BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF BUSINESS AND MANAGEMENT

FAKULTA PODNIKATELSKÁ

INSTITUTE OF INFORMATICS

ÚSTAV INFORMATIKY

DESIGN, CREATION AND IMPLEMENTATION OF SOFTWARE APPLICATIONS IN THE CORPORATE ENVIRONMENT

NÁVRH, TVORBA A IMPLEMENTACE SOFTWAROVÉ APLIKACE VE FIREMNÍM PROSTŘEDÍ

MASTER'S THESIS

DIPLOMOVÁ PRÁCE

AUTHOR AUTOR PRÁCE	Bc. Jakub Filo
SUPERVISOR VEDOUCÍ PRÁCE	Ing. Lukáš Novák, Ph.D.

BRNO 2022

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF BUSINESS AND MANAGEMENT

Assignment Master's Thesis

Department:	Institute of Informatics
Student:	**Bc. Jakub Filo**
Supervisor:	**Ing. Lukáš Novák, Ph.D.**
Academic year:	2021/22
Study programme:	Information Management

Pursuant to Act no. 111/1998 Coll. concerning universities as amended and to the BUT Study Rules, the degree programme supervisor has assigned to you a Master's Thesis entitled:

Design, Creation and Implementation of Software Applications in the Corporate Environment

Characteristics of thesis dilemmas:

Introduction
Aims and Objectives
Theoretical foundation
Problem analysis and current situation
Implementation and contribution of suggested solutions
Conclusion
Bibliography
List of appendices

Objectives which should be achieve:

The goal of the thesis is to design and implement a web application in a business environment.

Basic sources of information:

GÁLA, L., J. POUR a Z. ŠEDIVÁ. Podniková informatika. 2. přeprac. a aktualiz. vyd. Praha: Grada Publishing, 2009. ISBN 978-80-247-2615-1.

HARDCASTLE, E. Business Information Systems. Ventus Publishing ApS, 2008. ISBN 978-87-7681-463-2.

PRETTYMAN, S. Learn PHP 7: object oriented modular programming using HTML5, CSS3, Javascript, XML, JSON, and MYSQL. Apress, 2015. ISBN 978-1-4842-1730-6.

SODOMKA, P. a H. KLČOVÁ. Informační systémy v podnikové praxi. 2. vyd. Brno: Computer

Press, 2000. ISBN 978-80-251-2878-7.

Deadline for submission Master's Thesis is given by the Schedule of the Academic year 2021/22

In Brno dated 28.2.2022

L. S.

doc. Ing. Miloš Koch, CSc.	doc. Ing. Vojtěch Bartoš, Ph.D.
Branch supervisor	Dean

ABSTRACT

The goal of this thesis is to analyze, design and implement an information system for inventory management of a dermatologic clinic. The thesis is structured into three main chapters. The first chapter describes a theoretical foundation required for the implementation of an information system in form of a web application. It also describes various analytical methods used for designing and implementing a change in a company. The second chapter analyzes the company and current situation of the inventory management. The last part of the thesis is concerned with a design, implementation and deployment of the application using industry best practices. The process of change is also analyzed, potential risks are evaluated and the project schedule is estimated. Achieved results are evaluated and several further improvements suggested.

KEYWORDS

information system, inventory management, web application, PESTLE, PORTER, 7S, SWOT, Lewin's change model, risk analysis, REST API, SQL, SPA

ABSTRAKT

Cílem této práce je analyzovat, navrhnout a implementovat informační systém pro řízení zásob dermatologické kliniky. Práce je strukturována do tří hlavních kapitol. První kapitola popisuje teoretická východiska potřebná pro implementaci informačního systému ve formě webové aplikace. Dále jsou popsány různé analytické metody určené pro návrh a implementaci změny ve společnosti. Druhá kapitola analyzuje společnost a současný stav řízení zásob. Poslední část práce se zabývá návrhem, implementací a nasazením aplikace s využitím osvědčených postupů. Je analyzován proces změny, zhodnocena potenciální rizika a odhadnut harmonogram projektu. Dosažené výsledky jsou vyhodnoceny a je navrženo několik dalších vylepšení.

KLÍČOVÁ SLOVA

informační systém, řízení zásob, webová aplikace, PESTLE, PORTER, 7S, SWOT, Lewinův model změny, analýza rizik, REST API, SQL, SPA

ROZŠÍŘENÝ ABSTRAKT

Cílem této diplomové práce je analyzovat, navrhnout a implementovat informační systém pro řízení zásob dermatologické kliniky. Informační systém by měl splňovat všechny požadavky definované manažmentem firmy. Měl by být implementován s ohledem na bezpečnost dat, použitelnost a možnost dalšího rozšíření. Je také potřebné implementovaný systém nasadit do provozu a vyhodnotit provedenou změnu.

Firma, ve které bude systém implementován je klinikou dermatologie a estetické medicíny se sídlem v Bratislavě. Specializuje se na diagnostiku a léčbu různých kožních onemocnění (psoriáza, kožní nádory apod.) a také na estetickou medicínu, jako například vlasová mezoterapie, laserová epilace a plastická chirurgie. Kromě lékařských vyšetření a zákroků také prodává koncovým zákazníkům širokou škálu dermatologických a kosmetických produktů.

První kapitola popisuje teoretická východiska potřebná pro implementaci informačního systému ve formě webové aplikace. Jsou zde popsány základní koncepty a definice týkající se informačních systémů. Pozornost je věnována technologiím informačních systémů běžících na webu – technologie klient-server, HTTP protokol, relační databáze, jazyk SQL a také aplikační rozhraní(API), JavaScript, JSON a další stavební bloky webového informačního systému. Jsou také popsány bezpečnostní hrozby, jako například SQL injection, nebo XSS útoky.

V další sekci první kapitoly jsou popsány analytické metody, které nám umožňují analyzovat interní a externí faktory, které mají vlyv na podnikání firmy – metody PESTLE, Porterova analýza pěti sil, analýza 7S a SWOT. Je zde také popsána analýza rizik, Lewinův model změn a časová analýza PERT.

Druhá kapitola analyzuje společnost a současný stav řízení zásob. Analytické metody popsané v první kapitole jsou prakticky aplikovány na vybranou firmu. V sourhné analýze SWOT byly identifikovány silné a slabé stránky společnosti, její zranitelosti, příležitosti a hrozby.

Jednou z největších slabostí společnosti je aktuální systém pro správu skladových zásob. Nyní je realizován pouze pomocí excelovských tabulek. Tento přístup byl vyhovující, dokuď byl objem tovaru se kterým se na klinice manipuluje velmi malý. Závěr analýz potvrdil, že s rostoucím počtem zákazníků kliniky a objemem tovaru a materiálů, které se na klinice spotřebují, vyvstala nutnost implementace nového informačního systému.

Ve třetí kapitole se práce věnuje samotné implementaci změny. Nejprve jsou definovány požadavky na nový informační systém. Ten by měl obsahovat uživatelské účty a role (administrátor, běžný uživatel). Musí umožňovat prohlížení, vytváření, editaci a mazání skladů, produktů, dodavatelů, kontaktů a přejímek tovaru. Měl by poskytovat funkcionalitu naskladňování a vyskladňování tovaru, s možností sken-

ování čárových kódů. Také by měl manažmentu kliniky poskytnout různé manažerské výstupy – například přehled nedostatkového tovaru, tovaru s blížící se exspirací, přehled všech manipulací s tovarem a audit systému.

Nejdůležitější procesy správy skladových zásob byly vyjádřeny pomocí EPC diagramů.

V další části třetí kapitoly byl na navrhovanou změnu aplikován Lewinův model řízené změny. V první fázi byla provedena analýza silového pole, dále byly popsány aktéři změny a jednotlivé kroky, které ke změně vedou. Analýza potvrdila, že hnací síly jsou silnější než brzdící síly a tak bylo možné ve změně pokračovat.

Dále byla provedena analýza rizik pomocí skórovací metody a navržena protiopatření. Možná rizika byla minimalizována.

V další části byla provedena časová analýza pomocí metody PERT a získána kritická cesta. Odhadovaná doba trvání projektu byla stanovena na 234 člověkodní. Byly také odhadnuty náklady na změnu v intervalu 22 500 až 36 000 EUR.

Stěžejní součástí třetí kapitoly je implementace informačního systému. Je zde popsána architektura aplikace, která využívá tří vrstvou architekturu. Podrobně je popsán backend i frontend aplikace, postavený na frameworku Ruby on Rails a Vue.js. Je popsána aplikační logika, hlavní metody a použité technologické postupy.

V další části je popsán proces nasazení aplikace do provozu – výběr hostingu, kontajnerizace aplikace a automatizace pomocí CI/CD pipeline.

Implementovaný systém byl podroben auditu pomocí nástroje ZEFIS. Byly identifikovány některé nedostatky implementovaného systému, jako například fakt, že uživatelé nejsou pravidelně nuceni měnit svá hesla. Na nedostatky systému byly navrženy protiopatření.

Byly také navrženy možná vylepšení v oblasti další automatizace nového informačního systému. Konkrétně se jedná o integraci s účetním systémem a automatizace procesu objednávek tovaru.

V závěru je možné zhodnotit, že implementovaný systém naplňuje všechny definované požadavky, zlepšuje proces řízení zásob ve firmě a poskytuje mnohem lepší uživatelský zážitek. Data mají nesrovnatelně lepší integritu a je zajištěna jejich bezpečnost pomocí uživatelských účtů a rolí. Systém také obsahuje popisované manažerské výstupy, které umožňují manažmentu firmy příjmat lepší rozhodnutí.

FILO, Jakub. *Design, Creation and Implementation of Software Applications in the Corporate Environment*. Brno: Brno University of Technology, Fakulta podnikatelská, Ústav informatiky, 2022, 116 p. Master's Thesis. Advised by Ing. Lukáš Novák, Ph.D.

Author's Declaration

Author:	Bc. Jakub Filo
Author's ID:	196249
Paper type:	Master's Thesis
Academic year:	2021/22
Topic:	Design, Creation and Implementation of Software Applications in the Corporate Environment

I declare that I have written this paper independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the paper and listed in the comprehensive bibliography at the end of the paper.

As the author, I furthermore declare that, with respect to the creation of this paper, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll. of the Czech Republic, Section 2, Head VI, Part 4.

Brno . . . .
author's signature*

*The author signs only in the printed version.

ACKNOWLEDGEMENT

I would like to thank my supervisor Ing. Lukáš Novák Ph.D. for all the help, guidance and an enormous patience during my work on this master's thesis.

Contents

Introduction

Information systems are essential part of every company's workflow. Nowadays, we can't imagine how difficult the work would be without their help. They greatly reduce the amount of repetitive tasks an employee has to do and they provide an invaluable source of insight for the management. The need for a capable information system is even more crucial, when a company grows. Spreadsheets can't compete with the level of advanced functionality and automation only an information system can provide.

In this thesis we will analyze, design and implement an information system for the inventory management of a dermatologic clinic.

In the first chapter, the theoretical foundation is presented. The key concepts of the information systems are explained first, following the theory of modern web-based information system. This include client-server architecture, web protocols, databases, programming languages and concepts needed to implement the system. Various security threats are also described in this chapter, including theory on how to mitigate or prevent them.

The second part of the first chapter consists of description of analytical methods, which help to analyze external and internal factors in the company's business and allow for a successful implementation of a change.

In the second chapter, the company is introduced and analyzed using methods described in the first chapter. The current problem of the inventory management is described in more detail and the analyses are summarized. This chapter serves as input for the final chapter.

In the final chapter, the system requirements for the new information system are presented. Next, the proposed change is analyzed using Lewin's change model analysis and the potential risks are identified and evaluated. The estimated time schedule is created using the PERT method.

In the second part of the final chapter, the implementation itself is discussed in detail, including description of information system backend and frontend and its overall architecture along with the most important components.

The deployment process of the implemented information system is described next, along with various automation using industry best-practices.

Finally, the implemented system is audited and several shortcomings identified and solutions proposed. Contribution of implemented solution is discussed and further improvements for the implemented system are proposed.

Aims and Objectives

Aims

The aim of this thesis is to analyze, design and implement an information system for inventory management of a dermatologic clinic. The information system should meet all of its requirements defined by the company's management and should be implemented with a consideration to security, usability and further extendability. It is also important to successfully deploy the implemented system, put it into operation and evaluate the change.

Objectives

For the aims to be achieved, we have to:

- Analyze the internal and external factors, which affect the company's business.
- Propose the change of the information system.
- Analyze the proposed change, evaluate if the change is meaningful.
- Create an estimated time schedule of the proposed change.
- Implement the information system using industry best-practices.
- Deploy the implemented information system.
- Audit the implemented information system and identify its shortcomings.
- Suggest solutions to the identified shortcomings.
- Propose further improvements to the implemented system.

1 Theoretical foundation

This chapter presents theoretical foundation, which is needed for the further understanding of this thesis.

The key concepts of information systems are explained in the first section. We define the DIKW hierarchy, information systems, particularly ERP systems.

In the next section, we explain theory needed for the understanding of a modern web applications. These topics include a description of the HTTP protocol and encryption, basic building blocks of a web applications – database, backend and frontend and current security threats every developer should know about – SQL injection and XSS attacks.

In the final section of the first chapter, we describe various analytical methods, which will be used in the Chapter 2, including PESTLE, PORTER, 7S and SWOT analysis. Lewin's change model method is described next, following the theory of the PERT analysis.

1.1 Key concepts

Before getting any further, it is necessary to explain some of the key concepts of information systems.

1.1.1 Data, Information, Knowledge, Wisdom hierarchy

These terms are often misinterpreted, mainly due to its highly abstract nature. There is no common accepted conceptual reference and terminology[27], so we present the reader with the most fitting definitions in relation to information technology.

Data

Data are discrete, objective facts or observations, which are unorganized and unprocessed, and do not convey any specific meaning[27]. For example, we can think of a binary representation of the company's inventory. We can see only the raw data - ones and zeros. It doesn't provide us with any information.

Information

Information is a message about an unstable phenomenon, that lowers the degree of uncertainty about this phenomenon among its receivers[58](author's translation). In other words, information is data which adds value to the understanding of a subject[27]. To continue with our example, we can imagine inventory in a human readable form (e.g. in table). We can answer many questions with this information - what products are in stocks, how many pieces, etc. It makes the data useful.

Knowledge

Knowledge is a combination of data and information, with added expert opinion, skills, and experience, which result in a valuable asset which can be used to aid decision making[27]. We can apply this definition to our example. A person (or an information system) with knowledge of how many pieces of a given product should be in stock, can decide to order and stock up the product when its quantity is low.

Wisdom

Wisdom is accumulated knowledge, which allows you to understand how to apply concepts from one domain to new situations or problems[27]. In regards to our example, we can optimize a process of ordering goods to be the most effective - increase its effectiveness. This requires a wisdom.

1.1.2 Information systems

Information system represent a consistent, ordered set of components, which interact in order to create, collect, process, transfer and distribute information. Information system consists of people, who use the information, and of information resources. Information system's component consists of one or more elements[58](author's translation).

We can see a general scheme of an information system in the Figure 1.1. It contains[58]:

- **Input:** Components, which gather an information, which is to be processed.
- **Processing:** Components, which transform the inputs to the outputs.
- **Output:** Components, which transfer an information to the user of the information system.

An information system can also contain `control` components and `feedback`. A control components setup the standards of processing, measuring if the standards were achieved and controlling actions which minimize deviations from the standard[58].

The feedback can change the next input by analyzing outputs. It can also change the processing of the system[58].

Business information systems

There are many types of information systems. This thesis is concerned with a business information systems, which are used in organizations. Their purpose is defined by business requirements. They can be used to support forecasting, planning, control, coordination, decision making and operational activities in an organization[17].

In most organizations business information systems make extensive use of information technology (computers), primarily because of their advantages in speed, accuracy and dependability. They also have a high degree of flexibility due to their ability to be programmed to carry out a wide variety of tasks[17].

ERP information systems

Enterprise Resource Planning information systems allow to operate and coordinate business resources and activities. Their main attributes are[58]:

- **Integration of business operation tasks:** Different sections of an organization (marketing, sales, logistic) use the same or similar data, which can be stored in different information systems, that are not compatible with each other. This can cause many discrepancies. By integrating them, ERP systems help to keep data integrity and effectivity.
- **Transactional character:** ERP system have a transactional character. They keep databases up-to-date (adding new suppliers, products, updating their information), create and record various business documents (invoices, orders) and realize business transactions.

ERP information systems usually consist of different modules and tools. They are often specialized for specific industry solutions such as Industrial Equipment Manufacturing or Asset Management.

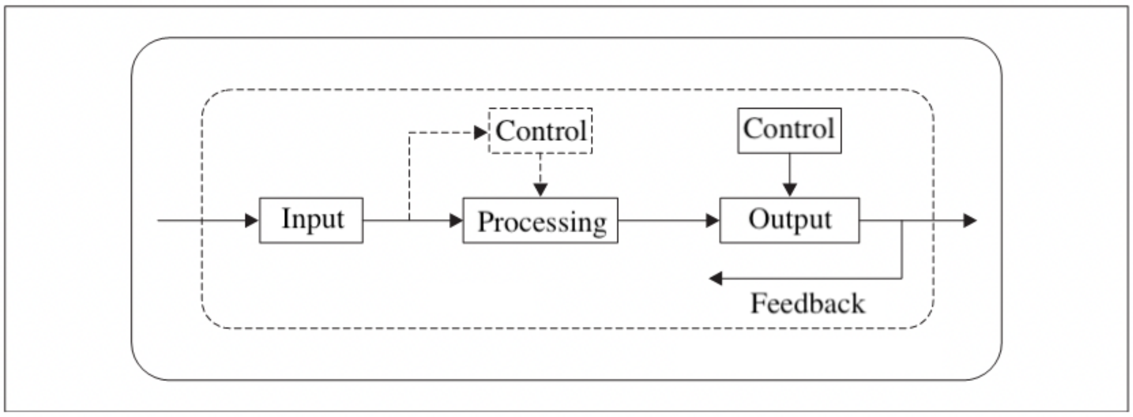


Fig. 1.1: Information system components[58](edited)

1.1.3 Process

Process is defined as a group of related actions, interacting with each other, which transforms the inputs to the outputs. Actions use resources (people, material, etc.). Process can have many inputs and outputs[58](author's translation).

Process is triggered by some action. It can have a various origin, for example time based trigger (every monday) or some input (new invoice). The inputs represent all the material or energy-based inputs of the process which enter the process at the beginning – e.g. an order. The outputs are similar to the inputs – they can be material (a new product) or energy-based (order sent to the customer). The process is finished in the final state – e.g. order fulfilled, or product manufactured[58].

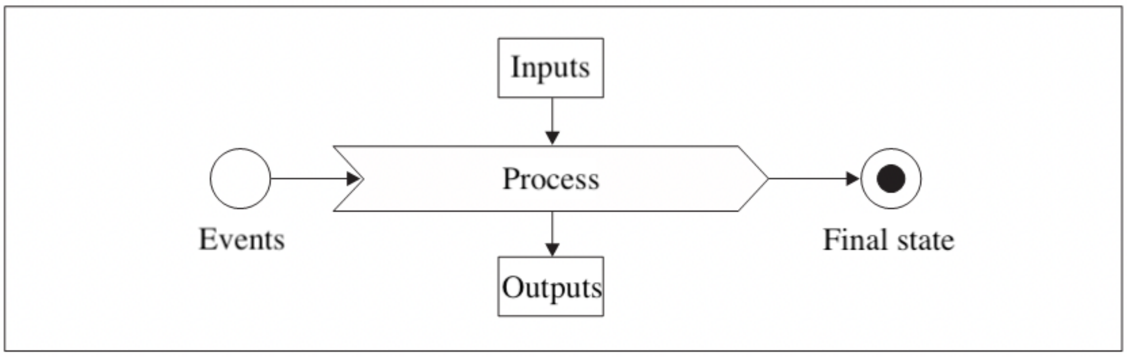


Fig. 1.2: Process model[58](edited)

1.2 Web-based information systems

Modern information systems gravitate towards a web-based architecture. Nowadays, browsers allow for a rich functionality while keeping hardware requirements of the user's computer low. In this section, we will explain web-based information systems architecture and key concepts, which are needed for implementing and running them reliably, securely and efficiently.

1.2.1 Client-Server architecture

In the information technology, client-server is a system architecture model consisting of two parts – client systems (primarily user's computer), and server systems (primarily high performance computers located in data centers), both communicating over a computer network[28].

The client process continuously launches a connection to the server, while the server processes the requests. One server can serve many clients.[28].

The client-server architecture typically consists of three layers:

- **Presentation layer:** It is a layer, which user can directly interact with, typically a web page. It communicates with other layers and displays information from their responses (e.g. listing of products).
- **Application layer:** This layer handles the application logic, its functionality. It performs data processing such as data filtering or various calculations.
- **Data layer:** Data layer, typically a database, stores the data and provides an access to this data, commonly via an API (see Section 1.2.4).

The client-server architecture can be categorized into four types[28]:

- **One-tier architecture:** Handles all of the application layers (presentation, application layer, data layer) in one package – e.g. MS Office.
- **Two-tier architecture:** Client tier handles presentation and application layer layer, while the server tier handles the data layer. Client sends requests to the server, server processes the request and sends a response to the client.
- **Three-tier architecture:** Client tier handles the presentation layer, while the server handles application layer and data layer.
- **N-tier architecture:** Also called a distributed architecture, separates tiers between client and server as the three-tier architecture, but the number of servers processing the application and data layer is increased, which creates a distributed system.

Typically, web-based information systems use three-tier or n-tier architectures. The three-tier architecture can be seen in Figure 1.3.

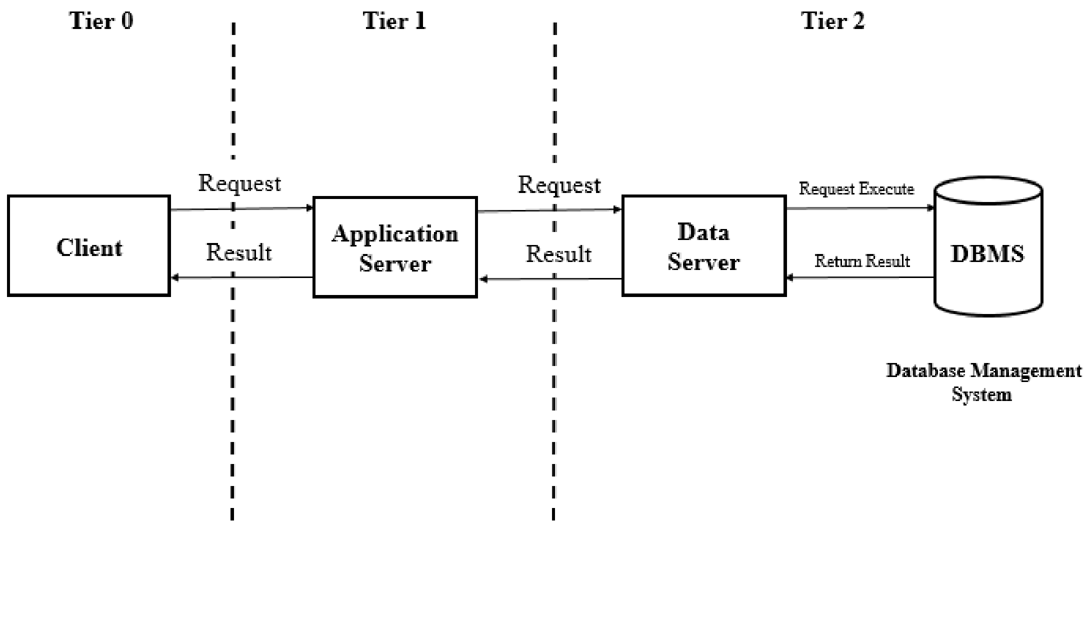


Fig. 1.3: Client-server three-tier architecture[6]

Hypertext Transfer Protocol

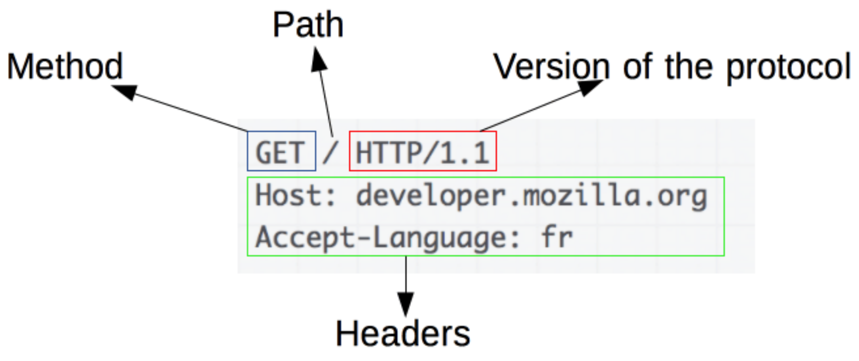
HTTP is de-facto standard and foundation of any data exchange on the web. It is a client-server protocol (see previous section). Clients and servers communicate by exchanging individual messages. It commonly uses the underlying Transmission Control Protocol (TCP) on the transport layer[29]. This makes the communication reliable and ordered. Next, we will describe the HTTP client-server communication flow in more detail.

When a client wants to communicate with a server, it performs the following steps (see the Figure 1.6)[29][44]:

1. Client initiates the handshake message (also called SYN or Synchronization Sequence Number) message to the server.
2. Server acknowledges the client request by sending back the SYN-ACK message (Synchronization and Acknowledgement).
3. Client sends the acknowledgment (ACK) to the server. After this step the connection is established between the client and the server and data can be transmitted.
4. Client now can send HTTP messages (for example a GET message).
5. Server responds with a status code indicating if the request was successful or not and why. It can also contain data encoded in various formats.
6. After the exchange, client requests the server to terminate the established connection by sending FIN message.
7. Server responds with sending back the FIN-ACK message.
8. After client receives the FIN-ACK message, it confirms it by sending ACK message to the server and the exchange is completed.

The client can send various types of messages. The `request` message consists of (see Figure 1.4)[29]:

- **Method:** Defines an operation client wants to perform. Typically it wants to fetch a resource (using `GET`) or post a data to a server (using `POST`)
- **Path:** A path or URL of the resource to fetch.
- **Version of the protocol:** Version of the HTTP protocol.
- **Optional headers:** Optional headers that convey additional information for the server – e.g. authentication headers, language, etc.
- **Body of the message:** For some methods a request can contain data which is to be send to a server.

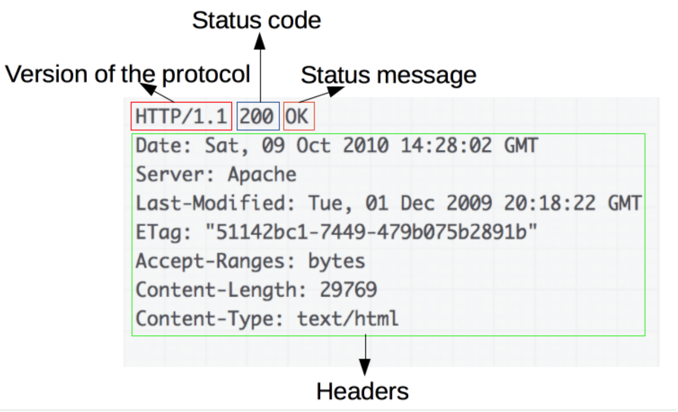


```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

Fig. 1.4: HTTP request message[29]

Server sends a response to the client's request. `Response` consists of (see Figure 1.5)[29]:

- **Version of the protocol:** Version of the HTTP protocol.
- **Status code:** Indicating if a request was successful or not and why.
- **HTTP headers:** Similarly as with the request.
- **Optional body:** Contains a fetched resources.



