

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE**  
**FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-5384-80331

**Inteligentný inventarizačný informačný systém**  
**DIPLOMOVÁ PRÁCA**

Študijný program:	Aplikovaná informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Ústav informatiky a matematiky
Vedúci diplomovej práce:	doc. Ing. Mikuláš Bittera, PhD.





## ZADANIE DIPLOMOVEJ PRÁCE

Autor práce: Bc. Viktor Chovanec  
Študijný program: aplikovaná informatika  
Študijný odbor: informatika  
Evidenčné číslo: FEI-5384-80331  
ID študenta: 80331  
Vedúci práce: doc. Ing. Mikuláš Bittera, PhD.  
Miesto vypracovania: Ústav elektrotechniky FEI STU

Názov práce: **Inteligentný inventarizačný informačný systém**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania: Inteligentný inventarizačný informačný systém (3IS) v súčasnosti pomáha odbúravať náročnú papierovú činnosť pri inventarizácii hmotného investičného majetku ako aj drobného hmotného majetku. Cieľom práce je vytvoriť 3IS pre potreby inventarizácie majetku na Ústave elektrotechniky FEI STU.

Úlohy:

1. Vymedzte funkcionálne a nefunkcionálne požiadavky popisujúce 3IS, ktoré zahŕňujú špecifikáciu potrebných dát a procesov zabezpečujúcich inventarizáciu a lokalizáciu majetku ústavu.
2. Navrhňte zapojenia hardvérovej časti 3IS a prislúchajúceho firmvéru zabezpečujúceho identifikáciu osôb prístupujúcich do pozorovaných miestností pomocou čipových kariet a sledovanie pohybu hmotného majetku na základe pasívnych RFID čipov umiestnených na jednotlivých zariadeniach, pričom účelom hardvérovej časti je získanie dát o lokalizácii zariadení a ich následný prenos pomocou bezdrôtového pripojenia do nadradenej jednotky.
3. Vytvorte webové rozhranie 3IS poskytujúce intuitívne ovládanie navrhnutého systému a interaktívne zobrazenie pozorovaných dát, ako lokalizácia zariadení, inventár jednotlivých miestností. Rozhranie má taktiež poskytovať možnosť importu a zadávania nových dát do systému.
4. Navrhňte a implementujte aplikačné programové rozhranie 3IS zabezpečujúce agregáciu lokalizačných dát získaných z prístupových jednotiek, autentifikáciu a autorizáciu používateľov pre použitie jednotlivých súčastí systému, zabezpečenie lokálneho uchovávaní dát a vytvorenie rozhrania poskytujúceho zabezpečený a centrálny prístup k dátam s využitím architektonického štýlu sietí prepojených systémov REST.
5. Navrhňte infraštruktúru zabezpečujúce bezpečnú komunikáciu medzi jednotlivými komponentami 3IS. Prevádzka celého systému musí byť od základu navrhnutá s možnosťou operácie na systémoch s obmedzeným výpočtovým výkonom (Raspberry), avšak je potrebné uvažovať aj s možnosťou neskoršieho nasadenia na výkonnejší hardvér alebo do distribuovaného centra. Táto požiadavka by mala byť zabezpečená využitím kontajnerizačnej technológie poskytujúcej uniformnosť systému pre nasadenie do veľkého spektra prostredí.
6. V rámci obmedzených možností vyhodnotte realizované riešenie.

### Literatúra:

- UPTON, E. -- HALFACREE, G. *Raspberry Pi: Uživatelská příručka*. Brno : Computer Press, 2013. 232 s. ISBN 978-80-251-4116-8.
- AHSON, S. -- ILYAS, M. *RFID handbook: Applications, Technology, Security, and Privacy*. Boca Raton : CRC Press, 2008. 689 s. ISBN 1-4200-5499-6.

Dátum zadania: 23. 09. 2019

Dátum odovzdania: 15. 05. 2020

Bc. Viktor Chovanec  
študent

Dr. rer. nat. Martin Drozda  
vedúci pracoviska

prof. Dr. Ing. Miloš Oravec  
garant študijného programu



Čestne vyhlasujem, že som túto prácu vypracoval samostatne, na základe konzultácií a s použitím uvedenej literatúry.

V Bratislave, máj 2020

.....  
Bc. Viktor Chovanec

## **Pod'akovanie**

V tomto krátkom segmente by som sa chcel poďakovať vedúcemu projektu doc. Ing. Mikulášovi Bitterovi, PhD., za jeho ochotu pri procese od návrhu, až po samotný vývoj a za poskytnutie cenných informácií počas konzultácií. Taktiež by som chcel vyjadriť vďaku rodine a kamarátom za motivačné citáty *Už to máš? Kedy to odovzdáš?*. Ďakujem!

# ANOTÁCIA DIPLOMOVEJ PRÁCE

Slovenská technická univerzita v Bratislave  
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný odbor: Aplikovaná informatika

Študijný program: Informatika

Autor: Bc. Viktor Chovanec

Diplomová práca: Inteligentný inventarizačný informačný systém

Vedúci práce: doc. Ing. Mikuláš Bittera, PhD.

Mesiac a rok odovzdania: máj 2020

Kľúčové slová: hmotný majetok, lokalizácia, Golang, ESP-MESH, RFID

Diplomová práca sa zaoberá problematikou inventarizácie hmotného majetku rozšírenou o aktuálnu polohu jednotlivých položiek. Monitorovanie pozície hmotného majetku a inventáru organizácie, či spoločnosti je veľmi dôležitou súčasťou optimálneho fungovania interných procesov. Cieľom práce je vytvorenie systému zabezpečujúceho monitorovanie položiek inventáru na Ústave elektrotechniky FEI STU v Bratislave.

Analýza projektu sa zaoberá rozdelením aktuálne dostupných inventarizačných systémov podľa ich určenia a využitej lokalizačnej technológie. Kapitola taktiež obsahuje priblíženie samotných metód sledovania s ich výhodami a nevýhodami v produkčnom prostredí.

Hlavnou časťou projektu je špecifikácia funkcionálnych a nefunkcionálnych požiadaviek aplikácie s následnou implementačnou časťou komplexného inteligentného inventarizačného informačného systému. Vývoj a popis systému je rozdelený na štyri hlavné segmenty. Hardvérová časť projektu sa venuje získavaniu autorizačných dát používateľov a identifikátory sledovaných zariadení. Spracovaniu údajov sa venuje kapitola aplikačného programového rozhrania. Kvôli zabezpečeniu pohodlnej správy systému, bola implementovaná obslužná aplikácia vo forme webového rozhrania. Počas špecifikácie nefunkcionálnych požiadaviek vznikli špecifické predpoklady na produkčné nasadenie systému. Tejto problematike sa venuje kapitola popisujúca infraštruktúru prostredia.

Pre overenie riešenia boli vykonané záťažové testy poskytujúce predstavu o spoľahlivosti a možných vylepšení finálneho riešenia.

# **ABSTRACT OF THE DIPLOMA THESIS**

Slovak University of Technology in Bratislava  
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION  
TECHNOLOGY

Study Branch:	Applied Informatics
Study Programme:	Computer Science
Author:	Bc. Viktor Chovanec
Bachelor Thesis:	Intelligent Inventory Information System
Supervisor:	doc. Ing. Mikuláš Bittera, PhD.
Submitted:	may 2020

Keywords: assets, monitoring, Golang, ESP-MESH, RFID

Master thesis deals with techniques of inventory tracking and asset managing, enhanced with position system for its individual items. Position monitoring of organisation's assets plays very important part in overall optimal functioning of internal processes. Aim of this project is to create a complex system providing monitoring capabilities for items located in the Institute of Electrical Engineering FEI STU Bratislava.

Project analysis breaks down types of currently available inventory systems based on their purpose and used monitoring technology. Part of the chapter deals with the overview of different positioning techniques, stating their advantages and disadvantages that they are posing for production environment.

Core of project consists of functional and non-functional requirements posed on the system operation, followed by implementation process of said inventory system. Description of development process is divided into four main segments. Purpose of the hardware part is to obtain user authorization data and asset tracking information. Subsequent chapter deals with data processing procedures performed by application programming interface. Web interface in form of single page application is created to ensuring best user experience. Last chapter describes system infrastructure, that deals with specific production environment setup.

Implementation part of project was finished by performing stress tests, to get an idea of system reliability and possible improvements to the final product.





# Obsah

Zoznam použitých skratiek a označení .....	15
Úvod.....	17
1 Analýza.....	19
1.1 Klasifikácia systému pozorovania.....	19
1.2 Monitorovanie hmotného majetku .....	19
1.3 Pozorovanie pozície inventára .....	20
1.4 Stratégie a hardvérové implementácie sledovania majetku .....	21
1.4.1 Manuálne spracovanie záznamov .....	21
1.4.2 Možnosti zjednodušenia manuálneho procesu sledovania .....	21
1.4.3 Využitie technológie krátkeho dosahu NFC .....	22
1.4.4 Využitie technológie RFID v procese monitorovania .....	23
1.4.5 Sledovacie zariadenia založené na technológií Bluetooth .....	24
1.4.6 Technológia GPS poskytujúca externú lokalizáciu .....	24
1.5 Monitorovanie majetku a Industry 4.0 .....	25
2 Špecifikácia vypracovaného zadania.....	27
2.1 Cieľ projektu .....	27
2.2 Špecifikácia funkcionálnych a nefunkcionálnych požiadaviek .....	28
2.3 Rozdelenie projektu.....	31
3 Návrh hardvérovej časti.....	33
3.1 Riadiaca jednotka .....	33
3.2 Priblíženie použitej platformy ESP32 .....	33
3.3 Autentifikácia používateľov .....	34
3.3.1 Čítačka založená na čipe MFRC522.....	34
3.3.2 Čítačka založená na protokole Wiegand.....	37
3.3.3 Implementácia protokolu Wiegand v platforme FreeRTOS.....	39
3.4 Prenos údajov využitím ESP-MESH.....	42
3.4.1 Topológia siete.....	43
3.4.2 Druhy uzlov v sieti.....	44
3.4.3 Proces vytvorenia siete .....	46
3.4.4 Voľba rodičovského uzlu.....	47
3.4.5 Zamedzenie vytvoreniu slučiek a ich detekcia .....	48
3.5 Monitorovanie mobility hmotného majetku.....	48

3.6	Autentifikácia používateľa .....	50
4	Návrh webového rozhrania.....	51
4.1	Špecifikácia použitých technológií .....	51
4.2	Priblíženie vytvorených grafických komponentov .....	52
4.2.1	Zoznam systémových zdrojov .....	52
4.2.2	Detail a vytvorenie systémového zdroja .....	53
4.2.3	Priradenie jednotnej systémovej položky .....	54
4.2.4	Komponent umožňujúci výber viacerých položiek .....	55
4.2.5	Dialóg potvrdenia akcií.....	56
4.2.6	Komponent zobrazenia stavových hlásení.....	57
4.2.7	Priradenie systémových zdrojov k importovateľným položkám .....	58
5	Návrh aplikačného programového rozhrania .....	60
5.1	Výber implementačného jazyka .....	60
5.2	Softvérové technológie a hierarchia projektu.....	62
5.2.1	Aplikačné programové rozhranie.....	63
5.2.2	Objekty zabezpečujúce prístup k údajom a migračné operácie .....	64
5.2.3	Funkcionalita konzolového rozhrania.....	65
5.3	Implementácia systémových rozhraní .....	66
5.3.1	Univerzálne operácie rozhrania .....	67
5.3.2	Rozhranie pre import dát .....	68
5.3.3	Časový a dátumový formát vstupných dát.....	70
5.3.4	Transformačné operácie nad dátami .....	71
5.4	Rozhranie zabezpečujúce monitorovanie majetku .....	72
6	Návrh infraštruktúry systému .....	74
6.3	Cieľová hardvérová architektúra .....	74
6.4	Využitie technológie kontajnerizácie .....	74
7	Overenie implementovaného riešenia .....	76
7.3	Odozva aplikačného programového prostredia pod záťažou .....	76
7.4	Odozva bezdrôtového prenosu v závislosti od počtu zariadení .....	77
7.5	Test komplexnej záťaže systému .....	78
	Zhodnotenie .....	79
	Prílohy.....	i
	Príloha A: Ukážka prototypov hardvérových zariadení .....	i
	Príloha B: Systémová príručka .....	iii

Príloha C: Štruktúra repozitára .....	viii
---------------------------------------	------

## Zoznam použitých skratiek a označení

AP	Access Point Prístupový bod
CLI	Command-Line Interface Rozhranie príkazového riadku
CRC	Cyclic redundancy check Kontrola cyklickým kódom
DAC	Digital-to-Analog converter Digitálno-Analógový prevodník
DMS	Document Management System Systém pre správu dokumentov
ERP	Enterprise Resource Planning Plánovanie podnikových zdrojov
GPS	Gloal Positioning System Globálny lokalizačný systém
I2C	Inter-Integrated Circuit Dvojžilová obojsmerná zbernica
IIoT	Industrial Internet of Things Priemyslový internet vecí
IP	Internet Protocol Internetový protokol
ISIC	International Student Identity Card Medzinárodný preukaz študenta
ITIC	International Teacher Identity Card Medzinárodný preukaz pedagogického pracovníka
IoT	Internet of Things Internet vecí
JSON	JavaScript Object Notation Popisná schéma JavaScript objektov
NFC	Near Field Communication Krátkodosahové vysokofrekvenčné bezdotykové spojenie
P2MP	Point-to-multipoint communication

	Konunikácia z jedného vysielacza do viacerých prijímačov
POE	Power over Ethernet Napájanie pomocou sieťového kábla
RFID	Radio Frequency Identification Vysokofrekvenčná identifikácia
SPA	Single-page application Jednostránkové aplikácie
SPI	Serial Peripheral Interface Sériové periférne rozhranie
TTL	Transistor-Transistor logic Logická úroveň Tranzistor-Tranzistor
UX	User Experience Používateľský zážitok
WLAN	Wireless Local Area Network Bezdrôtová lokálna sieť

# Úvod

Monitorovanie pozície hmotného majetku a inventáru organizácie, či spoločnosti je veľmi dôležitou súčasťou optimálneho fungovania interných procesov. Správnym nastavením techniky sledovania aktív môžeme dosiahnuť redukciiu administratívnych nákladov, čím prispejeme k zlepšeniu celkového profitu a umožňuje ľahšie škálovanie podniku. Získavané dáta navyše poskytujú možnosť identifikácie položiek s častým využitím, čo môže viesť k lepšej reštrukturalizácii uloženia vo fyzickom priestore. Skutočná hodnota prichádza s integráciou sledovania inventáru a hmotného majetku do systému plánovania podnikových zdrojov (ERP), umožňujúca precízne riadenie zdrojov s minimálnou stratou, vyplývajúcou z nepresnosti zozbieraných dát.

Z historického hľadiska predstavovalo jedinú možnosť vykonania inventúry zdĺhavý manuálny proces prostredníctvom pera a papiera. Celá operácia sa spoliehala na denník aktivít obsahujúci čas prijatia a výdaja konkrétnej položky, prípadne zaznačenie zodpovednej osoby vykonávajúcej daný úkon. Príchodom výpočtovej techniky vo forme osobných počítačov spôsobilo uľahčenie celého procesu sledovania a správu položiek. Zatiaľ čo tabuľkové procesory a databázy nahradili rôzne papierové záznamy, neodstránili potrebu fyzickej prítomnosti personálu pri samotnom úkone sčítavania a následného zadávania do systému.

Hlavný prelom v oblasti sledovania majetku nastal postupným nasadením technológií inteligentných tovární označovaných ako Industry 4.0, poskytujúcich plnohodnotnú automatizáciu celého procesu monitorovania aktív organizácie. Tento proces zahŕňa kategorizáciu produktov umiestnením rôzneho druhu aktívnych a pasívnych štítkov, poskytujúcich rôznu úroveň presnosti a množstva získaných dát v závislosti od konkrétneho prípadu použitia.

Technológie zabezpečujúce sledovanie a monitorovanie aktív sú v súčasnosti žiadaným rozšírením priemyselného sektoru. Transformácia a informatizácia procesu predstavuje dôležitú úlohu v kľúčových odvetviach, pod ktoré spadá logistika, skladovanie výrobných materiálov, farmaceutický priemysel či automobilová výroba. Nasadením monitorovacieho systému je možné dosiahnuť minimalizáciu skladových zásob, predstavujúcich značné finančné zaťaženie spoločnosti, použitím metódy doručovania just-in-time. Obdobné metódy by neboli účinné v prípade skreslených údajov skladových zásob.

Cieľom diplomovej práce je priblíženie aktuálne dostupných hardvérových riešení s ich benefitmi využitia. Z výsledku analýzy bude následne možné vytvoriť kvalifikovaný odhad

výberu výslednej technológií spĺňajúcej požiadavky pre nami navrhovaný systém. Prehodnotením aktuálne dostupných riešení pre monitorovanie hmotného majetku dokážeme klasifikovať potrebnú funkcionalitu pre úspešné automatizovanie celého procesu. Navrhnutý a následne implementovaný systém inventarizácie hmotného majetku je priamo určený pre nasadenie na Ústav elektrotechniky FEI STU v Bratislave, čím sú ovplyvnené rozhodnutia počas návrhu cieľového riešenia, ktoré najviac vyhovujú tejto aplikácii.



# 1 Analýza

Cieľom kapitoly je priblíženie existujúcich systémov zabezpečujúcich sledovanie hmotného majetku ako aj priblížiť rôzne formy a hardvérové riešenia poskytujúce potrebnú funkcionálnosť pre vykonávanie monitorovania.

## 1.1 Klasifikácia systému pozorovania

Principiálne môžeme systémy monitorujúce pozíciu majetku rozdeliť na dve základné kategórie s ohľadom na problém, ktorý sa snažia adresovať. Pojmy monitorovanie hmotného majetku a riadenie inventáru subjektu sa často zamieňajú napriek tomu, že sú medzi nimi zásadné rozdiely v poskytovanej funkcionalite a softvérovom vybavení zabezpečujúcom operáciu systému [1].

Obe popisované riešenia sú primárne zamerané na optimalizáciu využitia zdrojov a s tým spojené zníženie prevádzkových nákladov. Napriek tomu, že riadenie inventáru a sledovanie hmotného majetku poskytuje spoločnú funkcionálnosť vo forme monitorovania jednotlivých položiek, obe riešenia k tomu pristupujú rôznym spôsobom s ohľadom na to ako prebieha získavanie pozície. Primárne taktiež rozlišujeme dôvody potreby znalosti ich pozície. Zovšeobecniť dosiahneme nasledovné rozdelenie:

- **Hmotný majetok** je zariadenie alebo položka, ktorá je v priamom vlastníctve a je využívaná pre zabezpečenie fungovania prevádzky.
- **Inventár** klasifikuje položky, ktoré sú určené na priamy predaj alebo sú využité pri spracovaní finálneho produktu vo výrobnom procese.

Hmotný majetok je v priebehu času pomerne stabilný a predvídateľný súbor položiek, zatiaľ čo zloženie inventáru je vysoko volatilné s premenlivým cieľom využitia [1].

## 1.2 Monitorovanie hmotného majetku

Sledovanie pozície hmotného majetku je primárne zamerané na položky ako osobné počítače a rôzne softvérové vybavenie, prístroje alebo vybavenie nachádzajúce sa v prevádzke, prípadne nábytok a ostatné položky spoločnosti interne využívané pre svoju vlastnú. Takéto aktíva často krát spadajú pod kapitálové výdavky a v danom čase predstavujú stabilný počet kusov [1].

Systém umožňujúci sledovanie aktív poskytuje spoločnosti značné znížené úsilie pri vykonávaní auditu majetku a zároveň udržiava presnosť záznamov takmer na 100%. Tieto výhody vedú k podstatnej úspore nákladov vynaložených na pracovnú silu a predídaniu

zbytočným stratám vedomosti o zariadení, čo môže viesť k duplicitným nákupom. Ak je požadované sledovanie dôležitej dokumentácie, strata jedného súboru by mohla predstavovať drastické právne alebo finančné dôsledky pre spoločnosť. Pri takomto scenári predstavuje moderný lokalizačný systém majetku rovnako dôležitú úlohu ako zabezpečenie kvality pri výrobných procesoch [1].

Všetky špecifikované výhody vyplývajú zo schopnosti lepšieho určenia polohy jednotlivých položiek aktív spoločnosti, identifikácia používateľov používajúcich dané zariadenia a hodnoty ako čas posledného použitia, vykonanej kalibrácie alebo opravy [1].

### **1.3 Pozorovanie pozície inventára**

Systémy riadenia zásob poskytujú spoločnostiam možnosť spravovať zásoby spotrebného materiálu v rôznych odvetviach ktoré zastupuje. Spadá do úvahy široké spektrum trhov, počnúc od optimalizácie vstupných materiálov potrebných na spracovanie až po správu hotových výrobkov určených na predaj alebo export [2].

Veľkým rozdielom v účele inventáru organizácie od jej hmotného majetku je fakt, že inventárne zásoby sa neustále vyčerpávajú a dopĺňajú. Takáto premenlivosť môže viesť k zložitejšej správe riadenia, a to hlavne v závislosti od veľkosti a komplexnosti použitého inventára. Príkladom môže byť množstvo surovej ocele v automobilovom závode alebo množstvo konkrétneho lieku v lekárni, ktorých množstvo sa odvíja v závislosti od trhových podmienok, sezónnosti alebo predaja [2].

Typický inventárny systém pracuje s údajmi od dodávateľov, ako sú náklady na zaobstaranie, dostupnosť jednotlivých súčiastok, ale poskytuje aj klientske informácie vo forme záznamov predaja, ktoré môžu prispieť k lepšiemu odhadu budúcich výdavkov. Dôležitou funkcionalitou je taktiež sledovanie spotrebného materiálu, ktoré je rozšírené o doplňovanie zásob a vyhľadávanie položiek v sklade alebo na predajniach [2].

## **1.4 Stratégie a hardvérové implementácie sledovania majetku**

Precízny výber spôsobu inventarizácie v značnej miere ovplyvní nastavenie parametrov celého systému od infraštruktúry až po použitú technológiu sledovania. Nasledovná kapitola popisuje metódy a hardvérové riešenia zabezpečujúce sledovanie [2].

### **1.4.1 Manuálne spracovanie záznamov**

Základnou stratégiou je manuálne sledovanie, ktoré v závislosti od veľkosti a rozsahu inventáru môže predstavovať najvhodnejšie riešenie nevyžadujúce značné investície. Pre vykonanie inventúry existujú možnosti vo forme služieb externých spoločností, prípadne alokácia vlastných ľudských zdrojov v rozsahu jedného alebo dvoch pracovných dní v priebehu roka [2].

Každý manuálny krok vykonaný človekom v sebe nesie riziko vzniku chyby z nepozornosti, od spočítania jednotlivých skladových položiek až po samotné zadávanie údajov do systému alebo tabuľkového procesora. Kumulácia takýchto chýb značne zhoršuje presnosť a často je jedinou možnosťou dosiahnutia presnosti opakované vykonanie celého procesu [2].

### **1.4.2 Možnosti zjednodušenia manuálneho procesu sledovania**

Manuálny proces inventarizácie je možné vylepšiť použitím ľahko rozoznateľnými štítkami s čiarovým kódom, ktoré sú už priradené k jednotlivým produktom alebo paletám. Táto forma označenia umožňuje použitie čítacej jednotky predstavujúca zjednodušenie práce personálu a skrátenie procesu registrácie produktu na niekoľko sekúnd. Získavané údaje sa priamo ukladajú v systéme využiteľnej digitálnej forme pripravenej na ďalšie spracovanie. Takto je možné dosiahnutie výsledného počtu aktív v reálnom čase. Výstup pri sčítaní zabezpečenom použitím čiarových kódov predstavuje presnejšie vyčíslenie počtov, z čoho je následne možné robiť lepšie úsudky pri nákupe nových zásob [2].



