TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Webová aplikace pro skladové hospodářství

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **František Jukl**
Vedoucí práce: Mgr. Jiří Vraný, Ph.D.

Liberec 2019

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Web application for storage facility management

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **František Jukl**
Supervisor: Mgr. Jiří Vraný, Ph.D.

Liberec 2019

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Webová aplikace pro skladové hospodářství

Jméno a příjmení: **František Jukl**
Osobní číslo: M14000035
Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie
Zadávající katedra: Ústav nových technologií a aplikované informatiky
Akademický rok: **2018/2019**

Zásady pro vypracování:

1. Seznamte se s problematikou programování webových služeb s podporou REST API a s problematikou ukládání většího množství provázaných dat.
2. Navrhněte aplikaci pro vedení skladového hospodářství, která umožní získávat data importem z dat poskytných systémem SAP a nad uloženými daty umožní provádět obvyklé skladové operace. Při návrhu se zaměřte zejména na bezpečnost systému a vyřešení konkurenčních operací obvyklých v tomto typu aplikací.
3. Navrženou aplikaci implementujte v jazyce Python a ověřte její praktickou použitelnost.

Rozsah grafických prací:	dle potřeby
Rozsah pracovní zprávy:	30 – 40 stran
Forma zpracování práce:	tištěná/elektronická

Seznam odborné literatury:

[1] GRINBERG, Miguel. Flask web development. Sebastopol, CA: O'Reilly, 2014. ISBN 1449372627.
[2] RICHARDSON, Leonard a Michael AMUNDSEN. RESTful Web APIs. Beijing: O'Reilly, 2013. ISBN 978-1449358068.

Vedoucí práce:	Mgr. Jiří Vraný, Ph.D. Ústav nových technologií a aplikované informatiky
Datum zadání práce:	18. října 2018
Předpokládaný termín odevzdání:	30. dubna 2019

L. S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci 18. října 2018

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že texty tištěné verze práce a elektronické verze práce vložené do IS STAG se shodují.

24. 4. 2019 František Jukl

Poděkování

Rád bych na tomto místě poděkoval vedoucímu práce za jeho trpělivost a cenné rady. Dále bych chtěl poděkovat všem zaměstnancům firmy Electro World s.r.o., kteří se zapojili do vývoje svými nápady, svým časem nebo poskytnutím prostředků.

Abstrakt

Cílem této práce je vytvořit webovou aplikaci pro skladové hospodářství, která umožní import dat ze systému SAP a provádět nad uloženými daty obvyklé skladové operace. Návrh aplikace má být zaměřen na bezpečnost celého systému. Autor aplikaci následně implementuje v programovacím jazyce Python a ověří její praktickou použitelnost v rámci firmy Electro World s.r.o. V první části jsou popsány požadavky na systém a možnosti využití technologií. Druhá část se zabývá konstrukčním řešením celého systému. V poslední části jsou specifikována potenciální vylepšení navrženého systému.

Klíčová slova:

skladové hospodářství, Python, REST, API, Flask, bezpečnost, zboží

Abstract

The aim of this work is to create a web application for warehouse management, which will process imported data from SAP system and allow common storage operations over stored data. Application design should be focused on the security of the entire system. The author then implements the application in Python and verifies its practical applicability within Electro World s.r.o. The first part describes the requirements for the system and the possibilities of using technologies. The second part deals with the design of the whole system. The last part specifies potential improvements to the proposed system.

Key words:

Warehouse management, Python, REST, API, Flask, security, goods

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

Seznam zkratek

EAN	European Article Number
HTML5	Hypertext Markup Language verze 5
CSS	Cascading Style Sheets
AJAX	Asynchronous JavaScript and XML
REST	Representational State Transfer
CORS	Cross-Origin Resource Sharing
ORM	Object-Relational Mapping
RFID	Radio Frequency Identification

Úvod

Pro maloobchodní prodejny existují různé možnosti, jak evidovat skladové zásoby. Vzhledem k tomu, že každá firma má své specifické požadavky a možné prostředky k využití nějakého řešení, je třeba návrh značně přizpůsobit, nebo vytvořit vlastní řešení.

Právě řešením evidence skladových zásob pro konkrétní společnost se práce zabývá. V práci je popsána firma Electro World s.r.o., pro kterou je aplikace navrhována. Nastiňuje bezpečnostní principy při zpracování zboží a zdůvodňuje přidání dalšího bezpečnostního prvku. Dále jsou popsány možnosti firmy a požadavky na systém. Autor hledá nejlepší řešení, které by vyhovovalo požadavkům, s možnostmi, které jsou mu poskytnuty. Vzhledem k množství volně dostupných technologií autor zdůvodňuje výběr softwarových prostředků, ale zároveň i hardwarových částí systému.

Druhá část je rozdělena na popis webové aplikace a restového API. Pro webovou aplikaci je popsán vzhled jednotlivých obrazovek a funkcionalita formulářů. Jsou zde řešeny hlavní požadavky na uživatelskou přívětivost a jednoduchost celého uživatelského rozhraní. V části pro restové API jsou popsány základní endpointy, hlavní metody a využité knihovny. Zároveň je zde v krátkosti popsán databázový systém.

V samotném závěru práce autor poukazuje na možnosti vylepšení systému a zabývá se možnými změnami v již už navrženém řešení. Tyto myšlenky se týkají softwarové i hardwarové části.

1 Společnost

Firma, ve které autor práce pracoval a pro kterou vypracoval aplikaci, jež tato práce popisuje, má dominantní postavení na trhu s elektronikou v České republice a na Slovensku. Celý obchodní název organizace je Electro World s.r.o. (dále jen EW). Autor práce byl zaměstnán střídavě na pobočkách v Liberci a Mladé Boleslavi.

Zaměstnanci každé pobočky jsou rozděleni do čtyř skupin. První skupinu tvoří manažeři, kteří jsou kontrolním orgánem ostatních skupin. Druhou skupinou jsou prodejci, kteří nemají žádná speciální oprávnění. Zprostředkovatele informací ze skladu pro tuto skupinu zaměstnanců představují zbývající dvě skupiny. Poslední dvě skupiny spolu úzce souvisí. Jsou to skladníci a pokladní. Zaměstnanci obsluhující sklad, pokladnu a v rámci toho reklamace jsou proškoleni na obě stanoviště. Což v praxi znamená, že pokladní na jedné směně je druhý den ve skladu a naopak (interní zdroje firmy, 2018).

1.1 Oběh zboží a bezpečnost

Společnost EW obecně obchoduje s elektronikou, ale hlavní pozornost padá na malé drahé zboží. Touto sledovanou kategorií jsou mobilní telefony, tablety, notebooky, stolní počítače, navigace, softwarové licence, fotoaparáty, outdoorové a jiné videokamery a jiné drobné zboží. Všechny tyto druhy jsou většinou vystaveny na prodejně a zbytek je skladován mimo oči zákazníka a prodejců.

Toto bezpečnostní opatření je z důvodu možné nevystopovatelné krádeže, pokud by ke zboží měl přístup kdokoliv. Dále je lepší držet zboží za bezpečnostními dveřmi, pro případ nočního vloupání. Pro vstup do takto zabezpečeného skladu slouží RFID (Radio Frequency Identification) čip, který po přiložení ke čtečce odemkne zámek dveří. Takový čip mají sice všichni zaměstnanci, ale pouze pracovník skladu, pokladen a manažer má oprávnění pro vstup do skladu. Čipy jiných zaměstnanců takové dveře neotevřou a slouží pouze pro vstup do zázemí prodejny. Každá pobočka může mít více takových skladů. Zpravidla nejsou zaměstnanci omezeni přístupem pouze do některých skladů. Mají tedy přístup do všech bezpečnostních skladů na dané pobočce.

Do těchto skladů se zboží dostává z hlavního skladu prodejny, kam ho složí skladník z kamionu, jež zboží dováží z centrálního skladu. Do hlavního skladu mají přístup taktéž

pouze výše zmínění zaměstnanci s vyšším oprávněním. Zde se po příjezdu kamionu zboží vyloží, převezme se dodací list se soupisem dodaného zboží a po odjezdu kamionu se počet zboží kontroluje. Kontrola probíhá speciálním scannerem napojeným na systém SAP, který firma pro správu zboží využívá. Po naskenování dodávky systém porovná rozdíly, které se dále řeší.

Veškerý pohyb zboží je kontrolován kamerovým systémem firmy. Kamery jsou jedním ze základních bezpečnostních systémů firmy. Po kontrole přijatého zboží nastává fáze třídění. V této fázi se zboží roztřídí do skupin podle lokace. Pro zajištění bezpečnosti jsou první zpracovány položky patřící do bezpečnostních skladů. Od přijetí zboží z kamionu po odvezení za dveře bezpečnostního skladu je každá položka zaznamenaná na kamerách a skladník za ně nese osobní odpovědnost. Jakmile se zboží dostane do bezpečnostního skladu, přebírá jej zpravidla jiný zaměstnanec, který má za úkol zboží roztřídit do regálů, popřípadě zajistit zboží pro vystavení. Tyto sklady jsou také hlídané kamerovým systémem, takže jakákoliv manipulace se zbožím je zaznamenána. Odtud pak pracovník pokladen bere zboží a vydává ho zákazníkům.

Dalším bezpečnostním systémem jsou inventury. Ty se dělají podle kategorií zboží (mobilní telefony, pračky, televize, notebooky...). Znamená to, že se na případné chybějící zboží přichází pozdě a jeho dohledání je zdlouhavé a obtížné.

Mimo inventurní ztráty se také řeší problém s umístěním zboží. Zvláště na pobočkách s mnoha sklady je občas problém dané zboží nalézt. Sice jsou nepsaná pravidla, kam se dává, jaký druh a jak by se zboží mělo řadit, ale kvůli lidskému faktoru se to nedaří dodržovat. Zvláště v době většího počtu dovolených nebo u předvánočních prodejů a povánočních výprodejů, je kvůli většímu množství brigádníků obtížnější stanovit a dodržovat určitá zavedená pravidla (interní zdroje firmy, 2018).