```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html
```

Fig. 1.5: HTTP response message[29]

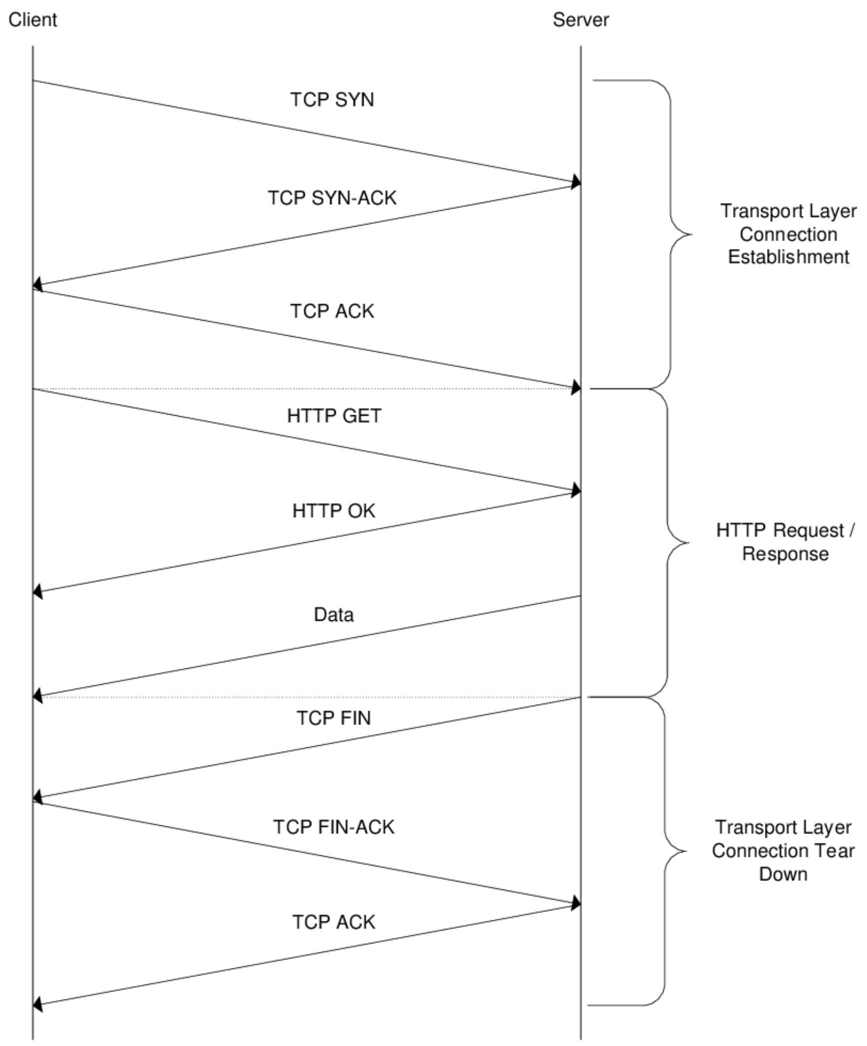


Fig. 1.6: HTTP communication flow[52]

1.2.2 Encrypted communication

It is crucial to keep an information system safe and secure. HTTP communication is by design not encrypted and thus presents a great security risk. Data transferred via HTTP protocol can be spied on in plain text, leaking passwords and other confidential information.

That is why the HTTPS protocol was developed, which encrypts the communication and makes it more secure. The encryption is provided via SSL certificates. SSL certificate is a data file hosted in a website's origin server. It contains a website's public key and its identity, along with related information. Devices attempting to communicate with an origin server will reference this file to obtain the public key and verify the server's identity. A private key is kept secret and secure[8].

Certificate authority (CA) is a trusted third party organization, that generates and gives out SSL certificates. It also signs the certificate with its own private key, which allows devices to verify it[8].

1.2.3 Databases

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. Data in a database are usually modeled in tables, which consist of rows and columns. Typically, database requires a software program called a database management system (DBMS), which allows users of the database to retrieve, create, update and delete information stored in it. The most common DBMSs include PostgreSQL, Microsoft SQL Server, or Oracle Database[35][40].

Relational database

Relation database is based on the relational model. A relation is represented by a table. Each table is composed of rows (data) and columns (attributes). Each row has a unique key, also called a primary key[37].

Relational databases are compatible with the `ACID` principle, which represents these features[37]:

- **Atomicity:** In simple terms, an atomicity is a feature, which ensures, that every database operation (also called transaction) is either completely successful, or completely failed, meaning that the database state is left unchanged.
- **Consistency:** Data stored in a relational database keep their integrity. They must be valid for all of the rules and constrains, that can be defined over them. This feature is assured for every transaction.
- **Isolation:** Ensures independence of multiple transactions that are executed at the same time, meaning the end state of the database after execution of these concurrent transactions is the same as if they were executed sequentially.
- **Durability:** Ensures, that the data stored in the database remain unchanged even in a case of a failure (e.g. power outage or crash).

SQL language

SQL stands for Structured Query Language and it is de-facto the standard database language used for accessing, storing and manipulating data in relational databases[39].

Commands of the SQL language can be categorized into 5 types:

- **DQL (Data Query Language):** Represented by the `SELECT` command, is used to retrieve saved records from the database.

- **DDL (Data Definition Language):** Allows for table creation, modification and deletion using commands such as: `CREATE`, `ALTER` and `DROP`.
- **DML (Data Manipulation Language):** Handles data manipulation within a table. Provides commands such as: `INSERT`, `UPDATE` and `DELETE`.
- **DCL (Data Control Language):** It is used for changing the privileges of database's users using commands like `GRANT` or `REVOKE`.
- **TCL (Transaction Control Language):** Controls database transactions, allows to make permanent changes (`COMMIT` command) or reverse them (`ROLLBACK` command).

As an example, we can select all suppliers from the table called `suppliers` using the `SELECT` command:

```sql
SELECT * FROM suppliers;
```

We can also use various conditions for the `SELECT` command. This example selects suppliers from Slovakia:

```sql
SELECT * FROM suppliers WHERE country='Slovakia';
```

This is but a short example of the SQL language. It is very powerful in the hands of a skilled programmer. It provides a common interface for relational databases and can be used for a wide variety of tasks – from data analysis to batch data modifications.

Boyce-Codd Normal Form

It is important to build a normalized database structure, because of data integrity, performance and scalability. Non-normalized database scheme can lead to various anomalies and side effect when manipulating the data. Boyce-Codd Normal Form is the most advanced version of a normalized database scheme.

A table is in BCNF if every functional dependency $X \to Y$, X is the super key of the table and the table is also in the third normal form (3NF)[41].

We can explain it better on an example using a table with three columns – `Student`, `Teacher` and `Subject` (see Table 1.1)[41]:

Student	**Teacher**	**Subject**
Student One	Teacher One	A
Student One	Teacher Two	B
Student Two	Teacher Two	A
Student Two	Teacher Three	B

Tab. 1.1: Non-normalized table example[41](modified)

Table 1.1 is not in the BCNF form, because in the functional dependency ($Teacher \rightarrow Subject$), `Teacher` is not a key[41]. This can lead to anomalies – e.g deleting the `Student Two` leads to the loss of an information, that `Teacher Three` teaches `Subject B`.

To transform the non-normalized table to the BCNF form, we have to decompose the original structure to two tables (Table 1.2 and Table 1.3)[41]. Now there are two separate relationship, which satisfy the BCNF definition and the anomalies are removed.

Teacher	**Subject**
Teacher One	A
Teacher Two	B
Teacher Three	B

Tab. 1.2: Normalized Teacher-Subject table example[41](modified)

Student	**Teacher**
Student One	Teacher One
Student One	Teacher Two
Student Two	Teacher Two
Student Two	Teacher Three

Tab. 1.3: Normalized Student-Teacher table example[41](modified)

1.2.4 Application backend

We refer to application and data layer as an application backend (see Section 1.2.1). It denotes all of the application logic and data processing, simply things on the background, which users don't see.

There are many theoretical principles regarding the application backend, which we will explain in this section.

API

Application Programming Interface (API) allows for two applications to communicate with each other. In web-based applications it typically represents a communication between the application backend and frontend[2].

REST API (REST stands for Representational state transfer) is an API type, which provides client-server communications for web applications over HTTP protocol. It works on a principle of endpoints (URL addresses), which provide the CRUD operations (Create, Read, Update and Delete) to the application. They correspond to the most popular SQL commands (`INSERT`, `SELECT`, `UPDATE` and `DELETE`)[2].

The main design patterns of the REST API are[43]:

- **Addressability:** Every resource has a unique identifier, which is typically a noun (e.g `/suppliers`).
- **Representation-Orientation:** Requests for resources return representations, which contain the state of the resource. The client can update or delete resources through their representations.
- **Uniform interface:** Actions follow small number of verbs, typically derived from HTTP methods, such as `GET`, `POST`, `PATCH` or `DELETE`.
- **Statelessness:** The server does not store any state of the client. Every request from the client must contain all the necessary information (such as credentials).

As an example, we can design a simple REST API for the `Suppliers` resource (see Table 1.4).

`GET /suppliers`	Get all or some suppliers (it should be possible to filter the resource).
`POST /suppliers`	Create a new supplier. Server sets the ID of the resource and returns its representation.
`PATCH /suppliers/3`	Update an existing supplier. Client puts a whole representation of the resource into payload.
`DELETE /suppliers/3`	Delete a supplier with an ID.

Tab. 1.4: REST API endpoints example

Model–view–controller

Model-view-controller (MVC) is a design pattern used for developing applications. Basic components of MVC are (see Figure 1.7)[24]:

- **Model:** The model manages data – encapsulates the application state, handles its processing. It informs other components about data modification.
- **View:** The view manages user interface (UI) – it displays data to the user by retrieving it from the model.
- **Controller:** The controller serves as the interface between the model and the view – it translates user events to request for the model.

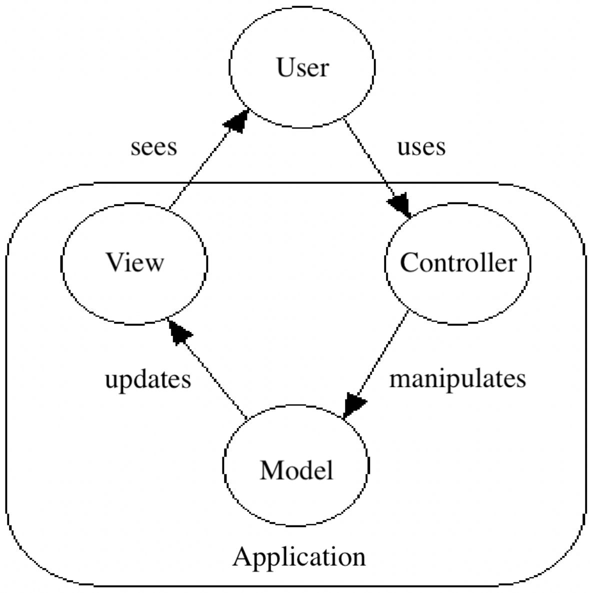


Fig. 1.7: MVC pattern[24](edited)

This approach has several benefits[24]:

- **Complexity:** Separation of responsibilities into independent components reduces complexity of the application architecture.
- **Scalability:** An application using the MVC pattern is highly scalable – new components can be added or removed without affecting the existing functionality.
- **Maintainability:** An application is easier to maintain. Programmers can also implement it in parallel, which speeds up the development.

Backend frameworks

Implementation of an information system is quite complex task. It is therefore in both management's and programmer's best interest to make this process as easy as possible.

Frameworks solve this problem by providing a ready-made components or solutions that are customized in order to speed up development[42]. It can provide features such as[42]:

- **ORM:** Or Object-relational mapping provides an abstraction for access and manipulation of database data without having to consider inner workings of the underlying database management system.
- **Templating system:** Templating systems provide an easy way to generate dynamic web pages. The programmer does not have to write a raw HTML

and JavaScript, but can leverage the framework's templating system which simplifies this process.

- **URL generation:** Framework can provide an easy way to generate paths and URLs, avoiding the need to hardcode strings in the application.
- **Security:** Some frameworks provide a built-in methods for authentication and authorization, which increases security, because these methods are usually properly tested and audited, compared to programmer's own implementation.

1.2.5 Application frontend

Application frontend handles the presentation layer (see Section 1.2.1), that is things that user directly see and interact with. We will explain the main building blocks of an application frontend in this section.

HTML

HTML (HyperText Markup Language) is the code that is used to structure a web page and its content[30]. It includes paragraphs, links, images, tables and other elements, which make the content of the web page appear or behave a certain way. HTML elements consists of (see Figure 1.8)[30]:

- **Opening tag:** Consists of name of the element wrapped in brackets (<>). It states where the element begins.
- **Closing tag:** Same as the opening tag, but denotes where the element ends.
- **Content:** The content of the element. HTML elements can be nested, that is we can put elements inside other elements.

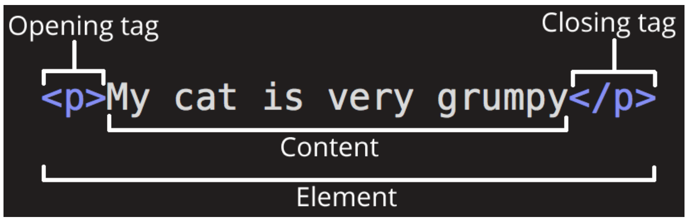


Fig. 1.8: Structure of a HTML element[30]

The most important HTML elements are links, which allow us to navigate a web page. The HyperText (HT) in HTML puts an emphasis on this fact – it is text which contains links to other texts[30].

A basic HTML page can look like in the code below[30]:

```html
<html>
  <head>
    <title>Inventory management system</title>
  </head>
  <body>
    <a href="/suppliers">Suppliers</a>
  </body>
</html>
```

It is made of these nested elements[30]:

- `<html></html>`: The root element, which wraps the whole page.
- `<head></head>`: Includes elements, which are not directly visible to the user. It can be title, CSS styles, page encoding definitions, etc.
- `<body></body>`: All of the visible content of the page is put inside this element.
- `<title></title>`: Sets a title of the page, which can be seen in the browser's navigation bar.
- `<a></a>`: As mentioned before, it allows to navigate to other HTML documents, e.g. `suppliers` in our example.

JavaScript

JavaScript is a programming language that allows to implement complex features on web pages. It can transform static HTML pages into dynamic and interactive applications. It runs on the client side – in the user's browser.

JavaScript is an interpreted language – its commands are executed directly, without a need for compilation. This allows for a rapid development – programmer doesn't have to wait for a compilation and can directly see the result[33].

It is a complex language with various use cases, but one of the most important is a DOM manipulation. DOM is a programming API for HTML documents. It stores a document's structure in a tree-like model. Using JavaScript we can manipulate a DOM – add elements to a web page, change content of an existing elements, etc.[33].

As an example, we can change a link's content by a following JavaScript code:

```html
<a href="/suppliers/1">Supplier 1</a>

<script>
let supplierLink = document.querySelector ('a')

supplierLink.href = '/suppliers/2'
</script>
```

We first select the `<a>` element and in the second step we change its `href` attribute to a new URL address.

JavaScript offers even more features, but listing them all would be past the merit of this thesis.

JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate[23]. It is used in web applications as a common language in which a server can communicate with a client.
The main building blocks of JSON format are[23]:

- **Collection of name/value pairs:** Denoted as `{ key: value }` object.
- **Ordered list of values:** Denoted as an array – e.g. `[1,2,3]`

For example, we can send a HTTP search request to a server to find a certain type of suppliers. The server sends a HTTP response containing body with the founded suppliers in the JSON format:

```
[
  { "name": "Supplier 1" },
  { "name": "Supplier 2" }
]
```

The result can be easily parsed and displayed to the user via JavaScript, which can transform the JSON into a HTML structure – e.g. list:

```
<ul>
  <li>Supplier 1</li>
  <li>Supplier 2</li>
</ul>
```

CSS

CSS (Cascading Style Sheets) language allows to style HTML elements. We can for example change the color, size or font of a text, give it a background, or even create a layout with a sidebar menu and main content area.

CSS is a rule-based language – we define the rules by specifying groups of styles that should be applied to particular elements or groups of elements on a web page[32].

We can store CSS rules in a separate file, or include them directly into a web page.

As an example, we can change a text color with the following rule:

```
<span>RED TEXT</span>

<style>
span {
  color: red;
}
</style>
```

Single-page application

An SPA (Single-page application) is a web app implementation that loads only a single web document, and then updates the body content of that single document via JavaScript[31].

SPAs have many benefits, mainly:

- **Speed:** The speed of an SPA application is better compared to a standard web, because the browser does not fetch a whole new page upon each interaction or navigation.
- **Better UX:** An SPA application feels more like a standard, native desktop or mobile app. The user experience is therefore better and more fluid.

1.2.6 Security threats

Running web-based information systems possess multiple security threats. It is crucial to minimize their probability, especially in a corporate environment, where we usually store and process various confidential information. In this section we explain two major security threats to web-based information systems and also countermeasures to them.

SQL injection

The injection process works by prematurely terminating a text string and appending a new command. Because the inserted command may have additional strings appended to it before it is executed, the malefactor terminates the injected string with a comment mark `--`. Subsequent text is ignored at execution time[53].

Let's say we have a code, which builds an SQL query by concatenating strings entered by the user (e.g. supplier search form).

```
userCountry = params['city']
sql = "SELECT * from suppliers WHERE country = '" + userCountry + "
    '";
```

When the user enters a normal string, such as `Slovakia`, the resulting query is safe:

```
SELECT * from suppliers WHERE country = 'Slovakia';
```

The attacker can however easily enter a malicious query such as:

```
Slovakia'; DROP TABLE suppliers--
```

This will result in SQL query:

```
SELECT * from suppliers WHERE country = 'Slovakia'; DROP TABLE
    suppliers--';
```

The semicolon (`;`) denotes the end of one query and the start of another. The double hyphen (`--`) indicates that the rest of the current line is a comment and should be ignored[53]. Therefore, the malicious command is processed and SQL server deletes the suppliers table.

We can prevent the SQL injection using parametrized queries, where the query statement is pre-compiled allowing only predefined parameters. This allows to recognize the user input, which is escaped and thus the threat is prevented – execution is not allowed, the user input is always just a string.

XSS

Cross-Site Scripting or XSS is a type of attack, in which malicious scripts are injected into otherwise trusted websites[26]. It primarily occurs when data enters a web application through an untrusted source.

We can explain this attack on an example:

- Web application contains a discussion, where users can post comments.
- These comments are outputted directly to the rendered page.
- An attacker enters a malicious comment, e.g. a script, which steals user's cookies.
- The comment is sent via an HTTP request and saved in a database.
- When the next user visits the comment section, the script is executed and cookies are sent to the attacker's address.

To mitigate these attacks, it is crucial to validate user input and sanitize it – escape special words such as `<script>` tags. This way a script from an attacker cannot be executed on the web application's page.

1.3 Analytical methods

In this section, we will explain analytical methods that help us to evaluate external and internal factors, which affect the company's business. In the second part of the section, we describe Lewin's change model, which allows us to successfully implement a change in the company. Next, we explain basics of the risk analysis. Finally, the PERT time analysis is explained, which helps us to estimate a time schedule of the change.

1.3.1 PESTLE analysis

Companies operate as part of a larger ecosystem. They are vulnerable to a variety of exogenous factors, which can have a major impact on the firm's competitive positioning[45]. PESTLE analysis examines the following factors[45]:

- **Political factors:** Includes factors like taxation, labor, political stability, commercial restrictions, etc. From the recent events we can mention Great Britain leaving the European Union. Political factors have serious impact on business strategy and orientation.
- **Economic factors:** Have the most obvious imact on the profitability of a market. The most popular indicator of economic performance is the Gross Domestic Product (GDP) per capita, which is useful for those industries, which are highly income elastic. We can also consider Purchasing Power Parity (PPP), inflation, price of raw materials and other inputs that the company needs.
- **Social factors:** Social trends affect consumer tastes and demand for a product or service. It includes demographics, lifestyle, perception of brands and products, influencers, etc. Social factors enable the company to predict what pressures are likely to be made by various stakeholders[45].
- **Technological factors:** Technology evolves rapidly and yesterday's tech becomes obsolete almost overnight. It can be used to exploit a competitive advantage – cheaper production, product quality, more advanced business intelligence, etc. It is important to be able to quickly adapt to the new technologies and be aware of how it influences the industry.
- **Legal factors:** Legal factors such as employment laws and various regulations affect both the internal and the external environment of a company. They can force competitors out of business, or on the other hand, drive new business to enter an industry.
- **Environmental factors:** For certain industries, an environmental factors play an important role in their strategy. These include e.g. farming and agriculture, where the climate change directly affect its business.

1.3.2 PORTER analysis

PORTER analysis, also called Porter's five forces model, is used for the analysis of the company's microenvironment. Porter identified five forces, which affect the company and represent possible threats or opportunities. These forces can act together, or there can be only one active force[47]. The five forces are[47]:

- **Competition in the industry:** Other companies, which can affect the business. The number of competitors is inversely proportional to the company's strength – more competitors means lesser power of the company.
- **Potential of new entrants into the industry:** The risk level of a new competition is determined by various barriers, which complicate an entry to the industry.

- **Power of suppliers:** Areas, where suppliers have power. It can be high diversification of the suppliers, threat of vertical integration, few or non-existent substitutes, etc.
- **Power of customers:** We evaluate the power of customers – ask questions like if the customer can choose between a large number of competitors, whether there is a threat of vertical integration (customer can manufacture their own product), or if the customer is in position of a monopoly or an oligopoly.
- **Threat of substitute products:** We ask if there are any substitute products or services, which are similar to ours. Lesser substitutes, the better.

1.3.3 McKinsey's 7S analysis

The 7S analysis was developed by business consultants Robert H. Waterman, Jr. and Tom Peters in the 1980s. It is used for evaluation and monitoring of the internal situation of an organization.

The core concept of the analysis consists of 7 elements, which need to be aligned and mutually reinforcing (see Figure 1.9)[56]. They are[25]:

- **Strategy:** Strategy is characterized by the long-term orientation of the company, its goals which it can realize. The most common strategies are the cost leadership strategy and the differentiation strategy.
- **Structure:** Structure defines how the roles and responsibilities in the company are distributed. Every company should have an organizational structure, which suits its needs the best. We can recognize various typical organizational structures – e.g. linear, functional, divisional, etc.
- **Systems:** Represents all of the information systems and procedures, which are in use.
- **Shared Values:** Also a company's culture, represents a collection of myths, practices and values, which are common in the company.
- **Style:** Style of the company's management differs, but we recognize three basic principles – authoritative, democratic and liberal.
- **Staff:** People are the main resource, which helps to improve the company's performance. They have to be properly compensated and company has to create a good working environment for them.
- **Skills:** Defines the core skills and competencies of the company. Nowadays, the most valuable skill is to be able to quickly adapt to the new circumstances.

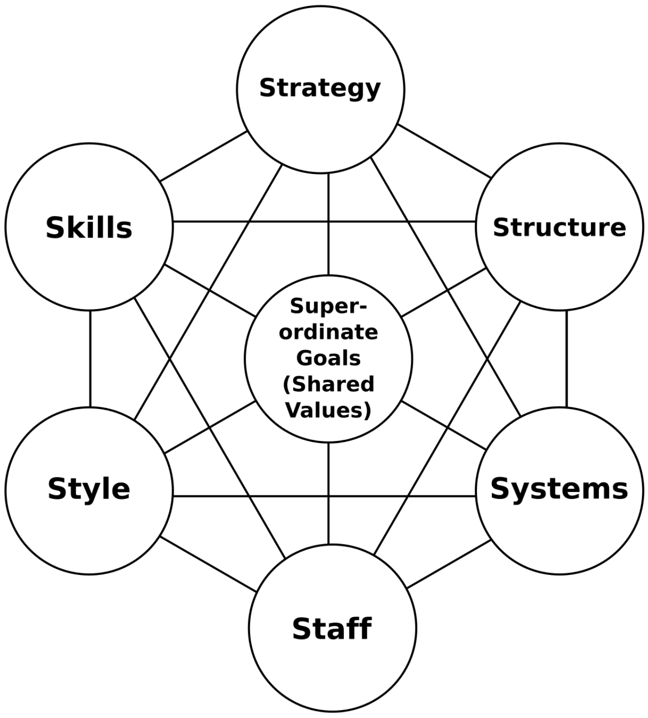


Fig. 1.9: Visual representation of the 7S analysis model[38]

1.3.4 SWOT analysis

SWOT analysis combines previously described methods. It should synthesize both internal and external factors. We can imagine it as a matrix of four quadrants (see Figure 1.10)[47]:

- **Strengths:** Describe, where the company is better compared to its competitors – e.g. quality management, unique products, etc. More strengths the company have, the better.
- **Weaknesses:** It is necessary to reason about the company's weaknesses – e.g. not enough experience, bad pricing of the products, etc. We should then try to come with solutions to these weaknesses.
- **Opportunities:** We should identify and compare possible opportunities based on its attractiveness and feasibility. The should point us in the right direction on where to focus company's energy and can also help to identify possible risks and problems.
- **Threats:** It is necessary to identify all of the possible threats to the company's business. We cannot hide any facts, which can be negative – e.g. competition, various barriers, etc.

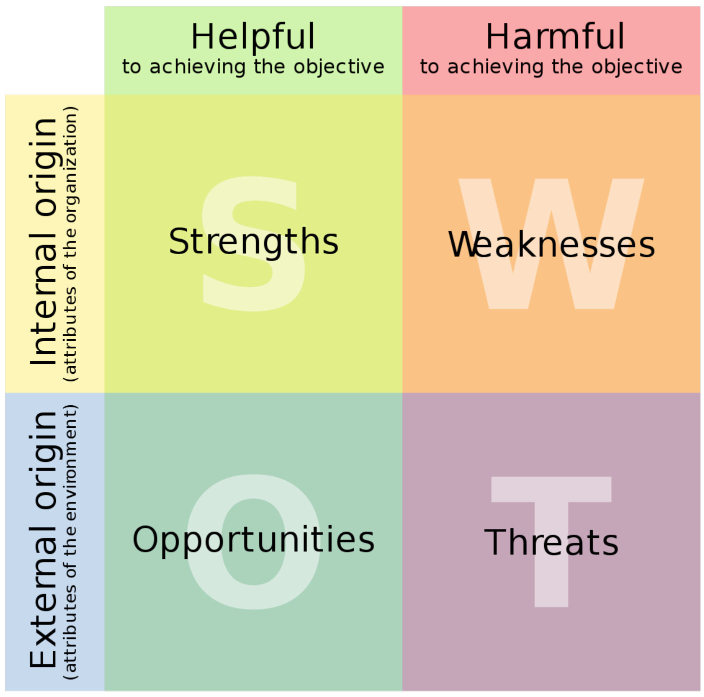

Helpful
to achieving the objective
Harmful
to achieving the objective
Internal origin
(attributes of the organization)
External origin
(attributes of the environment)
Strengths
Weaknesses
Opportunities
Threats
S
W
O
T

Fig. 1.10: SWOT analysis matrix[57]

1.3.5 Lewin's change model

Lewin's change model was developed by Kurt Lewin. It is concerned with implementing a change in an organization. According to Lewin, each successful change consists of three steps (see Figure 1.11)[25]:

- **Unfreezing:** In the first step of the Lewin's change model, we have to analyze a situation in the company. This is achieved by the use of force field analysis, where we compare forces which are driving the change with forces which are against the change. If the driving forces are stronger than the restraining forces, we can proceed.
  In the next step, we have to identify an agent of the change. It can be an individual or an organization, which helps to realize the change. There are two other actors, which should be identified – sponsor (who provides the resources needed for the change) and advocate (who supports the change, but is not directly involved in its implementation).
  In the final step of the unfreezing step, we identify areas of the intervention – areas, which are affected by the change.
- **Changing:** In the second step, we implement the change. Details of the implementation differs with every change.

- **Refreezing:** In the last step, we refreeze the implemented change. We interpret the results of the change and if it was successful, we try to fixate the results.

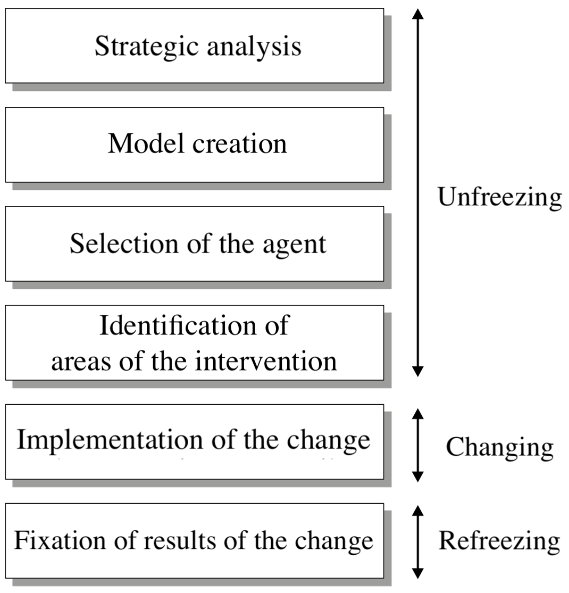

Strategic analysis
Model creation
Selection of the agent
Identification of areas of the intervention
Implementation of the change
Fixation of results of the change
Unfreezing
Changing
Refreezing

Fig. 1.11: Lewin's change model steps[25](edited)

1.3.6 Risk analysis

We can define `risk`, as a threat level of an asset, where the threat is fulfilled and it causes an unwanted results or some loss[25](author's translation).

In the risk analysis, we[25]:

1. Identify possible risks and their causes, which can negatively affect an asset, or a change process.
2. Determine serverity of the identified risks.
3. Propose solutions on how to deal with the risks:
   - Implement mitigations, which lowers the risk severity.
   - Accept the risk.
   - Avoid the risk.
   - Transfer the risk to a third party.

We can categorize methods of the risk analysis into two categories[25]:

- **Qualitative methods:** They are based on a description of the risk's potential impact and probability of its occurrence. We can score the risk impact and its probability using a scale (e.g. from 1 to 10), or verbally (small, medium, large). Qualitative methods are relatively easier to perform, but we cannot exactly calculate a possible financial costs of the risks. They are used in cases, where we cannot obtain quality data to be able to quantify the risks.

- **Quantitative methods:** They are based on the mathematical calculation of a risk, which is calculated using the frequency of its occurrence and its severity. Commonly we use the annualized loss expectancy metric, which is represented as the amount of assumed financial loss. Quantitative methods are more exact, but harder to perform. Its formalized nature can also cause an overlook of some specifics of the risk, which can result in a wrong calculations and potential loss.

1.3.7 PERT analysis

Program Evaluation and Review Technique (PERT) was developed for American navy by several companies such as Lockheed or Allen & Hamilton. It allows for a flexible creation of project schedules. It handles possible changes in the plan and it is used primarily for projects, where the activity length cannot be precisely estimated. It uses probability for the activity duration estimation, which is computed as the mean of the optimistic, pessimist and the most probable duration[49].

A critical path is computed as the result of the PERT analysis. It is the longest path of scheduled activities that must be met to execute a project. Activities on the critical path are critical – if an activity is delayed, project duration increases[49].

To construct the analysis, we use a finite, continuous, oriented, acyclic, edge-rated or node-rated graph (see Figure 1.12) that expresses dependencies on individual activities[21].

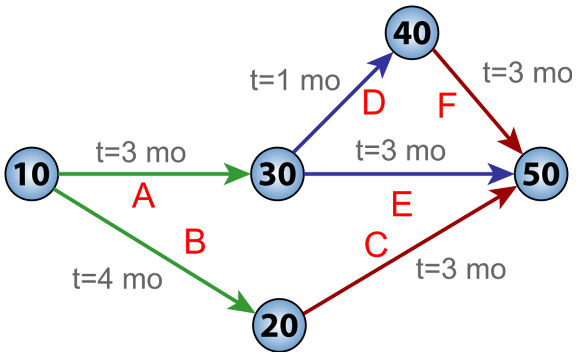


Fig. 1.12: PERT graph example[22]

2 Problem analysis and current situation

The content of this chapter is focused on analysis of the current situation of inventory management in the company DERMAREVOLTA s.r.o. for which the information system will be built. In the first section the company is briefly introduced, following the analysis of the external and internal factors. Finally, the current state of inventory management is described and conducted analysis are summarized.

2.1 Company introduction

The company is a private dermatology and aesthetic medicine clinic based in Bratislava, Slovakia. Thanks to its location near the borders of Austria and Hungary it has an international clientele. It specializes in dermatovenereology - diagnoses and treatment of various skin diseases (acne, psoriasis, skin tumors, etc.) and in the aesthetic medicine such as hair mesotherapy, laser hair removal and dermal fillings. Since its establishment in 2013, it has built a very good reputation and despite its small size the company employs some of the best professionals in the industry and uses the state of the art technology. In addition to the medical practice, the company sells a variety of dermatology products.

Business name	DERMAREVOLTA, s.r.o.
Legal form	company with limited liability (s.r.o.)
Address	Smaragdová ul. 1, 851 10 Bratislava
Date of establishment	October 16th, 2013
Number of employees	13
Main lines of business	• Provision of health care with a professional focus on dermatovenereology and aesthetic medicine • Purchase of goods for the purpose of its sale to the end consumer (retail) or other businesses (wholesale)

Tab. 2.1: General information about the company[46]

2.2 PESTLE analysis

2.2.1 Political factors

Part of the dermatological products the company sells and the medical supplies it uses is imported from the countries outside of the European Union – mainly U.S.A. The company is therefore affected by the international political situation, mainly by import duties and possible import restrictions. These can negatively affect prices of the dermatological products and medical supplies.

2.2.2 Economic factors

There is a growing issue with lack of a qualified medical personal in Slovakia[4]. The reason for this is mainly labor mobility, where educated young people have a tendency go work and live in the richer Western European countries. This problem is not as profound in the private sector of the business, it is far more prevalent in the state operated hospitals. Nevertheless, it pushes companies to offer a competitive financial compensation and other benefits to a potential employees.

Consumer prices are negatively affected by the inflation, which is currently at its highest in 17 years at the rate of 10.4%. Median income is not growing proportionally to the inflation. This means that an average Slovakian citizen can afford less goods and services[48]. Services the analyzed company provides, mainly aesthetical dermatology, are not essential and an average citizen will try to reduce his spending by not using them well before he cuts his essential needs. This can result in lower company's profits.

2.2.3 Social factors

Current social factors and trends have a good overall impact to business of the analyzed company. There is a trend in current society to use services of the aesthetic dermatology. An average age of people using these services is lower, which brings the company new customers[15].

The one socio-economical problem to pay attention to is that of ageing of the doctors, as mentioned in the Section 2.2.2.

2.2.4 Technological factors

Technology in dermatology is progressing at a fast pace. Laser-based procedures dominate the market owing to increasing inclination towards minimally invasive procedures, less pain, faster recovery option, and less hospital stay, as is concluded in the Europe Aesthetic Medicine Market report[15].

Furthermore, businesses are leaning towards informatization and digitalization of their processes. The widespread use of information systems allows them to optimize the costs and reduce their staff. It is important to follow this trend and integrate new information systems to the company's workflow.

2.2.5 Legal factors

Every medical clinic has to conform to a large number of hygienic standards. It is therefore necessary to keep an eye on the current legislation and to have all the necessary certifications and licenses. New legislation concerning the processing of the biological waste is causing a large increase of the expenses. Company has to comply with the GDPR legislation to avoid unnecessary fines.

2.2.6 Environmental factors

The company has to ecologically process all the produced waste, including the biological waste, which is produced during the procedures. This presents the company with additional expenses, but we can conclude that the business is not directly negatively affecting out environment.

2.3 PORTER analysis

2.3.1 Competition in the industry

State operated dermatology clinics

State operated dermatology clinics presents a competition in the area of diagnostics and treatment of the skin disorders. In area of the aesthetic dermatology they are not a serious threat. More wealthy clients prefer private clinics, mainly because of the quality of the services. The analyzed company employs several doctors, who were previously employed in a state hospital, thanks to the better financial compensation and working conditions.

Interklinik

Interklinik is the current leader in private dermatology clinics in the region. It has 23 year long tradition and operates in two cities – Bratislava and Poprad. It has a substantial client base and provides services in several medical fields, e.g. ORL, stomatology, urology. This provides Interklinik a large competition advantage, because when a patient visits another doctor within this clinic, it is probable he will also visit a dermatology services there. The main goal of the Interklink is to remain the local market leader and further expand its business. There exists a possibility of a fusion with other, smaller clinics in the area.

Inštitút estetickej medicíny (IEM)

IEM is a private clinic of aesthetic dermatology, similar in size to the analyzed company. It is far ahead in the area of digital marketing and web presence. It has a modern web with client reservations and plans to open an online shop. It is opened 7 days a week 12 hours a day, which the clinic promotes as a big advantage over competition. They provide financing of the more expensive procedures – clients can pay in a form of installments. They cannot compete with the analyzed company in terms of quality of the services. They do not provide diagnosis and treatment of a serious skin diseases. The analyzed company has to pay more attention to the digital marketing and web presence, because IEM can threaten its position if it broadens its services.

2.3.2 Potential of new entrants into the industry

Potential of new entrants into the industry in the local market is rather small. It is mainly due to the market being saturated and necessity of a high initial investment. High property and medical instruments prices, lack of qualified doctors, demanding hygienic standards and certification are all working against a potential new competition. A fusion of the existing, smaller clinics therefore presents a greater threat.

2.3.3 Power of suppliers

There is a high demand of the filling materials in the market. Suppliers of these medical material have a strong negotiating position. Common medical material has a better availability and can be acquired relatively cheaply when ordering wholesale. There exists a strong rivalry among the suppliers.

2.3.4 Power of customers

Customer can choose among a large number of dermatology care providers. Clinics run various discount promotions, provide discount vouchers and fidelity cards. Wealthy clients are often in focus of the dermatology clinics. They prefer quality over low price and require an individual approach. We can conclude that the customer has a great power in the market.

2.3.5 Threat of substitute products

There are various dermatologic and aesthetic products in the market (creams, hair preparations, etc.), but they cannot substitute a dermatologist. Medical care, professional diagnostics and treatment of skin diseases can hardly be subsidized by a substitute products. Threat of substitute products is therefore small.

2.4 McKinsey's 7S analysis

2.4.1 Strategy

The company's strategy is to provide the best dermatological care with a great focus on individual approach to each customer. It focuses primarily on a wealthy clientele. It is an important strategy to acquire foreign customers. Duration of each procedure is far longer compared to competition. Doctors offer free consultation to regular customers. The goal of the company is to build a family-like atmosphere in the clinic. We can conclude, that the company's strategy is focused on differentiation.

2.4.2 Structure

The analyzed company has a flat organization structure. Owner of the clinic is also a doctor. There are two managers, several doctors (dermatologists) and one surgeon. Integral part of the clinic is the receptionist and support staff.

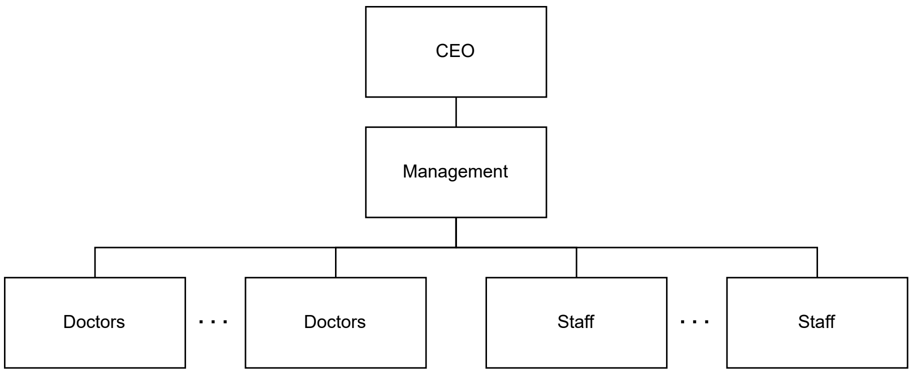


Fig. 2.1: Organizational structure of the analyzed company

2.4.3 Systems

The company uses several information systems. It is primarily an accounting system and reservation system. These information systems lack a common integration and it causes a necessity to fill some data manually multiple times.

It is very unfortunate that the company does not use an inventory management system. Existing solution is based on Excel tables (see Section 2.6), which is highly ineffective and does not suit the amount of goods, which are used in procedures or sold to customers.

2.4.4 Shared Values

The core value of the company is an individual, human approach to all customers and also between the staff. It is integral value of the company to provide the best possible service and aspire to be a market leading clinic.

2.4.5 Style

Employees of the clinic have a lot of freedom in the work they do. There is a functional, proactive communication between the regular employees and the top management. Every employee can and is encouraged to contribute to new projects and strategies and their views are considered when company makes a strategic decisions.

2.4.6 Staff

Staff and mainly the qualified medical personal is crucial to the company's success. The company tries to hire and keep the best doctors in the market by providing them a competitive financial compensation and other benefits, such as continual education. Doctors of the clinic regularly visit international conferences and undergo various certifications, which helps to build the clinic's prestige.

2.4.7 Skills

Doctors of the clinic are highly educated and skilled professionals. Nevertheless it is necessary to keep with the times and provide them with a continual education. Some employees have insufficient skill using computers and new information systems. This presents a problem considering the ever increasing rate of integration of these information systems in clinic's processes.

2.5 SWOT analysis

2.5.1 Strengths

- **Existing clientele:** The company has built a good reputation and it manages to acquire a loyal clientele. This is possible thanks to the differentiation strategy, focus on wealthy customers and individual approach.
- **Top doctors:** The clinic employs doctors who are the leading professionals in the local market. This results in the company's competition advantage. Some of the doctors are active in various media (TV, radio), which raises an awareness of their quality and also of the clinic itself.
- **Top quality instruments and used materials:** The top quality instruments and materials are used in every procedure. Some of them are imported from the countries outside of the European Union, mainly U.S.A. The company does not try to save in this aspect and has built a reputation based on this fact.

2.5.2 Weaknesses

- **Non-existent inventory management system:** The existing solution of inventory management is not at all sufficient and cannot scale as the company grows. There is a high risk of errors in the data due to the manual filling of information and lack of automatization, which can cause a negative consequences in strategic planning and in audits.
- **Low computer literacy:** Computer illiteracy presents a problem mainly with the older doctors. It is necessary to improve the computer literacy of all employees, more so, if the company is planning to introduce new information systems in its processes.
- **Sub-optimal web presentation and digital marketing:** The current website of the clinic was built in 2014 and it is build on a sub-optimal framework. Non-programmer staff cannot edit the pages and SEO of the website is not at all ideal. In the current world of social media it is very important to have a good digital marketing. Therefore it is necessary to improve the website and social media presence if the company aspires to be a leader in the local market.

2.5.3 Opportunities

- **Integration and automatization of the information systems:** The company does not have a suitable inventory management information system. Currently it uses spreadsheets for this purpose. This solution was somewhat usable while the amount of sold products was rather small. Currently, with a raising demand, a need for a better solution of inventory management became ever more pressing. Introduction of the new inventory management system can highly improve and speed up the clinic's processes and save the time, which can be used on further improvements in the medical care itself. Integration and automatization of inventory management lowers the risk of errors in the data. It is a foundation stone to a better digitalization of the company's processes, which enables to further expand its business.

- **Digital marketing:** Web presentation and digital marketing are on a sub-optimal level. This weakness presents an opportunity to improve these factors which can lead to more clients and higher sales.

2.5.4 Threats

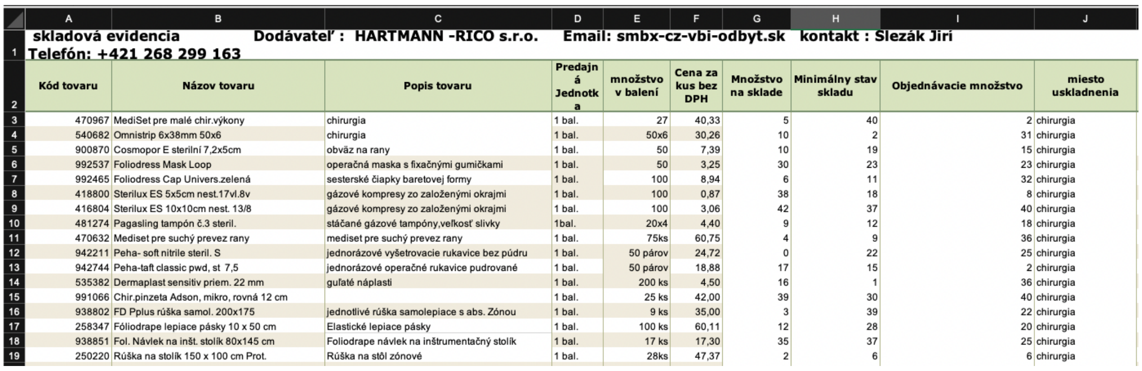
- **Lack of doctors:** Labor mobility and ageing of the doctors are a big threat to the company and the industry as a whole, as mentioned in the Section 2.2.2. The analyzed clinic has a quality doctor personal, but there is a threat of difficulties in trying to find a replacement doctor in a case of some unexpected situation, such as illness or maternal leave.
- **Competition:** There is a threat of competition, mainly in a form of a fusion of smaller clinics. Competition is also more advanced in the area of digital marketing, which can lead to less customers among the young generation.