Obr. 1: Ukážka QR kódu

Vylepšenou formou štítkov je využitie QR kódov reprezentovaných na Obr. 1. Ide o špecifický typ čiarového kódu, ktorého veľkou výhodou je možnosť 360 stupňového snímania v ľubovoľnom smere napríklad pomocou fotoaparátu v mobilnom zariadení. Takáto forma štítkovania otvára možnosť využitia univerzálnych riešení vo forme aplikácie pre mobilné zariadenie v kontraste s tradičnými ručnými čítačkami vyžadujúcim investíciu. Rozšírením, ktorým štandardne čiarové kódy nedisponujú, je možnosť obsiahnutia rozširujúcich dát o samotnej položke. Forma využitého kódovania v prípade QR kódov poskytuje zakomponovanie redundantných korekčných dát až do veľkosti 30%. Toto predstavuje hlavný benefit pri využití v industriálnom prostredí, kde môže prísť k poškodeniu alebo znehodnoteniu počas manipulácie. Kategorizácia zariadení využitím QR štítkov predstavuje najlacnejšiu a najrýchlejšie implementovateľnú možnosť automatického sledovania majetku [3].

#### **1.4.3 Využitie technológie krátkeho dosahu NFC**

Technológiu NFC sme sa rozhodli klasifikovať samostatne aj napriek faktu, že sa jedná o štandard založený na RFID. Hlavným dôvodom je osobitosť atribútov, ktorými disponuje. Výhodou tejto technológie je možnosť načítania využitím moderných mobilných zariadení bez nutnosti externého vybavenia, ako je to v prípade technológie RFID. Takto je možné ušetriť finančné prostriedky odbúraním nutnosti zaobstarávania špecifického druhu hardvéru pre konkrétnu úlohu, pretože obsluhujúci personál dokáže využiť služobné smartfóny. Rovnako ako v prípade QR kódu, technológia slúži na jedinečnú identifikáciu majetku pri jeho skenovaní. Značky vybavené technológiou NFC je možné využiť ako náhradu klasických čiarových alebo QR kódov, ale môžu slúžiť aj ako doplnujúci faktor identifikácie, napriek ich vyššej zaobstarávanej cene [3].



Obr. 2: Ukážka NFC značky [3]

Spoločným faktorom NFC tagov(Obr. 2) a QR kódov(Obr. 1) je obmedzená čítacia vzdialenosť, rádovo v jednotkách až desiatkach centimetrov, čo značne obmedzuje ich využiteľnosť pri diaľkovej lokalizácii, a preto nepredstavujú vhodné riešenie pre systémy vyžadujúce automatické sledovanie majetku v reálnom čase [3].

Medzi ďalšie výhody využitia NFC a principiálne celej rodiny technológie RFID je vyššia rýchlosť načítania identifikátorov v porovnaní s QR kódom. Operačná rýchlosť predstavuje značnú výhodu pri spracovaní veľkého množstva položiek v priebehu dňa. V porovnaní s QR kódom, technológia NFC poskytuje vyššiu flexibilitu v možnosti okamžitej zmeny uložených údajov využitím smartfónu, čo otvára možnosti pre nové aplikácie, ako je uloženie histórie aktíva priamo v štítku [3].

#### **1.4.4 Využitie technológie RFID v procese monitorovania**

Vyššiu formu sledovania je možné dosiahnuť technológiou RFID čipov umiestnených na jednotlivých položkách vyžadujúcich sledovanie. Pri voľbe inventarizačného systému založenom na tejto technológii máme možnosť výberu aktívneho alebo pasívneho sledovania. Aplikáciou technológie aktívnych štítkov vyžadujúcich kontinuálne napájanie, napríklad formou batérie dosiahneme zber údajov o počte a samotnom umiestnení zásob v reálnom čase. Využitie tejto možnosti sa odporúča v prípade, ak je potrebné zabezpečiť väčšiu vzdialenosť prenosu signálu [2].

Obe formy poskytujú automatickú aktualizáciu výsledných počtov okamžite po načítaní identifikátora položky. Pasívna forma technológie môže pri dobrých podmienkach poskytovať čítaciu vzdialenosť do 10 metrov. Prípadným využitím aktívnych štítkov môžeme predpokladať niekoľkonásobne väčšie vzdialenosti, až do 100 metrov. Inštaláciou fixných čítačiek je možné dosiahnuť nepretržitú prevádzku sledovania a aktualizáciu systému [2].

### **1.4.5 Sledovacie zariadenia založené na technológií Bluetooth**

Monitorovacie zariadenia založené na technológií BLE sú popularizáciou IoT a IIoT zariadení na vzostupe. Ide o malé bezdrôtové zariadenia periodicky vysielajúce signál do svojho okolia. Takýmto spôsobom je oznamovaná pozícia pre ostatné položky, ktoré môžu zaregistrovať signál a určiť svoju vzdialenosť od zdroja. Využitím technológie Bluetooth Low Energy dokážeme zabezpečiť kontinuálnu prevádzku v rozpätí niekoľkých rokov, využitím malej batérie ako napájacieho zdroju. Funkčnosťou sú si veľmi podobné aktívnym RFID štítkom [3].

Ide pravdepodobne o najdostupnejší variant v prípade nutnosti aktívneho sledovania hmotného majetku alebo inventáru. Výhoda technológie spočíva v jej rozšírenosti v spotrebiteľskej sfére, čo sa prejaví pri možnosti využitia štandardných smartfónov počas manipulácie, na rozdiel od RFID technológie, ktorá si vyžaduje špecifické vybavenie [3].

### **1.4.6 Technológia GPS poskytujúca externú lokalizáciu**

Špecifickým prípadom použitia môže byť technológia GPS, ktorá je vhodná pre globálne sledovanie aktív, ktoré sa nachádzajú v exteriéri. Využitie tejto technológie nie je optimálne pre sledovanie menších pohybov zariadení nachádzajúcich sa v budovách blokujúcich signál. Nevýhodou využitia je zaobstarávacía cena a forma vyhotovenia samotných sledovacích zariadení, ktoré sú veľkosťou porovnateľné s moderným smartfónom. Prevádzka takéhoto systému si taktiež vyžaduje špecifického poskytovateľa komunikačných služieb disponujúceho globálnou dátovou sieťou, ktorá predstavuje signifikantné mesačné poplatky. Využitie je preto obmedzené na prípady, kedy je potrebné zabezpečiť automatické sledovanie pohybu majetku značnej hodnoty po celom svete [3].

## 1.5 Monitorovanie majetku a Industry 4.0

Koncept internetu vecí(IoT) spočíva v prepojení fyzických zariadení, ako sú senzory, vozidlá alebo aj budovy do siete zabezpečujúcej komunikáciu vo forme výmeny údajov, zhromažďovania informácií, prípadne možnosť diaľkového ovládania. Využitie tejto technológie prináša obrovské rozšírenie možností pre dosiahnutie sledovania hmotného majetku a aktív organizácie. Príkladom môže byť využitie senzorov teploty, vlhkosti, tlaku alebo aj akcelerácie, ktoré na rozdiel od čistej pozície ponúkajú rozšírenú sadu informácií o sledovanom prvku. Získané údaje je následne možné využiť pri analýze kondície, z čoho je možné dosiahnuť prediktívnu údržbu [3].

Špecifikované technológie opísané v kapitole 1.4 predstavovali dostačujúce riešenie, ale úplne nespĺňali požiadavky moderných podnikov, ktoré prejavujú snahu optimalizácie svojich procesov.

S veľkým pokrokom v radarovej technológii dokážu výrobcovia ponúknuť cenovo dostupné širokospektrálne(UWB) radary vhodné pre zabezpečenie lokalizácie hmotného majetku. Výroba a operácia radarovej technológie dlhodobo predstavovali značné finančné prostriedky na zadováženie, a preto bola primárne určená pre vojenské účely. Pokrokom v tomto odbore sa podarilo dosiahnuť cenu za jednotku v desiatkach eur, čo z nej robí vhodného kandidáta na použitie v oblasti inventarizácie. Využitie radarovej technológie prináša nasledovné výhody [4]:

- Desaťnásobné zvýšenie presnosti v porovnaní s použitím GPS, WIFI alebo Bluetooth. Typicky je možné dosiahnuť presnosť lokalizácie na 10 cm.
- Dosah signálu v stovkách metrov s možnosťami dátovej komunikácie
- Veľmi nízka spotreba elektrickej energie a intenzita signálu, ktorá nepredstavuje zdravotné riziká pre ľudí. Emisia vyžiarenej energie zvyčajne predstavuje iba zlomok typického Wi-Fi smerovača.
- Využitie bezlicenčného ISM prenosového pásma, ktoré nevyžaduje certifikáciu ETSI/FCC, vyúsťujúce do nižších nákladov na vývoj.
- Silné penetračné vlastnosti signálu umožňujúce komunikáciu cez steny a dvere, čo predstavuje ideálne využitie v skladoch a budovách.

Využitím širokospektrálnej radarovej technológie umožňuje skutočnú implementáciu sledovania pohybu inventáru v reálnom čase. Či už je vyžadované sledovanie kontajnerov v dodávateľskom reťazci, optimalizácie výrobných procesov, alebo sledovanie hmotného majetku, využitím systému lokalizácie majetku v reálnom čase postavenej na technológii

UWB radarov poskytneme ERP systémom údaje o aktuálnej situácii, na základe ktorej je možné okamžite reagovať na zmeny [4].



## 2 Špecifikácia vypracovaného zadania

Kapitola sa zaoberá špecifikáciou funkcionálnych a nefunkcionálnych požiadaviek potrebných pre realizáciu inteligentného inventarizačného systému.

### 2.1 Cieľ projektu

Dôvodom návrhu a implementácie predmetného informačného systému je zefektívnenie práce, prípadne výučby na fakulte a zvýšenie prehľadnosti o stave a umiestnení hmotného majetku. Účinnosť aktuálne zavedeného systému je vysoko závislá na ľudskom faktore vo forme papierových evidencií a byrokratických procesov, ktoré je možné plnohodnotne automatizovať formou informačného systému, čím bude odstránená ťarcha zodpovednosti a závislosti na človeku.

Komerčné systémy zabezpečujúce obdobnú funkcionalitu sú na trhu dostupné, avšak predstavujú významnú finančnú investíciu do softvérovej licencie a vyžadujú monitorovacie zariadenia industriálnej kvality určené predovšetkým do veľkých firemných skladov. Z tohto dôvodu predstavujú neefektívne riešenie v nami navrhovanom prostredí. Ako príklad si môžeme uviesť popis funkcionálneho modulu majetok zo zdroja [5] z ktorého môžeme pozorovať, že obdobné systémy sú cielené hlavne pre podnikateľsky aktívne subjekty vyžadujúce prepojenie správy majetku s ekonomickým systémom. Takáto funkcionalita nie je v prvom priblížení systému vyžadovaná, a preto by mohla predstavovať zbytočné navýšenie režijných nákladov na správu z hľadiska hardvéru, ale aj operatívneho riadenia z pohľadu obsluhujúceho personálu. Preto sme sa rozhodli identifikovať hlavnú funkcionalitu zabezpečujúcu základné sledovanie hmotného majetku fakulty s čo najväčším ohľadom na používateľskú dostupnosť a jednoduchosť pre kľúčový personál.

Pre dosiahnutie požadovaného odbremenenia používateľov od nutnosti evidencie umiestnenia majetku papierovou formou, ktorá zaberá zbytočný čas a energiu, je potrebné vytvoriť bezkontaktný a „bezobsluhový“ systém zabezpečujúci automatickú lokalizáciu sledovaných položiek. Pred úspešnou realizáciou projektu je potrebné určenie formy sledovania, výber vhodnej technológie zabezpečujúcej zber polohy zariadení a vymedzenie presnosti poskytovanej lokalizácie. Hmotný majetok je v aktuálnom systéme evidovaný k jednotlivým laboratóriám s konkrétnym kódovým identifikátorom, reprezentujúcim fyzickú miestnosť na fakulte. Obdobne bude zabezpečené priradenie zariadenia k miestnosti nami navrhnutým systémom, čím dosiahneme kompatibilitu pre prípadnú migráciu dát. Taktiež

zachováme zaužívanú formu kategorizácie, a tým nenarušíme návyky ľudí, prípadne sa vyhneme namáhavej znovu kategorizácii majetku.

Z tejto formálnej požiadavky nám vychádza ako vhodné riešenie umiestnenie sledovacieho zariadenia do vchodu pri jednotlivých laboratóriách. Toto umiestnenie sa taktiež javí ako vhodné z dôvodu, že väčšina miestností disponuje jediným vstupným bodom, ktorý sa v našom riešení stáva kontrolným stanoviskom jednak pre autorizáciu prístupujúceho používateľa ale aj sledovanie tranzitu samotných zariadení. Posledným bodom je voľba vhodnej technológie zabezpečujúcej sledovanie, ktorá bude viac priblížená v nasledujúcich kapitolách.

## **2.2 Špecifikácia funkcionálnych a nefunkcionálnych požiadaviek**

Z predošlej kapitoly nám vznikla potreba autorizácie prístupujúceho používateľa do jednotlivých zariadení podliehajúcim monitorovaniu. Takáto operácia si taktiež vyžaduje špecifikáciu identifikácie používateľov z čoho následne vzniká potreba výberu vhodného hardvérového riešenia zabezpečujúceho zber autorizačných údajov. Ako vhodnú formu identifikácie používateľa sme vymedzili čipové karty, ktoré sú univerzitou vydávané jednotlivým pedagogickým pracovníkom, a to vo forme ITIC kariet vybavených RFID technológiou poskytujúcou jedinečný identifikátor karty slúžiaci na priradenie k osobnému účtu v našom systéme. Výhodou tohto riešenia je aj fakt, že ide o štandardizovanú a rozšírenú technológiu a z toho dôvodu je možné zabezpečiť prístup generickou čipovou kartou pre osoby nespĺňajúce požiadavky pre vydanie ITIC karty. Voľba čítacej jednotky je bližšie špecifikovaná v kapitole 3.3.

Všetky špecifikované procesy slúžiace na identifikáciu používateľov a lokalizáciu hmotného majetku je potrebné spracovávať lokálne na mieste zberu týchto údajov. Z tejto požiadavky vyplýva nutnosť implementácie vlastného hardvérového riešenia, preferenčne pracujúceho na mikropočítačovej platforme, pomocou ktorej dosiahneme minimalizáciu nákladov na realizáciu, požiadaviek na zabezpečenie napájacej siete a možnosť rýchleho a nezávislého rozšírenia pozorovaných miestností. Pre maximalizáciu mobility nasadenia nami vyvíjaného systému je vhodné použitie bezdrôtových komunikačných technológií založených na technológiách WiFi, čo ovplyvní finálny výber použiteľnej hardvérovej platformy. Vhodne zvolená mikročipová platforma by mala disponovať všetkými potrebnými softvérovými knižnicami zabezpečujúcimi prenos údajov medzi jednotlivými aktívnymi vstupnými bránami systému. Takouto voľbou sú minimalizované náklady na vývoj a testovanie vlastného implementačného riešenia, ktoré by navyše mohlo obsahovať bezpečnostné riziká v prípade

nutnosti využitia vlastného kryptografického overenia dát alebo autorizácie jednotlivých staníc.

Z výslednej špecifikácie hardvérovej časti riešenia sme klasifikovali základnú sadu potrebných dát, ktoré sú touto platformou získavané a vyžadované:

- **Identifikačné číslo karty** – zabezpečuje autentifikáciu a identifikáciu používateľa prístupujúceho do pozorovanej miestnosti.
- **Identifikačné číslo zariadenia** – vo forme pasívneho identifikátora umiestneného sa na jednotlivých položkách hmotného majetku vyžadujúcich lokalizáciu.
- **Identifikátor prístupového miesta** – slúžiaci na priradenie lokalizačných dát k jednotlivým zariadeniam.

Po úspešnej realizácii bezdrôtového prenosu získaných dát z riadiacej jednotky je potrebné vytvoriť centrálné zberné miesto, ktorého hlavnou úlohou je spracovanie a následné uchovanie dát. Preferenčne by malo ísť o centrálny informačný systém implementujúci všetky funkcionálne požiadavky zabezpečujúce spracovanie lokalizačných a autorizačných dát. Pre zabezpečenie intuitívneho a pohodlného ovládania celého systému je potrebná implementácia webového rozhrania. Moderné jednostránkové aplikácie(SPA) vyžadujú istú formu rozhrania pre prístup k dátam pre zabezpečenie manipulácie s nimi. Túto úlohu bude zabezpečovať aplikačné programové rozhranie(API), ktoré bude súčasťou implementácie centrálného informačného systému. Výber technológie a protokolu zabezpečujúceho prenos je opísaný v neskorších kapitolách.

Pre dosiahnutie bezproblémovej migrácie dát do nového systému je potrebné zabezpečiť mostík vo forme súboru s produkčnými údajmi. Aktuálne používaný systém správy hmotného majetku disponuje možnosťou vytvorenia zálohy dát vo formáte tabuľkového procesora Microsoft Excel. Pri samotnej implementácii je potrebné zabezpečiť dátovú kompatibilitu medzi starým a novým formátom údajov a taktiež vybaviť systém potrebnými knižnicami pracujúcimi nad kódovaným súborovým formátom XLSX. Migračná funkcionálna projekta si pre svoje fungovanie vyžaduje implementáciu jednoduchého systému pre správu dokumentov, označovaného aj ako DMS. Slúži ako modul funkcionality zameranej priamo na spracovanie digitálnych súborov, ich ukladanie na hardvérovej úrovni a kategorizácia v databázovom systéme. Takouto izoláciou dosiahneme väčšiu znovu-použitelnosť a zapúzdrenosť systému, kde každý modul spracováva zdroje, ktoré priamo súvisia s jeho činnosťou. Kvôli dosiahnutiu transparentnosti a možnosti návratu do predošlého stavu v prípade nesprávneho spracovania

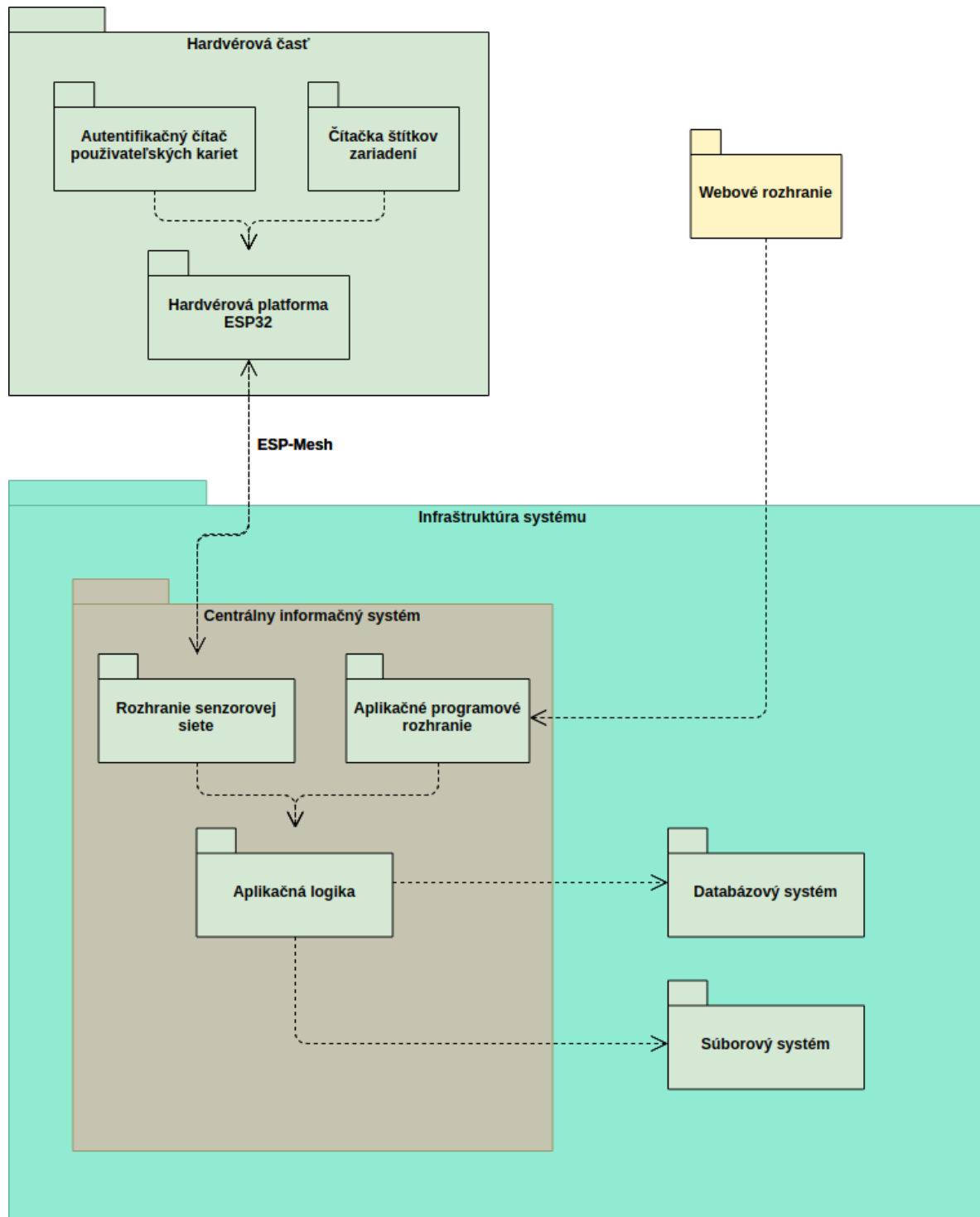
dát bude pri vkladaní do systému vytváraný záznam obsahujúci zoznam referencií novo vytvorených zdrojov.

Dáta obsiahnuté v tabuľkovom súbore určenom pre importovanie do systému sú vyťažované v jednoduchom textovom formáte, ktorý nie je priamo kompatibilný s našim systémom. Preto je potrebné v priebehu procesu zabezpečiť transformáciu dátových formátov. Táto operácia nebude v prvom priblížení vykonávaná automaticky, ale bude závislá od voľby používateľa vykonávajúceho migráciu. Dátové polia, ktoré vyžadujú alebo umožňujú akúkoľvek transformačnú operáciu budú vo webovom rozhraní ponúkať možnosť výberu operácie, ktorá bude vykonaná. Ako príklad takejto operácie môžeme uviesť vyhľadanie jedinečného systémového identifikátora miestnosti na základe jeho kódového označenia, ktoré je ukladané v textovom formáte a pri vytváraní je určované používateľom. Z toho dôvodu sa tento formát nejaví ako vhodný kandidát na unikátny identifikátor záznamu.

Jednou z nefunkcionálnych požiadaviek je dôraz na možnosť prevádzkovať softvérovú časť projektu na nízko nákladovej platforme, ktorú môže predstavovať RaspberryPi alebo iné obdobné zariadenie. Obmedzujúcim faktorom je v tomto prípade hardvérová platforma označovaná ako ARMHF alebo ARMv71, na ktorej sú dané zariadenia postavené a vyžadujú si špecificky prispôsobený operačný systém Linux podporujúci danú architektúru. Z tohto dôvodu vzniká obmedzenie na využitie štandardných softvérových balíčkov určených pre x86 a AMD64 platformu. Táto požiadavka sa prejaví hlavne pri výbere programovacieho jazyka aplikačného programového rozhrania a návrhu infraštruktúry celého systému.

## 2.3 Rozdelenie projektu

Pre dosiahnutie jednoduchšej orientácie počas vývoju systému a popisu samotnej implementačnej funkcionality bol projekt rozdelený na niekoľko hlavných kategórií.



Obr. 3: Diagram komponentov systému

Diagram komponentov na Obr. 3 vyobrazuje rozdelenie systému na nasledovné časti:

- *Hardvérová časť* projektu zabezpečujúca aktívne sledovanie majetku a autorizáciu vstupujúcich používateľov, špecifikovaná v kapitole 3.
- *Webové rozhranie* poskytujúce komfortné riadenie systému pre používateľa opísané v kapitole 4.
- *Centrálny informačný systém* vykonávajúci logiku nad získanými dátami a poskytujúci rozhranie pre webovú a hardvérovú časť projektu, ďalej špecifikovanú v kapitole 5.
- *Infraštruktúra systému* ako celok poskytuje podporu pre prevádzku centrálného informačného systému vo forme výpočtového výkonu a úložného priestoru pre získané dáta. Upresnenie použitých komponentov je opísané v kapitole 6.

