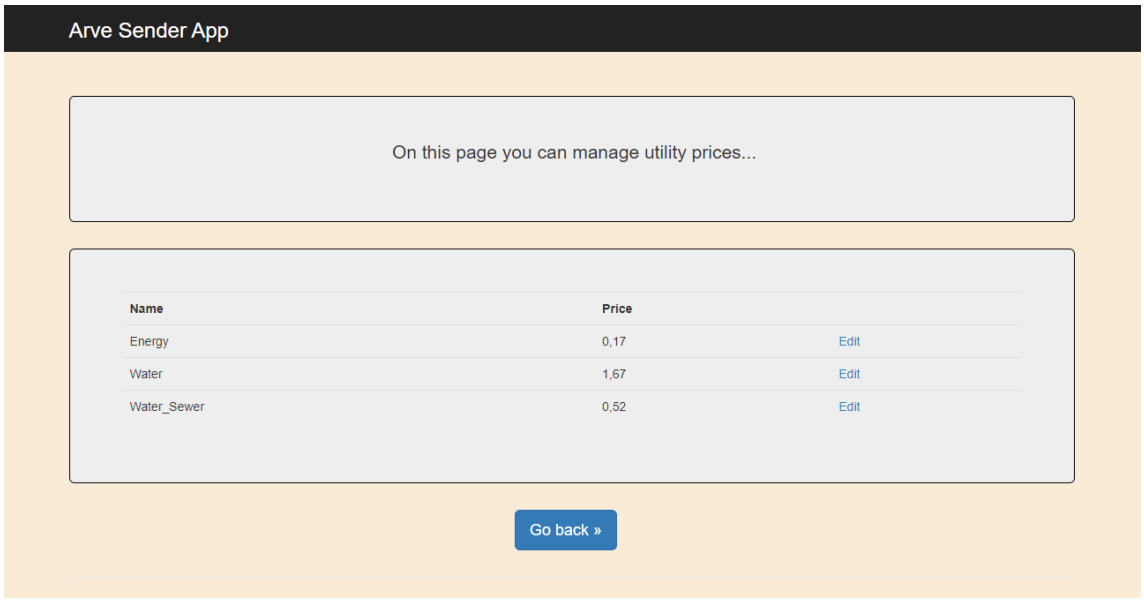

Figure 25. *Prices page (Source: author)*

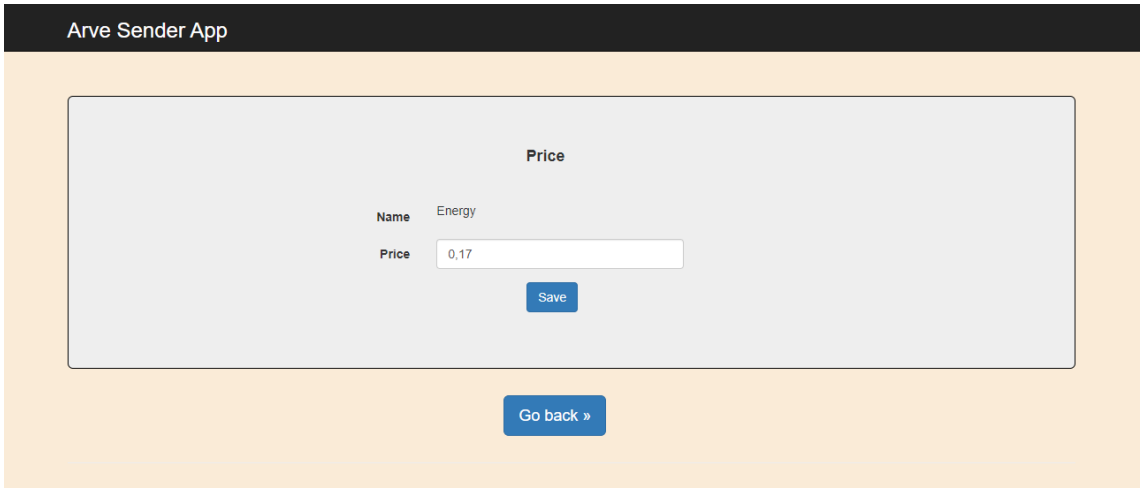

Figure 26. *Prices edit page (Source: author)*

4.4.7 Invoices Page

On the invoices page (**Figure 27**), the user sees data about invoices that have been sent to tenants.