2.6 Current situation of the inventory management

The current situation of the inventory management is less than ideal. It consists of one Excel file stored centrally on the company's network drive. It contains:

- Several sheets, one for each supplier with their contacts.
- Each supplier has many products.
- Each product has information about:
  - Order code
  - Name and description
  - Units in the package
  - Price per unit
  - Pieces in stock
  - Minimal quantity
  - Optimal quantity
  - Location

The Excel file (see Figure 2.2) is manually updated several times a week based on a manual check of the inventory. The history of changes cannot be tracked. You cannot make any informed managerial decision based on the history of the item's usage – there isn't any history of its movement. Validation of the entered data is non-existent. This can cause many discrepancies. The file is password protected, but once you have the password, you can see everything – there are not any user accounts, roles or special privileges present. Both company's management and employees are not satisfied with the current situation.



skladová evidencia Dodávateľ : HARTMANN -RICO s.r.o. Email: smbx-cz-vbi-odbyt.sk kontakt : Slezák Jirí
Telefón: +421 268 299 163

Kód tovaru	Názov tovaru	Popis tovaru	Predajná Jednotka	množstvo v balení	Cena za kus bez DPH	Množstvo na sklade	Minimálny stav skladu	Objednávacie množstvo	miesto uskladnenia
470967	MediSet pre malé chir.výkony	chirurgia	1 bal.	27	40,33	5	40	2	chirurgia
540682	Omnistrip 6x38mm 50x6	chirurgia	1 bal.	50x6	30,26	10	2	31	chirurgia
900870	Cosmopor E sterilní 7,2x5cm	obväz na rany	1 bal.	50	7,39	10	19	15	chirurgia
992537	Foliodress Mask Loop	operačná maska s fixačnými gumičkami	1 bal.	50	3,25	30	23	23	chirurgia
992465	Foliodress Cap Univers.zelená	sesterské čiapky baretovej formy	1 bal.	100	8,94	6	11	32	chirurgia
418800	Sterilux ES 5x5cm nest.17vl.8v	gázové kompresy zo založenými okrajmi	1 bal.	100	0,87	38	18	8	chirurgia
416804	Sterilux ES 10x10cm nest. 13/8	gázové kompresy zo založenými okrajmi	1 bal.	100	3,06	42	37	40	chirurgia
481274	Pagasling tampón č.3 steril.	stáčané gázové tampóny,veľkosť slivky	1bal.	20x4	4,40	9	12	18	chirurgia
470632	Mediset pre suchý prevez rany	mediset pre suchý prevez rany	1 bal.	75ks	60,75	4	9	36	chirurgia
942211	Peha- soft nitrile steril. S	jednorázové vyšetrovacie rukavice bez púdru	1 bal.	50 párov	24,72	0	22	25	chirurgia
942744	Peha-taft classic pwd, st 7,5	jednorázové operačné rukavice pudrované	1 bal.	50 párov	18,88	17	15	2	chirurgia
535382	Dermaplast sensitiv priem. 22 mm	guľaté náplasti	1 bal.	200 ks	4,50	16	1	36	chirurgia
991066	Chir.pinzeta Adson, mikro, rovná 12 cm		1 bal.	25 ks	42,00	39	30	40	chirurgia
938802	FD Pplus rúška samol. 200x175	jednotlivé rúška samolepiace s abs. Zónou	1 bal.	9 ks	35,00	3	39	22	chirurgia
258347	Fóliodrape lepiace pásky 10 x 50 cm	Elastické lepiace pásky	1 bal.	100 ks	60,11	12	28	20	chirurgia
938851	Fol. Návlek na inšt. stolík 80x145 cm	Foliodrape návlek na inštrumentačný stolík	1 bal.	17 ks	17,30	35	37	25	chirurgia
250220	Rúška na stolík 150 x 100 cm Prot.	Rúška na stôl zónové	1 bal.	28ks	47,37	2	6	6	chirurgia

Fig. 2.2: Current situation of the inventory management

2.7 Summary of the analyses

From the analyses we can conclude the current situation is not satisfactory and can causes many problems as the company expands its business. It is necessary to implement and adopt a new inventory management information system.

3 Implementation and contribution of suggested solutions

In this chapter, implementation of the inventory management system is discussed along with the impact and contribution it has on the employees and the company itself. First, the system requirements are created including EPC diagrams of the main processes. Following is the Lewin's change model analysis of the process of migrating to the new system. Next, the potential risks are identified and counter measures suggested. The proposed change time schedule is created via the PERT method along with the costs estimation. The implementation itself is discussed in detail, including deployment and running of the implemented application. Finally the information system is analyzed using ZEFIS[11] audit tool and contribution of the implemented application is analyzed along with the suggestion of further improvements.

3.1 Information system requirements

Current state of the inventory management is not acceptable. The old system of Excel tables is to be replaced by a modern information system. System requirements were composed based on the analysis in Chapter 2 and in cooperation with the clinic's top management.

System must be capable of:

- Creation, editing, viewing and removal of:
  - Warehouses and rooms
  - Products including their photographs and barcodes
  - Suppliers
  - Contacts
  - Receipts of goods
  - Users
  - Roles (administrator, manager, regular user, etc.)
- Stocking in and stocking out the goods by scanning its barcode or manually
- Providing various managerial outputs such as:
  - Insufficient goods
  - Goods with ending expiration
  - Transactions of goods with an advanced filtering
  - Audit of the information system – when the concrete user executed which action with what parameters

System must be smoothly put into the production, it cannot affect day-to-day operation of the clinic. It must be extendable and allow further improvements such as integrations with other information systems.

3.1.1 Analysis of inventory management processes

Two main processes of the inventory management were identified:

1. Adding items to the inventory
2. Removing items from the inventory

The EPC diagram was created for each process. They represent a new workflow using the new information system which will be implemented.

Adding items to the inventory

Every ordered item has to be registered in the new inventory management system. The process can be described in the following steps:

1. After the goods are delivered, a responsible employee (warehouseman) unpacks the shipment.
2. He checks the goods for the missing items and in case the order is incomplete, he makes a complaint using email or phone.
3. Next, for every item he scans its barcode, which is identified by the information system in the database.
4. In case the item is not found, he adds the barcode to the item's record.
5. Warehouseman then selects an expiration date for every item and room where the item will be stored.
6. Finally, he physically stores the items. The process is now completed.

Removing items from the inventory

Each time an item is physically removed from the inventory it should be registered in the information system. The process of removing the items from the inventory is described in the following steps:

1. An item is physically removed from the inventory – it is to be sold to the customer or used during the medical procedure.
2. Employee scans the item's barcode using his smartphone and the information system finds it in the database.
3. The removed item is precisely identified by filtering of its expiration date and placement in the clinic.

4. The identified stock is removed from the inventory using a confirmation button in the information system. The process is now completed.

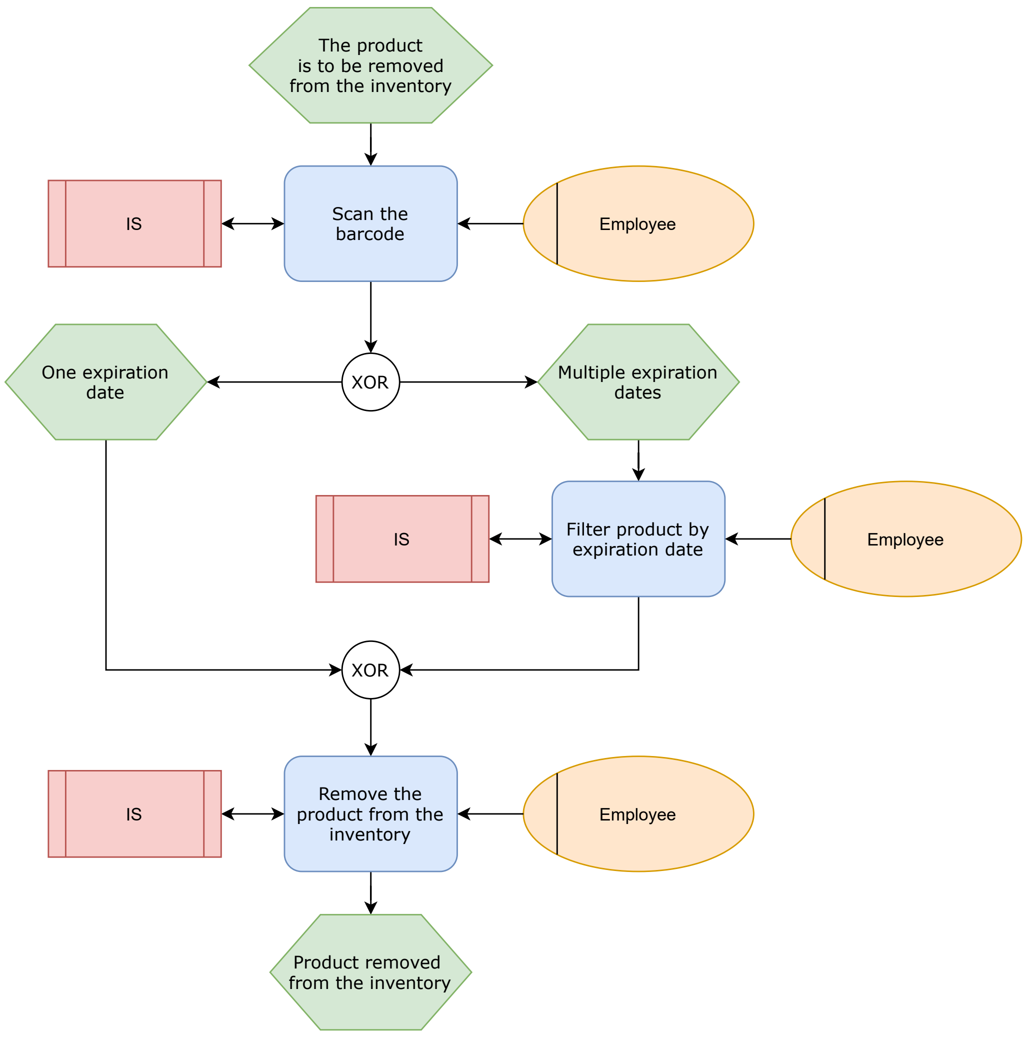

The product is to be removed from the inventory
IS
Scan the barcode
Employee
One expiration date
XOR
Multiple expiration dates
IS
Filter product by expiration date
Employee
XOR
IS
Remove the product from the inventory
Employee
Product removed from the inventory

Fig. 3.1: EPC diagram for removing items from the inventory

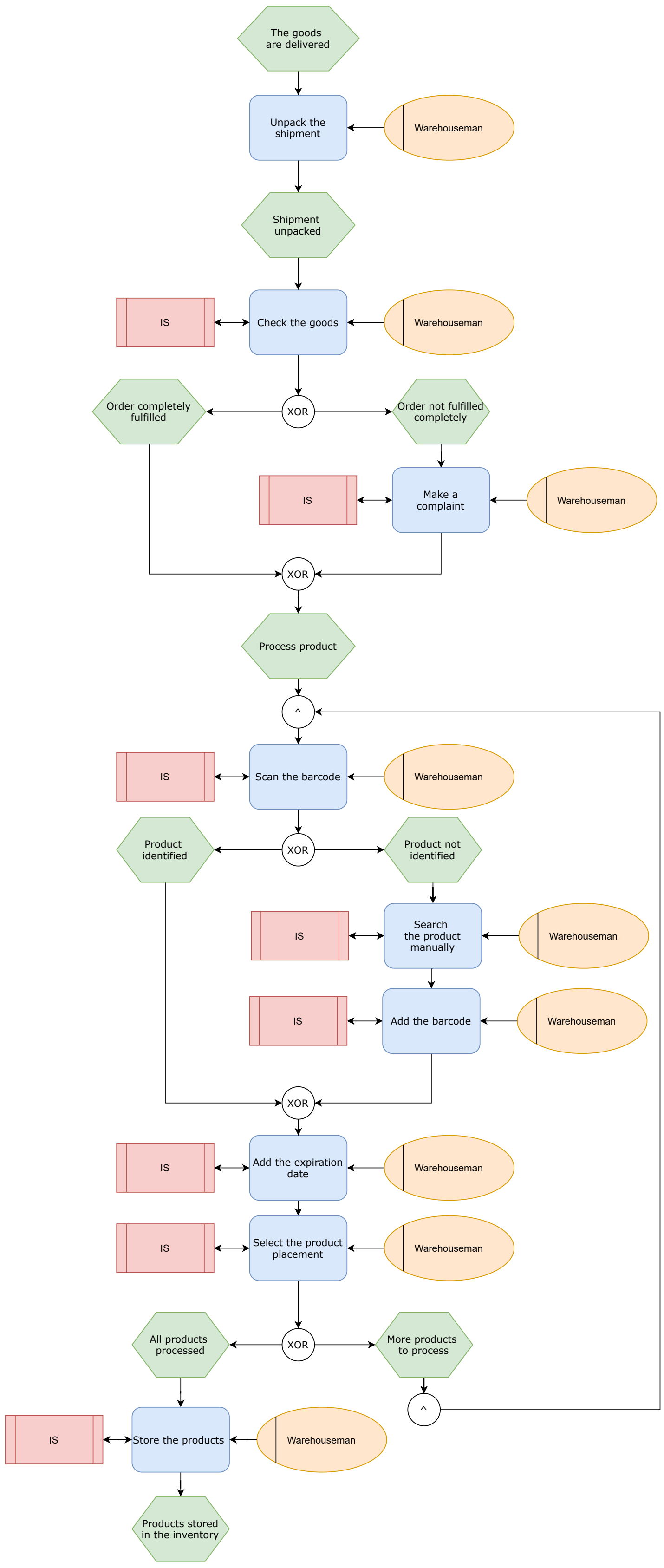


Fig. 3.2: EPC diagram for adding the items to the inventory

3.2 Lewin's change model

The Lewin's change model was applied to the problem of the new information system implementation and integration. All three phases of the model will be described in this section.

3.2.1 Unfreezing phase

It is necessary to inform all relevant employees about the change of the inventory management system. All of its benefits must be emphasized to allow the unfreezing.

Force field analysis

As the first step, force field analysis was performed. The results show that the driving forces are stronger than restraining forces, therefore the change is possible.

Driving forces		**Restraining forces**	
Description	**Force**	**Description**	**Force**
Need for a capable inventory management system	10	Lower computer literacy of employees	-7
Better data structure and integrity	9	Possibility of bugs during the first production run	-5
Automatization, saving time of the employees	8	Need for an employee training	-4
Better order planning thanks to various managerial outputs	6	Old data has to be imported	-6
Possibility of integration with the accounting information system	5	System operating costs (server, storage)	-3
User roles and different privileges – better data security	4	Implementation costs	-5
Possibility of further expansion and increase of sales	5		
Total	**47**	**Total**	**-30**

Tab. 3.1: Force field analysis

Roles in the change process

Roles in the change process were identified as the next step of the analysis.

Agent
The main agent of the change is the clinic's top manager, who enforced the change. He is responsible for a potential failure of the change. He represents an intermediary between the employees and the programmer, coordinates and plans the individual steps leading to the proposed change. He formulates the system requirements.

The next agent of the change is author of this thesis, who implements the information system. He implements the change and has a responsibility to ensure that the resulting system meets all required functionality. He corrects bugs and system errors and tries to meet the management's requirements.

Sponsor
The clinic owner is the sponsor of the change. She understands the need for a change and appreciates its benefits. She helps the agents with her resources. She also provides resources for the operational costs of the application.

Advocate
The main advocate of the change is the clinic's administrator, who also manages the inventory. The proposed change will greatly simplify his work and subsequently he will have more time for his other responsibilities.

Areas of the intervention

The proposed change will affect many areas of the business.

Technology
The new inventory management system will use modern technologies. Instead of a manual writing of product codes, user will be able to scan the product's barcode and stock it in or out easily. There is no need to buy new barcode scanners – thanks to the IS being a web application, employees can simply use their smartphones. It will save both the precious time and resources.

Processes
The inventory management processes will be simplified and automatized. Better data structure and integrity will minimize the discrepancies between the IS and the real state of the stocks. There will be no need to check the state of the inventory manually. System will notify the employee of the insufficient goods and prompt him to create an order.

Organization structure
The proposed change will not change the organization structure of the company.

Human resources
The proposed change requires to train the employees. It would be beneficiary to create a user manual. The company may need to employ an IT administrator if the information systems would grow large.

3.2.2 Change phase

The necessary steps leading to the proposed change were identified.

Step	Description
A	Decision on the implementation of the new information system
B	Creation of the system requirements
C	Consultation of the system requirements with the programmer
D	Implementation of the information system
E	Demonstration of beta version of the IS
F	Modification of requirements
G	Implementation of the modified requirements
H	Implementation of the system's hosting and CI/CD pipeline
I	Deployment of the IS
J	Testing of the IS
K	Handover of the final version of the system
L	Data migration
M	Data correction after migration
N	Barcodes scanning/import
O	Products photos import
P	Creation of the user accounts
R	Creation of the documentation
S	Training of the employees
T	Running the system in production for the first time
U	Creation and evaluation of the empoloyees feedback
V	Implementation of the proposed changes from the feedback
X	Fully running the system in production

Tab. 3.2: Steps leading to the proposed change

3.2.3 Refreezing phase

The agents of the change need to communicate with the employees and watch for an eventual bugs. The number of bugs, errors and missing features will be the metric for a successful integration of the proposed change. It is necessary to promptly solve all the bugs and errors to render the refreezing phase successful.

The fact that the information system will be built internally provides an opportunity to effectively change and improve it to suit the needs of the company.

3.3 Risk analysis

The suggested change was subjected to a risk analysis via the scoring method. First, the possible risks were identified based on the author's professional programming experience and top management's experience of managing a business (table 3.3).

Label	**Threat**	**Scenario**
A	A proper communication with the programmer is lacking.	System does not provide the requested functionality.
B	Deadlines and periodic checks were not clearly set.	System deployment is delayed.
C	Insufficient testing of the system.	System contains a lot of bugs.
D	The chosen server was not properly dimensioned.	System is slow.
E	Employees did not undergo a training.	Employees do not know how to use the system.
F	Automatic backup system was not setup.	Loss of an important data.
G	System is not secure enough.	Breach of a confidential data.
H	Employees do not have the required hardware.	Employees cannot scan the barcodes.
I	Data from Excel were not successfully migrated.	System's data is corrupted or incomplete.
J	Internet outage.	System is not usable.

Tab. 3.3: Identified risks of the proposed change

A scoring table was constructed for both the probability and impact. These criteria will help us evaluate the risk.

Value	**Percentage**	**Verbal evaluation**
1–2	0% – 19%	very improbable
3–4	20% – 39%	improbable
5–6	40% – 59%	probable
7–8	60% – 79%	more probable
9–10	80% – 100%	very probable

Tab. 3.4: Probability evaluation

Value	**Impact**	**Effect**
1 – 2	minimal	Delay > 7 days or Costs > +5%
3 – 4	minor	Delay > 14 days or Costs > +25%
5 – 6	medium	Delay > 30 days or Costs +50%
7 – 8	high	Delay > 60 days or Costs +75%
9 – 10	critical	Delay > 90 days or Costs +100%

Tab. 3.5: Impact evaluation

Next, each individual identified risk was evaluated based on the constructed scoring tables and the value of its risk was calculated as combination of the probability and impact. Also, a map of the evaluated risks was plotted.

Label	**Probability**	**Impact**	**Risk evaluation**
A	5	7	35
B	5	5	25
C	2	6	12
D	3	8	24
E	2	10	20
F	3	10	30
G	4	10	40
H	3	6	18
I	6	5	30
J	5	2	10

Tab. 3.6: Risk evalutation before the suggested measures

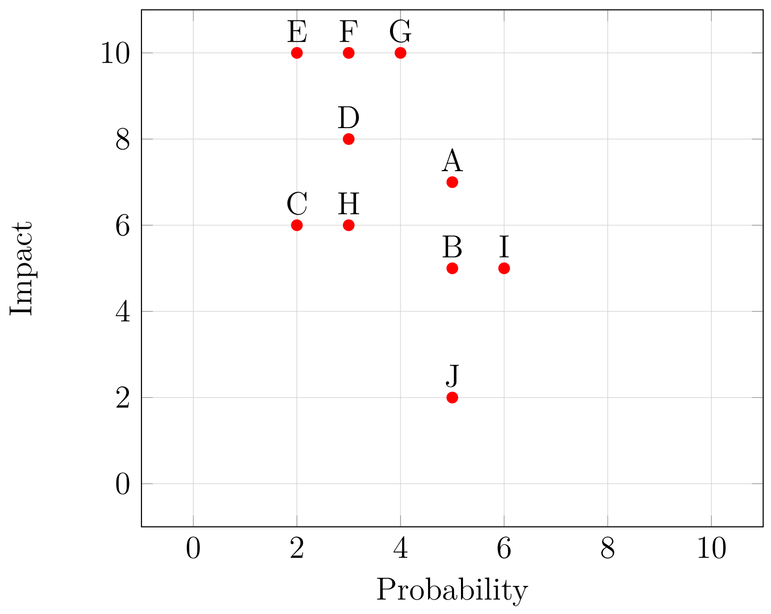


Fig. 3.3: Risk evalutation map before the suggested measures

Based on results of the risk evaluation, the identified risks are of relatively high probability and impact. They can seriously endanger the proposed change and both slow it down and cause the company a high loss. It is therefore necessary to try to mitigate these risks. The table of suggested measures (3.7) was created for this purpose.

Label	Threat	Scenario	Suggested measure
A	A proper communication with the programmer is lacking.	System does not provide the requested functionality.	Set clear rules of communication with the programmer and organize regular meetings.
B	Deadlines and periodic checks were not clearly set.	System deployment is delayed.	Create a project schedule with the expected duration for each task.
C	Insufficient testing of the system.	System contains a lot of bugs.	Write unit and integration tests, create test scenarios and also test the system with real people (employees).
D	The chosen server was not properly dimensioned.	System is slow.	Select a higher tier server with overdimensioned performance.
E	Employees did not undergo a training.	Employees do not know how to use the system.	Create an operating manual for the system, consider creating a help within the system.
F	Automatic backup system was not setup.	Loss of an important data.	Setup an automatic system backup using the 3-2-1 rule – three copies of data, backed up on two types of media, with one copy outside of the company's headquarters.
G	System is not secure enough.	Breach of a confidential data.	Use industry best practices when implementing the system.
H	Employees do not have the required hardware.	Employees cannot scan the barcodes.	Buy the missing HW before deploying the system.
I	Data from Excel were not successfully migrated.	System's data is corrupted or incomplete.	Give the programmer an access to the original data, rigorously test the migration on the data sample.
J	Internet outage.	System is not usable.	Use a high quality internet provider, consider self-hosting within the company's network.

Tab. 3.7: Suggested measures for the identified risks

Finally, the risks were evaluated after applying the suggested measures. The results shows that the risks no longer seriously endanger the suggested change.

Label	**Probability**	**Impact**	**Risk evaluation**
A	3	3	9
B	3	4	12
C	1	2	2
D	1	1	1
E	1	1	1
F	1	1	1
G	2	5	10
H	1	1	1
I	3	3	9
J	3	1	3

Tab. 3.8: Risk evalutation after the proposed mitigation

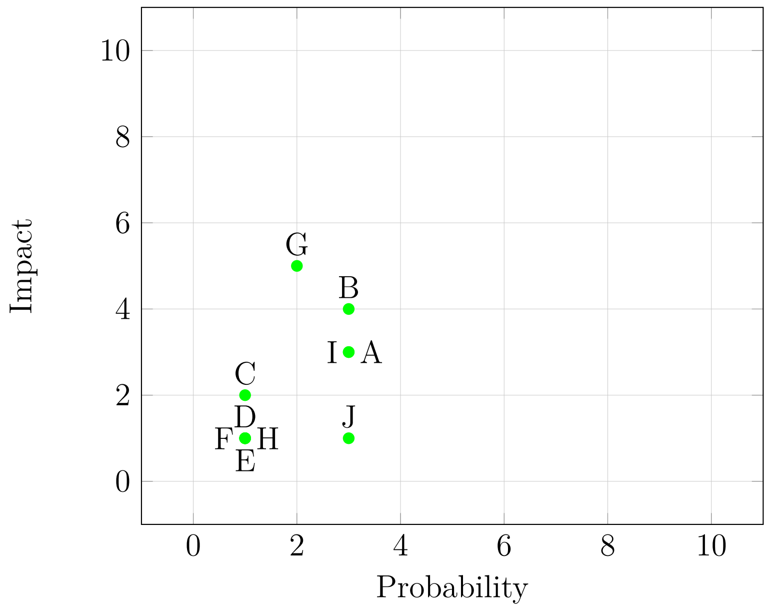


Fig. 3.4: Risk evalutation map after the suggested measures

3.4 PERT analysis

The PERT analysis was performed on the suggested change. This method was chosen because tasks durations cannot be exactly determined – they can be only estimated based on the author's experience. Also, the information system requirements are well defined – the project can be modeled like Waterfall.

All durations are measured in man days. The mean duration of each individual task (y_{ij}) was determined from the optimistic estimate (a_{ij}), pessimistic estimate (b_{ij}) and the most probable estimate (m_{ij}):

$$y_{ij} = \frac{a_{ij} + 4 * m_{ij} + b_{ij}}{6}$$

We also have to consider the following equations in the PERT analysis:

- **EF**(Earliest finish):
$$EF_{ij} = ES_{ij} + y_{ij}$$
- **LF**(Latest finish):
$$LF_{ij} = EF_{ij} - y_{ij}$$
- **TR**(Total reserve):
$$TR_{ij} = TP_j - TM_i - y_{ij}$$

A time schedule of the proposed change was created. It includes all the task from the Lewin's change model (section 3.2). As we can see in the Table 3.9, several task can be performed in parallel.

Next, the PERT chart and graph were created. We can determine a critical path, which shows, that the overall duration of the proposed change is estimated at 234 man days. The most demanding task is an implementation of the information system itself. It can take up to 150 man days. Several activities can be performed in parallel, such as barcode scanning/import, product photos uploading, documentation creation etc.

Critical path:
$A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \rightarrow G \rightarrow I \rightarrow J \rightarrow K \rightarrow R \rightarrow S \rightarrow T \rightarrow U \rightarrow V \rightarrow X$

Task	**Predecessor**	$\mathbf{a_{ij}}$	$\mathbf{m_{ij}}$	$\mathbf{b_{ij}}$	$\mathbf{y_{ij}}$
A		1	2	3	2
B	A	7	11	14	11
C	B	1	1	2	1
D	C	100	125	150	125
E	D	1	1	2	1
F	E	5	7	10	7
G	F	10	15	30	17
H	C	1	2	3	2
I	G, H, M	3	4	5	4
J	I	5	7	10	7
K	J	1	2	3	2
L	C	5	7	10	7
M	L	3	4	5	4
N	P	1	2	3	2
O	P	1	2	3	2
P	K	1	1	2	1
R	K	10	12	14	12
S	N, O, R	5	7	10	7
T	S	1	2	3	2
U	T	5	7	10	7
V	U	10	15	30	17
X	V	5	7	10	7

Tab. 3.9: Time schedule of the proposed change

Task	**Description**	**i**	**j**	**a_{ij}**	**m_{ij}**	**b_{ij}**	**y_{ij}**	**ES**	**EF**	**LS**	**LF**	**TR**
A	Decision on the implementation of the new information system	1	2	1	2	3	2	0	2	0	2	0
B	Creation of the system requirements	2	3	7	11	14	11	2	13	2	13	0
C	Consultation of the system requirements with the programmer	3	4	1	1	2	1	13	14	13	14	0
D	Implementation of the information system	4	5	100	125	150	125	14	139	14	139	0
E	Demonstration of beta version of the IS	5	6	1	1	2	1	139	145	139	145	0
F	Modification of requirements	6	7	5	7	10	7	145	152	145	152	0
G	Implementation of the modified requirements	7	8	10	15	30	17	152	169	152	169	0
H	Implementation of the system's hosting and CI/CD pipeline	4	8	1	2	3	2	14	16	167	169	153
I	Deployment of the IS	8	9	3	4	5	4	169	173	169	169	0
J	Testing of the IS	9	10	5	7	10	7	173	180	173	180	0
K	Handover of the final version of the system	10	11	1	2	3	2	180	182	180	182	0
L	Data migration	4	12	5	7	10	7	14	21	158	165	144
M	Data correction after migration	12	8	3	4	5	4	21	25	165	169	144
N	Barcodes scanning/import	16	17	1	2	3	2	183	185	192	194	9
O	Products photos import	16	17	1	2	3	2	183	185	192	194	9
P	Creation of the user accounts	11	16	1	1	2	1	182	183	191	192	9
R	Creation of the documentation	11	17	10	12	14	12	182	194	182	194	0
S	Training of the employees	17	18	5	7	10	7	194	201	194	201	0
T	Running the system in production for the first time	18	19	1	2	3	2	201	203	201	203	0
U	Creation and evaluation of the empoloyees feedback	19	20	5	7	10	7	203	210	203	210	0
V	Implementation of the proposed changes from the feedback	20	21	10	15	30	17	210	227	210	227	0
X	Fully running the system in production	21	22	5	7	10	7	227	234	227	234	0

[1] **ES:** Earliest start
[2] **EF:** Earliest finish
[3] **LS:** Latest start
[4] **LF:** Latest finish
[5] **TR:** Total reserve

Tab. 3.10: PERT chart

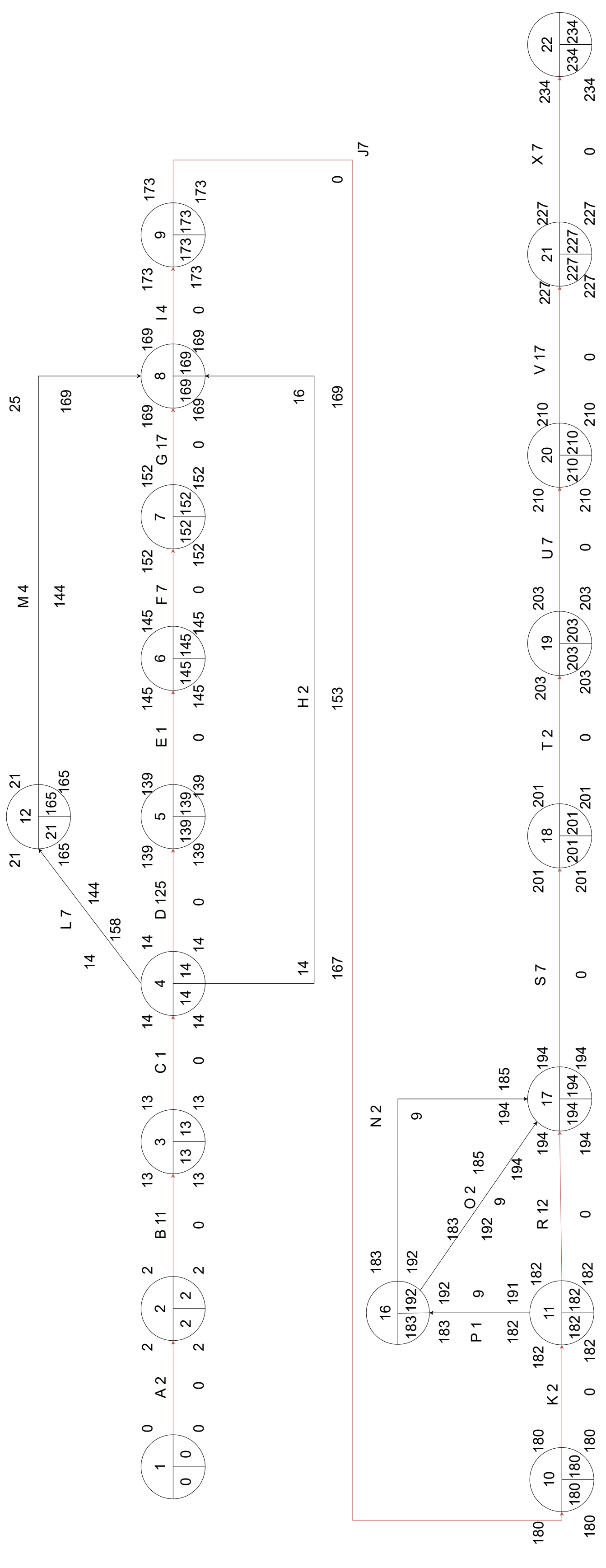


Fig. 3.5: PERT graph

3.5 Costs estimation

The majority of expenses is made of the programmer's compensation. We also have to consider the cost of employees training and creation of information system's documentation. A substantial amount can be saved on barcode scanners, as the employee's smartphones can be used.

Name	Optimistic estimation in €	Pessimistic estimation in €
Programmer's compensation	20 000	30 000
Documentation creation	500	1000
Employees training	2 000	5 000
Total	**22 500**	**36 000**

Tab. 3.11: Costs estimation of the suggested change

There will be a recurring expense in form of the information system hosting costs. We've tried to pick the best option in terms of both performance and costs as described in Section 3.6.3. We also have to consider costs of the system maintenance – the system and all of its components must be kept up-to-date, the programmer has to regularly fix the bugs and small improvements or features may be added on a regular basis.

Name	Optimistic estimation in €	Pessimistic estimation in €
Monthly cost of the hosting of the information system	30	50
Programmer's monthly maintenance budget – bugfixes, small improvements	300	500
Total monthly expenses	**330**	**550**

Tab. 3.12: Costs estimation of the recurring expenses

It is probable, that the monthly maintenance budget will be considerably lower after the initial bugs and features are patched. From this point on, we can consider minimal maintenance budget and monthly cost of the system's operation will consists mostly of the hosting expenses.

3.6 Application architecture and implementation

It was decided to implement the information system as a web application. The nature of the web applications provides several benefits both to the end user and software developers. The end user can use the application from practically any modern device which can connect to the internet – he is not limited by the operating system and hardware resources as the application resides on a server. Software developers have much better control of the whole user experience compared to a traditional desktop applications. The application can be easily updated (see Section 3.6.3) and users always see the current version.

The implemented application is a Single Page Application or SPA (see Chapter 1). The backend is implemented as REST API and thus it is decoupled from the frontend side of the application. This has a great benefit as the frontend can be developed independently from the backend and the same backend API endpoints can be reused with another frontend – for example a mobile application or a 3rd party service. Data is stored in a relational database. All of the assets (images, static files) are stored in an object storage. The general overview of the application architecture is shown on the Figure 3.6.

In the next sections the application implementation will be explained in detail. The source code of the information system is fully open source and can be found on `https://github.com/lifo9/skladis-backend` and `https://github.com/lifo9/skladis-frontend` respectively. It is licensed under GPL-3.0 license.

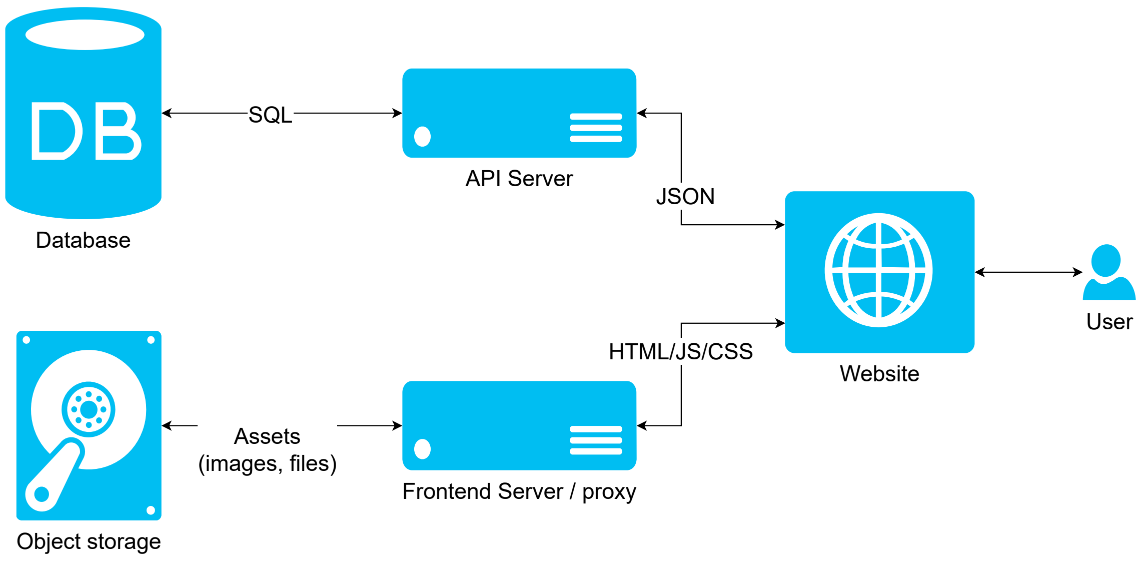


Fig. 3.6: Application architecture overview

3.6.1 Backend

Backend of the implemented information system is discussed in this section. The backbone of the application – database and its models are explained first, following the application logic, its API and the most important methods and constructs. Next, the authentication and authorization is explained along with the storage solution.

Database

The PostgreSQL database was chosen as the database solution for the implemented information system. It is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads[51]. It has been ACID-compliant (see Chapter 1) since 2001.

The database was not created via raw SQL commands – the ORM layer of the framework was used as will be discussed in the Section 3.6.1. This way, we can see a complete history of each database structure change and easily rollback to a previous version in case of a bug or another problem.

The database model was created following the system requirements as discussed in Section 3.1. The individual tables and relationships can be seen in the Entity Relationship Diagram (A.1). The tables were created in the Boyce–Codd normal form (see Chapter 1).
Next, we will discuss the most important tables:

- **Warehouses**
  The clinic can have multiple warehouses where the goods are stored. It holds the name of the warehouse and its address reference. The address is modeled as a separate table, because it is used for multiple models.
- **Rooms**
  Each warehouse can contain multiple rooms. This entity represents the most granular location of a given item. It can thus be identified by the warehouse it is located in and a room it is stored in.
- **Products**
  This table holds information about the products. It contains the name, order code, ideal and critical number of pieces of the product, a reference (foreign key) to the product's barcode and the suppliers references (via a junction table).
- **Barcodes**
  Each product has one barcode. It stores the information about the barcode type and its code.

- **Suppliers**
  Product can have multiple suppliers. Supplier has its name, VAT number, website URL, address reference and contact reference, which is modeled as a separate table.
- **Stocks**
  Stocks table holds information about the individual items in the inventory. It contains references to the product and room, where it is stored, number of pieces and expiration date. A unique key is set on these three columns:

  ```
  product_id, room_id, expiration
  ```

  This is because the stock is uniquely identified by these three values and thus it is not desirable to create duplicates. The logic behind the stock creation will be discussed further in the Section 3.6.1.
- **Stock transactions**
  A new row in the stock transaction table is created whenever user manipulates a stock. It thus contains reference to the stock, number of pieces and the action, which is an enumerate of:

  ```
  stock_in, stock_out
  ```

  We can also model a stock transfer to another room or warehouse via this structure. The logic behind the stock transactions will be discussed further in the Section 3.6.1.
- **Invoices**
  This table could be called Receipts of goods, but the author wanted to use a shorter form. In conjunction with the Invoice items table it contains the information about the ordered products – invoice code and date, which user created the invoice, wether the goods are stocked in etc. The Invoice items table contains the invoice, product and supplier's references, unit price of the product and its quantity.
- **Users**
  Users are an integral part of the information system. Each user has his own account. This table stores the user's email, first and last name, phone, password digest (a hashed password) and a boolean column, which determines if the user's account is active or not.
- **Roles**
  This table stores the system's roles (administrator, regular user, etc.). It is used in the junction table to give each user a role (or more roles). This

information is later used to determine if the user has an access to a given resource.

- **Active storage tables**
  Many models can have associated files or images. These tables hold the information about these associations and about the blobs (files) – file name, content type, key or URL etc.

There are other tables in the database which were not discussed in detail – mainly helper tables and junction tables. These can be omitted as the core of the database model was successfully described without them.

Application logic

The backend of the information system was developed using the Ruby on Rails framework. It is based on the Ruby programming language. It greatly simplifies the development process and tries to optimize for the programmer's happiness[10]. It emphasizes use of convention over configuration (CoC), and don't repeat yourself (DRY) practices. It uses the MVC pattern, but we will be using only the Model and Controller part – the View will be created as a standalone Single Page Application (see Section 3.6.2).

As mentioned before, the backend is created as a REST API. It exposes many endpoints, which provide various functionality. The theoretical foundation on how the HTTP REST API server works is described in Chapter 1.

Each individual endpoint has its controller where the application logic is defined. A controller has multiple methods, which represent the individual endpoints. Inside these methods the appropriate services are called, data is manipulated, etc.

Next we will explain the most important controllers and services.

Application Controller

The core of the application logic is defined in the `ApplicationController` class. All other controllers inherit from it. It provides several public methods and callbacks, which are called upon each request. For example:

- `api_index`
  This method is used for each endpoint which lists the data for a given model. For example, if we want get a list of products, we leverage this helper method. It provides several arguments:
  - `model_class`: We have to pass the class of the model for which we want to get the listing.

– `params`: We pass the request parameters – such as search query, order, filters, etc.

– `associations`: Determines if we want to search in associations when using search query or filters.

– `custom_query`: It uses custom query when specified, else gets all items.

– `paginate`: Determines if we want to paginate the results – that is to return the HTTP response with a limited amount of items and providing links for the next/previous page in the HTTP headers.

We include the source code of the `api_index` method as an example:

```ruby
def api_index(model_class, params, associations = true,
    custom_query = false, paginate = true)
  if custom_query
    items = custom_query
  else
    items = model_class.all
  end

  if items.respond_to?(:api_filter)
    items = items.api_filter(params, associations)
  end
  if items.respond_to?(:search_all_fields)
    items = items.search_all_fields(params[:search], associations)
     if params[:search]
  end
  if items.respond_to?(:api_order_by)
    items = items.api_order_by(params[:order_by], params[:order],
    associations) if params[:order_by] || params[:order]
  end

  if paginate
    paginate items
  else
    items
  end
end
```

The result is then rendered by an appropriate controller in the JSON format. For example the API request `/products?page=1&per_page=1&order=asc&order_by=id` will return the following response:

```
{
  "data": [
    {
      "id": "1",
      "type": "product",
      "attributes": {
        "name": "Dermaheal HL",
        "order_code": "1",
        "pieces_ideal": 6,
        "pieces_critical": 2,
        "images": [
          {
            "id": 58,
            "url": "https://static.skladis.com/image123"
          }
        ],
        "barcode_type": "ean_13",
        "barcode_code": "8809171382265",
        "in_stock": 2
      },
      "relationships": {
        "suppliers": { "data": [{ "id": "36", "type": "supplier" }]
    }
      }
    }
  ]
}
```

Stocks Controller

We will describe the application logic on another important controller – the StocksController. Other controllers in the application follow similar patterns and their detailed description can thus be omitted.

There are multiple public methods defined in the `StocksController`, each one representing one endpoint. They are:

- `index`: Renders a JSON formatted list of stocks. Can be filtered, searched and sorted (as explained in the Section 3.6.1).
- `show`: Given a stock ID, it renders a JSON formatted details of the stock.
- `stock_in`: Creates a stock given the required parameters – `product_id`, `expiration`, `room_id`, `quantity`.
- `stock_out`: Similarly to the `stock_in` method, it decreases the quantity of a given stock.
- `stock_transfer`: Moves a stock from one room to another.

The logic behind creating, removing and transferring the stocks is governed by the `StockService` class. Thus the controller is kept clean and readable. For example the `stock_out` method is (from the controller's point of view) as simple as the source code below:

```ruby
# POST /stocks/out
def stock_out
  authorize Stock

  stock = @stock_service.stock_out(stock_in_out_params[:room_id],
   stock_in_out_params[:quantity])

  render json: StockSerializer.new(stock, { include: [:product, :
   room] })
end
```

First, the user is authenticated and authorized (will be described in Section 3.6.1), next the `StockService` is called with the given parameters and finally the response in form of a JSON containing the created stock is rendered.

Each controller in the application follows a similar pattern. First, the user is authenticated, next the called method is authorized and finally the operation is performed and the response is rendered.

Stocks Service

The `StockService` class contains the application logic concerning the stock movement. Every method works as a transaction – the operation is either 100% successfully completed, or it is not executed (and written to the database) at all. There are various validations present. The most important validation check is that after any operation the stock quantity cannot be negative. This helps to keep the data integrity. A stock transaction record is created with each performed stock operation. This allows to keep a log of every transaction and audit the changes made by the employees.

As mentioned in the database model description, each unique stock is determined by a three column unique index (`product_id, room_id, expiration`). This way we can keep a clean structure of the data. When we are for example adding a new stock, we first try to find an existing stock row matching the mentioned parameters and if it is found, we simply increment the `quantity` column. Else, a whole new row is inserted into the database.

The rest of the application logic follows a similar pattern. Each service contains required validations and checks to keep the data integrity and relevant database operations (`select`, `insert`, `update`, `delete`).

Individual queries are not written in the raw SQL language, but they are created using the Rails Active Model library (also called the ORM layer). This greatly simplifies the logic and also helps with security (e.g. it automatically escapes the query parameters to prevent SQL injection as discussed in Chapter 1).

As an example, the SQL query:

```sql
INSERT INTO stocks(product_id, room_id, expiration, pieces)
VALUES(12, 4, NULL, 10);
```

can be written using Active Model as simply as:

```ruby
Stock.create!(product_id: @product_id, room_id: room_id, expiration
   : @expiration, pieces: quantity)
```

Serializers

The frontend part of the application (will be discussed in the next section) consumes the created API in form of a JSON objects. The needed JSON object often differs from the representation given by the Active Model layer. There comes the need for a serialization.

Serializer allows us to format the response object, e.g. include data from another model. This is very useful in the frontend part of the application as we want to show the user rich dashboards with data across the models.

For a product, we may want to include, along with its primary data, its barcode (which is a different model) and image URLs. The serializer will then simply include these fields as in the example below:

```ruby
class ProductSerializer < ApiSerializer
  attributes :name, :order_code, :pieces_ideal, :pieces_critical

  attribute :images do |product, params|
    images_url = []

    product.images.sort_by(&:created_at).each do |image|
      type = params[:image_type] || :thumb
      images_url.push({
                        id: image.id,
                        url: attachment_url(image.variant(type))
                      })
    end

    images_url
  end

  attribute :barcode_code do |product|
    product.barcode&.barcode_code
  end
end
```

We can also specify different serialization attributes for different user roles. This comes useful in the `UserSerializer`, where we want to show information about active status of the given user only to the administrators:

```
# UsersController
...
# GET /users/1
def show
  authorize @user

  render json: UserSerializer.new(@user, { params: { admin:
    current_user.has_role? :admin }, include: [:roles] })
end
...
# UserSerializer
...
attribute :active, if: Proc.new { |_, params|
  params && params[:admin] == true
}
...
```

Application concerns

Concerns allow us to include modules with methods (both instance and class) and constants into a class so that the including class can use them[1]. This is very useful when we want to reuse some common logic across multiple classes.

In the implemented application we use concerns for the common interface for filtering, searching and ordering the model's data. This way, we only have to include the concerns in the model's class and without any other steps needed, we can filter, search and order its data.

```
class Product < ApplicationRecord
  include Searchable
  include Orderable
  include Filterable
...
```

We will discuss the `Searchable` concern as an example. It works as a scope. Scoping allows us to specify commonly-used queries which can be referenced as method calls on the association objects or models. With these scopes, we can use every ORM's method such as where, joins and includes[9].

The `Searchable` concern provides the `search_all_fields` scope, which takes two parameters – `searchTerm` (specifies the search query) and `associations` (whether to search in model's associations). The way it works can be described as such:

1. If we want to search in the model's associations, map all model's associations (available thorough the Active Model interface) to a hash containing its unique name as a key and an array of its column names as a value.
2. If we want to search in the model's associations, create a hash of all possible join tables names.
3. Create hash of all searchable columns in a form of: `table_name.column_name`. Use both model's own columns and its association's columns if needed.
4. Create a scope with a `DISTINCT SELECT` and `LEFT OUTER JOIN` on the join tables (if we want to search in the model's associations) and `WHERE` clause which contains all previously built searchable columns compared via `LIKE %%` operator with the `searchTerm`, connected via the `OR` operator.

```ruby
module Searchable
extend ActiveSupport::Concern

included do
  scope :search_all_fields, -> (searchTerm, associations) {
    ... # building of the searchable columns hash

    distinct
    .where("#{searchable_cols_strings.map { |col| "COALESCE(lower(#{col[:name]}), #{col[:name]}, '')" }
                                    .join(' || ')} LIKE ?", "%#{searchTerm.downcase}%")
  end
end
```

This concern then allows us to search any model by simply invoking:

```ruby
Model.all.search_all_fields(params[:search], true)
```

There would be a security risk of SQL injection (see Chapter 1) if the query parameters were not escaped. The Active Model handles this for us, so the use of this concern is safe.

Authentication and authorization

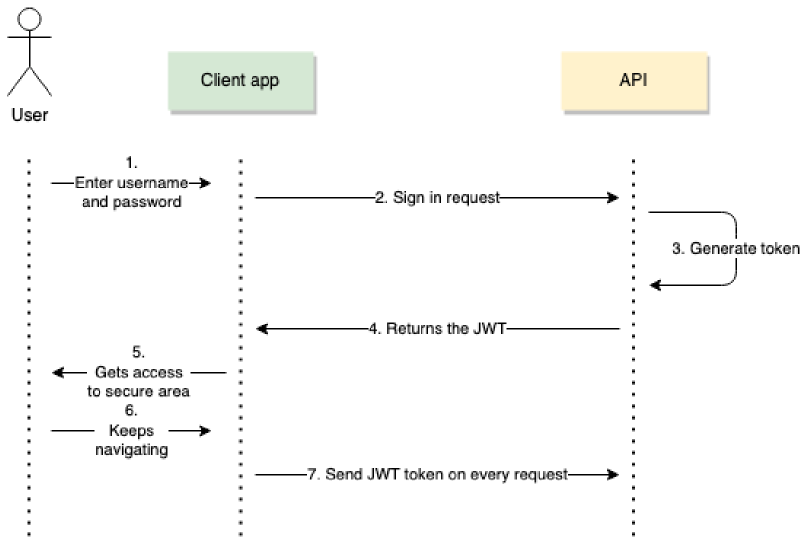
Authentication and authorization are an integral part of a secure application. It was implemented using JSON Web Token (JWT). The core concept behind JWT is that each user's session is represented by a pair of tokens: access and refresh. The access token is used to retrieve secure resources and the refresh token is used to renew the access token once it has expired. The tokens are stored in Redis store, which is an in-memory data store.

An access token is short lived (1 hour), while the refresh token has a relatively long life span (two weeks). It is not secure to pass the refresh token to the frontend application, because it is prone to various methods of hijacking. Thus, upon user's successful login a short lived access token is passed to the client in form of a HTTP-only cookie. Each request includes this token and user can be authenticated. When the access token expires, it is possible to issue a new access token by passing the expired one. As both refresh and access tokens are linked to each other it is easy to detect if the access has been stolen from the client and flush the leaked session. In this case two users – the original user and the attacker eventually will have 2 different access tokens pointing to the same refresh token.

With each request, the user is authenticated by the access token. The backend then determines, if the user is allowed to access given resource. This is achieved via classes called policies. Each controller has a defined policy, where each method is given a set of conditions for which the user is allowed to use it. For example the `UserPolicy` class has the access determined by the user's role:

```ruby
class UserPolicy < ApplicationPolicy
  def index?
    user.has_role? :admin
  end
  ...
end
```

This way the application is secure and roles allow us to have different privileges for different users.



User
Client app
API
1. Enter username and password
2. Sign in request
3. Generate token
4. Returns the JWT
5. Gets access to secure area
6. Keeps navigating
7. Send JWT token on every request

Fig. 3.7: JWT authentication diagram[13]

Storage

Many models in the application can store files or images. For example each product can have multiple images. We are thus presented with a challenge on how to store these files both securely and effectively, so they are quickly available.

It was decided to create a separate server for static content and store files on a S3 compatible object storage. This provides many benefits both to the programmer and end user:

- **CDN:** The static content server is using a Cloudflare CDN[7]. This free service delivers a content to the end user from a geographically closest possible location via its global infrastructure, thus rendering the loading times very fast.
- **File caching:** Images are cached both on the server and in the client's browser. This greatly speeds up the loading times.
- **Expiring URLs:** File URLs can be setup with an expiration time. This is useful for security reasons. For example when the user requests a receipt of goods (PDF attachment), it is set to expire after only few minutes. This way we can prevent unwanted sharing of the files – after the expiration, the URL returns a not found response.
- **Rails integration:** The S3 compatible storage is well integrated with the used Ruby on Rails framework. It is therefore trivial for the programmer to setup file uploading, give models a file association etc.

We also compress the uploaded images and deliver them in an appropriate sizes (thumbnails, icons, etc.). This further speeds up their loading and decreases server load.

Automatized testing

The use of automatized testing is crucial for a high quality application. It largely minimizes the number of bugs and makes the application more stable and trustworthy.

We use RSpec[5] library for the so called behavior testing. Every controller behavior is described in the set of automated tests. We can thus be sure that the implemented functionality works as intended. The RSpec uses a simple syntax with keywords `describe` and `it`, which naturally allows us to express concepts like a conversation.

The code is indeed very readable. Below we can see an example from the `UsersController` tests:

```ruby
RSpec.describe UsersController, type: :controller do
  context 'for unauthorized users' do
    let(:user) { create :user }

    before(:each) do
      sign_in_as user
    end

    describe '#index' do
      send(:index, :get)

      expect(response.status).to eq 403
    end
  ...
  end
  ...
end
```

In this example, a regular user (without administrator privileges) is created and signed in before every `describe` statement. We then describe (test) the `#index` method – we send the request and expect to receive a HTTP response with code 403 (Unauthorized). This is because regular user should not have an access to the listing of all of the application's users.

This way all the application's backend functionality is tested, including creation and editing of resources, authorization and authentication etc.

Automated tests run before every deployment to the production server (will be described in the Section 3.6.3) and this allows for a smooth, almost bug free experience.

3.6.2 Frontend

Frontend part of the application is discussed in this section. It is the part that users see and interact with. The application should therefore be easy to use, responsive (optimized for the smartphone devices) and provide all the requried functionality. We will present the implementation solutions which helps to achieve these goals in this section.

Vue.js

The Vue.js framework was used for the implementation of the frontend. It is a JavaScript framework for building user interfaces. It builds on top of standard HTML, CSS and JavaScript, and provides a declarative and component-based programming model[54].

The declarative approach differs from the traditional imperative JavaScript programming. It is not needed to explicitly write the code step-by-step, instead we write the code by declaring what we want to achieve. This greatly simplifies the writing of complex applications and keeps the codebase and the programmer sane.

Let's explain use of declarative programming in Vue.js on the example of a button, which when it is clicked increments a counter.

In the imperative style we would have to define the event listeners, write the increment function and directly access the DOM objects:

```
<form>
  <input type="text" id="number" value="0"/>
  <button onclick="incrementValue()" value="Click me" />
</form>

<script>
function incrementValue() {
    let value = document.getElementById('number').value
    value++
    document.getElementById('number').value = value
}
</script>
```

The same functionality can be achieved in the declarative style of the Vue.js framework by this statement:

```
import { createApp } from 'vue'

createApp({
  data() {
    return {
      count: 0
    }
  }
}).mount('#app')

// Vue template
<div id="app">
  <input type="text" :value="count">
  <button @click="count++"/>Click me</button>
</div>
```

Vue extends standard HTML with a template syntax that allows us to declaratively describe HTML output based on JavaScript state. It also automatically tracks JavaScript state changes and efficiently updates the DOM when changes happen[54].

Each part of the website is modeled as a self-contained component within its own file. This way it is easy to reason about the code and separate the individual concerns of the application.

Tailwind CSS

For the application styles, it was decided to use the Tailwind CSS framework[50]. It differs from the classic approach of writing CSS rules into the `.css` files. Instead, we write a predefined classes, which represents individual CSS rules, directly to each of the HTML elements.

For example if we would like to make the color of some text red, traditionally, we would write a CSS rule:

```
<span>RED TEXT</span>

<style>
span {
  color: red;
}
</style>
```

Using the Tailwind CSS we can achieve the same result with:

```
<span class="text-red-500">RED TEXT</span>
```

This nicely complements the Vue.js style of writing templates for individual components – even CSS styles are self-contained.

Tailwind is designed with the responsiveness and mobile first approach in mind, which helped us to create a good looking application for both mobile and desktop (see the attached screenshots A.13).

Localization

It is important to mention that the implemented application is localized in two languages – English and Slovak. The individual translations are defined in JSON files and the application language can be changed from the menu (as seen in the screenshot A.2).

Application structure

The application structure consists of two main building blocks – `views` and `components`. Views represent the individual pages of the application (Homepage, Stock transactions dashboard,...). Components fill up the content of the view – they are for example CRUD table (described in detail in Section 3.6.2), various inputs and buttons, etc.

The main views are:

- **Login:** The user has to log in to the application before doing any work. The Login view is presented to him when he first visits the information system's website (see the attached screenshot A.2).
- **Homepage:** This is the main dashboard in the application. User is redirected here when he logs in. It provides the managerial outputs such as stock transactions, insufficient goods, etc.(see screenshot A.3).
- **Stocks dashboard:** This is probably the most important screen in the application. User can see the list of the goods available in the clinic, the room it is stored in, number of available pieces and its expiration date. It is also possible to search and filter the goods by various parameters – product, warehouse, room and expiration date range (see screenshot A.4). User can scan the product's barcode (see screenshot 3.10) and it automatically filters the scanned product stock. He can then stock out or transfer the selected product as seen on the screenshot A.6. Administrator can also manually add products to the inventory.
- **Stock transactions:** Administrator can view all the movements of inventory items in this view (see screenshot A.7). He can also filter it by event (stock in, stock out, stock transfer), user, product and many other parameters. This way he has a good overview of the stock movement.

- **Invoices:** This view is used for stock-in of the goods (see A.5). User can view, create, edit and remove individual invoices. Each invoice has many invoice items and other parameters as described in the Section 3.6.1.
- **Product dashboard:** In this view the products can be listed, created, edited and removed (A.10). When creating the product, the barcode can be scanned. User can also define multiple suppliers of the product. There are many managerial outputs in this dashboard – user can view the price history, available stock, and stock transactions for a given product (see attachment A.8).
- **Supplier dashboard:** The suppliers and their contact can be viewed, created, edited and removed in this dashboard (see A.9). User can also view all the suppliers's products there.
- **User dashboard:** Administrator can manage the user accounts in this dashboard. He can also activate or deactivate individual users there (see the screenshot A.11).
- **Audit dashboard:** This dashboard provides the administrator with a complete audit of the information system events and changes. He can filter logs by an Object (product, user, supplier,...), Event (creation, editing, removal) and User who executed the change (see the screenshot A.12).

There are some other views we did not mention in detail – for example Contacts view, Warehouse view, etc. Their purpose is self explanatory, the user can view, create, edit and remove individual items of a given model there.

In the next section we will present the most important components which were implemented.

CRUD Table

This is the single most important component in the whole application. CRUD stands for Create, Read, Update and Delete operations. This component provides the Read and Delete operations by default, but can be extended to handle other mentioned operations as well. It displays the data in form of a sortable, searchable and filterable table. It has built in pagination, so user can view large datasets. It is also easy to make the table's content dynamic – e.g. include hyperlinks for values, create custom actions, etc. It is reusable to the point where if the programmer wants to implement a new dashboard for a model, he simply sets attributes of this component and the view is ready to use.

We will present the most important CRUD Table component parameters:

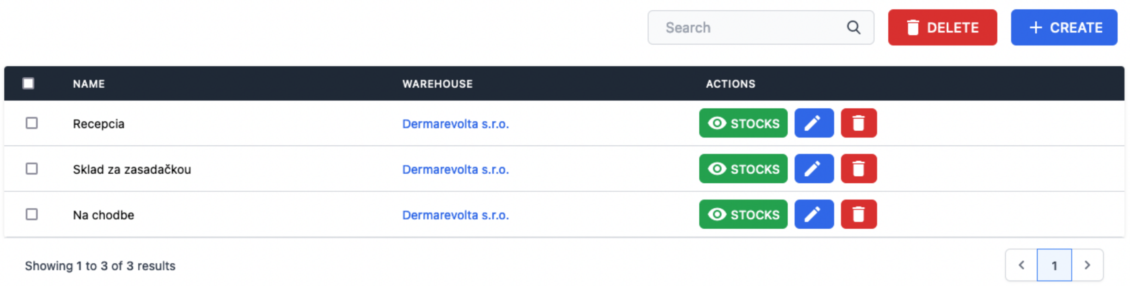
- `getEndpoint`: This is a required parameter which sets the API endpoint for getting the model's records from the database. All the endpoints follow similar logic, so it is enough to pass its URL and the component handles everything from fetching the records to pagination.

- `deleteEndpoint`: Same as for the `getEndpoint`, this parameter sets the URL of the delete endpoint for a given record.
- `customCols`: By default, the CRUD Table component displays all the columns from the API endpoint in form of a text value. This is sufficient for simple uses, but we want some more advanced features for the rich dashboards – mainly hyperlinks, images, etc. This parameter takes an array of components, which transform the column's value.
- `customActions`: The default actions for the CRUD Table are: `Create`, `Edit` and `Remove`. We can specify more actions using this parameter. For example, in the `UsersView`, we use `UserActivation` component for each one of the rows. It allows us to activate or deactivate the user.

The following example shows the usage of the CRUD Table component in the `RoomsView`:

```
<template>
  <crud-table
    :get-endpoint="getEndpoint"
    :delete-endpoint="deleteEndpoint"
    :custom-cols-after="customCols"
    ...
  ></crud-table>
</template>
...
data() {
  return {
    getEndpoint: getRooms,
    deleteEndpoint: deleteRoom,
    customCols: [
      {
        header: this.$t('warehouse'),
        component: shallowRef(CrudLink),
        options: {
          relationship: 'warehouse',
          attribute: 'name',
          editLink: true,
          editRouteName: 'WarehouseEdit',
          sort: true,
          orderBy: 'warehouses.name'
        }
      }
    ]
    ...
  }
}
...
```

We defined the `getEndpoint` and `deleteEndpoint` parameters. For the `customCols` parameter, we included the `CrudLink` component, which transforms the `warehouse` relationship column as a hyperlink to the `WarehouseEdit` route. The resluting table is shown in the Figure 3.8.



	NAME	WAREHOUSE	ACTIONS
☐	Recepcia	Dermarevolta s.r.o.	STOCKS
☐	Sklad za zasadačkou	Dermarevolta s.r.o.	STOCKS
☐	Na chodbe	Dermarevolta s.r.o.	STOCKS

Showing 1 to 3 of 3 results

Fig. 3.8: CRUD Table component

Barcode Scanner

The Barcode Scanner component fulfills the system requirements (Section 3.1) by providing an easy interface to scan the product's barcode. It uses 3rd party library QuaggaJS[34], which supports real-time localization and decoding of various types of barcodes such as EAN, CODE 128, etc. Its implementation features a barcode locator which is capable of finding a barcode-like pattern in an image resulting in an estimated bounding box including the rotation. This reader is invariant to scale and rotation, whereas other libraries require the barcode to be aligned with the viewport[34]. It can be used with a computer's webcam, or a smartphone.

A Vue.js wrapper for the library was implemented. It is very simple to use as shown in the example below:

```
...
<barcode-scanner @input="searchProduct" />
...
```

This code renders the scanner button (Figure 3.9), which when clicked opens the scanner (Figure 3.10).

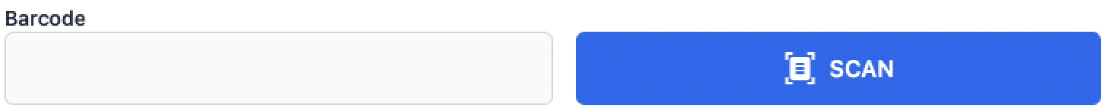


Fig. 3.9: Barcode Scanner button

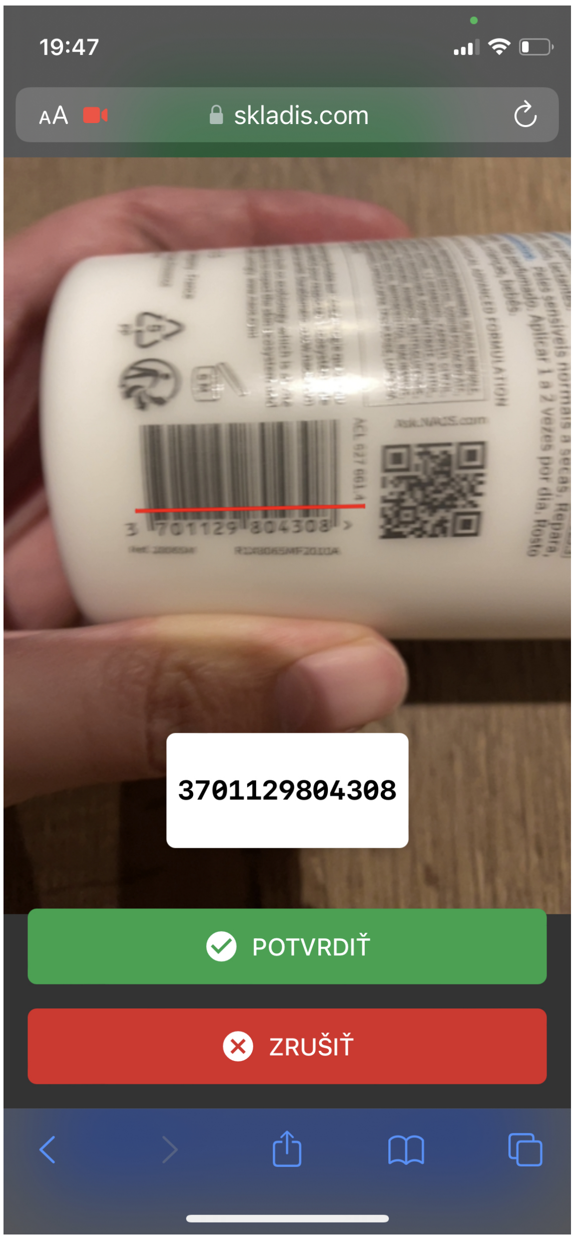


Fig. 3.10: Barcode Scanner

When the scan is successful, it returns the scanned barcode – its type and value.

Programmer can then write a custom function, which further processes the barcode. For example it can search for a product:

```
...
async searchProduct(barcode) {
  if (barcode) {
    const code = barcode.code
    const product = await getProducts({ filters: { '
   barcode_barcode_code': code } })
  }
}
...
```

3.6.3 Deployment

It is necessary to setup a server deployment for the implemented information system. We will describe a selection of the hosting provider in this section, along with the containerization of the application and implementation of continuous integration and delivery (CI/CD) pipeline.

Selection of the hosting provider

Multiple hosting providers were compared for the application hosting. A server is needed for the operation of database, backend and frontend parts of the information system. Additionally, S3 compatibile storage solution is necessary for hosting of the application's files and images.

Following criterions were chosen for the comparison of hosting solutions:

- Price per month (in €)
- CPU configuration
- RAM capacity
- Internal storage size and speed
- Internet connection speed

We've chosen four offerings from different providers in the same category for the comparison (see Table 3.13).

The first candidate – VPS (Virtual Private Server) from the company Hetzner seems to be the best choice for our needs. It is configured with the newest AMD Ryzen processor, has a DDR4 RAM and fast NVMe SSD storage. Internet speed is sufficient and price is also fair. Big positive of this offering is proximity of the server location – it is based in Austria, just a few hundred kilometers from the clinic. The hosting offering comes without commitment – you can cancel it anytime.

OVHcloud is a canadian hosting provider, but offers server location in Germany, too. It has slower internet connection and slower (non NVMe) storage compared to the first candidate. The price per month is the highest among the offerings, so we decided to pass on it.

The third provider – Forpsi is based in the Czech Republic. Its offering doesn't beat the Hetzner's – only 2 CPU cores and the slowest internet connection disqualify it from the selection.

The last provider – Wedos is also a Czech company. It has comparable parameters, but a higer monthly price.

It was therefore decided to go with the Hetzner's offering.

	Price	**CPU**	**RAM**	**Storage size**	**Internet speed**
Hetzner Online GmbH. – CPX31[18]	14.76	4 vCPU	8 GB	160 GB NVMe SSD	1 Gbit/s
OVHcloud – Essential VPS[36]	20.82	4 vCPU	8 GB	160 GB SSD	500 Mbit/s
Forpsi – Large VPS[19]	14.92	2 vCPU	4 GB	80 GB SSD	100 Mbit/s
Wedos – VPS SSD[55]	15.80	4 vCPU	8 GB	60 GB SSD	1 Gbit/s

Tab. 3.13: Server hosting comparison

As for the S3 storage solution, it was decided to use the industry standard – Amazon S3. The pricing is very compelling, because you pay for what you use. For the first 50TB/month of storage it means only $0.0245 per GB. Additional fees apply to data transfers and API requests – first 10 TB/Month of egress traffic costs $0.09 per GB and API costs are negligible for the current amount of traffic[3]. In the time of writing, stored images and files take up only 0.5 GB of storage. In conjunction with the efficient server proxy caching it means that the storage costs are very low, almost negligible.

SSL Encryption

It is important to maintain a secure connection while using the application. A secure SSL certificate from the Let's Encrypt is used for this purpose. Let's Encrypt is a free, automated, and open certificate authority[20]. The certificate is automatically renewed when its expiration comes near, so the information system's connection is always secure. For more information about SSL encryption, please refer to the Chapter 1.

Containerization of the application

Containerization is an application level virtualization. It helps us to isolate software in spaces called containers. They can be run in virtually any environment and are defined by a code, much like the application logic itself. Many benefits come from this technique:

- The infrastructure of the application is defined in the code and thus can be easily modified and reasoned about.
- The isolation helps the security of the application. The container runs only the software needed for the application's operation, which results in fewer possible attack vectors.
- The same container can be run virtually on any system (production server, developer's machine, etc.) without any change. This is a big advantage, because the behaviour of the code is the same throughout the systems.
- It integrates very well with the CI/CD pipeline (see Section 3.6.3).

The Docker[12] platform was chosen for the containerization of the implemented application. We can define the containers behaviour in files called Dockerfiles. For example the Dockerfile for the information system's frontend part is shown below:

```dockerfile
# build stage
FROM docker.io/node:16-alpine as build-stage
WORKDIR /app
COPY package*.json ./
RUN yarn install
COPY . .
RUN yarn build

# production stage
FROM docker.io/nginx:stable-alpine as production-stage
COPY --from=build-stage /app/dist /usr/share/nginx/html
COPY ./nginx/nginx.conf /etc/nginx/nginx.conf
COPY ./nginx/default.conf /etc/nginx/conf.d/default.conf
EXPOSE 8080
CMD ["nginx", "-g", "daemon off;"]
```

It first selects the base image, which is in this case `node` – JavaScript runtime. This defines the base environment in which the container runs. Then it copies the application source code to the container and installs all the required 3rd party packages. In the production stage the server is setup and port `8080` is exposed. The last command `CMD` defines the code, which will run when the container starts. In this case it starts the `Nginx` server.

With these few steps we have a working application environment. We can then build the container using our CI/CD pipeline (see Section 3.6.3) and run it on virtually any server. The configuration is defined in the code and we can expect that the application will behave the same regardless of the used server.

CI/CD pipeline

It is industry best practice to implement an automatized deployment of applications. This way a developer doesn't need to manually upload the source code to a production server. In addition, various integrations can be included in a deployment process, e.g running automatized tests. This automatized series of steps to deliver a new version of an application is called a CI/CD pipeline.

The source code of the implemented information system is versioned via GitHub, which provides a CI/CD platform via GitHub Actions[14]. When the code is committed to the master branch, a series of steps is executed:

- Run automatized integration tests, which check wether the application runs correctly.
- If the automatized test were successfull, build and deploy the new version to the production server.

GitHub Actions allow us to easily define these steps via YML files. For example the following code will automatically run the deploy script on the remote server to change the application version to the newest after pushing code to the master branch:

```yaml
name: skladis_ci_cd
on:
  push:
    branches: [master]

jobs:
  ...
  deploy:
    name: Deployment of the SkladIS Frontend to production
    runs-on: ubuntu-latest
    steps:
      - name: Configure SSH
        run: ...
        env: ...
      - name: Execute deployment script
        run: ssh deployment 'bash --login -c "deploy-skladis-frontend"'
```

Data backup

The database of the information system is backed up to a S3 storage every 12 hours. This guarantees that in event of a server malfunction we can rollback the latest backup easily.

It is implemented using a simple Bash script which executes periodically using CRON job. The database dump is then uploaded using AWS-CLI utility:

```
# CRON job
# m h  dom mon dow   command
0 */12 * * * /home/skladis/docker/deploy/backup.sh

