CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of Nuclear Sciences and Physical Engineering

System for inventory in educational organisation

Systém pro inventury ve školství

Bachelor's Degree Project

Author:	**Vít Vágner**
Supervisor:	**doc. Ing. Miroslav Virius, CSc.**
Language advisor:	**Mgr. Hana Čápová**
Academic year:	2022/2023

Katedra: matematiky   Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	Vít Vágner
Studijní program:	Aplikovaná informatika
Název práce (česky):	Systém pro inventury ve školství
Název práce (anglicky):	System for inventory in educational organisation

Pokyny pro vypracování:

1) Seznamte se s problematikou evidence a inventarizace majetku ve školství.
2) Sestavte a analyzujte požadavky na aplikaci pro podporu inventarizace majetku ve školství.
3) Navrhněte aplikaci pro podporu evidence a inventarizace majetku ve školství. Aplikace by měla umožňovat zadávání položek majetku, jejich převody mezi místnostmi, změnu zodpovědné osoby, tisk inventarizačních sestav, spolupráce s čtecími zařízeními pro elektronickou inventarizaci.
4) Navrženou aplikaci implementujte a otestujte.
5) K vytvořené aplikaci vypracujte uživatelskou příručku.

Doporučená literatura:

1) M. Virius: Programování v C# od základů k profesionálnímu použití. Praha: Grada Publishing 2020. ISBN 978-80-271-1216-6 (tisk), 978-80-271-4004-6 (elektronická publikace), ISBN 978-80-271-4003-9 (pdf).
2) TypeScript. Dostupné na https://www.typescriptlang.org/docs/. [citace 6.10.2022]

Jméno a pracoviště vedoucího bakalářské práce:

doc. Ing. Miroslav Virius, CSc.
Fakulta jaderná a fyzikálně inženýrská, ČVUT Praha, Trojanova 13, 120 00 Praha 2

Jméno a pracoviště konzultanta:

Datum zadání bakalářské práce: 31.10.2022

Datum odevzdání bakalářské práce: 2.8.2023

Doba platnosti zadání je dva roky od data zadání.

Acknowledgment:
I would like to thank doc. Ing. Miroslav Virius, CSc. for his expert guidance and express my gratitude to Mgr. Hana Čápová for her language assistance.

Název práce: Systém pro inventury ve školství

Název práce: System for inventory in educational organisation

Autor: Vít Vágner

Obor: Aplikovaná informatika

Druh práce: Bakalářská práce

Vedoucí práce: doc. Ing. Miroslav Virius, CSc., Fakulta jaderná a fyzikálně inženýrská, ČVUT Praha, Trojanova 13, 120 00 Praha 2

Abstrakt: Každá úřední budova by měla mít inventarizační systém, který udržuje pořádek v majetku, a vzdělávací organizace nejsou výjimkou. Tato práce představuje analýzu inventarizačního systému, přehled současný konkurenčních systémů, poté poskytuje rozbor technologií a software architektury, a následně uvádí nasazení jak webové, tak i mobilní aplikace. Cílem této práce je poskytnout školním organizacím možnost využítí internetového inventarizačního systému.

Klíčová slova: .NET, .NET MAUI, Entity Framework, TypeScript, Vue.js, databáze, inventarizační systém, mobilní aplikace, webová stránka

Title: **System for inventory in educational organisation**

Author: Vít Vágner

Abstract: Every official building should have an inventory system to keep track of its assets. This paper presents an analysis of an inventory system and reviews comparable systems. Additionally, it presents a comprehensive evaluation of the technology and software architecture. The deployment of both mobile and web applications is then presented. The purpose of this thesis is to provide school organizations with the opportunity to use a web-based inventory system.

Key words: .NET, .NET MAUI, Entity Framework, TypeScript, Vue.js, database, inventory system, mobile application, website

Contents

Introduction

Maintaining an inventory of a building is complex and time-consuming when performed manually. Inventory management software can simplify the process but can also impede the process if the system is not designed for inventory management. Therefore, having software tailored to the specific needs of educational institutes is crucial for achieving optimal productivity. The aim of this paper is to explore the importance of these systems and provide a detailed analysis of an inventory system created explicitly for academic buildings.

The application should provide a primary inventory function, such as populating the system with data, data management, data manipulation and inventory checking.

Analysing the inventory system is essential to understand what can be expected from it. The analysis involves reviewing competing software to gain a better understanding of best practices in the sector. Additionally, the research presents a breakdown of use cases that describe the actions involved in how the inventory system should work.

Before developing software, it is important to carefully consider the use of technology. Choosing suitable technologies to support application development and code implementation is crucial.

The software will be developed based on a multi-layer architecture. The data layer processes and stores data in the database. The logic layer functions as a security layer and intermediary between the other two layers. The presentation layer is divided into two parts, as it represents a website and a mobile application. Including mobile application deployment acknowledges the increasing importance of mobile accessibility.

Software must undergo testing to verify its correctness during development. After testing, the testers may suggest new features that can be either implemented or developed in the future.

This thesis aims to provide a web-based inventory system with a mobile application that can efficiently track and manage properties and assets. The system shall provide flexibility to create an environment for each building, credibility in protecting stored data, reliability in the new technologies used, and a user manual to guide users. The source code of the software will be available on GitHub.

Chapter 1

Analysis

Designing and developing an application involves a number of steps. Considering the expected application functionality is crucial. Reviewing the state of the art, gathering ideas and making assumptions. Examining competing software provides a comprehensive view of the features required for an application to have the upper hand over competing software. Merely reviewing competing software isn't sufficient. Additional ideas must be developed. The overall workflow of the inventory system also needs to be considered. Moreover, the application's use needs clarification, including tasks like record creation, data management, inventory checks, and others. Without all of this, the application would be just another average inventory system that would not stand out.

1.1 Review of competing software

The review provides essential suggestions for the development of coherent and more sophisticated software. The first step is to identify a selection of inventory systems that are considered to be competing software. The selection is based on inventory systems that meet various criteria. The criteria are as follows: they should be free of charge, accessible via the Internet, easily accessible to all, and designed to meet educational institutions' demands. The review consists of points on an easy-to-navigate interface, record creation, key features of competing software and criticisms. The selection process is outlined below. Feedback on ideas from the reviews to improve the application process is also provided.

1.1.1 Sortly

Sortly [1] is an inventory software for small businesses. It offers inventory management, asset tracking, supplies, consumables, and sales.

At first glance, the interface seems clear and practical. However, it becomes more complicated as more complex actions are performed. The software introduces folders into which records of items are placed. Each folder has a name, a tag and a note.

When creating a new record, the user can fill in only the mandatory fields or all the fields. If only the required fields are selected, a modal window appears, and if all fields are selected, the user is taken to a more detailed page. There are fields such as item quantity, price, tag, note and barcode. An interesting approach is that the system offers to add custom fields to the item.

The most important feature is probably a tag. There is a dedicated page for displaying tags. They are a kind of filter and categorisation of the items.

There is an overview page that shows some statistics, such as the total value of all items or the total number of items in the system. The page also displays activity, which shows what has happened to each item recently. The statistics can be filtered by including or excluding certain folders.

The inventory implementation seems fine, but there is no practical use for all the features.

1.1.2 Zoho Inventory

Zoho Inventory [2] is inventory management software for growing businesses. The software includes features such as inventory control, customer lifecycle, supplier relationships, integrations and automation.

Zoho Inventory gives the impression of being a sophisticated system, but it seems to be more than just an inventory system. The software also implements accounting such as vendors, purchases, invoices and bills. It focuses on import and export. It could be said that the system represents a small shop that buys and sells items.

It does not meet the standards of a typical inventory system, but some features can improve the workflow of this developed application. The system offers page tips that explain what to do with the currently open page. There is an option to upload any document.

Creating a new item record is very detailed. There are fields such as dimension, weight and manufacturer, then optional fields, sales information and purchase information. In the case of Zoho Inventory, this is necessary. However, as an inventory system for educational institutions, it lacks a lot of information and categorisation. In addition, the use of suppliers, customers and sales orders is unnecessary in this sector.

1.1.3 SalesBinder

SalesBinder [3] is a sophisticated inventory software that has options to customise itself. Users can create custom fields, create item variations or view profits.

SalesBinder meets most of the requirements set out in the selection criteria but contains unnecessary sections in sales which are not important for an inventory system for educational institutions. It has manageable locations with zones, manageable categories, an easy-to-read list of registered item records and item transfer on a single page.

Unfortunately, it contains many features that are unusable for an educational inventory system. For example, the Orders and Accounts section is not helpful as the academic organisation does not deal with these sections.

The creation of a new record is very detailed. Before a record can be created, at least one category has to be available. If not, there is an option for the creation of one. Once the category has been selected, SalesBinder takes the user to the details of the record. The detail fields are primary actions involving stock units and money, such as quantity, price per unit and selling price, all of which do not correspond to the basic inventory of records.

1.1.4 BoxStorm

BoxStrom [4] is an inventory management solution that integrates with third party applications. It has a mobile application for scanning barcodes and cloud-based storage.

BoxStorm has not only a web application but also a desktop application and a mobile application. The above systems also include sales sections such as purchase and sales orders with suppliers and customers. It can be said that it represents a simple digital warehouse or store.

An item can have a photo, quantity, SKU code and price. When the item is created, there is an option to see how each field has been changed. BoxStorm also keeps track of the quantity in stock.

One of the main features is the integration with third party applications. BoxStorm can connect to various applications such as shipping services FedEx or UPS, or payment services such as Stripe.

For running a small business or warehouse, BoxStorm looks ideal, but on a larger scale, BoxStorm will not be sufficient.

1.2 Feedback from reviews

As expected, none of the selections fully fit an educational inventory system. Based on the reviews above, the key features should be implemented in the application.

- **Clear and practical interface:** The design should be intuitive and user-friendly to ensure ease of use, and follow material design standards to remain relevant to modern regulations.
- **Mandatory and optional fields:** Providing the option of not filling in all fields allows the user to quickly create a new record and fill in the rest later or to fill in the optional field at the beginning for more detailed creation.
- **Reports:** Allows the user to view graphs and statistics.
- **Activity Tracking:** Keeps track of recent activity.
- **History Tracking:** Tracks what has been changed in the records.
- **Page Tips:** Including page tips in the web page helps the user to quickly understand what to expect from the page and how to interact with it.
- **Location and Category Management:** Locations and categories are introduced to effectively filter and sort items.
- **Scalability:** The inventory system should be prepared for possible future changes. The system should be model-based so that changes can be easily implemented. Ensure that the software can handle larger scale operations for businesses that may grow.

- **Export:** Exporting data, for example, to an Excel spreadsheet, where the user can manipulate and interact with the data.
- **Document Upload:** Allows the user to upload relevant documents.
- **Mobile Application:** Develop a mobile application that is compatible with a range of devices to ensure that the software is accessible and usable beyond the confines of a computer.

Incorporating these features into the application can meet the requirements of educational institutions and enable efficient inventory management and workflow.

1.3 Use Cases

The reviews provided valuable insights into decomposing the workflow into individual use cases. The use case explains how the user performs a specific task. A declaration of the necessary resources is provided before describing the use cases.
An actor is an entity that performs actions within a use case. *A flow* describes the process used to achieve a use case. *A precondition* is a state that the actor must have before performing the flow. *A postcondition* is the state in which the actor ends after performing the flow.

Actors:

- **A user:** a person using a website or mobile application.
- **An administrator:** the user with access to the Administration section (see section 5.4.3).
- **API (Application Programming Interface):** the logic layer (see section 2.2).
- **A website:** the first part of the presentation layer (see section 2.3).
- **A mobile application:** the second part of the presentation layer (see section 2.3).

1.3.1 UC1: Logging in

Actors: the user, API, the website
Preconditions: User is on the login page.
Postcondition: User has access to the system.
Flow:

1. User enters a username and password.
2. User clicks on the ’Login’ button.
3. Website sends a HTTPS POST request with a login model containing the user’s credentials.
4. API computes the request and returns a corresponding response to the website.
5. Website checks that the response is successful and retrieves a bearer token from the response.
6. User is logged into the website and taken to the Evidence Page.

1.3.2 UC2: Creating a new user

Actors: the administrator, API, the website
Preconditions: Administrator is on the user administration page.
Postcondition: New user is added to the system.
Flow:

1. Administrator clicks on the ’Add User’ button.
2. Websites opens a modal window with fields for *username*, *first name*, *last name* and *password*.
3. Administrator fills in the fields and clicks the ’Save’ button.
4. Website sends an HTTPS POST request with a user model containing user information.
5. API computes the request and returns a true or false response depending on whether the request was successful.
6. Website closes the modal window and refreshes the page to display the created user.

1.3.3 UC3: Creating a new category

Actors: the administrator, API, the website
Preconditions: Administrator is on the category administration page.
Postcondition: New category is added to the system.
Flow:

1. Administrator clicks the ’Add Category’ button.
2. Websites opens a modal window with the *name* and *skuPrefix* fields.
3. Administrator fills in the fields and clicks the ’Save’ button.
4. Website sends an HTTPS POST request with a category model containing the category information
5. API computes the request and returns a true or false response depending on the success of the request
6. Website closes the modal window and refreshes the page to display the created category

1.3.4 UC4: Creating a new room

Actors: the administrator, API, the website
Preconditions: administrator on the room administration page
Postcondition: a new room is added to the system
Flow:

1. Administrator clicks the ’Add Room’ button.

2. Websites opens a modal window with *name* and *location* fields.
3. Administrator selects the appropriate location for the room.
4. Administrator fills in the fields and clicks the 'Save' button.
5. Website sends an HTTPS POST request with the room model containing the room information.
6. API computes the request and returns a true or false response depending on the success of the request.
7. Website closes the modal window and refreshes the page to display the created room.

1.3.5 UC5: Creating a new location

Actors: the administrator, API, the website
Preconditions: Administrator is on the location administration page.
Postcondition: New location is added to the system.
Flow:

1. Administrator clicks the Add Location button.
2. Websites opens a modal window with *name*, *shortcut*, *address* and *photo* fields.
3. Administrator fills in the fields and uploads a location photo.
4. Administrator clicks the 'Save' button.
5. Website sends an HTTPS POST request with a location model containing the location information.
6. API computes the request and returns a true or false response depending on the success of the request.
7. Website closes the modal window and refreshes the page to display the created location.

1.3.6 UC6: Creating a new item

Actors: the user, API, the website
Preconditions: User is on the evidence page. There is an existing location, room and category entities.
Postcondition: New item is added to the system.
Flow:

1. User clicks on the 'Add Record' button.
2. Websites redirects the user to the detail page.
3. User selects appropriate *ItemPart* - if none exists, the user creates *ItemPart* by clicking the 'New' button and entering category, name and description.

4. API generates SKU code according to category and inserts today's date in CreationDate.
5. User selects the location, room and responsible person.
6. User clicks the 'Save' button, and the website sends an HTTPS POST request with the item model containing the item information.
7. API computes the request and returns a true or false response depending on the success of the request.
8. Website returns to the evidence page where the new record is displayed.

1.3.7 UC7: Editing records

Editing is similar across all records, so only one use case exists.

Actors: the user, API, the website
Preconditions: User has selected the record.
Postcondition: Record is modified.
Flow:

1. User can edit all editable fields and then click the 'Save' button.
2. Website sends an HTTPS PUT request containing the modified information with the record class model.
3. API computes the request and returns a true or false response depending on the success of the request.
4. Website returns and the modified record can be found in the table.

1.3.8 UC8: Deleting records

Actors: the administrator, API, the website
Preconditions: Administrator has selected the record.
Postcondition: Record is deleted.
Flow:

1. Administrator clicks on the 'Delete' button.
2. Website sends an HTTPS DELETE request with the record ID.
3. API computes the request and returns a true or false response depending on the success of the request.
4. Website returns, and the record is deleted.

1.3.9 UC9: Transferring records

Actors: the administrator, API, the website
Preconditions: User is on the transfer page.
Postcondition: Records are transferred.
Flow:

1. User selects the location and the room within it in the left column.
2. Website shows the user which item records are in the selected room.
3. User selects the desired location and the room within it in the right column.
4. User clicks the '+' button on the *Item* record in the left column.
5. Website moves the item record to the right column and provides the '-' button if the user changes their mind.
6. When all desired item records have been moved to the right column, the user clicks the 'Submit' button.
7. Website sends an HTTPS POST request containing the list of *Item* records and where to move them.
8. API computes the request and returns a true or false response depending on the success of the request.
9. Website resets the page to its default state, and the process is complete.