On this page, the user sees:

- Type of invoice.
- Number of invoice.
- Date when invoice was sent.
- Total price of invoice.

User can download any invoice if needed.

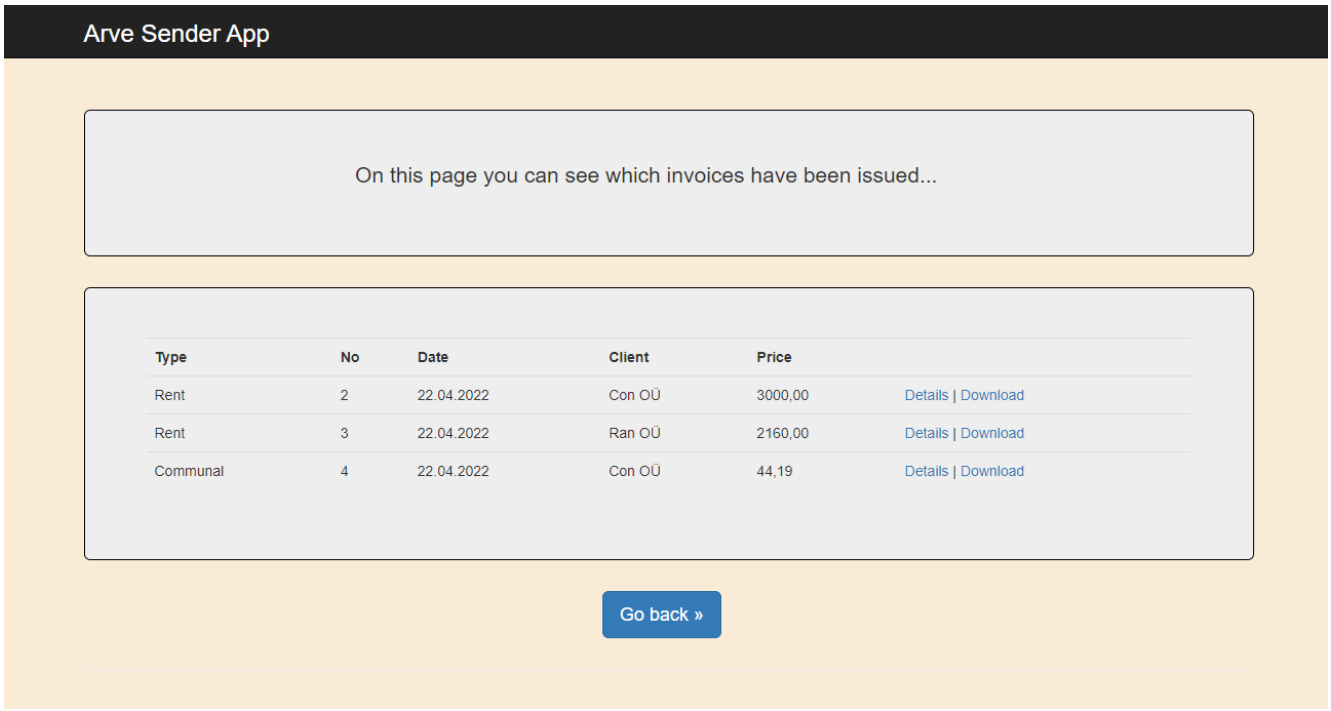

Figure 27. *Prices edit page (Source: author)*

4.4.8 Send Invoice Page

On the "Sent invoice" page (**Figure 28**), the user sees a list of all tenants. To send a rental invoice, the user clicks the "Rent" button. To send a utility bill, the user clicks the "Communal" button. On the next page (**Figure 29**), the application asks if the user is sure that he wants to send an invoice. If the invoice is for rent, then after clicking the "Yes" button, the invoice is automatically created and sent. If there is a utility bill, the user is taken to a page where he must enter meter readings. If the tenant does not have to pay utility costs, then the web application will inform about it with the appropriate page (**Figure 30**).