### 3 Návrh hardvérovej časti

Kapitola sa zaoberá návrhom hardvérovej časti projektu a procesom výberu jednotlivých fyzických komponentov a zariadení slúžiacich na interakciu systému s používateľom. Taktiež obsahuje teoretické informácie o použitom hardvéri a softvérových protokoloch zabezpečujúcich prenos údajov v rámci infraštruktúry systému.

#### 3.1 Riadiaca jednotka

Základom hardvérovej časti celého projektu je riadiaca jednotka zabezpečujúca zber a spracovanie všetkých potrebných údajov pre chod systému. Počas analýzy funkcionálnych a nefunkcionálnych požiadaviek projektu sme špecifikovali nasledovné body záujmu, ktoré musí navrhnutý hardvér spĺňať:

- Autentifikácia používateľa vstupujúceho do miestnosti pomocou RFID čipovej karty pridelenej jednotlivým zamestnancom fakulty (ITIC/ISIC) alebo generickou prístupovou kartou v prípade, ak používateľovi nebola vydaná zamestnanecká karta.
- Zber údajov o pohybe hmotného majetku v rámci pozorovaného prostredia zabezpečeného bezkontaktným a bezobslužným systémom snímania RFID tagov umiestnených na položkách vyžadujúcich monitoring.
- Bezpečný prenos zozbieraných údajov do aplikačného rozhrania centrálného informačného systému prostredníctvom bezdrôtovej technológie WiFi.

#### 3.2 Priblíženie použitej platformy ESP32

Hardvérová časť projektu je založená na mikropočítačovej platforme ESP32 od spoločnosti Espressif Systems. Kapitola približuje dôvody pre výber tohto produktu ako aj funkcionality, ktoré ponúka.

Predošlá generácia označovaná ako ESP8266 bola vybavená jedným interným výpočtovým jadrom. Aktualizácia Wi-Fi zásobníka vyžadujúca intenzívny multitasking rezervovala väčšinu systémových zdrojov a z toho dôvodu bolo nutné vykonávať používateľskú aplikáciu na separátnom hardvérovom čipe pre dosiahnutie efektívneho spracovania. Príchodom vylepšenej platformy ESP32 bola nutnosť využitia externých komponentov odstránená použitím 32-bitovým dvojjadrovým mikroprocesorom s označením Xtensa® LX6. Štandardne ponúkané moduly pracujú na taktovacej frekvencii v rozmedzí 160 až 240 MHz, čo predstavuje dostatočný výkon pre mikropočítačové aplikácie vyžadujúce konektivitu [6].

Dve dostupné výpočtové jadrá sú pomenované ako Protocol CPU(*PRO\_CPU*) a Application CPU(*APP\_CPU*), čím je ich zameranie rozdelené. Jadro označované ako *PRO\_CPU* je primárne využité pri spracovaní konektivity prostredníctvom rozhraní WiFi a Bluetooth, ale aj iných interných rozhraní ako SPI alebo I2C. Druhé výpočtové jadro označované ako *APP\_CPU* je rezervované pre vykonávanie používateľskej aplikácie. Takéto rozdelenie vykonávania úloh je zabezpečené plánovačom(scheduler), ktorý je integrovaný vo vývojovom rozhraní ESP-IDF, na ktorom je založená naša aplikácia. Základ tohto softvérového balíka je implementovaný na operačnom systéme FreeRTOS, ktorý zabezpečuje prepínanie vykonávaných úloh [6].

Hlavným dôvodom výberu tohto hardvérového riešenia je zabezpečenie bezdrôtovej konektivity prostredníctvom rozhrania WiFi, implementujúci plnohodnotné rozhranie TCP/IP a podpora štandardu 802.11 b/g/n/e/i WLAN MAC doplnené o špecifikáciu Wi-Fi Direct. Pomocou tohto rozhrania je možné zabezpečiť pripojenie s veľkým množstvom štandardných WiFi routerov pracujúcich v režime stanice(vysielača). ESP32 je taktiež schopné vytvoriť prístupový bod s plnou podporou štandardu 802.11 b/g/n/e/i, ktorého využitie prichádza s implementáciou protokolu ESP-MESH, zabezpečujúceho riadenie komunikácie v rámci senzorovej siete [6].

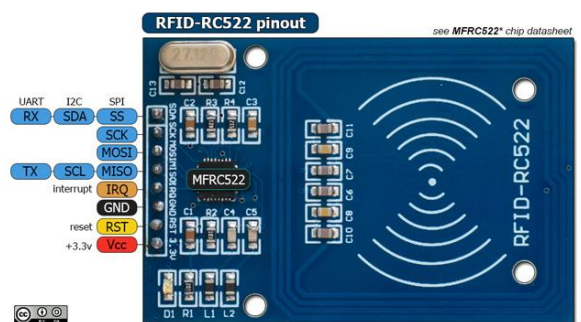
### **3.3 Autentifikácia používateľov**

Pre správne fungovanie celého systému je potrebné zabezpečiť identifikáciu prístupujúceho používateľa do pozorovaného priestoru. Autorizovaný používateľ je vybavený čipovou kartou, ktorá obsahuje jedinečný identifikátor, podľa ktorého je možné danú kartu priradiť konkrétnemu používateľovi v našom systéme. Pre získanie tohto identifikátora je potrebné hardvérovú časť projektu vybaviť čítačkou. Nasledovné podkapitoly sa zaoberajú procesom výberu vhodnej čítačky, popisom použitého protokolu a následnou integráciou do systému.

#### **3.3.1 Čítačka založená na čipe MFRC522**

Prvotnou voľbou čítacieho modulu pre systém bol generický produkt na Obr. 4, založený na čipe MFRC522 od firmy NXP Semiconductors. Dôvodom výberu bola predošlá skúsenosť vývojového tímu s týmto produktom a z toho dôvodu bola možná okamžitá skúška bez problémov a procesu ladenia softvéru.





Obr. 4: Čítacia jednotka založená na čipe MFRC522 [8]

Ide o vysoko integrovanú čítaciu a zapisovaciu jednotku pre bezkontaktnú komunikáciu založenú na frekvencii 13,56 MHz. Integrovaný čip MFRC522 podporuje štandardy pre čipové karty ISO / IEC 14443 A / MIFARE a NTAG. Integrovaný vysielateľ je schopný zabezpečiť riadenie čítania a zápisu pomocou antény spĺňajúcej dané štandardy bez ďalších aktívnych komponentov, čím je možné dosiahnuť vyššiu integráciu na doske plošných spojov. Prijímač poskytuje spoľahlivú a efektívnu implementáciu pre demoduláciu a následné dekódovanie signálu z kariet. Čip je taktiež vybavený funkcionalitou pre detekciu chýb, ktoré môžu nastať v procese bezdrôtového prenosu dát, kontrolou parity dát využitím metódy CRC [7].

Pre rýchly test na zariadení ESP32 sme sa rozhodli využiť platformu Arduino, ktorá obsahuje všetky potrebné knižnice na komunikáciu s čipom. V prípade úspešného testu by sme následne vykonali konverziu všetkých potrebných závislostí pre platformu FreeRTOS. Zdroj [8] uvádza detailný postup, ako je možné zapojiť potrebné komponenty a zdrojový kód potrebný pre skúšku funkcionality.

Nevýhodou implementovaného kódu je fakt, že kontrola prítomnosti karty a možnosť jej načítania je kontinuálne kontrolovaná v nekonečnom cykle, čo spôsobuje zaťaženie a spomalenie celého systému. V rámci optimalizácie je vhodné danú funkcionalitu upraviť tak, aby pracovala v asynchrónnom režime spracovania údajov. Špecifikácia pre čip MFRC522 udáva možnosť aktivácie externých prerušení pomocou výstupu IRQ, ktorý je taktiež dostupný na vývojovej doske. Tab. 1 obsahuje zoznam kandidátov na vhodný typ prerušenia, ktorý by mohol byť použitý na detekciu prítomnosti dát.

Tab. 1: Špecifikácia dostupných typov prerušení čipu MFRC522 vhodných pre detekciu ukončenia prenosu [7]

Kód prerušenia	Zdroj prerušenia	Popis akcie
RxIRq	Prijímač	Ukončenie prijímania dát

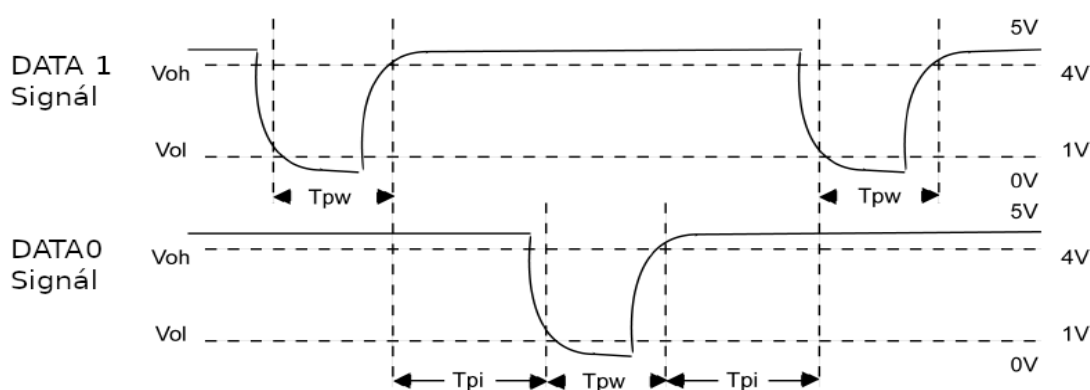
Bitová hodnota označená ako RxIRq registra ComIrqReg indikuje ukončenie prijímania údajov. Vyvolanie prerušenia pri danom stave je podľa dokumentácie možné dosiahnuť jeho aktiváciou v dátovom registri označenom ako ComIEnReg, kde nastavíme príslušnú hodnotu RxIEn na binárnu hodnotu 1.

Testom firmvéru sme nedosiahli očakávané výsledky. Signalizácia pomocou prerušenia nereprezentovala stav ukončenia prijímania dát z čipovej karty, pretože boli zaznamenané v zjavne náhodných časových intervaloch aj v prípade, ak nebola priložená žiadna karta. Pre kontrolu našej implementácie sme využili ukážkový príklad z knižnice zo zdroja [9]. Pri teste sme pozorovali obdobné správanie ako v prípade našej implementácie, a preto sme chybu priradili k hardvérovému riešeniu tejto čítacej jednotky. Zdroj [9] taktiež uvádza nefunkčnosť implementácie vývodu pre prerušenia.

Nefunkčná asynchrónna indikácia prítomnosti karty predstavovala prvý a hlavný dôvod, prečo sme sa rozhodli nevyužiť tento produkt. Druhým dôvodom bolo prevedenie samotného hardvéru, ktorý bol na úrovni určenej pre domáce použitie, kde nie je vyžadovaný žiadny stupeň ochrany krytom špecifikovaný normou IP (International Protection / Ingress Protection). Touto ochranou je zamedzené prístupu používateľa k elektronickým súčastiam zariadeniam a zamedzenie poškodenia hardvéru prachom a vodou.

### 3.3.2 Čítačka založená na protokole Wiegand

Druhou voľbou pre projekt bolo využitie priemyselne založených technológií, ktoré sú overené v praxi. Takouto možnosťou sú čítačky založené na komunikačnom protokole Wiegand. Ide o štandard systémov pre kontrolu prístupu a od 80. rokov sa stal preferovanou komunikačnou voľbou pre tieto systémy. Hlavnou úlohou Wiegand štandardu je prepojenie čítačiek kariet, snímačov otlakov prstov, prípadne iných biometrických snímačov medzi sebou a systémom zabezpečujúcim kontrolu prístupu. Historicky taktiež označuje 26 bitový formát, ktorý reprezentuje konkrétny binárny formát využitý v magnetických páskach a číповých kartách založených na frekvencii 125 kHz [10] [11].



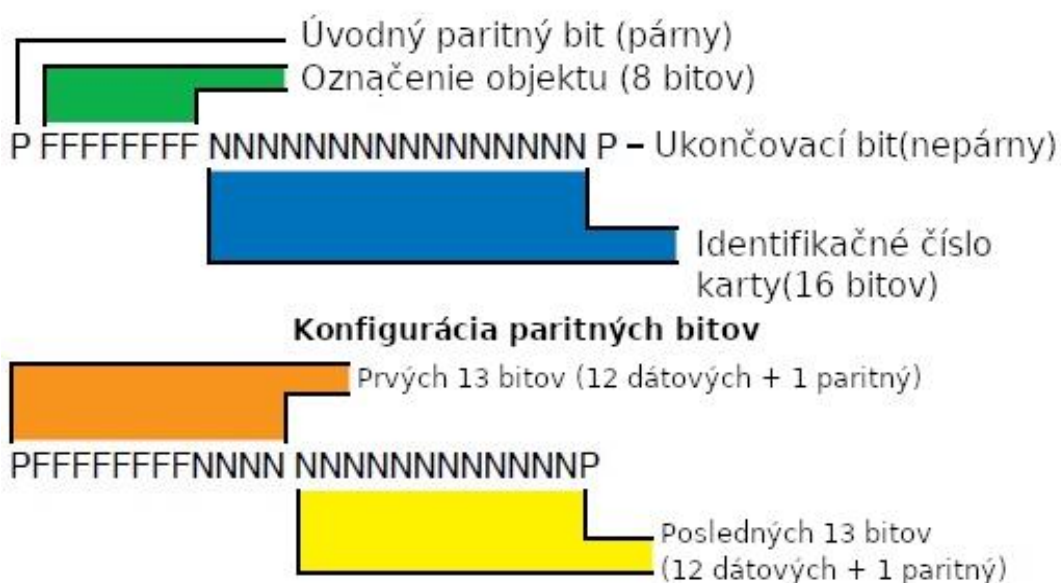
Obr. 5: Reprezentácia typického dátového prenosu protokolu Wiegand [11]

Rozvodný systém protokolu Wiegand využíva pri prenose dát spínané napätie, najčastejšie TTL logiky(+5V) na každom dátovom vodiči označovanom DATA0 a DATA1. V prípade, ak neprebíha prenos údajov, je na oboch vodičoch pripojené rovnaké kladné napätie. Prenos logickej nuly prebieha uzemnením vodiča DATA0, pričom napätie na vodiči DATA1 zostáva nezmenenej pozitívnej hodnote. Logická jednotka je signalizovaná obdobným spôsobom na dátovom vodiči DATA1, kde na linke DATA0 ostáva napätie stabilné. Dátový prenos reprezentovaný na Obr. 5 ukazuje signalizáciu logických hodnôt, ktorá je obmedzená na časový interval 50 mikrosekúnd(Tpw), pričom interval medzi nasledujúcim prenášaným bitom je 1 až 2 milisekundy(Tpi) [11] [10].



Obr. 6: Úsek z výstupu logického analyzátoru pozorujúceho protokol Wiegand 34

Využitím digitálneho logického analyzátoru signálu môžeme získať reálne časové intervaly, ktoré je možné pozorovať na Obr. 6. Z dát, ktoré sú nám poskytnuté vieme vyčítať, že reálna hodnota intervalu  $T_{pw}$  je 400 mikrosekúnd, a to predstavuje osemnásobné predĺženie intervalu oproti stanovenej hodnote protokolu. Takáto zmena oproti predpísanej špecifikácii však neprestavuje problém pri implementácii, pretože nejde o nami softvérovo spracovávaný údaj; naopak môže priniesť pozitívny efekt v prípade mikropočítačov pracujúcich na nižších vzorkovacích frekvenciách. Údaj, ktorého hodnota by nás mohla ovplyvniť, je výpočet hodnoty  $T_{pi}$ , ktorú vieme získať z periódy signálu 2,387 ms. Hodnota  $T_{pi}$  potom predstavuje rozdiel času periódy a hodnoty  $T_{pw}$  čo vo výsledku predstavuje približne 2 ms. Toto je hodnota, ktorá spadá do rozsahu definovaného špecifikáciou protokolu. Z dôvodu, že ide o hornú hranicu, sme sa rozhodli v softvérovej implementácii stanoviť čas ukončenia zberu údajov na 5 milisekúnd, čím zaistíme ochranu pred nechceným predčasným ukončením zberu v prípade väčšieho oneskorenia prichádzajúcich impulzov.



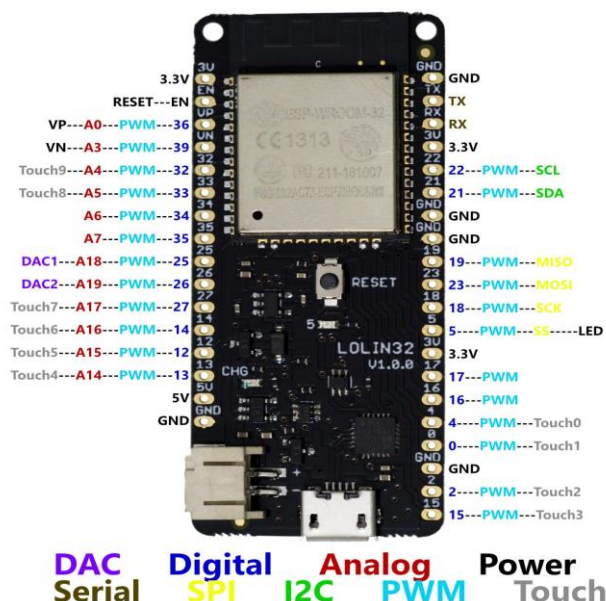
Obr. 7: Rozloženie dátových bitov protokolu Wiegand 26 [11]

Obsah dátovej komunikácie je zložený z 26 bitov(Obr. 7); obsahuje dva kontrolné paritné bity, 8 bitov označujúcich objekt (facility alebo site kód) a 16 bitov rezervovaných pre

identifikačné číslo karty. Táto minimalistická bitová sekvencia poskytuje len veľmi málo možností pre kryptografické operácie a overovanie, čím sa stáva zraniteľná a ľahko napadnuteľná. Toto bol jeden z dôvodov prečo bola navrhnutá veľká rada proprietárnych úprav štandardu, rozšírených na 30 až 42 bitov. Označenie Wiegand 26 označuje konkrétny štandardizovaný protokol, ale pri označeniach Wiegand 34 existuje viac ako 100 odlišných 34 bitových formátov. Pri využití moderných čítačiek založených na prenosovej technológii Wiegand sa očakáva spracovanie kódu kariet na strane riadiacej jednotky, a preto sa z protokolu zachovalo iba zapojenie a princíp prenosu dát [11] [10].

### 3.3.3 Implementácia protokolu Wiegand v platforme FreeRTOS

Podľa špecifikácie popísanej v predošlej podkapitole je možné vytvoriť načítanie údajov a ich následné dekódovanie v nami preferovanej platforme FreeRTOS, ktorú využíva hardvérová časť projektu. Z charakteristiky protokolu je možné usúdiť, že asynchrónne načítanie údajov je možné zabezpečiť pomocou prerušení, keďže ide o spínaný protokol. V tomto prípade sú vyžadované dva hardvérové vstupy pre dátové linky DATA0 a DATA1.



Obr. 8: Označenie vstupno/výstupných periférií hardvérového zariadenia Lolin32 [6]

Prvým krokom pre úspešné načítanie prenášaných údajov z čítačky je voľba vhodných vstupov, ktoré ponúka použitá hardvérová platforma Lolin32. Detekcia zmien logickej hodnoty na vstupoch nevyžaduje žiadny špecifický dekódovací modul ako v prípade využitia

zbernice SPI alebo I2C a preto je vhodné tieto periférie prenechať pre prípadné neskoršie využitie. Z tohto dôvodu vynechávame nasledovné vstupy:

- 21,22 (I2C)
- 5, 18, 23, 19 (SPI)
- 25, 26 (DAC)

Ako vhodných kandidátov volíme vstupy označené 4 a 17, ktoré sú strategicky od seba vzdialené o vzdialenosť minimálne jedného výstupu, čím minimalizujeme možné elektromagnetické rušenie, ktoré vzniká rýchlou zmenou logických hodnôt na dátových linkách a mohlo by neskôr spôsobovať skreslenie pri procese čítania. Oba vstupy obsahujú aj alternatívnu funkcionality ako šírkový modulátor signálu (PWM) a kapacitnú detekciu dotyku (Touch), ale ako je možné vidieť na Obr. 8, táto funkcionality je dostupná na väčšom množstve vstupov, a preto nedochádza k blokovaniu pre neskoršie použitie.

Zvolením vhodných vstupov je potrebné špecifikovať režim, v ktorom budú operovať. Reprezentácia prenášaného signálu na Obr. 5 obsahuje všetky potrebné informácie pre vykonanie konfigurácie. Dátová hodnota je signalizovaná odpojením napájacieho napätia, čím nastane záporná zmena, a preto vstupy konfigurujeme na detekciu negatívnej hrany *GPIO\_PIN\_INTR\_NEGEDGE*. Stabilitu prenášaného signálu zabezpečíme aktivovaním pull-up odporu, ktorým predídeme rušivým elementom vznikajúcich pri spínaní linky. Posledným konfiguračným krokom je aktivácia systému zabezpečujúceho spracovanie prerušení zavolaním príslušnej funkcie *gpio\_install\_isr\_service* a registrácia obslužnej funkcie použitím *gpio\_isr\_handler\_add*, ktorá je zavolaná pri detekcii prerušenia na danom vstupe.

Úvodným testom implementácii sme pomocou jednoduchého výpisu získali nasledujúcu binárnu sekvenciu: 0110111110100001111100000101110110, čím sme si potvrdili funkčnosť nami implementovaného riešenia, a zároveň sme získali identifikátor, že ide o protokol Wiegand 34 na základe prijatého počtu dát.

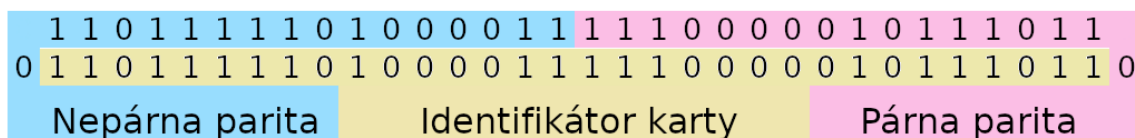
Pred začiatkom vývoja samotného algoritmu zabezpečujúceho dekódovanie a overenie získanej binárnej sekvencie je potrebná dôkladná analýza vyžadovaného modelu uchovávaného potrebné dáta. Z popisu v kapitole 3.3.2 sme získali nasledovné body záujmu:

- **Uchovanie prijatej sekvencie** – pri tomto bode je dôležité si uvedomiť veľkosť spracovávaných dát, ktorá prekračuje kapacitu štandardne využívaného číselného typu *unsigned int* limitovaného na 32 bitov. Protokol vyžaduje minimálnu veľkosť 34 bitov, a preto sa nám naskytujú dve možnosti riešenia; rozdelenie dát na dva segmenty(horných a dolných 32 bitov) obsiahnuté v premenných typu *unsigned int*.

Druhou, preferovanou alternatívou je využitie typu *unsigned long long*, ktorý je platformou ESP32 podporovaný a poskytujúci dátovú kapacitu 64 bitov.

- **Čas posledného prerušenia** – poskytuje možnosť využitia ako podmienené ukončenie zberu dát a ich následné spracovanie. Získavanie aktuálneho systémového času prebieha v obsluhu prerušenia, a preto je potrebné dbať na využitie funkcionality určenej pre tento prípad použitím funkcie *xTaskGetTickCountFromISR*, ktorá pracuje v neblokujúcom režime a predstavuje bezpečnú variantu v asynchrónnom režime.
- **Počet prijatých dát** – realizované ako jednoduché inkrementálne počítadlo. Na základe výsledku je možné determinovať, o akú variantu protokolu Wiegand ide(26/34), prípadná detekcia rušenia, ktoré sa prejaví nesprávnym počtom prijatých bitov(menej ako 26 alebo viac ako 34).