1.3.10 UC10: Getting a list of *Item* records in the room

Actors: the user, API, the mobile application
Preconditions: User is on the home page of the mobile application.
Postcondition: List of *Item* records is shown.
Flow:

1. User selects the room to check in the mobile application.
2. Mobile application sends HTTPS POST request.
3. API computes the request and sends a response as a list of *Item* parts.
4. Mobile application displays the list of item items, and the user can interact with the items.

1.3.11 UC11: Checking a *Item* record

Actors: the user, API, the mobile application
Preconditions: User has selected the room and has the list of *Item* records available.
Postcondition: Record is checked if it is correctly placed.
Flow:

1. User clicks on the record from the list of item records.
2. Mobile application displays a barcode scanner.
3. User scans the barcode of the item.
4. If the scanned barcode matches the record barcode, the record is marked as checked.
5. Mobile application returns to the list of *Item* records to check another.

1.3.12 UC12: Inspecting a room

Actors: the user, API, the mobile application
Preconditions: User has selected the room and has the list of *Item* records available.
Postcondition: Record is checked if it is correctly placed.
Flow:

1. User clicks on the record from the list of *Item* records and performs the UC11 (see section 1.3.11).
2. User repeats the use case until there are no more unchecked records.
3. After checking the entire list, the user marks the room as checked by clicking the ’checked’ button.
4. The mobile application sends an HTTPS POST request with the checked data, including the misplaced records.
5. API computes the request, generates an inspection report and sends a true or false response depending on the success of the request.
6. Mobile application returns to room selection.

1.3.13 UC13: Reporting a misplaced Item record

Actors: the user, API, the mobile application
Preconditions: User has found a misplaced Item record.
Postcondition: Record is reported and ready to be moved.
Flow:

1. User clicks the ’Report’ button to report the misplaced record.
2. Mobile application sends an HTTPS POST request containing the record ID.
3. API computes the request and returns a true or false response.
4. Mobile application displays the *Item* record list, and the user can interact with the records.

1.3.14 UC14: Resolving a reported item

Actors: the user, API, the website
Preconditions: User is on the Inspection detail page, where is a table of reported items.
Postcondition: Reported item is edited and marked as resolved.
Flow:

1. User clicks the button with the pen icon to edit a reported item.
2. Clicking the button takes users to the item details, where they can edit the item as requested in the report.
3. After a successful edit, the user returns and marks the reported item as resolved.
4. Item disappears from the table of reported items.

1.3.15 UC15: Completing an inspection

Actors: the user, API, the website
Preconditions: User is on the Inspection detail page, where is a table of reported items.
Postcondition: Inspection is completed, and the user is taken to the list of inspections.
Flow:

1. User resolves items according to UC14 (see section 1.3.14).
2. User continues until there are no reported items.
3. If there are no reported items in the table, the user can complete the inspection.
4. After clicking the 'Complete' button, API sends a request to complete the inspection, and the website receives a response.
5. If the response is successful, the website takes the user to the list of inspections.

Chapter 2

Software Architecture

Multi-tier architecture [5], also known as three-tier architecture, is a design pattern that divides an application into three layers: data, logic and presentation. Each layer has a specific purpose, making the application more modular, robust, scalable and maintainable.

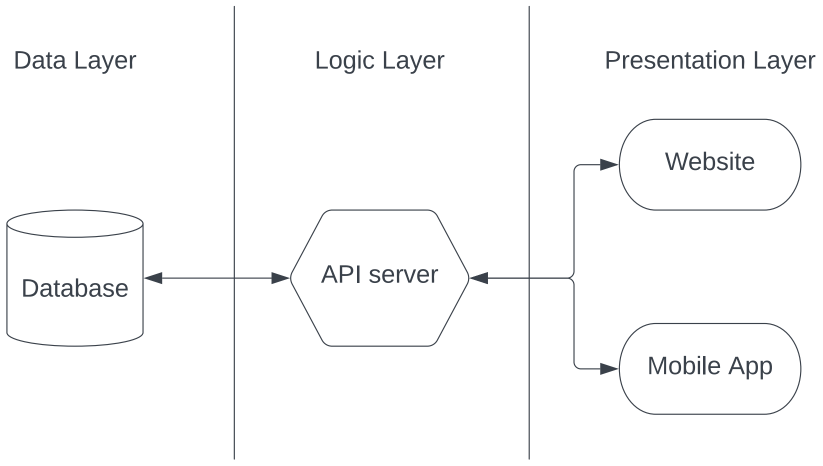


Figure 2.1: Representation of multi-layer architecture

2.1 Data Layer

The purpose of the data layer is the management of data storage for the application. The integrity of the data is ensured by this layer. In this particular case, the data layer is defined by a database system. The data layer's detailed description can be found in chapter 4 – Data Layer (Database).

2.2 Logic Layer

The logic layer contains the core business logic. The logic layer processes and manipulates data from the data layer. Data validation, calculations, and business workflows are handled by this layer. The logic layer acts as an intermediary between the data layer and the presentation layer. Upon receiving a request from the presentation layer, the data layer processes it and transforms the data into a format that is compatible with the presentation layer. This transformed data is then returned as a response. The Application Programming Interface (API) represents the logic layer. Further information can be found in chapter 5 – Logic Layer (API).

2.3 Presentation Layer

The presentation layer presents application data and provides functionality to the user. It manages user input, displays information and delivers a user-friendly interface. The presentation layer must request the data from the data layer to display information. This process is initiated by sending a request to the logic layer, which processes the request and provides a response with the relevant data. Considering this application, the presentation layer is divided into two parts. The first part is a website, and the second part is a mobile application.
Further information about the website in chapter 6 – Presentation Layer (Website) and about the mobile application in chapter 7 – Presentation Layer (Mobile application).

2.4 Model-View-Controller

Model-View-Controller (MVC) [6] is a software architectural pattern that divides an application (in this case, layers) into three components: *a model*, *a view* and *a controller*. This pattern offers a structured and modular approach. Separating these components results in improved code organisation and maintainability. MVC is implemented in both the logic and presentation layers.

Model component represents the business logic of the data layer. Typically, the component includes classes, objects, or data structures that represent different entities or operations. This component is intended to be independent of the other components.

View component presents the data from the model components to the user. It provides a visual representation of the data. This component concentrates on the visual and interactive features of the layer.

Controller component is an intermediary between the above components. It receives user input from the view, performs appropriate actions and updates the rest of the components accordingly.

Chapter 3

Technologies

Choosing the right technologies is the basis for effective, error-free, and intuitive application development. Development can be done without them, but it is not advisable. Frameworks and high-level languages are designed to run efficiently, support code development, and prevent code violations that can lead to the exposure of sensitive data.

3.1 Entity Framework

Entity Framework (EF) [7] is an object-relational mapping (ORM) framework developed by Microsoft. It is used to access data in .NET applications. EF simplifies interaction with the database by providing an abstraction over database operations that allows developers to work with data as objects. EF includes these features:

- **Fluent API** - EF provides its version of a programming query language that compiles into several options such as Structured Query Language (SQL) Server, MySQL, PostgreSQL and SQLite. Using its language, EF is scalable to many projects involving different technologies.
- **Structures** - as mentioned above, EF maps the model classes in the code to tables and relations in the database. Every property of the class has a type. The type can be anything from strings, booleans, integers, dates or even enums. EF maps these types to the types of relational database language. In this case, the query language is Microsoft SQL.
- **Migrations** - EF is versioning the database, i.e. migrations are versions of the database. Creating a migration creates a class that contains two methods. The first is the *UP* method, which represents the current changes that will be implemented in the new database version. To maintain reversibility, the *DOWN* method must be added, which reverses these recent changes. The migration is created with the command *'Add-Migration <name>'*. EF translates the model classes into tables. Fluent API then forms each table according to the class models. The code is generated automatically, so the class models may have been misinterpreted. It is recommended that the generated code is checked.

- **Migration table** - EF also creates a migration table to keep track of all migrations performed. Migrations act as versions of the database, allowing database versions to be rolled back or upgraded by the next migration.

3.2 .NET

NET [8] is a software development framework and platform developed by Microsoft. It enables the development of various types of applications, including web, desktop, mobile and cloud-based applications. The framework consists of an extensive library and a large community of developers who create third-party NuGets that extend the capabilities of the base library. The application uses version .NET 7. This is currently the most up-to-date version and offers the best optimisation for the .NET platform. Unfortunately, this version is not long-term supported (LTS). However, this is not a problem as technological progress is so advanced that it would be inefficient to stay with .NET 6.

3.3 C#

C# [9] is a programming language designed for the .NET framework. It combines the best elements of C and C++. The language is designed to run on the object-oriented programming concept. It offers many resources and tools, as well as extensive community libraries that help to better implement the code. Here are three of the many features included in C#:

- **Language Integrated Query (LINQ)** allows data to be queried from multiple places, making it easier to manipulate data via enumerated objects.
- **Asynchronous Programming** enables the use of all available threads (multi-threading).
- **Specialised libraries** make the language well-suited to building complex applications.

3.4 Visual Studio

Visual Studio [10] is an Integrated Development Environment (IDE) developed by Microsoft. It is widely used for working with C# language because that is the main purpose of it. It supports a lot of features that are designed to help developers to increase their productivity. It suggests a code completion tool for each object in the code through the intellisense. When running the program through Visual Studio, the program can be debugged, which means there are breakpoints that stop the program. The developer can see actual parameters and properties as the program runs.

3.5 Barcode

The system uses barcodes to quickly and accurately check stored item records. There are two types of barcodes: Stock Keeping Unit (SKU) and Universal Product Code (UPC). UPC is commonly used in the global marketplace, such as grocery stores, warehouses or supermarkets. Its purpose is to uniquely identify the product with a predefined set of codes using the GS1 standard [11]. This includes the type of product it represents, the organisation that manufactured it, the country it comes from, or when it expires.

Because the UPC is used globally and contains only digits, the best way to track internal items with barcodes is the SKU barcode. Its advantages are that it can hold many characters and is smaller and more flexible because it is not standardised. The code is not readable by other systems and can be read without a barcode scanner.

The barcode is represented by a string of characters that are translated into binary code and then displayed as black and white lines, where the white stripe represents a binary one. The black line represents a binary zero.

In this case, the barcode consists of two labels, a hyphen and a set of numbers. The two labels are predefined by the *Category* to which the item belongs. There is a hyphen to separate the labels from the numbers, and the set of numbers is an increment of the last barcode generated in the category.

3.6 .NET MAUI

NET Multi-platform App UI (MAUI) [12] is a modern UI framework based on the evolution of Xamarin. It enables cross-platform development using the .NET framework. MAUI allows developers to build native applications for mobile devices, computer desktops and web applications from a single code base. MAUI leverages the strengths of Xamarin to provide a unified and streamlined approach to cross-platform development. Each platform should be uniquely modified and has different requirements, but MAUI takes this into account.

Hot Reload is a feature implemented using Visual Studio and MAUI. It comes from Xamarin, where the feature was introduced but was ineffective. This changes with MAUI because MAUI has no visual interpretation, so it relies on the Hot Reload. The interpretation is projected to the device or simulator. Changes made to the code are immediately (or after saving, depending on the configuration) loaded into the device. The changes can be seen without resetting the program, so developers do not have to build the program, send it to the device and install it. Visual Studio installs the program and the configuration from the Hot Reload. Sometimes the Hot Reload is not enough because the change goes deep into the core of the program, so the program has to be reset.

3.7 HTTPS

HTTPS (Hypertext Transfer Protocol Secure) [13] is a secure version of the HTTP protocol used to transfer data. It ensures that data exchanged between the user's browser and the website remains encrypted and secure, protecting it from potential eavesdropping, tampering or data theft.

3.8 Refit

Refit [14] is a type-safe REST client library for the .NET framework. It simplifies HTTP request generation through interface definition. Refit handles HTTP communication, JSON serialisation and deserialisation of data. It maps JSON objects to C# class models. It is implemented by dependency injection for easier use in code.

3.9 JavaScript

JavaScript (JS) [15] is high-level, just-in-time compiled code based on the ECMAScript [16] standard. Just-in-time means that the code is compiled in the browser as it is needed, so the code is not hidden and is easily accessible and editable. Some disadvantages of JS are that it is compiled in the browser, so there are no pre-compile errors to discover. Another disadvantage is that JS does not include static typing, i.e., there are no pre-declared variables, and variables can change type while the code is running. Lastly, an integrated development environment (IDE) does not check whether the code contains syntax errors, logical errors, compile errors or has no intellisense. The intellisense is a highly valued feature in development because it suggests the developer list of members, parameter info, quick info or completes the word.

3.10 TypeScript

TypeScript (TS) [17] is a powerful open-source programming language developed and maintained by Microsoft. It is a superset of JavaScript, meaning that any valid JavaScript code is also valid TypeScript code. TypeScript negates almost all the negatives of JS, e.g., TS has static typing. The IDE suggests method documentation or error checking. It is also ECMAScript compliant because it is derived from JS. Overall, TypeScript combines the best of JavaScript's dynamic nature with static typing, providing developers with improved productivity, code quality and maintainability.

3.11 Axios

Axios [18] is a widely used JavaScript library for HTTP requests from the browser. Its popularity stems from its simplicity, flexibility and robustness, making it a first choice for developers when communicating with API servers. The use of Axios is interpreted in section 6.1 – Communication with API.

3.12 Vue.js

Vue.js (Vue for short) [19] is a JavaScript framework designed for building the user interface of websites. It is perfect for small to large applications like this one. The reasons for using Vue are as follows: Vue can provide declarative rendering, which allows declarative rendering of data to the Document Object Model (DOM). The DOM is the foundation of the web page. Another reason is that it provides directives and reactivity. These two points can be explained as Vue provides pre-built components for the developer to use, and reactivity is the automatic updating of the UI so that when there is a change in the data, the DOM is rendered again to provide instant feedback of the changes. Vue uses components.

A component is a part of the page that can stand alone and be used in more than one place to eliminate duplication and reduce code complexity.

When managing multiple pages, Vue also provides navigation through the Vue Router. The router dynamically changes URLs and switches pages, can redirect data to another page, or restrict access to navigate to an inaccessible web page. Vue.js is supported by a large community of developers who are constantly improving the flow of the framework and creating components and packages that can be used freely.

3.13 Vuetify

Vuetify [20] is an open-source Material Design component framework for Vue.js, an advanced JavaScript framework for building user interfaces. It includes several reusable UI components and patterns that follow Material Design guidelines, such as buttons, calendars, selections and animations. It gives developers the right tools to create attractive and responsive web applications.

Material Design [21] is a system created and supported by Google. It provides a set of guidelines for creating a visually appealing, consistent, and intuitive user interface. Material Design combines principles with modern technology and graphics.

3.14 IIS Server

IIS Server [22] stands for Internet Information Services Server. IIS is a web server developed by Microsoft. It is designed to host and manage websites, web services and web applications. It offers a user-friendly interface, extensive configuration options, support for SSL certificates and hosting of dynamic web applications developed using .NET, JavaScript and others.

3.15 ZXing.Net.MAUI

ZXing.Net MAUI (ZXing for short) [23] is a community library created by Redth for the MAUI framework. ZXing provides a solution for barcode and QR code scanning using Android and iOS development tools.

Chapter 4

Data Layer (Database)

The database uses the Entity Framework (see section 3.1). Entity Framework (EF) converts model classes that represent individual entities (e.g., items, users, rooms) into a relational database. The database consists of tables, relationships, and indexes.