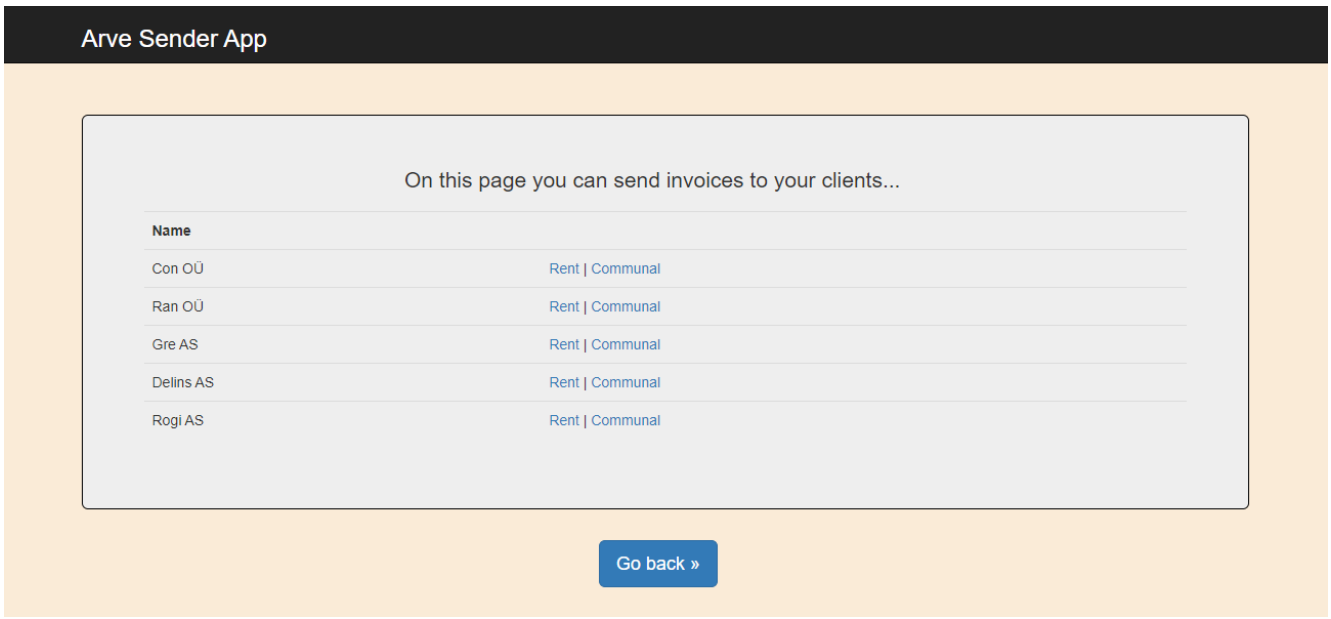

Figure 28. *Send invoice page (Source: author)*

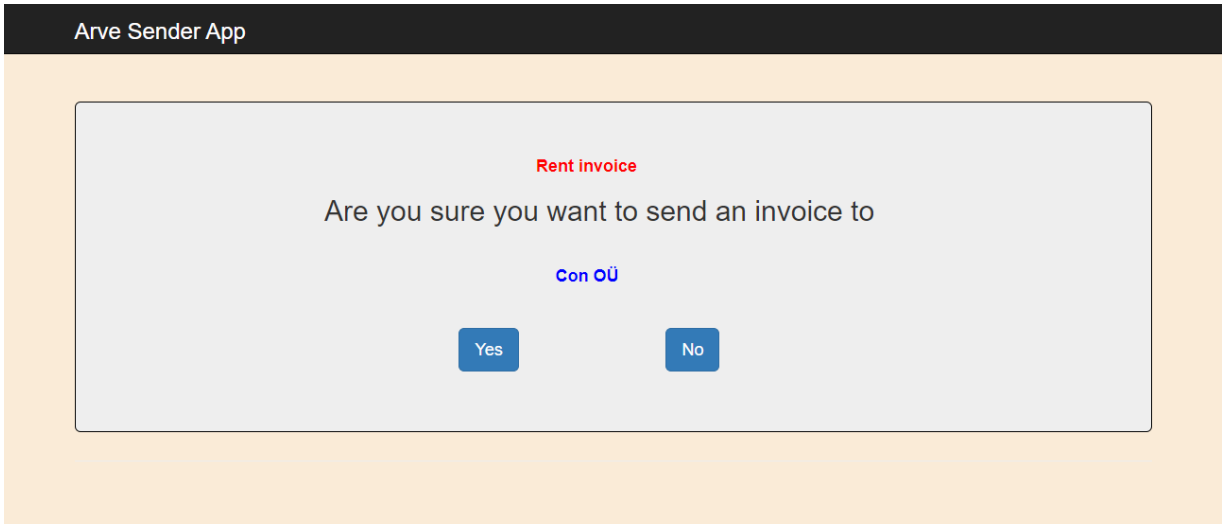

Figure 29. "*Sure" page (Source: author)*

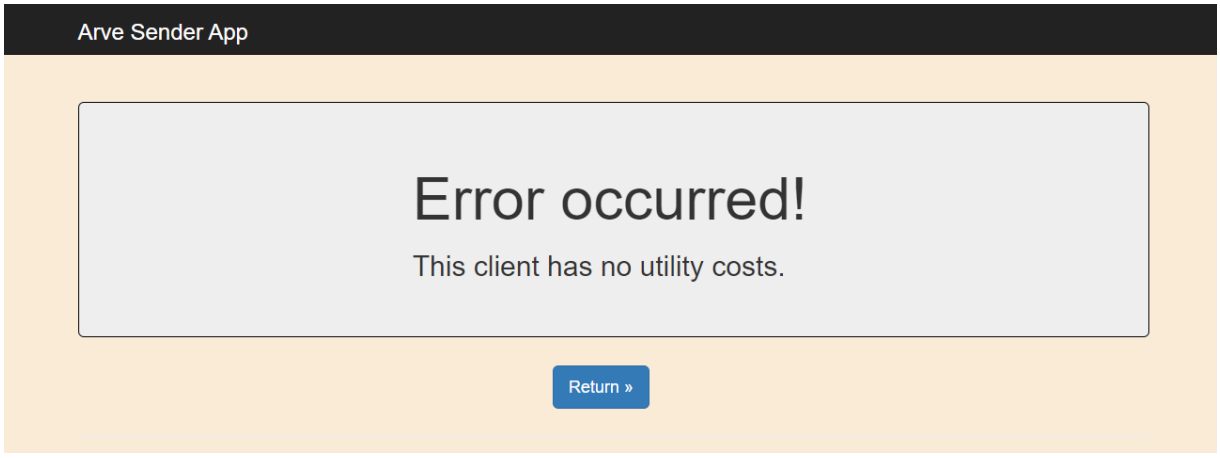

Figure 30. *Utility error page (Source: author)*

4.4.9 Error Page

On this page (**Figure 31**), the user receives a message