Systém evidence zboží s využitím QR kódů

Bc. Michal Vetr

Diplomová práce
2017

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2016/2017

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Michal Vetr**
Osobní číslo:	**A13400**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Bezpečnostní technologie, systémy a management**
Forma studia:	**kombinovaná**
Téma práce:	**Systém evidence zboží s využitím QR kódů**
Téma anglicky:	**A Goods Evidence System Using QR Codes**

Zásady pro vypracování:

1. **Specifikujte požadavky na systém evidence.**
2. **Navrhněte HW schéma pro řešení.**
3. **Vytvořte SW základnu pro evidenci.**
4. **Implementujte Vaše řešení v testovacím prostředí.**
5. **Proveďte ověření funkčnosti a specifikujte možné omezení Vašeho řešení.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **MATULA, Terry. Laravel application development cookbook. Birmingham: Packt Publishing, 2013. ISBN 978-1-78216-283-4.**
2. **MEW, Kyle. Android 5 Programming by Example By Example. Birmingham: Packt Publishing, 2015. ISBN 978-1-785288-449-5.**
3. **MORRIS, Jason. Android User Interface Development Beginner's Guide. Birmingham: Packt Publishing, 2011. ISBN 978-1-84951-448-4.**
4. **LACKO, Ľuboslav. SQL: hotová řešení. Brno: Computer Press, 2003. K okamžitému použití. ISBN 80-7226-975-5.**
5. **WAHER, Peter. Learning Internet of Things. Birmingham: Packt Publishing, 2015. ISBN 978-1-78355-353-2.**
6. **Linux: dokumentační projekt. 4., aktualiz. vyd. Přeložil Lubomír PTÁČEK. Brno: Computer Press, 2007. ISBN 978-80-251-1525-1.**
7. **UPTON, Eben a Gareth HALFACREE. Raspberry Pi: uživatelská příručka. 2., aktualizované vydání. Přeložil Jakub GONER. Brno: Computer Press, 2016. ISBN 978-80-251-4819-8.**
8. **Jelínek, Tomáš. Grafické kódy pro identifikaci výrobků a služeb. Pardubice, 2010. BAKALÁŘSKÁ PRÁCE. Univerzita Pardubice. Vedoucí práce Ing. Milan Tomeš.**

Vedoucí diplomové práce: **Ing. David Malaník, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **3. února 2017**

Termín odevzdání diplomové práce: **24. května 2017**

Ve Zlíně dne 3. února 2017

doc. Mgr. Milan Adámek, Ph.D.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

doc. RNDr. Vojtěch Křesálek, CSc.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 23. 5 .2017 …….

podpis diplomanta

ABSTRAKT

V rámci této práce vznikl systém pro evidenci zboží pomocí QR kódů. Navržený systém využívá dvou aplikací. První z nich má úlohu čtečky a je naprogramována pro mobilní telefony se systémem Android. Druhá aplikace se stará o skladovou bussines logiku. Pro její programování byl zvolen jazyk PHP s využitím frameworku Laravel. Aplikace je naistalována na serveru, jehož hardwarovou základnu tvoří micro počítač Raspberry Pi. Programovou výbavu serveru dotváří operační systém Linux, webový server Apache a databázový systém MySQL.

Klíčová slova: QR kód, Raspberry Pi, Android, MySQL, PHP, Linux, evidenco zboží, Laravel

ABSTRACT

As a part of this thesis, a system for registering goods with QR codes has been created. The proposed system uses two applications. The first one has a reading role and is programmed for Android phones. The second application takes care of warehouse business logic. For programming, the PHP language was chosen using the Laravel framework. The application is installed on a server whose hardware base is the Raspberry Pi micro computer. The server's software package completes the Linux operating system, the Apache web server and the MySQL database system.

Keywords: QR code, Raspberry Pi, Android, MySQL, PHP, Linux, goods register, Laravel

Děkuji panu Ing. Davidu Malaníkovi, Ph.D. za ochotu, přátelský přístup a trpělivost. Dále děkuji Ing. Jaroslavu Železníkovi za cenné rady a pozitivní motivaci během vývoje aplikací. Můj dík patří taktéž Nikole Hájkové za pokus o pomoc s korekturou této práce.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Skladové hospodářství je v dnešní době nezanedbatelnou součástí výrobního i nevýrobního průmyslu. Zejména ve výrobních společnostech, které ve svých metodikách řízení využívají prvky filozofie *„Just in Time“*, je objem výroby produktu výrazně závislý na způsobu, jakým fungují dodávky zboží do společnosti [11].

S nárůstem dodávek zboží do společnosti se přímo úměrně zvedá administrativní zátěž při příjmu zboží nebo jeho výdeji. V souvislosti s tím a přihlédnutím k faktu, že se za posledních několik let výpočetní technika výrazně zlevnila, jsou ve společnostech nasazovány více či méně rozsáhlé informační systémy pro evidenci pohybu zboží. Buď mohou být součástí většího celku nebo se může jednat o dílčí subsystém, který přes rozhraní komunikuje např. s účetním systémem apod. Při správném způsobu implementace jsou společnosti díky těmto systémům schopny zefektivnit jednotlivé výrobní procesy a implikovat tak zvýšení produkce.

Systém pro příjem a výdej zboží však nemusí potřebovat pouze velké výrobní společnosti. V dnešní době se pořádá spousta kulturních akcí, kdy se veškerý proviant uskladňuje v tzv. mobilních skladech, v takovém případě se pořadatele potýkají se spoustou omezení. Zboží může být na místo dopraveno v krátkém intervalu a ve velkém množství. Takže není dostatek času na uspokojivou instalaci HW prostředků na místo určení. Další komplikaci může způsobovat nedostatečný zdroj energie.

Takové pracovní podmínky vybízí k úvahám použití jednodeskového počítače. V případě této práce byl zvolen počítač Raspberry Pi 3. Jako operační sytém na něj bude naistalována verze GNU/Linuxu přesněji Raspbianu, což je upravená verze Debianu určená přímo pro potřeby Raspberry Pi [4]. Pro účely evidence pohybu zboží bude nutné využít další komponenty, které nám linuxová platforma nabízí. Jedná se hlavně o databázový systém, webový server a interpret programovacího jazyka. Tyto technologie budou blíže představeny v první teoretické části.

V druhé teoretické části se práce zaměří na návrh testovacího prostředí za použití technologií popsaných v části první. Návrh realizace bude popsán v první praktické části. Zde bude kladen důraz především na tvorbu webové aplikace, kterou bude možné evidovat pohyby zboží v našem skladu. Důležité budou operace pro příjem a výdej zboží ze skladu. Jelikož by měl být celý systém co možná nejrychleji a nejjednodušeji ovladatelný, bude

evidence zboží probíhat pomocí QR kódu. Proto by v systému neměla chybět možnost generovat a tisknout QR štítky, díky nimž bude možno rozlišit jednotlivé zboží.

Následně vyvstane potřeba pro tvorbu čtecího zařízení. Tím může být cokoliv, co dokáže načíst data z QR kódu. V této práci se pro tyto účely pokusíme vytvořit jednoduchou aplikaci pro operační systém Android. Její rolí bude načíst data z QR kódu a za pomocí REST API [12] tyto operační data poslat na aplikační server. Tento proces by měl představovat primární operaci k evidenci příjmu a výdeje zboží.

I. TEORETICKÁ ČÁST

1 SPECIFIKACE POŽADAVKU NA SYSTÉM EVIDENCE

Problematika skladového hospodářství zahrnuje spoustu dílčích procesů. Elementárními úkony skladových zaměstnanců bývají zejména naskladnění a vyskladnění zboží. Aby pracovníci byli schopni zboží ze skladu vydat, musí mít proviant skladem. Z toho vyplývá, že v pokročilém skladovém systému musí existovat funkce pro sestavení objednávky zboží. Jednou takovou může být Paretova (ABC) analýza[1]. Její pomocí lze zkoumat historii pohybu jednotlivého zboží a na základně minimální skladové hodnoty určit, kdy bude nejvhodnější doba pro vytvoření nové objednávky. Z hlediska demonstrace funkčnosti systému se v rámci této práce zaměříme na procesy pro příjem nebo výdej zboží.

V rámci této kapitoly bude specifikován návrh systému z perspektivy funkční. K tomu nám poslouží prvky jazyka UML, konkrétněji bude použit *„use case diagram"*[7]. Tento typ je vhodný spíše k popisu chování uživatelského rozhraní, proto pro specifikaci s ohledem na programování business aplikace, budou v rámci práce použity vývojové diagramy. Aplikace se taktéž neobejde bez databázového systému. Tuto funkcionalitu bude zajišťovat databáze MySQL a pro návrh je použit ERD diagram[3].

1.1 Funkční požadavky

Aplikace vyvíjené v rámci této práce musí uživatelům poskytovat určitou funkčnost. I když hlavním zaměřením je příjem a výdej zboží ze skladu, mimo tyto funkce by měl být systém schopný dalších operací a to především z důvodu bezproblémového chodu aplikace.

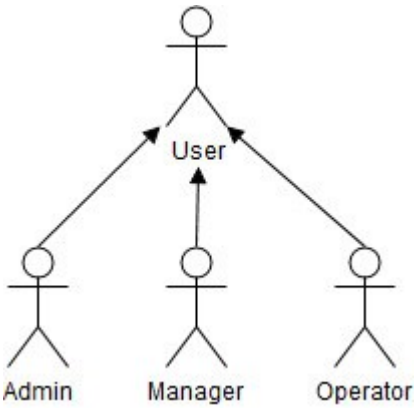
1.1.1 Funkční požadavky

- Systém bude schopný přijmout zboží na sklad.
- Systém bude umožňovat vydání zboží ze skladu.
- Systém bude evidovat pohyby zboží.
- Součásti systému bude evidovat uživatele a jejich role.
- Výdej a příjem zboží do systému bude možný přes aplikační rozhraní i pomocí android aplikace.

1.1.2 Nefunkční požadavky

- Systém bude disponovat webovým uživatelským rozhraním.
- Systém bude založen na open-source řešení.

- Pro ukládání dat bude využit databázový systém.
- Webová aplikace bude na serveru sestávajícího se z Raspberry PI.



Obr. 1. Schéma aktérů v systému

2 UŽIVATELE V SYSTÉMU

Primárním uživatelem systému bude operátor skladu. Jeho hlavní pracovní náplní bude přijímat zboží na sklad a vydávat jej ze skladu. Druhou důležitou rolí bude manažer skladu, jehož hlavní pracovní náplní bude udržovat sklad v provozu. Třetí rolí v systému bude administrátor, ten by neměl zasahovat do práce výše zmíněných rolí.

2.1 Administrátor

Role administrátora se stará o hladký chod celého systému. Jeho úkolem je spravovat uživatelské účty, nastavení systému a správu masteru zboží.

2.1.1 Přiřazené funkce

- Správa uživatelský účtů.
- Vytváření a správa masteru zboží.
- Tvorba QR kódu zboží.
- Nastavení systému.

2.2 Manažer skladu

Role manažera skladu bude specifická tím, že se bude starat o činnosti související s hladkým chodem skladového hospodářství. Především by se měl starat o stavy zboží, zakládat mastery nových druhů zboží a měnit aktuální stav zboží na skladě. V tomto případě se jedná především o dodání objednávky zboží nebo jeho odpisy.

2.2.1 Přiřazené funkce

- Měnit stav zboží na skladě.
- Vytváření a správa masteru zboží.
- Tvorba QR kódu zboží.

2.3 Operátor skladu

Operátor skladu je osoba zodpovědná za fyzické vydávání nebo přijímání zboží na sklad. Jeho pracovním nástrojem je mobilní zařízení s operačním systémem Android, které je vybavené kamerou s nainstalovanou aplikací pro snímání QR kódu a umožňující komunikaci s aplikačním serverem. Taktéž by měl být schopný online kontrolovat aktuální stav skladu.

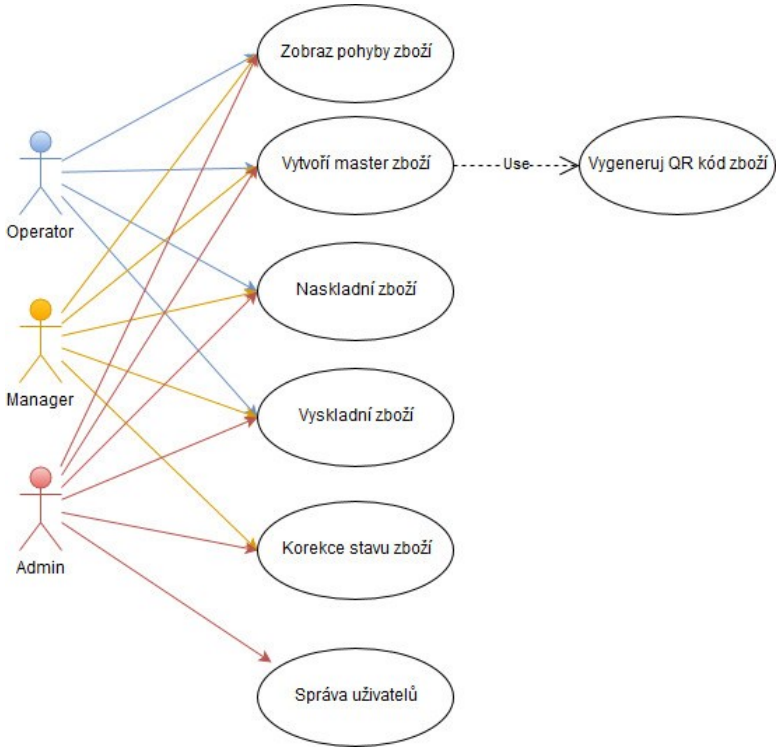
2.3.1 Přiřazené funkce

- Příjem a výdej zboží ze skladu pomocí QR kódu
- Kontrola aktuálního stavu skladu

2.4 Návrh systému z pohledu uživatele

S rostoucí komplexností informačních systémů je nutné při jejich návrhu postupovat systematicky a využívat k tomu vhodné nástroje. Tyto nástroje jsou neocenitelné, zejména pokud se pracuje na projektu v týmu. Sloučení několik metodik pro tvorbu návrhů informačních vznikl v 90. letech standart UML (Unified Modeling Language). UML představuje souhrn několika grafických notací, který je používán pro vývoj softwaru a tento formát se dnes považuje v této problematice za standart [7].