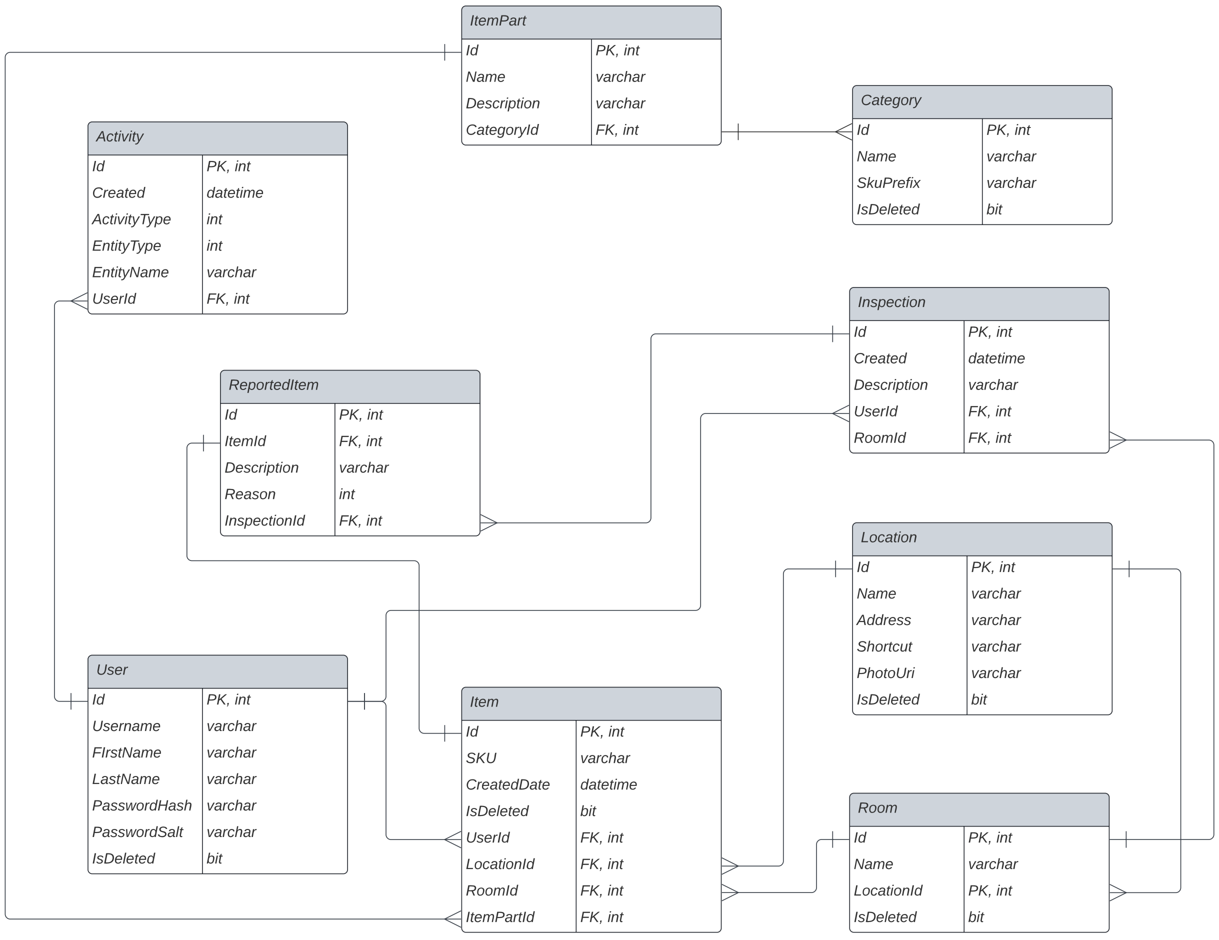


Figure 4.1: ER diagram

4.1 ER diagram

The figure concerning the ER diagram is above (see figure 4.1). The ER diagram illustrates the arrangement of tables in the database. The tables contain properties with their names, types and whether they have primary keys (PK) or foreign keys (FK). Furthermore, the diagram explicates the relationships between every table. The line connecting the primary key of one table to the foreign key of another table indicates the relationship.

4.1.1 Primary key

Typically, a primary key is set by convention, a property called *Id* or *<name>Id*. Fluent API (see section 3.1) defines which property represents the primary key.

4.1.2 Relationships

A relational database includes five types of relationships. These relationships are one-to-one, one-to-many, many-to-one, many-to-many and self-referencing. They are formed by pairing primary keys with foreign keys. EF makes relationships simpler than they were before. Declaring foreign keys and linking them with primary keys is unnecessary. For instance, consider the one-to-many relationship. To create this relationship, it is sufficient to add a collection of classes to the C# class model. EF will then recognise the one-to-many relationship and generate an appropriate SQL query that creates the foreign key and then references the primary key of the second table.

4.2 Tables

Each table contains a unique primary key called *Id*. This primary key has an auto-increment feature, which means that when a new row is added to the table, the *Id* is automatically incremented by one, making the primary key unique for each row.

4.2.1 User Table

The User table has the following properties *Username* for quick identification: *FirstName* and *LastName* for easy recognition. Then *PasswordHash* and *PasswordSalt* together form the password. The *PasswordSalt* hashes the password, which is unique for each user, to the *PasswordSalt*, which is added and then hashed a second time using 256-SHA encryption, after which the password is stored in the *PasswordHash* property. Password security is explained in detail in section 5.4.2 – Password Encryption.

4.2.2 Category Table

The Category table represents the categorisation of the product, for example, some values such as chair, table, monitor and so on. The *Name* property corresponds to the categorisation. The *SkuPrefix* is an essential field for SKU code generation (see section 3.5). As the name suggests, it contains the code prefix so that each category has its unique code and is easily recognised by the prefix.

4.2.3 Location Table

The Location table stands for the physical building. It contains the *Address* where the building is located, the *Name* of the building or the *Shortcut* for quick implementation, and also the *PhotoUri*, where the URI address of the image of the building is stored.

4.2.4 Room Table

The Room table means the physical rooms in the building where the items are located. As can be seen in the ER diagram (see figure 4.1), Room Table has a one-to-many relationship with the Location table, so there is a relation that one *Location* has several *Room* entities.

4.2.5 ItemPart Table

The *ItemPart* represents the physical model of an item, so if there are 20 items of the same product, one ItemPart record describes the product. In terms of description, the *Category* table has one-to-many relationships with the *ItemPart*.

4.2.6 Item Table

Now that the necessary tables have been described, there is the most important table of the inventory systems. The *Item* table corresponds to the physical unique item in the room. The ER diagram shows many relationships. The first relationship is one-to-many with *ItemPart*, as described in the ItemPart Table section (see section 4.2.5). The *ItemPart* represents the model, and the *Item* is an individual record of an item with that model. The second and third are location-based. The relationships show where the *Item* record is located, and the last relationship is the ownership described by pairing the primary key of the User table with the *UserId* field.

4.2.7 Activity Table

The Activity table keeps track of what has been modified and when. The modification is divided into create, update, delete, restore, transfer between rooms, rename and so on, and these values are stored in *ActivityType*. The aim is to track all table changes, so there cannot be a strict relationship between Activity and all tables. This is where the *EntityType* and *EntityName* fields are useful. The *EntityType* indicates which table the record represents, and the *EntityName* shows which specific record of that table it is. It then keeps track of who made the changes by keeping the *UserId* for future reference.

4.2.8 ReportedItem Table

The ReportedItem table is a table representing a reported item. It contains data from the report such as *Reason* – a reason for the report, *Description* – a short description of why it was reported, *ItemId* – an item being considered, and *InspectionId* – an inspection to which the reported item belongs.

4.2.9 Inspection Table

The Inspection table pairs with the ReportedItem Table, with the one-to-many relationship indicating that the inspection has multiple reported items. The table contains properties such as *Created* – a date when the inspection was created, *Description* – a short description of the user, *UserId* – a user responsible for the inspection, and *RoomId* – a room on which the inspection was made.

Chapter 5

Logic Layer (API)

The logic layer is made up of the Application Programming Interface (API). API is a kind of proxy between the presentation layer and the data layer. It accepts a request from the presentation layer, decides whether it passes all the security measures, and then computes the request by accessing the data in the data layer over a secure line. Upon reception of the data, API generates a suitable response to the request and subsequently sends the response back to the presentation layer with the requested data. The source code is available on GitHub [1].

API uses an MVC architecture (see section 2.4). As a result, the code is split into multiple small files, with each file containing the code relevant to its filename. Although it may appear to be an unnecessary partition initially, an experienced developer can easily identify and fix bugs in the code because of the modularity.

5.1 Server

It is not possible to implement the API server on the same server as the website. This is because the website may experience performance issues when the API server is under high load. To address this issue, the server has its own IIS server (see section 3.14). This provides additional security as the IIS can be configured with Cross-Origin Resource Sharing (CORS) to allow or block communications only from specific sources, making it difficult for attackers to target the API server with DDoS attacks. So the API server will only allow sources from the website and mobile application and nothing else.

5.2 Asynchronous programming

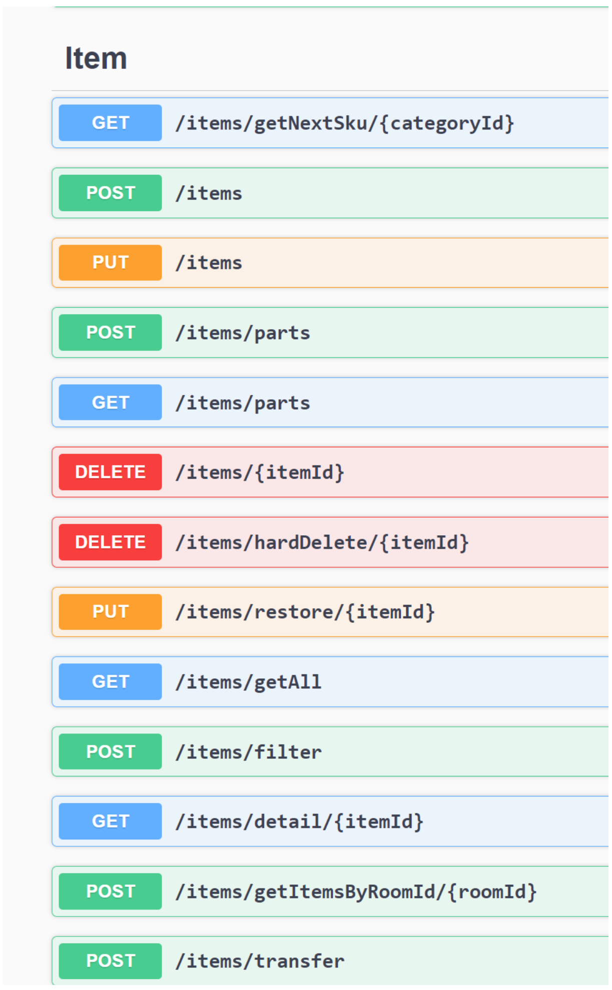
Asynchronous programming can be a powerful tool when used correctly. It enables the application to use thread resources. It is useful for network requests, file I/O or database access. Synchronous execution of the tasks may cause the application to become unresponsive due to the overloading of the main thread, which is responsible for executing the main cycle. During task computation, the main thread pauses the application cycle until the workload is complete. The power of asynchronous programming

[1]GitHub: `https://github.com/VagnerVit/inventory-api`

lies in its ability to distribute the workload among the available threads and relieve the main thread of its duties. By dividing the workload among available threads and reducing the burden on the main thread, asynchronous programming can improve the performance of applications.

5.3 Endpoints

The main concept of the API server is its endpoints. It provides the presentation layer with a way to receive data. HTTP requests are used to 'call' each endpoint with a specific URL. Clients, such as the website and mobile application, can access and interact with these endpoints. They can use HTTP methods like GET, POST, PUT and DELETE via Axios (see section 3.11) and Refit (see section 3.8) to perform various actions on these resources.



Item

GET /items/getNextSku/{categoryId}
POST /items
PUT /items
POST /items/parts
GET /items/parts
DELETE /items/{itemId}
DELETE /items/hardDelete/{itemId}
PUT /items/restore/{itemId}
GET /items/getAll
POST /items/filter
GET /items/detail/{itemId}
POST /items/getItemsByRoomId/{roomId}
POST /items/transfer

Figure 5.1: Endpoints working with *Item* entity

5.4 Security Measures

Security is crucial on the internet. Therefore, this section provides an in-depth explanation of every security measure within the application.

5.4.1 Authentication

Authentication is performed using Bearer authentication [24]. Bearer authentication is a type of OAuth 2.0 token authentication. This application uses a JSON Web Token (JWT) as the bearer token. Thus, information that needs to be encrypted, like the *UserRole* or *Username*, can be encrypted in the token. When a request is sent from the presentation layer to the API, a token is included as part of the security measures. Subsequently, the API verifies the token's validity and checks whether it has expired. In the event of an invalid token, the API will refuse access and respond with HTTP error code 401 – Unauthorised. If the token is verified, it is decrypted by the API to reveal the associated username with *UserRole* and checks if the user has permission for the request. If the user does not have the required permission, access is denied, and the API responds with HTTP error code 401.

5.4.2 Password Encryption

Storing passwords as plain text is considered the worst programming practice. Passwords must be encrypted because relying on their security is impractical, so the next security measure is to encrypt the password. The standard encryption method involves hashing user passwords with the SHA-256 cryptographic function. Hashes differ significantly when hashing similar passwords while remaining identical when hashing two identical passwords with the same function. This explains the role of cryptographic salt in encryption. The cryptographic salt is a unique combination of letters, symbols and numbers randomly generated. The cryptographic salt for each user is stored in the User table (see section 4.2.1), and then the application takes the username, salt and password and uses the SHA-256 hash function. Due to these precautions, breaking or replicating the hash to gain access to the password is highly unlikely.

5.4.3 User Roles

The introduction of user roles is a quick and reliable security measure. User roles are represented by a structure called the *enum*. The user table in the database contains a column called *UserRole*, which has an integer data type. The *enum* represents a name/value pair, which means the code implementation is limited to text-based only. The text is translated into numbers that correspond to the enum pair during the compilation of the code.

The API implementation is accomplished by adding the user role to the bearer token. Subsequently, a security layer can be established by extending the endpoint implementation with the parameter *[Authorize("<name of the user role>")]*, which first verifies the token. Afterwards, it decrypts the token, confirms the user role and then allows access to the endpoint.

Implementation of inherited roles from lowest to highest.

1. **User:** General role for using the site. This role has limited access to the administration and the mobile application.
2. **Mobile:** This role inherits the access of the User role and also has access to the mobile application.
3. **Administrator:** Access to the Administration page and, therefore, access to the whole system.

5.5 Barcode Generation

The barcode is described in the section 3.5.

The following outlines the process for creating a barcode: The *ItemPart* is selected by the user. The *ItemPart* has a property *Category*, which determines what the code prefix will look like. The *Category* has a *skuPrefix* property which establishes the two labels. The labels are predefined when the *Category* is created. At the moment, the barcode looks like this 'XX', where XX stands for the two labels. After that, a hyphen is added to separate the labels from the rest of the upcoming barcode, and the barcode becomes 'XX-'. Next, the API checks which highest number belongs to the *Category*. If there are no items generated under that *Category*, the number '0001' is added to the barcode. If there are items under this Category, API finds the highest and increments it by one. This number is added to the barcode, which looks like 'XX-YYYY', where YYYY is the generated number, and the generation is complete.

ZD-0001

Figure 5.2: Example of SKU code 'ZD-0001'

Chapter 6

Presentation Layer (Website)

The data and logic layers are established. Once the data is stored in the database and handled by the API server, the website is one of the places where it can be displayed. The website should provide clear ways for users to work efficiently and effortlessly with the data, allowing them to create, read, update and delete specific records. The source code is available on GitHub [1].

6.1 Communication with API

API communication is provided by Axios (see section 3.11). Axios is a JavaScript library that simplifies making HTTP requests from the browser. It provides an interface that offers simple HTTP methods such as get, post, put and delete. Axios can extend the request header with a *Content-Type* property having the value *'application/json'* to enable sending of a JSON object through an HTTP request. It also implements a promise-based workflow, which handles HTTP requests asynchronously. The website can respond after a successful request or when an error occurs.

6.2 Data Formatting

Upon receiving the raw data via API request, the data must undergo additional formatting and manipulation to display the data. This formatting and manipulation is done using TypeScript. The code filters the raw data and then formats it visually.

6.3 Pages

Every website has pages to distinguish different structures and use cases that users can perform. There are multiple types of page layouts. The two most commonly discussed layouts are the one-page layout and the multi-page layout. The one-page layout, as the name suggests, consists of a single page that can be scrolled and contains all the data within the page. This design is solely for presentation purposes and is not suitable in this case. This website uses a multi-page design. It includes a menu that

[1]GitHub: `https://github.com/VagnerVit/inventory-web`

directs users to different pages. Normally, a single page serves a single purpose to align with the design of a multi-page layout.