1.1.1 Bezpečnostní agentura

I přes tato omezení se pobočky nemusely více zaměřovat na zabezpečení zboží, jelikož byla na každé pobočce bezpečnostní agentura. Ta spolu se zaměstnanci zodpovídala za bezpečnostní organizaci všech prvků. Zároveň byla jako kontrolní orgán zaměstnanců a výše uvedené činnosti prováděla vždy s nimi.

Vzhledem k finančním výsledkům firmy se vedení rozhodlo, že začne šetřit. Rozvázáním spolupráce s bezpečnostní agenturou vznikl další problém, jelikož zmizela dvojí kontrola u běžně kontrolovaných prací (interní zdroje firmy, 2018).

2 Požadavky na řešení

I přes problémy na všech pobočkách, nejvíce se o řešení zajímá pobočka v Liberci. Přidáním dalšího bezpečnostního prvku do celého oběhu by vzrostla možnost kontroly. Je potřeba přidat záchytný bod mezi příjmem do bezpečnostního skladu a výdejem z něj. Takový systém musí splňovat několik požadavků.

2.1 Nezávislost

Jelikož se jedná o řešení lokálního rozsahu, která si pobočka chce zavést sama, mělo by být nezávislé na vnitřních systémech firmy. Ačkoliv fungují vnitřní mechanismy primárně na systému SAP, některé věci jsou zdlouhavé, pomalé, a hlavně doba zpracování požadavků na úpravy je dlouhá. Zároveň firma odmítá zpřístupnit API nebo jiný přístup k datům.

2.2 Práce s volně dostupnými technologiemi

Pro využití technologií, které by systém mohl využít, si musí vystačit s těmi, co nevyžadují zásah z vnitřku firmy. Například je možné využít čtečky čárových kódů, které jsou ve firemních počítačích již připojeny a dají se volně konfigurovat. Ovšem veškeré návody je potřeba získat přímo od výrobce. Další omezení jsou firemní počítače a firemní síť. Počítače mají zakázané instalace jiných softwarů, než které jsou dodány z IT oddělení. Dále mají omezený přístup do vnější sítě neboli internetu. Zároveň je znemožněno připojení externích zařízení pomocí USB, jelikož je nemožné nainstalovat pro toto zařízení ovladače.

Řešení navíc nemůže stát větší obnos peněz, jelikož si ho pobočka hradí sama ze svých zdrojů, které musí před centrálou obhajovat. V případě přichází řešení za velmi malý finanční obnos, nebo finanční motivace zaměstnance.

2.3 Lidský faktor

Další požadavek na systém se váže na lidský faktor. Ve firmě pracují lidé, jejichž počítačové znalosti nejsou na dobré úrovni. Výsledek by tedy měl být natolik jednoduchý,

aby ho zvládl jakýkoliv zaměstnanec. Zároveň musí být rychlý, aby zákazník nečekal na odbavení pracovníkem příliš dlouho. Pro práci s ním musí být jasné a intuitivní rozhraní.

2.4 Analýza

Autor práce byl vybídnut store managerem k vypracování analýzy vzniklého problému. Při všech omezeních byl manažerem navrhnut Excel, kam by se příjmy a výdeje zboží zapisovaly. Tento způsob by však nedokázal plnit předpoklady, ke kterým byl určen.

Jako alternativa byla předložena aplikace, jež by správu pohybů obstarávala. Aplikace by mohla splňovat veškeré požadavky, jaké jsou na systém kladeny.

Jelikož není možnost instalovat vlastní software, musí být aplikace co nejméně závislá na operačním systému. Tomu odpovídá koncept webové aplikace. Ta může běžet na interním serveru a nenaruší se tím bezpečnost infrastruktury. Díky internímu serveru nejsou třeba certifikáty, jelikož takový server je vždy důvěryhodný. Pro umístění na interní server je třeba komunikovat s vedoucím IT oddělení firmy, aby virtualizoval nový server.

Vzhledem k bezpečnosti a co největšímu zamezení zneužití aplikace by každý uživatel měl mít své přihlašovací údaje. Nejlepší kombinací bude osobní číslo, které každý zaměstnanec identifikuje každý svůj prodej, takže ho zná nazpaměť a vymyslí k němu heslo.

Hlavní cíl aplikace je sledování pohybu zboží. Proto s každým pohybem musí odpovědný zaměstnanec vložit záznam do systému. Pro rychlost celé operace je vhodné využít jednoduchý formulář. V němž se zaměstnanec identifikuje a pomocí čtečky načte potřebné informace o zboží. Identifikace zaměstnance může probíhat pomocí výš zmíněné kombinace, nebo pro rychlost a pohodlí může využít svůj čip. Jelikož čip nosí stále u sebe a identifikuje se pomocí něj do jiných interních systémů firmy, dá se předpokládat unikátnost jeho otisku.

Jenže aby byl zaměstnanec schopen využít čip v takovém systému, musela by být možnost nainstalovat RFID čtečku. V rámci pohodlí zaměstnanců a jasného vymezení pracovní stanice by bylo vhodné využití nějakého staršího počítače jen pro účely systému. Tento počítač by byl připojen do firemní sítě, ale zároveň by nepodléhal kontrole instalovaného softwaru a zařízení. Takto by pracovník ve skladu mohl pracovat za zavřenými dveřmi a aplikace by se zároveň mohla využívat v celé firemní síti.

Aby měla aplikace vhodný informační a kontrolní charakter, je zapotřebí aby dokázala zobrazit popisy zboží a unikátní firemní identifikátory zboží. Tyto informace lze získat ze systému SAP. Vzhledem k nemožnosti přistoupit k datům skrze API, je nutnost umožnit import dat jiným způsobem. Pomocí menu pro zobrazení zásob včetně nulových položek lze zobrazit veškeré informace o veškerém zboží. SAP umožňuje vytvořit a uložit layout, takže je pro uživatele jednodušší zachování konzistence dat. Pro export dat SAP nabízí mimo jiné i formát CSV. Ten je vhodný pro další parsování dat.

Vzhledem k nemožnosti vyloučení lidského faktoru a pro kontrolu stavu zboží, by aplikace měla obsahovat modul s inventurou. Díky němu lze zjistit ztrátu zboží a upravit stav zboží v aplikaci dle reálného počtu.

3 Technologie

Tato část práce se zabývá popisem možných technologií, vysvětlením pro užití dané technologie a praktickým popisem aplikace. Pojem technologie seskupuje jak softwarovou, tak hardwarovou část. Ze softwarové části se řeší problém vhodného programovacího jazyka, frameworku, struktura a principy aplikace. Z hardwarové části je zmíněno použití čteček a princip jejich spolupráce s aplikací pro komfort uživatele.

3.1 Koncept systému

Celý systém je rozdělen na dvě hlavní části. První část je datová. Obstarává úpravy, vytváření, mazání a čtení dat z databázového systému. Jako databázový server bylo zvoleno MySQL. S databázovým serverem komunikuje webové REST API. Druhá část systému je webový klient. Funkce klienta je obsluha klientských požadavků skrze grafické uživatelské rozhraní. Je zaměřen na zobrazování a transformaci dat do uživatelsky přívětivé podoby. S datovým API komunikuje ve formátu JSON. Klient pomocí HTML, CSS a JavaScriptu vykresluje celé rozhraní, které lze zobrazit ve kterémkoliv moderním webovém prohlížeči.

3.2 Programovací jazyk

Pro API i klientskou část byl zvolen stejný programovací jazyk. V současné době je v oblasti informačních technologií možnost využít mnoha programovacích jazyků. Původně nejpoužívanější jazyk pro serverovou část webových aplikací, kterým je PHP (insights.stackoverflow.com, 2016), má přes širokou komunitu proti jiným modernějším jazykům řadu nevýhod. Jednou velkou nevýhodou je například řada zranitelností od těch méně vážných a po ty nejvážnější jako je třeba elevace oprávnění nad systémem (cvedetails.com, 2019). Výhodou PHP je, že díky svému stáří má mnoho různých frameworků a knihoven, jež práci s ním značně ulehčují. Dalším vhodným jazykem pro tvorbu serverové aplikace je C#, který spolu s frameworkem .NET a rozšířením WebApi tvoří vhodným nástroj. Díky velké podpory Microsoftu má spoustu knihoven přístupných pomocí balíčkovacího systému Nuget (Nuget.org, 2019). Poslední diskutovanou možností je Python. Ačkoliv je Python brán spíše jako skriptovací programovací jazyk, jeho možnosti jsou daleko větší. Jeho dynamická typová kontrola s podporou pro nejlepší

praktiky při programování, nabízí střední zlatou cestu mezi probíranými variantami. Jeho jednoduchá syntaxe, velká podpora komunity, dostatečné množství dokumentace a stáří jazyka se stálými updaty z něj dělá jeden z momentálně nejoblíbenějších programovacích jazyků (Insights.stackoverflow.com, 2018). Po diskuzi s vedoucím této práce byl zvolen právě Python jako vhodná varianta pro API a zároveň webovou aplikaci. Zvolena byla, v době zadání práce, nejnovější verze jazyka 3.7.1.

3.3 Framework Flask

Python nabízí nepřeberné množství různých webových frameworků. Nejrozšířenějším je Django, který má nejširší základnu uživatelů a nejvíce knihoven (Codementor.io, 2017). Pro jednodušší aplikaci je použití tohoto frameworku zbytečné. Stejně výkonnou aplikaci lze naprogramovat pomocí mikroframeworku Flask. Flask sice není full-stack framework jako Django, ale o to větší je flexibilita pro programátora (Wiki.python.org, 2019). Možnost volby vlastního šablonovacího systému a úprava funkcí knihoven. Z těchto důvodů byl zvolen právě Flask. Tento framework sám o sobě disponuje šablonovacím systémem Jinja2, vlastním debuggerem, vývojovým serverem, systémem pro zpracování požadavků a správou cookies a session. Byla zvolena nejnovější verze 1.0.2 (Flask.pocoo.org, 2019).