# backup.sh script
#!/bin/bash
BACKUP_FILE_NAME=dump_$(date +"%Y-%m-%d_%H_%M_%S").gz
podman exec -t skladis-db pg_dumpall -c -U postgres | gzip > "/home
    /skladis/db_backup/$BACKUP_FILE_NAME"
aws s3 cp "/home/skladis/db_backup/$BACKUP_FILE_NAME" "s3://skladis
    -backup/$BACKUP_FILE_NAME"
```

3.7 Audit of the implemented application

An audit of the implemented information system was performed using the ZEFIS[11] tool. Several questionnaires were filled and the audit tool identified a few shortcomings. We will describe the audit results in this section and will suggest measures to improve the identified shortcomings.

3.7.1 Shortcomings of the implemented information system

Results of the implemented information system are positive. A few identified shortcomings are relatively easy to resolve and they are not critical for the correct operation of the system. In the Table 3.14 the identified problems are presented along with the suggested measures.

No.	Problem description	Suggested solution
1	The GDPR rules for the client's data management are not implemented.	We can ignore this problem, because the implemented system does not store any client's personal data. Nevertheless, it would be wise to consult this with a lawyer.
2	The user's passwords are not periodically changed.	The system should periodically enforce a password change and the password validation should require the password length of at least 8 characters including numbers and special characters.
3	There are no SLA agreements closed.	The system is developed internally, therefore it is not required to make any SLA agreements. The hosting provider guarantees their service in its contract.
4	The accountability for the data is not defined in a written form.	Management of the clinic should prepare a document for the accountability for the data in a written form.

Tab. 3.14: Shortcoming identified in the ZEFIS audit and suggested measures

3.8 Contribution of implemented solutions

The implemented inventory information system meets the requirements defined in Section 3.1. It is a big improvement over the old system of Excel tables. The system has been successfully deployed and employees started using it. They have no problem with its operation so far. It brings many benefits both to the company and its employees:

- **User experience**: The user experience of the new system is vastly better compared to the old system. The application can be used from user's smartphone, tablet or PC. Its data model is interconnected and user can quickly see related resources – e.g. when viewing suppliers, user can view supplier's products with one click, same with the clinic's rooms and its stock, etc. The barcode scanner simplifies the process of finding a given item. All information is presented in a clearer way.
- **Data structure and integrity**: The data structure and integrity is secured by the used database, system's constraints and validation checks. The old system didn't have these features, which resulted in several discrepancies compared to the real state of the inventory. The new system greatly helps the company to keep an order in its inventory.
- **Managerial outputs**: The new system offers managerial outputs which the old system completely lacked. Managers can see the insufficient goods, goods with ending expiration, price history for different suppliers, etc. These outputs helps them make a better decisions.
- **Security**: The new system has different user roles, so employees can't see everything. The system is password protected and built with a high security standard in mind. The data is regularly backed up. All of these changes help to keep the system and company's data safe and secure.

3.9 Further improvements

In this section we will try to suggest possible improvements of the implemented information system. We identified two processes which could be simplified and automatized to a certain degree.

3.9.1 Integration with the accounting information system

As described in Section 3.1.1, the ordered goods have to be manually added to the inventory management system. This process could be automatized by integration with the accounting information system:

- After creating the invoice in the accounting information system, an unconfirmed receipt of goods would be created in the inventory management system. It would include all the ordered goods from the invoice.
- When the goods arrive, the responsible employee would only have to enter the expiration date and the room where the goods are to be placed in. He wouldn't have to go trough the process of finding the individual items by scanning its barcode.

Benefits

- Saving the employee's time.
- Better data integrity between the accounting information system and inventory management information system, less errors.

Implementation

The accounting information system offers an API, which allows us to download the invoice data. Unfortunately, it does not support a webhook integration, where we would subscribe to the invoice creation event and the accounting information system would automatically send a HTTP request to the inventory information system, which would then create a receipt of goods using the request's data.

The suggested solution to this problem is to periodically, e.g. once a day, send a request to the accounting system's API and if a new invoice is detected, create a receipt of goods. This solution is not as elegant as webhook method, but it solves the problem without the need of interacting with the accounting system's developer. A diagram of this method can be seen on Figure 3.11.

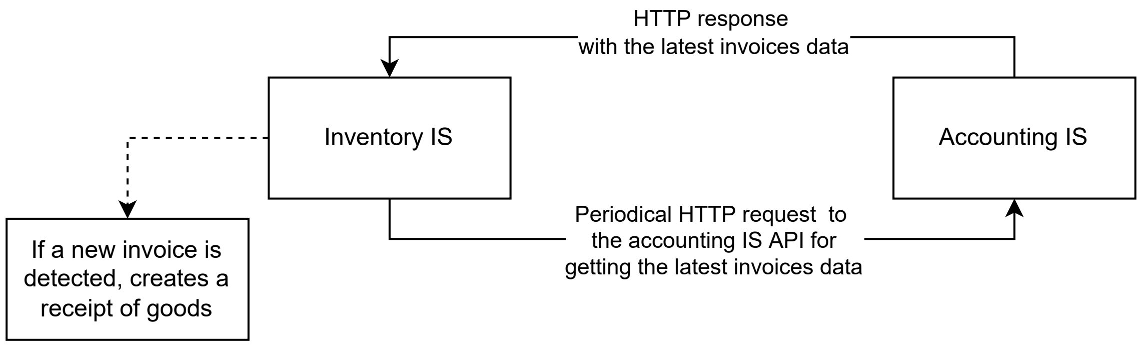

HTTP response
with the latest invoices data
Inventory IS
Accounting IS
If a new invoice is detected, creates a receipt of goods
Periodical HTTP request to the accounting IS API for getting the latest invoices data

Fig. 3.11: Automatization of importing the ordered goods to the inventory IS

Cost estimation

We can estimate the cost of this change thanks to the inventory information system being implemented internally. API of the accounting information system is well documented, which makes the implementation less difficult. We estimate the duration of the implementation to be around 100-120 man hours, which represents cost of 4000-4800 EUR at the programmer's hourly rate of 40 EUR.

3.9.2 Order process automatization

The next improvement is an automatization of the order process. Right now, the goods are ordered using different methods – e-shop, email, phone and from different suppliers. It is not feasible to fully automatize the order process, as every supplier uses a different order method and most of them do not offer an API.

The suggested automatization lies in the generation of the email order templates, which would contain all the necessary information about ordered products and their quantity. This comes useful, because many supplier communicate only via an email.

Furthermore, the information system could identify the best supplier based on the historical data, which are already present in the system's database.

Benefits

- Saving the employee's time.
- Cost optimization – choosing the cheapest supplier for a given product using the historical price data.

Implementation

The sending of scheduled emails is already implemented in the information system's framework. The implementation would therefore mean to implement the content of these emails and its sending to the selected employees.

The content of the emails is relatively easy to implement. We have to find products which have a critical quantity of its stocks and find the cheapest supplier for these products using the historical data. These parameters can be easily programmed using the ORM layer of the used framework.

Cost estimation

Thanks to the inventory information system being implemented internally, the cost of the change would be relatively low. Rough estimate is from 50-70 man hours, which represents the cost of 2000 - 2800 EUR, if we consider the hourly rate of the programmer to be 40 EUR.

Conclusion

The problem of the inventory management of the dermatologic clinic was analyzed and a change in form of a new information system was designed and implemented.

The new inventory management information system is now deployed and fully operational. The system meets all of its requirements. It greatly simplifies the inventory management of the clinic. The inventory data has better integrity, which will result in less errors. User experience is incomparable with the old system – it allows advanced filtering of the data, barcode scanning using a smartphone camera, image attachments for products and many other features. It also includes various managerial outputs, which will help the company's management to make a better managerial and strategic decisions. User accounts and user roles were implemented, which make the system far more secure, compared to the old system of Excel files, which lowers the risk of data loss or breach.

Several further improvements were proposed, including integration with the accounting information system and automatization of the order process. These advanced features would not be possible to implement using the old system. The new system is extensible and implementation of further improvements can be easily achieved.

We can say that all of the implemented changes help the company to achieve better results, save the employees time, lower risks and errors in the inventory management process and generally move the company in a good direction regarding digitalization and informatization of its processes.

Bibliography

[1] Akshay Khot. *How Rails Concerns Work and How to Use Them*. 2021. URL: https://dev.to/software_writer/how-rails-concerns-work-and-how-to-use-them-gi6 (visited on 04/03/2022).

[2] A Ais Alimuddin et al. "Design and Implementation of REST API for Academic Information System". In: *IOP Conference Series: Materials Science and Engineering* 875 (July 2020), p. 012047. DOI: 10.1088/1757-899X/875/1/012047.

[3] Amazon Web Services, Inc. *Amazon S3*. 2022. URL: https://aws.amazon.com/s3/ (visited on 04/12/2022).

[4] Asociácia súkromných lekárov Slovenskej republiky. *Nedostatok zdravotníckeho personálu na Slovensku*. 2022. URL: http://new.aslsr.sk/aktualita/?articleId=nedostatok-zdravotnickeho-personalu-na-slovensku--velmi-vazna-situacia-je-u-tzv--strazcov-dveri (visited on 04/01/2022).

[5] Steven Baker. *RSpec*. 2022. URL: https://rspec.info/ (visited on 04/03/2022).

[6] Michel Bakni. *Client-Server 3-tier architecture*. 2022. URL: https://commons.wikimedia.org/wiki/File:Client-Server_3-tier_architecture_-_en.png (visited on 04/15/2022).

[7] Cloudflare, Inc. *Cloudflare*. 2022. URL: https://www.cloudflare.com/ (visited on 04/03/2022).

[8] Cloudflare, Inc. *What is an SSL certificate?* 2022. URL: https://www.cloudflare.com/learning/ssl/what-is-an-ssl-certificate/ (visited on 04/15/2022).

[9] David Heinemeier Hansson. *Active Record Query Interface*. 2022. URL: https://guides.rubyonrails.org/active_record_querying.html#scopes (visited on 04/03/2022).

[10] David Heinemeier Hansson. *Ruby on Rails*. 2022. URL: https://rubyonrails.org/ (visited on 04/03/2022).

[11] Doc.Ing.Miloš Koch,CSc. *ZEFIS - audit informačních systémů*. 2022. URL: https://www.zefis.cz/ (visited on 04/03/2022).

[12] Docker Inc. *Docker overview*. 2022. URL: https://docs.docker.com/get-started/overview/ (visited on 04/12/2022).

[13] Fernando Doglio. *JWT Authentication Best Practices*. 2020. URL: https://blog.openreplay.com/jwt-authentication-best-practices (visited on 04/03/2022).

[14] GitHub, Inc. *Github Actions.* 2022. URL: https://docs.github.com/en/actions/automating-builds-and-tests/about-continuous-integration (visited on 04/12/2022).

[15] Graphical Research. *Europe Aesthetic Medicine Market Size.* 2021. URL: https://www.graphicalresearch.com/industry-insights/1812/europe-aesthetic-medicine-market (visited on 04/01/2022).

[16] Sodomka Petr, Klčová Hana. *Informační systémy v podnikové praxi. 2. vydání.* Brno: Computer Press, 2000. ISBN: 978-80-251-2878-7.

[17] Elizabeth Hardcastle. *Business Information Systems.* Ventus Publishing ApS, 2008. ISBN: 978-87-7681-463-2.

[18] Hetzner Online GmbH. *Hetzner Cloud.* 2022. URL: https://www.hetzner.com/cloud (visited on 04/12/2022).

[19] INTERNET CZ, a. s. *Classic VPS.* 2022. URL: https://www.forpsi.com/virtual/ (visited on 04/12/2022).

[20] Internet Security Research Group. *Let's Encrypt.* 2022. URL: https://letsencrypt.org/ (visited on 04/12/2022).

[21] Zuzana Janková. "Časová a síťová analýza. Metoda PERT". lecture. 2022.

[22] Jeremykemp. *Pert chart colored.* 2005. URL: https://commons.wikimedia.org/wiki/File:Pert_chart_colored.svg (visited on 04/15/2022).

[23] JSON.org. *Introducing JSON.* 2022. URL: https://www.json.org/json-en.html (visited on 04/15/2022).

[24] Medha Kalelkar, Prathamesh Churi, and Deepa Kalelkar. "Implementation of Model-View-Controller Architecture Pattern for Business Intelligence Architecture". In: *International Journal of Computer Applications* 102 (Sept. 2014), pp. 16–21. DOI: 10.5120/17867-8786.

[25] Smejkal Vladimír, Rais Karel. *Řízení rizik ve firmách a jiných organizacích. 4., aktualiz. a rozš. vyd.* Praha: Grada, 2013. ISBN: 978-80-247-4644-9.

[26] KirstenS. *Cross Site Scripting (XSS).* 2022. URL: https://owasp.org/www-community/attacks/xss/ (visited on 04/15/2022).

[27] Saša Baškarada, Andy Koronios. "Data, Information, Knowledge, Wisdom (DIKW): A Semiotic Theoretical and Empirical Exploration of the Hierarchy and its Quality Dimension". In: *Australasian Journal of Information Systems* 18.1 (2013). URL: https://journal.acs.org.au/index.php/ajis/article/view/748/550 (visited on 03/29/2022).

[28] Santosh Kumar. "A REVIEW ON CLIENT-SERVER BASED APPLICATIONS AND RESEARCH OPPORTUNITY". In: *International Journal of Scientific Research* 10 (Aug. 2019), pp. 33857–33862. DOI: 10.24327/ijrsr.2019.1007.3768.

[29] Mozilla Foundation. *An overview of HTTP*. 2022. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview (visited on 04/15/2022).

[30] Mozilla Foundation. *HTML basics*. 2022. URL: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics (visited on 04/15/2022).

[31] Mozilla Foundation. *SPA (Single-page application)*. 2022. URL: https://developer.mozilla.org/en-US/docs/Glossary/SPA (visited on 04/15/2022).

[32] Mozilla Foundation. *What is CSS?* 2022. URL: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS (visited on 04/15/2022).

[33] Mozilla Foundation. *What is JavaScript?* 2022. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript (visited on 04/15/2022).