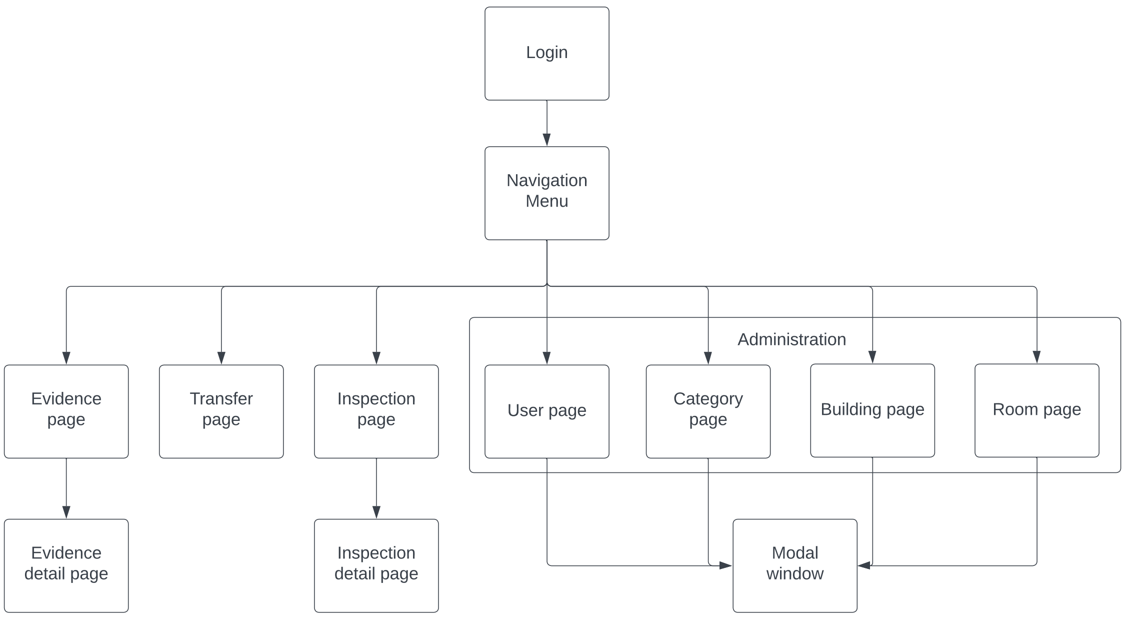


Figure 6.1: Navigation diagram

6.3.1 Login Page and Navigation Menu

The Login page allows users to log in, but this is not the end of the process. After logging in, the users can access the navigation menu, which contains pages they can interact with. The pages are shown based on the users' system access. If users access a page they do not have access to, they are returned to the login page. The page contains one use case – UC1 (see section 1.3.1).

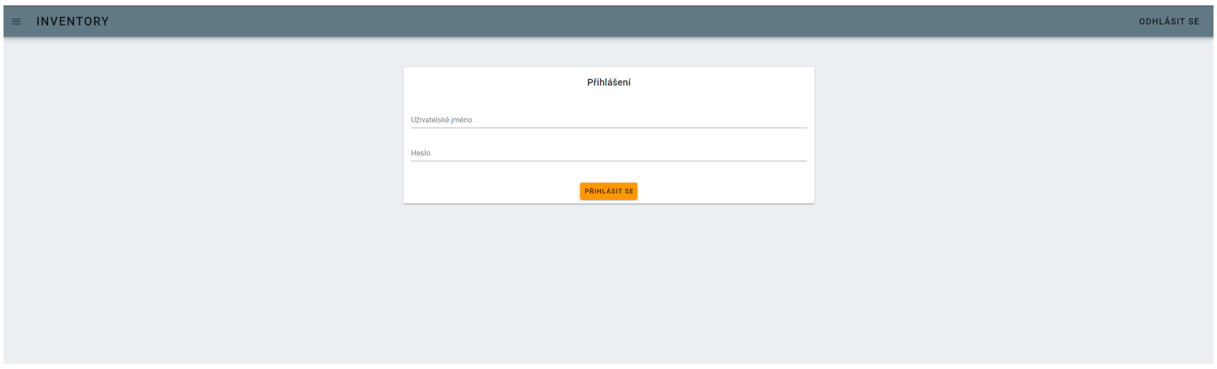
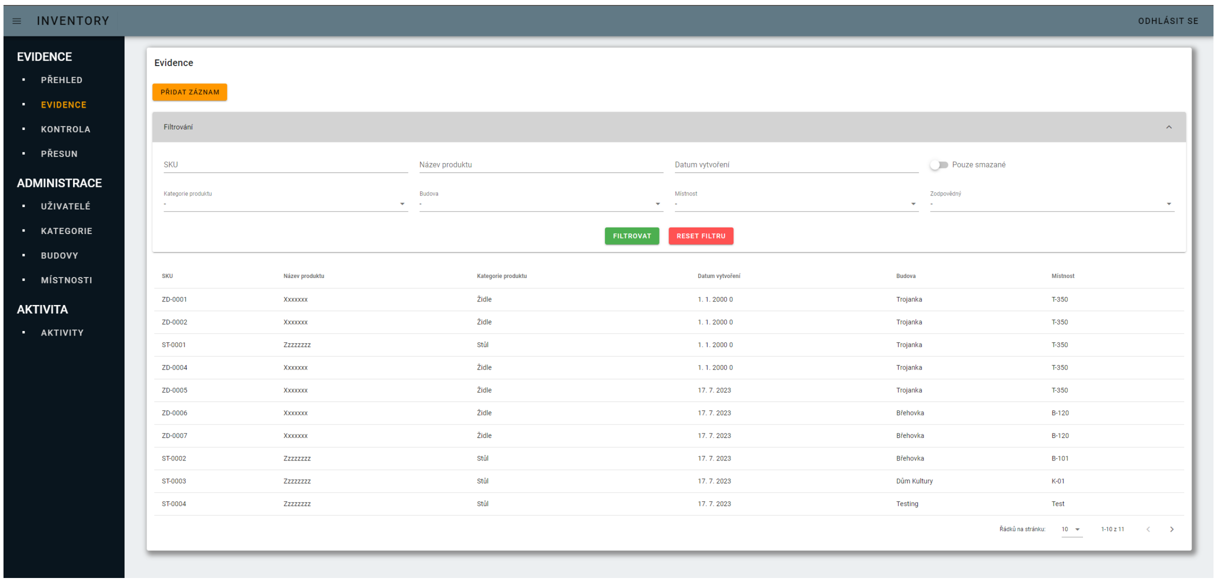


Figure 6.2: Login page

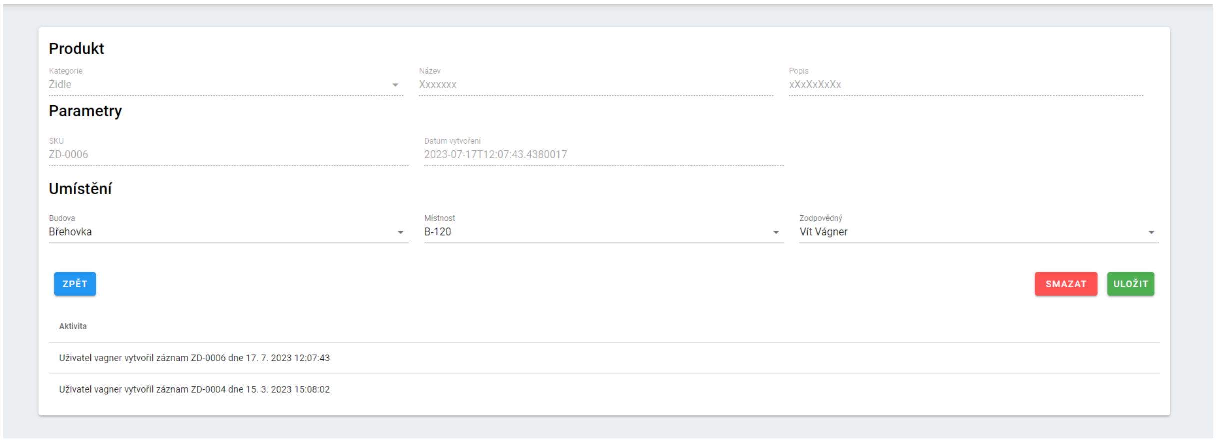
6.3.2 Evidence Page

The Evidence page displays *Item* records. The first feature introduced is a filter, which enables users to sort and filter records based on various values. Underneath the filter, there is a table that shows data of the *Item* records, such as *SKU*, *Name*, *Category*, date created, or where the item is located. By clicking on a record in the table, the user is taken to a subpage that provides a detailed overview of the selected *Item* record. On the subpage, the record can be viewed in greater detail, and the user can edit some of its values. The page contains use cases – UC6 (see section 1.3.6), UC7 (see section 1.3.7) and UC8 (see section 1.3.8).



SKU	Název produktu	Kategorie produktu	Datum vytvoření	Budova	Místnost
ZD-0001	Xxxxxxx	Židle	1. 1. 2000 0	Trojanka	T-350
ZD-0002	Xxxxxxx	Židle	1. 1. 2000 0	Trojanka	T-350
ST-0001	Zzzzzzzz	Stůl	1. 1. 2000 0	Trojanka	T-350
ZD-0004	Xxxxxxx	Židle	1. 1. 2000 0	Trojanka	T-350
ZD-0005	Xxxxxxx	Židle	17. 7. 2023	Trojanka	T-350
ZD-0006	Xxxxxxx	Židle	17. 7. 2023	Břehovka	B-120
ZD-0007	Xxxxxxx	Židle	17. 7. 2023	Břehovka	B-120
ST-0002	Zzzzzzzz	Stůl	17. 7. 2023	Břehovka	B-101
ST-0003	Zzzzzzzz	Stůl	17. 7. 2023	Dům Kultury	K-01
ST-0004	Zzzzzzzz	Stůl	17. 7. 2023	Testing	Test

Figure 6.3: Evidence page



Kategorie: Židle
Název: Xxxxxxx
Popis: xXxXxXxXx

SKU: ZD-0006
Datum vytvoření: 2023-07-17T12:07:43.4380017

Budova: Břehovka
Místnost: B-120
Zodpovědný: Vít Vágner

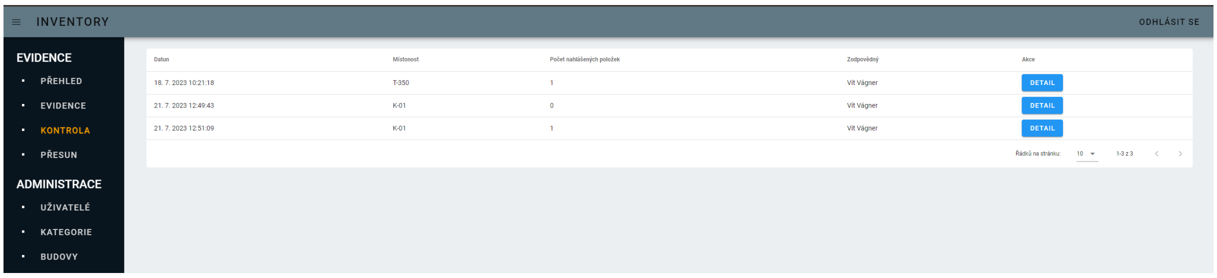
Uživatel vagner vytvořil záznam ZD-0006 dne 17. 7. 2023 12:07:43

Uživatel vagner vytvořil záznam ZD-0004 dne 15. 3. 2023 15:08:02

Figure 6.4: Evidence page – a detail

6.3.3 Inspection Page

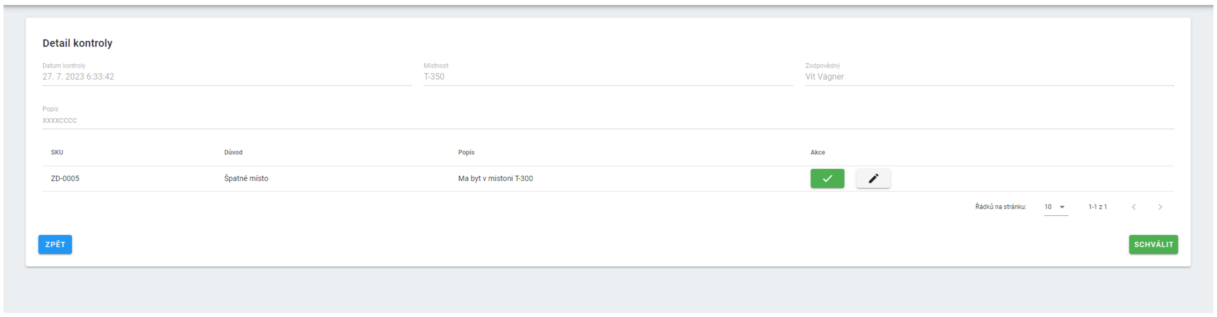
The Inspection page displays individual inspections performed in UC12 (see section 1.3.12). The inspection is carried out via the mobile application and processed on the website. The inspection process is as follows: the inspection details are displayed by clicking on the inspection item. The inspection record provides details about the date and location of the inspection, as well as the name of the inspector. The inspection report includes a table of reported items that have been reported during the inspection. Every reported item is editable, and after modification, it can be labelled as resolved and deleted from the reported items table. Once all the reported issues are resolved, the user can close the inspection.



Datun	Místonost	Počet nahlášených položek	Zodpovědný	Akce
18. 7. 2023 10:21:18	T-350	1	Vít Vágner	DETAIL
21. 7. 2023 12:49:43	K-01	0	Vít Vágner	DETAIL
21. 7. 2023 12:51:09	K-01	1	Vít Vágner	DETAIL

Řádků na stránku: 10 1-3 z 3

Figure 6.5: Inspection page


Detail kontroly

Datum kontroly
27. 7. 2023 6:33:42

Místnost
T-350

Zodpovědný
Vít Vágner

Popis
XXXXCCCC

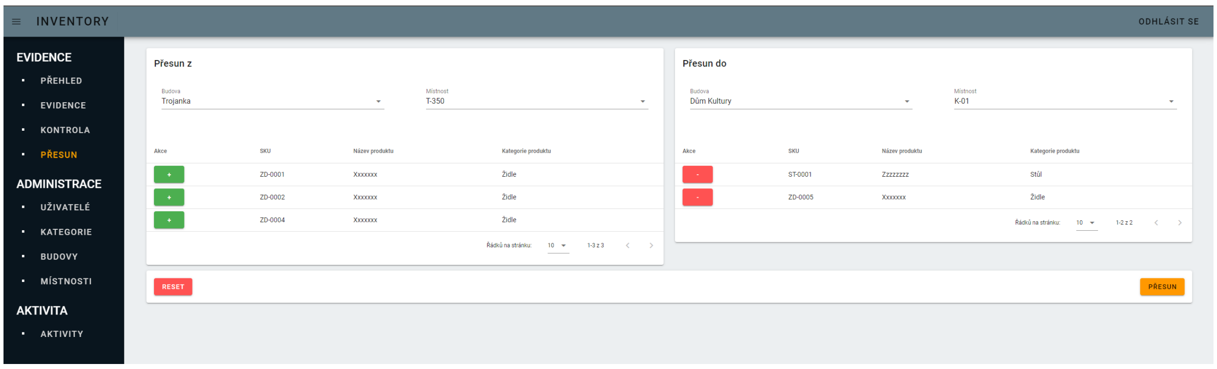
SKU	Důvod	Popis	Akce
ZD-0005	Špatné místo	Ma byt v mistoni T-300	✓

Řádků na stránku: 10 1-1 z 1

Figure 6.6: Inspection page – a detail

6.3.4 Transfer Page

Record transfer is a daily routine. Therefore, the web application has a distinct page for the transfer process. Two columns are available. The left column allows the user to select the location from which the records are to be transferred, and the right column shows the records to be transferred to the desired location. To select records to move, click the '+' button, and to remove the records to be moved, click the '-' button. The page contains one use case – UC9 (see section 1.3.9).