Pro grafické znázornění funkcionality popsané v této kapitole byl vytvořen doménový model, který je výsledkem hledání analytických tříd. Ty byly vytvořeny zjednodušenou analýzou procesů, souvisejících se skladovým hospodářstvím [8].



Obr. 2. Use Case Diagram

3 DATOVÝ MODEL

V případě tvorby pokročilejší aplikace, a pokud jsou na aplikaci kladeny požadavky na ukládání stavů a dat, je nutné použít některý z databázových systémů. Databáze nám představuje organizovaný souhrn dat. Podrobněji se může jednat o kolekci schémat, tabulek, dotazů, reportů, pohledů a jiných objektů [3].

V procesu modelování databáze procházíme třemi fázemi, od konceptuálního modelu k logickému a fyzickému modelu. Konceptuální model je zobecnění, v němž se nestaráme o hardwarovou a softwarovou závislost [13].

Pojem databázový systém sdružuje údaje uložené a spravované v databázi, i software pro přístup k těmto údajům (SŘBD[1]). Databázové systémy mohou být rozděleny do několika skupin [3]:

1. Hierarchické a sítové – aplikační programy závislé na databázi, obtížná údržba [3].
2. Relační – neprocedurální manipulace s daty, ukládání dat s pevnou strukturou (tabulky) [3].
3. Objektové – používají se složité datové struktury a pravidla vycházející z Business rules [3].

Systém vytvářený v rámci této práce bude používat k uchování a manipulaci s daty databázový systém MySQL. To je systém uplatňující relační databázový model a proto je vhodné zvolit při návrhu databáze nebo konceptuálního modelu entitně relační diagram (ERD) [13].

3.1 E-R model

V logickém smyslu jsou entity ekvivalentem gramatických podstatných jmen, například zaměstnanci, oddělení, měrná jednotka, apod. Entitu lze definovat pomocí vlastností, které jsou nazývány atributy. Relace je potom slovesným vyjádření vztahu mezi entitami, např. členství ve skupině, nákup, objednání zboží apod. Vztah lze dále definovat podle počtu přidružených objektů, známých jako stupeň nebo kardinalita vztahu [3].

---

[1] SŘBD – systém řízení báze dat, je speciální software umožňující přístup k údajům uloženým v databázi, aplikační program nebo uživatel, díky němu nemusí znát fyzickou strukturu uložení dat. Komunikace uživatele nebo aplikace probíhá pomocí jazyka SQL.

3.2 Entity

Pro funkčnost procesu evidence zboží, je nutné v databázi evidovat několik základních entit: goods_master, goods_movement, makers, measure_units,, operation_types, roles, user_roles a users.

3.2.1 Entita goods_master

Entita goods_master je elementární prvkem systému. Měla by obsahovat všechny atributy, které jsou zajímavé z pohledu zboží, jenž se ve skladu bude vyskytovat. Jedním z nejdůležitějších atributů v tabulce je „no_“, zde je uložen identifikátor zboží, pod kterým je položka vedena v jiných systémech. Tato hodnota by mohla být použitelná například z čárového kódu zboží, se kterým dorazí artikl do skladu.

Atributy entity:

- **id** – primární klíč, typu integer, autoincrement.
- **maker_code** – cizí klíč entity makers, typ integer, určuje informaci o výrobci/dodavateli zboží.
- **created_by** – cizí klíč entity users, typ integer, udává informaci o uživateli, který založil v systému záznam goods_master.
- **updated_by** – cizí klíč entity users, typ integer, udává informaci o uživateli, který naposledy změnil v systému záznam goods_master.
- **no_** – typ string, unikátní, jednoznačný identifikátor zboží z cizích systémů.
- **description** – typ string, editovatelný atribut, jehož obsahem by měl být popis zboží.
- **note** – typ string, pole slouží pro vkládání poznámek obsluhy skladu.
- **shelf_no_** - typ string, údaj o místě, kde je zboží běžně uskladňováno.
- **goods_group_code** – typ integer, nevlastní klíč tabulky goods_group, údaj o skupině zboží, pod kterou je artikl zařazen, slouží pro výpočet prodejní ceny zboží.
- **quantity** – typ integer, počítané pole z tabulky goods_movement, udává aktuální počet kusů skladem.
- **minimum_stock** – typ integer, údaj pro určení minimální počtu kusů zboží skladem, slouží pro indikaci bodu přiobjednání zboží.
- **maximum_stock** – typ integer, údaj pro určení maximálního počtu kusů zboží skladem, slouží pro indikaci stavu, kdy by se zboží nemělo objednávat.

- **created_at** – typ datetime, údaj pro evidenci, kdy byl záznam vytvořen.
- **updated_at** – typ datetime, údaj pro evidenci, kdy byl záznam změněn.

3.2.2 Entita goods_movement

Hlavním úkolem této entity je evidovat pohyby jednotlivých artiklů zboží. Tedy po sejmutí QR kódu pomocí čtecího zařízení a odeslaní dekódovaných dat, se zapíše řádek s příznakem příslušného pohybu.

Atributy entity:

- **id** – primární klíč, typu integer, autoincrement.
- **stored_by –** cizí klíč entity users, typ integer, určuje informaci o tom kdo provedl akci výdeje nebo naskladnění zboží.
- **id_operation_type –** cizí klíč entity operation_types, typ integer, obsahuje informaci o typu pohybu.
- **id_goods_master –** cizí klíč entity goods_master, typ integer, reference na položku zboží, před zápisem se musí zkontrolovat, zda je položka zboží vytvořena v tabulce goods_master.
- **puchase_price –** typ decimal, přenesená nákupní cena zboží z QR kódu.
- **sell_price –** typ decimal, prodejní cena zboží, hodnota se vypočítá na základě hodnoty goods_group_code a nákupní ceny zboží.
- **quantity** – typ integer, počet jednotek zboží přenášeny z QR kódu v aktuálním pohybu.
- **created_at** – typ datetime, údaj pro evidenci, kdy byl záznam vytvořen.
- **updated_at** – typ datetime, údaj pro evidenci, kdy byl záznam změněn.

3.2.3 Entita users

Entita users nám shromažďuje informace o uživatelích v systému. Hlavní uplatnění najde především při autorizaci uživatelů v procesu přihlášení do systému. Dále by mělo být díky ní možné v systému ukládat informace pro určení, kteří uživatelé a kdy vytvořili nebo editovali záznamy v entitách goods_master a goods_movement.

Atributy entity:

- **id** – primární klíč, typu integer, autoincrement.
- **name** – typ string, obsahuje jméno uživatele.

- **email** – typ string, obsahuje emailovou adresu uživatele, na základě tohoto atributu je uživatel při první fázi procesu autentizace v databázi vyhledán, kvůli nalezení přístupového hesla, takže tato hodnota by měla být v záznamech entity unikátní.
- **password** – typ string, atribut slouží pro uložení hashované hodnoty hesla, na základě znalosti správného řetězce, je uživatel v druhé fázi procesu autentizace přihlášen do systému.
- **remember_token** – typ string, uživatel má možnost nechat systém si jeho přihlášení zapamatovat, pokud tak učiní je do paměti prohlížeče vygenerován soubor cookie s hashovanou hodnotou. Pokud je při dalším přístupu k aplikaci tento soubor v paměti prohlížeče stále přítomen, proběhne ověření proti tomuto atributu v databázi a uživatel je automaticky přihlášen.
- **created_at** – typ datetime, údaj pro evidenci, kdy byl záznam vytvořen.
- **updated_at** – typ datetime, údaj pro evidenci, kdy byl záznam změněn.

3.2.4 Další entity

Další entity v systému doplňují informace o výše uvedených entitách. Relace mezi users,user_roles a roles, nám v modelu vytváří vazbu N:M, díky které může jeden uživatel mít v systému několik rolí a obráceně jedna konkrétní role může být přidělena mnoha uživatelům.

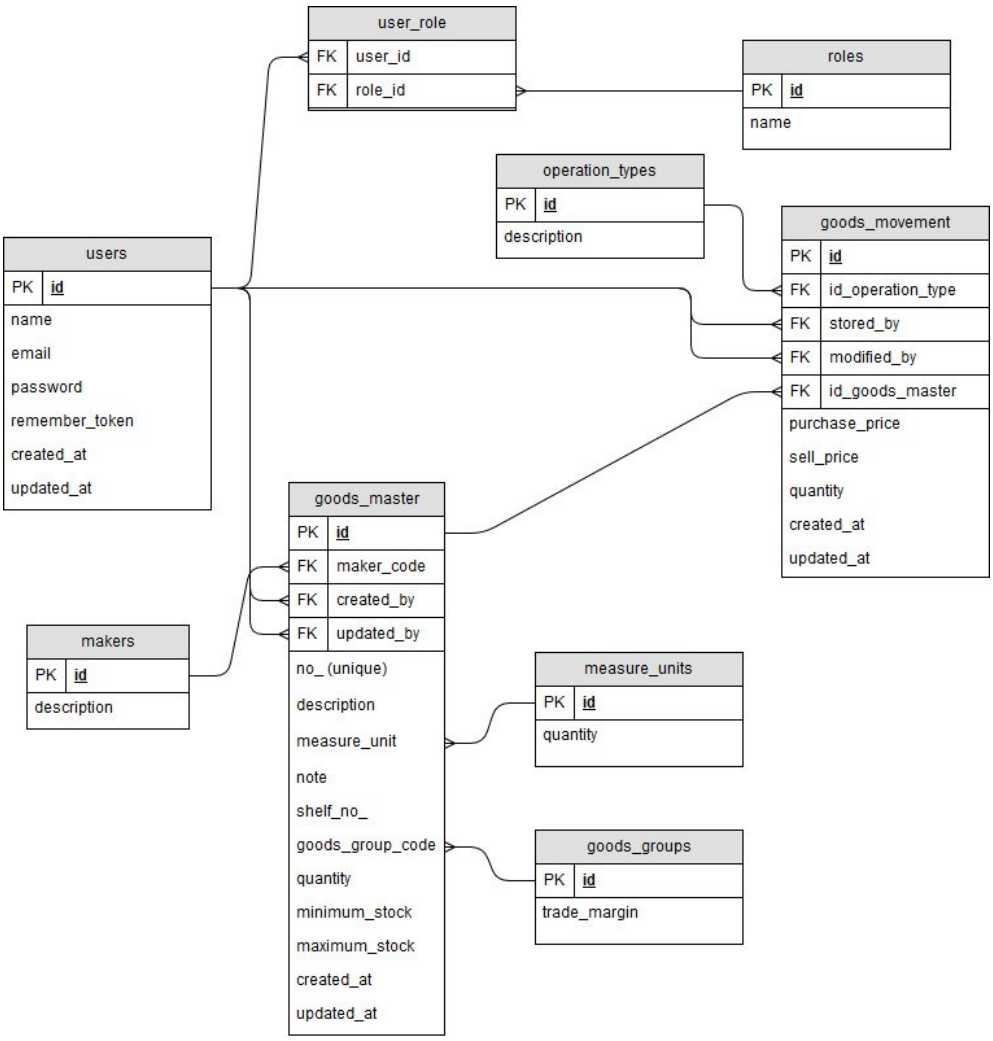
Entita operation_types je číselník typů operací, které jsou důležité pro evidenci pohybu zboží v goods_movement, díky oboru hodnot který obsahují, je možné rozeznat, o jaký druh pohybu se jedná, například výdej zboží, příjem zboží či korekce stavu.

Jelikož v reálném světě mohou nastat případy, kdy jeden druh zboží vyrábí více výrobců, je dobré v entitě goods_master udržovat informaci o výrobci zboží. Toho je dosaženo entitou makers. Jako příklad lze uvést SoC Apple A9, kdy jej pro společnost Apple vyráběli společně Samsung a TSMC [18].

Důležitým údajem, který ovlivňuje způsob naskladnění zboží, jsou měrné jednotky zboží. To může být dodáváno různě, v kusech, litrech, kilogramech, barelech apod. Proto je dobré na tuto eventualitu myslet a udržovat entitu goods_master v relaci s measure_unit.