3.4 Čtečka čárových kódů

Pro identifikaci zboží je používán systém čárových kódů. V Evropě je používán EAN (European Article Number). Nejčastěji je 13místný a zboží jednoho typu má stejný kód (Kodys.cz, 2018). Společnost EW používá čtečky čárových kódů, které se dají různě konfigurovat. Defaultně je čtečka nastavena, aby po naskenování vložila enter (interní zdroje firmy, 2018). Jelikož funkcionalita čtečky je nejvíce využívána při přidávání záznamu, je třeba v rámci pohodlí uživatele čtečku přenastavit. Po naskenování sekvence správných kódů je pro správnou funkčnost v aplikaci čtečka nastavena, aby po každém naskenování vložila znak tabulátoru. Tak se dokáže uživatel ve formuláři jednoduše pohybovat a ve výsledku jen skenuje a nemusí používat klávesnici.

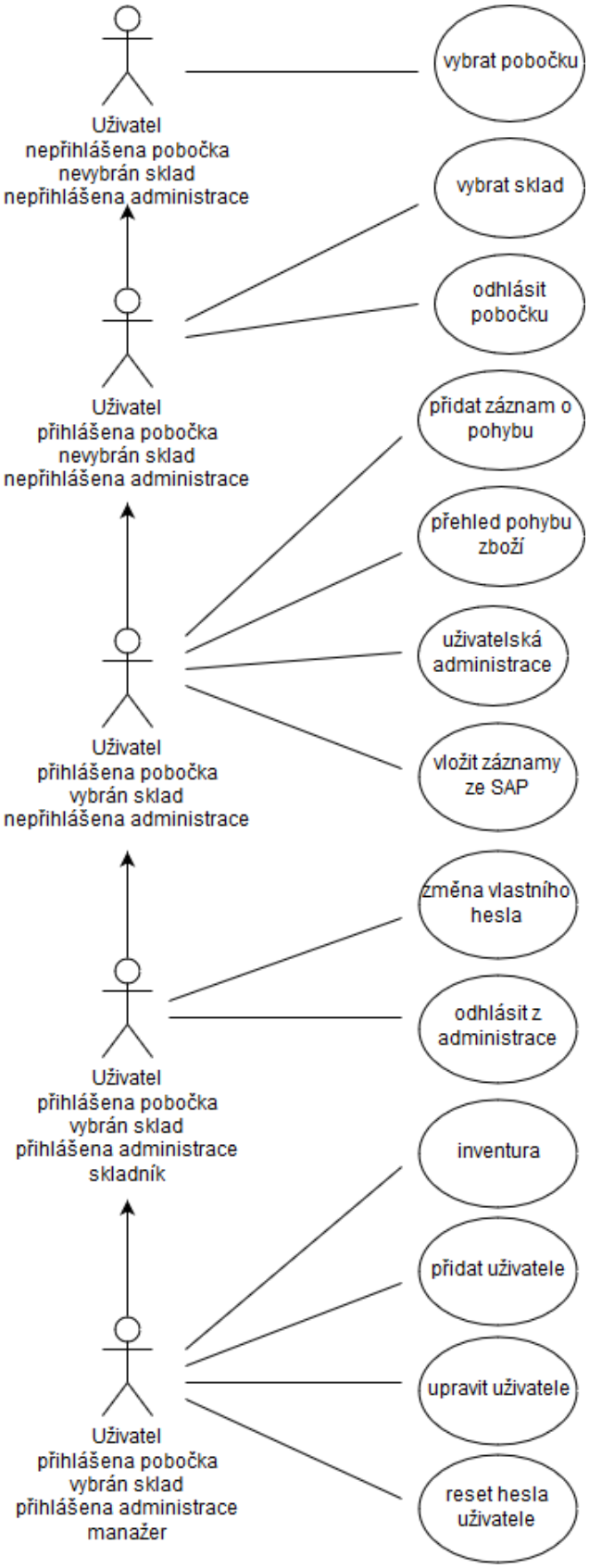
3.5 RFID čtečka

Pro komfort uživatele je možnost autorizace požadavku pomocí čipu. Čip má každý zaměstnanec a slouží pro identifikaci uživatele ve vnitřních systémech firmy. Čtečka s čipem komunikuje na frekvenci 13,56 MHz. Je použita USB čtečka, která slouží, podobně jako čtečka čárových kódů, jen jako vstup do pole formuláře.

4 Webová aplikace

Celý systém se skládá z REST API a webové aplikace. V této části se práce zabývá webovou aplikací a blíže popisuje principy a postupy v ní použité.

Aplikace s webovým rozhraním je psána v jazyce Python pomocí frameworku Flask. Díky funkcím frameworku je možné udělat směrování požadavků uživatele z webového prohlížeče skrze uživatelské rozhraní. Framework obstarává tyto požadavky a funkce, které jsou volány na základě URL adresy, realizují komunikaci s API a provádějí logiku nutnou pro funkčnost celé aplikace (Flask.pocoo.org, 2019).



Obrázek 1: Use case diagram aplikace

Zdroj: Vlastní zpracování, 2019

4.1 Uživatelské rozhraní

Pro vykreslení uživatelského rozhraní byl použit šablonovací systém Jinja2, který je defaultně dodáván v balíčku frameworku Flask. Díky jednoduché syntaxi lze tvořit sekce stránky a různě je kombinovat (Flask.pocoo.org, 2019). Jako základ jsou vytvořeny dvě šablony. První šablona je holá a slouží pro zobrazení nepřihlášené pobočky. Druhá šablona zobrazuje menu s možnostmi uživatele jako je přidání záznamu, vyhledání záznamů pro zboží, správa uživatele a integrace výstupu systému SAP. Šablonovací systém třídí prvky HTML do znovupoužitelných bloků a vytváří schéma každé jednotlivé stránky. Pro vykreslení stránek je použito HTML5 (Hypertext Markup Language verze 5) a CSS (Cascading Style Sheets). Pro vzhled stránek je použit CSS framework Bootstrap. Funkcionalitu stránek doplňuje JavaScript podle standardu Ecmascript 2017 za pomoci knihovny jQuery.

4.1.1 Přihlášení pobočky

První stránkou, kterou uživatel uvidí je přihlášení pobočky. Firma má v této chvíli v České republice 18 poboček. Pobočky, které tuto aplikaci používají, jsou uloženy v databázi a zobrazují se v selectboxu. Zbývající vstupy jsou pro osobní číslo uživatele a jeho heslo.

4.1.2 Výběr skladu

Některé pobočky mívají více bezpečnostních skladů, které by chtěly monitorovat. Proto je na této obrazovce vidět selectbox, ve kterém si uživatel zvolí příslušný sklad, ve kterém se počítač, který je určen pro běh aplikace, nachází.

4.1.3 Formulář na přidání záznamu

Hlavní část aplikace se nachází v okně pro přidání záznamu. Je zde formulář, který obsahuje pole pro autentizaci uživatele. Jsou dvě možnosti, jak může uživatel prokázat svou identitu. První možnost je zadáním svého osobního čísla. Druhá možnost je přiložením čipu k RFID čtečce. Pomocí checkbox pole je možné provádět vícenásobné přidávání záznamů se zapamatovanou autentizací. Po odškrtnutí checkboxu dojde k vymazání přihlašovacích údajů uživatele z formuláře, aby nemohly být zneužity jiným zaměstnancem.

Další část formuláře obsahuje pole pro číslo faktury, kam je možné pomocí čtečky naskenovat kód z účtenky, a pole pro dodatečný text.

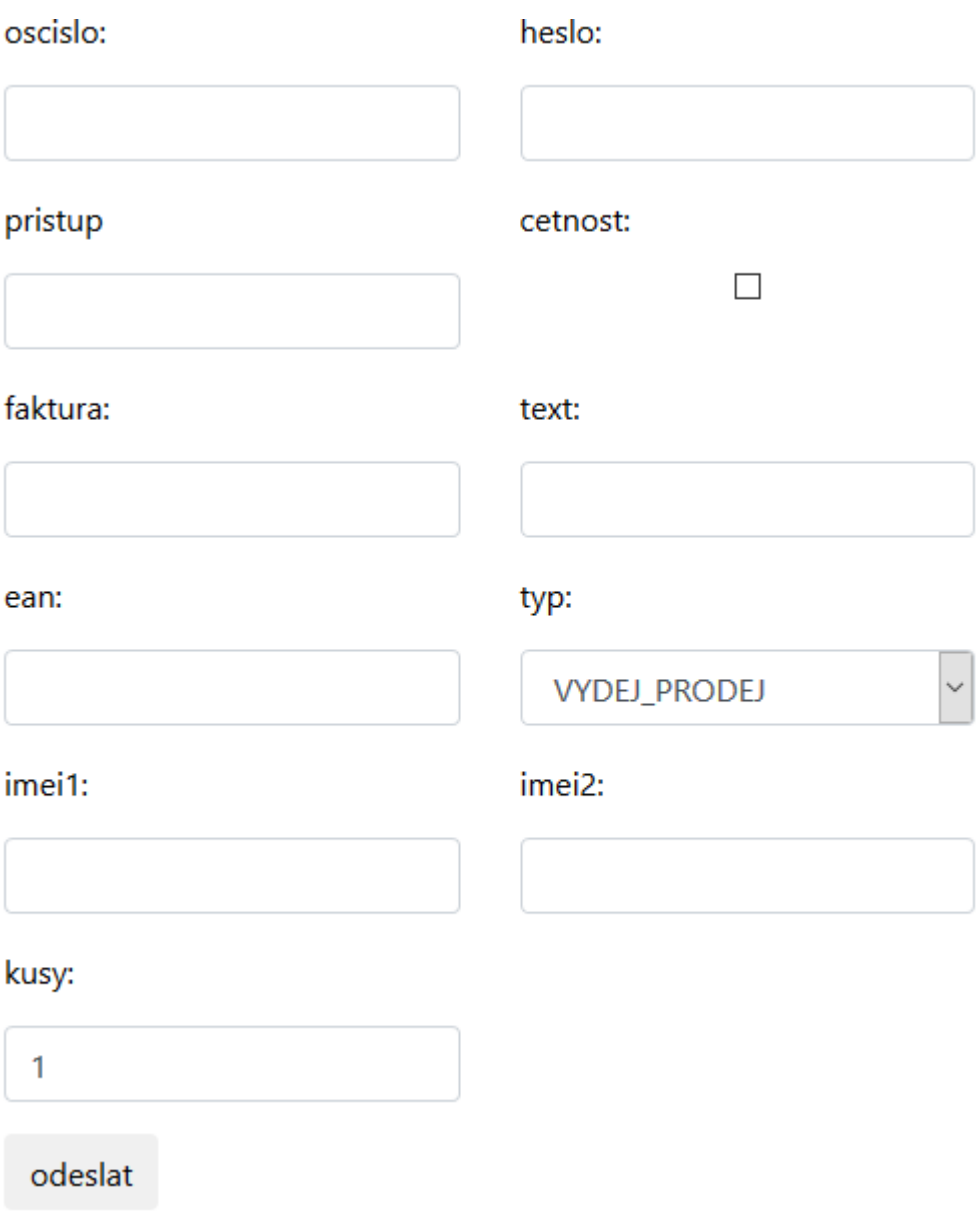
Třetí část formuláře obsahuje selectbox pro typ pohybu zboží. V něm zaměstnanec vybere typ pohybu, který se dělí na příjem a výdej. V těchto skupinách jsou možnosti s důvody, které mohou nastat a možnost jiné. V případě položky s názvem jiné zaměstnanec vyplňuje doplňující text.