that an error has occurred, and the operation cannot be performed.

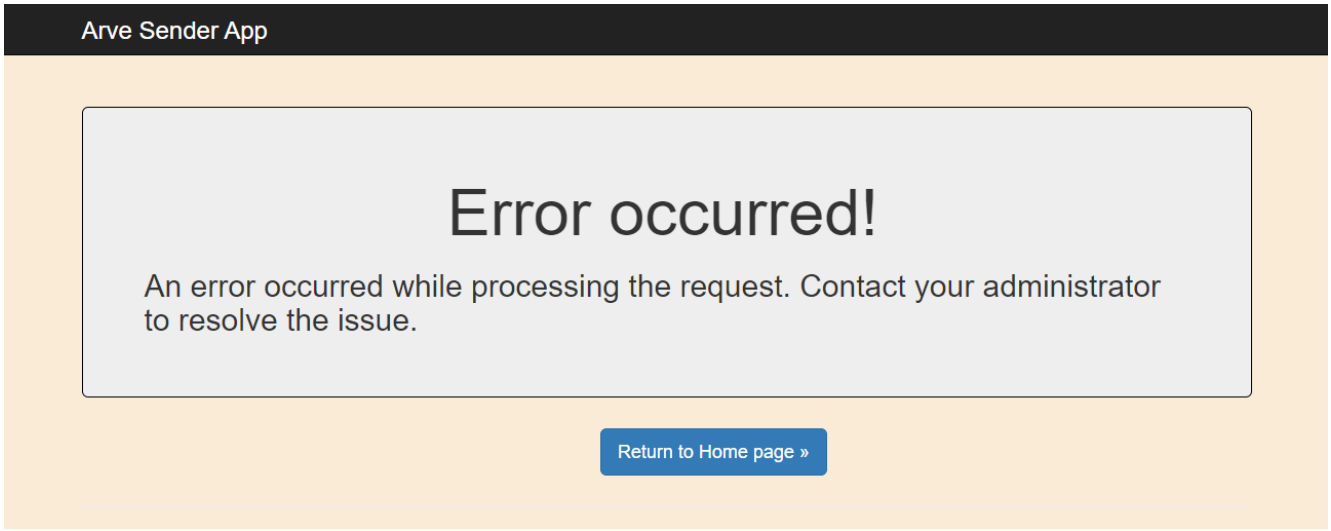

Figure 31. *Main error page (Source: author)*

4.6 Android Application

The application was created in the Microsoft Visual Studio 2019 development environment. Author decided to use C# because he had better experience with it.