Pro potřeby stanovení prodejní ceny nám slouží entita. U zboží nám udává, do jaké skupiny patří a na základě této informace lze zjistit prodejní marži konkrétního artiklu. Posléze tak lze vypočítat koncovou prodejní cenu.

Navržený datový model lze znázornit grafem na Obr.3



Obr. 3 ERD diagram skladového systému

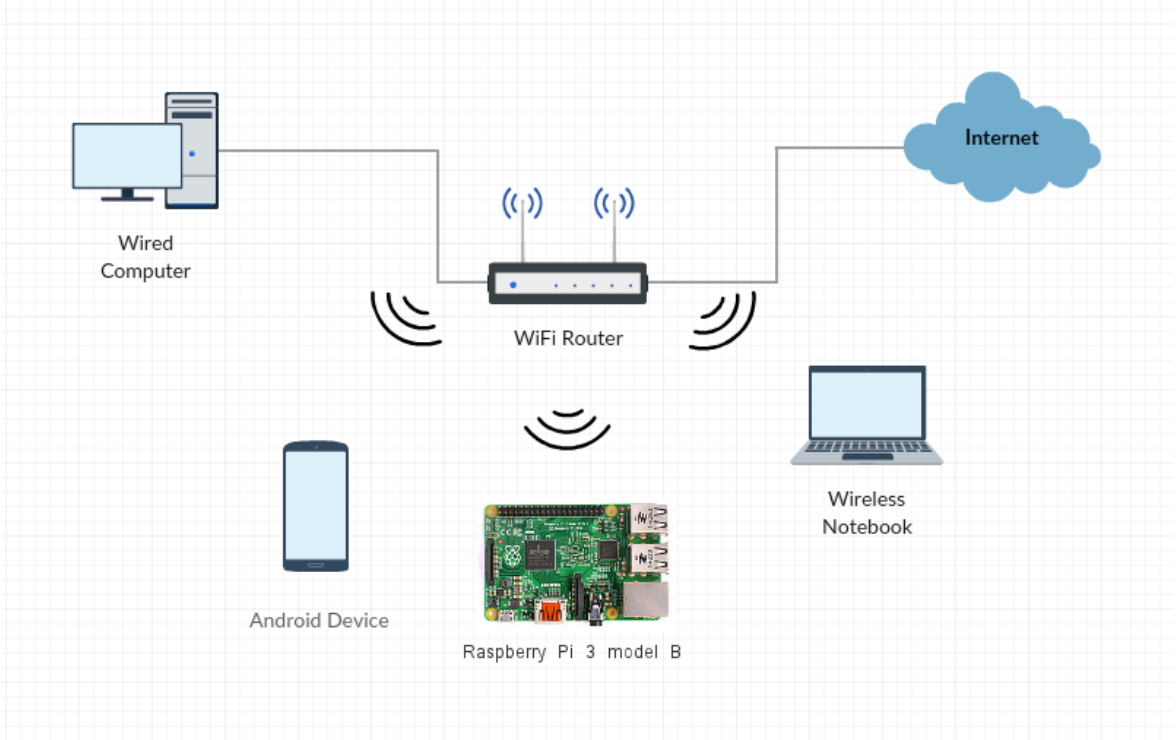
4 NÁVRH HARDWAREVÉHO ŘEŠENÍ

Návrh řešení je jedním z důležitých bodů realizace. V zadání práce byla podmínka, že systém musí být postavený na platformě počítače Raspberry Pi. Jedná se o platformu otevřenou a lehce modifikovatelnou, z čehož vyplývá široké spektrum možností realizací různých úkolů. Čtečka QR kódu by mohla být realizována přímo samotným zařízením, kdy by k počítači stačilo připojit modul s kamerou pro snímání obrazu a napsat program, který by zajišťoval zaznamenání obsahu QR kódu a jeho překladu do lidem srozumitelné podoby. K doplnění konfigurace by k počítači mohly být připojeny dotyková obrazovka a baterie, tak by se zařízení dalo považovat za jakýsi prototyp handheldu. Případně by mohlo být Raspberry Pi připojené přes rozhraní HDMI k monitoru, ke klávesnici a kameře přes rozhraní USB [19]. Takto nakonfigurované by fungovalo jako specializovaná pracovní stanice.

4.1 Realizované řešení

V rámci této práce je pro snímání QR kódu navrhnut systém založený na součinnosti několika různých zařízení. Raspberry Pi zaujímá roli aplikačního serveru, na kterém je naistalován operační systém, webový server, interpret jazyka PHP a databázový server [20].

Pro načítání QR kódu byl zvolen telefonní přístroj s operačním systémem Android. Aby tato dvě zařízení mohla mezi sebou komunikovat, bylo nutné do systému zahrnout i síťový prvek např. WiFi router. Neboť bude možné systém pro správu skladu ovládat přes webový prohlížeč, a jelikož všechny smartphony dnes disponují alespoň základním webovým prohlížečem, mohla by tato 3 zařízení tvořit teoretickou základnu celého systému. Pro pohodlnou práci obsluhy skladu budeme považovat i pracovní stanici, která může být připojena k routeru pomocí sítě WiFi nebo síťového kabelu. Další volitelnou možností je připojení celé takové lokální sítě k síti internet. To by mohlo být užitečné hlavně z důvodu pravidelných aktualizací operačního systému a aplikací nainstalovaných na Raspberry Pi nebo z důvodu synchronizace lokální databáze s databází centrální.



Obr. 4 Síťové schéma možného řešení

4.1.1 Specifikace zařízení použitých v realizovaném řešení

Je patrné, že pro chod systému je třeba několika samostatných zařízení. Nejdůležitějšími prvky jsou počítač Raspberry Pi, mobilní telefon se systémem Android a WiFi router. V této části práce budou blíže představeny hardwarové komponenty použité pro realizaci řešení.

4.1.1.1 Raspberry Pi

Jedná se jednodeskový počítač s deskou z plošných spojů a s velikostí blížící se rozměrům platební karty. První verze Raspberry Pi byla vyvinuta v roce 2012 britskou nadací Raspberry Pi Foundation. Prvotním účelem bylo podpořit výuku informačních technologií na školách a seznámit studenty, jak pomocí počítače řídit různá zařízení [4].

V únoru roku 2016 byl uveden zatím poslední model Raspberry Pi 3, který proti druhé generaci obsahuje 64 bitový procesor, taktovaný na frekvenci 1,2 GHz. CPU se nachází v čipu Broadcom BCM2837 SoC[2] a disponuje čtyřmi jádry architektury ARM Cortex-A53

[2] Zkratka z anglického slovního spojení system on chip, v podstatě se jedná o integrovaný obvod obsahující jednotlivé komponenty počítače. Procesor, operační paměť, grafický procesor a třeba i radio frekvenční

s 512KB sdílené L2 cache. Stejně jako předchůdce Raspberry Pi 2 má k dispozici 1GB LPDDR2 (900MHz) operační paměti, která je částečně sdílená s grafickou kartou Broadcom VideoCore IV [4].

Podobně jako u druhé generace ani u třetí nebylo změněno připojení ethernetového portu, který je připojen skrze rozhraní USB 2.0. Toto řešení zapříčiňuje omezení přenosové rychlosti ze sítě na přibližně 35MB/s [4][14]. Díky tomu není zařízení příliš vhodné na implementaci jako NAS, protože přenos dat byl omezen propustností USB můstku. Oproti předchozím generacím Raspberry Pi 3 přišlo s novinkou v podobě čipu Broadcom BCM43438, který mu zajišťuje konektivitu v pásmu 2.4 GHz standartu 802.11n WiFi sítě, nízkoenergetické Bluetooth a klasické Bluetooth 4.1 [4].

Technická specifikace:

- **SoC:** Broadcom BCM2837.
- **CPU:** 4x ARM Cortex-A53, 1,2 GHz.
- **GPU:** Broadcom VideoCore IV.
- **Operační paměť:** 1GB LPDDR2 (900 MHz).
- **Síťová připojení:** 10/100 Ethernet, 2.4 GHz 802.11n wireless.
- **Bluetooth:** Bluetooth 4.1. Classic, Bluetooth LE.
- **Uložiště:** microSD.
- **GPIO:** 40 - pinová hlavička, osazená.
- **Porty:** HDMI, 3.5 mm audio-video jack, 4x USB 2.0, Ethernet, Camera Seriál Interface (CSI), Display Serial interface (DSI).

4.1.1.2 Asus ZenFone 5

Jako zařízení určené pro snímání QR kódu a komunikaci s Raspberry Pi nám poslouží běžný smartphone, pro účely této práce bylo vybráno zařízení Asus ZenFone 5. Přednosti tohoto zařízení jsou kvalitní 5 palcový display, vysoký výpočetní výkon díky dvou jádrovému CPU Intel Atom Z2560 a 2GB operační paměti. Jako grafická karta byla do zařízení zvolena PowerVR SGX544MP4. Vnitřní paměť tohoto modelu je 16 GB s možnosti rozšíření pomoci micro SD slotu o dalších až 64GB. O přenos dat se starají

---

funkce jsou obsažené v jednom chipu. Tento koncept je hojně rozšířen v aplikacích pro mobilní trh, neboť takto sestrojená zařízení nejsou energicky náročná.

moduly pro HSPA (42/ 5,76 Mb/s), WiFi (802.11b/g/n) a Bluetooth (4.0) [15]. Telefon dále disponuje dnes již zcela běžnými komponentami, jako jsou fotoaparát a vestavěný GPS modul a samozřejmostí je GSM modul. Tovární systém tohoto zařízení byl Android 4.3, ten byl postupem času aktualizován na Android 5.0.

Technická specifikace:

- **Display:** TFT IPS, dotykový, kapacitní, uhlopříčka 5 palců (1 280 x 720 px).
- **Fotoaparát:** rozlišení 8 Mpx (3 264 x 2 448 px), LED dioda, autofocus.
- **CPU:** Intel Atom Z2560, 2x1,6 GHz.
- **GPU:** PowerVR SGX544MP4.
- **Operační paměť:** 2 GB RAM.
- **Síťová připojení:** Wi-Fi 802.11 b/g/n, Wi-Fi Direct, hotspot.
- **Bluetooth:** Bluetooth 4.0 Classic, A2DP, EDR.
- **GPS:** A-GPS, GLONASS.
- **USB:** microUSB 2.0.

4.1.1.3 LTE modem HUAWEI E5180

Místo klasického síťového routeru, je v řešení použit LTE modem HUAWEI E5180, který kromě LTE připojení k mobilní síti, poskytuje i vysokorychlostní bezdrátové síťové připojení (WiFi), díky čemuž může být celý systém mobilní a zároveň připojený k síti internet. Pro připojení k LTE disponuje modem slotem pro micro SIM kartu. Pokud není LTE síť k dispozici, je možné modem připojit k 3G nebo 2G sítím. Kromě bezdrátových technologií dále disponuje jedním LAN portem. Modem nedisponuje vlastní baterií, napájení je řešeno prostřednictvím napájecího adaptéru připojeného k síti 230V [16].

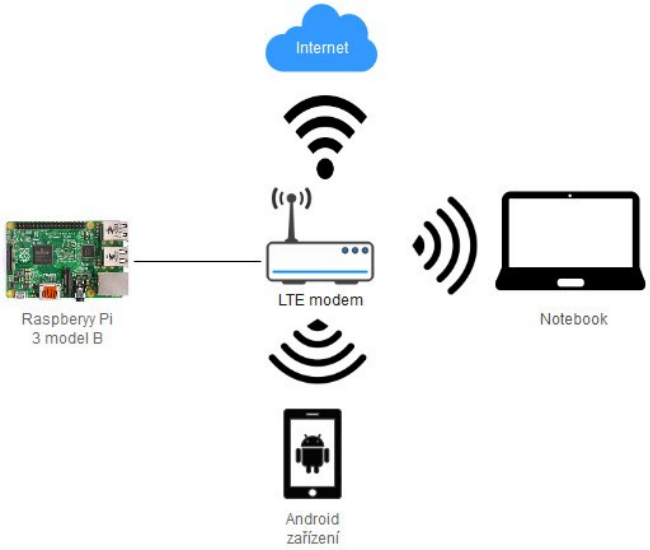
Ovládat a nastavovat zařízení je možné prostřednictvím webového prohlížeče. Zařízení je jednoduché a uživatelsky co nejpřívětivější a tak u něj nejdou nastavit některé běžné síťové funkce. V síti LAN je možné nastavit pouze DHCP a rozsah adres. Není možné nastavit rezervaci IP adresy pro určité MAC adresy. Pro Wi-Fi lze nastavit pouze SSID, kanál na kterém bude síť vysílána a metoda šifrování. Zařízení tedy není zcela ideální pro nastavení sítě LAN, ale pro nároky testování jej lze vyhodnotit jako dostačující.

Technická specifikace [16]:

- **GSM:** 850/900/1800/1900 MHz.
- **UMTS:** 900/2100 MHz (21,6/5,76 Mb/s).

- **LTE:** 800/900/1800/2100/2600 MHz (150/50 Mb/s).
- **Síťová připojení:** WiFi 802.11 b/g/n.
- **Porty:** Ethernet.