Následující je pole pro EAN zboží. EAN se naskenuje pomocí připojené čtečky. Po naskenování čárového kódu se díky nakonfigurované čtečce kurzor sám posune do dalšího pole pro IMEI. Zároveň je odeslán AJAX (Asynchronous JavaScript and XML) požadavek na server. Odpověď požadavku obsahuje ORA kód, krátký popis a ITEM zboží, nebo prázdnou hodnotu. Tyto hodnoty se vloží do tabulky vedle formuláře. Tímto má zaměstnanec vydávající zboží jistotu, že skutečně vydává zboží, které vydat chce a nepřehlídl se například v barvě nebo verzi zařízení. V případě prázdných hodnot může poznat, že buď nenaskenoval EAN, nebo je třeba aktualizovat databázi ze systému SAP.

IMEI je unikátní identifikátor mobilních zařízení, které mají přístup do sítě mobilních operátorů (Dotekomanie.cz, 2015). Některé telefony mají možnost dvou a více sim karet. Na obalech těchto mobilních telefonů bývají zpravidla všechny záznamy IMEI. Tyto záznamy jsou vyobrazeny pomocí čárových kódů, které připojená čtečka dokáže přečíst.

Po naskenování všech potřebných záznamů ze zboží lze zadat počet kusů. Zaměstnanec ručí za správnost záznamů, proto pole s kusy vyplňuje pouze pokud má stoprocentní jistotu, že se jedná o více kusů stejného zboží. Jakmile jsou potřebná pole vyplněna, zaměstnanec odešle záznam. Odpovědí ze serveru získá jistotu, že se záznam uložil, nebo mu je oznámen problém.

Po přidání záznamu se aktualizuje tabulka v pravé části stránky. Tabulka obsahuje posledních deset záznamů. Slouží to jako kontrola vedoucím pracovníkům, že jejich lidé systém využívají a zároveň kontrola zaměstnancům, kteří se systémem pracují. Zaměstnanec díky tomu zjistí, že zapomněl vydané zboží vložit do systému, anebo například při příjmu velkého množství zboží toto zboží již naskenoval.



Obrázek 2: Formulář pro přidání záznamu o pohybu zboží

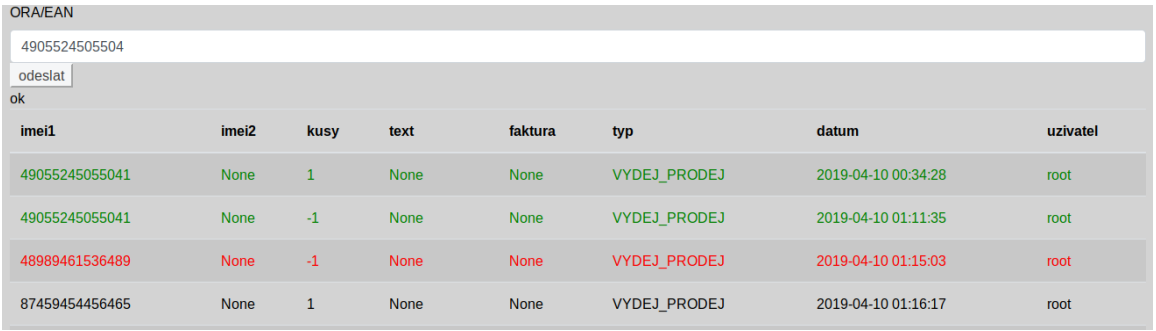
Zdroj: Vlastní zpracování, 2019

4.1.4 Přehled

Modul přehledu slouží pro zobrazení pohybu zboží. Je tvořen formulářem s jedním vstupem, kam pracovník zadá EAN nebo ORA kód. Po odeslání údaje je mu ze serveru vrácena odpověď. V případě negativní odpovědi je dle zprávy patrné, že selhal požadavek na zboží. Selhání požadavku může být z důvodu chyby na serveru, nebo v případě zadání ORA kódu neaktuální databáze. Další možností pro negativní odpověď je nulový počet záznamů. Znamená to, že žádný kus tohoto zboží ještě systémem neprošel. Kladná odpověď vykreslí všechny záznamy do tabulky.

Tabulka se skládá z polí IMEI 1, IMEI 2, kusy, text, faktura, typ, datum a uživatel. Tato pole obsahují záznamy z modulu s názvem přidej. V tabulce je možné vyhledat historické záznamy a dohledat pohyb například ztraceného zboží. Zároveň jsou zde uvedeny důvody nad rámec systému SAP. Zaměstnanci zde zjistí, kolik kusů má být v konkrétním bezpečnostním skladě, aby nehledali zboží zbytečně a dlouho. Lze zde zjistit, kolik kusů zboží je k dispozici, jelikož je tento počet očištěn o zboží vystavené, rezervace na objednávky a jiné.

Pro záznamy s IMEI je tabulka přínosná automatickým zbarvením záznamů posloupnosti pohybů. Pokud je zboží vydáno před příjmem, je řádek zbarven do červena. Jestliže je zboží se stejným IMEI přijato a následně vydáno, jsou oba záznamy zbarveny do zelena. Pokud je zboží přijato, a ještě není vydáno, zůstává záznam v původní barvě. Při kontrole stávajícího zboží s aplikací lze jednoduše najít záznamy například odcizených mobilních telefonů a nahlásit IMEI policii a operátorovi.



imei1	imei2	kusy	text	faktura	typ	datum	uzivatel
49055245055041	None	1	None	None	VYDEJ_PRODEJ	2019-04-10 00:34:28	root
49055245055041	None	-1	None	None	VYDEJ_PRODEJ	2019-04-10 01:11:35	root
48989461536489	None	-1	None	None	VYDEJ_PRODEJ	2019-04-10 01:15:03	root
87459454456465	None	1	None	None	VYDEJ_PRODEJ	2019-04-10 01:16:17	root

Obrázek 3: Přehled zboží s IMEI

Zdroj: Vlastní zpracování, 2019

4.1.5 Správa uživatele

Modul pro správu uživatele obsahuje více obrazovek a funkcí. Prvotní obrazovkou je přihlášení uživatele pomocí osobního čísla a hesla.

Po úspěšném přihlášení se uživateli zpřístupní nové menu, které obsahuje položky změna hesla, přidat uživatele, upravit uživatele, reset hesla a odhlásit. Změna hesla a odhlášení je zpřístupněna všem uživatelům. Přidání nového uživatele, úprava informací o stávajícím a reset hesla uživatele je zpřístupněna pouze adminovi a manažerským pozicím.

Obrazovka pro změnu hesla obsahuje dvě pole pro nové heslo a kontrolu proti překlepům nového hesla. Obrazovka pro přidání nového uživatele obsahuje formulář pro jméno, heslo, osobní číslo uživatele, uživatelův otisk z čipu a roli. Další možností pro manažera

je upravit informace již stávajícího uživatele. Pro úpravu je zpřístupněno pouze jméno, otisk čipu a role. Obnova hesla určitého uživatele vygeneruje nové heslo a zobrazí ho na obrazovce. Předpokládá se, že manažer jako nejvyšší instance na pobočce je důvěryhodný a nezneužije této možnosti. Nové heslo je náhodné několikamístné číslo, takže je uživatel nucen si nastavit heslo nové, které si bude pamatovat.

4.1.6 Inventura

Pro kontrolu aktuálního stavu zboží je potřeba dělat v pravidelných intervalech inventuru. Kontrola stavu bezpečnostního skladu doplňuje kontrolu jednotlivých kategorií zboží v celém obchodě. Pro spuštění inventury se přihlásí uživatel, který disponuje manažerskými právy, a nahraje csv soubor. Server poté vyhodnotí záznamy v souboru a zobrazí uživateli rozdílovou tabulku s informacemi o zboží a možností opravit množství zboží na skladě.

ean	text	ora	item	category	scan	ucetni	
4905524505504	SONY KDL-26U4000K	1000017	KDL26U4000	EA0101	1	3	oprav
5678945645654					1	2	oprav
9055545587878	SAMS S3 BK	1000457	SAMS3IIBK	CA0101	0	2	oprav

Obrázek 4: Náhled inventury

Zdroj: Vlastní zpracování, 2019

4.1.7 SAP

Aby byly informace poskytované systémem vždy aktuální, je zapotřebí, aby zaměstnanci pravidelně aktualizovali databázi. Vzhledem k neumožnění připojení systému rovnou na API systému SAP jsou uživatelé nuceni aktualizovat tuto databázi exportem csv souboru se všemi informacemi. Soubor se nahraje do webové aplikace, která ho zpracuje a odesílá do API, které ho uloží do databáze. Tato operace trvá vzhledem k velikosti souboru pár desítek vteřin. V současné době má takový soubor přes 10 MB a je v něm uloženo přes 130 000 záznamů. Výsledek ukládání je zobrazen uživateli.