The Xamarin android platform was used to create the application. Even though the author had no experience working on this platform, it was chosen because of the large amount of information and instructions in the public domain, as well as because the functionality and interface of the mobile application is not supposed to be complicated.

For the mobile application to work, is needed a constant and stable Internet connection and a minimum version of the Android operating system v5.0

The application images shown in the work are not final and will still be improved.

4.5.1 Main Page

When the application is launched, the user gets to the main screen (**Figure 32**), where the user must enter his user username and password. After entering the authorization data, the user must click on one of the two buttons. One button (ren) leads to a page for sending a rental invoice. The second button (utilities) goes to the page to send the utility bill.

When clicking on any button, the correctness of the username and password entered by the user with data from the database is first checked. If the data is incorrect, the user will see a pop-up error message.

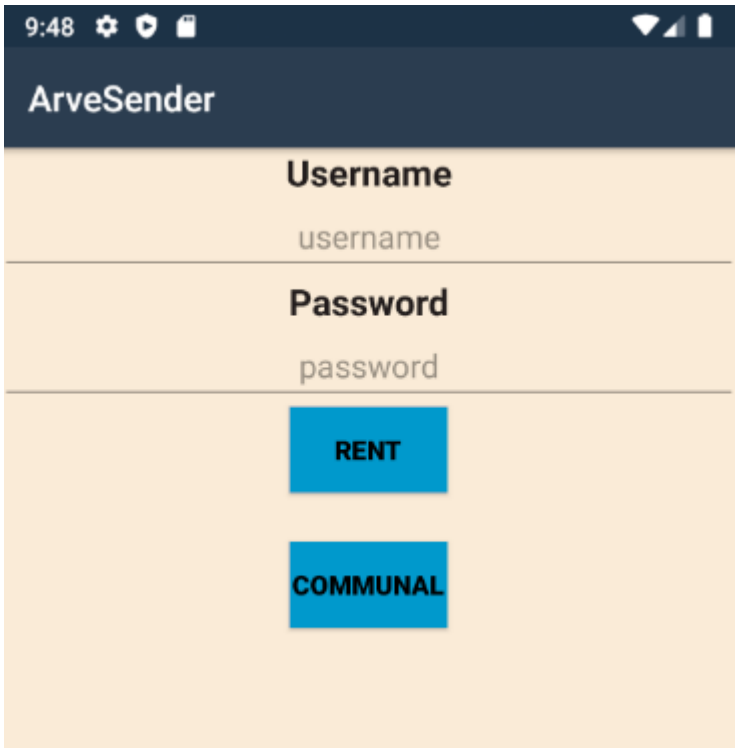

Figure 32. *Main page (Source: author)*

4.5.2 Rent Page

If the user entered the correct data on the main page and clicked on the "Rent" button, then he gets to the page (**Figure 33**) where all the names of tenants who can be billed for rent are displayed. The user can click on one name from the list and the application will immediately automatically send a request with a parameter to a web application that will create and send an invoice to the specified tenant.

If the request was successful and the invoice was sent, the user will see a pop-up message about the result of his action.

Figure 33. *Rent Page (Source: author)*

4.5.3 Communal Page

If the user entered the correct data on the main page and clicked on the "Communal" button, then he gets to the page (**Figure 34**) where all the names of tenants who can be billed for utilities are displayed. The user can click on one name from the list and the user will be redirected to a page where he can enter data taken from utility meters. After entering the data and clicking the "Send" button the application will immediately automatically send a request with a parameter to a web application that will create and send an invoice to the specified tenant.

If the request was successful and the invoice was sent, the user will see a pop-up message about the result of his action.