Díky LTE modemu lze zapojení celého systému ilustrovat diagramem na Obr. 5



Obr. 5 Síťový diagram

5 QR KÓD

QR kód (Quick Response code) je typ 2D čárového kódu, jenž byl vynalezen společností DENSO WAVE INCORPORATED pro automobilový průmysl v roce 1994. Kód vznikl pro potřeby označování automobilových dílů, pro něž již nestačily klasické čárové kódy [6]. S nástupem mobilních telefonů s fotoaparátem a mobilního internetu, se stávají QR kódy oblíbeným prostředkem pro šíření informací a výborným doplňkem zejména reklamy. Hodí se všude, kde je třeba předat rychle větší množství informací.

Název quick response neboli „rychlá odezva“ napovídá, že kód je možné velice rychle detekovat a přečíst a to dokonce při jeho libovolném otočení nebo vysokému náklonu vůči čtecímu zařízení. Velikost samotného kódu je definovaná ve 40 verzích (od 1 do 40). Zároveň je kód navržen tak, aby byl alespoň z části odolný proti poškození. K tomuto účelu se běžně využívají technologie samoopravných kódů. V případě QR kódu to jsou Reed-Solomonovy kódy. Kód je složen z několika vrstev, které používají různé algoritmy a slouží různým účelům [17].

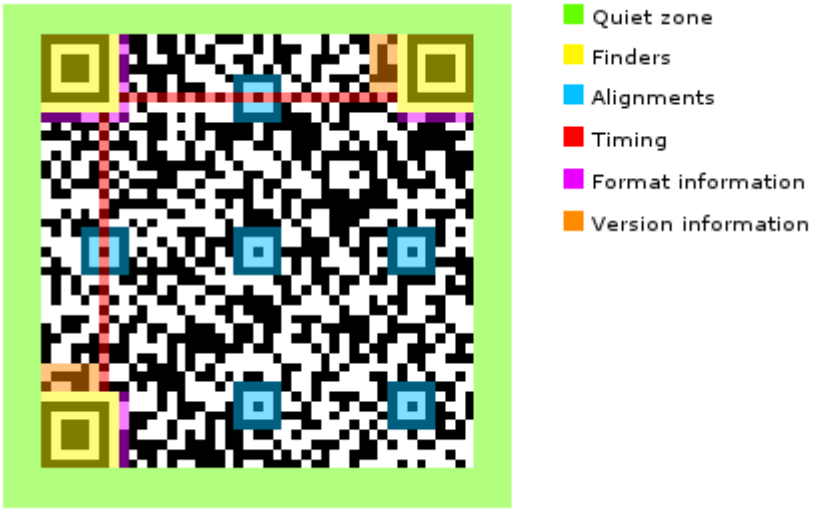
5.1 Struktura QR kódu

K přesné lokalizaci geometrických pozic v kódu slouží geometrická vrstva. Základ tvoří jednobarevná část šířky alespoň 4 body ohraničující celý kód tzv. „tichá zóna“. Rozměr modulů je při dekódování určen podle čtverců velikosti 7x7, které se nacházejí ve 3 rozích kódu. Uvnitř čtverce je bílý obvod 5x5, vnitřek je potom 3x3. Tyto čtverce bývají od ostatního kódu odděleny bílým proužkem separátorem 8x8, tyto obrazce jsou specifikovány jako Finders (hlavní zaměřovací značky). Všechny verze kódu mají vyhrazen 6 řádků a sloupec na tzv. Timing vzor (synchronizační značky) o šířce jednoho modulu, v němž se mezi sebou střídají černé a bílé bity [6]. Ve verzi 1 je geometrická vrstva složena pouze z Finders a Timing, ve vyšších verzích jsou ještě použity prvky tzv. Alignments (vedlejší zaměřovací značky).

Alignments jsou podobně jako Finders tvořen třemi soustřednými čtverci. Vnější je černý o velikosti 5x5, uprostřed je bílý 3x3 a uvnitř je pouze 1 černý čtverec. Význam Alignments spočívá v rozdělení kódu na jednotlivé podoblasti, v nichž jsou nezávisle korigovány geometrické deformace. Toto opatření napomáhá ve čtení prohnutého nebo šikmo sejmutého kódu, kdy geografická vrstva dostatečně přesně lokalizuje středy jednotlivých bodů [17].

Kód dále musí obsahovat informaci o formátu, ta je rozmístěna okolo všech finders. Informace je prezentována 15 moduly. Avšak informace o verzi zabírá pouze 5 bitů. První dva označuji úroveň zabezpečení (01 = L, 00 = M, 11 = Q, 10 = H), další tři obsahují kód masky. Zbylých 10 bitů je dopočítáno pomocí samoopravného BCH kódu. Jelikož se jedná o klíčová data, jsou v kódu umístěna na dvou místech současně. První umístění je celých 15 bitů v celku a jsou umístěny vedle separátoru levé horní značky. Druhý výskyt je rozdělený, nultý až sedmý bit tvoří řádek pod pravou horní značkou, kdežto osmý až čtrnáctý bit tvoří sloupec napravo od levé dolní značky. Sloupec má velikost 8 bitů, uloženou je pouze 7, poslední nevyšší bit se nechává vždy černý [6].

Kódy od verze 7 a vyšší obsahují informaci o verzi. Ta je tvořena osmnácti bity, z těch jsou 6 bitů data obsahující číslo verze. Zbylých 12 bitů je dopočítáváno pomocí Golayova kódu. Podobně jako údaj o verzi je tato informace uložena na dvou místech. Data jsou uskupena do obdélníku 3x6 a umístěna jsou horizontálně nad levou dolní hlavní zaměřovací značkou a vertikálně nalevo od pravé horní hlavní zaměřovací značky. Pravou stranou vždy přiléhají k pruhu synchronizační značky [6].



Obr. 6 Struktura QR kódu verze 7

Při generováním se na plochu kódu binárně přičte jeden z osmi různých černobílých vzorů tzv. maska. Algoritmus vybírá masku tak, aby poměr černých a bílých modulů byl co nejvíce v poměru 50:50 a zároveň tak, aby nevznikaly vzory připomínající hlavní nebo vedlejší zaměřovací značky. Zároveň není maska aplikována na zaměřovací značky, synchronizační značky ani na informace o verzi a formátu [6].

5.2 Prezentace dat v QR kódech

Data v QR kódech se skládají z malých černých nebo bílých čtverečků (modulů), černý je pro binární jedničku a bílý představuje binární nulu. Samotná data v QR kódu mohou být prezentována v několika různých formátech. Tímto faktem je přímo ovlivněna maximální kapacita QR kódu. V kódu mohou být přítomna numerická data, alfanumerická data, klasický text kódovaný pomocí UTF-8 nebo znaky Kanji [17].

Dalším faktorem, který ovlivňuje maximální kapacitu kódu, je stupeň zvolené ochrany. Díky němu je čtečka schopná kód dekódovat, i když je částečně mechanicky poškozen nebo jsou pro snímání špatné podmínky. Ve specifikaci existují celkem 4 různé stupně ochrany obsahu [6]:

- **Úroveň L** – až 7% poškozené plochy.
- **Úroveň M** – až 15% poškozené plochy.
- **Úroveň Q** – až 25% poškozené plochy.
- **Úroveň H** – až 30% poškozené plochy.

Je-li zvolena největší verze kódu 40, která má obsah 177x177 modulů s nejnižším stupněm ochrany L, jsme schopní v jednom QR kódů přenést [6]:

- 7089 numerických znaků.
- 4296 alfanumerických znaků.
- 2953 bajtů binárních dat.
- 1817 japonských znaků Kanji.

V nejmenší verzi 1, která může obsahovat pouze 21x21 modulů s nejvyšším stupněm ochrany H, je kód schopny přenést [6]:

- 17 numerických znaků.
- 10 alfanumerických znaků.
- 7 bajtů binárních dat.
- 4 japonské znaky Kanji.

II. PRAKTICKÁ ČÁST

6 TVORBA SOFTWAROVÉ ZÁKLADNY PRO SYSTÉM EVIDENCE

Jedním z úkolů této práce je vytvořit softwarovou základnu pro systém evidence. Jelikož je systém rozdělen na dvě části, je třeba vytvořit aplikaci pro čtení a detekování kódu pomocí telefonu Asus, a zároveň business aplikaci, která bude dostupná z aplikačního serveru Raspberry Pi a jejím úkolem bude obsluhovat evidenci zboží. U telefonu je nejsnazší vytvořit aplikaci pro čtečku pomocí vývojového prostředí Android Studio [5]. V tomto případě je jako programovací jazyk využita Java.

V případě serverové aplikace je práce vázána na programovací jazyk PHP. Avšak psát aplikaci v čistém PHP s ohledem na bezpečnost, a aby umožňovala nám vytvořit veškerou potřebnou funckionalitu, by bylo jak časově tak programátorský náročné. Proto byl pro účely vytvoření background aplikace zvolen PHP Framework Laravel [2], který má již v sobě spoustu běžných funkcionalit připravených k použití.

V rámci této kapitoly budou popsány použité nástroje pro vývoj aplikací. Dále se kapitola bude věnovat popisu jednotlivých funkcionalit aplikací a to i za pomocí vývojových diagramů. Jelikož bylo zapotřebí, aby si obě aplikace mezi sebou mohly předat data, vzniklo na straně serverové aplikace jednoduché API, jehož funkčnímu popisu se tato kapitola taktéž bude věnovat.

6.1 Návrh datového obsahu QR kódu

V rámci této práce slouží QR kódy jako nositelé informací důležitých pro vytvoření skladového pohybu. Podle druhů informací se lze setkat s několika druhy QR kódu. V obrázku mohou být uloženy např. kontaktní informace nebo webová stránka. V případě těchto kódů se prakticky jedná o text v nějaké datové struktuře. V případě kontaktu může být text ohraničen tagy ***BEGIN:VCARD*** a ***END:VCARD***, kde se mezi těmito značkami nachází další atributy, kterými mohou být jméno, telefon, popřípadě e-mail. Takto strukturovaný kód dokáže většina dostupných QR čteček rozpoznat a uživateli přímo uložit kontaktní informace do telefonu.

Pro potřeby evidence zboží byl navržen formát QR kódu, kdy jsou jednotlivé atributy jednoduše uloženy v přesném pořadí za sebou a rozděleny speciálním znakem („|“). Při generování QR kódu tak místo komplexní datové struktury vzniká spíše datová věta, jejíž struktura by se dala vyjádřit následovně:

```
|<1>|<2>|<3>|<4>|<5>|<6>|<7>|<8>|
```

- <1> - Typ operace, který chce obsluha se zbožím provést.
- <2> - Označení výrobce/dodavatele zboží.
- <3> - Číslo zboží z jiného systému (No_).
- <4> - Poznámka operátora ke skladovému pohybu.
- <5> - Označení skupiny zboží.
- <6> - Měrná jednotka zboží.
- <7> - Množství zboží ve skladovém pohybu.
- <8> - Nákupní jednotková cena zboží.

6.2 Android Studio

Android studio je oficiální IDE pro tvorbu aplikací na platformě Android. Jeho koncepce vychází z editoru IntelliJ IDEA. Zároveň jsou do něj přidány další funkce za účelem zvýšení produktivity při vytváření aplikací pro Android [9]. Android Studio lze volně stáhnout z oficiálních stránek. Je šířeno pod Apache Licence 2.0 a podporuje všechny operační systémy: Windows, Linux i OS X.

Pro vývoj na této platformě je třeba dobře chápat jazyk Java, Android API a architekturu Android aplikací. V případě platformy Android jsou převážně využívány tři podobné návrhové vzory MVC[3], popřípadě MVP[4] nebo MVVM[5] [10]. Od nich se do značné míry odvíjí struktura celého projektu a způsob vývoje v prostředí se jim podřizuje. Program pro čtečku QR kódů realizovaný v této práci do značné míry respektuje architekturu MVC.

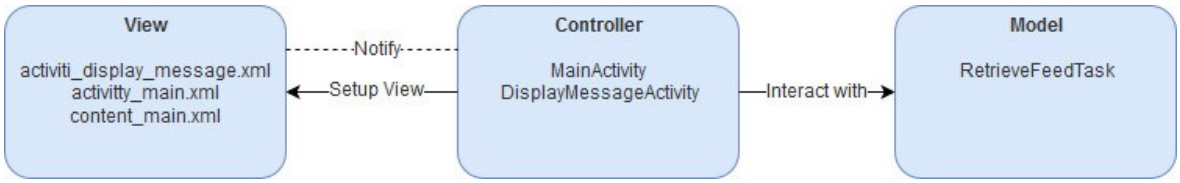
Využití operačního systému Android má své opodstatnění. V současnosti se jedná o nejrozšířenější operační systém, jak v úrovni mobilních operačních systémů, tak na úrovni operačních systému obecně. Dále pak nejsou vyvíjené aplikace vázány žádnou licencí [9]. Zároveň zařízení s tímto operačním systémem se dnes dají pořídit již za pár tisíc korun.