Po vypracovaní špecifikácie, ktorú musí spĺňať algoritmus, je možné prejsť do vývojovej fázy. Spracovanie autorizačných dát z čítačky prebieha izolovane od samotného zberu a to ako samostatná inštancia FreeRTOS úlohy označovanej ako *Task*. Oddelenie časti spracovania dát predstavuje optimalizáciu systému, kedy komplexné výpočty vyťažujúce systémové prostriedky neprebiehajú v obsluhu prerušenia a tak nenastáva blokovanie jeho priebehu, čím minimalizujeme pravdepodobnosť straty prijímaných dát v dôsledku znemožnenia spracovania nasledujúceho impulzu. Úvodnou podmienkou pre vykonanie dekódovania je kontrola času posledného prijatého bitu. Špecifikácia klasifikuje rozmedzie medzi jednotlivými impulzmi na 1 až 2 milisekundy, a preto sme sa rozhodli pre bezpečnú voľbu 5 milisekúnd, aby sme pokryli možné odlišnosti medzi jednotkami rôznych výrobcov. Nasledujúcim krokom je detekcia protokolu na základe veľkosti prijatých dát, kde v aktuálnej implementácii podporujeme šírku 26 a 34 bitov.



Obr. 9: Rozloženie dát protokolu Wiegand 34

Identifikačný kód karty je získavaný aplikovaním bitovej masky predstavujúcej hodnotu *0xFFFFFFFF* prostredníctvom operácie binárneho násobenia *AND* na dáta neobsahujúce prvý paritný bit odstránený binárnym posunom o jeden bit vpravo. V tomto bode si môžeme uvedomiť fakt, že vykonaním tejto matematickej operácie dostávame číslo ohraničené maximálne na 32 bitov, čo predstavuje hraničnú hodnotu typu *unsigned int*, a preto zaniká

potreba využitia typu *unsigned long long* pre uskladnenie dát, čím ušetríme pamäť a zjednodušíme budúce operácie nad danou hodnotou.

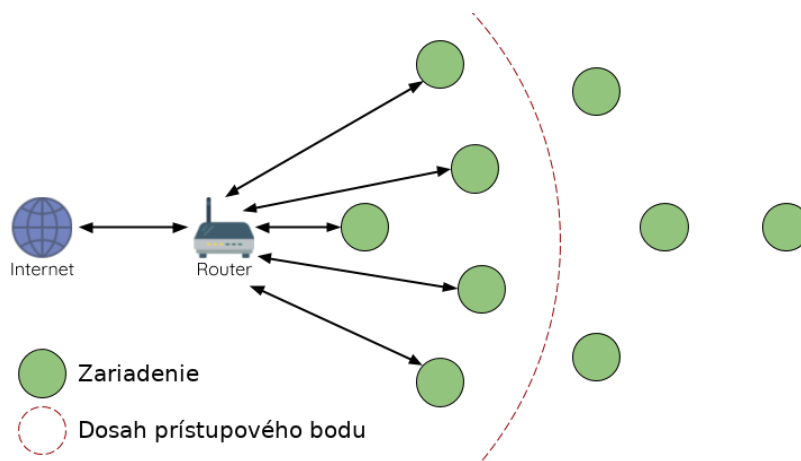
Kvôli zabezpečeniu integrity prijatých dát vykonávame kontrolu paritných bitov posielaných spolu s identifikátorom. Na prvom riadku Obr. 9 môžeme vidieť rozloženie dát na dva rovnomerné segmenty s ich príslušnosťou k paritnému bitu nachádzajúceho sa v prijatých dátach. Protokol Wiegand využíva dva druhy parity:

- **Párna parita** – je reprezentovaná binárnou hodnotou 1 v prípade, ak je zvyšok po delení dvoma súčtu kladných binárnych hodnôt v kontrolovanom segmente 0, čo reprezentuje fakt, že súčet je párne číslo.
- **Nepárna parita** – reprezentuje stav, kedy je súčet jednotiek v segmente nepárnym číslom, takže výsledok po delení dvoma je nenulová hodnota.

Po úspešnom overení oboch paritných bitov je identifikačné číslo karty zaradené do fronty pre ďalšie spracovanie v systéme.

### 3.4 Prenos údajov využitím ESP-MESH

Hardvérová časť projektu využíva technológiu ESP-MESH, ktorá zabezpečuje bezdrôtovú komunikáciu medzi riadiacou jednotkou a aplikačným rozhraním. Ide o sieťový protokol, ktorý predstavuje nadstavbu Wi-Fi štandardu. Využitím tohto protokolu zabezpečíme prepojenie zariadení na veľkej ploche prostredníctvom jednotnej WLAN, pričom je taktiež zabezpečená autonómna organizácia a oprava bezdrôtovej siete [12]

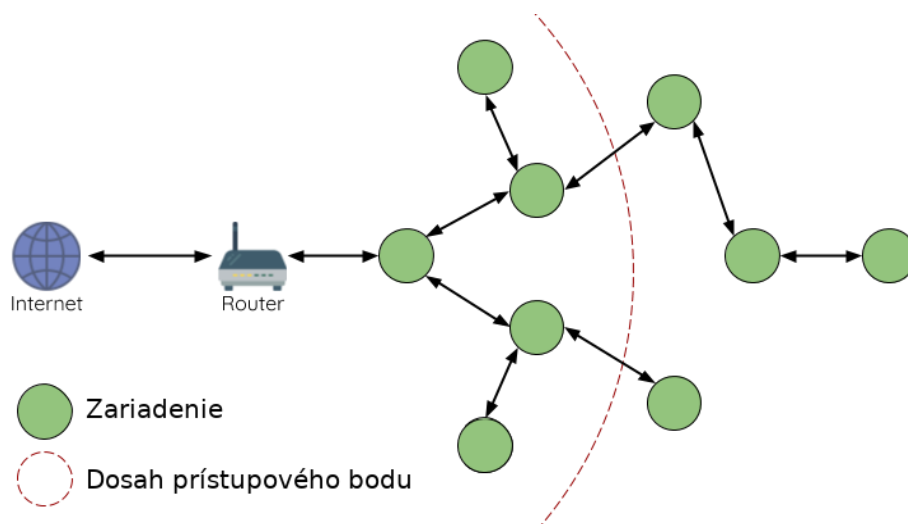


Obr. 10: Tradičné rozloženie Wi-Fi architektúry [12]

Príklad na Obr. 10 reprezentuje tradičnú bezdrôtovú infraštruktúru založenú na technológii Wi-Fi pracujúcej na princípe P2MP, kde jeden ústredný uzol, označovaný ako prístupový bod (AP), je priamo prepojený so všetkými ostatnými uzlami. Hlavnou úlohou AP



je smerovanie prenosu medzi pripojenými zariadeniami a v niektorých prípadoch zabezpečiť komunikáciu s externou IP sieťou poskytnutou routerom. Veľkou nevýhodou tohto štandardu je obmedzené pokrytie z dôvodu požiadavky na jednotlivé stanice, ktoré musia byť v dosahu prístupového bodu, aby mohla byť zabezpečená priama dátová komunikácia. Druhým limitujúcim faktorom je maximálny počet staníc povolených na konkrétnom AP ohraňovaných jeho kapacitou a výpočtovým výkonom [12].



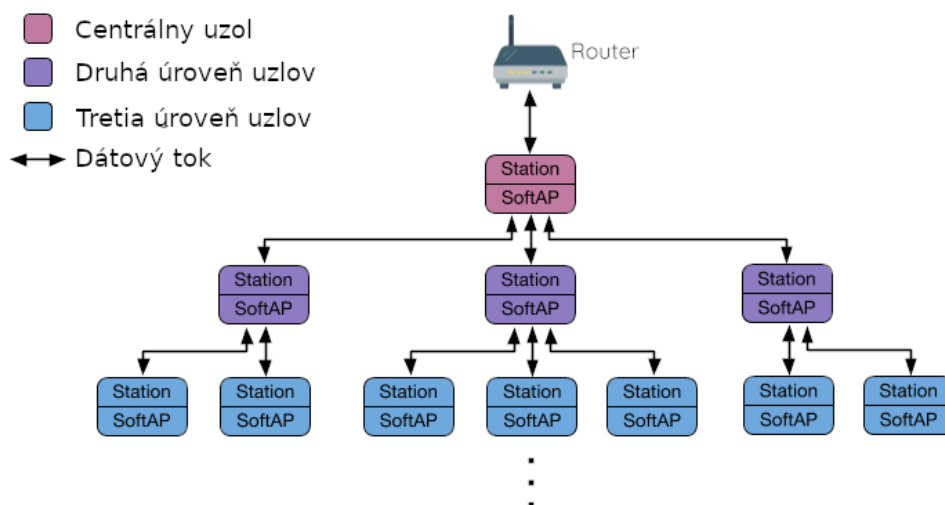
Obr. 11: Architektúra siete založená na technológii ESP-MESH [12]

Sieť založená na technológii ESP-MESH vyobrazená na Obr. 11 sa od tradičnej WiFi infraštruktúry siete líši tým, že nevyžaduje pripojenie všetkých uzlov k jednému centrálnemu uzlu. Na rozdiel od toho sa jednotlivé zariadenia pripájajú do siete prostredníctvom svojich susedov, ktorí zabezpečujú preposielanie a smerovanie dát v sieti. Týmto vylepšením je následne možné dosiahnuť väčšie plošné pokrytie dátovou sieťou, pretože nie je vyžadované priame pripojenie k centrálnemu prístupovému bodu. Táto úprava taktiež odstraňuje problém náchylnosti siete na preťaženie, pretože počet povolených uzlov už nie je obmedzený jedným spoločným uzlom, ako to bolo v prípade WiFi siete [12].

### 3.4.1 Topológia siete

Protokol ESP-MESH je rozšírením štandardného Wi-Fi protokolu, ktorý kombinuje viacero jednotlivých bezdrôtových sietí do jednej siete WLAN. Počet pripojení jedného uzlu k centrálnemu prístupovému bodu je limitované na jedno spojenie, zatiaľ čo prístupový bod môže byť súčasne pripojený k viacerým staniciam. ESP-MESH prostredníctvom rozhrania označovaného ako *softAP* – softvérový prístupový bod - umožňuje zariadeniam simultánne pracovať ako stanica aj prístupový bod. Preto môže mať viac pripojení po prúde (downlink), pričom zostáva obmedzenie jedného pripojenia proti smeru toku pomocou svojho rozhrania

stanice(uplink). Všetky tieto faktory vedú k usporiadaniu siete do stromovej hierarchie(Obr. 12) rodič – dieťa, ktorá je rozdelená do viacerých vrstiev [12].

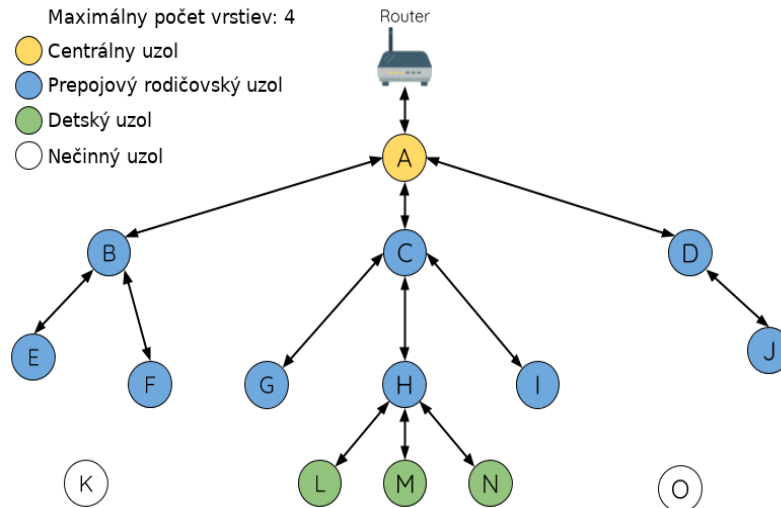


Obr. 12: Stromová topológia siete ESP-MESH [12]

ESP-MESH je špecifikovaná ako multi-hop sieť, čo znamená, že uzly prenášajú pakety do iných uzlov v sieti prostredníctvom jedného alebo viacerých bezdrôtových smerovaní a jednotlivé zariadenia preto nevysielať len svoje vlastné pakety, ale súčasne smerujú dátovú komunikáciu pre ďalšie uzly. Za predpokladu, že existuje cesta medzi akýmikoľvek dvoma uzlami vo fyzickej vrstve prostredníctvom jedného alebo viacerých bezdrôtových spojení, môže prebiehať komunikácia medzi ktorýmkoľvek párom uzlov v ESP-MESH sieti [12].

### 3.4.2 Druhy uzlov v sieti

Topológia vyobrazená na Obr. 13 reprezentuje príklad bežného usporiadania jednotlivých uzlov v sieti rozložených na veľkej fyzickej rozlohe, pričom nie je možné zabezpečiť dosah signálu všetkých prepojitových, detských a nečinnných uzlov(B-O) siete k centrálnemu uzlu(A) a routeru.



Obr. 13: Príklad topológie siete ESP-MESH [12]

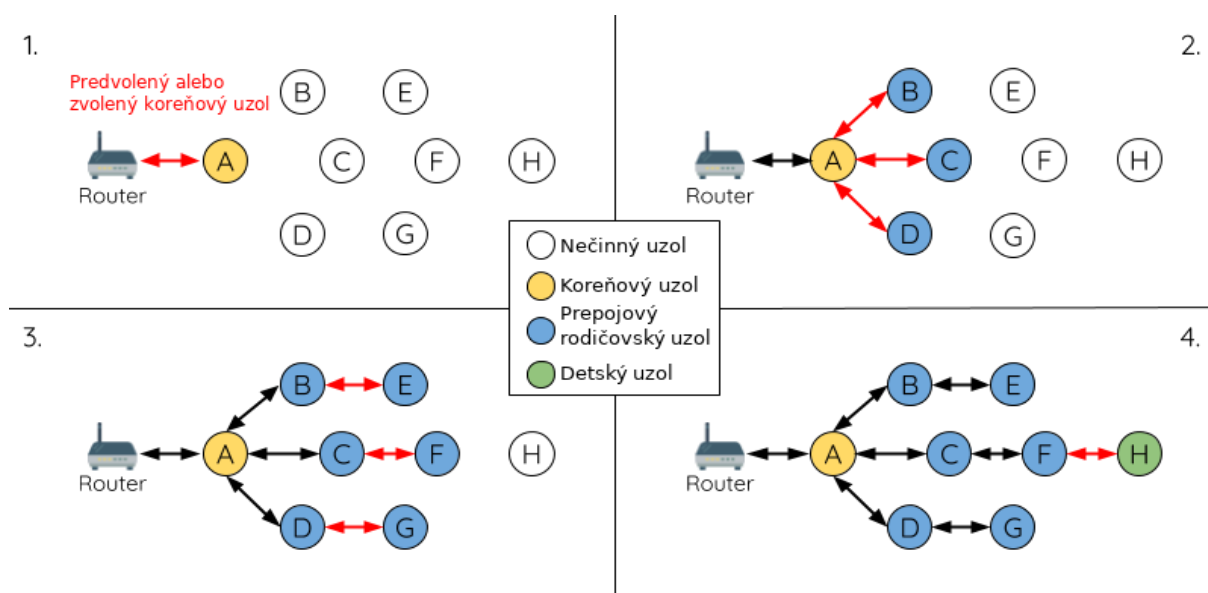
- **Centrálny uzol** predstavuje koreň stromovej topológie a je najvyšším uzlom v sieti. Úlohou tohto zariadenia je sprostredkovať rozhranie medzi ESP-MESH a externou IP sieťou. Koreňový uzol je pripojený k štandardnému Wi-Fi smerovaču, pomocou ktorého prenáša pakety medzi externou sieťou a jednotlivými uzlami v sieti ESP-MESH. Protokol stanovuje pravidlo, že každá takáto sieť môže mať iba jeden koreňový uzol, ktorého pripojenie proti prúdu(uplink) je vždy iba so smerovačom [12].
- **Detské uzly** (listy stromu) nemôžu mať žiadne podriadené uzly (žiadne downstream pripojenia). Takéto zariadenie prijíma a odosiela len dátové pakety, ktoré sú jemu určené. V prípade, ak sa uzol nachádza v maximálnej povolenej hĺbke, je klasifikovaný ako list stromu, čím je zabezpečené obmedzenie pridávania nových vrstiev do siete. Zariadenia, ktoré nie sú vybavené softvérovým prístupovým bodom sú taktiež zaradené ako detské uzly, pretože nedokážu vytvárať nadväzujúce spojenia dole prúdom(downstream). Na Obr. 13 sú zariadenia označené písmenami L, M a N v maximálnej hĺbke topológie, a preto boli algoritmom klasifikované ako listy stromu [12].
- **Prepojové rodičovské uzly** predstavujú strednú vrstvu topológie, kedy nie sú ani koreňovým uzlom, ani uzlom listovým. Sprostredkovateľský nadradený uzol musí mať práve jedno predradené pripojenie(upstream), ale môže mať viac podradených pripojení(downstream). Podradené pripojenie nie je pre funkčnosť siete vyžadované, a preto nemusí byť nadviazaná žiadna komunikácia s podriadeným zariadením. Z toho vyplýva, že prepojové uzly môžu vysilať a prijímať dátové pakety, ktoré sú im určené, ale tiež zabezpečujú preposielanie paketov zo svojho pripojenia proti prúdu(upstream) podradeným zariadeniam po prúde(downstream). Zariadenia reprezentované písmenami B až J na Obr. 13 predstavujú prepojové rodičovské uzly. Uzly E, F, G, I a J nie sú klasifikované ako listové uzly, pretože sa nenachádzajú v

maximálnej hĺbke stromu a v priebehu času majú možnosť nadviazať spojenie s iným zariadením, ktoré pripoja do siete [12].

- **Nečinnné uzly** sú zariadenia, ktoré nie sú v danom momente pripojené k sieti. Takéto uzly sa za správnych podmienok pokúsia nadviazať spojenie k nadradenému uzlu (rodičovi) alebo sa stanú novým koreňovým uzlom inej siete [12].

### 3.4.3 Proces vytvorenia siete

Prvým krokom budovania siete je voľba koreňového uzla, po ktorom nasleduje vytvorenie nadväzujúcich spojení po vrstvách, pokiaľ nie sú všetky zariadenia pripojené do siete. Presné rozloženie siete závisí od viacerých faktorov, ako je výber koreňového uzla, voľba rodičovských uzlov a asynchrónny reštart zariadení pri ich zapnutí. Tento proces je však možné zovšeobecniť do nasledujúcich krokov [12]:



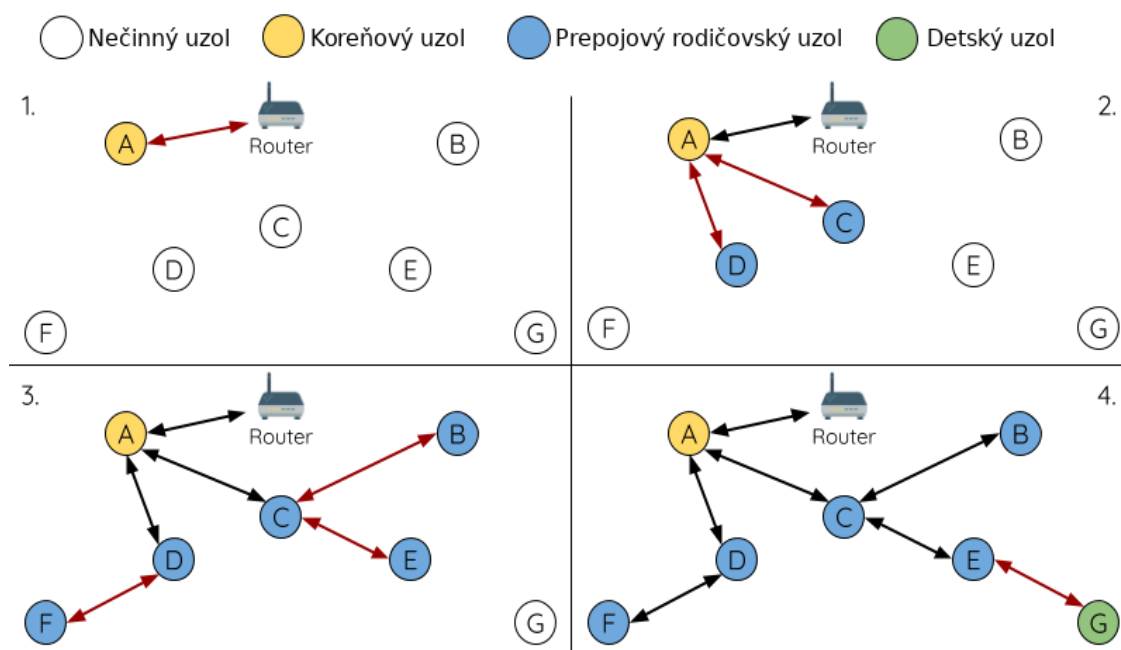
Obr. 14: Proces vytvorenia ESP-MESH siete [12]

1. **Výber koreňového uzla** je možné manuálne ovplyvniť v konfigurácii siete, alebo môže byť dynamicky zvolený na základe intenzity signálu medzi každým uzlom a smerovačom. Po ukončení výberu koreňového uzla nastáva prepojenie so smerovačom a následne začne pripájať nadväzujúce spojenia. Na Obr. 14 je možné vidieť, že uzol s označením A bol zvolený za koreňový uzol siete [12].
2. **Vytvorenie druhej vrstvy stromu** sa uskutoční po úspešnom výbere koreňového uzla, pričom všetky nečinnné uzly v jeho dosahu s ním začnú nadväzovať spojenie, čím je vytvorená druhá vrstva siete. Úspešným pripojením sa z uzlov stáva prepojový rodičovský uzol za predpokladu, že maximálna hĺbka topológie je vyššia ako 2. V príklade reprezentovanom na Obr. 14 môžeme vidieť uzly B a D v dosahu koreňového uzla, preto vytvoria nadväzujúce spojenie s koreňovým uzlom a stávajú sa prepojovými rodičovskými uzlami [12].

3. **Vytvorenie ostatných vrstiev siete** prebieha, keď zostávajúce nečinné uzly nadviažu spojenie s prepojovými uzlami v dosahu signálu. Po úspešnom nadviazaní spojenia s nadradeným uzlom môžu nastať dve situácie; bude klasifikovaný ako prepojový rodičovský uzol, alebo v prípade dosiahnutia maximálnej hĺbky siete je zaradený ako detský(listový) uzol [12].
4. **Limitujúci faktor siete** je stanovený ako hĺbka stromu, čím je zabezpečená ochrana siete pred prekročením maximálneho počtu povolených vrstiev. Pripojený uzol nachádzajúci sa na poslednej vrstve siete je automaticky klasifikovaný ako detský(listový) uzol, čím je zabezpečené ukončenie procesu tvorby siete [12].

### 3.4.4 Voľba rodičovského uzlu

V prípade projektu inteligentného inventarizačného systému je vhodné využitie možnosti manuálnej voľby koreňového uzlu v sieti. Toto rozhodnutie vyplýva z deterministického stacionárneho rozloženia jednotlivých zariadení, a preto je možné zvoliť uzol, ktorý sa nachádza najbližšie k smerovaču.



Obr. 15: Príklad formovania siete v prípade predvoleného koreňového uzlu [12]

Pri využití tohto módu siete je potrebné vypnúť systém voľby koreňového uzla. Všetky ostatné zariadenia v sieti sa musia vzdať možnosti stať sa riadiacim uzlom, aby sa predišlo výskytu konfliktov. Schéma na Obr. 15 reprezentuje zjednodušený proces vytvorenia ESP-MESH siete v prípade ak bol manuálne zvolený koreňový uzol [12].

1. Uzol A predstavuje nami vybraný riadiaci uzol a z toho dôvodu prebieha priame pripojenie k smerovaču. Všetkým ostatným zariadeniam je odobratá možnosť voľby primárneho uzlu [12].

2. Uzly C a D sa pripájajú k uzlu A, ako ich preferovaná voľba nadradeného zariadenia. Obe zariadenia tak vytvorili druhú vrstvu siete [12].
3. Obdobným spôsobom sa uzly B a E spoja s uzlom C a zariadenie označené písmenom F zvolilo uzol D ako svoje preferované nadradené zariadenie. Uzly B, E a F v tom momente vytvorili tretiu vrstvu v sieti [12].
4. V poslednom kroku je uzol G priradený do siete k nadradenému zariadeniu s označením E. Na rozdiel od predošlých prípadov je zariadenie označené ako detské(lisové), a to z dôvodu dosiahnutia maximálnej hĺbky siete stanovenej na 4 [12].