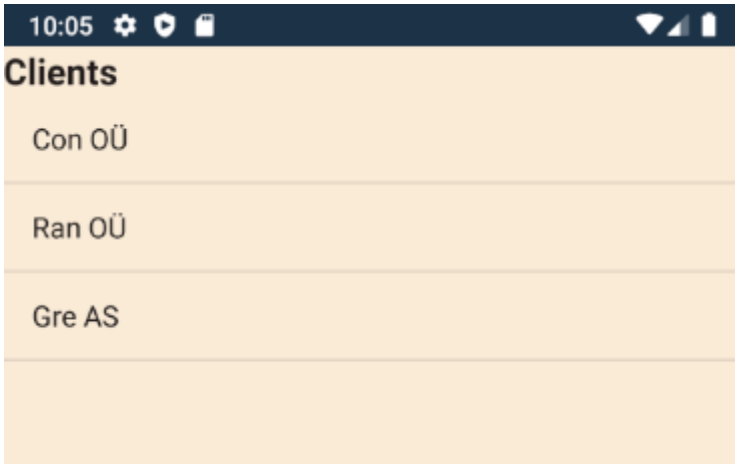

Figure 34. *Rent Page (Source: author)*

4.5.4 Communal Send Page

On the page for sending meter readings (**Figure 35**), the user sees four fields for entering data and the "Send" button. After entering the data and clicking the "Send" button the application will immediately automatically send a request with a parameter to a web application that will create and send an invoice to the specified tenant.

If the request was successful and the invoice was sent, the user will see a pop-up message about the result of his action.

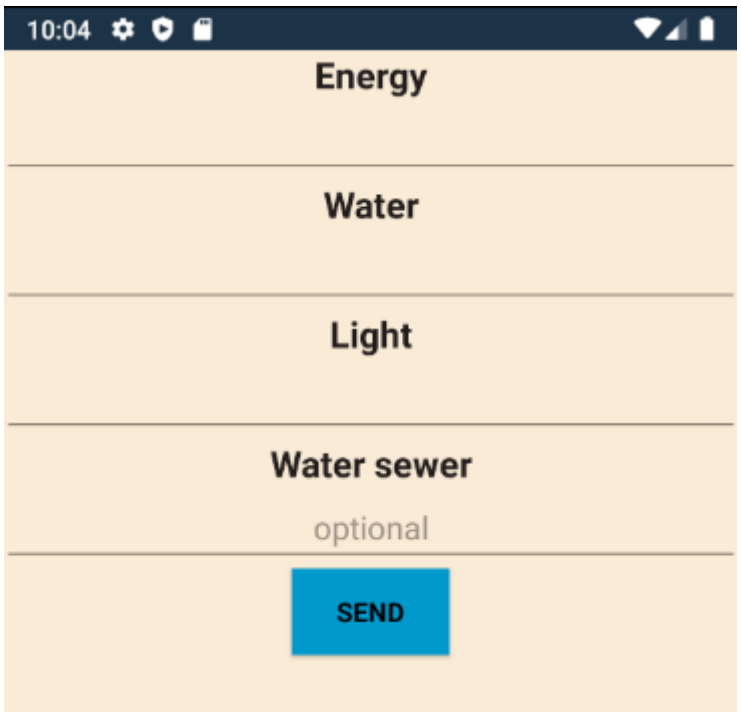

Figure 35. *Communal Send Page (Source: author)*

4.7 API

In the application of this project, it is the web application that does all the key work. The web application connects to the database. But to transfer data between a web application and a mobile application, the author wrote a small API application in C# and ASP.NET MVC framework that he integrated into a web application.

The API application uses the Http library built into the ASP.NET MVC framework. When creating and sending invoices, the mobile application sends a request to the web application with text parameters using the “HttpPost” method. The API controller processes this request, extracts these parameters, recognizes them, and passes the data to a special class of web application that creates an invoice, sends it and updates the data in the database.

There are two methods in the API controller.

A method for creating and sending invoices for the rental of space. One parameter comes to this method – the name of the tenant, which the user selects in the mobile application. The method connects to the database, gets the necessary data by name, and passes it to an internal class to create and send an invoice.

The second method is to create a utility bill. 5 parameters come to this method – this is the tenant's name, data from the electricity, water meter, and so on. Since the parameters come in the form of text, the method processes this parameter, converts them, connects to the database to get additional information, and passes it to the internal class to create and send an invoice.

5 RESULT

In this chapter, the author will list the functionality implemented in the system and the functionality that the author was unable to implement.

5.5 Implemented functionality

At this stage of application development, they fully perform the tasks that were defined by the customer. To do this, the author has implemented the following steps:

- The author has developed and created a simple database in which all the information necessary for the operation of the application is stored.
- The author placed the database in a cloud service.
- The author has developed and implemented a web application.
- Posted a web application on the Microsoft Azure server.
- Developed and implemented a mobile application.
- Developed and implemented an API application and integrated it into a web application.