Tabulka 1: Ukázka dat ze systému SAP

Zboží	Model	Kód GTIN	Krát.text zboží	Volně použitelná	Kateg.zboží
1000015	KV21CL10K	4901780820672	SONY KV-21CL10K	0	EA0101
1000017	KDL26U4000	4905524505504	SONY KDL-26U4000K	0	EA0101
1000018	KDL26S4000K	4905524505597	SONY KDL-26S4000	0	EA0101
1000019	KDL26B4050	4905524464320	KDL-26B4050K	0	EA0101
1000023	KDL32P3000	4905524419191	KDL-32P3000K	0	EA0101
1000025	KDL32W4000	4905524495911	KDL-32W4000K	0	EA0101
1000027	KDL32L4000K	4905524511307	KDL-32L4000	0	EA0101
1000028	KDL32U4000K	4905524505474	KDL-32U4000K	0	EA0101
1000032	KDL32V4500	4905524496000	KDL-32V4500K	0	EA0101
1000033	KDL32V5500	4905524550726	KDL-32V5500K	0	EA0101
1000034	KDL32W5500	4905524550160	KDL-32W5500	0	EA0101
1000035	KDL32S5500	4905524552478	KDL-32S5500	0	EA0101
1000036	KDL37P3020	4905524433340	KDL-37P3020	0	EA0101
1000041	KDL37V5500	4905524550603	KDL-37V5500	0	EA0101
1000043	KDL40EX1	4905524533941	KDL-40EX1BAEP	0	EA0101

Zdroj: Vlastní zpracování dle dat ze systému SAP, 2019

4.2 Backend webové aplikace

Struktura aplikace je tvořena pomocí doporučeného vzoru v oficiální dokumentaci frameworku Flask (Flask.pocoo.org, 2010). Po založení projektu ve vývojovém prostředí PyCharm, které pro projekt pomocí nástroje `venv` zároveň vytváří nové virtuální prostředí, se vygeneruje složka pro statické soubory a složka pro šablony. Základem aplikace je soubor `application.py`, ve kterém je definována základní konfigurace celé aplikace. Ostatní proměnné prostředí jsou definovány přímo skrze terminál.

4.2.1 Moduly

Soubor `modules.py` obsahuje balíčky, které jsou potřeba inicializovat s instancí aplikace. Je zde modul pro ochranu proti útoku Cross-site Request Forgery a modul pro správu session. Oba tyto moduly jsou poté importovány a inicializovány při tvorbě instance aplikace.

4.2.2 Formuláře

Nedílnou součástí aplikace jsou formuláře. Do frameworku je přidán balíček WTForms, díky kterému lze každý formulář definovat jako třídu a pro každé pole zavést jeho validaci. Balíček obsahuje všechny typy polí a má definované základní validátory. Výhodou je možnost vlastní validace každého pole, a to třeba i vůči ostatním polím z formuláře (Flask-wtf.readthedocs.io, 2018). Všechny formuláře jsou definovány v souboru `forms.py`, odkud jsou importovány do logické části aplikace. Pro vykreslení stačí přidat do tagu `form` příkaz pro každé pole a systém vygeneruje validní HTML kód. Zároveň lze na libovolné místo stránky umístit zprávy z validátorů.

```
class SelectStoreForm(FlaskForm):
    store_id = SelectField('Pobocky', coerce=int, validators=[validate_empty_select])
    name = IntegerField(id='name', label='os. cislo', validators=[DataRequired('Spatny format osobniho cisla.')])
    password = PasswordField(id='password', label='heslo', validators=[DataRequired('Nebylo zadano heslo.')])
```

Zdrojový kód 1: Formulář Flask-WTF

Zdroj: Vlastní zpracování, 2019

4.2.3 Komunikace s API

Soubor `api.py` obsahuje předpis pro funkce, které se využívají pro komunikaci s API. Každá funkce je wrapper na funkce z knihovny `requests`. Do defaultních funkcí wrapper přidává hlavičku `Authorization`, do níž se umístí řetězec `Bearer`, za který se přidá token pro autentizaci zařízení. Každá funkce zároveň přebírá z konfigurace aplikace adresu pro API a doplňuje pouze název endpointu.

4.2.4 Server

Srdcem aplikační logiky je soubor `server.py`. V něm probíhá směrování uživatelského dotazu, provádí se zpracování dat z formulářů, komunikuje s API a vrací uživateli překreslené stránky jeho rozhraní.

Správa uživatelských dat jako je přihlášený uživatel, přihlášená pobočka, klíč pro komunikaci s API a vybraný sklad obstarává session manager. Pro tyto klíče do pole session jsou definovány konstanty, aby nedošlo k překlepům a záměnám.

Dále je definován výčet pro typy pohybů zboží. V aplikaci a dále v databázi se pak pracuje pouze s číselnými identifikátory těchto typů.

Díky frameworku lze definovat různé cesty pomocí dekorátorů nad funkcemi. Dekorátor `@app.route('/selectstore', methods=['GET', 'POST'])` nám v aplikaci registruje trasu `/selectstore` a povoluje pro něj metody GET a POST. V případě přístupu s jinou metodou vrací framework `Method not allowed` s kódem 405. Když uživatel zkouší přistoupit na neexistující cestu, framework vrátí `Not Found` s kódem 404 a vykreslí defaultní obrazovku (Flask.pocoo.org, 2010).

První funkcí, ke které se uživatel dostane, je přihlášení pobočky. V této funkci se smažou údaje nastavené v session a zobrazí se uživateli formulář s přihlášením a výběrem pobočky. Při odeslání formuláře se pole zvalidují a proběhne autentizace uživatele skrze API. Kladná odpověď z API značí, že uživatel zadal správné osobní číslo, heslo a vybral pobočku, ke které je přiřazen. Identifikátor pobočky s uloží do session. S kladnou odpovědí ze serveru přichází klíč. Tento token je taktéž uložen do session. Uživatel je přesměrován na funkci výběru skladu. Negativní odpověď na autentizaci vrátí uživatele na formulář s přihlášením se zprávou o špatné kombinaci.

Metoda pro výběr skladu načte z API všechny sklady, které jsou přiřazeny pobočce a tyto možnosti zobrazí uživateli. Zde není potřeba autentizace a ani autorizace uživatele, jelikož všichni uživatelé mají přístup ke všem skladům dané pobočky. Jakmile uživatel vybere sklad, uloží se jeho výběr do session a je přesměrován na funkci pro přidání záznamu.

Funkce `add_entry` nejprve načítá posledních deset záznamů, jelikož se vyskytují ve všech návratových případech funkce. Při metodě GET se pouze vyrenderuje formulář. S metodou POST se validuje formulář s potenciálním záznamem. Funkce mimo prázdnost povinných polí kontroluje, aby záznam s vyplněným IMEI měl pouze jeden kus. Následuje autentizace uživatele s jeho nastavenou pobočkou. Možností autentizace je více. První se kontroluje prázdnost polí s osobním číslem a heslem. Pokud je alespoň jedno z těchto polí prázdné, pokračuje se k autentizaci pomocí autentizačního řetězce, což je digitální otisk čipu uživatele. Když jsou obě možnosti nevyplněné, vrací se formulář zpět uživateli s hláškou o chybějících údajích. Při selhání autentizace s vyplněnými údaji se vrací formulář s hláškou o špatných údajích. Po autentizaci následuje úprava údaje o kusech, které chodí z formuláře vždy kladné, i při výdeji zboží. Do databáze se sice ukládá typ pohybu, ale ten slouží spíše pro informativní účely manažera s cílem identifikovat konkrétní záznam a smysl pohybu zboží. Proto proběhne kontrola a pokud je typ pohybu nastaven na výdej, je počet kusů převrácen do záporné hodnoty. Po kontrole proběhne

odeslání dat do API, které je buď uloží a vrátí identifikátor uloženého záznamu, nebo vrátí chybové hlášení. Při negativní odpovědi se uživateli zobrazí chyba. S novým identifikátorem je aktualizován seznam posledních záznamů, aby odpovídal nové skutečnosti. Uživateli je přerenderován formulář s defaultními hodnotami a zobrazeno hlášení o úspěchu. Když uživatel nastavil vícečetné zadávání, nastaví se do skrytého pole formuláře hodnota uživatelova autentizačního řetězce. Tuto hodnotu může uživatel kdykoliv smazat po odškrtnutí checkboxu ve formuláři. Takto upravený formulář se vrátí uživateli.

Za inventuru zodpovídá manažer, který musí být přihlášen pro odeslání csv souboru do funkce `stock_taking`. Po odeslání souboru tato funkce zpracuje všechny záznamy a uloží data do formátu JSON, která jsou odeslána do API. Rozhraní vrací rozdílový seznam. Poté dojde k vykreslení formuláře, kde uživatel může upravit stav zboží. Pro úpravu stavu je použito metody `update_stock_state`, která do API kromě údajích o uživateli a skladu odešle EAN, nové kusy a typ pohybu podle kladného nebo záporného počtu. Řádek je po zpracování odstraněn z formuláře.