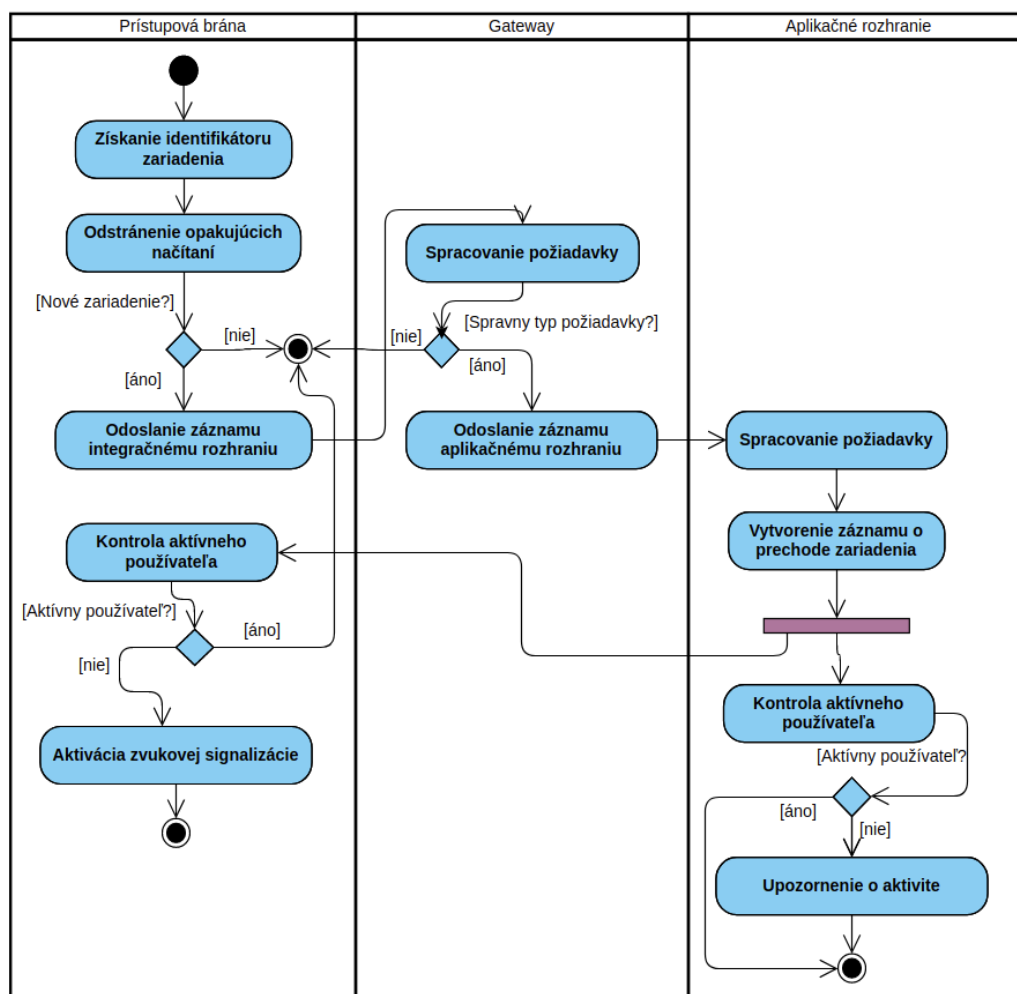
### **3.4.5 Zamedzenie vytvoreniu slučiek a ich detekcia**

V samo organizujúcich sieťach môže vzniknúť situácia, kedy konkrétny uzol vytvorí pripojenie proti smeru toku dát(upstream) s jedným z jeho potomkov(zariadenie v podsieti daného uzla). Výsledkom tejto operácie je vytvorenie kruhovej spojovacej cesty, čím sa poruší topológia stromu. ESP-MESH zabraňuje tvorbe spätného toku už počas výberu rodiča(nadradeného uzlu) vylúčením zariadení, ktoré sú už zaradené v smerovacej tabuľke výberového uzla, čím je zabránené v pokuse o pripojenie k akémukoľvek podradenému uzlu v sieti [12].

Pre prípad, že nastane vytvorenie slučky v sieti, existujú v systéme mechanizmy na verifikáciu prenosovej cesty. Rodičovský uzol zodpovedný za dopredné pripojenie, ktoré spôsobilo slučku informuje podradený uzol a iniciuje zastavenie spojenia [12].

## **3.5 Monitorovanie mobility hmotného majetku**

Úspešnou implementáciou rozhrania zabezpečujúceho získavanie identifikátorov jednotlivých zariadení a komunikačnej siete poskytujúcej konektivitu medzi zariadením a aplikačným programovým rozhraním je možné plnohodnotne špecifikovať proces a vytvoriť systém zaznamenávania pohybu. Z funkcionálnych požiadaviek definovaných v kapitole 2.2 vyplýva obmedzenie algoritmu na relatívne sledovanie polohy, ktoré predstavuje špecifické výzvy. Taktiež je potrebné dbať na citlivosť vstupných zariadení, pretože môže dochádzať k skresleniu údajov vo forme viacnásobného načítania rovnakej karty pri jednom prechode.



Obr. 16: Diagram aktivít popisujúci proces sledovania hmotného majetku

Zjednodušenú špecifikáciu procesu je možné vidieť na Obr. 16. Prvým krokom je získanie identifikátora zariadenia z hardvérovej čítačky umiestnenej pri vchode do pozorovanej miestnosti. Počas vykonávania úvodných testov jednotlivých hardvérových komponentov sme vypožarovali opakovanú registráciu karty zariadenia počas jedného prechodu bránou. Takéto správanie by mohlo predstavovať problém pri spracovaní údajov v aplikačnom prostredí a to z dôvodu, že sledovanie prebieha v relatívnom režime. Z toho vyplynula nutnosť kalkulácie smeru pohybu pri prechode vzhľadom k poslednému známemu transferu zariadenia. Ak by nebola implementovaná správna filtrácia nežiadaných načítaní, dochádzalo by k „oscilácií“ umiestnenia zariadenia v rámci jednej miestnosti, čo by malo za následok narušenie integrity zbieraných dát. V prvom priblížení projektu sme navrhli zavedenie pomocného stavu registrujúceho identifikátor a čas prechodu zariadenia čítačkou. Po úspešnom načítaní platného kódu je v prvom kroku vykonaná kontrola zhody aktuálneho a predošlého identifikátora. V prípade pozitívnej zhody je následne vypočítaná dĺžka intervalu od posledného prechodu. Porovnaním získanej hodnoty s konfigurovateľnou prahovou

hodnotou dokážeme zabezpečiť separáciu nového zariadenia od opakovaného načítania rovnakej karty. Exaktná hodnota tohto intervalu je závislá od použitého čítacieho zariadenia a jej určenie vyžaduje odladenie v produkčnom prostredí na konkrétnom hardvéri. V prípade negatívnej zhody je identifikátor zaradený do zásobníka pre ďalšie spracovanie.

Odosielanie prijatých a spracovaných kódov zariadení do aplikačného programového prostredia prebieha v asynchrónnej úlohe postupným získavaním hodnôt zo zásobníka. Komunikácia je zabezpečená sieťou ESP-MESH bližšie špecifikovanou v kapitole 3.4. V prípade úspešného odoslania je identifikátor zaradený do separátneho zoznamu určeného pre aktuálne spracovávané prechody, ktoré neboli potvrdené aplikačným rozhraním. Funkcia tohto zoznamu je zabezpečiť istú formu redundancie pre prípad, že počas prenosu alebo spracovania požiadavky nastala chyba a server nepotvrdil túto transakciu v určenom časovom intervale. Po prekročení tohto intervalu je záznam opätovne odosielaný na server. Identifikátory zariadení čakajúce v tejto fronte majú stanovenú vyššiu prioritu pri retransmisii, aby bola dodržaná sekvencia transakcií.

Po prijatí potvrdenia prechodu zariadenia je konkrétny záznam odstránený zo zoznamu nepotvrdených transakcií. Obsah odpovede je následne spracovaný kvôli kontrole aktívneho používateľa priradeného k aktuálnej miestnosti. V prípade negatívneho výsledku je aktivované upozornenie na strane prechodovej brány vo forme zvukovej signalizácie, ktorá slúži na upozornenie používateľa, aby vykonal autentifikáciu.

### **3.6 Autentifikácia používateľa**

Pre zabezpečenie správnej funkčnosti celého systému je potrebné zabezpečiť odosielanie autorizačných dát aplikačnej platforme. Proces odosielania údajov prebieha obdobným spôsobom ako je to definované v prípade registrácie pohybu hmotného majetku z kapitoly 3.5. Špecifikom v tomto procese ale je fakt, že v ňom nefiguruje ochrana pred mnohonásobným načítaním prístupovej karty, pretože táto funkcionality je v štandardnej výbave samotných čítačiek. Ide predovšetkým o zariadenia s krátkym dosahom, a tak je možné zabezpečiť jedinečnú detekciu priloženia a následného odobratia karty.



## 4 Návrh webového rozhrania

Kapitola sa zaoberá vytvorením interaktívneho webového rozhrania, pomocou ktorého je možná konfigurácia systému. Cieľom je priblíženie kľúčovej funkcionality zabezpečujúcej monitorovanie majetku.

### 4.1 Špecifikácia použitých technológií

Počas vývoja webovej aplikácie sme sa zamerali na použitie najnovších dostupných technológií umožňujúcich vytvorenie takzvaných jednostránkových aplikácií, ktoré poskytujú niekoľko kľúčových výhod. Hlavným benefitom tohto prístupu je forma, ktorou daný web pracuje v klientskom prehliadači. Jednostránkové aplikácie nevyžadujú opätovné načítanie všetkých zdrojov v prípade vykonania akcie. Z pohľadu UX sa snažia napodobniť prirodzené uniformné prostredie, na ktoré sú používatelia zvyknutý z moderných mobilných aplikácií.

Požadované správanie je dosiahnuté izoláciou prezentačnej vrstvy softvéru od serverom spracovávaných požiadaviek na strane aplikačného programového rozhrania. Takýmto rozdelením zdrojových kódov jednotlivých projektov je dosiahnutá nezávislosť vývoja oboch komponentov, ktoré medzi sebou zdieľajú iba schému poskytovaných dát.

Využitím jednostránkových aplikácií dokážeme zabezpečiť efektívne využitie vyrovnávacej pamäte prehliadača, ktorý dokáže všetky potrebné zdroje perzistovať po prvom použití, čím je zabezpečená úspora prenášaných údajov. Toto môže byť kritickým faktorom v prípade prezerania na mobilných zariadeniach využívajúcich dátovú sieť poskytovateľa služieb.

Vývoj prebiehal využitím JavaScript-ovej knižnice ReactJS. Napriek tomu, že existuje veľa druhov platforiem umožňujúcich tvorbu front-end webových aplikácií, rozhodli sme sa využiť React. Hlavným dôvodom je podpora vývoja v štandardnom formáte JavaScript kódu, čím je zabezpečený rýchly vývoj bez nutnosti zdĺhavého procesu štúdia novej štruktúry zápisu. Zapúzdrenie jednotlivých komponentov aplikácie je dosiahnuté využitím syntaxe s označením JSX, ktorá kombinuje zápis HTML s JavaScript-om v jednom súbore.

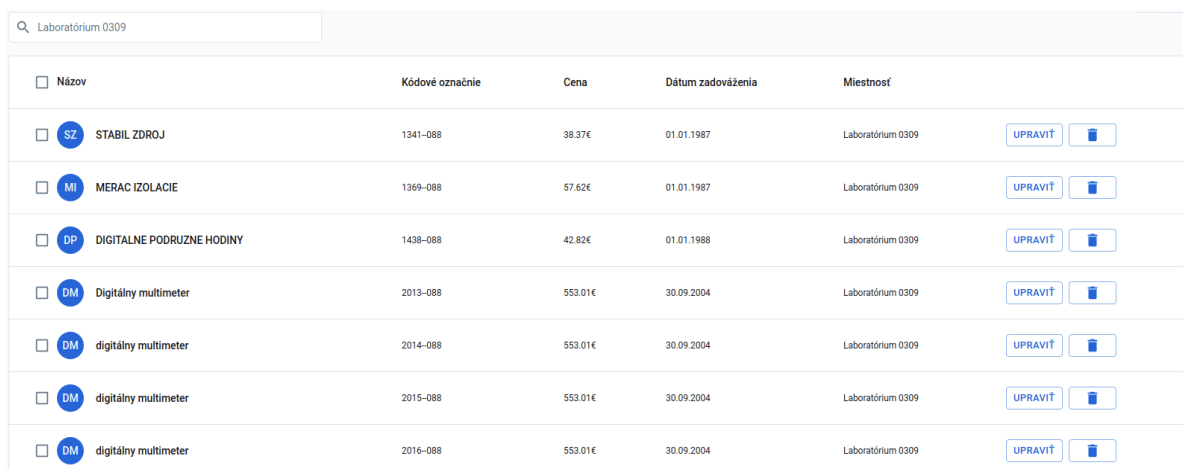
Jadro zabezpečujúce vykresľovanie využíva jednosmerný tok dát založený na báze architektúry Flux zabezpečujúci riadenie smerovania údajov do cieľových komponentov cez spoločný kontrolný dispečingový mechanizmus. Využitie takéhoto prístupu uľahčuje ladiaci proces aplikácie, a tak dokážeme jednoduchšie zabezpečiť integritu dát celej aplikácie.

## 4.2 Priblíženie vytvorených grafických komponentov

Implementácia webového rozhrania systému je založená na úrovni zdieľaných funkcionálnych komponentov. Nasledovné kapitoly špecifikujú vytvorené modulárne prvky, ktoré zabezpečujú hlavnú funkcionálnu monitorovania majetku. Správa ostatných systémových zdrojov, ako sú používatelia, miestnosti a prístupové karty, zdieľajú popisovanú metodológiu a z toho dôvodu nie sú bližšie popisované.

### 4.2.1 Zoznam systémových zdrojov

Základným navigačným komponentom v systéme je zobrazenie zoznamu všetkých dostupných položiek konkrétneho typu.



<input type="checkbox"/> Názov	Kódové označenie	Cena	Dátum zadávania	Miestnosť	
<input type="checkbox"/> SZ STABIL ZDROJ	1341-088	38.37€	01.01.1987	Laboratórium 0309	<input type="button" value="UPRAVIŤ"/>
<input type="checkbox"/> MI MERAC IZOLACIE	1369-088	57.62€	01.01.1987	Laboratórium 0309	<input type="button" value="UPRAVIŤ"/>
<input type="checkbox"/> DP DIGITÁLNE PODRUŽNE HODINY	1438-088	42.82€	01.01.1988	Laboratórium 0309	<input type="button" value="UPRAVIŤ"/>
<input type="checkbox"/> DM Digitálny multimeter	2013-088	553.01€	30.09.2004	Laboratórium 0309	<input type="button" value="UPRAVIŤ"/>
<input type="checkbox"/> DM digitálny multimeter	2014-088	553.01€	30.09.2004	Laboratórium 0309	<input type="button" value="UPRAVIŤ"/>
<input type="checkbox"/> DM digitálny multimeter	2015-088	553.01€	30.09.2004	Laboratórium 0309	<input type="button" value="UPRAVIŤ"/>
<input type="checkbox"/> DM digitálny multimeter	2016-088	553.01€	30.09.2004	Laboratórium 0309	<input type="button" value="UPRAVIŤ"/>

Obr. 17: Zoznam dostupných položiek hmotného majetku

Ukázková tabuľka na Obr. 17 vyobrazuje zoznam aktuálne dostupných položiek hmotného majetku pozorovaného subjektu. Najväčší vizuálny priestor obrazovky zaberajú dáta prislúchajúce jednotlivým položkám hmotného majetku, podľa ktorých sa môžeme ľahšie rozhodnúť, nad ktorým prvkom chcem vykonať požadovanú akciu.

Pre zabezpečenie optimálnej práce nad veľkým počtom záznamov disponuje zobrazenie funkciou vyhľadávania. Toto predstavuje výhodnú voľbu v prípadoch, kedy používateľ disponuje znalosťou o vyhľadávanej položke, vo forme jej názvu, kódového umiestnenia, prípadne miestnosti, v ktorej by sa malo nachádzať. Implementačná časť vyhľadávania prebieha v aplikačnom programovom rozhraní, čím dosiahneme minimalizáciu dát spracúvaných v klientskej aplikácii. Delegovanie tejto úlohy serverovej časti projektu umožňuje budúce rozšírenie o techniky fulltextového vyhľadávania poskytujúce presnejšie výsledky.

Pravá strana obrazovky disponuje akčnými tlačidlami, vykonávajúcimi požadované operácie nad zvoleným záznamom. Taktiež poskytuje možnosť explicitného definovania akcie pre prípad, že jej optimalizovaná forma nie je používateľovi zrejmá. Ako ukážka môže slúžiť operácia *Upraviť*, ktorú je možné vyvolať kliknutím na názov samotného záznamu. Z dizajnového hľadiska nebolo použité podčiarknutie textu, ktoré býva jasným indikátorom odkazu, a preto je takúto akciu jednoduché prehliadnuť.

#### 4.2.2 Detail a vytvorenie systémového zdroja

Operácie spravujúce systémové zdroje vyžadujú vytvorenie vstupného formuláru, pomocou ktorého môže používateľ zadávať údaje v prehľadným spôsobom .

Informácie o zariadení	
Názov zariadenia * STABIL ZDROJ	Kódové označenie * 1339-088
Zostávajúca cena 38.37	Dátum zaobstarania 01.01.1987
<button>ULOŽIŤ</button>	

Obr. 18: Formulár detailu systémovej položky

Jednoduchá forma obdobnej obrazovka je reprezentovaná na Obr. 18. Poskytuje možnosť zadávania jednotlivých údajov v štruktúrovanej forme s jasným označením o aký údaj ide. Zobrazenie môže poskytovať dodatočné údaje vo forme validačných hlásení pre prípad nevyplnenia povinných polí alebo zlého formátu samotného údaj. Pre zabezpečenie pohodlnejšieho zadávania štruktúrovaných údajov, ako je dátum, obrazovka poskytuje dialógové okno s interaktívnym kalendárom.

### 4.2.3 Priradenie jednotnej systémovej položky

Počas návrhu systému boli databázové údaje rozdelené do samostatných celkov, ktoré zdieľajú konkrétny cieľ pri práci systému. Napriek tomu, že sa údaje nenachádzajú v jednom dátovom celku, je potrebné zabezpečiť konfigurovateľnosť vzťahov jednotlivých položiek daných systémových zdrojov.

The figure consists of four screenshots of a web form titled "Informácie o miestnosti". Each screenshot shows a form with two input fields: "Názov miestnosti" and "Kód miestnosti". The "Názov miestnosti" field is a dropdown menu, and the "Kód miestnosti" field is a text input. The form also includes a blue "UPRAVIŤ" button and two purple buttons: "ULOŽIŤ" and "ZRUŠIŤ".

- Top-left screenshot:** The "Názov miestnosti" dropdown is set to "Laboratórium 0309". The "Kód miestnosti" field contains "0309". The "UPRAVIŤ" button is visible.
- Top-right screenshot:** The "Názov miestnosti" dropdown is set to "Laboratórium 0309". The "Kód miestnosti" field is empty. The "ULOŽIŤ" and "ZRUŠIŤ" buttons are visible.
- Bottom-left screenshot:** The "Názov miestnosti" dropdown is set to "0309". A dropdown menu is open, showing two options: "03092 (03092)" and "Laboratórium 0309 (0309)". The "Kód miestnosti" field is empty. The "UPRAVIŤ" button is visible.
- Bottom-right screenshot:** The "Názov miestnosti" dropdown is set to "Laboratórium 0309 (0309)". The "Kód miestnosti" field is empty. The "ULOŽIŤ" and "ZRUŠIŤ" buttons are visible.

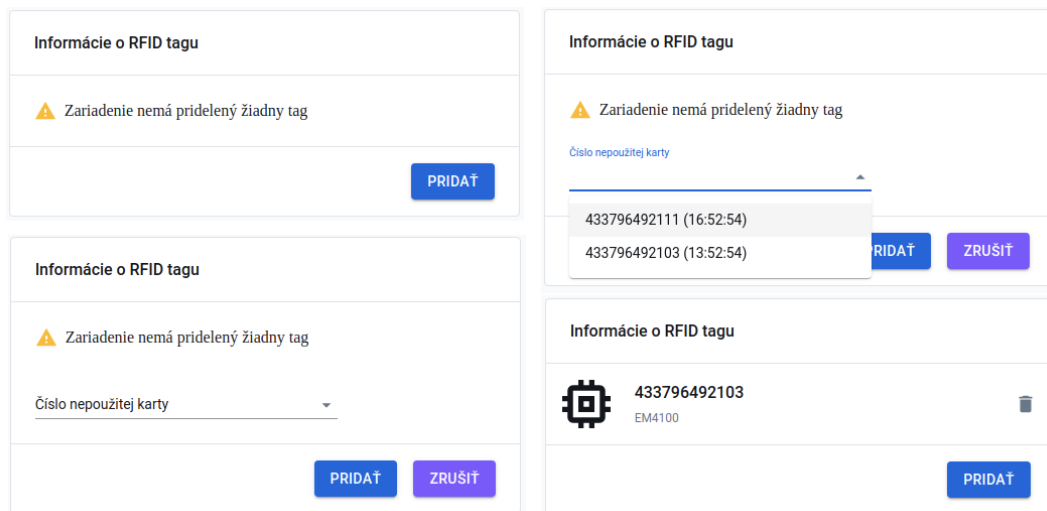
Obr. 19: Stavy komponentu pre výber jednej položky

Ako príklad môže slúžiť formulár na Obr. 19, vykonávajúci priradenie hmotného majetku ku konkrétnej miestnosti, v ktorej sa nachádza. Ide o vzťah 1:1, a preto je používateľovi umožnený výber jednej konkrétnej položky, ktorá uložením nahradí aktuálnu.

Pre zabezpečenie pohodlnej správy je formulár vybavený vyhľadávacím prvkom vo forme vyberača hodnôt. Takto je zabezpečená integrita zadávaných údajov, pretože používateľ musí vybrať už existujúcu hodnotu zo systému, pričom nie je nutnosť, aby poznal presný názov konkrétnej položky.

#### 4.2.4 Komponent umožňujúci výber viacerých položiek

Niektoré systémové zdroje ponúkajú možnosť priradenia viacerých položiek k jednému systémovému záznamu. Pre vykonanie tejto úlohy preto nepostačuje použitie komponentu z kapitoly 4.2.3, ktorý je obmedzený na jednu položku.



Obr. 20: Ukážka stavov komponentu s výberom viacerých prvkov

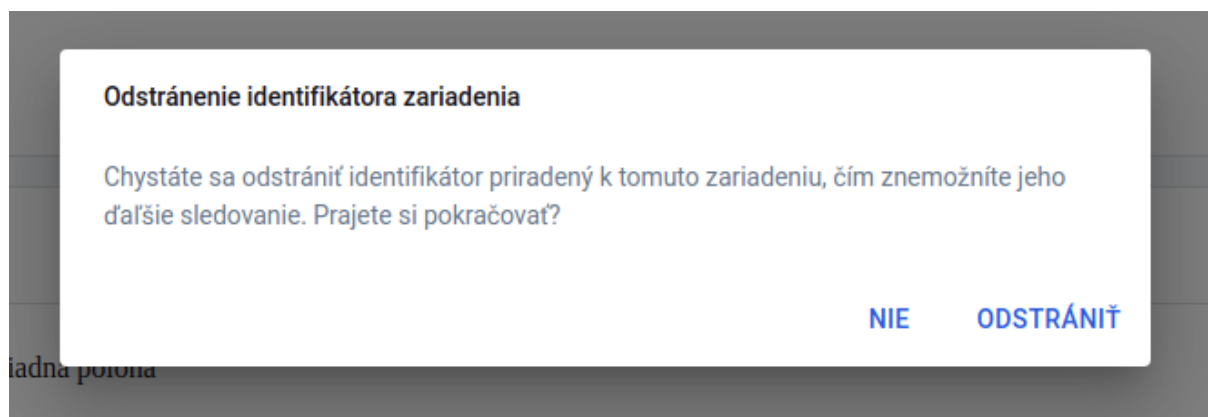
Priradenie štítkov k jednotlivým zariadeniam(Obr. 20) môže slúžiť ako príklad vzťahu označovaného 1:N, ktorý reprezentuje fakt, že jedno zariadenie môže mať priradených viacero štítkov, ale každý štítok musí reprezentovať iba jedno konkrétne zariadenie.