Přesun z

Budova: Trojanka
Místnost: T-350

Akce	SKU	Název produktu	Kategorie produktu
+	ZD-0001	Xxxxxxx	Židle
+	ZD-0002	Xxxxxxx	Židle
+	ZD-0004	Xxxxxxx	Židle

Řádků na stránku: 10 1-3 z 3

Přesun do

Budova: Dům Kultury
Místnost: K-01

Akce	SKU	Název produktu	Kategorie produktu
-	ST-0001	Zzzzzzzz	Stůl
-	ZD-0005	Xxxxxxx	Židle

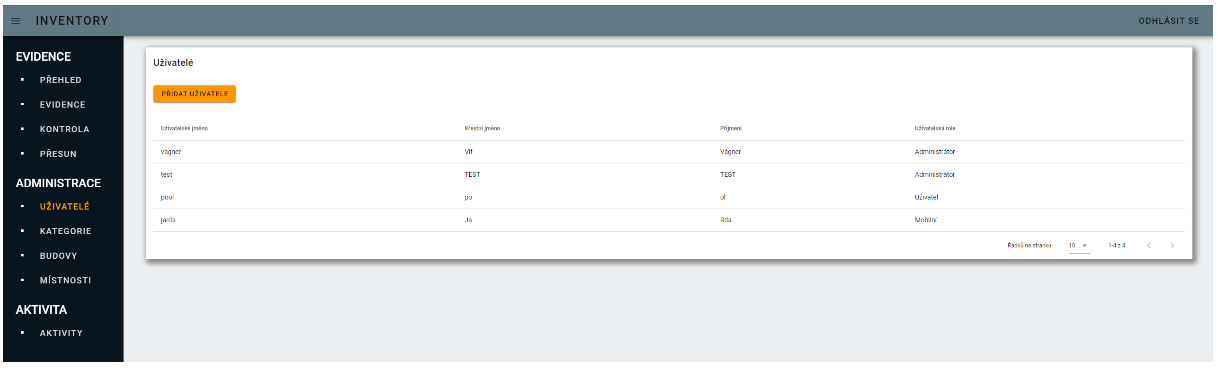
Řádků na stránku: 10 1-2 z 2

Figure 6.7: Transfer page

6.3.5 Administration Pages

The administration is divided into several pages, depending on what the administrator wants to manage. The administration is available for users, categories, buildings and rooms. To access the administration pages, the user must have a user role with the value *'Admin'*.

The layout of the pages for managing users, categories, and rooms is similar. A filter system is present for sorting records, as well as a table of key values for identifying records. Clicking on a record opens a pop-up window. The administrator can modify values, delete and recover records, or view an activity log that displays the changes made to records, all from the pop-up window. There is a button above the filtering system to add a new record. By clicking on this button, a modified pop-up window will open, displaying only the values required to create the record.

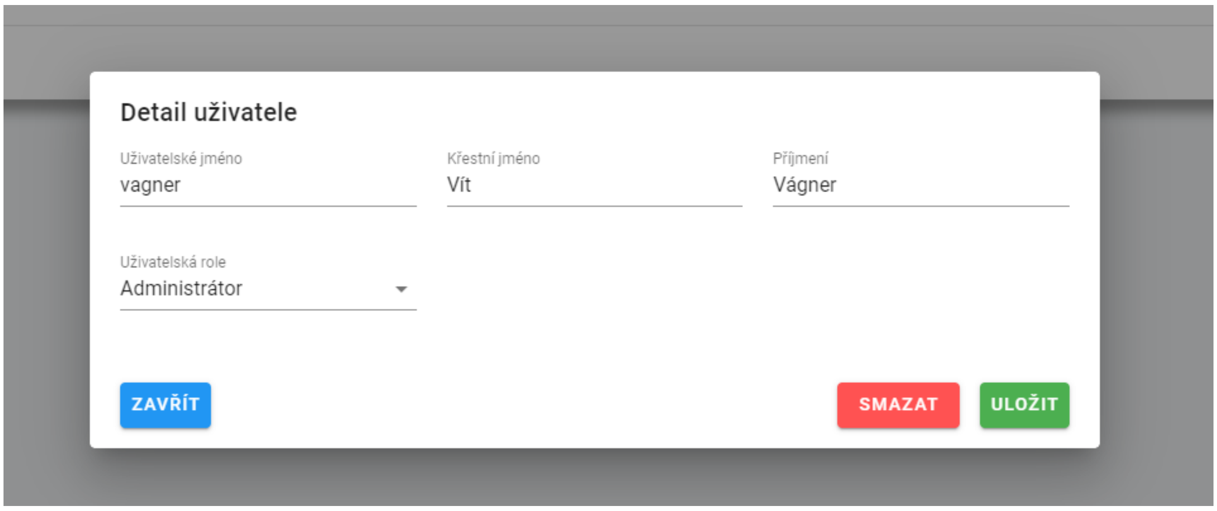


Uživatelé

Uživatelské jméno	Křestní jméno	Příjmení	Uživatelská role
vagner	Vít	Vágner	Administrátor
test	TEST	TEST	Administrátor
pool	po	ol	Uživatel
jarda	Ja	Rda	Mobilní

Řádků na stránku: 10 1-4 z 4

Figure 6.8: Administration page – users


Detail uživatele

Uživatelské jméno: vagner

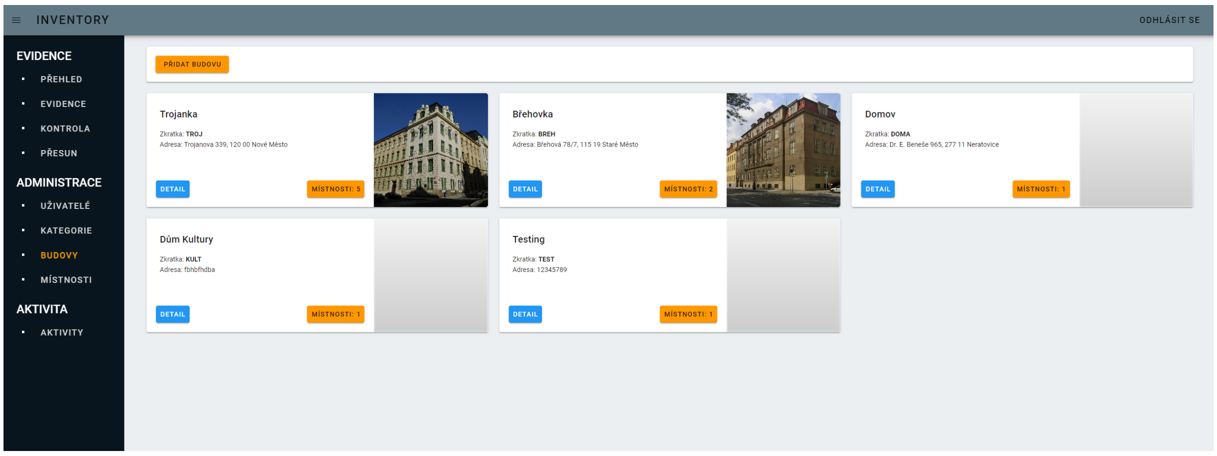
Křestní jméno: Vít

Příjmení: Vágner

Uživatelská role: Administrátor

Figure 6.9: Administration page – a detail

The building administration page differs from the others due to its card layout. Instead of a table, the layout of cards displays the name, abbreviation, address and photo of the building. Additionally, the card features two buttons; the 'Detail' button that opens the mentioned pop-up window and the right button that opens the room administration page, where the filtration system displays the rooms of the selected building. The pages contain use cases – UC2 (see section 1.3.2), UC3 (see section 1.3.3), UC4 (see section 1.3.4), UC5 (see section 1.3.5), UC7 (see section 1.3.7) and UC8 (see section 1.3.8).



Trojanka
Zkratka: **TROJ**
Adresa: Trojanova 339, 120 00 Nové Město
DETAIL MÍSTNOSTI: 5

Břehovka
Zkratka: **BREH**
Adresa: Břehová 78/7, 115 19 Staré Město
DETAIL MÍSTNOSTI: 2

Domov
Zkratka: **DOMA**
Adresa: Dr. E. Beneše 965, 277 11 Neratovice
DETAIL MÍSTNOSTI: 1

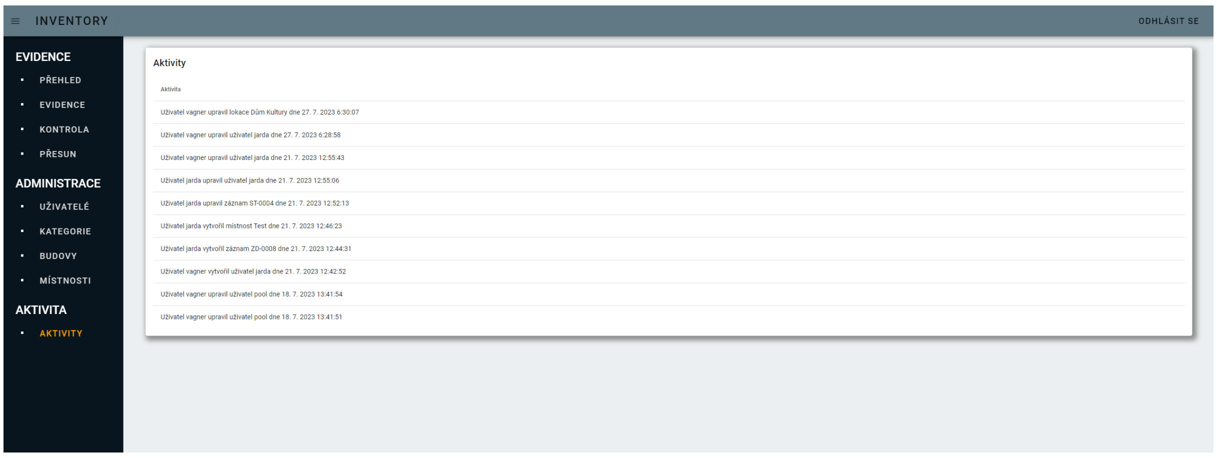
Dům Kultury
Zkratka: **KULT**
Adresa: fbhbfhdba
DETAIL MÍSTNOSTI: 1

Testing
Zkratka: **TEST**
Adresa: 12345789
DETAIL MÍSTNOSTI: 1

Figure 6.10: Administration page – buildings

6.3.6 Activity Page

A table is displayed on the Activity page. Each row in the table indicates whether a record has been added, modified, deleted, or transferred. Each row also displays the performer and timestamp of the action.



Aktivity

Aktivita
Uživatel vagner upravil lokace Dům Kultury dne 27. 7. 2023 6:30:07
Uživatel vagner upravil uživatel jarda dne 27. 7. 2023 6:28:58
Uživatel vagner upravil uživatel jarda dne 21. 7. 2023 12:55:43
Uživatel jarda upravil uživatel jarda dne 21. 7. 2023 12:55:06
Uživatel jarda upravil záznam ST-0004 dne 21. 7. 2023 12:52:13
Uživatel jarda vytvořil místnost Test dne 21. 7. 2023 12:46:23
Uživatel jarda vytvořil záznam ZD-0008 dne 21. 7. 2023 12:44:31
Uživatel vagner vytvořil uživatel jarda dne 21. 7. 2023 12:42:52
Uživatel vagner upravil uživatel pool dne 18. 7. 2023 13:41:54
Uživatel vagner upravil uživatel pool dne 18. 7. 2023 13:41:51

Figure 6.11: Activity page

Chapter 7

Presentation Layer (Mobile Application)

Developing a mobile application alongside a website can improve the overall user experience in many ways. The mobile application is tailored to mobile devices, offering a more individualised experience and enabling new use cases for the inventory system. Furthermore, device-specific features like touch gestures and camera integration can be utilised. The user can take advantage of these mobile applications while carrying out inventory inspections. This application is cross-platform and can run on both Android and iOS devices. The MAUI framework (see section 3.6) ensures a single code base for Android and iOS. Moreover, if the application needs to scale to desktops, MAUI could be used to develop desktop applications for Windows and MacOS. The source code is available on GitHub [1].

7.1 Communication with API

Communication is managed through Refit (see section 3.8). Refit makes communication with the API seamless and efficient, allowing the developer to focus on business logic rather than low-level HTTP details. The API endpoints are represented by declaring the service interface. The endpoints are specified with request type, URL, path, query parameters, headers, and expected response type. The service is registered through dependency injection.

7.2 UI Design

The application can have a large number of interesting functionalities. However, without a tested, modern design, the functionality will not be usable because the user will be overwhelmed or lost. Therefore, implementing Material Design gives the user an excellent and intuitive experience by implementing several points.

- **Icons** provides intuitive recognition of the activity represented by the button. The user will know what the button does without having to read any text. The rule results in a less cluttered page because the icons support the text.

[1] GitHub: `https://github.com/VagnerVit/inventory-mobile`

- **Colour Pallet** gives the user clear pages without intrusive colours. It matches the colours of the actions on the buttons to indicate what action is being performed. A green button indicates a successful action that saves something, and a red button indicates an action that reports or deletes something.

7.3 Barcode Scanning

The Barcode scanning process utilises the ZXing.Net.MAUI library (see section 3.15). The barcode is scanned with the mobile camera. The user aligns the camera with the barcode, and then the camera detects the barcode and reads the barcode value.

7.4 Pages

Pages can be described as pages on the website, although in mobile applications, there is no URL to interact with. The mobile application performs the redirection in code that is invisible to the user. Some pages have subpages that act as modal windows. The way this works is that the page has the role of a root page, i.e. when the modal window is opened, and the user presses the return button, it takes the user back to the root page, and the root page remains unchanged.

7.4.1 Navigation Menu

The navigation panel is placed at the bottom after the user logs in. There are Home, Inventory Check and New Record pages. The pages are divided into three sections, as shown in the figure. Navigation is done by clicking on the text representing the page, and the user is taken there.

7.4.2 Login Page

Similar to the website Login page (see section 6.3.1), the mobile Login page has a username and password field. The mobile application has a stricter form. While accessing the website, the user can view the complete page, whereas, in the mobile application, only the login page is visible without any URL. Once the login process is complete, the user is directed towards the Home page, which has a visible navigation menu.

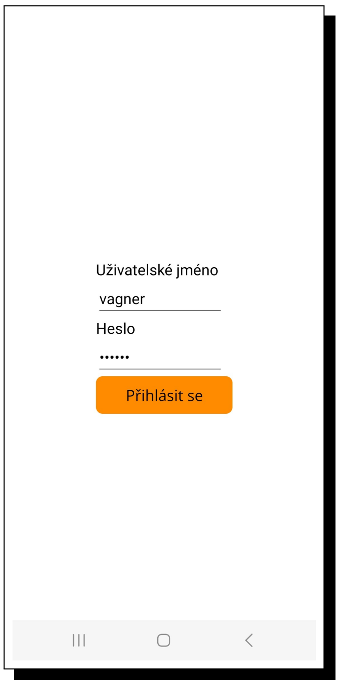


Figure 7.1: Login page

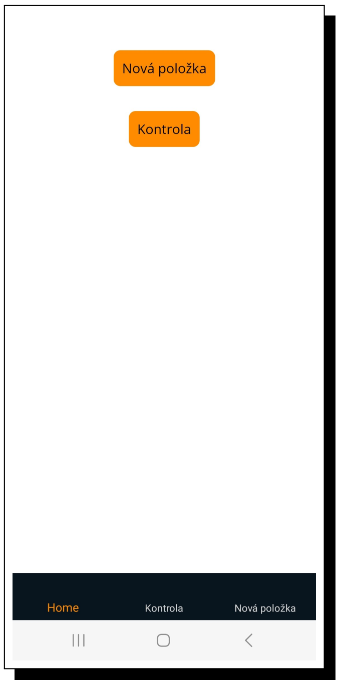


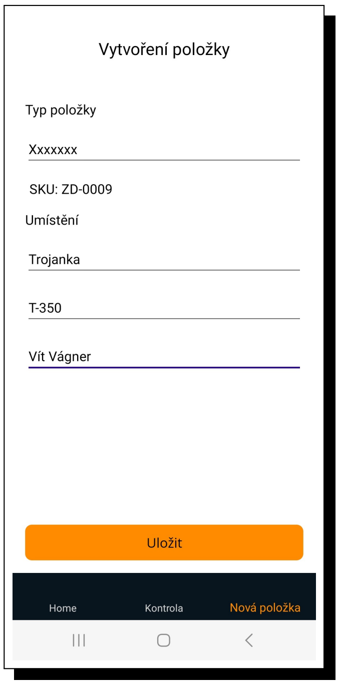
Figure 7.2: Home page

7.4.3 Home Page

The Home page provides users with a navigation menu, essential information and a few buttons for interaction.