[34] Christoph Oberhofer. *quaggaJS*. 2022. URL: https://github.com/serratus/quaggaJS (visited on 04/12/2022).

[35] Orcale. *What Is a Database?* 2022. URL: https://www.oracle.com/database/what-is-database/ (visited on 04/15/2022).

[36] OVH US LLC. *OVH Cloud*. 2022. URL: https://us.ovhcloud.com/vps/compare/ (visited on 04/12/2022).

[37] Hazael Phiri and Douglas Kunda. "A Comparative Study of NoSQL and Relational Database". In: *Zambia ICT Journal* 1 (Dec. 2017), pp. 1–4. DOI: 10.33260/zictjournal.v1i1.8.

[38] Pkor43 at English Wikipedia. *McKinsey 7S framework*. 2007. URL: https://commons.wikimedia.org/wiki/File:McKinsey_7S_framework.svg (visited on 04/15/2022).

[39] Positive Stud. *A Brief Overview of SQL (Structured Query Language)*. 2020. URL: https://medium.com/analytics-vidhya/a-brief-overview-of-sql-structured-query-language-4c20e4352726 (visited on 04/15/2022).

[40] Steve Prettyman. *Learn PHP 7 : object oriented modular programming using HTML5, CSS3, Javascript, XML, JSON, and MySQL*. Berkeley, CA New York, NY: Apress, 2015. ISBN: 978-1-4842-1730-6.

[41] Bhanu Priya. *Explain BCNF with an example in DBMS*. 2021. URL: https://www.tutorialspoint.com/explain-bcnf-with-an-example-in-dbms (visited on 04/15/2022).

[42] Ritesh Ranjan. *What is a Framework in Programming and Why You Should Use One*. 2022. URL: https://www.netsolutions.com/insights/what-is-a-framework-in-programming/ (visited on 04/15/2022).

[43] Abdelmonaim Remani. *REST*. 2022. URL: https://cs.lmu.edu/~ray/notes/rest/ (visited on 04/15/2022).

[44] Urmila Samariya. *3-Way Handshake Connection Establishment Process*. 2021. URL: https://www.tutorialspoint.com/tcp-3-way-handshake-process (visited on 04/15/2022).

[45] Tanya Sammut-Bonnici and David Galea. "PEST analysis". In: Jan. 2015. ISBN: 9781118785317. DOI: 10.1002/9781118785317.weom120113.

[46] Ministerstvo spravodlivosti SR. *Obchodný register*. 2021. URL: https://www.orsr.sk/ (visited on 12/19/2021).

[47] Jitka Srpová. *Podnikatelský plán a strategie*. Praha: Grada, 2011. ISBN: 978-80-247-4103-1.

[48] Statistical Office of the SR. *Inflation — consumer price indices in March 2022*. 2022. URL: https://slovak.statistics.sk/wps/portal/ext/products/informationmessages/inf_sprava_detail/ (visited on 04/01/2022).

[49] Alena Svozilová. *Projektový management: systémový přístup k řízení projektů. 3., aktualizované a rozšířené vydání*. Praha: Grada, 2016. ISBN: 978-80-271-0075-0.

[50] Tailwind Labs Inc. *Get started with Tailwind CSS*. 2022. URL: https://tailwindcss.com (visited on 04/12/2022).

[51] The PostgreSQL Global Development Group. *PostgreSQL*. 2022. URL: https://www.postgresql.org/ (visited on 04/03/2022).

[52] Gareth Tyson. *Overview of Protocol Exchange for HTTP Delivery*. 2022. URL: https://www.researchgate.net/figure/Overview-of-Protocol-Exchange-for-HTTP-Delivery_fig10_225082716 (visited on 04/15/2022).

[53] Vidushi, Prof. S. Niranjan. "A Review on SQL Injection". In: *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH and TECHNOLOGY (IJERT)* 3.10 (2015). URL: https://www.ijert.org/a-review-on-sql-injection (visited on 04/15/2022).

[54] Vue.js. *Vue Introduction*. 2022. URL: https://vuejs.org/guide/introduction.html (visited on 04/12/2022).

[55] WEDOS Internet, a. s. *VPS SSD*. 2022. URL: `https://www.wedos.com/vps-ssd` (visited on 04/12/2022).

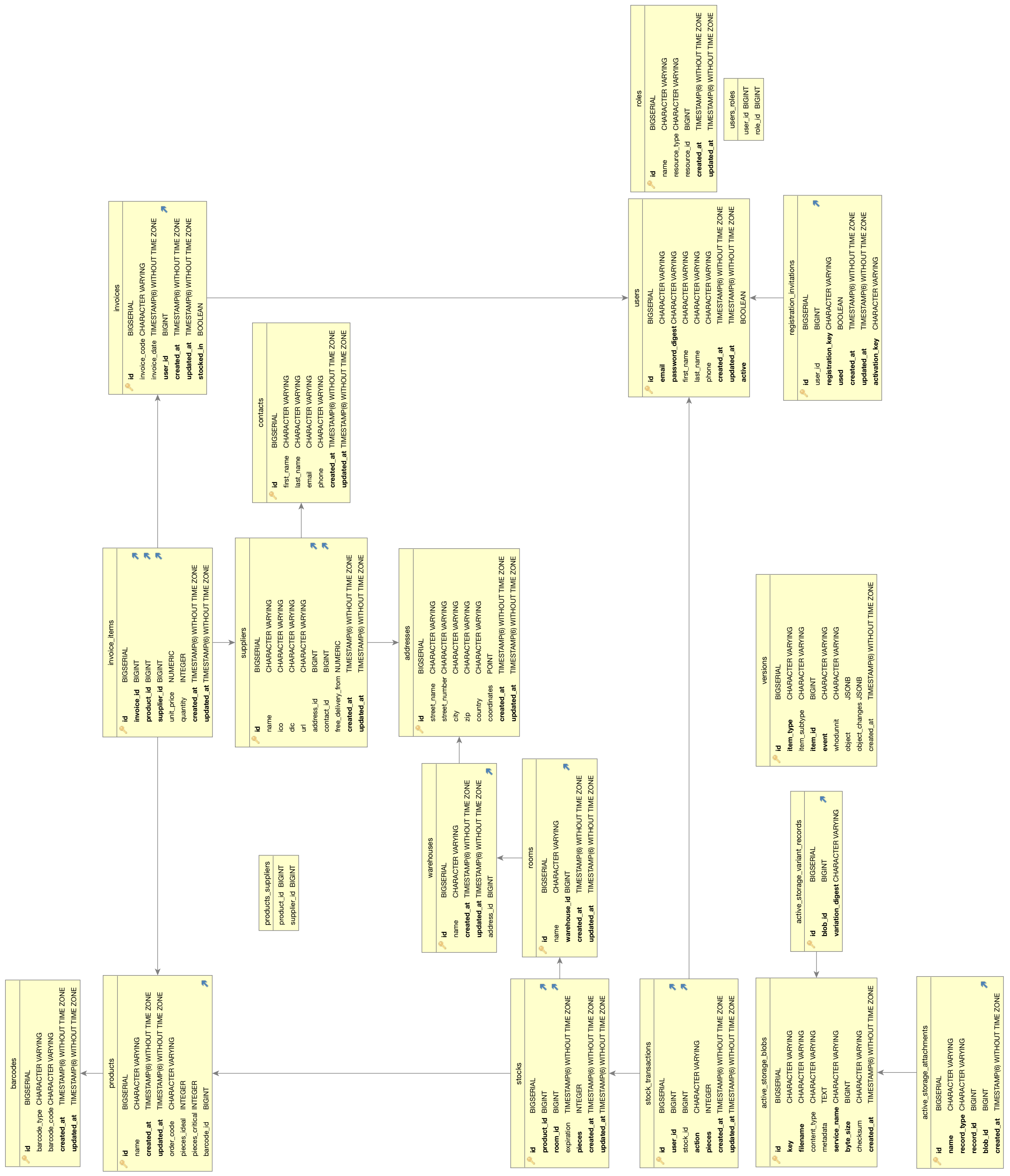
[56] Wikipedia. *McKinsey 7S Framework — Wikipedia, The Free Encyclopedia*. 2022. URL: `https://en.wikipedia.org/wiki/McKinsey_7S_Framework` (visited on 04/15/2022).

[57] Xhienne. *SWOT*. 2007. URL: `https://commons.wikimedia.org/wiki/File:SWOT_en.svg` (visited on 04/15/2022).

[58] Gála Libor, Pour Jan, Šedivá Zuzana. *Podniková informatika. 2., přepracované a aktualizované vydání*. Praha: Grada Publishing, 2009. ISBN: 978-80-247-2615-1.

List of Figures

List of Tables



roles
id BIGSERIAL
name CHARACTER VARYING
resource_type CHARACTER VARYING
resource_id BIGINT
created_at TIMESTAMP(6) WITHOUT TIME ZONE
updated_at TIMESTAMP(6) WITHOUT TIME ZONE
users_roles
user_id BIGINT
role_id BIGINT
invoices
id BIGSERIAL
invoice_code CHARACTER VARYING
invoice_date TIMESTAMP(6) WITHOUT TIME ZONE
user_id BIGINT
created_at TIMESTAMP(6) WITHOUT TIME ZONE
updated_at TIMESTAMP(6) WITHOUT TIME ZONE
stocked_in BOOLEAN
contacts
id BIGSERIAL
first_name CHARACTER VARYING
last_name CHARACTER VARYING
email CHARACTER VARYING
phone CHARACTER VARYING
created_at TIMESTAMP(6) WITHOUT TIME ZONE
updated_at TIMESTAMP(6) WITHOUT TIME ZONE
users
id BIGSERIAL
email CHARACTER VARYING
password_digest CHARACTER VARYING
first_name CHARACTER VARYING
last_name CHARACTER VARYING
phone CHARACTER VARYING
created_at TIMESTAMP(6) WITHOUT TIME ZONE
updated_at TIMESTAMP(6) WITHOUT TIME ZONE
active BOOLEAN
registration_invitations
id BIGSERIAL
user_id BIGINT
registration_key CHARACTER VARYING
used BOOLEAN
created_at TIMESTAMP(6) WITHOUT TIME ZONE
updated_at TIMESTAMP(6) WITHOUT TIME ZONE
activation_key CHARACTER VARYING
invoice_items
id BIGSERIAL
invoice_id BIGINT
product_id BIGINT
supplier_id BIGINT
unit_price NUMERIC
quantity INTEGER
created_at TIMESTAMP(6) WITHOUT TIME ZONE
updated_at TIMESTAMP(6) WITHOUT TIME ZONE
suppliers
id BIGSERIAL
name CHARACTER VARYING
ico CHARACTER VARYING
dic CHARACTER VARYING
url CHARACTER VARYING
address_id BIGINT
contact_id BIGINT
free_delivery_from NUMERIC
created_at TIMESTAMP(6) WITHOUT TIME ZONE
updated_at TIMESTAMP(6) WITHOUT TIME ZONE
addresses
id BIGSERIAL
street_name CHARACTER VARYING
street_number CHARACTER VARYING
city CHARACTER VARYING
zip CHARACTER VARYING
country CHARACTER VARYING
coordinates POINT
created_at TIMESTAMP(6) WITHOUT TIME ZONE
updated_at TIMESTAMP(6) WITHOUT TIME ZONE
versions
id BIGSERIAL
item_type CHARACTER VARYING
item_subtype CHARACTER VARYING
item_id BIGINT
event CHARACTER VARYING
whodunnit CHARACTER VARYING
object JSONB
object_changes JSONB
created_at TIMESTAMP(6) WITHOUT TIME ZONE
products_suppliers
product_id BIGINT
supplier_id BIGINT
warehouses
id BIGSERIAL
name CHARACTER VARYING
created_at TIMESTAMP(6) WITHOUT TIME ZONE
updated_at TIMESTAMP(6) WITHOUT TIME ZONE
address_id BIGINT
rooms
id BIGSERIAL
name CHARACTER VARYING
warehouse_id BIGINT
created_at TIMESTAMP(6) WITHOUT TIME ZONE
updated_at TIMESTAMP(6) WITHOUT TIME ZONE
active_storage_variant_records
id BIGSERIAL
blob_id BIGINT
variation_digest CHARACTER VARYING
barcodes
id BIGSERIAL
barcode_type CHARACTER VARYING
barcode_code CHARACTER VARYING
created_at TIMESTAMP(6) WITHOUT TIME ZONE
updated_at TIMESTAMP(6) WITHOUT TIME ZONE
products
id BIGSERIAL
name CHARACTER VARYING
created_at TIMESTAMP(6) WITHOUT TIME ZONE
updated_at TIMESTAMP(6) WITHOUT TIME ZONE
order_code CHARACTER VARYING
pieces_ideal INTEGER
pieces_critical INTEGER
barcode_id BIGINT
stocks
id BIGSERIAL
product_id BIGINT
room_id BIGINT
expiration TIMESTAMP(6) WITHOUT TIME ZONE
pieces INTEGER
created_at TIMESTAMP(6) WITHOUT TIME ZONE
updated_at TIMESTAMP(6) WITHOUT TIME ZONE
stock_transactions
id BIGSERIAL
user_id BIGINT
stock_id BIGINT
action CHARACTER VARYING
pieces INTEGER
created_at TIMESTAMP(6) WITHOUT TIME ZONE
updated_at TIMESTAMP(6) WITHOUT TIME ZONE
active_storage_blobs
id BIGSERIAL
key CHARACTER VARYING
filename CHARACTER VARYING
content_type CHARACTER VARYING
metadata TEXT
service_name CHARACTER VARYING
byte_size BIGINT
checksum CHARACTER VARYING
created_at TIMESTAMP(6) WITHOUT TIME ZONE
active_storage_attachments
id BIGSERIAL
name CHARACTER VARYING
record_type CHARACTER VARYING
record_id BIGINT
blob_id BIGINT
created_at TIMESTAMP(6) WITHOUT TIME ZONE

Fig. A.1: Entity relationship diagram of the application's database

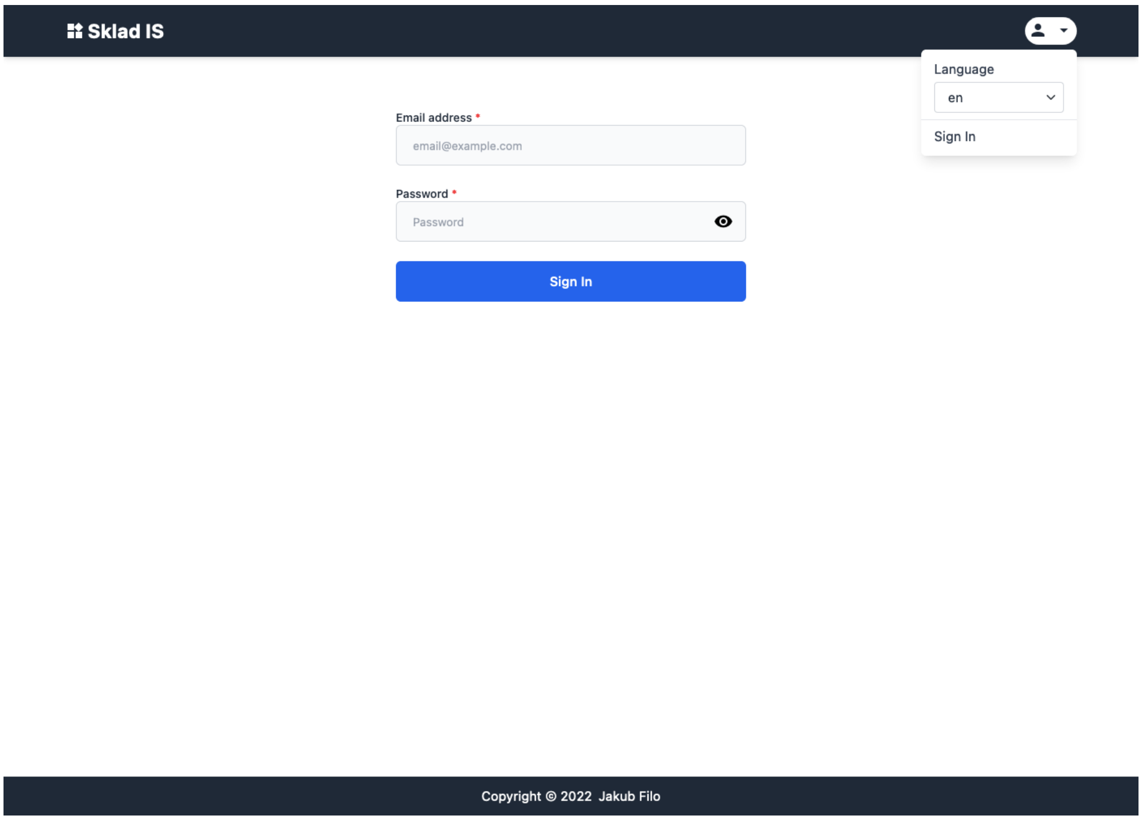
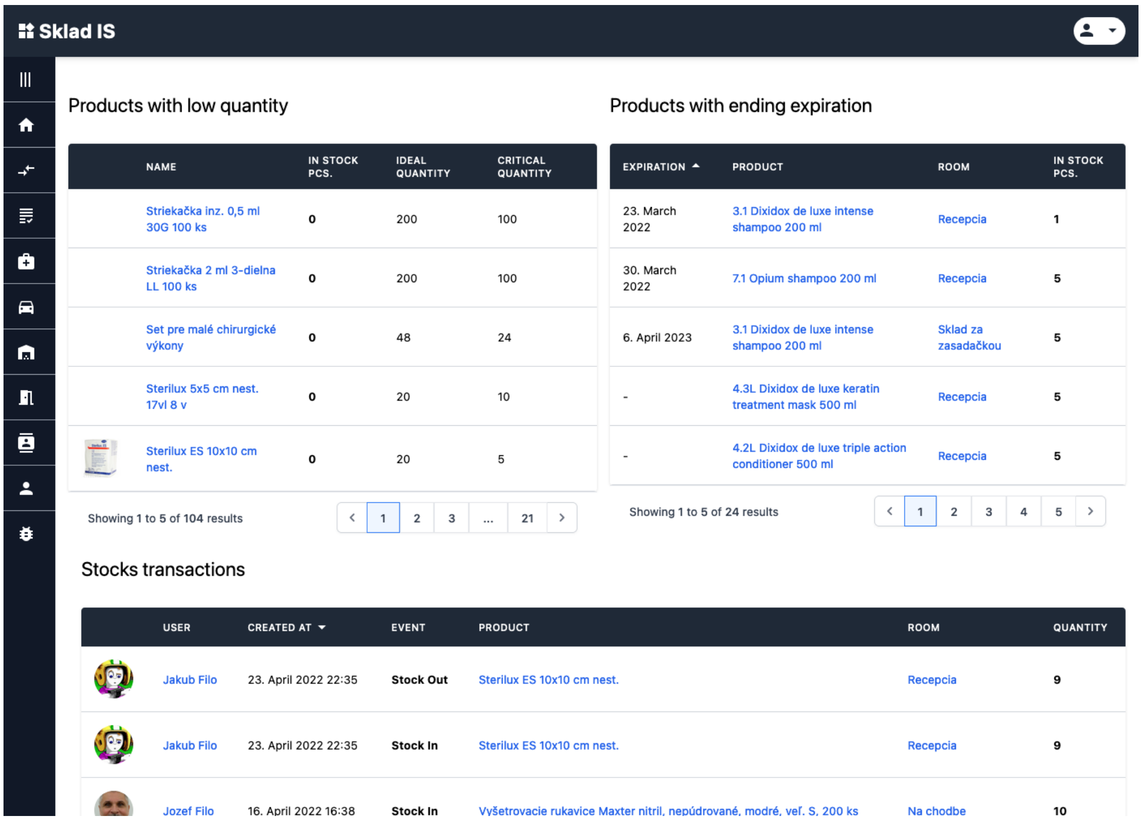


Fig. A.2: Login screen



Products with low quantity

NAME	IN STOCK PCS.	IDEAL QUANTITY	CRITICAL QUANTITY
Striekačka inz. 0,5 ml 30G 100 ks	0	200	100
Striekačka 2 ml 3-dielna LL 100 ks	0	200	100
Set pre malé chirurgické výkony	0	48	24
Sterilux 5x5 cm nest. 17vl 8 v	0	20	10
Sterilux ES 10x10 cm nest.	0	20	5

Showing 1 to 5 of 104 results

< 1 2 3 ... 21 >

Products with ending expiration

EXPIRATION	PRODUCT	ROOM	IN STOCK PCS.
23. March 2022	3.1 Dixidox de luxe intense shampoo 200 ml	Recepcia	1
30. March 2022	7.1 Opium shampoo 200 ml	Recepcia	5
6. April 2023	3.1 Dixidox de luxe intense shampoo 200 ml	Sklad za zasadačkou	5
-	4.3L Dixidox de luxe keratin treatment mask 500 ml	Recepcia	5
-	4.2L Dixidox de luxe triple action conditioner 500 ml	Recepcia	5

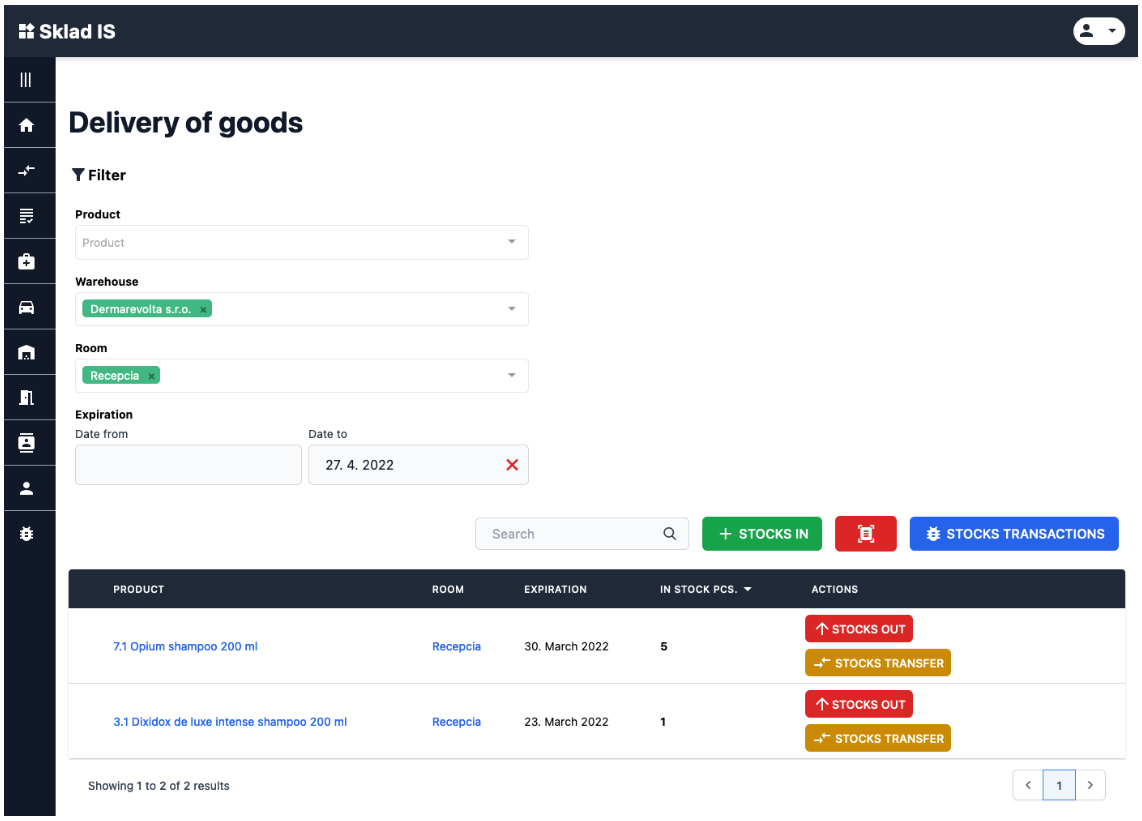
Showing 1 to 5 of 24 results

< 1 2 3 4 5 >

Stocks transactions

USER	CREATED AT	EVENT	PRODUCT	ROOM	QUANTITY
Jakub Filo	23. April 2022 22:35	Stock Out	Sterilux ES 10x10 cm nest.	Recepcia	9
Jakub Filo	23. April 2022 22:35	Stock In	Sterilux ES 10x10 cm nest.	Recepcia	9
Jozef Filo	16. April 2022 16:38	Stock In	Vyšetrovacie rukavice Maxter nitril, nepúdrované, modré, veľ. S, 200 ks	Na chodbe	10

Fig. A.3: Homepage

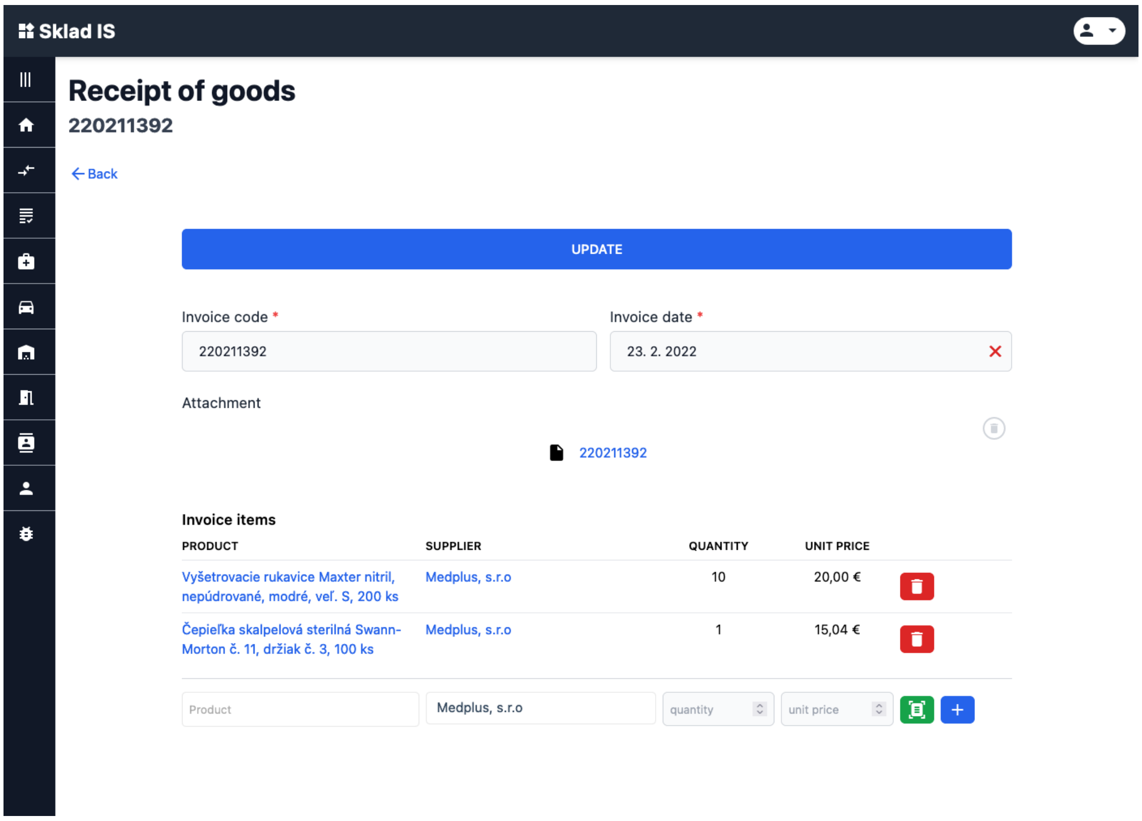


Date to: 27. 4. 2022

PRODUCT	ROOM	EXPIRATION	IN STOCK PCS.	ACTIONS
7.1 Opium shampoo 200 ml	Recepcia	30. March 2022	5	STOCKS OUT STOCKS TRANSFER
3.1 Dixidox de luxe intense shampoo 200 ml	Recepcia	23. March 2022	1	STOCKS OUT STOCKS TRANSFER

Showing 1 to 2 of 2 results

Fig. A.4: Stocks dashboard



Invoice code *

220211392

Invoice date *

23. 2. 2022

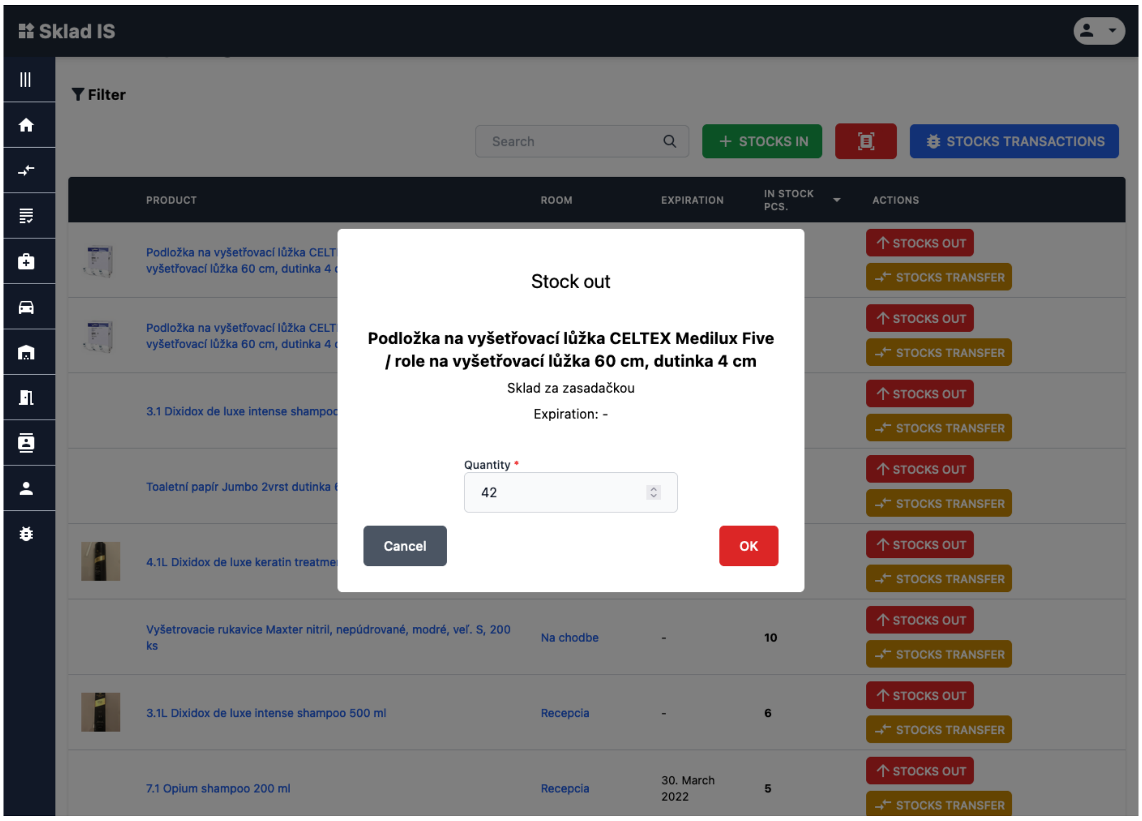
Attachment

220211392

Invoice items

PRODUCT	SUPPLIER	QUANTITY	UNIT PRICE
Vyšetrovacie rukavice Maxter nitril, nepúdrované, modré, veľ. S, 200 ks	Medplus, s.r.o	10	20,00 €
Čepieľka skalpelová sterilná Swann-Morton č. 11, držiak č. 3, 100 ks	Medplus, s.r.o	1	15,04 €

Fig. A.5: Stock in view



PRODUCT	ROOM	EXPIRATION	IN STOCK PCS.	ACTIONS
Podložka na vyšetřovací lůžka CELT vyšetřovací lůžka 60 cm, dutinka 4				STOCKS OUT STOCKS TRANSFER
Podložka na vyšetřovací lůžka CELT vyšetřovací lůžka 60 cm, dutinka 4				STOCKS OUT STOCKS TRANSFER
3.1 Dixidox de luxe intense shampoo				STOCKS OUT STOCKS TRANSFER
Toaletní papír Jumbo 2vrst dutinka				STOCKS OUT STOCKS TRANSFER
4.1L Dixidox de luxe keratin treatme				STOCKS OUT STOCKS TRANSFER
Vyšetrovacie rukavice Maxter nitril, nepúdrované, modré, veľ. S, 200 ks	Na chodbe	-	10	STOCKS OUT STOCKS TRANSFER
3.1L Dixidox de luxe intense shampoo 500 ml	Recepcia	-	6	STOCKS OUT STOCKS TRANSFER
7.1 Opium shampoo 200 ml	Recepcia	30. March 2022	5	STOCKS OUT STOCKS TRANSFER

Stock out

Podložka na vyšetřovací lůžka CELTEX Medilux Five / role na vyšetřovací lůžka 60 cm, dutinka 4 cm

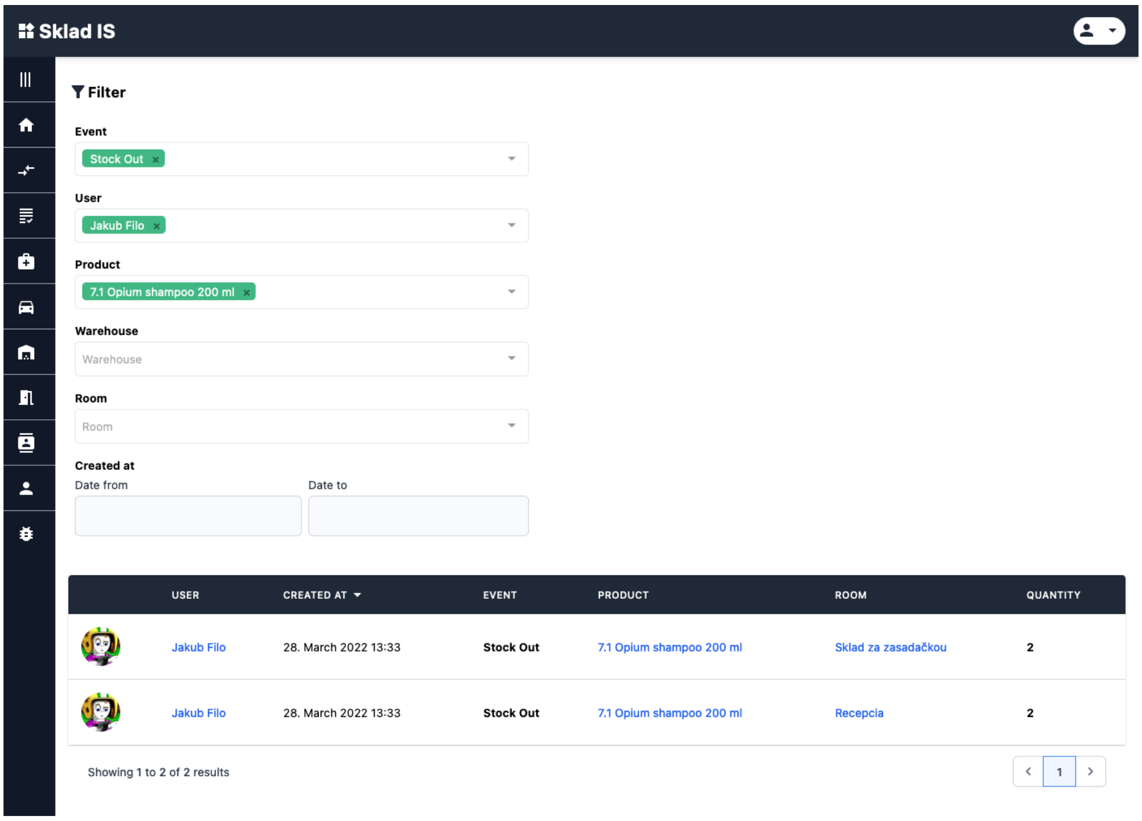
Sklad za zasadačkou

Expiration: -

Quantity *

42

Fig. A.6: Stock out view


Sklad IS

Filter

Event

Stock Out

User

Jakub Filo

Product

7.1 Opium shampoo 200 ml

Warehouse

Warehouse

Room

Room

Created at

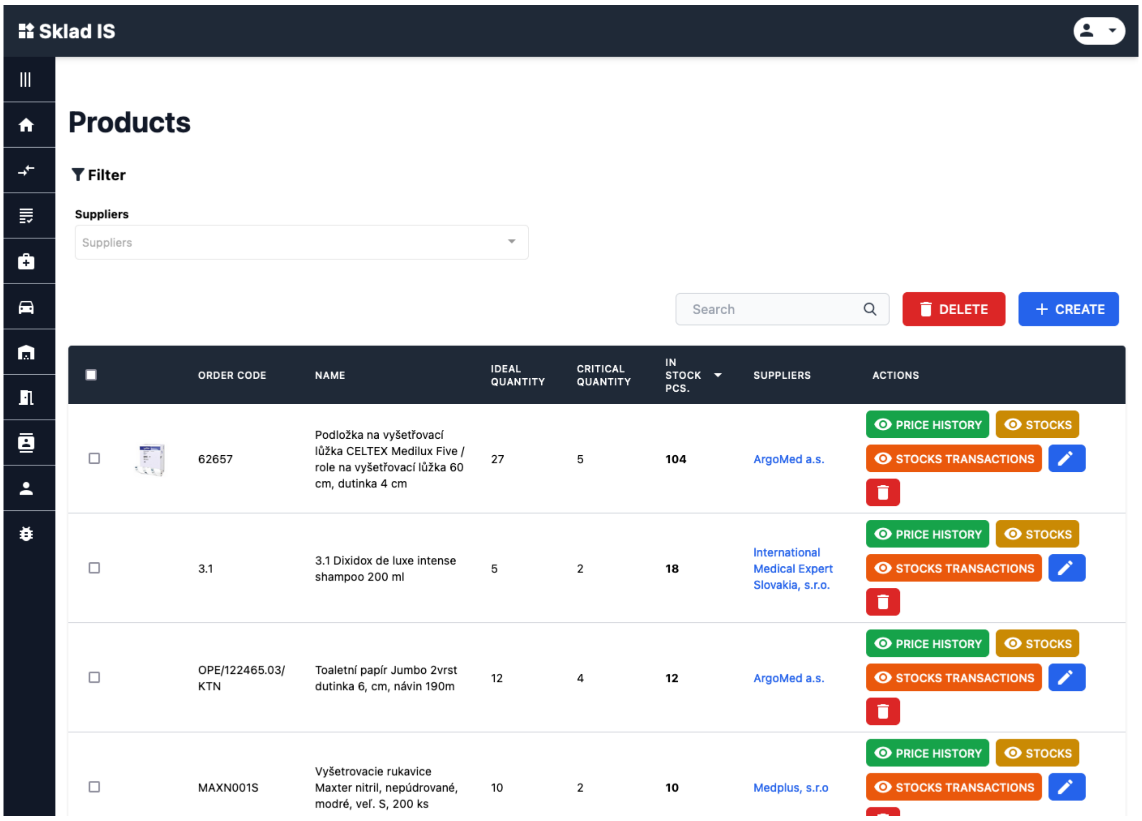
Date from

Date to

	USER	CREATED AT	EVENT	PRODUCT	ROOM	QUANTITY
	Jakub Filo	28. March 2022 13:33	Stock Out	7.1 Opium shampoo 200 ml	Sklad za zasadačkou	2
	Jakub Filo	28. March 2022 13:33	Stock Out	7.1 Opium shampoo 200 ml	Recepcia	2

Showing 1 to 2 of 2 results

Fig. A.7: Stock transactions view



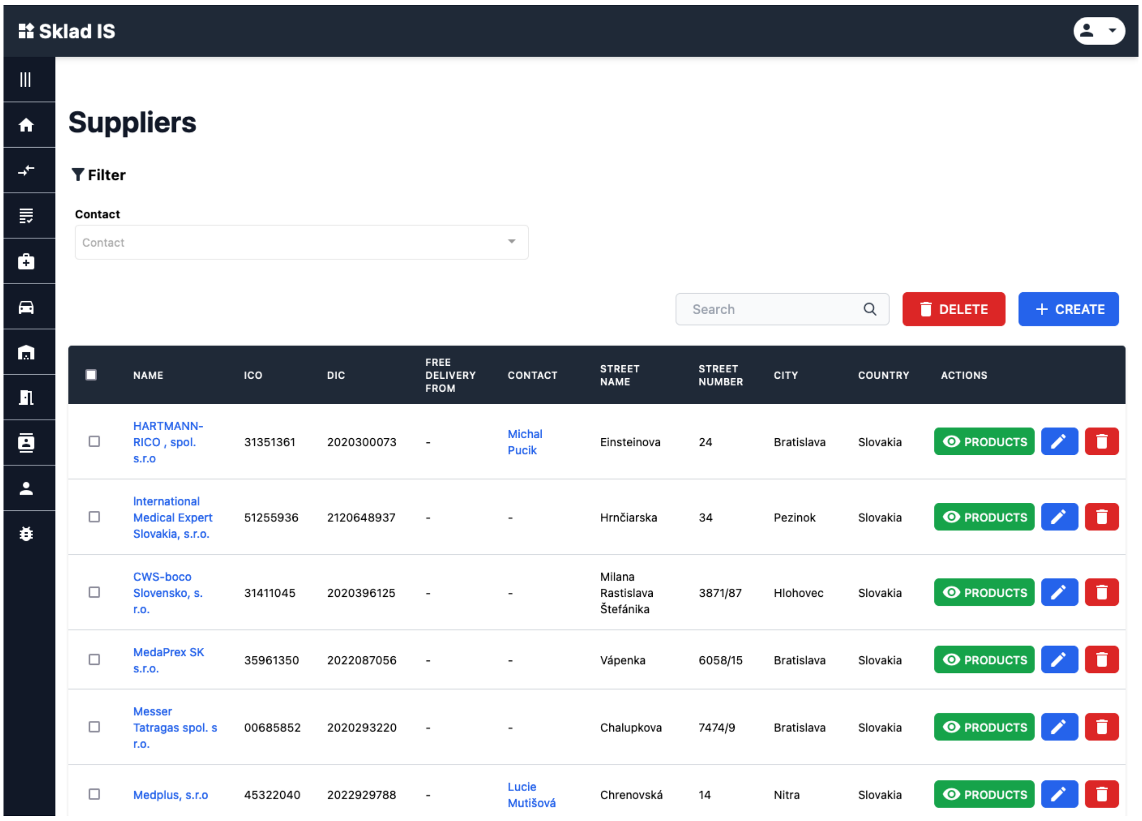
Products

Filter

Suppliers

		ORDER CODE	NAME	IDEAL QUANTITY	CRITICAL QUANTITY	IN STOCK PCS.	SUPPLIERS	ACTIONS
☐		62657	Podložka na vyšetřovací lůžka CELTEX Medilux Five / role na vyšetřovací lůžka 60 cm, dutinka 4 cm	27	5	**104**	ArgoMed a.s.	PRICE HISTORY STOCKS STOCKS TRANSACTIONS
☐		3.1	3.1 Dixidox de luxe intense shampoo 200 ml	5	2	**18**	International Medical Expert Slovakia, s.r.o.	PRICE HISTORY STOCKS STOCKS TRANSACTIONS
☐		OPE/122465.03/ KTN	Toaletní papír Jumbo 2vrst dutinka 6, cm, návin 190m	12	4	**12**	ArgoMed a.s.	PRICE HISTORY STOCKS STOCKS TRANSACTIONS
☐		MAXN001S	Vyšetrovacie rukavice Maxter nitril, nepúdrované, modré, veľ. S, 200 ks	10	2	**10**	Medplus, s.r.o	PRICE HISTORY STOCKS STOCKS TRANSACTIONS

Fig. A.8: Products dashboard



Sklad IS

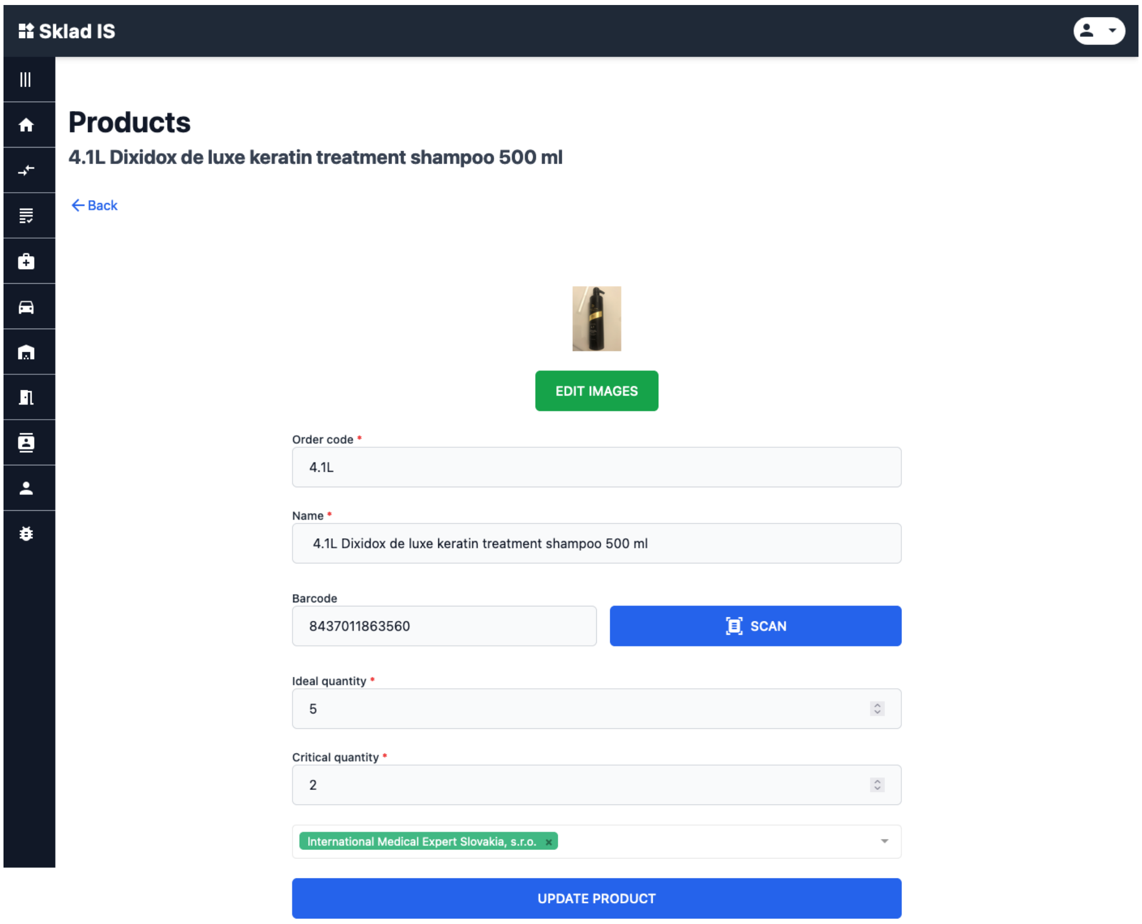
Suppliers

Filter

Contact: Contact

NAME	ICO	DIC	FREE DELIVERY FROM	CONTACT	STREET NAME	STREET NUMBER	CITY	COUNTRY	ACTIONS
HARTMANN-RICO , spol. s.r.o	31351361	2020300073	-	Michal Pucik	Einsteinova	24	Bratislava	Slovakia	PRODUCTS
International Medical Expert Slovakia, s.r.o.	51255936	2120648937	-	-	Hrnčiarska	34	Pezinok	Slovakia	PRODUCTS
CWS-boco Slovensko, s. r.o.	31411045	2020396125	-	-	Milana Rastislava Štefánika	3871/87	Hlohovec	Slovakia	PRODUCTS
MedaPrex SK s.r.o.	35961350	2022087056	-	-	Vápenka	6058/15	Bratislava	Slovakia	PRODUCTS
Messer Tatragas spol. s r.o.	00685852	2020293220	-	-	Chalupkova	7474/9	Bratislava	Slovakia	PRODUCTS
Medplus, s.r.o	45322040	2022929788	-	Lucie Mutišová	Chrenovská	14	Nitra	Slovakia	PRODUCTS

Fig. A.9: Suppliers dashboard



Products

4.1L Dixidox de luxe keratin treatment shampoo 500 ml

Order code *
4.1L

Name *
4.1L Dixidox de luxe keratin treatment shampoo 500 ml

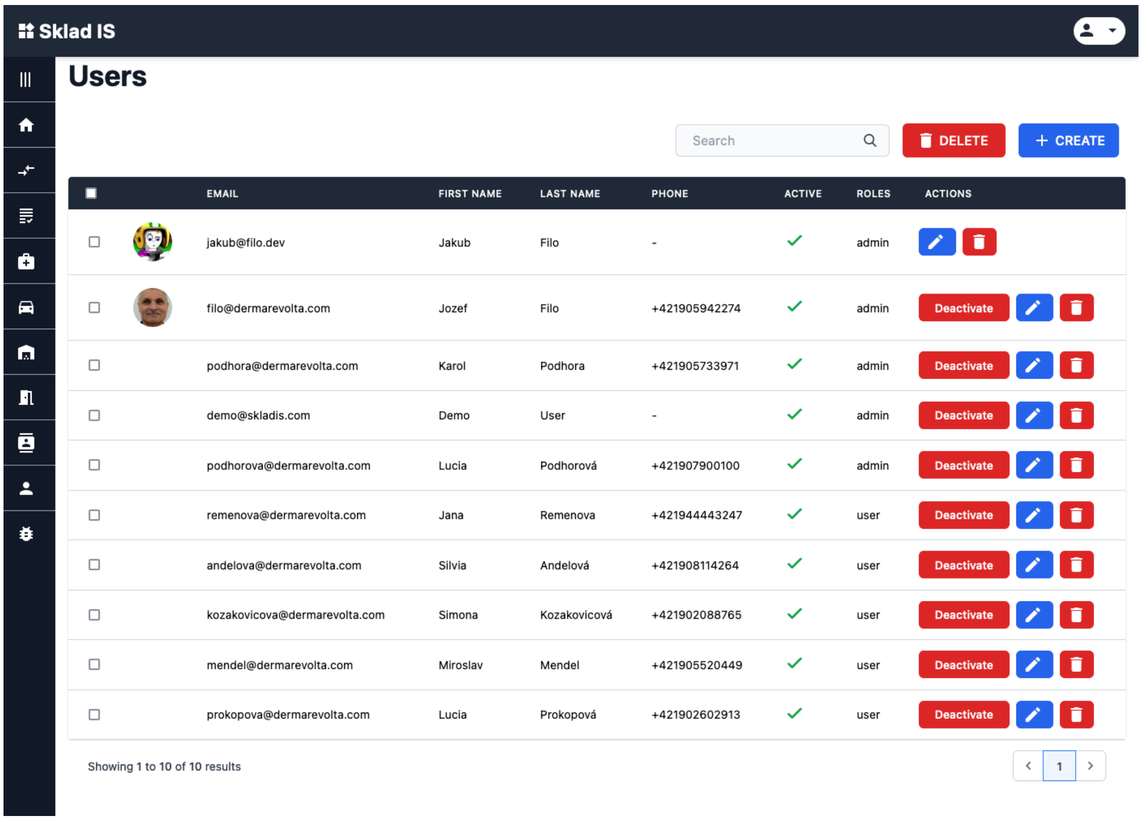
Barcode
8437011863560

Ideal quantity *
5

Critical quantity *
2

International Medical Expert Slovakia, s.r.o.

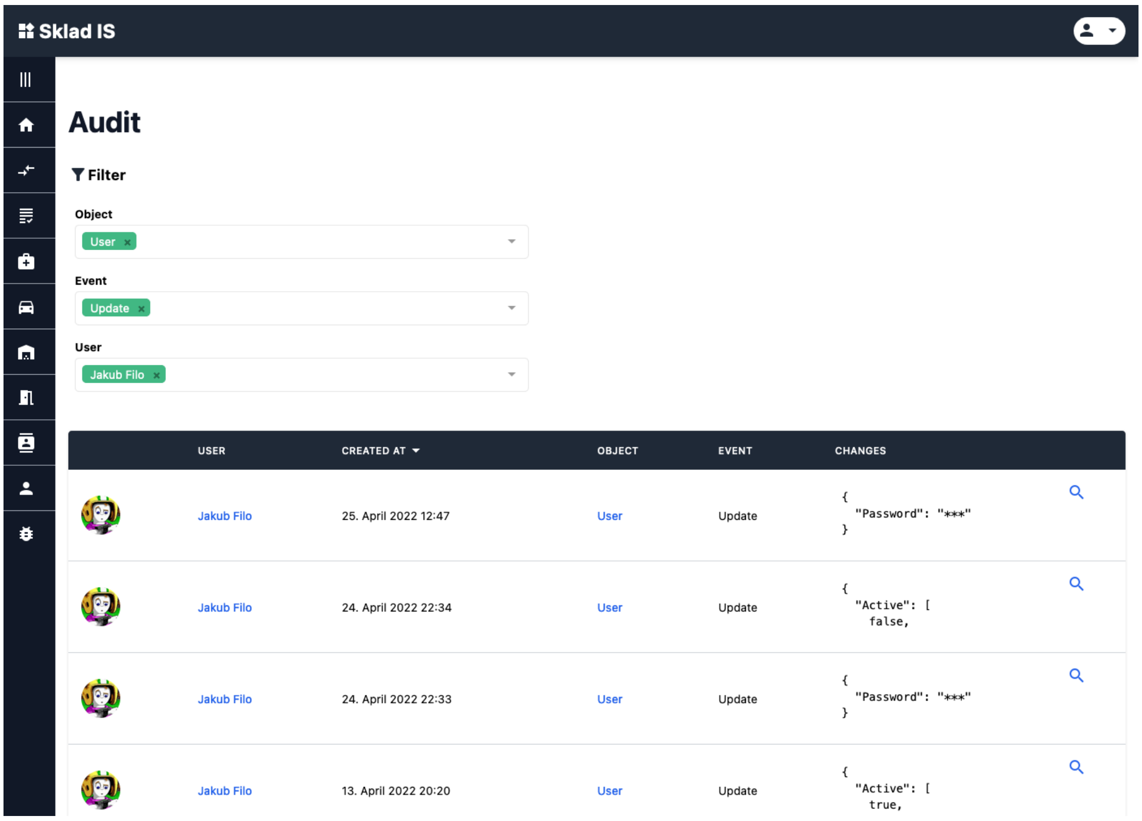
Fig. A.10: Editation of the product



EMAIL	FIRST NAME	LAST NAME	PHONE	ACTIVE	ROLES	ACTIONS
jakub@filo.dev	Jakub	Filo	-	✓	admin	
filo@dermarevolta.com	Jozef	Filo	+421905942274	✓	admin	Deactivate
podhora@dermarevolta.com	Karol	Podhora	+421905733971	✓	admin	Deactivate
demo@skladis.com	Demo	User	-	✓	admin	Deactivate
podhorova@dermarevolta.com	Lucia	Podhorová	+421907900100	✓	admin	Deactivate
remenova@dermarevolta.com	Jana	Remenova	+421944443247	✓	user	Deactivate
andelova@dermarevolta.com	Silvia	Andelová	+421908114264	✓	user	Deactivate
kozakovicova@dermarevolta.com	Simona	Kozakovicová	+421902088765	✓	user	Deactivate
mendel@dermarevolta.com	Miroslav	Mendel	+421905520449	✓	user	Deactivate
prokopova@dermarevolta.com	Lucia	Prokopová	+421902602913	✓	user	Deactivate

Showing 1 to 10 of 10 results

Fig. A.11: Users dashboard



Audit

Filter

Object

User

Event

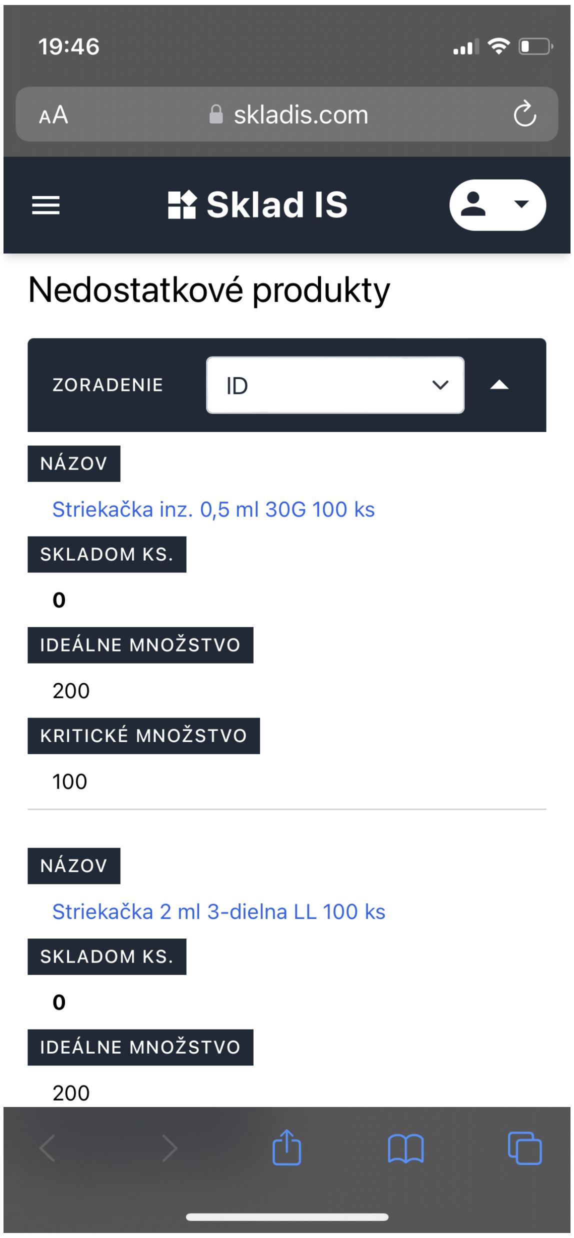
Update

User

Jakub Filo

	USER	CREATED AT	OBJECT	EVENT	CHANGES
	Jakub Filo	25. April 2022 12:47	User	Update	{ "Password": "***" }
	Jakub Filo	24. April 2022 22:34	User	Update	{ "Active": [false,
	Jakub Filo	24. April 2022 22:33	User	Update	{ "Password": "***" }
	Jakub Filo	13. April 2022 20:20	User	Update	{ "Active": [true,

Fig. A.12: Audit dashboard



Nedostatkové produkty

ZORADENIE ID

NÁZOV
Striekačka inz. 0,5 ml 30G 100 ks
SKLADOM KS.
0
IDEÁLNE MNOŽSTVO
200
KRITICKÉ MNOŽSTVO
100

NÁZOV
Striekačka 2 ml 3-dielna LL 100 ks
SKLADOM KS.
0
IDEÁLNE MNOŽSTVO
200

Fig. A.13: Responsivity of the application