S GET metodou pro funkci `populate_sap` se vykreslí formulář s možností nahrát csv soubor. Zde probíhá validace přípony souboru. Soubor je uložen do složky `uploads`. Pro parsování souboru je použit modul `csv`, který je součástí instalace jazyka. Jelikož automatická detekce kódování souboru není spolehlivá, je upraveno nastavení na windows-1250, aby odpovídalo nastavení firemních počítačů a jejich výstupů. Jako oddělovač v souboru je vyžadován středník. Pro čtení je použit DictReader, který načte řádky do slovníku. Po přeskočení prvního řádku, který je brán jako hlavička souboru se všechny slovníky uloží do listu, jež se posílá do API. Každý slovník je formátu `{"hlavička 1": "hodnota 1", "hlavička 2": "hodnota 2"}`. Pro správnou funkčnost metody v API je zapotřebí, aby soubor vždy obsahoval všechny potřebné sloupce, které jsou vždy stejně pojmenované. Soubor se generuje ze systému SAP a je třeba zachování konzistence těchto výstupů. Zpracování těchto dat na straně API trvá průměrně pár desítek sekund a záleží hlavně na výkonnosti serveru. Po obdržení odpovědi se zpráva ukáže uživateli.

```
@app.route('/sap', methods=['GET', 'POST'])
def populate_sap():
    form = SapForm(request.files)

    if form.validate_on_submit():
        f = form.file.data
        filename = secure_filename(f.filename)
        path = os.path.join(
            os.path.dirname(os.path.abspath(_file_)), 'uploads', filename
        )
        f.save(path)

        all_list = []
        with open(path, mode='r', encoding='windows-1250') as csv_file:
            csv_reader = csv.DictReader(csv_file, delimiter=';')
            line_count = 0
            for row in csv_reader:
                if line_count == 0:
                    line_count += 1
                else:
                    all_list.append(row)
                    line_count += 1
        data = json.dumps(all_list)
        answer = post_req('sap', data, session[constants['SESSION_API_KEY']]).json()
        if answer['ok']:
            flash('Uspesne ulozeno', 'success')
        else:
            flash('Chyba', 'error')
        form.file.data = None
        return render_template('sap/sap.html', form=form)
    else:
        return render_template('sap/sap.html', form=form)
```

Zdrojový kód 2: Načtení souboru na backendu webové aplikace

Zdroj: Vlastní zpracování, 2019

Metoda přehledu vykreslí při požadavku GET formulář o jednom poli. Při POST metoda zjistí, zda uživatel zadal EAN, nebo ORA kód. Zjišťování probíhá na základě délky řetězce, jelikož platí podmínka, že EAN kód bude delší jak 11 znaků. Pro každý typ kódu má API jiný endpoint. Odpověď má buď kód stavu 404, což značí, že žádný záznam nebyl pro zadaný řetězec nalezen, nebo odpověď obsahuje kód 200 a seznam záznamů. Uživatel musí brát na vědomí, že i když se vrátí nula záznamů a on zadal ORA kód, nemusí to znamenat, že zboží má skutečně nula pohybů. V případě nového zboží by měl ověřit aktuálnost databáze ze systému SAP. Pokud se nevrátí záznamy pro EAN, znamená to, že opravdu žádné záznamy v systému neexistují.

Následující funkce zajišťující správu uživatele. První funkcí, kterou uživatel využije je přihlášení. Aplikace skrze API ověří kombinaci uživatelského osobního čísla, hesla a

vybrané pobočky. Při správné kombinace uloží do session identifikátor uživatele. Funkce slouží zároveň jako odhlášení, jelikož při požadavku GET vymaže záznam ze session.

Pro obnovu uživatelského hesla je vykreslen formulář pro zadání nového hesla a jeho ověření. Jestliže se hesla shodují, odešlou se do API, kde proběhne hash hesla a update v databázi.

Manažer má možnost přidat nového uživatele. Jelikož je tato funkce zpřístupněna pouze manažerům, tak odpovědný pracovník ručí za správnost vyplněných údajů. Při přiřazení čipu starého zaměstnance novému je manažer povinen smazat u starého zaměstnance záznam čipu.

Další funkcí zpřístupněnou manažerům je správa dat uživatele. Je možné změnit jméno a roli. Digitální otisk čipu je možné buď změnit, v případě výměny čipu, nebo zcela smazat.

Pro ztrátu uživatelova hesla má vedoucí pracovník možnost obnovit heslo jakémukoliv zaměstnanci. Provizorní heslo je číslo mezi čísly 1000000 a 10000001. Tento řetězec se zobrazí ve hlášení na obrazovce, takže je možné ho ihned změnit. Manažer se bere jako důvěryhodná osoba, která nezneužije této možnosti.

5 REST API

Datová část celého systému je tvořena API s architekturou REST (Representational State Transfer) a databázovým serverem. REST implementuje metody, které popisují akce nad zdroji. Základní metody jsou GET pro získání dat, POST pro vytvoření dat, DELETE pro mazání data a PUT pro úpravu dat. K jednotlivým zdrojům se přistupuje pomocí identifikátoru URI. Kromě dat lze poznat výsledek volání podle stavového kódu. V rozhraní jsou využívány kódy 200 pro OK stav s daty, 201 pro vytvoření nového záznamu, 204 pro úpravu nebo smazání záznamu, 404 pro nenalezení dat pomocí identifikátoru. Další chybové kódy jsou 400 pro špatný požadavek nebo špatných dat v požadavku a 401 pro neautorizovaný požadavek. Konečně jsou využity kódy 405 pro špatnou metodu a kódy ve formátu 5xx pro neočekávané chyby na serveru. API komunikuje s klientem ve formátu JSON. Tato část se blíže zabývá konstrukcí tohoto rozhraní (Zdrojak, 2009).

5.1 JWT

JSON Web Token je otevřený standard, který definuje možnost bezpečného přenosu informace mezi stranami jako JSON objekt. Každá informace může být lehce ověřena, jelikož je digitálně podepsána. Token je v rozhraní podepsán tajným klíčem. Ačkoliv může být pro bezpečnost přenášených dat token zašifrován, tak v tomto případě se přenáší pouze identifikátor přihlášeného uživatele, takže v případě odcizení tokenu je útočníkovi informace k ničemu. Když útočník token pozmění, díky podpisu s tajným klíčem selže verifikace. Token je složen ze tří částí. První částí je hlavička, kam se ukládá použitý algoritmus pro podepsání a typ tokenu. V rozhraní je algoritmus pro podepisování zvolen HMAC SHA256. Druhou částí je payload, kam se ukládá vydavatel tokenu, doba expirace, atributy definované serverem a jiné. Zde je nastavena expirační doba na nekonečně dlouhou dobu a atribut identifikátor uživatele, který si o token zažádal. Poslední částí je podpis. Vygenerovaný token lze využít v hlavičce požadavku, nebo jako argument URL adresy. API je nakonfigurováno, aby hledalo token v hlavičce požadavku. Hlavička využívaná pro token je ve tvaru `Authorization: Bearer <token>`. Tím se eliminují problémy spojené s CORS (Cross-Origin Resource Sharing).

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3OD
kwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxw
RJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

Zdrojový kód 3: Zakódovaný token

Zdroj: Jwt.io, 2019

Pro získání tokenu klient provede požadavek na endpoint `/authforstore`. Cílem funkce je ověřit uživatelské osobní číslo, heslo a vybranou pobočku. Jestliže uživatel zadal správnou kombinaci všech tří parametrů, je mu vygenerován token. Tento token je poté použit jako ověření u ostatních endpointů. V případě špatné kombinace je vrácen kód 401 (unauthorized). Díky modulu Flask-JWT-Extended lze kromě funkcí pro vytvoření tokenu a jeho validaci a verifikaci využít i dekorátor funkce `@jwt_required`. Dekorátor ověří platnost tokenu a autorizuje požadavek. Pro získání uživatele, pro kterého byl token vydán, slouží funkce `jwt_identity`, která je zaregistrována při inicializaci s instancí aplikace (Jwt, 2019).

5.2 Flask-Security

Aby nebyla hesla, pro případ jejich odcizení z databáze, uložena jako plain text, je vhodné z nich vytvořit hash. Pro tvorbu hashe je použito funkcí z modulu Flask-Security. Funkce `hash_password` využívá systému bcrypt, pomocí kterého vytvoří hash hesla, který je uložen do databáze. Pro porovnání hashe hesla s plain text heslem slouží funkce `verify_password` (Flask-login.readthedocs.io, 2012; Pythonhosted.org, 2012).

5.3 Flask-SQLAlchemy a modely

Rozšíření, které pomáhá implementovat ORM (Object-Relational Mapping) SQLAlchemy do frameworku Flask. Díky němu lze pracovat s code-first databázovým přístupem a v mnohém zjednodušuje získávání dat z databáze. Místo psaní selectů lze volat automaticky parametrizované dotazy. Díky propojení tabulek cizími klíči lze joinovat tabulky dle jasně daných kritérií (Flask-sqlalchemy.palletsprojects.com, 2010). Pro databázový systém jsou vytvořeny modely Store, Room, User, Entry a Sap.

Store je pobočka, která má svůj identifikátor jako primární klíč. Dále u pobočky evidujeme město a má vztah ke skladům (1:n) a k uživatelům (1:n).

Room zastupuje sklad. U skladu evidujeme rovněž identifikátor jako primární klíč a název. Provázanost je cizím klíčem na identifikátor pobočky. Sklad má vazbu na záznamy (1:n).

Entry znamená záznam pohybu zboží. U záznamu definujeme identifikátor jako primární klíč, ean, kusy, typ, datum, což jsou nenulovatelné atributy. Dále nulovatelné imei1, imei2, text, faktura. Je zde cizí klíč pro identifikátor skladu a pro identifikátor uživatele.

User je pro uživatele, kde je identifikátor jako primární klíč, osobní číslo, jméno, heslo, digitální otisk čipu, datum vytvoření a změny. Cizí klíče odkazují na identifikátor role a pobočky. Vztah k záznamům je 1:n.

Model Role slouží jako identifikace rolí uživatelů. Evidujeme u něj identifikátor jako primární klíč, název, popis a vztah k uživateli (1:n).

Posledním modelem je Sap, který slouží pro definici záznamů ze systému SAP. Pro chod aplikace je dobré znát identifikátor jako primární klíč, ORA, ITEM, EAN, text, kusy a kategorii. Zde není provázanost s tabulkou záznamů, jelikož není možné dodržet referenční integritní omezení. Nejsme schopni zaručit, že v době přidání záznamu do tabulky se bude nacházet alespoň jeden záznam se stejným klíčem v tabulce sap.