As a result of the actions carried out by the author, the application works stably and constantly. The database with the necessary information is always available from the Amazon RDS service. The web application is always accessible from Microsoft servers. The mobile application transmits data through the API application to the web application, which implements the function of creating and sending invoices.

Functional result:

- Now the user can automatically create and send an invoice by pressing a button in the mobile application.
- The user can enter data from utility meters and automatically create and send an invoice.
- The user can create and send an invoice automatically in the web application.
- The user can add, change, and delete tenants in the web application.
- The user can change the prices in the web application.
- The user can see which invoices have been sent and download them.

5.6 Unimplemented functionality

An important feature that would be very useful but was not implemented by the author is the multilingualism of the application. Even though the application (both mobile and web) has a very simple

and intuitive interface this is a very simple function, but it was not originally in the plans and the customer did not want this function. Another "Send to all" feature will be added to the apps, which will send rent invoices to all tenants at once with the click of a button.

5.7 Future development

In this chapter, the author will describe features and changes that may be added in the future. The author plans to continue developing and supporting the application after defending his thesis, so he considers this chapter to be very important.

The author also needs to improve the interface of the program in near future.

Also, it makes sense to consider a radical expansion of functionality. Make the system two-sided. That is, so that users who rent a room (residential or non-residential) can use the android application themselves and send meter readings for utilities to the owner and immediately receive electronic bills. In this case, it is necessary to expand the web application and mobile application.

In this case, the application will become publicly available. In the current situation, there are many native speakers of different languages living in Estonia. Then it needs to add support for different languages so that it would be convenient for everyone to use the application.

CONCLUSION

During the diploma project, the author received feedback from an entrepreneur on how he bills tenants for renting commercial premises. As a result, it became clear that for some businesses there was a billing problem that involved the effort required to complete the rent and utility bills. The author also approached other businesses that issue similar bills with this question, and they confirmed the problem and showed interest in finding a solution that could optimize the billing procedure.

The author of the project decided to conduct a study of those billing applications that already exist and chose three of the most famous programs for a full study. After studying these applications, it became clear that all of them cannot solve the problem stated by the customer. The main disadvantages of these applications are that they cannot create an informative utility bill and even billing the rent also requires time or additional user intervention. More inconvenient is billing several tenants at once.

After researching the identified problem, the author of the project had a clear vision of the situation and came up with a very simple and effective solution - the development of mobile and web applications for invoicing, the feature of which will be a simple and fast procedure for invoicing tenants. The author thought over the principle of application operation, functional and non-functional requirements and made a development plan. The author used technologies and development tools already familiar to him, and because of additional study, he improved his skills in working with the C# language, and the ASP.NET and Xamarin frameworks. The author also gained new experience with cloud services.

The author has achieved the result, most of the declared functions work. The author plans to continue developing this system and the Android application. Even though, in the process of development, new non-functional requirements appeared with applications that are mainly related to interface design, the author intends to implement them soon. Since this program is not intended for mass use, it is necessary to introduce changes in the database and in the code for each new user (customer) according to their requirements.

The author believes that this work gave him a large amount of experience in developing Web applications with ASP.NET framework, applications for Android, building databases, communicating with the database using API.

REFERENCES

C# info available at https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/, accessed on April 24, 2022

Microsoft Visual Studio 2019 info available at https://en.wikipedia.org/wiki/Microsoft_Visual_Studio, accessed on April 24, 2022

ASP.NET MVC info available at https://en.wikipedia.org/wiki/ASP.NET_MVC accessed on April 24, 2022

Xamarin info available at https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin, accessed on April 24, 2022

MySQL info. Available at https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html, accessed on April 24, 2022

MySQL Workbench info available at https://dev.mysql.com/doc/workbench/en/wb-intro.html, accessed on April 24, 2022

Amazon Web Service – RDS info available at https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html, accessed on April 24, 2022

Amazon Web Service – Amazon EC2 info available at https://docs.aws.amazon.com/ec2/index.html, accessed on May 14, 2022

EPPlus info available at https://epplussoftware.com/en/Developers/Features, accessed on April 24, 2022

Spire.XLS info available at https://www.e-iceblue.com/Introduce/excel-for-net-introduce.html#.YmWjOOrP2Uk, accessed on April 24, 2022