7.4.4 New Item Page

This page offers a shortened version of item creation. For the administrator's ease of use, it is recommended that item creation be predominantly done on the website since there are also administration pages available. Due to the frequency of item creation, it is essential to include it as a feature in the mobile application. This page serves this purpose. On this page, users can choose *ItemPart* to create an SKU code and determine the new item's location and responsible personnel.


Vytvoření položky

Typ položky

Xxxxxxx

SKU: ZD-0009

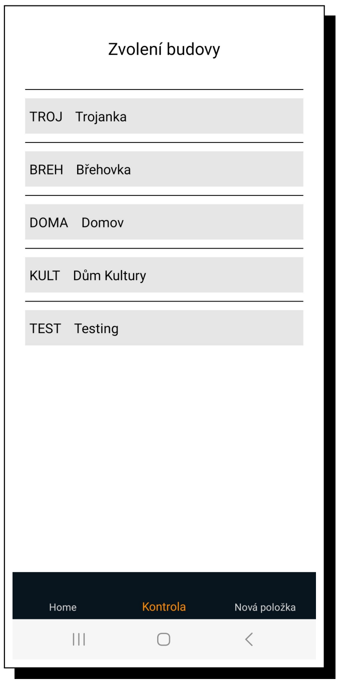
Umístění

Trojanka

T-350

Vít Vágner

Figure 7.3: New Item page

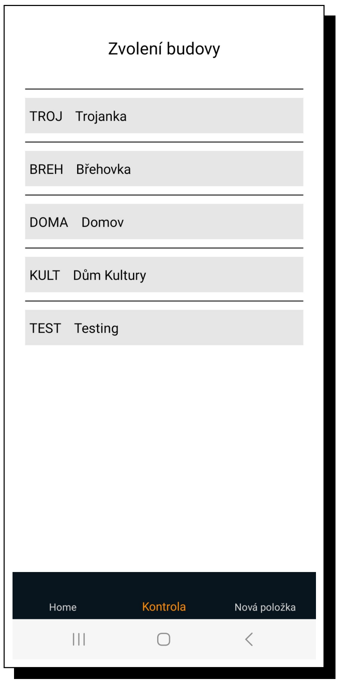

Zvolení budovy

TROJ	Trojanka
BREH	Břehovka
DOMA	Domov
KULT	Dům Kultury
TEST	Testing

Figure 7.4: Selection of buildings

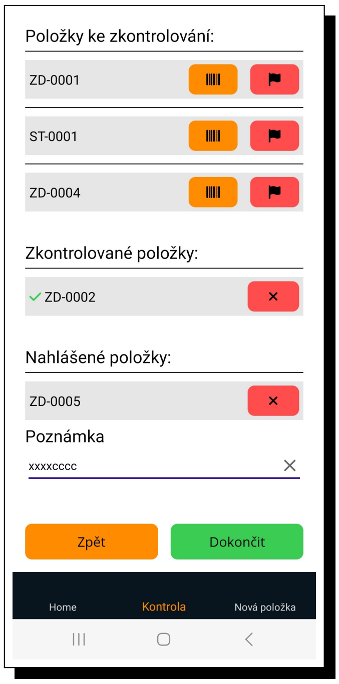
7.4.5 Inspection Page

The Inspection page displays a list of available locations. The user chooses the apt building and room to perform the inspection.


Zvolení budovy

TROJ	Trojanka
BREH	Břehovka
DOMA	Domov
KULT	Dům Kultury
TEST	Testing

Figure 7.5: Selection of rooms


Položky ke zkontrolování:

ZD-0001

ST-0001

ZD-0004

Zkontrolované položky:

✓ ZD-0002

Nahlášené položky:

ZD-0005

Poznámka

xxxxcccc

Figure 7.6: Inspection page

Several lists are displayed during the inspection, including a list of unchecked items, a list of checked items, and a list of reported items. To inspect an item, click on the button with the barcode icon. The button opens a modal window that includes a barcode scanner. When opening the modal window, the user will see which barcode to scan and can then scan the appropriate one. When the correct barcode is scanned, the item will be moved to the list of checked items. A notification will indicate the unsuccessful scan if the barcode does not match. Sometimes, the item may not be found within the room. The user can report this issue by clicking the button, which is marked with a flag. When clicked, this button opens a modal window that includes a field for users to fill in the reason and description for the report. After the description has been submitted, users must click the 'Report' button to send the report and move the item to the list of reported items. In case of an error, the checked or reported item can be removed from the list by clicking on 'X' located next to the item in the list.

Once all items have been inspected, the user enters a description of the inspection and then clicks on the 'Complete' button in green. After processing the request, the inspection is then sent to the API and stored in the database.

The website allows viewing of the inspection, and reported items can be edited as per the report. The Inventory Check page displays a table containing available locations.

Chapter 8

User Acceptance Testing

Testing is a process that aims to ensure that the inventory system is user-friendly, intuitive, and, above all, useful. Three individuals were selected to examine the system. Individuals are from diverse backgrounds and have varying levels of computer proficiency. Four distinct tasks were developed. The individuals tested the workflow of these tasks by following the user manual. After testing, the participants evaluated and reviewed the difficulty level of completing the tasks.

8.1 Testers

8.1.1 Person A

Person A is employed as a caretaker at a school. They are familiar with inventory control procedures and should be able to assess the inventory check procedure in the rooms. An opinion is required from this person on whether the application would be a suitable replacement for the current systems.

8.1.2 Person B

Person B is an elderly adult who possesses limited computer skills. It is crucial to obtain feedback from individuals who are not considered professionals, in addition to the experts, as the application's eventual users remain unknown.

8.1.3 Person C

Person C is a full-stack software developer. Person C has experience working with comparable systems, primarily in developing them, and possesses the capacity to scrutinise the internal workings and architecture of the application.

8.2 Tasks

The tasks demonstrate the standard use of the application. These tasks include creating an item, transferring an item, inspecting the items in the room, and resolving conflicts in the inspection process.

8.2.1 Creating an item

The task involves creating a new item. This requires navigating to the evidence page where the creation is hosted, filling in the form, and selecting an *ItemPart*. If the *ItemPart* is missing, testers will create it. Next, testers select the building and room where the item will be placed.

8.2.2 Transferring an item to a different location

There are two ways to transfer the item.

The first method is to use the Transfer page. This involves selecting the source location (building and room) and destination location. After selecting the locations, the transfer will commence. The website provides the functionality for users to transfer items between the source and the destination.

Alternatively, one can access the Evidence page, apply the necessary filters to find the desired items, and then move to the Evidence detail page to modify them as necessary. Nevertheless, if there are multiple items to be transferred, the first method is recommended. It is to be noted that the process can only be performed individually for each item on the Evidence page.

8.2.3 Inspecting items in the room

The process is performed through the mobile application. The user navigates the mobile application to review all of its features. The main feature is inspecting items in the room, so the task's focus is on that. A room is provided to the user for inspection. When the user chooses the required room, the list of unchecked items is displayed. The user scans barcodes and verifies items as per the instructions. If an item is missing, the user must report it. After the process, the user writes a brief description and concludes the inspection.

8.2.4 Resolving conflicts in the inspection

The inspection is generated upon completion of the *'Inspecting items in the room'* task. Next, the user must address the inspection. Initially, the user must access the website and go to the Inspection page. The Inspection page displays a table of unprocessed inspections. The user chooses the unresolved inspection and examines it in detail. The inspection's details include a table showing reported items. The user can view a reason and a brief description of the reported item. The user can edit the reported item by clicking on the button with a pencil icon. Once the item is edited, it can be marked as resolved and removed from the table of reported items. The steps are repeated by the user until no more items are listed in the table. After the table has been cleared, the user can close the check.

8.3 Results

he reviews are categorized based on the tasks into four sections. Each tester expressed their perspective on the workflow, as well as the task's ease and intuitiveness. The reviews have been translated from Czech to English.

8.3.1 Person A's comments

Having worked as a caretaker in a school institution, he possesses considerable experience in managing inventory and ensuring seamless operations within the facility.

Creating an item

The application offers a user-friendly interface for creating new inventory items. It allows for the precise input of essential details such as the item name, description and location. The process is designed to be intuitive and easily comprehensible to users with minimal training. In addition, the application provides the ability to attach relevant images or documents to the items, which aids in reference and documentation efforts.

Transfer an item to a different location

The application enables the smooth transfer of items between different locations within the school. During the transfer process, the user needs to choose the relevant items and indicate both the source and destination locations.

Inspecting items in the room

Users can use the inspection feature to conduct room checks and verify the location of all items. The process includes scanning barcodes that are attached to items and verifying their presence. For each room, the application generates inspection reports that streamline the auditing process.

Resolving conflicts in the inspection

Conflicts may arise during inspections, such as inconsistencies between the actual inventory and the expected items listed in the system. The application provides a conflict resolution feature that enables users to record discrepancies and append comments.

Summary

The inventory management application is a reliable and efficient tool for the school institution. It performs well in various areas such as item creation, transfer, room inspections, and conflict resolution. The application's interface is user-friendly, enabling individuals with limited technical expertise to efficiently manage the inventory.

8.3.2 Person B's comments

Initially, he felt slightly apprehensive about utilising the computer for the inventory system application. However, he found the interface to be unexpectedly simple and was able to navigate it with ease.

Creating an item

The system's step-by-step instructions were followed by Person B. These instructions included entering the item details, category, and any other relevant information. Person B successfully created the item in the inventory after some experimentation.

Transfer an item to a different location

The objective was to move an item from its current location to a different one. The task was initiated by using the built-in transfer functionality on the Transfer page of the system. The transfer process was unproblematic, as a target room was selected to where the items would be transferred, and the destination location was specified. Afterwards, the items to be transferred were selected, and the transfer was confirmed.

Inspecting items in the room

The next step was to inspect the room. To do this, he used a mobile device to scan the barcodes of each item. Although he was initially hesitant to use a touch device, he navigated it intuitively. The system displayed the relevant information upon scanning each item, allowing him to report any conflicts, like an item being missing.

Resolving conflicts in the inspection

During the inspection, conflicts or discrepancies were discovered in the recorded information. The conflicts had to be resolved and the item data was updated accordingly. He successfully updated the item to reflect its current condition with some guidance from the user manual on the conflict resolution process.

Summary

It was conceivable that he could work with the system. Initially, the system appeared to be non-threatening and manageable. The tasks were easy to understand, and with the help of the manual, he was able to complete them without difficulty.

8.3.3 Person C's comments

The inventory system was found to be functional overall, but several areas require improvement to enhance its efficiency and user-friendliness.

Creating an item

Creating an item within the system was a straightforward process. However, having more validation checks during item creation would be beneficial. Currently, items do not have sufficient identifiers. Adding more fields to differentiate items would aid in preventing conflicts and enhancing data accuracy.

Transfer an item to a different location

Transferring items to different locations was a relatively simple task. Although it appeared puzzling to him initially, as he grew familiar with the process, it became more manageable. After performing the transfer a few more times, he gained a better understanding of how it works.

Inspecting items in the room

The inspection feature is a valuable addition to the system. Nevertheless, it lacks the flexibility to customise inspection checklists. Presently, inspections are generic and may not address all the unique requirements of individual rooms. Enabling the users to create custom inspection templates for each room type would significantly enhance the feature's usefulness.

Resolving conflicts in the inspection

No irregularities were found during the inspection. Upon accessing the inspection page to address any discrepancies, However, he faced limitations in exercising greater control over the inspection. Despite acknowledging that conflict resolution is infrequent, this is his personal perspective. Furthermore, he recommends implementing a system to keep an accurate audit trail and monitor the history of conflict resolution.

Summary

In conclusion, the inventory system shows promise as a reliable tool for managing items. By addressing the issues above and incorporating user feedback, it can be transformed into a strong and user-friendly inventory management solution.

8.4 Testing Summary

The reviews from the testers expanded the scope of the application. It facilitated a diverse perspective of the inventory system. The subsequent analysis resulted in the integration of some feedback from the reviews into the application. The changes were primarily focused on the UI since it is the most visible part. Typically, this refers to web pages and the mobile application's appearance and user experience.

The testers provided suggestions for additional features, but these were deemed less crucial for the basic functioning of the application. Consequently, the suggested features were duly noted but not executed. The implementation of these features can be postponed until the application is in use. The most commonly mentioned features were purchase orders and stock replenishment.

The testers expressed their satisfaction with the organisation. They appreciated the potential of the inventory system to improve the organisation and believed that with more use, it would become even more efficient at locating items.

They also mentioned some negative aspects regarding the initial impression of some individual tasks, but they were able to complete them by using the user manual. After becoming familiar with the workflow, it was a straightforward process. In order to prevent this issue, page tips could be introduced to guide the user through the task, instead of relying on the user guide.

Conclusion

Implementation of the inventory system in educational institutions has been successful overall. The application offers fundamental functions to transition from manual inventory to online inventory. This paper has identified several challenges in inventory management and, as a response, proposed a comprehensive system to address these issues.

The analysis consisted of reviews of competing software, providing insights into the expected functionality of the inventory system. The analysis is considered the most critical point in developing a software application since it outlines the system's foundations and helps to understand the impending challenges. Moreover, the analysis shapes the workflow, describes how it will be performed and identifies the use cases that the users will execute. The analysis considers and implements the theory of ideas, software architectures, designs, concepts and technologies used.

The multi-layer architecture was found to be the most suitable for the system. The logic layer addressed the communication between the two presentation layers and the database. Moreover, these separate layers provided enhanced security to the application while simplifying error detection.

Selecting appropriate technologies is a crucial aspect of development. In this paper, the technologies used and the reasons why they were chosen are mentioned. The technologies used can provide a lot of benefits to the application. However, if the chosen technologies cannot address the difficulties, problems, and challenges posed by the system, they can significantly impede development.

Data is managed and manipulated through the data and logic layers. The data layer is a database system responsible for storing the data, while the logic layer transfers the data from the database to the website or mobile application.

The presentation layer comprises the website and the mobile application. The website has an administration section for populating the system with data such as users, buildings, rooms, and categories. The website also offers an option to view, manage, and manipulate the data. The mobile application is designed for checking the inventory. Use cases that cannot be performed on the website, such as scanning the barcode, are available in the mobile application.

After implementing all the above, the inventory system was submitted for testing. Three individuals were selected for testing. The selected individuals performed the given tasks, and then the testers reviewed the workflow of these tasks.

The software's design allows for scalability. The application can be further developed and expanded in terms of functionality.

In conclusion, this bachelor thesis describes the development process of the inventory system for educational institutions and provides ready-to-use software. The source code of this inventory software can be found on GitHub – the website [1], the mobile application[2], and API[3].

[1]Website GitHub: https://github.com/VagnerVit/inventory-web
[2]Mobile Application GitHub: https://github.com/VagnerVit/inventory-mobile
[3]API GitHub: https://github.com/VagnerVit/inventory-api

Attachments

A. GitHub

Source code of API: https://github.com/VagnerVit/inventory-api.
Source code of the website: https://github.com/VagnerVit/inventory-web.
Source code of the mobile application: https://github.com/VagnerVit/inventory-mobile.

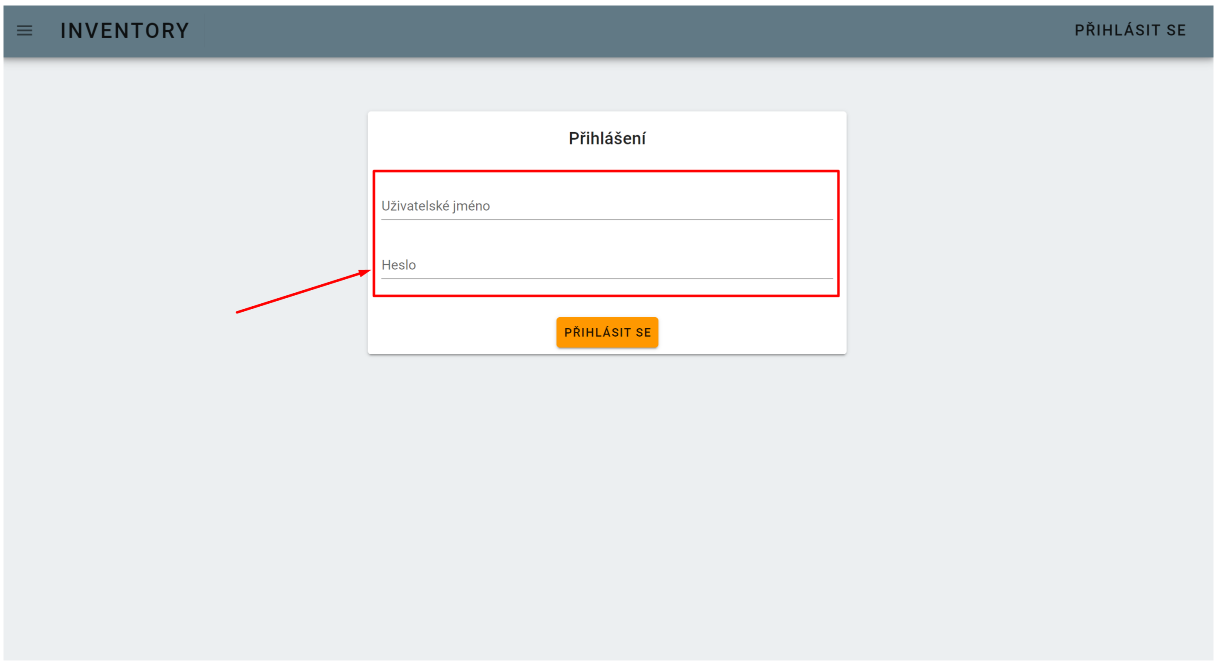
B. User Manual

The user manual describes each use case in the Czech language with corresponding pictures.