[3] MVC (Model – View – Controller) – návrhový vzor rozdělující aplikaci do 3 sad odpovědností.
[4] MVP (Model – View – Presenter) – podobný jako MVC, Controller je rozdělen do dvou vrstev, jedna je částečně spojná s View a druhá se chová spíše jako klasický Controller.
[5] MVVM (Model – View – Viewmodel) – vrstva model-view slouží jako konvertor hodnot mezi view a modelem. V tomto ohledu je spíše modelem a stará se o celou logiku zobrazení.

6.2.1 Aplikace QRReader

QRReader je mobilní aplikace naprogramována v prostředí Android Studio. Vzhledem k tomu, že testovací telefon disponuje operačním systémem Android 5.0, je napsána pro verzi SDK API 21. Jejím hlavním účelem je jednoduchý příjem nebo výdej zboží ze skladu.

Projekt je rozdělen do několika logicky oddělených vrstev. Prezentační vrstva se skládá z View, jejíž úkolem je komunikovat s uživatelem. View vrstvy jsou napsané v jazyce XML[9]. Od prezentační vrstvy si data berou controllery, v prostředí Android jsou nazývány Activity a jedná se o třídy napsané v jazyce Java [5]. Jejich úkolem je překládat data mezi View a Modelem. Modelovou vrstvu taktéž obstarává Activita [10]. Model je v projektu použit pouze jeden a jeho hlavním úkolem je starat se o komunikaci s API aplikačního serveru Raspberry Pi. Struktura projektu je ilustrována na obr.7



Obr. 7 Architektura projektu

Princip funkce je navržen tak, aby uživatel po otevření aplikace vybral pomocí tlačítka operaci, kterou chce se zbožím provádět. Definice View se nachází v souborech activity_main.xml [9], do kterého je vložen soubor content_main.xml. V tomto případě se může jednat o příjem nebo výdej zboží. Zvolením tlačítka se zavolá obslužná funkce v *MainActivity*. V ní dojde k nastavení příznaku zvolené operace. Následně se aktivuje kamera telefonu za účelem zachycení QR kódu.

Pro účely dekódování je do projektu přidána knihovna ZXing ve verzi 1.9. Přidání do projektu je provedeno pomocí konfiguračního souboru build.gradle (Module.app) automatizačního nástroje Gradle [24]. Ten se nachází ve složce projektu Gradle Scripts. Do oddílu *dependencies* byl přidán řádek:

```
compile 'me.dm7.barcodescanner:zxing:1.9'
```

Po automatickém překompilování projektu lze použít funkce scanneru. V aktivitě main bylo posléze možné vytvořit objekt ***ZXingScannerView*** a pomocí něj načíst obsah QR kódu.

```
mScannerView = new ZXingScannerView(this); // Inicializace objektu
setContentView(mScannerView);
mScannerView.setResultHandler(this); // Registrace sebe sama jako
handleru pro výsledek
mScannerView.startCamera(); // Spuštění kamery
```

O manipulaci s výsledkem se stará funkce ***handleResult()***, která jako parametr očekává výsledek ***rawResult***, jež by měla obdržet od scanneru. Předaná data lze potom číst z proměnné následovně:

```
rawResult.getText()); // obsahuje výsledek scanu v textové podobě
rawResult.getBarcodeFormat().toString()); // formát scanovaného kódu
```

Po úspěšném načtení QR kódu a jeho dekódování se vypne kamera. Načtená data se společně s příznakem akce přidají do proměnné ***EXTRA_MESSAGE***. Pro účely předání dat jiné aktivitě byl vytvořen objekt typu Intent, do nějž jsou vložena data z proměnné ***EXTRA_MESSAGE***. Následně se spustí nová aktivita.

```
Intent myIntent = new Intent(getBaseContext(),
DisplayMessageActivity.class);
myIntent.putExtra("Data",EXTRA_MESSAGE);
startActivity(myIntent);
```

V aktivitě *DisplayMessage* nejdříve program vybere data z objektu ***myIntent***. Validuje, zda text obsahuje alespoň jeden oddělovací znak „|“. Není-li znak přítomen, aplikace uživatele informuje, že se snažil načíst kód, který není určen pro danou aplikaci. Poté uživatele vrátí na obrazovku s výběrem akce, kterou chce provést.

Pokud je text vyhodnocen jako validní, program řetězec převede do datového pole a uživateli zobrazí informační obrazovku s daty obsaženými v QR kódu. Zjistí-li uživatel, že v datech není něco v pořádku, může se opět vrátit na úvodní obrazovku s výběrem akce.

V opačném případě stačí, když klepne na potvrzovací tlačítko *„Poslat“*. Po jeho stisknutí je zavolána funkce ***sendData()***, v níž proběhne transformace dat z pole do podoby URL. Vzápětí je řízení předáno modelové třídě *RetrieveFeedTask*. Ta očekává řetězec URL, z něhož vytvoří objekt typu URL, naváže spojení a pošle HTTP request aplikačnímu interface na webovém serveru Raspberry Pi.

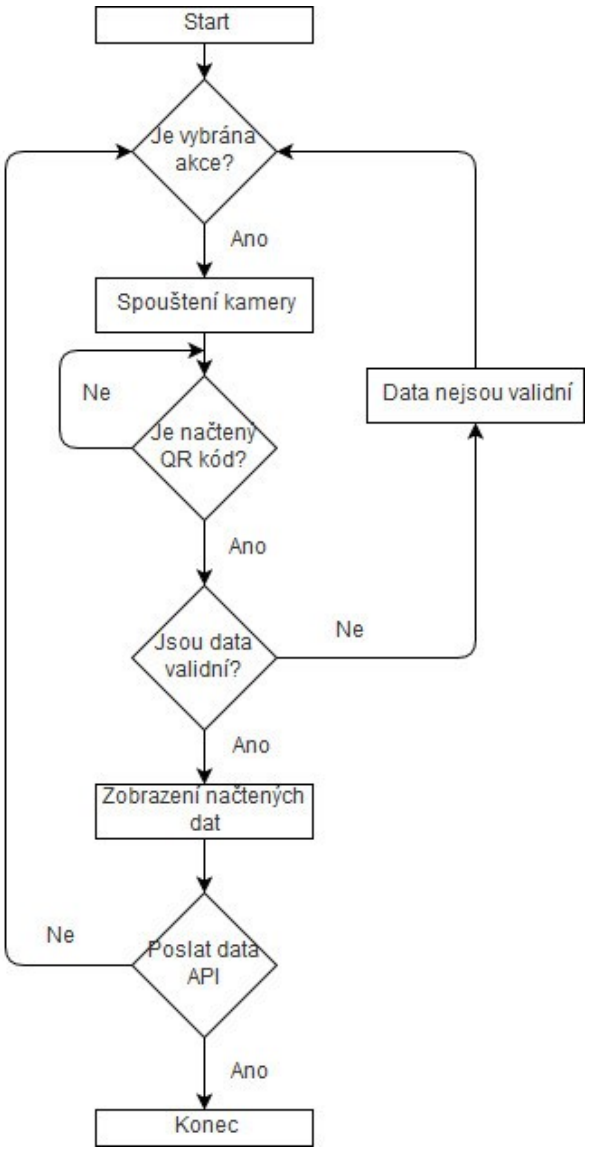
Veškerá potřebná data jsou obsažena v hlavičce HTTP requestu, tělo zůstává prázdné. Začátek hlavičky pro komunikaci s API pak může vypadat následovně:

```
GET / HTTP/1.1
Host:192.168.0.101:8080/storeitem/1/4/O125436A45/Note/2/12/3457/
Connection: keep-alive
Cache-Control: max-age=0
```

```
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.
8
```

Pro aplikační rozhraní jsou důležité údaje o typu požadavku (GET) a URL. URL obsahuje adresu, kam se má požadavek zaslat. URL se sestává z lokální IP adresy, která byla přidělena serveru Raspberry Pi z DHCP. Údaj za dvojtečkou zastupuje číslo portu, na kterém aplikace naslouchá HTTP protokolu. Údaj storeitem představuje název funkce aplikačního contolleru, který požadavek obsluhuje. Za ním již následují data oddělená znakem „/“, která jsou potřebná pro vytvoření záznamu v tabulce *Goods_movements*.

Funkčnost aplikace pro čtení QR kódu je možné znázornit následujícím vývojovým diagramem.



Obr. 8 Vývojový diagram funkce QRReaderu

6.3 Business aplikace Warehouse

Pro vývoj webové aplikace byl zvolen programovací framework Laravel. Jedná se o svobodný open-source PHP webový framework, šířený pod licencí MIT [2]. Byl vytvořen programátorem Taylorem Otwellem a první vydání se datuje k únoru 2012. Vývoj aplikace probíhá podle vzoru architektury MVC. Dnes se řadí mezi jeden z nejoblíbenějších frameworků pro vývoj aplikací v PHP, srovnat jej lze s Nette, Zend, Codeigniter nebo Yii.

Základem systému je framework Symofony, ten poskytuje Broserkit, Console, Degug a FileSystem. Pro instalaci doplňků se využívá balíčkovací systém se specializovaným manažerem závislosti Composer, který deklaruje vztahy v JSON souboru a umožňuje migraci projektu. O přístup k relačním databázím se stará Eloquent ORM (object relation mapper). Využívá návrhový vzor ActiveRecord a dokáže spravovat relační vztahy typu 1:1, 1:N a N:M. Komponenta starající se o VIEW se nazývá Blade, a díky jejímu použití je programátor schopný View jednoduše šablonovat. Veškerá syntaxe kódu se snaží být co nejvíce příjemná a lidsky čitelná tzv. Syntactitc sugar[9].

6.3.1 Autentizace a autorizace přístupu uživatelů

V Laravelu lze autentizace implementovat velice jednoduše. Ve složce projektu lze instalaci provést pomocí terminálového příkazu:

```
php artisan make:auth
```

Artisan je interface příkazové řádky distribuovaný s Laravelem. Během tvorby projektu poskytuje spoustu užitečných příkazů, kterými zjednodušuje a urychluje jeho tvorbu. V tomto případě ***make:auth*** vytvoří všechny potřebné třídy pro model, controller, middleware a view tak, aby bylo možné přihlášení uživatele do systému [21]. Dokonce je vytvořen i soubor migration, podle kterého se vytvoří tabulka *Users* v databázi. Po implementaci potom stačí v controllerech a view, ke kterým mají mít přístup pouze přihlášení uživatelé, volat konstrukce jazyka PHP.

Ověření uživatele v controlleru:

```
public function __construct()
{
      $this->middleware('auth');
}
```

Od verze 5 umožňuje PHP využívat metodu constructoru [22]. Takže pokaždé, když je vytvářen objekt obsahující metodu ***__cosnstuct*** je tato metoda zavolána. V tomto případě se pokaždé constructoru dotáže middleware *auth*, zda je uživatel do systému přihlášen.

V případě view lze ověření provést pomocí:

```
@if (Auth::guest())
// kód stránky, který uvidí nepřihlášení uživatele
@else
// kód stránky, pro přihlášené uživatele
```

Jelikož v aplikaci potřebujeme ověřovat přihlášené uživatele na úrovni přidělených rolí, byly v databázi vytvořeny další dvě tabulky *User_roles* a *Roles*. Tabulka *User_roles* je propojovací tabulka, s níž jsou v relaci tabulky *Users* a *Roles*. Díky vytvoření záznamu v nich je aplikace schopna poznat roli uživatele a podle té potom určit, do které sekce aplikace má přihlášený uživatel přistup. Aby toho bylo v aplikaci možné dosáhnout, musely vzniknout dvě třídy modelu *User* a *Role*.