Pro každý model je definována hybridní metoda `serialize`, která pomáhá dále v kódu při serializaci modelu do formátu JSON. Pro obecnou serializaci je definována jediná funkce `serialize`, která na základě typu parametru zavolá serializaci jednoho objektu, nebo provede serializaci celého listu objektů.

```
class Entry(db.Model):
    _tablename_ = "entries"
    id = db.Column(db.Integer, primary_key=True)
    ean = db.Column(db.BigInteger, nullable=False)
    imei1 = db.Column(db.BigInteger, nullable=True)
    imei2 = db.Column(db.BigInteger, nullable=True)
    pieces = db.Column(db.Integer, nullable=False, server_default="1")
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)
    text = db.Column(db.Text, nullable=True)
    type = db.Column(db.Integer, nullable=False)
    invoice = db.Column(db.BigInteger, nullable=True)
    date = db.Column(db.TIMESTAMP, server_default=func.now())
    room_id = db.Column(db.Integer, db.ForeignKey('rooms.id'), nullable=False)

    @hybrid_method
    def serialize(self):
        return {
            'id': self.id,
            'ean': self.ean,
            'imei1': self.imei1,
            'imei2': self.imei2,
            'pieces': self.pieces,
            'user_id': self.user_id,
            'text': self.text,
            'type': self.type,
            'invoice': self.invoice,
            'date': self.date.timestamp(),
            'room_id': self.room_id
        }
```

Zdrojový kód 4: Model Entry s pomocnou metodou pro serializaci

Zdroj: Vlastní zpracování, 2019

5.4 Flask-Restful

Zpřehlednění aplikace využívající standardu REST je docíleno pomocí rozšíření Flask-Restful. Rozšíření pomáhá definovat jednotlivé endpointy a serializovat odpovědi do formátu JSON se správnými návratovými kódy. Zároveň poskytuje parser argumentů s jejich jednoduchou validací (Flask-restful.readthedocs.io, 2018).

Pomocí tohoto modulu byly vytvořeny třídy pro typy dotazů. Od jednoduchých požadavků, které spravuje například endpoint `/users`, pro který je vytvořena třída `UsersApi`, jež metodami post a put dovoluje zakládat a upravovat uživatele, po požadavky složitější. Složitějšími požadavky jsou požadavky s filtry. Například pro požadavek na posledních několik záznamů dle určitého skladu byla použita třída `TopEntries`. V dokumentacích a fórech je možné najít dva přístupy. První přístup definuje filtrovací požadavek v URI záznamu argumentů. Druhý přístup radí sestavit URI

a zakomponovat filtr přímo do něj. Dle best-practices a doporučení k architektuře REST jsem zvolil endpoint `/rooms/<room_id>/entries/top/<number>`, což znamená druhý přístup. Endpoint nám definuje, že chceme určitý počet top záznamů z určitého skladu (Hackernoon.com, 2017; Docs.microsoft.com, 2018, Medium.com, 2018; Code-maze.com, 2018; Stackoverflow.com, 2011; Stackoverflow.com, 2008).

Další správa zdrojů definuje třídy a cesty pro operace nad pobočkami, uživateli určité pobočky, sklady určité pobočky, záznamy pohybu zboží a záznamy tabulky sap.

Mimo správy zdrojů jsou definovány funkce pro autentizaci uživatele. Autentizace může probíhat na základě osobního čísla, hesla a pobočky nebo digitálního otisku čipu.

Dále je definován endpoint pro změnu hesla a vložení záznamů ze systému SAP do databáze. Funkce pro vložení dat do sap je podřízena datům ze systému SAP. Z klienta přichází JSON obsahující list slovníků. Nejprve je nutné smazat všechny záznamy z tabulky sap. Poté se projde celý list a pro každý slovník proběhne kontrola správnosti údajů. V případě chybějícího EAN kódu v záznamu je záznam přeskočen. Pro momentálně necelých 131 000 záznamů je to circa 5 jinak validních záznamů. Tato chyba bývá způsobena duplicitou záznamu v systému SAP a jiný záznam tento nahradí. Jelikož jsou klíče slovníku hlavičky sloupců, jsou klíče v českém jazyce s diakritikou a mezerami. Python tento styl klíčování zvládá bez problému. Potřebná úprava je u každého záznamu s klíčem kusů. SAP exportuje množství jako desetinné číslo, takže je provedena záměna čárky za tečku, aby mohly být kusy konvertovány na číslo. Po přidání všech slovníků do databázové session je třeba commitnout. V případě jakéhokoliv selhání je na celou session zavolán rollback, aby data z tabulky nezmizela úplně.

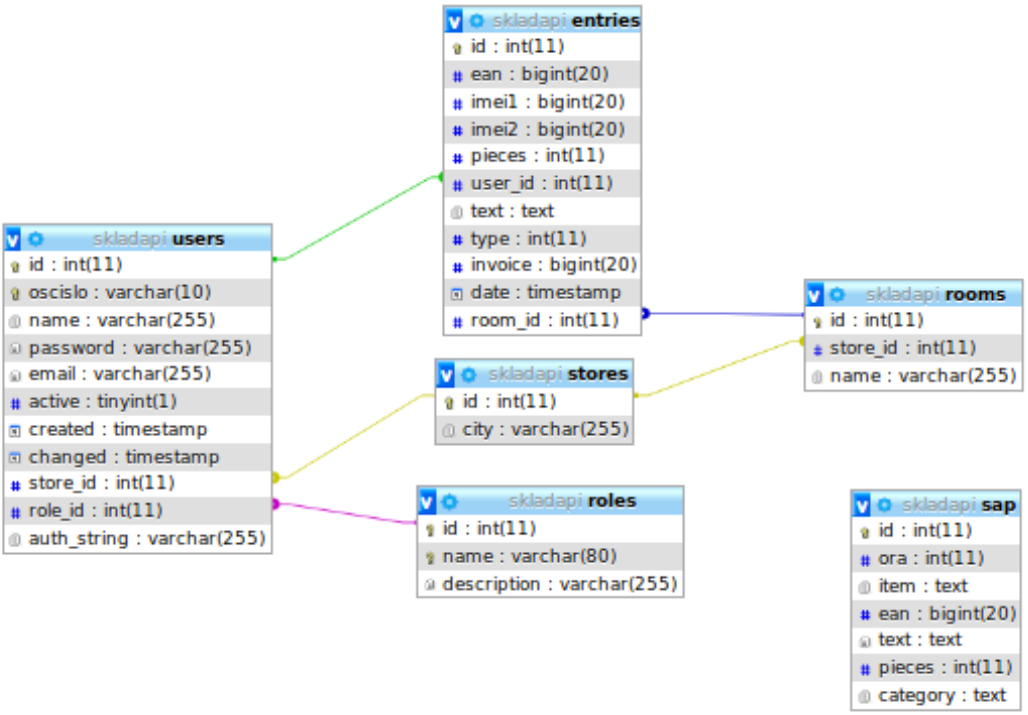
```
@app.route('/sap', methods=['POST'])
@jwt_required
def sap():
    try:
        all_list = json.loads(request.data)
        db.session.query(Sap).delete()
        for sap_entry in all_list:
            if sap_entry['Kód GTIN'] == '' or not sap_entry['Kód GTIN'].isdigit() or not sap_entry['Zboží'].isdigit():
                continue
            sap_e = Sap(ora=sap_entry['Zboží'], item=sap_entry['Model'], ean=sap_entry['Kód GTIN'],
                        text=sap_entry['Krát.text zboží'], pieces=sap_entry['Volně použitelná'].replace(',', '.'),
                        category=sap_entry['Kateg.zboží'])
            db.session.add(sap_e)
        db.session.commit()
        return jsonify({'ok': True})
    except:
        db.session.rollback()
        return jsonify({'ok', False})
```

Zdrojový kód 5: Zpracování dat ze SAP v API

Zdroj: Vlastní zpracování, 2019

6 Databáze

Nedílnou součástí takového systému je databázový systém. Vzhledem ke zkušenostem jsem si vybral databázový server MySQL běžící na operačním systému Ubuntu. MySQL má dobrou podporu u používaného ORM rozšíření. Schéma databáze je popsáno v sekci o SQLAlchemy a modelech.



Obrázek 5: Databázové schéma

Zdroj: Vlastní zpracování dle dat z phpMyAdmin

6.1 Inicializace databáze

Rozšíření SQLAlchemy je dobrým nástrojem pro vytvoření databáze přístupem code-first. Jasně definované sloupce v modelech aplikace s definovanými vztahy a omezeními jsou vítaným přístupem třeba při migraci databázového systému (Flask-sqlalchemy.palletsprojects.com, 2010). Byl vytvořen skript, který dropne a následně inicializuje celou databázi funkcí `create_all`. Následně proběhne vložení prvního uživatele, poboček a jejich skladů.

7 Možnosti vylepšení

Během návrhu a zpracování tohoto systému autor narazil na možná budoucí vylepšení systému. Každé ze zmíněných vylepšení závisí na různých faktorech. V této části jsou prezentovány autorovy názory.

7.1 Programovací jazyk

Na základě dohody s vedoucím práce byl zvolen programovací jazyk Python. Vzhledem k jiným možnostem má Python své přednosti a framework Flask značně zjednoduší práci na takové aplikaci. V budoucnu by se autor rozhodl pro JavaScript. Full stack JavaScriptová aplikace by měla nabídnout rychlejší zpracování dat a lepší možnosti pro programátora. Vhodná alternativa pro Python a Flask je systém Node.js a například framework express. Pro použití jiného jazyka by se musela mírně upravit struktura API, aby mohlo například zpracovávat soubory. Klientský kód by se prováděl ve webovém prohlížeči, takže odpadne serverová část aplikace, která se stará o vykreslování a komunikaci s API.

7.2 SAP

Získávání dat ze systému SAP je nyní nestabilní, vyžaduje uživatelskou interakci a spoléhá na lidský faktor a konzistenci externího systému. Pro lepší funkčnost systému je třeba lepší dohoda s firmou, která by byla ochotna poskytnout přístup k datům jinou cestou.

Možností by byl skript na straně firmy, který by se na základě nějaké události nebo v určitých intervalech spouštěl. Skript by provedl POST požadavek na API. Rozhraní by poté zpracovalo data v požadavku.

Další možností by byl přístup ke službě, která spravuje data v databázi systému SAP. Zde by stačil endpoint na GET požadavek s filtrem. Služba by vrátila relevantní filtrovaná data. Tento požadavek by se odesílal v případě, když API vyhodnotí, že nemá aktuální data.