UC1 – Přihlášení

Přihlášení je důležitou součástí používání aplikace. K vykonání ostatních úkolů je vyžadováno být přihlášen.

Přihlášení do webové stránky
Nemáte-li přihlašovací údaje, kontaktujte administrátora systému. Jakmile se dostanete na přihlašovací stránku, vyplníte přihlašovací údaje, a poté kliknete na oranžové tlačítko "Přihlásit se". Po přihlášení se dostanete na stránku Evidence a v navigaci po levé straně se vám otevře nabídka jednotlivých stránek.



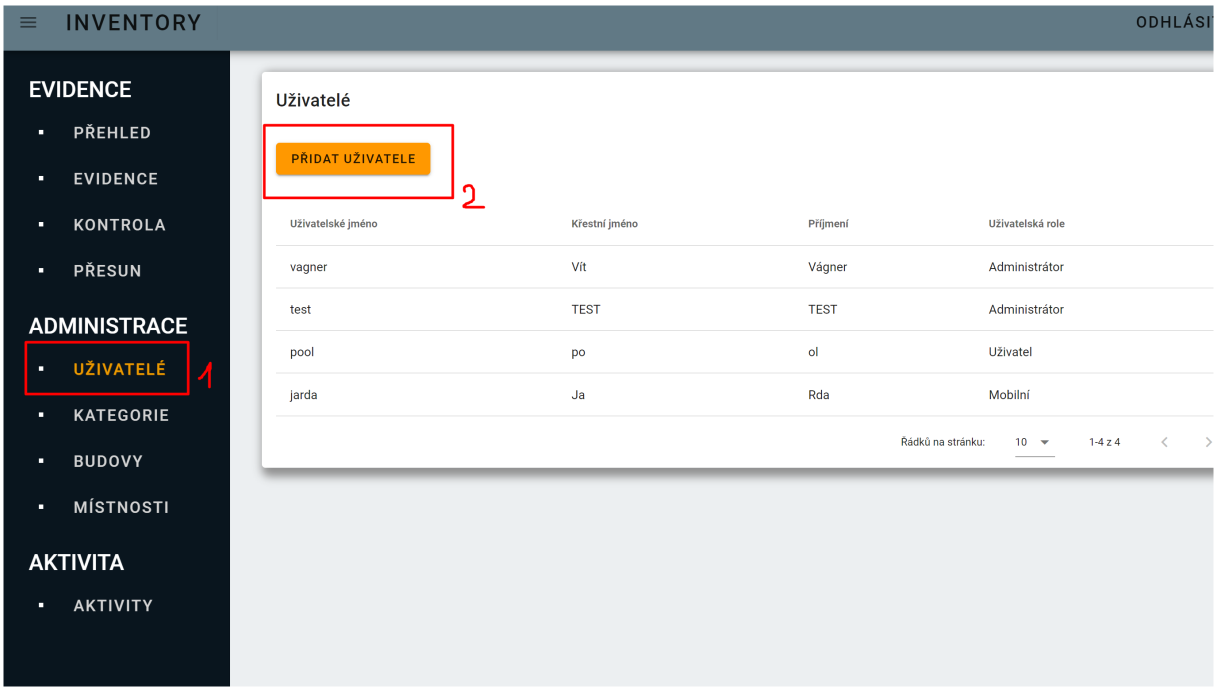
Obrázek 8.1: Ukázka v krocích průběhu UC1 - webová stránka

Přihlášení do mobilní aplikace
Po zapnutí aplikace se dostanete na přihlašovací stránku. Zde vyplníte své přihlašovací údaje, a poté se přihlásíte.

UC2 – Vytvoření uživatele

Vytvářet uživatele může pouze uživatel s rolí *"Admin"*, protože pouze administrátor má přístup do administrátorské sekce na webu.

K vytvoření uživatele je zapotřebí v navigačním panelu po levé straně v sekci ADMINISTRACE zvolit odkaz UŽIVATELÉ, viz postup na obrázku 8.2. Pro přidání nového uživatele kliknete na tlačítko "Přidat Uživatele"a otevře se vám vyskakovací okénko. V okénku je několik políček ohledně údajů uživatele. Uživatelské jméno je jméno, pod kterým se bude uživatel přihlašovat do systému. Nastavíte uživatelskou roli, poté vyplníte heslo uživatele a kliknete na "Uložit"a tím máte uživatele založeného.



Uživatelské jméno	Křestní jméno	Příjmení	Uživatelská role
vagner	Vít	Vágner	Administrátor
test	TEST	TEST	Administrátor
pool	po	ol	Uživatel
jarda	Ja	Rda	Mobilní

Řádků na stránku: 10 1-4 z 4

Obrázek 8.2: Ukázka v krocích průběhu UC2

UC3 – Vytvoření kategorie

K vytvoření kategorie je zapotřebí mít uživatelskou roli *"Admin"*.

V navigačním panelu po levé straně v sekci ADMINISTRACE zvolte odkaz KATEGORIE. Kliknutím na tlačítko se otevře okénko, kde jsou sloupce Název kategorie a Kód kategorie, který představuje, jaký kód budou mít jednotlivé kategorie přiřazený. Po vyplnění stačí stisknout tlačítko "Uložit"a kategorie je vytvořena.

UC4 – Vytvoření místnosti

K vytvoření místnosti je zapotřebí mít uživatelskou roli *"Admin"*.

V navigačním panelu vlevo v sekci ADMINISTRACE klikněte na tlačítko MÍSTNOSTI. Po otevření stránky zvolíme tlačítko "Přidat místnost". Kliknutím se otevře okénko se dvěma políčky. První políčko definuje, v jaké budově se místnost nachází. Není-li žádná budova k dispozici, nejprve se musí budova vytvořit. Přerušíme krok vytvoření místnosti, vrátíme se zpět a přejdeme na *"UC5 – Vytvoření budovy"*. Po vytvoření budovy se můžeme navrátit do stavu, kde jsme se nacházeli předtím a dokončit vytvoření místnosti. Vyplníme jméno místnosti, klikneme na tlačítko "Uložit"a máme hotovo.

UC5 – Vytvoření budovy

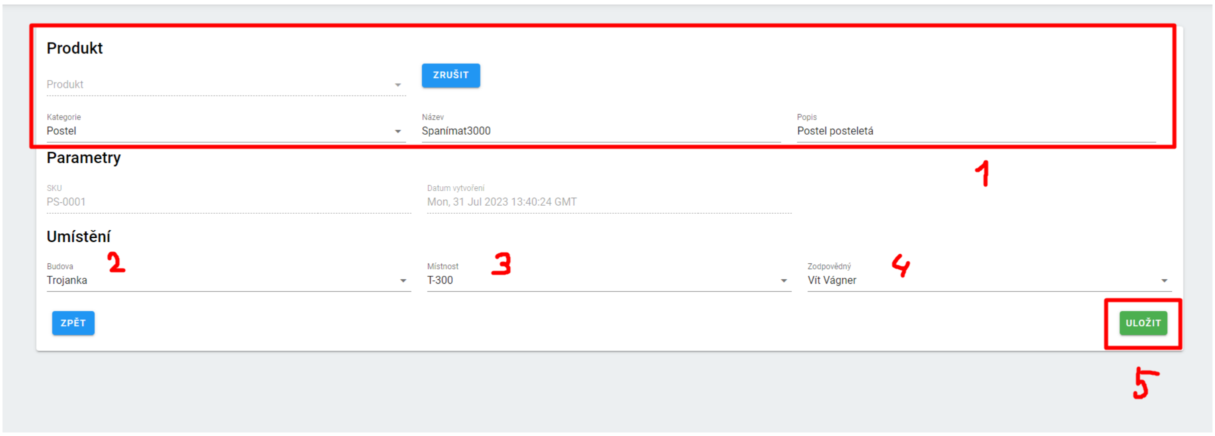
K vytvoření budovy je zapotřebí mít uživatelskou roli *"Admin"*.

V levém navigačním panelu v sekci ADMINISTRACE zvolíme tlačítko BUDOVY. Po otevření stránky naleznete v levém rohu tlačítko "Přidat budovu". Kliknutím se otevře okénko s políčky *Název*, *Zkratka*, *Adresa* a *Fotografie* budovy. První tři políčka vyplníte obvyklým způsobem. Políčko pro fotografii není povinné. Nahrání fotografie provedete kliknutím na Nahrání souboru, otevře se vám průzkumník souborů a zde vyberete fotku, kterou chcete nahrát. Po vyplnění políček stačí kliknout na "Uložit"a budova je vytvořena.

UC6 – Vytvoření nové položky

Vytvoření nové položky (záznamu) se provádí buďto na webové stránce nebo v mobilní aplikaci, kde je vytvoření rychlejší:

I. Webová stránka – Nejdříve se musíme dostat na stránku Evidence. Na stránce lze vidět seznam již vytvořených položek. Nad seznamem je filtrační systém a nad filtrem je tlačítko "Přidat Záznam". Po kliknutí na tlačítko budete přesměrováni na detail stránky Evidence. Prvním krokem k vytvoření položky je zapotřebí určit produkt. Produkt je možné vybrat ze seznamu. Když žádný produkt neexistuje nebo vámi požadovaný není v nabídce, kliknete přímo vedle Nabídky produktů na tlačítko "Nový". Tím se zpřístupní políčka pod nabídkou. Nastavíte danému produktu Kategorii (není-li požadovaná kategorie dostupná, vytvoření přerušíte a vykonáte nejprve krok "UC3 – Vytvoření kategorie", uvedete název produktu a stručný popis, co produkt představuje. Na základě zvoleného produktu systém vygeneruje Kód kategorie. Vedle políčka kódu je vidět i datum vytvoření.
Po vytvoření produktu je třeba položku zařadit do budovy, a poté do místnosti. To se provede vybráním budovy v nabídce (když není požadovaná budova v nabídce, vytvoření zrušíte a vykonáte nejdříve *"UC5 – Vytvoření budovy"*. Následně po vybrání budovy se vám zobrazí jednotlivé místnosti v budově. Zvolíte místnost, kam je položka určena (když není požadovaná místnost v nabídce, vytvoření přerušíte a vykonáte nejprve "UC4 - Vytvoření místnosti"). Nyní máme položku lokalizovanou a je třeba zvolit, jaký uživatel je za položku zodpovědný. Poté stačí jen kliknout na tlačítko "Uložit"a nová položka je vytvořena.


Produkt

Produkt		
Kategorie: Postel	Název: Spanímat3000	Popis: Postel posteletá

Parametry

SKU: PS-0001	Datum vytvoření: Mon, 31 Jul 2023 13:40:24 GMT

Umístění

Budova: Trojanka	Místnost: T-300	Zodpovědný: Vít Vágner

Obrázek 8.3: Ukázka v krocích průběhu UC6 - webová stránka

II. Mobilní aplikace – Vytvoření nové položky je v mobilní aplikaci kratší, jelikož v ní nemáte přístup k ADMINISTRACI. Proto, když budete vytvářet úplně novou položku, kde se bude muset vytvořit kategorie nebo bude například chybět budova či místnost, musíte použít webovou stránku. Vytvoření nové položky provedete tím, že dole na navigačním panelu kliknete na "Nová položka". Kliknutí vás přesměruje na stránku Vytvoření položky. Pokračujete vyplňováním políček odshora dolů. Zvolíte produkt, tím se vygeneruje kód, a poté zvolíte umístění vybráním budovy a místnosti, a pak zadáte, kdo je zodpovědný na položku.

Vytvoření provede tím, že se dole navigačním panelem kliknete na "Nová položka". Kliknutí vás přesměruje na stránku vytvoření položky. Pokračujete vyplňování políček ze shora dolů. Zvolíte produkt, tím se vygeneruje kód a poté zvolíte umístěni vybráním budovy a místnosti a pak zadáte, kdo je zodpovědný na položku.

UC7 – Editování položky

Editace se provádí na stránce Evidence. Zde se může využít filtrační systém k nalezení položky například podle SKU kódu. Po nalezení položky lze na položku v seznamu kliknout, to vás přesune na detail položky, kde můžete upravit některá políčka. Po upravení políček se klikne na tlačítko "Uložit"a položka byla zeditována.

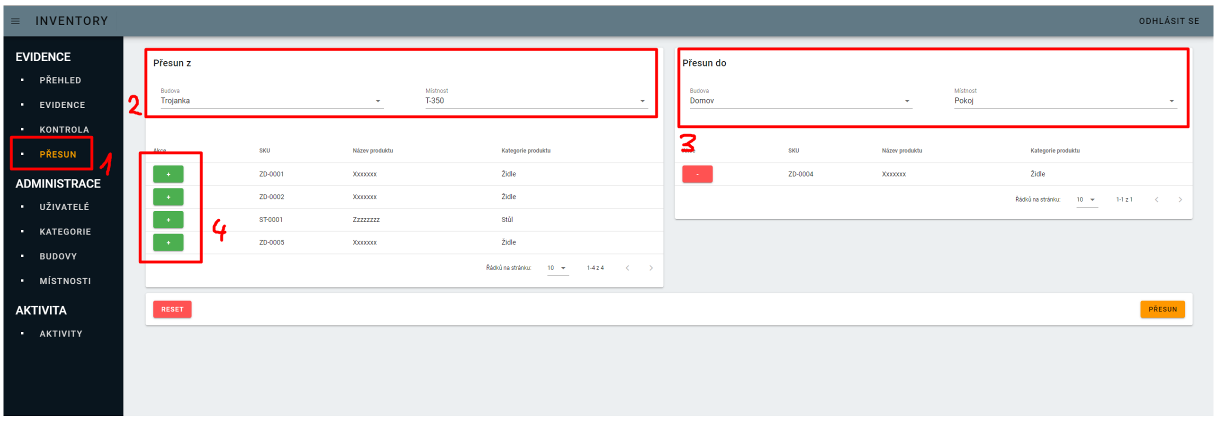
UC8 – Smazání položky

Přejděte na stránku Evidence a najděte položku pomocí filtru. Poté na položku v seznamu klikněte. Systém zobrazí detail položky. Zde je vidět vedle tlačítka "Uložit"červené tlačítko "Smazat". Kliknutím na červené tlačítko položku označíte jako smazanou. To znamená, že je dokladatelná, ale vystupuje jako smazaná. Stejným způsobem lze položku obnovit.

UC9 – Přesouvání položky

Přesun položky se může udělat dvěma způsoby:

První způsob je přes stránku Přesun. Na této stránce jsou dvě tabulky. Nejprve v levé tabulce vybereme budovu s místností, kde se nachází položky, které chceme přesunout. Poté v pravé tabulce vybereme budovu s místností, kam chceme položky přesunout. Když máme místnosti vybrané, můžeme za pomocí tlačítek "+"a "-"u položek přesouvat položky z místnosti do místnosti. Když máme přesunuty položky, které jsme chtěli přesunout, klikneme na tlačítko "Přesun".