Modelová třída pak obsahuje funkci ***roles()***, jejímž zavoláním je aplikace schopna přistupovat k záznamům v tabulce *User_roles*.

```
public function roles()
{
return $this->belongsToMany
('Warehouse\Role','user_roles','user_id','role_id');
}
```

Další metoda v modelu User slouží pro ověření uživatele podle jeho jména. Zkoumá zda je ke konkrétnímu jménu přidělena některá z rolí.

```
public function hasRole($role)
{
      if($this->roles()->where('name', $role)->first())
      {
            return true;
      }
      return false;
}
```

V podmínce funkce je patrné, že volá funkci ***roles()***, ve které je definována vazba na tabulku *User_roles*. Tato funkce nám ale vrátí hodnotu true nebo false v případě, že má uživatel přidělenou alespoň jednu roli. Protože je ale přidělování rolí uživateli řešeno vazbou N:M, musela vzniknout funkce pro vrácení všech přidělených rolí jednomu

uživateli. Ta je nazvána ***hasAnyRole*** a volá ***hasRole()***. Tímto mechanismem je aplikace schopná zjistit přesně, které role jsou uživateli přiřazeny.

```
public function hasAnyRole($roles)
{
      if(is_array($roles))
        {
            foreach($roles as $role)
            {
                if($this->hasRole($role))
                {
                    return true;
                }
            }
        } else
            if($this->hasRole($roles)){
                return true;
            }
        return false;
}
```

Tohoto principu je posléze využito v middleware *CheckRole*, jenž je v projektu využíván pro ověření přístupu uživatele k jednotlivým sekcím aplikace. Kód funkce middleware pro ověření přístupu je následovný:

```
public function handle($request, Closure $next)
{
      if($request->user() === null)
      {
            return response("Insufficient permissions", 401);
      }
      $actions = $request->route()->getAction();
      $roles =isset($actions['roles']) ? $actions['roles'] : null;

      if ($request->user()->hasAnyRole($roles) || !$roles)
      {
            return $next($request);
      }
      return response("Insufficient permissions", 401);
}
```

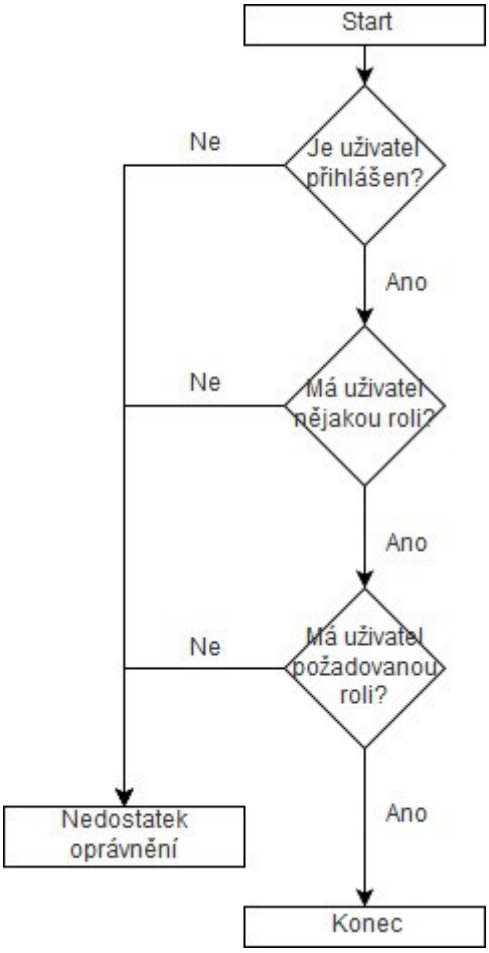
Funkce nejdřív ověří, je-li uživatel snažící se přistupovat k prvku route, do systému přihlášený. Pokud ne, přístup je mu zamítnut a zobrazí se hlášení, že uživatel nemá dostatečné oprávnění. Pokud je uživatel přihlášen, postupuje middleware ke zjištění, jaká jsou nastavení konkrétní route. Pokud Routa potřebuje ke svému přístupu od uživatele nějakou roli, přejde middleware k ověření, zda má uživatel nastavenou příslušnou roli.

Pokud tomu tak není, zobrazí se opět hlášení, že uživatel nemá dostatečné oprávnění. Jsou-li všechny podmínky splněné, umožní uživateli přístup k požadované stránce nebo funkci.

Přístup uživatelských rolí je možné potom řešit nastavením příslušných route způsobem:

```
Route::get('/admin',
[
    'as' => 'admin',
    'uses'=> 'AdminController@index',
    'middleware' => 'roles',
    'roles' =>['Admin']
]);
```

Z nastavení je patrné, že route *admin* využívá ke svému přístupu middleware *roles* a pokud k ní chce uživatel přistupovat, musí mít nastavenou roli Admin.



Obr. 9 Diagram ověření rolí uživatele

6.3.2 Karta zboží

Systém je navržen tak, aby skladové pohyby bylo možné provádět jenom v případě, že je v databázi uložen záznam s informacemi o zboží. Díky tomu není nutné tyto informace

přenášet v QR kódu. V důsledku toho je potřeba, aby byly údaje o zboží do systému nějakým způsobem zadány. Z tohoto důvodu je v aplikaci vytvořen odkaz *„Nová karta zboží“*, která je prezentována view (goods) pro vkládání údajů o zboží.

Ve view *goods* je nadefinován formulář pro zadání všech údajů potřebných pro evidenci zboží. Po vyplnění všech povinných údajů a odeslání pomocí potvrzovacího tlačítka, se data předají objektu request a řízení převezme obslužná route pomocí metody POST. Prvek Route následně zavolá obslužnou funkci ***store()*** v controlleru *Home*. Funkce v první kroku provede validaci dat, která přichází z requestu. Pokud nejsou data v požadovaném formátu nebo nebyla vyplněna všechna povinná pole, zobrazí se uživateli znovu původní formulář, naplněný původními daty s výzvou k jejich korekci. V případě, že jsou data vyhodnocena jako validní, přistoupí controller k jejich uložení do databáze. V tuto chvíli je v systému vytvořena karta zboží a je možné toto zboží naskladňovat.

6.3.3 Přehled karet zboží

Aby uživatelé systému měli přehled o již vytvořených kartách zboží, vzniklo v projektu view *„Karty zboží“*, ve kterém jsou data z modelu *Goods_master* vypsána v přehledné tabulce s důležitými informacemi. Kromě informačního výpisu je do tabulky zakomponovaný sloupec s informací o aktuálním počtu zboží skladem. Hodnotu do tohoto pole dodává funkce v modelu *Goods_movement*. Funkce ***countQuantity*** na základě poskytnutého id z tabulky *Goods_master* vrátí součet hodnot s quantity z tabulky *Goods_movements*. Takže po každém novém načtení view má obsluha k dispozici informaci o aktuálním stavu skladu.

V přehledu tohoto view jsou ve sloupci *„Akce“* nadefinovány dva odkazy. První z nich nazvaný *„Detail“* po kliknutí uživatele přesměruje do view *„goods-detail“*, které je nadefinované podobným formulářem jako view *„goods“*, pro založení masteru zboží. Detail ale navíc disponuje informacemi o datu vytvoření a změně záznamu, kdyby byl master zboží založen a změněn. Hlavním úkol tohoto view spočívá v uživatelské modifikaci karty zboží, neboť se v reálném světě běžně stává, že vlivem lidské chyby je záznam vytvořen se špatnými údaji. Jako další možná funkcionalita je v pravé části zobrazen vygenerovaný QR kód, ve kterém jsou zakódovány informace potřebné pro tvorbu karty zboží. Tento kód by mohlo být využito podobně jako u funkce pro naskladňování a vyskladňování zboží. Takže by se karta zboží dala vytvořit po načtení kódu pomocí mobilního telefonu. Funkcionalita zatím není implementována.

6.3.4 QR kód pro skladové pohyby

Pro vygenerování QR kódu pro skladové pohyby slouží druhý odkaz na stránce *goods-master*. Po kliknutí obsluhy na tlačítko nazvané *„QR“*, je stránka přesměrována na další view *goods-master-qr*, která obsahuje formulář pro tvorbu QR kódu. Většina dat je již nachystána z entity *Goods_master*, potřebné je pouze doplnit informaci o počtu kusů, nákupní jednotkové ceně zboží a volitelně poznámku obsluhy. Po vyplnění a potvrzení tlačítkem *submit*, se uživateli zobrazí view *print*. Ve view je vygenerovaný QR kód pro přenos, spolu s textovou prezentací pro kontrolu uživatele. Tento kód by obsluha mohla vytisknout a nalepit na krabici zboží, u kterého chce vytvořit skladový pohyb.

Podobně jako u aplikace na mobilním telefonu, i v tomto projektu byl pro generování QR kódu využit doplněk. Doplněk se jmenuje Simple QrCode a v podstatě se jedná o jednoduchý wrapper využívající knihovnu Bacon / BaconQrCode, což je port ZXing knihovny pro PHP [23]. Instalace proběhla pomocí zavedení balíčku do konfiguračního souboru composer.json, v sekci *„require“*. Po update composeru se do projektu doinstalovaly potřebné knihovny. Aby bylo možné používat jednoduchou syntaxi pro vykreslení QR codu, bylo ještě zapotřebí v config.app zaregistrovat provider a alias na tuto knihovnu[23]. Ve view *goods-print-qr* byl potom zavolán kód pro vykreslení QR kódu následovně:

```
{!! QrCode::encoding('UTF-8')->size(200)->generate($qrBuild); !!}
```

Volání QrCode nám vytvoří objekt, kterému jsou nastaveny vlastnosti pro kódování textu a velikost kódu v pixelech. Funkce ***generate*** podle nich vykreslí kód s daty, jež obdržela z proměnné ***$qrBuild***. Proměnná byla view předána z funkce ***printQR*** nadefinované v controlleru *Home*. Jedná se o řetězec sestavený ze směsice dat, obdržených controllerem z modelu *Goods_master* a requestu. Řetězec je sestavený následovně:

```
$qrBuild = $sep.$master->maker->id.
                $sep.$master->no_.
                $sep.$master->description.
                $sep.$request->note.
                $sep.$master->goods group id.
                $sep.$master->measure_unit_id.
                $sep.$request->quantity.
                $sep.$request->purchase_price.
                $sep;
```

Proměnná ***$sep*** je při inicializaci nastavena tak, aby obsahovala znak separatoru *„|“*. V podstatě je pro tvoření QR kódu využito prosté skládání řetězců do proměnné. Řetězec

uložený v proměnné by potom mohl vypadat následovně: *„|1|A849381|Alu kolo 17 palců|Poznámka obsluhy|1|2|4|3900|“* a tento text by byl obdržen i po dekódování QR kódu.

6.3.5 Přehled skladových pohybů

Pro sledování skladových pohybů vzniklo v aplikaci view *„goods_movements“*, které je dostupné z nabídky Skladové pohyby. Je prezentováno tabulkou, v níž každý skladový pohyb představuje jeden řádek. Hodnoty nacházející se ve sloupcích prodejní jednotková cena a celkem jsou dynamicky počítané. Prodejní jednotková cena je vyjádřena vztahem součinu nákupní ceny a marže z tabulky skupiny zboží (*Goods_groups*).

$$prodejni\ cena = purchase\_price * trade\_margin$$

Cena celkem je potom součinem prodejní ceny a počtu kusů v konkrétním skladovém pohybu. Díky těmto dvěma hodnotám má obsluha skladu přehled o aktuální hodnotě naskladněného zboží.

$$cena\ celkem = (purchasase\_price * trade\_margin) * quantity$$

6.3.6 API

Aby mezi sebou mohly android a business aplikace komunikovat, vzniklo na straně serveru komunikační rozhraní. To je dostupné za pomocí nadefinování prvku route.

```
Route::any('storeitem/{operation}/{maker}/{no}/{note}/{group}/{quantity}/
{price}',[
    'as' => 'storeItem',
    'uses' => 'StoreItemController@storeItem'
    ]);
```

Příznak ***any*** za dvojtečkami nastavuje metodu požadavku, který routa zpracuje. V tomto případě bude route přijímat všechny typy http verb, tedy GET, POST, PUT, PATCH, DELETE a OPTIONS. V kulatých závorkách je název storeitem, kterým voláme funkci API v URL. Prvky ve složených závorkách jsou potom proměnné, které prvek route předá funkci ***storeItem()*** v controlleru ***StoreItem***.

Funkce ***storeItem()*** očekává všechny proměnné z route jako svoje parametry, pokud není některý nastaven, jsou inicializovány s hodnotou ***null***. V dalším kroku přistupuje k validaci jednotlivých atributů, zda jsou všechny inicializovány s rozdílnou hodnotou než je právě

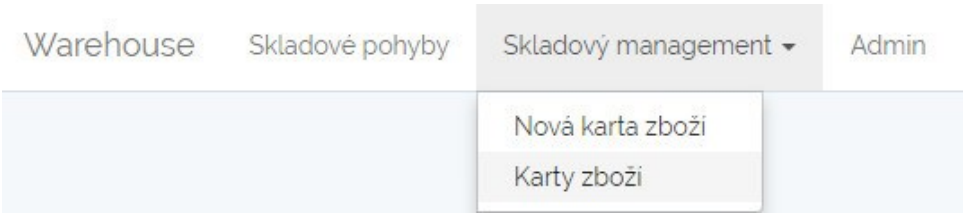
null. Pokud je tato podmínka splněna, postupuje funkce ověřování, jestli jsou hodnoty uložené v proměnných ***$maker***, ***$group***, ***$quantity*** a ***$price*** čísla. Pokud ano, je splněna podmínka integritního omezení databáze a funkce může vytvořit pole, které je naplněno všemi hodnotami z requestu. Po vytvoření pole se v modelu *Goods_master* vyhledá podle hodnoty uložené v proměnné no_ cizí klíč zboží. Pomocí hodnoty *trade_margin* dostupné z modelu *Goods_group* a nákupní ceny zboží uložené v proměnné ***$price*** se vypočítá prodejní jednotková cena zboží. V závěru je vytvořen objekt modelu *Goods_movement*, objekt je naplněn validovanými daty a následně proběhne zápis do databáze.

7 FUNKČNÍ TESTOVÁNÍ

V přechozí kapitole byly popsány všechny funkce, které dokáže systém vykonávat. V této kapitole se práce zaměří na funkční testování aplikace ve vytvořeném testovacím prostředí.