V prípade priradenia miestností bolo možné využitie familiárneho názvu pre systémový zdroj, aby užívateľ vedel pohodne zvoliť položku, ktorú vyhľadával. Štítky disponujú identifikačným číslom, ktoré nemusí byť vyobrazené na viditeľnom mieste v rôznych vyhotoveniach finálneho produktu. Aby bolo možné zabezpečiť správne priradenie, jednotlivé položky disponujú časovým údajom, popisujúcim čas prvotnej registrácie. Tento údaj má slúžiť ako dodatočný faktor rozhodovania.

Zobrazovanie priradených položiek je realizované vo forme riadkových záznamov disponujúcich popisnými údajmi o konkrétnej položke s rozšírením o akciu jej odstránenia, ktorá sa nachádza na pravej strane zobrazenia. Takto je zabezpečená plnohodnotná správa štítkov aj pre prípad, kedy musí byť vyradený z prevádzky z dôvodu poškodenia alebo straty.

#### 4.2.5 Dialóg potvrdenia akcií

Systém disponuje operáciami nad dátami s rôznou závažnosťou vykonanej zmeny. Úpravy predstavujúce zásadnú zmenu systému by mohli neodborným alebo náhodným vyvolaním spôsobiť nežiaduce straty údajov alebo integrity systému.



Obr. 21: Ukážka dialógu potvrdenia akcií vykonávajúcu zásadnú zmenu systému

Operácie poskytujúce zásadné alebo nezvratné zmeny systému disponujú vo webovom rozhraní dialógovým oknom vyobrazeným na Obr. 21, ktoré popisuje závažnosť konkrétnej operácie. Zabezpečením vizuálneho medzikroku dokážeme zabezpečiť v používateľovi pocit istoty, že nevykoná náhodnú operáciu, ktorá spôsobí problém. Taktiež zlepšuje využiteľnosť v mobilnej verzii, kde je riziko náhodného dotyku vyššie.

#### 4.2.6 Komponent zobrazenia stavových hlásení

Štruktúrované dáta sú vhodné predovšetkým pri softvérovom spracovaní a užívateľom systému by mohli sťažovať prehľadnosť a rýchlu reakciu.

Pozícia zariadenia
 K zariadeniu nie je aktuálne registrovaná žiadna poloha
 Zariadenie sa nenachádza v žiadnej miestnosti. Prevzaté používateľom Viktor Chovanec z miestnosti Laboratórium 0399
 Zariadenie sa nachádza v miestnosti Laboratórium 0399
 Zariadenie je aktuálne vypožičané používateľom Viktor Chovanec

Obr. 22: Ukážka stavových informácií systému

Z tohto dôvodu je potrebné formátovanie systémových dát v čitateľnej a porozumiteľnej forme pre užívateľa aplikácie. Hlásenia na Obr. 22 predstavujú ukážku reprezentácie takýchto dát. Zobrazenie je vizuálne rozdelené na dve časti; dôležitosť upozornenia vo forme farebnej ikony a textovou hláškou popisujúcou udalosť.

Rozšírenie jednotlivých záznamov pomocou piktogramov prispieva k rýchlejšiemu rozlíšeniu priority pre používateľa. Vhodnou voľbou dokážeme zabezpečiť okamžitú reakciu na dôležitosť obsahu. K zvýrazneniu prispieva využitie farebnej škály umocňujúcej prioritu, ale pre zabezpečenie dostupnosti pre používateľov trpiacim zrakovým postihnutím musí obsah piktogramu dostatočne popisovať daný stav.

### 4.2.7 Priradenie systémových zdrojov k importovateľným položkám

Počas analýzy systému bola špecifikovaná požiadavka na zabezpečenie importu dát aktuálne používaného hmotného majetku z dát v tabuľkovom procesore.

Názov stĺpca	Stĺpec súboru	Akcia
Name (string)	Názov majetku	
Code (string)	Inventárne číslo	
Price (float32)	Zost. cena účt.	
AcquiredAt (time.Time)	Dátum zar.	
RoomID (int)	Miest.	

ZATVORIŤ IMPORTOVAŤ

Obr. 23: Formulár priradenia systémových zdrojov k stĺpcom importovaného súboru

V prvom zameraní nie je tento proces plne automatizovaný a z toho dôvodu je potrebné zabezpečiť proces mapovania manuálnym spracovaním používateľom. Vstupné údaje nie sú v plnej forme kompatibilné s nami očakávaným vstupom, a preto je nutné vykonanie ručného priradenia jednotlivých stĺpcov záznamu k systémovým zdrojom.

Pre vykonanie tejto úlohy bolo implementované dialógové okno s formulárom vyobrazenom na Obr. 23. Zobrazenie je vizuálne rozdelené na tri stĺpce:

- **Názov stĺpca** reprezentuje názov nami systémom navrhnutých dátových štruktúr. Pre lepšie rozhodovanie je poskytovaný dátový typ reprezentujúci položku.
- **Stĺpec súboru** poskytuje zoznam s vyberateľnými položkami, ktoré reprezentujú hodnoty prvého riadka vstupného súboru.
- **Akcia** poskytuje rozširujúce operácie nad vstupnými dátami pre zabezpečenie kompatibility s našim systémom.

V prvom priblížení ide skôr o technickú obrazovku, ktorá nie je pripravená na neodborné spracovanie. Vylepšením pre druhú fázu vývoja je zmena názvu systémových zdrojov za viac familiárne pomenovania, ktoré by používateľovi poskytli bližšiu predstavu o funkcii konkrétneho údaj. Druhým zlepšením môže byť pokus o automatické priradenie záznamov k



ich hodnote v spracovanom súbore na základe názvu stĺpca. Takto by bolo zabezpečenie zrýchlenie procesu v prípade nutnosti častého opakovania.

## 5 Návrh aplikačného programového rozhrania

Kapitola opisuje postupy využité pri vytvorení aplikačného programového rozhrania slúžiaceho ako ústredné miesto spracovania a zberu dát vyžadovaných systémom. Úvodná časť špecifikuje voľbu programovacieho jazyka spĺňajúceho funkcionálne a nefunkcionálne požiadavky definované v kapitole 2.2. Taktiež obsahuje popis implementovaných operácií a transformácií nad dátami.

### 5.1 Výber implementačného jazyka

Pred samotnou implementáciou všetkých funkcionálnych požiadaviek systému je potrebné precízne zváženie výberu programovacieho jazyka, ktorý bude použitý pri vývoji a to na základe definovaných nefunkcionálnych požiadaviek systému. Podmienka, ktorá ovplyvní výber je možnosť prevádzkovania systému na platformách s obmedzeným výpočtovým výkonom a špecificky prispôbeným operačným systémom nepodporujúci plné spektrum funkcionality ako v prípade PC platformy pracujúcej na architektúre x86 alebo AMD64.

Počas výberu sme zvažovali jazyk Java ako stabilnú implementačnú platformu overenú v praxi a pomerne vývojovo mladý jazyk Golang(Google Go), ktorý si pomaly buduje popularitu v IT svete. V procese sme porovnali výhody a nevýhody jednotlivých platforiem a ich možný dopad na implementáciu projektu inteligentného inventarizačného systému.

Prvou vlastnosťou umožňujúcou kategorizáciu je spôsob interpretácie kódu na cieľovej platforme. Do kategórie interpretovaných jazykov patrí napríklad nami zvažovaná Java alebo iné populárne jazyky ako Node.js či Python. Táto kategória predstavuje nutnosť využitia špecifického prekladača(interpreter) zdrojového kódu vybraného jazyka na strojový kód platformy pri každom spustení aplikácie. Takýmto prístupom dokážeme zabezpečiť nezávislosť zdrojového kódu od hardvérového riešenia, na ktorom bude spúšťaný. Nevýhodou takéhoto riešenia je fakt, že pre spustenie nášho kódu je potrebná extra softvérová vrstva vo forme aplikácie, ktorá môže predstavovať záťaž navyše pre systém, na ktorom beží. Ďalšou podstatnou nevýhodou môže byť rýchlostný a optimalizačný faktor kódu. Špecifikom interpretovaných jazykov je vykonávanie zdrojového kódu riadok po riadku, čo môže mať negatívny vplyv na výkon. V prípade moderných interpretovaných jazykov je využitá technológia kompilácie zdrojového kódu v reálnom čase(just in time), a tak je možné dosiahnuť určitú úroveň optimalizácie vykonávania kódu a zabezpečiť mitigáciu tohto problému. Druhou kategóriou sú kompilované jazyky ako Golang alebo C++. V tomto prípade je spúšťaná jednorazová operácia transformácie zdrojového kódu priamo do

strojového kódu požadovanej platformy. Nevýhodou takéhoto prístupu je, že formát výsledného binárneho súboru je viazaný na špecifickú platformu, pre ktorú bol vytvorený. Týmto faktorom môže byť obmedzená kompatibilita zdrojového kódu pre jednotlivé platformy, ktorá je ovplyvnená podporou platforiem použitého kompilátora a knižníc. Špecifickou výhodou jazyka Golang je statické pripojenie všetkých využitých knižníc počas kompilačného procesu, a tak je možné dosiahnuť jediný výstupný binárny súbor obsahujúci všetky vyžadované závislosti pre jeho spustenie na konkrétnej platforme. Touto optimalizáciou odpadá nutnosť inštalácie rozširujúcich softvérových balíčkov, ako je to v prípade využitia jazyka C++ a platformy .NET od spoločnosti Microsoft.

Oba zvažované jazyky používajú obdobnú syntax zdrojového kódu, ktorá vychádza zo spoločnej rodiny založenej na programovacom jazyku C. Z toho vyplýva možnosť jednoduchého pochopenia fungovania programu vytvoreného v jazyku Golang pre vývojára pracujúceho s jazykom Java. Hlavnou vizuálnou zmenou je vynechanie nutnosti ukončovania príkazov bodkočiarkou, ktorá je vyžadovaná len v špecifických prípadoch [13].

Golang a Java disponujú vstavanou funkcionalitou, ktorá zabezpečuje ochranu pred pretečením pamäte automatickou správou pridelenia a následného odstránenia v prípade jej nepoužitia, označovanej ako Garbage Collector. Toto predstavuje veľkú výhodu na rozdiel od jazykov z rovnakej rodiny jazykovej syntaxe ako napríklad C alebo C++. Funkcionalita predstavuje jednoduchšiu správu pamäte, keďže nie je vyžadovaná manuálna alokácia segmentu pamäte s konkrétnou veľkosťou a jej následné uvoľňovanie v prípade nevyužívania [13].

Najväčším rozdielom je fakt, že Golang nie je objektovo orientovaný programovací jazyk. Golang vo svojom princípe nepodporuje mechanizmus dedičnosti, pretože neimplementuje štandardný polymorfizmus. Toto sa prejaví vo forme výhradného použitia štruktúr, pričom chýba podpora pre objekty, ako je to v prípade Javy. Chýbajúca podpora objektov sa prejaví hlavne pri potrebe využitia štandardných návrhových vzorov určených pre objektové programovanie, kde je potrebné prispôbenie vo forme implementácie rozhraní(interfaces) pre štruktúry. Golang poskytuje možnosť zaradenia štruktúry do samej seba. Tento mechanizmus obmedzuje využitie hostiteľských metód a dát pre vnorené štruktúry. Dedičnosť je nahradená kompozíciou, čím dosiahneme požadovanú funkcionalitu pre metódy a dáta [13].

Jazyk Go poskytuje jednoduchý a elegantný spôsob dosiahnutia súbežného spracovania inštrukcii vo forme modelu nazvaného komunikácia sekvenčných procesov(CSP). Plánovač využíva model N ku M, ktorý umožňuje M súbežne vykonávaných funkcií v N systémových

vláknach. Spustenie rutiny predstavuje jednoduché volanie kľúčového slova jazyka *go* *názov\_funkcie()*. Komunikáciu medzi procesmi je možné zabezpečiť dvoma spôsobmi; využitím spoločného bloku pamäte, čo nepredstavuje vhodnú voľbu, alebo zaradením špecifických komunikačných kanálov určených práve pre tento prípad použitia [13].

Medzi spomínané nevýhody Golang-u patrí chýbajúci polymorfizmus. Toto môžeme reprezentovať štandardnou situáciou jedného balíka obsahujúceho dve funkcie s rozdielnym počtom vstupných parametrov vykonávajúce principiálne rovnakú funkčnosť. Pri implementácii sme nútení zvoliť dva rozdielne názvy funkcií, čo môže viesť k opakovaniu sekcií rovnakého kódu s neprehľadnými názvami [13].

Po zvážení vyššie uvedených dôvodov sme sa rozhodli zvoliť Golang ako preferovaný implementačný jazyk aplikačného programového rozhrania. Hlavným faktorom výberu bola podpora kompilácie aplikácie do jedného binárneho súboru s využitím statického linkovania. Touto cestou sa vyhneme možným problémom chýbajúcich knižníc na obmedzených systémoch bežiacich na platforme RaspberryPi a zabezpečíme tak kompatibilitu pre viacero podobných zariadení bez nutnosti zdĺhavej inštalácie doplňujúcich balíčkov. Žiadne limitujúce faktory jazyka Golang nepredstavujú problém, ktorý by mohol obmedziť vývoj systému a je možné ich prekonať prispôbením programovacieho štýlu počas implementácie.

## 5.2 Softvérové technológie a hierarchia projektu

Pre dosiahnutie optimálnej implementácie funkcionality aplikačného rozhrania je potrebné stanovenie hierarchie projektu, pomocou ktorej zabezpečíme prehľadné rozdelenie kódu a funkčných blokov pre celkové urýchlenie vývoju a zrozumiteľnosti projektu. Kapitola taktiež definuje použité softvérové rozšírenia poskytujúce nadstavbu funkčného balíka a optimalizáciou procesu vývoja a nasadenia systému. Projekt je zložený z nasledujúcich systémových rozhraní:

- **Aplikačné programové rozhranie** definujúce štruktúru URI odkazov jednotlivých služieb a implementujúce funkčnosť vykonávajúcu operácie nad príslušným systémovým zdrojom. Rozhranie je ďalej rozdelené na dva samostatné balíky. Prvý poskytuje dáta pre webové rozhranie zabezpečujúce správu a konfiguráciu systému. Druhý balík poskytuje integračnú platformu medzi aplikačným rozhraním a konektivitou bezdrôtovej senzorovej siete ESP-MESH. Taktiež zabezpečuje smerovanie požiadaviek pre jednotlivé prístupové body na základe prijatých dát.
- **Autorizačná a autentifikačná platforma** poskytujúca funkčnosť pre zabezpečenie overenia prístupujúcich používateľov do systému a overenie prístupových práv pre použitie jednotlivých systémových zdrojov. Služba zabezpečuje vydávanie a obnovu podpisu používateľa(token), ktorý obsahuje všetky potrebné identifikačné údaje o

prihlásenom v kryptograficky podpísanom formáte. Taktiež zabezpečuje integračnú funkcionálnu autorizáciu požiadaviek prichádzajúcich z aplikačného programového rozhrania.

- **Funkcionalita konzolového rozhrania** implementujúca funkcionálnu pre prevádzku a správu celého systému. Ide o vstupný bod pri štarte aplikácie a zabezpečuje prípravu všetkých systémových zdrojov. Aktuálna verzia implementuje dva produkčné príkazy; *serve* slúži pre štart serveru aplikačného programového rozhrania spracúvajúceho požiadavky z webovej služby a integračnej brány senzorovej siete. Druhým príkazom je *migrate* zabezpečujúci jednoduchú formu prípravy databázového systému a jeho dátovej štruktúry pre potreby aplikácie.
- **Objekty pre prístup k údajom(DAO) a migračné operácie** poskytujú určitú formu abstrakcie vykonávaných operácií nad systémovými dátami od použitého databázového systému. Týmto spôsobom dokážeme zabezpečiť nezávislosť nami implementovaného riešenia od konkrétneho systému databáz v prípade potreby zmeny v priebehu alebo po ukončení prvej fázy implementácie. Z dôvodu závislosti tohto modulu na použitej platforme taktiež obsahuje procesy prípravy daného databázového systému a dátovej schémy pre konzolovú funkcionálnu *migrate*.
- **Dátové modely systémových zdrojov** popisujúce všetky údajové polia, ich typy a rozširujúce vlastnosti. Modul implementuje rozšírenú funkcionálnu vo forme validácie údajov pred ich spracovaním v systéme alebo prípadným uložením do databázy. Validácia funkcionálna zabezpečuje dodržanie typu údajov v prípade špeciálnych formátov ako sú e-mailové adresy alebo telefónne čísla, ich povinnosť vyplnenia alebo minimálnu/maximálnu povolenú dĺžku.

### 5.2.1 Aplikačné programové rozhranie

Systémová časť zabezpečujúca funkcionálnu pre webové rozhranie. Jazyk Golang ponúka vo svojom balíku rozšírení modul pre server poskytujúci spracovanie HTTP požiadaviek operujúci na báze asynchrónnych rutín, čím je zabezpečené neblokujúce spracovanie viacerých konkurentných pripojení z klientskych aplikácií. Štandardnou súčasťou balíka je aj funkcionálna poskytujúca smerovanie jednotlivých URL zdrojov k príslušnej obslužnej metóde. Programové rozhranie smerovača disponuje dostatočnou funkcionálnou pre vytvorenie jednoduchých webových aplikácií. Veľkou nevýhodou tohto riešenia pre použitie v našom projekte je chýbajúca komplexná segmentácia a zoskupovanie ciest vytvárajúcich jednotný blok funkcionality systémového zdroja. Ako príklad môžeme uviesť jednoduchú obsluhu používateľov a zariadení v administrácii. Aplikácia využívajúca vstavaný modul by definovala nasledovné metódy:

```
http.HandleFunc("/admin/pouzivatelia", obsluhaPouzivatela)
http.HandleFunc("/admin/pristroje", obsluhaPristrojov)
```

Obr. 24: Ukážka implementácie spracovania požiadaviek vstavanou knižnicou

Metódy *obsluhaPouzivatela* a *obsluhaPristrojov* vyžadujú autentifikovaného používateľa s administrácnymi privilégiami pre povolenie prístupu. Táto požiadavka by znamenala implementáciu autentifikácie pre obe tieto metódy zvlášť, a tak by dochádzalo k opakovanému využitiu totožného kódu zabezpečujúceho overovanie. Takýto prístup sa pri implementácii väčšieho počtu obsluhových funkcií stáva ťažko udržiateľným a predstavuje riziko vzniku chýb. Z nášho pohľadu je druhou nevýhodou chýbajúca možnosť priradenia rôznych druhov HTTP požiadaviek (GET, POST, ...) k rozličným funkciám zabezpečujúcim spracovanie požiadavky. Z toho vyplýva požiadavka implementácie komplexného spracovania všetkých druhov operácií nad dátami v jednej funkcii, čo môže mať dopad na prehľadnosť výsledného kódu. Tieto problémy sme sa rozhodli adresovať použitím knižnice tretej strany s označením *chi*, ktorá opravuje všetky popísané problémy a disponuje rozšírenou funkcionalitou zabezpečujúcou pohodlnejšie spracovanie požiadaviek.

```
r.Route("/admin", func(r chi.Router){
    r.Use(AutorizaciaPouzivatela);
    r.Get("/pouzivatelia", obsluhaPouzivatela)
    r.Get("/pristroje", obsluhaPristrojov)
})
```

Obr. 25: Implementácia spracovania požiadaviek využitím knižnice *chi*

Využitie knižnice *chi* nám umožňuje kód z Obr. 24 optimalizovať do podoby reprezentovanej na Obr. 25. Týmto spôsobom je možné dosiahnutie prehľadnejšej segmentácie kódu spadajúceho pod spoločný systémový zdroj (admin). Úpravou taktiež dosiahneme jednotné aplikovanie spoločnej middleware funkcionality pre celú skupinu metód, reprezentované funkcionalitou *AutorizaciaPouzivatela* zabezpečujúca overenie povolení prístupujúceho klienta.

### 5.2.2 Objekty zabezpečujúce prístup k údajom a migračné operácie

Pre urýchlenie a zjednodušenie vývoja systému sme sa rozhodli využiť knižnicu poskytujúcu objekto-relačné mapovanie (ORM) databázových údajov so systémovými. Výhodou tohto systému je automatické generovanie dopytov zabezpečujúcich operáciu nad databázovým systémom, čím môžeme zabezpečiť izoláciu a nezávislosť nami vyvíjaného kódu od finálne použitého databázového riešenia. Taktiež môže obsahovať rôzne optimalizačné riešenia vo

forme automatickej kontroly stavu pripojenia k serveru a opakovanie neúspešných dopytov. Napriek tomu, že tento systém umožňuje automatické vytvorenie databázových tabuliek, toto rozšírenie sme sa rozhodli nevyužiť. Pre zabezpečenie doprednej kompatibility sme zaviedli systém verzionovanej migračnej politiky, pomocou ktorej je možné automatické sledovanie verzii databázovej schémy. Tento systém obsahuje presné postupy ako je možné transformovať dáta medzi jednotlivými verziami a to bez nutnosti explicitnej špecifikácie pri spustení.

### 5.2.3 Funkcionalita konzolového rozhrania

Výsledný systém aplikačného programového rozhrania je určený pre použitie v serverovom prostredí, ktoré často nedisponuje grafickou nadstavbou umožňujúcou vizuálnu konfiguráciu nastavení. Konzolový modul je zložený z dvoch hlavných funkcionalít:

- Spracovanie CLI príkazov a rôznych prepínačov.
- Integrácia systému zabezpečujúceho externú konfiguráciu prostredníctvom premenných prostredia alebo konfiguračného súboru.

Využitie možnosti štartu aplikácie prostredníctvom príkazov konzolového rozhrania prináša viaceré benefity. Poskytuje komfort a voľnosť počas vývoja softvéru pri jeho rozdelení na viaceré moduly vykonávajúce rôznu funkcionalitu nezávisle od seba, ale potrebujú prostriedky iných rozhraní v rámci spoločného systému. Príkladom efektívneho využitia tohto modelu implementácie sú nami navrhnuté funkcie *serve* a *migrate*. Príkazy pracujú nezávisle od seba; *serve* pri štarte systému a *migrate* pri príprave prostredia na prevádzku. Skript pre prvotnú prípravu systému by bolo možné vytvoriť separovane a v úplne inom jazyku a platforme. Z dôvodu, že vyžaduje pre správne fungovanie väčšie množstvo totožných zdrojov ako príkaz *serve*, dosiahneme rýchlejšiu implementáciu potrebnej funkcionality bez nutnosti opakovaného vytvorenia existujúcich rozhraní a dát. Zaradením migračnej metódy do primárneho systému zjednodušujeme prípravu infraštruktúry integráciou potrebnej funkcionality do jedného binárneho súboru bez žiadnych pozorovateľných dopadov na celkový výkon systému.

Druhým podstatným rozšírením je externalizácia konfiguračných nastavení inventarizačného systému. Vhodnými kandidátmi sú údaje potrebné k úspešnému pripojeniu do databázy. Statické zadanie týchto hodnôt by mohlo predstavovať problém pri správe systému na viacerých prostrediach, ktoré by si vyžadovali kompiláciu unikátnych binárnych súborov pre dané prostredie. Podobné praktiky taktiež komplikujú opravu a nasadenie opravných balíkov. Z toho dôvodu sú všetky senzitívne údaje ako heslá a ostatné

konfiguračné entity získavané z externých zdrojov pomocou modulu *viper*. Rozšírenie funguje na princípe párov kľúč – hodnota. Kľúč predstavuje referenčnú textovú konštantu, podľa ktorej je možné počas behu programu získať aktuálnu hodnotu premennej. Existujú tri druhy definovania hodnoty:

- **Predvolená hodnota** – umožňujúca zadať rôzne štandardné konštanty, ako je závažnosť chybových hlásení, ktoré v prípade nasadenia do produkcie chceme obmedziť iba na kritické. Takto môžeme odfiltrovať rôzne vývojové správy, ktoré bývajú často obsiahle a spôsobovali by prudký nárast veľkosti záznamových súborov. Ak by sme ale nedisponovali možnosťou zmeny úrovne a vývojové hlásenia by boli trvalo odstránené z finálneho produktu, sťažil by sa proces vyhľadania chyby v produkčnom prostredí.
- **Získanie hodnoty z konfiguračného súboru** – nám umožňuje zadanie hodnôt, ktoré môžu byť závislé od použitej infraštruktúry alebo hardvérového riešenia a sú uložené na súborovom systéme prostredia. Ide o dáta, ktoré sú kritické pre operáciu, ale nepredstavujú riziko v prípade úniku tretej strane.
- **Systémové premenné** – poskytujú možnosť definície premenných na úrovni systému počas jeho behu. Táto možnosť je vhodná na ukladanie kritických údajov ako sú heslá, ktoré by v prípade úniku konfiguračného súboru z disku mohli umožniť získanie citlivých dát tretím osobám. Mechanizmus je využitý hlavne v kombinácii s kontajnerizačnou technológiou Docker, ktorý poskytuje šifrovaný beh systémov zamedzujúci prístup k systémovým premenným v zabezpečenom prostredí.