Přesun z

Budova: Trojanka | Místnost: T-350

Akce	SKU	Název produktu	Kategorie produktu
+	ZD-0001	Xxxxxxx	Židle
+	ZD-0002	Xxxxxxx	Židle
+	ST-0001	Zzzzzzzz	Stůl
+	ZD-0005	Xxxxxxx	Židle

Řádků na stránku: 10 1-4 z 4

Přesun do

Budova: Domov | Místnost: Pokoj

Akce	SKU	Název produktu	Kategorie produktu
-	ZD-0004	Xxxxxxx	Židle

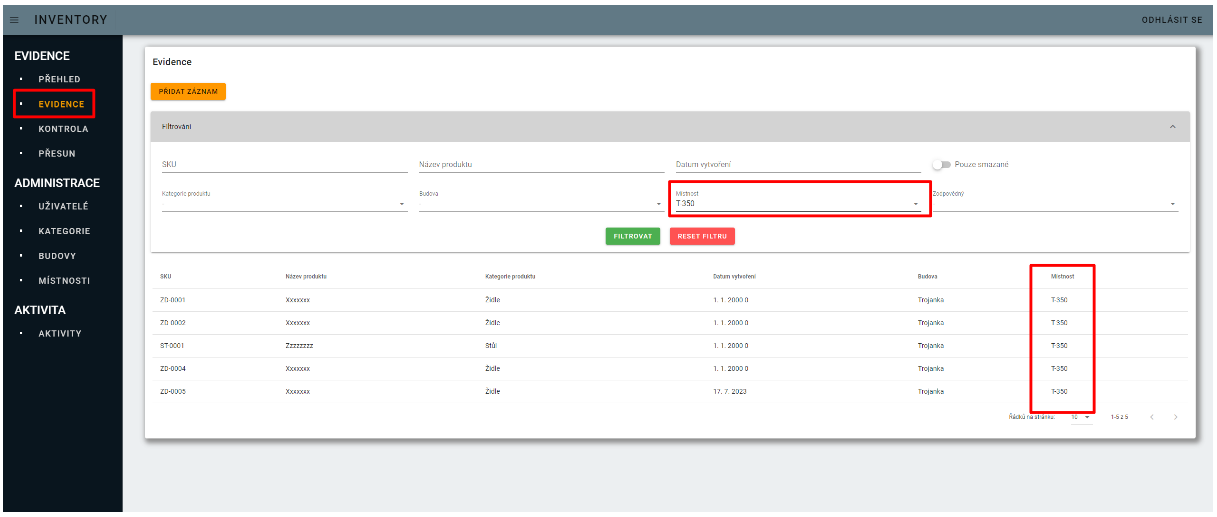
Řádků na stránku: 10 1-1 z 1

Obrázek 8.4: Ukázka v krocích průběhu UC9

Druhý způsob, který je spíše pro využití pro jednu nebo několik málo položek, se provede tak, že se jde na stránku Evidence a zvolí se *"UC7 – Editování položky"* a během editace se změní lokace položky.

UC10 – Seznam položek v místnosti

Seznam položek v místnosti lze získat pomocí stránky Evidence. Ve filtračním systému vyberete filtr pro danou místnost a seznam ukáže všechny položky, které jsou lokalizované ve vybrané místnosti.



Evidence

SKU | Název produktu | Datum vytvoření

Kategorie produktu: - | Budova: - | Místnost: T-350 | Zodpovědný: -

SKU	Název produktu	Kategorie produktu	Datum vytvoření	Budova	Místnost
ZD-0001	Xxxxxxx	Židle	1. 1. 2000 0	Trojanka	T-350
ZD-0002	Xxxxxxx	Židle	1. 1. 2000 0	Trojanka	T-350
ST-0001	Zzzzzzzz	Stůl	1. 1. 2000 0	Trojanka	T-350
ZD-0004	Xxxxxxx	Židle	1. 1. 2000 0	Trojanka	T-350
ZD-0005	Xxxxxxx	Židle	17. 7. 2023	Trojanka	T-350

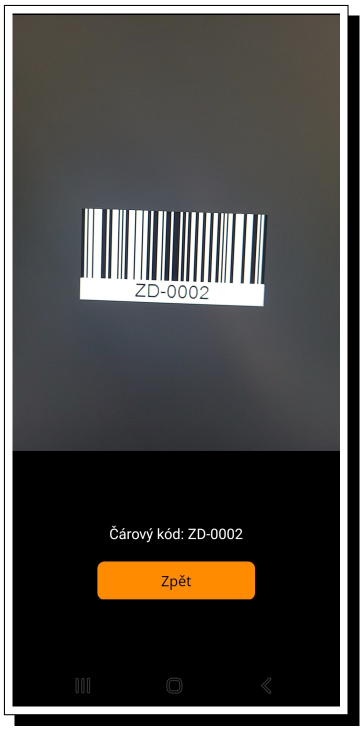
Řádků na stránku: 10 1-5 z 5

Obrázek 8.5: Ukázka v krocích průběhu UC10

UC11 – Kontrola položky

Kontrola položky probíhá přes mobilní aplikaci.

Kontrola spočívá v naskenování kódu na položce. Při kontrole se otevře kamera, která skenuje kód. Kameru srovnejte s kódem pro lepší detekování. Jestliže se kód shoduje s kódem skenované položky, položka se zkontroluje a označí se jako zkontrolovaná. Jestliže se kódy neshodují, objeví se oznámení, že kontrola selhala a musíte kontrolu opakovat.

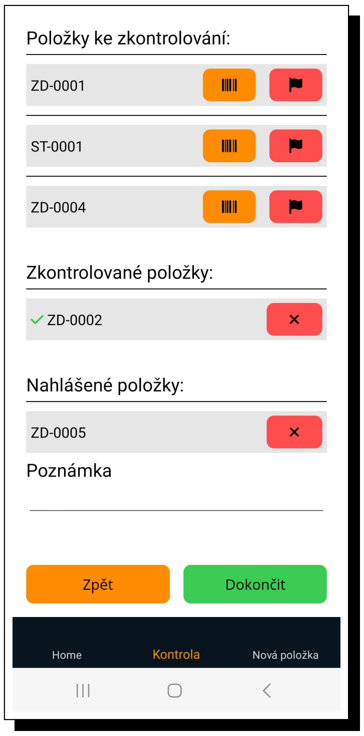


Obrázek 8.6: Ukázka skenování UC11

UC12 – Inventura místnosti

Pro inventuru místnosti musíte využít mobilní aplikaci. V navigačním panelu zvolíme záložku "Kontrola"a mobilní aplikace nás přesune na danou stránku. Na stránce je zobrazený seznam budov a kliknutím na řádek zvolíme, v jaké budově chceme inventuru vykonávat. Po vybrání budovy se zobrazí nabídka pro vybrání místnosti a stejným způsobem zvolíme místnost.

Poté, co zvolíte místnost, nám aplikace zobrazí tři seznamy. Seznam nezkontrolovaných položek, seznam zkontrolovaných položek a seznam nahlášených položek. Kontrolu položek provedeme zmáčknutím oranžového tlačítka s ikonkou čárového kódu. Tím se otevře sken čárových kódů a pokračujeme podle "UC11 – Kontrola položky". Tento krok opakujeme do té doby, než zbudou jen položky, které jsou v nepořádku. Ty poté nahlásíme pomocí červeného tlačítka s vlaječkou a pokračujeme podle "UC13 – Nahlášení položky". Tento krok opakujeme do doby, kdy v seznamu nezkontrolovaných položek už žádná položka není.

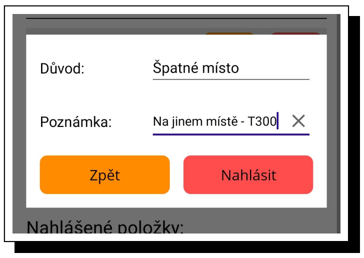


Obrázek 8.7: Ukázka stránky kontroly

Po zkontrolování položek napíšeme poznámku, pokud je třeba a můžeme vytvořit záznam inventury za pomocí tlačítka "Dokončit".

UC13 – Nahlášení položky

Nahlášení položky se provede během *"UC12 – Inventura místnosti"*, když se vyskytne neshoda. Nahlášení probíhá tak, že se klikne na červené tlačítko s vlaječkou a díky tomu se zobrazí okénko s dvěma políčky – důvod nahlášení a stručný popis. Důvod vybereme z nabídky a napíšeme poznámku. Potom už stačí kliknou na tlačítko "Nahlásit"a položka se označí jako nahlášená a je přesunuta do Listu nahlášených položek.

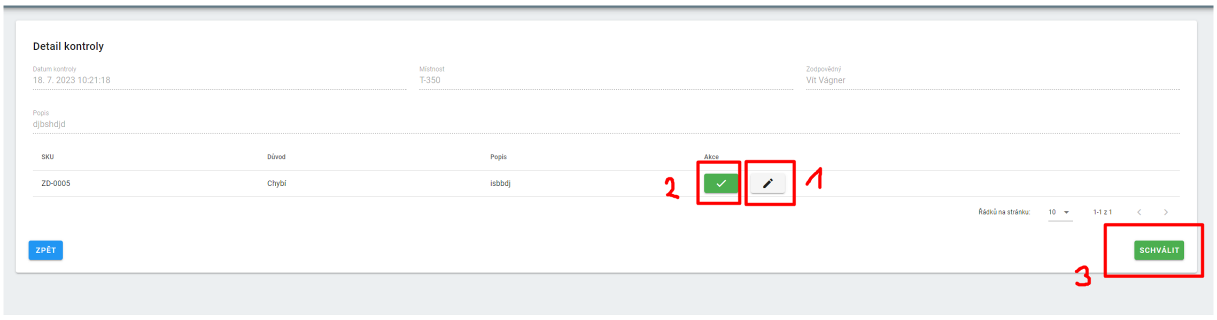

Důvod: Špatné místo

Poznámka: Na jinem místě - T300

Obrázek 8.8: Ukázka nahlášení UC13

UC14 – Vyřešení nahlášené položky

V detailu kontroly místnosti na stránce Detail kontroly je seznam nahlášených položek s důvodem nahlášení a popisem, jak vyřešit nesrovnalost. Kliknutím na tlačítko s ikonou pera přejdeme na detail nahlášené položky. Zde můžeme upravit neshodující se data, uložit položku a tlačítkem "Zpět"se vrátit na Detail kontroly. Po upravení položky můžeme položku označit jako vyřešenou a tím zmizí ze seznamu nahlášených položek.


Detail kontroly

Datum kontroly: 18. 7. 2023 10:21:18
Místnost: T-350
Zodpovědný: Vít Vágner
Popis: djbshdjd

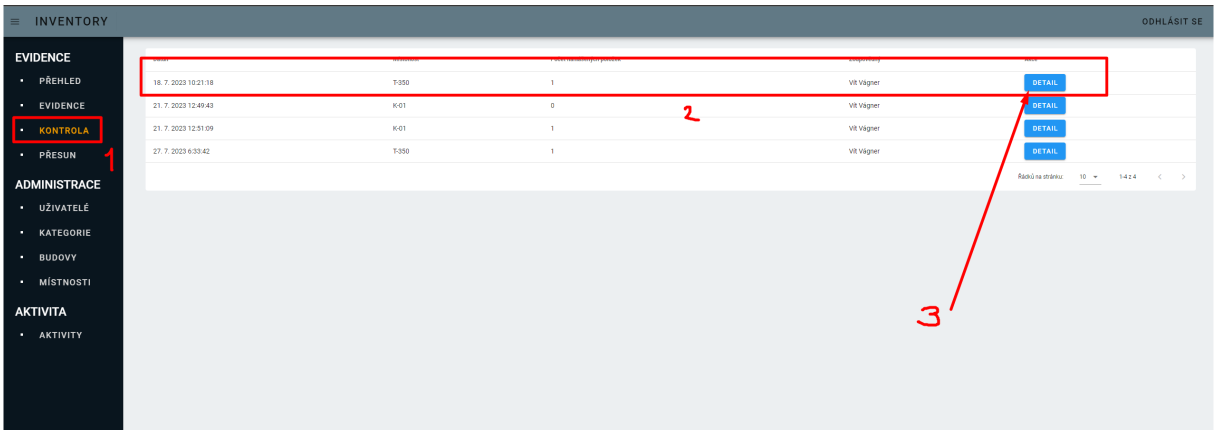
SKU	Důvod	Popis	Akce
ZD-0005	Chybí	isbbdj	✓ ✎

Řádků na stránku: 10 1-1 z 1

Obrázek 8.9: Ukázka v krocích průběhu UC14

UC15 – Dokončení inventury

Poté co jsme vytvořili záznam inventury skrz *"UC12 - Inventura místnosti"*, můžeme tento záznam najít na webové stránce. Jedná se o stránku Kontrola a když na ni přejdeme, uvidíme seznam nevyřešených inventur. Kliknutím na tlačítko detail u záznamu inventury přejdeme do detailu inventury, kde je vidět, kdo inventuru vykonal, kdy byla vykonána a důležitý seznam nahlášených položek, kde je nesrovnalost. Vyřešíme položky postupem popsaným v *"UC14 – Vyřešení nahlášené položky"* a kroky opakujeme, dokud nejsou všechny položky vyřešené. Po vyřešení položek je možné inventuru označit za dokončenou.


Datum	Místnost	Počet nahlášených položek	Zodpovědný	Akce
18. 7. 2023 10:21:18	T-350	1	Vít Vágner	DETAIL
21. 7. 2023 12:49:43	K-01	0	Vít Vágner	DETAIL
21. 7. 2023 12:51:09	K-01	1	Vít Vágner	DETAIL
27. 7. 2023 6:33:42	T-350	1	Vít Vágner	DETAIL

Řádků na stránku: 10 1-4 z 4

Obrázek 8.10: Ukázka v krocích průběhu UC15

Bibliography

[1] “Sorly,” [Online]. Available: https://www.sortly.com (visited on 08/02/2023).

[2] “Zoho inventory,” [Online]. Available: https://www.zoho.com (visited on 08/02/2023).

[3] “Salesbinder,” [Online]. Available: https://www.salesbinder.com (visited on 08/02/2023).

[4] “Boxstorm,” [Online]. Available: https://www.boxstorm.com (visited on 08/02/2023).

[5] “Multi-tier architecture,” [Online]. Available: https://en.wikipedia.org/wiki/Multitier_architecture (visited on 08/02/2023).

[6] “Mvc,” [Online]. Available: https://www.codecademy.com/article/mvc (visited on 08/02/2023).

[7] “Entity framework,” [Online]. Available: https://learn.microsoft.com/en-us/aspnet/entity-framework (visited on 08/02/2023).

[8] “.net,” [Online]. Available: https://dotnet.microsoft.com (visited on 08/02/2023).

[9] “C#,” [Online]. Available: https://learn.microsoft.com/en-us/dotnet/csharp (visited on 08/02/2023).

[10] “Visual studio,” [Online]. Available: https://visualstudio.microsoft.com (visited on 08/02/2023).

[11] “Gs1,” [Online]. Available: https://en.wikipedia.org/wiki/GS1 (visited on 08/02/2023).

[12] “.net maui,” [Online]. Available: https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui (visited on 08/02/2023).

[13] “Https,” [Online]. Available: https://en.wikipedia.org/wiki/HTTPS (visited on 08/02/2023).

[14] “Refit,” [Online]. Available: https://github.com/reactiveui/refit (visited on 08/02/2023).

[15] “Javascript,” [Online]. Available: https://www.javascript.com (visited on 08/02/2023).

[16] “Ecmascript,” [Online]. Available: https://en.wikipedia.org/wiki/ECMAScript (visited on 08/02/2023).

[17] “Typescript,” [Online]. Available: https://www.typescriptlang.org (visited on 08/02/2023).

[18] “Axios,” [Online]. Available: https://axios-http.com/docs/intro (visited on 08/02/2023).

[19] “Vue.js,” [Online]. Available: https://vuejs.org (visited on 08/02/2023).

[20] “Vuetify,” [Online]. Available: https://vuetifyjs.com (visited on 08/02/2023).

[21] “Material design,” [Online]. Available: https://m3.material.io (visited on 08/02/2023).

[22] “Iis server,” [Online]. Available: https://www.solarwinds.com/resources/it-glossary/iis-server (visited on 08/02/2023).

[23] “Zxing.net.maui,” [Online]. Available: https://github.com/Redth/ZXing.Net.Maui (visited on 08/02/2023).

[24] “Bearer authentication,” [Online]. Available: https://swagger.io/docs/specification/authentication/bearer-authentication (visited on 08/02/2023).