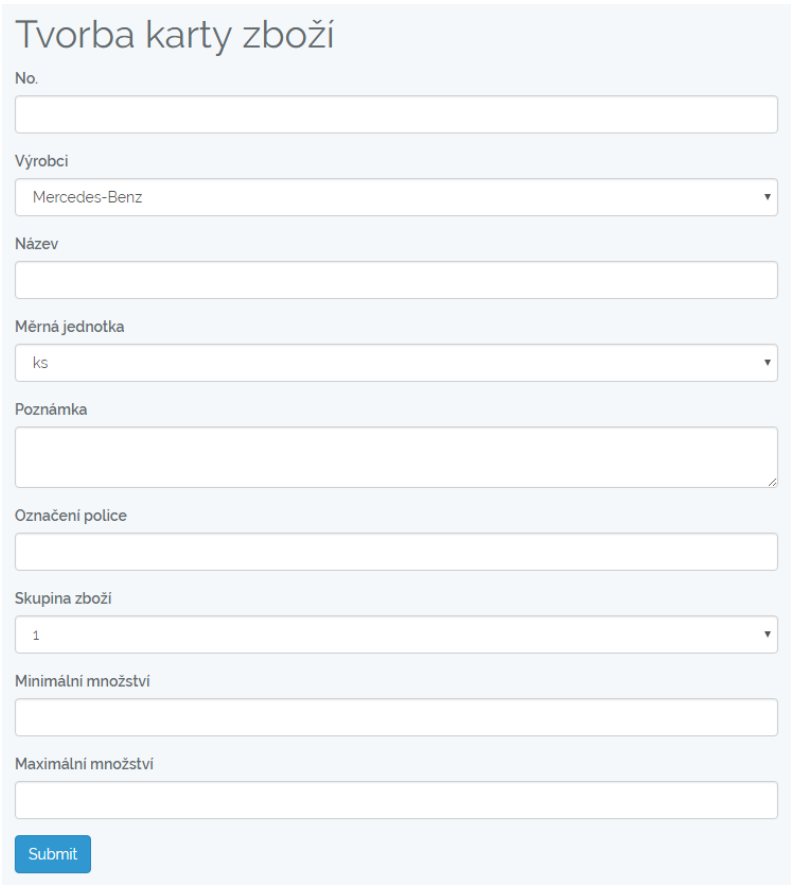
7.1 Tvorba karty zboží

Odkaz pro kartu zboží je umístěn v nabídce menu pod odkazem *„Skladový management“*. Rozklikne-li uživatel odkaz, vykreslí se drop-down prvek s podnabídkami *„Nová karta zboží“* a *„Karty zboží“*. V tomto případě je zvolena *„Nová karta zboží“*.



Obr. 10 Navigace – Nová karta zboží

Po načtení stránky se zobrazí jednoduchý formulář sloužící k vytvoření záznamu v tabulce goods_masters.

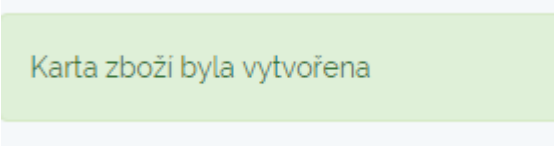


Obr. 11 Ilustrace formuláře pro založení nové karty zboží

Do formuláře byly postupně vloženy tyto údaje:

- No: O125436A452
- Výrobe: Nissan
- Název: Olej 10W40
- Měrná jednotka: l
- Poznámka: Motorový olej klasik
- Označení police: L125D4
- Skupina zboží: 2
- Minimální množství: 30
- Maximální množství 150

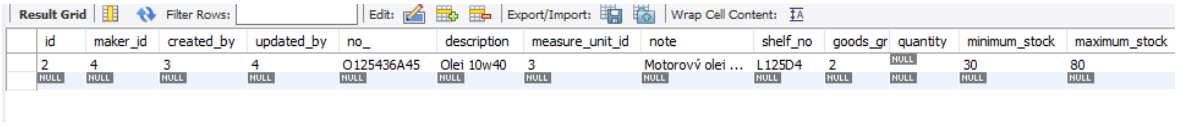
Po vložení těchto hodnot do příslušných polí, byl formulář odeslán pomocí tlačítka Submit. Stránka se načte znovu, tentokrát je formulář prázdný, připravený ke tvorbě další karty zboží. O úspěšnosti akce uložení do databáze uživatele informuje zelené pop-up okýnko v horní části stránky.



Obr. 12 Informace o úspěšnosti vytvoření karty zboží

O vytvoření karty zboží se můžeme přesvědčit dotazem přímo v databázovém systému MySQL.

```
select * from goods_masters where no_ like 'O125436A45';
```



Obr. 13 Část výsledku SQL dotazu

SQL dotaz vrátil jeden výsledek, z čehož vyplývá, že master zboží byl úspěšně založen.

7.2 Přehled karet zboží

Mimo SQL dotaz do databáze se lze také přesvědčit v druhém odkazu *„Karty zboží“*, který se nachází pod nabídkou *„Skladový management“*. Po načtení stránky se uživateli zobrazí v tabulce seznam všech karet zboží, založených v databázi skladového systému. Na obr. 14 je možné vidět, že karta zboží byla založena. Nyní pokud chceme se zbožím vytvořit

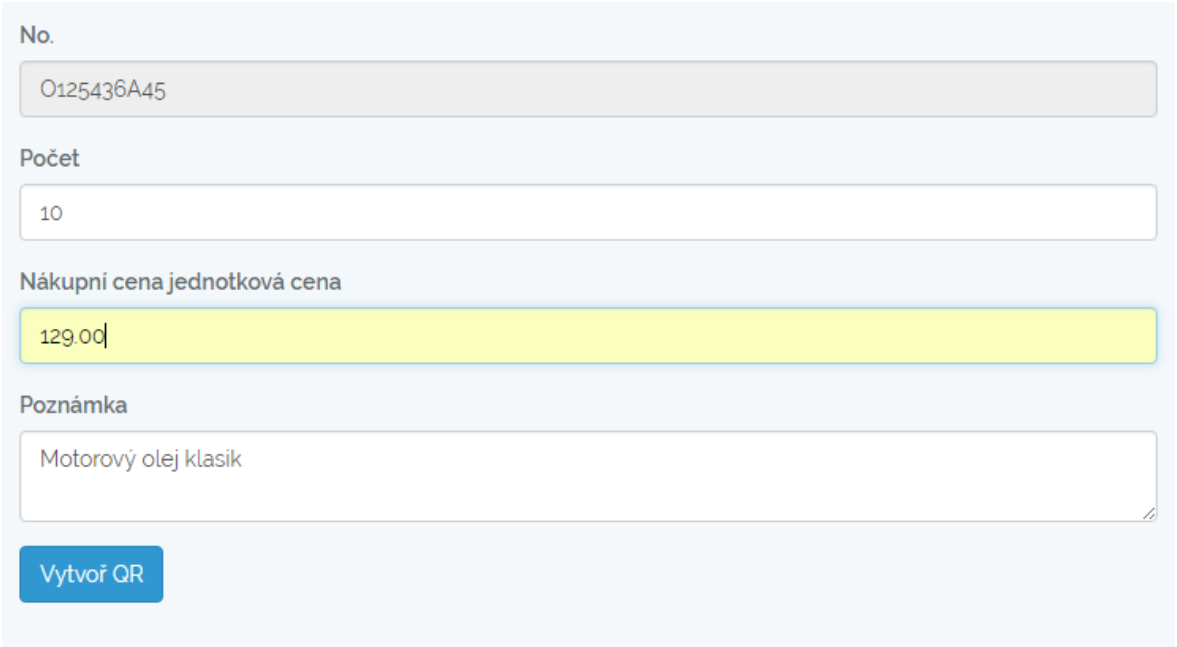
skladový pohyb, potřebuje vygenerovat příslušný QR kód. K této funkci je možné se dostat přes modré tlačítko QR v příslušném řádku karty zboží.

#	Číslo	Název	Výrobce	Uložení	zboží	Množství	jednotka	zásoba	zásoba	Vytvořil	Modifikoval	Akce
1	A00023564313	Převodovka A200 4 Matic	Mercedes-Benz	AJ564	4	2	ks	3	5	Admin	Admin	Detail QR
2	O125436A45	Olej 10w40	Nissan	L125D4	2	68	l	30	80	Admin	Manager	Detail QR

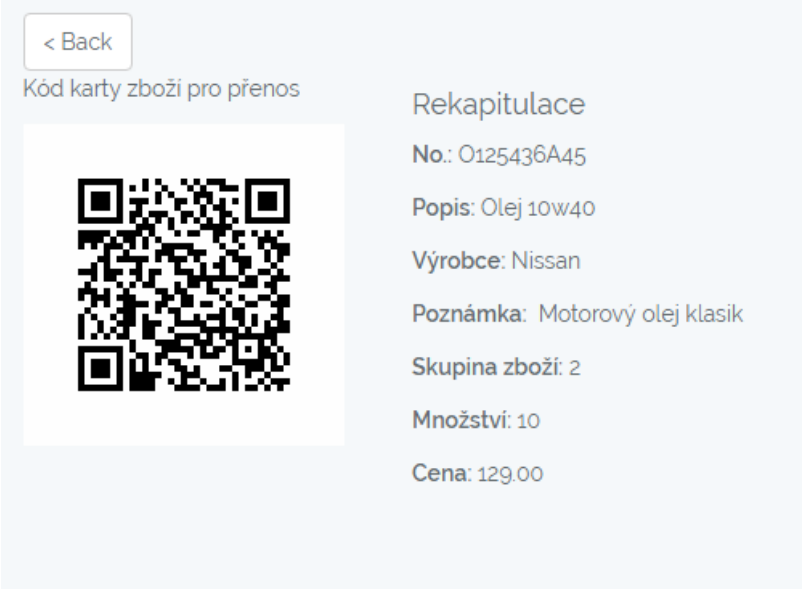
Obr. 14 Ukázka výpisu karet zboží založených

7.3 Generování QR kódu pro přenos zboží

Ve formuláři pro generování toho typu QR kódu, potřebuje doplnit pouze počet zboží, které chceme QR kódem přenést. Kvůli hodnotě za jakou chceme zboží přenášet, doplníme jednotkovou nákupní cenu. Pole poznámky je ve formuláři předvyplněné z tabulky goods_masters. Volitelně jej lze editovat, se změněnou poznámkou bude zboží zaznamenáno v řádce skladového pohybu. V testu byly doplněny hodnoty pro počet kusů 10 a pro nákupní jednotkovou cenu bylo zvoleno číslo 129.00. Po kliknutí na tlačítko *„Vytvoř QR“*, nás aplikace přesměruje na stránku s vygenerovaným QR kódem a s rychlou rekapitulací.



Obr. 15 Formulář pro tvorbu QR



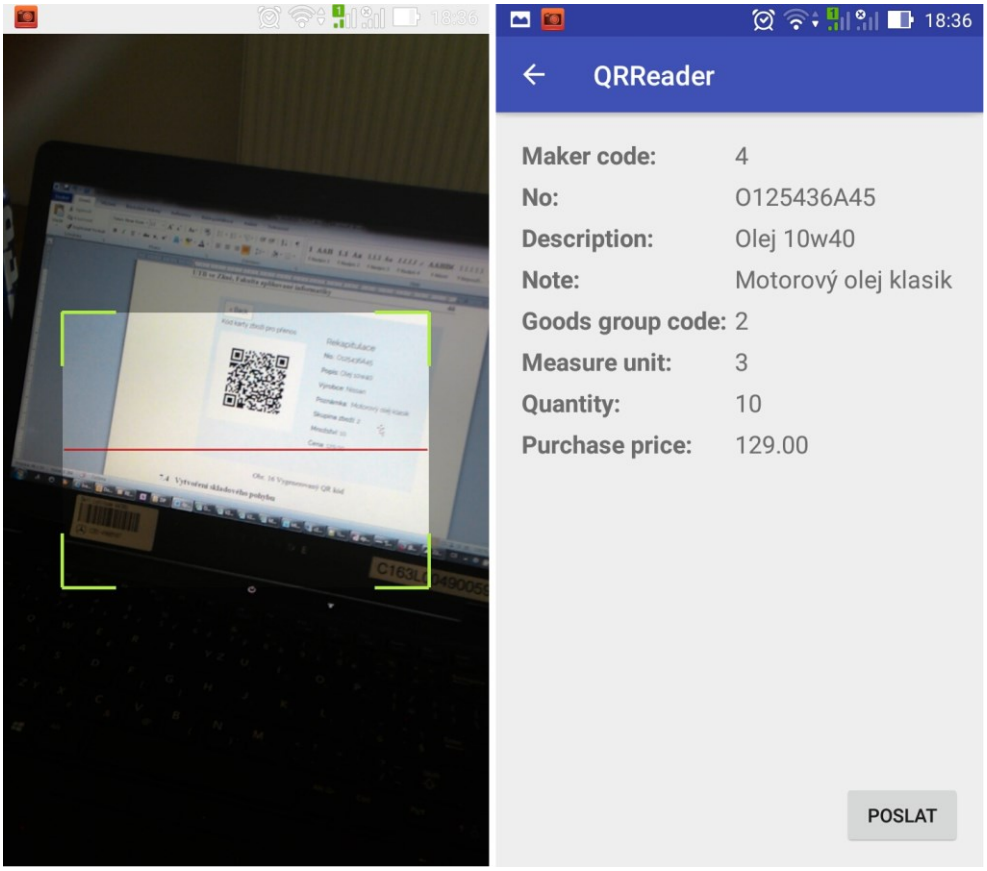
Obr. 16 Vygenerovaný QR kód

7.4 Vytvoření skladového pohybu

Nyní, když existuje vygenerovaný QR kód, můžeme použít Android aplikaci pro jeho načtení, dekódování a poslání výsledku API k vytvoření požadované operace. V zařízení telefonu vyhledáme ikonu QRReader a aplikaci spustíme. V prvním kroku zvolíme operaci pro příjem zboží.