### 5.3 Implementácia systémových rozhraní

Kapitola sa zaoberá vývojom samotných funkcionálnych požiadaviek systému, ktoré boli v procese analýzy definované pre aplikačné programové rozhranie. Každý systémový modul obsahuje špecifikáciu modelu nad ktorým pracuje a operácie ktoré môže s dátami vykonávať na príslušnej URI definovanej v relatívnej forme bez schémy(<http>/<https>/<ftp>) a authority(doména/IP adresa). Tieto časti sú vynechané kvôli prehľadnosti a faktu, že sú súčasťou nastavení infraštruktúry obsiahnutej v samostatnej kapitole 6.



### 5.3.1 Univerzálne operácie rozhrania

Ak nie je v nasledujúcich sekciách stanovené inak, všetky systémové moduly implementujú základnú sadu inštrukcií CRUD pre prácu s dátami. Toto označenie reprezentuje štyri operácie perzistentného úložiska; Vytvorenie(Create), Čítanie(Read), Úprava(Update) a Odstránenie>Delete). Jednotlivé operácie majú v REST špecifikácií priradené aj konkrétny druh HTTP operácie:

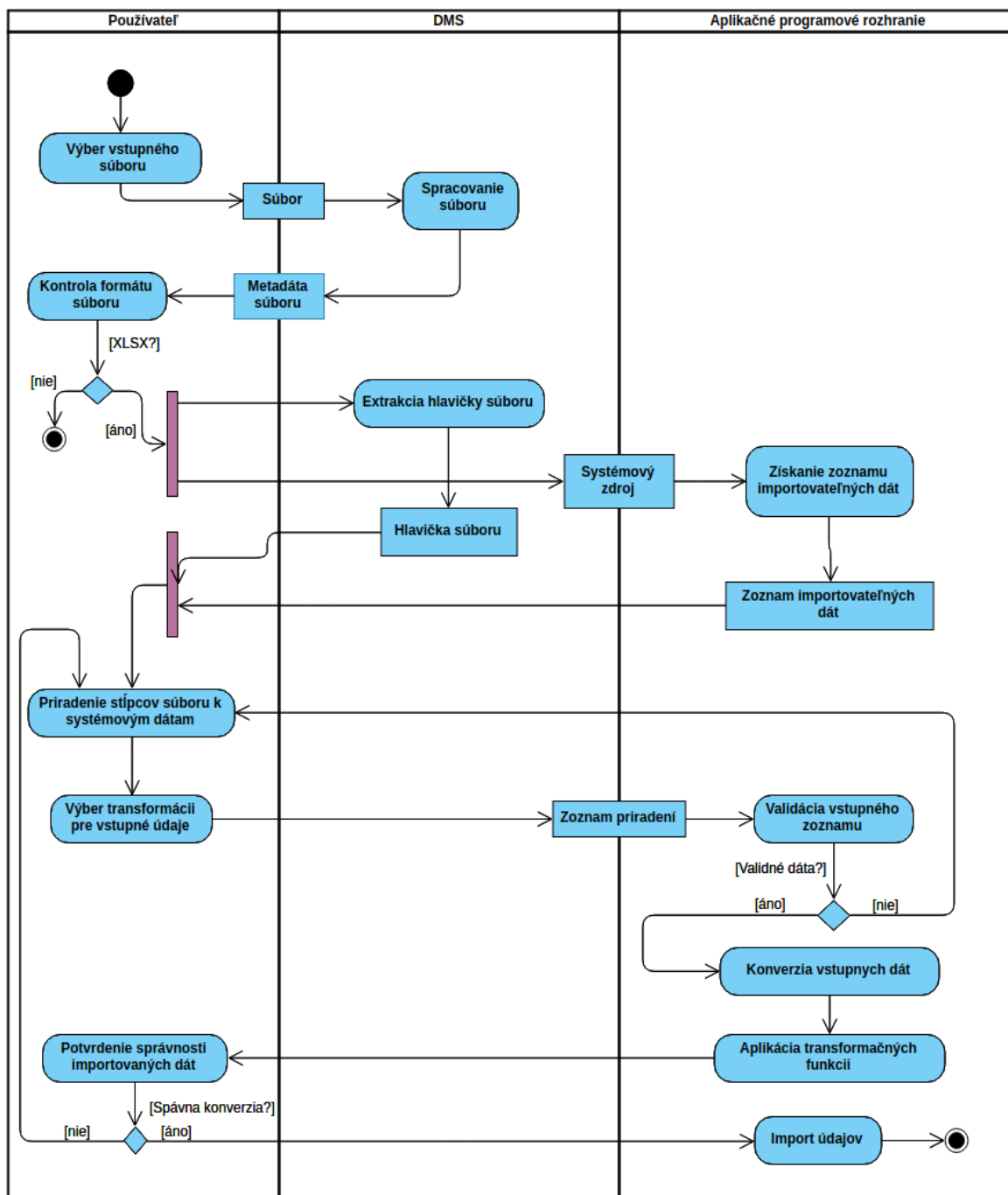
- Vytvorenie: POST
- Čítanie: GET
- Úprava: PUT
- Odstránenie: DELETE

Ide o odporúčané priradenie operácii pre dosiahnutie prehľadnosti aplikačného rozhrania inými vývojármi, a z toho dôvodu sme sa rozhodli tieto odporúčania dodržať.

Operácia zabezpečujúca vytvorenie záznamov pre jednotlivé systémové rozhrania využíva automatickú konverziu JSON formátu do príslušnej štruktúry popisujúcej schému zdrojového údaju. Jazyk Golang ponúka funkcionality značkovania schematických záznamov, ktorou je možné klasifikovať typ alebo označenie dát pre jednotlivé transformačné rozšírenia. Rovnako je to aj v prípade automatického dekódovania JSON formátu, kde značkou *json: "nazov"* zadefinujeme referenčný kľúč z prichádzajúcich dát na dáta používané našim systémom. Takáto funkcionality nám zabezpečuje flexibilitu v prípadoch, kedy štruktúra prijatých dát nie je v našej réžii a vstupné rozhranie je potrebné prispôbiť.

### 5.3.2 Rozhranie pre import dát

Z funkcionálnych požiadaviek definovaných v úvode projektu vyplynula nutnosť implementácie metódy umožňujúcej konverziu starého formátu dát pre využitie v našom systéme. Pred samotným vývojom je potrebná špecifikácia manuálnych a automatických krokov samotného procesu.



Obr. 26: Diagram aktivít popisujúci proces importu údajov do systému

Diagram aktivít na Obr. 26 reprezentuje zjednodušený proces spracovania vstupu importovaných údajov do systému. Nasledujúca kapitola popisuje aktivity vykonávané

aplikačným serverom v sekciách DMS a Aplikačné programové rozhranie. Sekcia používateľských úkonov je definovaná v samostatnej kapitole 4.

Všetky operácie zabezpečujúce spracovanie vstupných údajov majú na vstupe kompletnú sadu údajov, a preto nie je vyžadované vytvorenie žiadnej relácie medzi klientskou a serverovou aplikáciou. Jediným referenčným bodom, nad ktorým aplikačné rozhranie pracuje, je systémový identifikátor súboru.

Vstupný bod predstavuje modul správy dokumentov(DMS) zabezpečujúci prijatie, uloženie a následné spracovanie obsahu a metadát súborov. Akcia *Spracovanie súboru* získa všetky potrebné informácie o súbore, ako jeho formát, veľkosť alebo názov a následne vytvorí jedinečný identifikátor, ktorým bude v našom systéme odkazovaný. Výstupom tejto operácie je kolekcia dát popisujúca nahratý súbor.

Druhou operáciou podporovanou systémom DMS je extrahovanie hlavičky tabuľkového súboru. Pri volaní operácie s referenčným identifikátorom súboru prebieha kontrola formátu pre prípadnú nezhodu z podporovaným typom ako je XLSX. Po úspešnej validácii je načítaný prvý riadok súboru, ktorý štandardne predstavuje hlavičku tabuľky. Tento proces však nie je možné overiť, a preto môže nastať situácia, kedy sú získané priamo produkčné údaje. Tento faktor nepredstavuje problém pri fungovaní systému v prípade zohľadnenia v nasledujúcich krokoch procesu.

```
type Property struct {
    ID          int    `json:"- "`
    UUID        string  `json:"id"`
    CreatedAt   time.Time `json:"created_at,omitempty"`
    UpdatedAt   time.Time `json:"updated_at,omitempty"`

    Name        string  `json:"name" import:"true"`
    Code        string  `json:"code" import:"true"`
    Price       float32  `json:"price" import:"true"`
    AcquiredAt  time.Time `json:"acquired_at,omitempty" import:"true"`

    RoomID int    `json:"room_id" import:"true"`
    Room   *Room  `json:"room,omitempty" pg:"fk:room_id"`
}
```

Obr. 27: Ukážka definície modelu systémového zdroja umožňujúceho import

Dôležitou súčasťou pre zabezpečenie univerzálnosti vstupného rozhrania je možnosť označenia systémových údajov ako importovateľné objekty. Na Obr. 27 je možné vidieť exemplárny príklad definície modelu dát určených pre vstupné rozhranie. Pri návrhu sme využili rozšírenie štruktúr poskytované jazykom Golang umožňujúcu značkovanie jednotlivých prvkov špecifickým príznakom. Operácia *Získanie zoznamu importovateľných dát* prečíta schému štruktúry *Property* a z výsledného zoznamu vyfiltruje prvky obsahujúce

príznak *import* s hodnotou nastavenou na *true*. Takto zabezpečíme, aby mohli byť spracované iba používateľsky editovateľné dáta, pričom systémové údaje zostávajú izolované. Pri procese je taktiež zisťovaný aj očakávaný typ danej premennej, podľa ktorej je vykonávaná konverzia a priradenie správnej transformačnej funkcie.

Po dokončení a odoslaní priradenia systémových zdrojov k tabuľkovým stĺpcom nasleduje validácia získaného zoznamu. Je potrebné zabezpečiť integritu prijatých dát a odstrániť zdroje, ktoré nie sú určené pre import. Nesprávne vyplnenie údajov je používateľovi komunikované príslušným chybovým hlásením, ktoré pred znovu odoslaním vyžaduje opravenie chybných dát.

Úspešným ukončením validačného procesu je možné zahájiť konverziu vstupných údajov. Všetky údaje získané z priloženého súboru sú štandardne získavané v textovom formáte, ktorý nie je vždy vhodný pre niektoré systémové zdroje. Nasledujúce podkapitoly popisujú podporované operácie nad dátami.

Posledným krokom v procese je vizuálne potvrdenie importovaných dát používateľom vo webovom rozhraní. Týmto krokom je možné odhaliť chybné priradenie údajov, prípadne nesprávne konverzie a transformácie údajov. Na výstupe sú tieto dáta reprezentované dvojicou zoznamov obsahujúce správne údaje pripravené na zápis a nekompletné údaje, ktoré budú v procese ignorované.

### **5.3.3 Časový a dátumový formát vstupných dát**

Príkladom komplexnejšej konverzie môže byť dátumový formát definovaný tabuľkovým editorom Excel. Zdroj [14] popisuje tento zápis podrobnejšie.

Program Excel ukladá dátumové a časové údaje ako relatívne sériové číslo dátumu a času. Pri preskúmaní zobrazenia v programe je možné pozorovať, že tieto údaje sú reprezentované číslom s desatinnou časťou a formátovanie do dátumovej podoby prebieha iba vizuálne. Celočíselná časť hodnoty reprezentuje deň a čas je zakódovaný v desatinnej hodnote čísla. Výsledný dátum a čas je vypočítaný relatívne k dátumu 1. januára 1900, pričom tento konkrétny deň predstavuje číselnú hodnotu 1. Pri konverzii dátumov je potrebné dbať na fakt nesúladu s reálnym kalendárom. Dátumový formát XSLX počíta s rokom 1900 ako priestupný, napriek tomu, že nie je, a to z dôvodu spätnej kompatibility k formátu konkurenčného programu [14].

Tab. 2: Ukážka dátumov v súborovom formáte XSLX [14]

Reálny dátum	Hodnota reprezentovaná programom Excel
01. 01. 1900	1
01. 02. 1900	32
30. 07. 1960	22127
01. 04. 2018	43191

Hodnoty v Tab. 2 predstavujú ukážku popisovaného formátu. Ako príklad si môžeme uviesť dátum 1. 4. 2018, ktorý je reprezentovaný číselnou hodnotou 43191. To znamená, že tento deň je vzdialený od 31. 12. 1899 presne 43191 dní [14]. Túto vlastnosť môžeme využiť aj v programovej konverzii. V prvom kroku vytvoríme inštanciu dátumovej štruktúry *time.Time* poskytovanou jazykom Golang, ktorá disponuje metódou *AddDate* zabezpečujúcou posun nášho dátumu o daný počet dní. Funkcia taktiež zabezpečuje detekciu priestupných rokov, čím je zabezpečená integrita dát.

### 5.3.4 Transformačné operácie nad dátami

Úspešná konverzia dát je prvý krok v transformácii vstupných dát do požadovaného formátu. Môže sa však stať, že jednoduchá zmena typu údaju nie je dostačujúca a výsledok tejto operácie je v našom systéme v aktuálnej forme nepoužiteľný. Príkladom takéhoto údaju je kódové označenie miestnosti priradené k zariadeniu. Tento vstupný údaj je pri vstupe formátovaný ako textová konštanta. V našom systéme sú miestnosti k zariadeniam priradené pomocou kontextovo jedinečného inkrementálneho kľúča, automaticky generovaného pri vytvorení záznamu. Z tohto dôvodu je potrebná implementácia transformácie textového označenia kódom miestnosti na jej systémový identifikátor.

Pre zabezpečenie univerzálnosti tohto riešenia v prípade potreby budúceho použitia sme sa rozhodli vytvoriť rozširujúci modul poskytujúci túto funkcionality. Ako prvé je potrebné zvoliť identifikátor, podľa ktorého budú k jednotlivým dátam korešpondovať dostupné akcie. Takýmto údajom môže byť systémové kódové označenie prvkov, ako je možné vidieť v treťom stĺpci štruktúry na Obr. 23. Konkrétnym bodom záujmu je značka *json* špecifikujúca názov údaju v prípade jeho serializácie do JSON formátu. Tento formát je neskôr spracovaný vo webovom rozhraní aplikácie, kde je možné vykonať priradenie dostupných operácií nad konkrétnym prvkom.

V aktuálnom priblížení je potrebná operácia vyhľadávania identifikátorov miestností, ktorá bude v systéme figurovať pod kódovým označením *F\_RM\_ID\_CODE*. V prípade, ak pri

prechode transformačným algoritmom bude pri dátovom poli špecifikovaný daný kód, bude vykonaný dopyt do dátového úložiska. Pri úspešnom nájdení potrebného záznamu bude údaj miestnosti nahradený identifikátorom príslušného údaje. Vhodným rozšíreným tejto operácie je ponúknutie možnosti vytvorenia miestnosti v prípade, že nebol nájdený žiadany vhodný záznam. Takto môžeme uľahčiť prácu počas prvotnej migrácie dát.

## 5.4 Rozhranie zabezpečujúce monitorovanie majetku

Hlavnou súčasťou funkcionálnych požiadaviek systému je zabezpečenie monitorovanie pozície položiek hmotného majetku fakulty. Počas návrhu procesu sme identifikovali nasledovné body záujmu, ktoré vo finálnej podobe ovplyvnia štruktúru zaznamenávaných údajov.

Základným obmedzením celého systému je nedostatočná funkcionálna hardvérovej súčasti projektu, ktorá neposkytuje informáciu o smere prechodu jednotlivých zariadení. Riešenie poskytujúce obdobné údaje by vyžadovalo nasadenie dvojnásobného množstva senzorov, ktoré by pomocou časového oneskorenia prechodu dokázali určiť smer tranzitu. Táto možnosť by predstavoval značnú finančnú investíciu, ktorá by vo finálnej podobe nemusela priniesť požadované zlepšenie. Z tohto dôvodu sme sa rozhodli zabezpečiť softvérovú identifikáciu smeru prechodu jednotlivých zariadení.

Princíp určenia aktuálneho smeru prechodu zakladá na znalosti údajov o poslednom známom tranzite. V prípade ak takýto záznam nie je dostupný, algoritmus predpokladá fixnú polohu zariadenia v miestnosti, ktorá bola určená pri jeho úvodnej registrácii. Jednotlivé záznamy tak môžeme kategorizovať ako jeden z nasledovných typov:

- **Výstup** – špecifikuje, že dané zariadenie bolo vynesené z miestnosti, v ktorej sa aktuálne nachádzalo.
- **Vstup** – zariadenie bolo prenesené cez vstup a nachádza sa v danej miestnosti.
- **Vypožičanie** – ide o špecifický druh stavu *Výstup*, ktorý definuje umiestnenie zariadenia v osobnom vlastníctve autorizovaného používateľa vykonaného daný prechod. Použitie tohto stavu je v aktuálnom vyhotovení nutné špecifikovať manuálne pomocou webového rozhrania systému. Zavedením tohto stavu plánujeme dosiahnuť vyššiu transparentnosť historických údajov samotných zariadení a lepšiu prehľadnosť o polohe.

Záznam o prechode zariadenia obsahuje identifikátory štítka aj samotného zariadenia. Uchovávanie oboch informácií sa môže zdať ako redundantné, pretože každý štítok už obsahuje informáciu o zariadení, na ktorom je umiestnený. Oba údaje sú uchovávané z dôvodu dosiahnutia historickej presnosti záznamov, pretože počas návrhu systému sme sa rozhodli pre podporu možnosti znovu využitia identifikačných štítkov pre iné zariadenia.

Priebeh celého algoritmu je principiálne priamočiary. Po úspešnom overení existencie a správnosti všetkých vstupných údajov je vykonané získanie posledného známeho pohybu zariadenia. V prípade negatívneho výsledku načítania je uvažovaný krajný prípad, kedy je predpoklad umiestnenia zariadenia v danej miestnosti, a tak prechod bude vyhodnotený ako *Výstup*. Ak existuje historický záznam, nový stav je určený podľa hodnoty v Tab. 3.

Tab. 3: Špecifikácia prechodových stavov pozorovania majetku

<b>Starý stav</b>	<b>Nový stav</b>
Vstup	Výstup
Výstup	Vstup
Vypožičané	Vstup

Dôležitou funkciou algoritmu je overenie existencie aktívneho používateľa priradeného pre danú miestnosť. Výsledná odpoveď pre hardvérové zariadenie obsahuje príznak pre prípad neautorizovaného prechodu, ktorý vyvolá aktiváciu zvukového alarmu. Takto je osoba dodatočne vyzvaná na vykonanie overenia, kde v prípade jeho úspešného výsledku je prechod autorizovaný aj spätne.

## 6 Návrh infraštruktúry systému

Kapitola sa zaoberá použitou architektúrou systému produkčného prostredia, popisom využitých technológií umožňujúcich lepšiu správu a zabezpečením automatizácie.

### 6.3 Cieľová hardvérová architektúra

Z nefunkcionálnych požiadaviek vyplynula nutnosť prevádzky systému na zariadeniach s obmedzeným výpočtovým výkonom. Produkčný systém je počas vývojovej fázy prevádzkovaný na zariadení s označením Orange Pi Zero. Dôvodom voľby samotného zariadenia je vstavaná podpora pre bezdrôtové pripojenie technológie Wi-Fi, pomocou ktorého dokážeme vytvoriť vstupnú bránu pre hardvérovú platformu a jej technológiu ESP-MESH. Týmto rozhodnutím sme predišli nutnosti zaobstarávania externých USB Wi-Fi vysielačích staníc, ktoré môžu spôsobiť problémy s kompatibilitou ovládačov na upravenom operačnom systéme určeného pre našu platformu.

Systém disponuje nasledovnými parametrami [15]:

- Výpočtové jadro: H2 Quad-core Cortex-A7
- Systémová pamäť: 512MB DDR3 SDRAM
- Sieťové pripojenie: 10/100M s možnosťou pre POE
- Bezdrôtové pripojenie: XR819, IEEE 802.11 b/g/n
- Operačný systém: Armbian Linux 4.19.59

### 6.4 Využitie technológie kontajnerizácie

Jedná sa o technológiu určenú na uľahčenie vytvárania, nasadzovania a celkovej operácie aplikácie pomocou kontajnerov. Kontajnerom označujeme balík obsahujúci všetky potrebné závislosti na beh danej aplikácie. Takto dokážeme zabezpečiť plynulý beh našej aplikácie, bez dodatočných konfiguračných krokov počas nasadzovania na všetky podporované systémy [16] [17].

Svojim spôsobom sa technika kontajnerizácie funkcionálne podobá na operáciu virtuálneho stroja. Hlavným rozdielom je absencia virtuálneho operačného systému, nad ktorým následne beží klientska aplikácia. Docker umožňuje využitie spoločného Linuxového jadra ako operačného systému, čo zabezpečuje podporu využitia funkcionality hostiteľského systému. Týmto spôsobom je minimalizujeme počet potrebných komponentov a knižníc pribalovaných do samotných kontajnerov [17] [16].



Naša aplikácia využíva konkretizačnú technológiu z niekoľkých dôvodov. Portál Docker Hub obsahuje veľké množstvo oficiálnych ale aj komunitných balíčkov, ktoré je možné priamo využiť, alebo zobrať ako základ pre vytvorenie vlastného. Platforma taktiež poskytuje podporu pre inštrukčnú sadu ARM, ktorou disponuje naše produkčné prostredie, čo nám výrazne uľahčuje správu celého systému. Takýmto spôsobom vieme zabezpečiť bezproblémový prechod medzi prostrediami aj v prípade, že využívajú inú hardvérovú platformu. Táto úloha je delegovaná priamo konkrétnemu balíku na ktorom je aplikácia postavená.

Zdrojové kódy aplikačného programového prostredia a webovej aplikácie sa nachádzajú v externom Gitlab repozitári. Platforma nám umožňuje automatickú kompiláciu klientskej aplikácie a následné vytvorenie balíku určeného pre Docker prostredie. Tento úkon je označovaný ako CI/CD, pomocou ktorého je zabezpečená nepretržitá integrácia a nasadenie klientskej aplikácie.

Nepretržitá integrácia je definovaná ako proces zlúčenia všetkých zdrojových kódov jednotlivých aplikácií. Operácia zlúčenia často prebieha aj niekoľkokrát denne v externom zdieľanom úložisku, ako napríklad Gitlab. Označením zdrojového kódu verziou sú spustené automatické stavebné a testovacie úkony, ktoré zabezpečujú bezproblémové nasadenie a včasnú identifikáciu chybových stavov aplikácie [18].

Docker Compose je nástroj, ktorý slúži na definíciu a správu Docker aplikácii pozostávajúcich z viacerých kontajnerov. Konfigurácia parametrov pre spustenie jednotlivých kontajnerov je definovaná v rámci spoločného súboru formátu YAML. Celú definovanú infraštruktúru je tak možné vytvoriť a spustiť pomocou jedného príkazu so vstupným parametrom nášho konfiguračného súboru.

## 7 Overenie implementovaného riešenia

Kapitola sa zaoberá overením implementačnej časti projektu inteligentného inventarizačného informačného systému v simulovanej produkčnej prevádzke. Hlavným zámerom jednotlivých testovacích scenárov je overenie priechodnosti systému počas umelých záťažových testov.