Obr. 17 Úvodní obrazovka aplikace QRReader

Po výběru druhu požadované operace se aktivuje fotoaparát zařízení a začne se pokoušet snímat QR Kód.



Obr. 18 Proces scanování QR kódu s dekódovaným výsledkem

Po naskenování kódu se zobrazí výsledek s daty, která byla do QR kódu zakódována v aplikaci Warehouse. K vytvoření skladového pohybu nyní stačí klepnout na šedé tlačítko „*Poslat*“ vpravo dole. V rámci testu byl stejný kód načten pomocí operace pro výdej zboží ze skladu. Pro kontrolu zda se data v pořádku zapsala do databáze, slouží v aplikaci view dostupné přes odkaz *„Skladové pohyby“*. Výsledek je vyznačen na Obr. 19. Oba provedené pokusy o tvorbu skladového pohybu jsou zaznamenány v tabulce goods_movements a zobrazí se ve view skladových pohybů. Funkční test tak lze prohlásit za úspěšný.

#	Operace	No.	Název	Počet	Jednotka	Nákupní cena	Prodejní cena	Celkem	Vytvořil	Vytvořeno
1	PRI	A00023564313	Převodovka A200 4 Matic	1	ks	1390.00	1737.5	1737.5	Admin	-0001-11-30 00:00:00
2	PRI	O125436A45	Olej 10w40	2	l	1390.99	1856.97165	3713.9433	Operator	2017-05-15 14:52:41
3	PRI	O125436A45	Olej 10w40	10	l	99.00	132.165	1321.65	Operator	2017-05-15 16:49:31
4	PRI	O125436A45	Olej 10w40	12	l	3457.00	4615.095	55381.14	Operator	2017-05-16 07:57:31
5	PRI	O125436A45	Olej 10w40	12	l	3457.00	4615.095	55381.14	Operator	2017-05-16 08:15:39
6	VYD	O125436A45	Olej 10w40	12	l	3457.00	4615.095	55381.14	Operator	2017-05-16 08:17:07
7	PRI	A00023564313	Převodovka A200 4 Matic	1	ks	12865.00	16081.25	16081.25	Operator	2017-05-16 08:30:49
8	PRI	QLT131A	Hadicová spona	500	ks	2.43	3.78594	1892.97	Operator	2017-05-16 08:34:28
9	PRI	O125436A45	Olej 10w40	10	l	129.00	172.215	1722.15	Operator	2017-05-16 18:36:30
10	VYD	O125436A45	Olej 10w40	10	l	129.00	172.215	1722.15	Operator	2017-05-16 18:36:59

Obr. 19 Vyznačený výsledek zobrazený na stránce skladových pohybů

7.5 Specifikace omezení řešeni

Řešení jako takové se v laboratorních podmínkách jeví rychlé a efektivní. Jelikož je serverová část postavená na Raspberry Pi, bylo by zajímavé sledovat jeho vytížení, pokud by se v systému pohybovalo současně několik uživatelů, vytvářejících v jeden moment spousty různorodých požadavků.

V oblasti vytváření skladových přenosů se jeví jako největší problém zakódovaní cizích klíčů (maker, goods_group, measure_unit) databázových entit. V případě, že obsluhujeme jeden systém, není tento způsob řešení na škodu. Problém by mohl nastat v případě, že bychom měli zároveň několik poboček postavených na stejném řešení. Data v databázích se mohou drobně lišit, a tak co znamená např. identifikátor výrobce v jedné databázi, nemusí znamenat to samé v jiné. Tento problém by šlo odstranit dalším serverem, který by měl na starosti správu číselníkových tabulek. Ostatní zařízení by se jej mohla dotazovat a svá data navzájem zprostředkovaně synchronizovat.

ZÁVĚR

Výstupem práce je systém pro čtení, dekódování a evidenci skladových pohybů prostřednictví QR kódu. Evidence zboží pomocí QR kódu se z pohledu této práce jeví jako praktické využití této technologie, neboť se díky nim dají v jednom pohybu přenášet celé datové struktury s množstvím informací.

Stěžejní část čtečky zpracovávající obraz z kamery staví na využívání algoritmu z knihovny ZXing. Aplikace pro smartphony se systémem Android, nabízí jednoduché uživatelské rozhraní a díky využité knihovně i dostatečnou funkčnost.

Server je miniaturní počítač s nízkou spotřebou energie, ale dostatečným výkonem a počtem vstupů a výstupů. Navržený systém by mohl najít své uplatnění jako mobilní sklad. Zajímavou myšlenkou by mohlo být využití systému pro evidenci při rozvozu zboží dodávkou, tedy by mělo funkci pomyslného chytrého skladu. Dalším možným způsobem využití by mohla být evidence vypůjčených automobilu v autopůjčovně nebo výdej nových vozidel ze skladu u dealerů. Možností po určitých modifikacích by mohlo být zcela určitě spousta.

Podobně jako u programování čtečky bylo i u serverové aplikace využito knihovny pro tvorbu QR kódu, shodou okolností se jedná o port stejné knihovny pro PHP. A během testování se nestalo, že by při kódování a dekódování nastala nějaká nesrovnalost.

V rámci práce bylo zajímavé programovat dvě aplikace ve dvou podobných architekturách, ale v úplně rozdílných prostředích a programovacích jazycích. V prostředí Android Studio je až překvapivé, jak si člověk může rychle základy programování osvojit. Prostředí programátora neustále vede a programování se tím stává intuitivní. V případě frameworku Laravel, bylo trochu těžší si zvyknout na všechny nástroje, které upravují styl programování a měly by jej zjednodušovat. Nicméně pokud se jim uživatel vyloženě nebrání, přenáší jejich používání hned několik výhod. Kód je přehledný a v pozdějších fázích řešení problému není velký problém jej agilně upravovat.

Do systému by bylo dobré dodělat možnost tisknout štítky QR kódu v nějakém běžném formátu. Do grafického rozhraní přidat některé, dnes již běžné prvky, jako vyhledávání pomocí jednotlivých sloupců, drobečkovou navigaci nebo prvky pro stránkování výpisu. Za pozornost by určitě stálo napsat druhé API, kterým by bylo možné do aplikace vytvářet karty zboží z QR kódu, princip by byl dost podobný jako u skladových pohybů.

SEZNAM POUŽITÉ LITERATURY

[1] GROS, Ivan. *Kvantitativní metody v manažerském rozhodování*. Praha: Grada, 2003. ISBN 80-247-0421-8.

[2] MATULA, Terry. *Laravel application development cookbook*. Birmingham: Packt Publishing, 2013. ISBN 9781782162834.

[3] LACKO, Ľuboslav. *SQL: hotová řešení*. Brno: Computer Press, 2003. K okamžitému použití. ISBN 80-7226-975-5.

[4] UPTON, Eben a Gareth HALFACREE. *Raspberry Pi: uživatelská příručka*. 2., aktualizované vydání. Přeložil Jakub GONER. Brno: Computer Press, 2016. ISBN 978-80-251-4819-8.

[5] LACKO, Ľuboslav. *Vývoj aplikací pro Android*. Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.

[6] KUBÍK, Martin. *Pokročilá čtečka QR kódu pro Android*. 2014.

[7] VRANA, Ivan. *Projektování informačních systémů s UML*. V Praze: Česká zemědělská univerzita, Provozně ekonomická fakulta, 2008. ISBN 978-80-213-1817-5.

[8] BUCHALCEVOVÁ, Alena a Iva STANOVSKÁ. *Příklady modelů analýzy a návrhu aplikace v UML*. Praha: Oeconomica, 2013. ISBN 978-80-245-1922-7.

[9] MCCOOL, Shawn. *Laravel starter: the definitive introduction to the Laravel PHP web development framework*. Birminghan, UK: Packt Pub., 2012.

[10] HERODEK, Martin. *Android: jednoduše*. 2. aktualiz. vyd. Brno: Computer Press, 2014. Naučte se za víkend (Computer Press). ISBN 978-80-251-4298-1.SS

[11] JAKUBKOVÁ, Iva. *JUST-IN-TIME - přínosy, negativní stránky, uplatnění v podnikové praxi*. Brno, 2003.

[12] REST: architektura pro webové API. *Zdrojak.cz* [online]. Praha: Devel.cz Lab, 2009 [cit. 2017-03-23]. Dostupné z: https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/

[13] GILMORE, W. J. *Velká kniha PHP 5 a MySQL: kompendium znalostí pro začátečníky i profesionály*. Nové, 3. vyd. Přeložil Jan POKORNÝ. Brno: Zoner Press, 2011. Encyklopedie Zoner Press. ISBN 978-80-7413-163-9.

[14] UPTON, Eben a Gareth HALFACREE. *Raspberry Pi: uživatelská příručka*. Brno: Computer Press, 2013. ISBN 978-80-251-4116-8.

[15] Asus Zenfone 5 A501CG. *GSMArena.com* [online]. GSM Arena, 2015 [cit. 2017-04-12]. Dostupné z: http://www.gsmarena.com/asus_zenfone_5_a501cg-7160.php

[16] LTE modem HUAWEI E5180: RYCHLÝ START. *T-mobile.com* [online]. [cit. 2017-04-15]. Dostupné z: https://www.t-mobile.cz/dcpublic/Huawei_E5180_rychlystart.pdf

[17] *Biologické a edukační uplatnění QR kódů* [online]. Praha, 2014 [cit. 2017-05-03]. Dostupné z: https://www.google.cz/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&ved=0ahUKEwiSstrE2YXUAhWJGCwKHfBxB1EQFgg5MAM&url=https%3A%2F%2Fis.cuni.cz%2Fwebapps%2Fzzp%2Fdownload%2F130125838&usg=AFQjCNFFNg78pGbcGjA78KgMD5tBEpbpyA&sig2=NGlGPD1s1x81viZkIu3jGw&cad=rja. Bakalářská práce. Univerzita Karlova. Vedoucí práce PhDr. Petr Novotný, Ph.D.

[18] *The Apple iPhone 6s and iPhone 6s Plus Review* [online].[cit. 2017-05-15]. Dostupné z: http://www.anandtech.com/show/9686/the-apple-iphone-6s-and-iphone-6s-plus-review/2

[19] QR Code on Raspberry Pi - Hackster.io. *Hackster.io - The community dedicated to learning hardware.* [online]. Dostupné z: https://www.hackster.io/faweiz/qr-code-on-raspberry-pi-5f6764

[20] Setting up an Apache Web Server on a Raspberry Pi - Raspberry Pi Documentation. *Raspberry Pi - Teach, Learn, and Make with Raspberry Pi* [online]. Dostupné z: https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md

[21] Authentication - Laravel - The PHP Framework For Web Artisans. *Laravel - The PHP Framework For Web Artisans* [online]. Copyright © Taylor Otwell. [cit. 17.05.2017]. Dostupné z: https://laravel.com/docs/5.4/authentication

[22] PHP: Constructors and Destructors - Manual . *PHP: Hypertext Preprocessor* [online]. Copyright © 2001 [cit. 18.05.2017]. Dostupné z: http://php.net/manual/en/language.oop5.decon.php

[23] SimpleSoftware . *SimpleSoftware* [online]. Copyright © 2014 [cit. 23.05.2017]. Dostupné z: https://www.simplesoftware.io/docs/simple-qrcode

[23] SimpleSoftware . *SimpleSoftware* [online]. Copyright © 2014 [cit. 23.05.2017]. Dostupné z: https://www.simplesoftware.io/docs/simple-qrcode

[]24] GitHub - dm77/barcodescanner: Barcode Scanner Libraries for Android. The world's leading software development platform · GitHub [online]. Copyright © 2017 [cit. 23.05.2017]. Dostupné z: https://github.com/dm77/barcodescanner

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

2D	Dvoudimenzionální.
2G	Druhá generace.
3G	Třetí generace.
API	Application Programming Interface.
DHCP	Dynamic Host Configuration Protocol.
ERD	Entity–relationship diagram.
GPS	Global Positioning Systém.
GSM	Groupe Spécial Mobile.
HDMI	High-Definition Multimedia Interface.
HSPA	High Speed Packet Access.
HTTP	Hypertext Transfer Protocol.
HW	Hardware.
IDE	Integrated Development Environment.
JSON	JavaScript Object Notation.
LAN	Local Area Network.
LTE	Long Term Evolution.
MAC	Media Access Control.
MIT	Massachusetts Institute of Technology.
MVC	Model-View-Controller
NAS	Network Attached Storage.
ORM	Object-relational mapping.
PHP	Hypertext Preprocessor.
QR	Quick Response.
REST	RESTful API.
SDK	Software development kit.

SoC	System on Chip.
SŘDB	Systém řízení báze dat.
SSID	Service Set Identifier.
UML	Unified Modeling Language.
URL	Uniform Resource Locator.
USB	Universal Serial Bus.
UTF-8	UCS/Unicode Transformation Format.
WiFi	Wireless Fidelity.
XML	eXtensible Markup Language.

SEZNAM OBRÁZKŮ