### 7.3 Odozva aplikačného programového prostredia pod záťažou

Kvôli zabezpečeniu pohodlnej a spoľahlivej prevádzky systému v produkčnom prostredí je potrebné vykonať záťažový test zameraný na odozvu aplikácie. Takýmto procesom dokážeme určiť počet používateľov, ktorý je možné dosiahnuť na konkrétnom hardvérovom vybavení pri zachovaní rýchlej odozvy.

Test spočíva v kontinuálnom volaní služieb aplikačného programového rozhrania z klientskeho prostredia, kde následne prebehne jedna kontrolná iterácia pozorujúca odozvu systémových zdrojov. Takto dokážeme simulovať používateľské správanie, ktoré by vychádzalo z volania operácii vo webovom rozhraní aplikácie. Počet paralelných klientov kontinuálne volajúcich služby bez prestávky je postupne zvýšený, čím dosiahneme väčšiu záťaž na systém.

Tab. 4: Výsledky záťažového testu aplikačného programového rozhrania

Počet používateľov	Priemerný čas zápisovej operácie [ms]	Priemerný čas čítacej operácie [ms]
1	182,70	90,67
2	342,166	147,38
3	357,33	252,00
4	368,83	269,00
5	545,00	265,71
10	550,67	281,86

Tab. 4 obsahuje priemerný čas trvania operácie spustenej používateľom. Tento čas reprezentuje dobu, ktorú používateľ čaká na vizuálne potvrdenie odpovede webovej aplikácie. Jednotlivé operácie sú rozdelené na dve kategórie, tie ktoré vykonávajú zápis a zmenu údajov v databáze a tie, ktoré hodnoty len čítajú. Rozdelenie je vykonané z dôvodu, že zapisovacie operácie sú vykonávané zriedkavejšie a používateľ očakáva isté časové oneskorenie. V prípade operácií zabezpečujúcich čítanie údajov je optimálne dosiahnuť čas pod 150ms,

kde sa nachádza prah ľudského pozorovania zmeny. Po prekročení tohto okamihu je vhodné využitie animácie prebiehajúcej operácie, potvrdzujúcej používateľovi úspešné spustenie jeho akcie. Zvyšujúcimi hodnotami stráca aplikácia svižnosť ovládania a môže dochádzať k frustrácii používateľov. Z pohľadu, že sa jedná o simulovanú záťaž kontinuálneho volania služieb, považujeme výsledky za dobré a systém je aj v takomto extrémnom prípade použiteľný.

## 7.4 Odozva bezdrôtového prenosu v závislosti od počtu zariadení

Využitím technológie ESP-MESH nám umožnilo väčšiu voľnosť pri rozmiestnení hardvérových zariadení. Systém využíva bezdrôtové pripojenie pre prenos údajov z riadiacej jednotky do centrálného informačného systému. Zo samotného umiestnenia jednotlivých zariadení a ich vzdialenosti od prístupového bodu môže byť sieť usporiadaná do stromovej štruktúry z rôznou veľkosťou a hĺbkou. Väčšia hĺbka predstavuje väčšie množstvo medzi prechodov dátového prenosu, čo môže spôsobiť časové oneskorenie prijatia dát cieľovým zariadením. Skúška sa zameriava na čas jedného cyklu komunikácie, od bodu kedy bola požiadavka vyslaná z riadiacej jednotky až po moment, kedy bola prijatá odpoveď z aplikačného programového prostredia.

Tab. 5: Výsledky testu odozvy hardvérových zariadení

Počet zariadení v sieti	Priemerný čas cesty pre zariadenie 1[ms]	Priemerný čas cesty pre zariadenie 2[ms]	Priemerný čas cesty pre zariadenie 3[ms]
1	300,12	-	-
2	318,14	339,13	-
3	328,26	340,96	366,99

Výsledky testu sú špecifikované v Tab. 5. So zvyšujúcim sa počtom zariadení v sieti môžeme pozorovať nárast priemerného času odozvy, spojenou s nutnosťou prechodu údajov cez viacero prenosových bodov. Pozorovaný nárast v dĺžke trvania požiadavky nepôsobil žiadny diskomfort používateľa pri vstupnej bráne, čakajúceho na potvrdenie pri vstupnej bráne.

## 7.5 Test komplexnej záťaže systému

V reálnej prevádzke je predpoklad, že bude systém spracovávať požiadavky z webového rozhrania a hardvérovej siete simultánne. Samotný test kombinuje techniky predošlých testovacích scenárov. Prvým bodom záujmu je odozva hardvérovej časti projektu, ktorá predstavuje prioritný vstup dát, pretože pri jeho spomalení by mohlo dochádzať k narušeniu integrity dát. Druhým bodom záujmu je odozva aplikačného programového rozhrania, ktorého spomalenie by spôsobilo zhoršený používateľský zážitok(UX), ale nespôsobí zásadné problémy pre chod celého systému a jeho integrity.

Test prebiehal s tromi hardvérovými zariadeniami pripojenými do jednotnej siete, kontinuálne zasielajúcimi požiadavky o prechode zariadenia na jednotlivých bránach. V rovnakom čase boli simulované 4 klientske pripojenia k aplikačnému programovému prostrediu, ktoré neustále vytvárali požiadavky na čítanie a zápis dát.

Priemerný čas prijatia správy:

- Zariadenie v koreni hierarchie: 393,53ms
- Podriadené zariadenie č. 1: 419,48ms
- Podriadené zariadenia č. 2: 424,37ms

Priemerný čas odozvy aplikačného programového rozhrania:

- Operácie zápisu: 262,10ms
- Operácie čítania: 239,29ms

Získané výsledky potvrdzujú možnosť operácie systému na platforme s obmedzeným výpočtovým výkonom v produkčnom prostredí. Reakčné časy sú pocitovo pomalšie a menej svižné, napriek tomu sa jedná o akceptovateľné hodnoty z ohľadom na cenu operačného hardvéru. Jedná sa v princípe o hodnoty, ktoré je možné dosiahnuť na platforme, spĺňajúcej minimálne operačné požiadavky a v prípade nasadenia do plnohodnotnej prevádzky je možné očakávať výrazné zlepšenie odozvy.

## Zhodnotenie

V analýze diplomovej práce sme sa zaoberali aktuálne dostupnými systémami zabezpečujúcimi lokalizáciu a správu hmotného majetku. Bližšie sme si špecifikovali rozdiel medzi zámerom systému spravujúcim hmotný majetok a inventár spoločnosti. Analýza sa taktiež venovala hardvérovému vybaveniu umožňujúce získanie pozície položiek. Bližšie sme si špecifikovali vyhotovenie jednotlivých techník sledovania a ich prípady použitia. Na základe týchto informácií sme boli schopní urobiť informovaný odhad vhodnej technológie a techniky lokalizácie hmotného majetku pre potreby Ústavu elektrotechniky.

Výsledkom analýzy bola ucelená predstava o návrhu štruktúry nášho informačného systému a využití konkrétnych hardvérových komponentov. V cieľoch projektu sme bližšie špecifikovali funkcionálne a nefunkcionálne požiadavky pre implementačný proces.

Kľúčovým komponentom systému je hardvérová časť projektu, zabezpečujúca autorizáciu prístupujúcich používateľov do pozorovaných miestností. Výber samotného čítacieho komponentu zabezpečujúceho získavanie identifikačných údajov z prístupovej karty používateľa prešiel viacerými iteráciami. Precíznym overením hardvérových komponentov sme zabezpečili správne fungovanie systému z dlhodobého hľadiska. Druhou dôležitou funkciou vykonávanou riadiacou jednotkou je detekcia prechodu pozorovaných zariadení kontrolným bodom. Počas implementácie bolo potrebné zabezpečiť operáciu zariadenia v relatívnom vyhodnocovacom režime smeru tranzitu s jedným čítacím senzorom. Takýmto návrhom sme dosiahli značnú úsporu finančných prostriedkov vynaložených na potrebné vybavenie.

Projekt si taktiež vyžadoval vytvorenie centrálného bodu zabezpečujúceho spracovanie a uskladnenie získaných údajov. Hlavným rozhodovacím faktorom výberu implementačného jazyka sa stala výsledná hardvérová platforma, na ktorej bolo potrebné zabezpečiť bezproblémovú prevádzku. Implementácia centrálného informačného systému bola rozdelená na dve vstupné rozhrania, zabezpečujúce komunikáciu s hardvérovou časťou projektu a webovým užívateľským rozhraním.

Implementáciou jednostránkovej webovej aplikácie(SPA) sme spojili využitie najnovších technológií v oblasti so zabezpečením maximálneho používateľského komfortu, na ktorý sú zvyknutý z moderných mobilných aplikácií. Používateľovi zabezpečuje plnohodnotnú správu systému intuitívnou formou.

Počas návrhu a implementácie infraštruktúry systému sme využili technológiu kontajnerizácie, ktorá sa potvrdila ako efektívna voľba už počas samotného vývoja aj pri samotnej produkčnej prevádzke.

Efektívnosť prevádzky implementovaného inventarizačného systému sme potvrdili záťažovými testami. Výsledky potvrdili predpoklad, že počítače s obmedzeným výpočtovým výkonom podobné Raspberry Pi, dokážu pri využití správnych metód vývoja poskytnúť dostatočný výkon aj na mierne pokročilú produkčnú prevádzku.

## Literatúra

- [1] Steve Ciemcioch. (2018) Everything You Need to Know About Inventory Management. [Online]. <https://www.warehouseanywhere.com/resources/inventory-management/>
- [2] Lowry Solutions. (2014) What Is the Difference between Inventory Management and Asset Tracking? [Online]. <https://lowrysolutions.com/blog/what-is-the-difference-between-inventory-management-and-asset-tracking/>
- [3] Anni Junnila. (2017, Jún) 5+1 Asset Tracking Technologies: Which One to Use for Your Business? [Online]. <https://trackinno.com/2017/06/21/asset-tracking-technologies/>
- [4] Advsolned. (2018) Industry 4.0: Asset Track and Trace. [Online]. <https://www.advsolned.com/industry-4-0-asset-track-and-trace/>
- [5] Abitec s.r.o. (2020) MAJETOK. [Online]. <https://abitec.sk/portfolio-view/majetok/>
- [6] Explore Embedded. (2017) Overview of ESP32 features. What do they practically mean? [Online]. [https://www.exploreembedded.com/wiki/Overview\\_of\\_ESP32\\_features.\\_What\\_do\\_they\\_practically\\_mean%3F](https://www.exploreembedded.com/wiki/Overview_of_ESP32_features._What_do_they_practically_mean%3F)
- [7] NXP Semiconductors. (2016) MFRC522 Standard performance MIFARE and NTAG frontend. [Online]. <https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf>
- [8] Fernando Koyanagi. (2020) ESP32 With RFID: Access Control. [Online]. <https://www.instructables.com/id/ESP32-With-RFID-Access-Control/>
- [9] André Balboa. (2019) Arduino RFID Library for MFRC522. [Online]. <https://github.com/miguelbalboa/rfid>
- [10] Austin Leavitt. (2018, September) What Is The Wiegand Protocol? [Online]. <https://getsafeandsound.com/2018/09/wiegand-protocol/>
- [11] Robert Peška. (2017) Wiegand – od karty po sběrnici. [Online]. <https://automatizace.hw.cz/wiegand-od-karty-po-sbernici.html>
- [12] Espressif Systems. (2019) ESP-MESH. [Online]. <https://docs.espressif.com/projects/esp-idf/en/stable/api-guides/mesh.html>
- [13] John Griffin. (2019) When to Use Go vs. Java | One Programmer's Take on Two Top Languages. [Online]. <https://spiralscout.com/blog/when-to-use-go-vs-java-one-programmers-take-on-two-top-languages>
- [14] Mynda Treacy. (2017, Október) Excel Date and Time. [Online]. <https://www.myonlinetraininghub.com/excel-date-and-time>
- [15] Xunlong Software CO. (2020) What's Orange Pi Zero ? . [Online]. <http://www.orangepi.org/orangepizero/>
- [16] Jared Kobos. (2020) When and Why to Use Docker. [Online]. <https://www.linode.com/docs/applications/containers/when-and-why-to-use-docker/>
- [17] opensource.com. (2019) What is Docker? [Online]. <https://opensource.com/resources/what-docker>
- [18] Gitlab Inc. (2019) GitLab Continuous Integration (CI) & Continuous Delivery (CD). [Online]. <https://about.gitlab.com/stages-devops-lifecycle/continuous-integration/>

## Prílohy

### Príloha A: Ukážka prototypov hardvérových zariadení

Ukážka produkčného vybavenia, na ktorom prebiehal vývoj a testy z kapitoly 7 je možné vidieť na Obrázok 1.



Obrázok 1 Produkčný hardvér Orange Pi Zero



Pre správne fungovanie systému bolo potrebné zabezpečiť získanie autorizačných dát používateľa. K tomuto úkonu využívame čítačku na Obrázok 2.



Obrázok 2 Čítačka s rozhraním Wiegand 36 pre načítanie ISIC kariet

Výsledkom implementačného procesu hardvérového riešenia je prototyp(Obrázok 3) získavajúci autentifikačné údaje o prístupujúcom používateľovi a identifikátory pozorovaní položiek.



Obrázok 3 Prototyp prístupovej brány

## Príloha B: Systémová príručka

### Aplikačné programové rozhranie

Kapitola popisuje metódu spustenia aplikačného programového rozhrania zabezpečujúceho prístup k dátam pre webovú aplikáciu. Súčasťou rozhrania sú aj statické súbory webovej aplikácie, ktoré sú automaticky poskytované pri prístupe na webovú adresu servera. Týmto spôsobom je zjednodušená prevádzka systému, takže nie je potrebná inštalácia webových serverov ako Apache. Využitie tohto rozšírenia je dobrovoľné a v prípade potreby komplexnejšieho riešenia je možné využiť softvér tretej strany.

Spustenie prevádzky samotného systému je možné viacerými spôsobmi:

1. Využitím prostredia Docker a rozšírenia Docker Compose.
2. Manuálna inštalácia a nasadenie balíkov.

### Prevádzka produkcie využitím technológie Docker a Docker Compose

Jedná sa o preferovanú metódu využitia nášho riešenia, pretože obsahuje všetky potrebné balíky pre zabezpečenie optimálnej funkcionality.

Požadované systémové rozšírenia:

- Docker Engine (Docker Desktop v prípade operačného systému Windows)
  - <https://docs.docker.com/engine/install/>
- Docker Compose
  - <https://docs.docker.com/compose/install/>

V repozitáre projektu sa nachádza zložka označená ako *Stack*. Obsahom zložky sú konfiguračný súbor pre Compose, ktoré obsahujú všetky potrebné balíčky. Ukážkovo sú poskytnuté dve varianty pre finálne produkčné prostredie:

- AmrV7 – Určené pre spustenie na počítačoch s obmedzeným výpočtovým výkonom ako Raspberry Pi.
- AMD64 – Určené pre výkonovo plnohodnotné systémy platformy AMD64.

Spustenie samotného prostredia prebieha príkazom:

```
docker-compose -f docker-compose.PROSTREDIE.yml up -d
```

### **Prevádzka produkčného systému manuálnou inštaláciou**

V prípade, ak nie je možná inštalácia niektorého z potrebných komponentov určených na kontajnerizáciu, alebo používateľ nedisponuje administračnými privilégiami, systém umožňuje prevádzku manuálnou inštaláciou a spustením jednotlivých komponentov.

Predpoklady nutné pre manuálnu operáciu sú mimo informačný rozsah tohto manuálu a je potrebná ich ručná inštalácia podľa architektúry produkčného prostredia:

- Postgres 12
  - <https://www.postgresql.org/download/>

Zložka *Binaries* v repozitári projektu obsahuje binárne súbory jednotlivých častí projektu podľa prostredia, pre ktoré sú určené.

### **Aplikačné programové rozhranie určené pre ESP-MESH**

Tento projekt nesie označenie *gateway* a slúži ako rozhranie prepájajúce bezdrôtovú sieť ESP-MESH hardvérovej časti s aplikačnou logikou. Logika v aktuálnom priblížení neobsahuje žiadne konfiguračné nastavenia. Spustenie je možné prostredníctvom príkazu:

- `./gateway serve`
- `gateway.exe serve` (OS Windows)

### **Aplikačné programové rozhranie určené pre web**

Projekt je v repozitárovej štruktúre označovaný ako *facade* a vykonáva samotnú operačnú logiku celého systému.

Spustenie systému prebieha príkazmi:

- `./facade operácia [--prepínače]`
- `facade.exe operácia [--prepínače]`

Parameter *operácia* predstavuje jednu z podporovaných funkcionalít:

- *migrate* – Príkaz slúži na spustenie inicializačných skriptov, zabezpečujúcich prípravu dátovej a tabuľkovej štruktúry v databázovom systéme. Tento príkaz je potrebné vykonať pred prvým spustením aplikačného prostredia.
- *serve* – Príkaz spúšťa webový server a aplikačné programové rozhranie.

Parameter *prepínače* poskytuje dodatočnú konfigurovateľnosť systému a je možné použiť niektorý z nasledujúcich príkazov:

- *config* – Umožňuje špecifikáciu vstupného súboru obsahujúceho konfiguráciu celého systému
- *db\_debug* – Slúži ako vývojová pomôcka, ktorá do konzolového logu zaznamenáva všetky vykonané SQL dopyty do databázového systému

Parametre vstupného konfiguračného súboru je potrebné špecifikovať vo formáte YAML súboru:

- **DATABASE\_URL** – Špecifikuje cestu k databázovému serveru, je zadávaný vo formáte „*postgres://:používateľ@adresa:5432/databáza?sslmode=disable*“.
- **port** – špecifikuje adresu a port, na ktorom má byť aplikácia spustená. Zadávané vo formáte „*0.0.0.0:port*“.
- **url\_posfix** – Slúži ako rozširujúca funkcionálna, ktorá poskytuje možnosť separácie aplikačného programového rozhrania od webového serveru. V prípade uvedenia hodnoty bude aplikácia počúvať na adrese *http://port/url\_posfix*

### Konfigurácia a kompilácia hardvérovej časti projektu

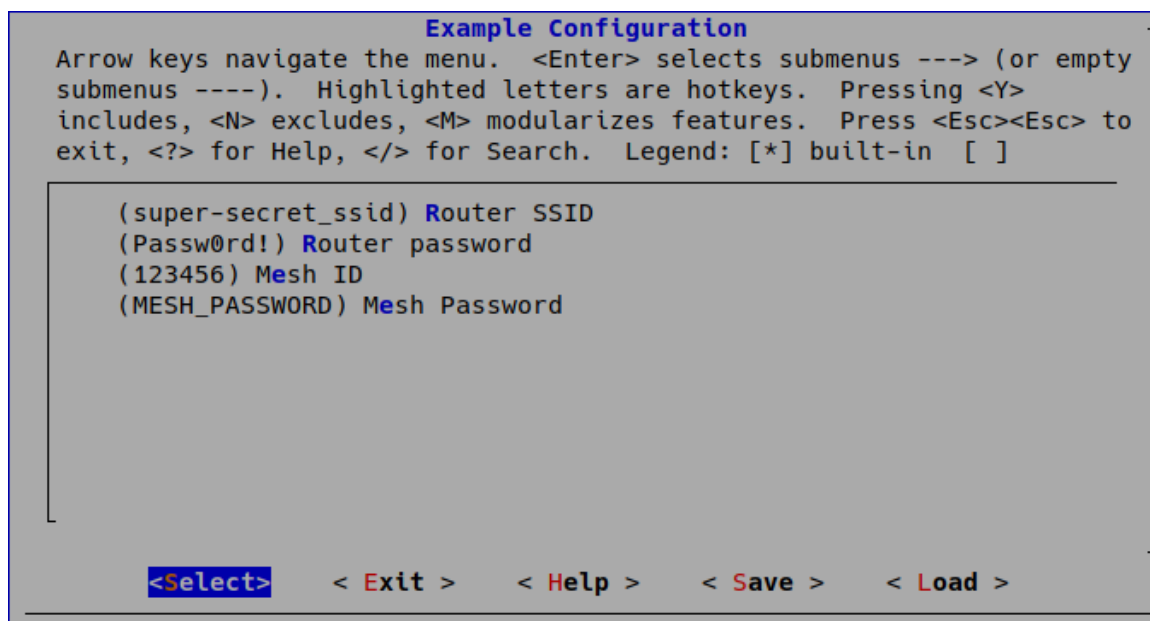
Súčasťou projektu je aj vyhotovenie hardvérovej platformy zabezpečujúcej získavanie autentifikačných údajov prístupujúcich užívateľov a lokalizačných dát zaradení. Pri vývoji tejto časti projektu bol vytvorený firmvér, ktorý je potrebné nahráť do jednotlivých riadiacich jednotiek.

Pre zabezpečenie jednoduchšej operácie s projektom sme využili možnosť kontajnerizácie projektu s pribalením potrebných balíčkov ESP-IDF a ESP-MDF. Všetky potrebné zdrojové súbory sa nachádzajú v adresári *ESP* v projektovom repozitári. Spúšťači skript *builder* slúži ako vstupná inštancia kompilačného prostredia. Po jeho spustení máme dostupné konzolové prostredie, poskytujúce možnosť konfigurácie projektu, jeho samotnú kompiláciu a napálenie do hardvérovej jednotky. Dostupné príkazy:

- *make flash* – Príkaz zabezpečuje kompiláciu projektu a následne napálenie firmvéru do centrálnej jednotky.
- *make menuconfig* – Príkaz poskytuje konfiguračné rozhranie projektu v grafickej podobe.
- *make monitor* - Príkaz poskytuje ladiacu funkcionálnu nad živým zariadením vo forme konzolových výpisov počas behu programu.

## Konfiguračné nastavenia centrálnej jednotky

Spustením príkazu *make menuconfig* vo vývojovom prostredí získame možnosť konfigurácie siete ESP-MESH, ako môžeme vidieť na Obrázok 4.



Obrázok 4 Konfiguračné nastavenia ESP-MESH

Jednotlivé konfiguračné polia predstavujú nasledovnú funkcionality:

- **Router SSID** – Reprezentuje názov prístupového bodu, pomocou ktorého je sieť ESP-MESH pripojená k aplikačnému programovému rozhraniu
- **Router password** – Predstavuje prihlasovacie heslo do danej siete
- **Mesh ID** – Slúži ako klasifikácia použitej siete, aby zariadenia pripojené v bezdrôtovej sieti dokázali rozlíšiť pri využití viacerých MESH sietí.
- **Mesh Password** – Slúži ako dodatočný autentifikačný faktor pri pripojení do senzorovej siete.

## Vývojové prostredie webovej aplikácie

Pre prípad vykonania rôznych úprav vo finálnej webovej aplikácii poskytujeme návod pre spustenie vývojového prostredia. Pre spustenie sú potrebné nasledovné systémové závislosti:

- Node.js verzie 10
- NPM verzie 6

Pred úvodným spustením aplikácie je potrebná inštalácia všetkých vyžadovaných knižníc:

- `npm install`

Po úspešnom ukončení inštalácie je možné spustenie samotného vývojového prostredia prostredníctvom príkazu:

- `npm run start`

Ak chceme vykonané zmeny využiť v produkčnom prostredí, je nutné vykonať kompiláciu webového prostredia, ktorým získame minimalizovanú formu určenú pre produkciu:

- `npm build`

Výsledkom tejto operácie je zložka *dist*, obsahom ktorej je výsledná webová stránka, ktorú je možné umiestniť na webový server.

## Príloha C: Štruktúra repozitára

Adresa repozitára: <https://github.com/xchovanecv1/DP-I3S>

- DP\_Chovanec.docx - Diplomová práca
- DP\_Chovanec.pdf - Diplomová práca
- Binaries/
  - AMD64 - Spustiteľné súbory aplikácie
  - ARMv7 - Aplikácia určená pre Raspberry Pi
- ESP - Zdrojové súbory hardvérovej časti projektu
- Facade - Zdrojové súbory aplikačného programového rozhrania
- Frontend - Zdrojové súbory webovej aplikácie
- Gateway - Zdrojové súbory rozhrania ESP-MESH a API
- Stack - Konfiguračné súbory pre spustenie Docker Stacku