7.3 RFID tagy a brány

Pro snížení účinku lidského faktoru na aplikaci je nutné investovat do technologie. Vzhledem k financování celého systému ze strany společnosti nebyla možnost představit koncept založený na RFID technologii. Kromě zjednodušení celého procesu výdeje zboží by taková změna znamenala rozsáhlou změnu procesu příjmu zboží.

RFID (Radio Frequency Identification) je technologie pro identifikaci založenou na elektromagnetických vlnách. Systém pro identifikaci zboží může mít podobu tagu nebo takzvané chytré etikety a snímače.

Výhodou tagů je jejich znovupoužitelnost a výhodou etiket je množství informací viditelných pro uživatele. Na etiketu je možné vytisknout informace o zboží a třeba i čárový kód. Vzhledem k řešení je cena takové RFID tiskárny příliš vysoká (od 30 000 Kč). Daleko vhodnější jsou například jednoduché samolepící tagy (pouze režim čtení) dostupné od 2 Kč, nebo znovupoužitelné tagy (zápis), které lze sehnat od 6 Kč. Do znovupoužitelných tagů lze zapisovat více informací, což by usnadnilo a zrychlilo výdej zboží.

Podoba snímače může být malé čtecí zařízení nebo brána. V případě, že by firma trvala na doplňování dodatečných informací, stačilo by USB zařízení. Druhou možností, která je z hlediska bezpečnosti a kontroly lepší, je brána. Její cena se pohybuje od 30 000 Kč a lze ji nainstalovat do vchodu skladu. Pak by bylo možné sledovat pohyb zaměstnance včetně zboží a jeho směru. Takové řešení by postrádalo část ze svého informativního charakteru. Zaměstnanec nemá kontrolu vydávaného zboží a manažer neví, za jakým účelem zboží ze skladu odešlo (Esp.cz, 2014).

Dobrým řešením je oba přístupy spojit. Pomocí serveru, který komunikuje s bránou a aplikací, která komunikuje s USB čtečkou, lze poznat, kdy zboží ze skladu odchází v základním nastavení (například výdej – prodej) a kdy je k němu pomocí USB zařízení přidružena operace zadaná uživatelem.

Závěr

Tématem této bakalářské práce bylo navrhnout a vytvořit aplikaci pro skladové hospodářství. Tato aplikace měla být šita na míru potřebám firmy. Pro firmu byl hlavním důvodem vzniku aplikace chybějící bezpečnostní prvek. Společnost Electro World s.r.o. sice podniká bezpečnostní kroky zabezpečení zboží, jako jsou například kamery nebo menší bezpečnostní sklady přístupné pouze určitým osobám, ale chybějící evidence pohybu zboží často znemožňovala nalezení určitého zboží v širokém oběhu. Zároveň bylo potřeba zjednodušit inventuru výrobků citlivých na krádeže, a pokud možno za pomoci všech prvků včas odhalit zcizení a podniknout patřičná opatření.

Výsledkem práce je aplikace, která je využívána zaměstnanci firmy a díky možnosti implementovat záznamy ze systému SAP, se stala nedílnou součástí firmy. Díky ní firma šetří peníze za čas zaměstnanců dříve strávený hledáním zboží ve vícero skladech. Nyní lze díky jednoduchému přehledu zjistit, že se zboží nalézá právě v konkrétním skladu. Skladník nyní nemusí na dotaz prodejce hledat zboží ve skladech. V přehledu zjistí, že zboží bylo vystaveno, odesláno na jinou prodejnu, reklamováno u dodavatele, nebo že není ve skladu z jiné příčiny. Takto lze nápomocně suplovat systém SAP, který pro zjištění takových úkonů funkce nemá vůbec, nebo jsou pro běžného uživatele zdlouhavé a složité.

Požadavky na řešení od společnosti Electro World s.r.o. byly hlavně rychlost obsluhy, jednoduchost, propojitelnost se systémem SAP, překonání technického omezení, a především nízká cena. Aplikace je z větší části složena z formulářů, které jsou intuitivní, a přesto obsahují všechny náležitosti a prvky, které jsou pro obsluhu potřeba. Pro zaručení rychlosti obsluhy bylo provedeno krátké školení zaměstnanců, kde jim byly vysvětleny principy funkčnosti a možnosti aplikace. V průběhu byl několikrát změněn hlavní formulář pro přidání záznamu.

V rámci propojitelnosti se systémem SAP se uživatelé musí spolehnout sami na sebe a na své manažery. Každý manažer může ze systému SAP exportovat seznam zásob, který lze ve formátu csv nahrát do aplikace. V poslední části této práce byly popsány možnosti vylepšení komunikace systémů jako třeba otevření endpointu SAP pro získání seznamu zásob.

Pro maximální využití potenciálu technického vybavení prodejen byl na každý sklad vyčleněn počítač, který slouží jako pracovní stanice uživatelů. Na ten bylo povoleno připojit čtečky EAN kódů, jež byly modifikovány pro uživatelský komfort. Dále k nim byly připojeny RFID čtečky, které jsou využívány pro rychlejší identifikaci uživatele.

Momentálně navržené řešení stojí firmu pouze náklady na čas zaměstnanců při příjmu a výdeji zboží, čas manažerů při správě údajů ze SAP a náklady na technické prostředky jako je hosting aplikace a využití starších modelů počítačů. Čas zaměstnanců je zde zcela kompenzován časem, který by zaměstnanci strávili při hledání zboží a jiných úkonech se zbožím spojené. Zároveň takto nastavený bezpečnostní prvek kryje každého zaměstnance před neoprávněným nařknutím, nebo jeho vlastními náklady při kolektivní hmotné odpovědnosti za zboží. Starší počítače, které firma jako pracovní stanice využívá, jsou připraveny na odpis. Takto může společnost zastaralé a pomalé modely využít bez další investice. Náklady na technické prostředky jsou při hostování takového systému jsou na velmi malé úrovni. Navíc systém ze strany IT oddělení firmy nepotřebuje další zásahy.

Celá aplikace je řešena v programovacím jazyce Python za pomoci frameworku Flask a několika knihoven. V poslední části práce se autor zabývá myšlenkou vhodnosti využití právě tohoto nástroje. Pro správu provázaných dat slouží databázový server MySQL a jeho propojení s SQLAlchemy, jež umožňuje code-first přístup, utvořilo jednotnou datovou vrstvu. Pro práci nad touto vrstvou byla zvolena architektura REST API.

Řešení vyhovuje požadavkům firmy, nicméně by takové řešení mohlo být mnohem komplexnější za využití modernějšího přístupu k evidenci pohybu zboží. Autor se v poslední části zabývá možného využití RFID tagů, bran, což by vedlo ke složitějšímu a dražšímu, ale vzhledem k lidskému faktoru, bezpečnějšímu návrhu.

Citace

Code-maze. Top REST API Best Practices. *CODE MAZE* [online]. 2018 [vid. 2019-03-11]. Dostupné z: https://code-maze.com/top-rest-api-best-practices/

Codementor. Flask vs. Django: WhyFlaskMightBeBetter. *Codementorcommunity* [online]. 2017 [vid. 2019-03-10]. Dostupné z: https://www.codementor.io/garethdwyer/flask-vs-django-why-flask-might-be-better-4xs7mdf8v

Cvedetails. PHP: Vulnerability Statistics. *CVE Details* [online]. 2019 [vid. 2019-03-10]. Dostupné z: https://www.cvedetails.com/product/128/PHP-PHP.html?vendor_id=74

Docs.microsoft. API design. *Microsoft Azure* [online]. 2018 [vid. 2019-03-11]. Dostupné z: https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design

Dotekomanie. Potřebujete IMEI k ukradnutému mobilu? Google ho zná. *Dotekomanie* [online]. 2015 [vid. 2019-03-11]. Dostupné z: https://dotekomanie.cz/2015/01/potrebujete-imei-ukradnutemu-mobilu-google-ho-zna/

Esp. Jak fungují RFID čtečky. *ESP* [online]. 2014 [vid. 2019-03-10]. Dostupné z: https://esp.cz/cs/blog/funguji-rfid-ctecky

Flask-login.readthedocs. Flask-Login. *Flask login* [online]. 2012 [vid. 2019-03-11]. Dostupné z: https://flask-login.readthedocs.io/en/latest/

Flask.pocoo: Application Factories. *Flask.pocoo* [online]. 2010 [vid. 2019-03-11]. Dostupné z: http://flask.pocoo.org/docs/0.12/patterns/appfactories/

Flask.pocoo. Flask - web development, one drop at a time. *Flask* [online]. 2019 [vid. 2019-03-10]. Dostupné z: http://flask.pocoo.org/

Flask.pocoo. Routing. *Flask.pocoo* [online]. 2010 [vid. 2019-03-11]. Dostupné z: http://flask.pocoo.org/docs/1.0/quickstart/#routing

Flask.pocoo. View Decorators. *Flask.pocoo* [online]. 2010 [vid. 2019-03-11]. Dostupné z: http://flask.pocoo.org/docs/0.12/patterns/viewdecorators/

Flask-restful.readthedocs. Flask RESTful. *Flask-restful* [online]. 2018 [vid. 2019-03-11]. Dostupné z: https://flask-restful.readthedocs.io/en/latest/

Flask-sqlalchemy.palletsprojects. Flask SQLAlchemy. *Flask-sqlalchemy* [online]. 2010 [vid. 2019-03-11]. Dostupné z: https://flask-sqlalchemy.palletsprojects.com/en/2.x/

Flask-wtf.readthedocs. Simple integration of Flask and WTForms, including CSRF, file upload, and reCAPTCHA. *Flask WTF* [online]. 2018 [vid. 2019-03-11]. Dostupné z: https://flask-wtf.readthedocs.io/en/stable/

Hackernoon. RESTful API Designing guidelines—The best practices. *Hackernoon* [online]. 2017 [vid. 2019-03-11]. Dostupné z: https://hackernoon.com/restful-api-designing-guidelines-the-best-practices-60e1d954e7c9

Insights.stackoverflow. Developer SurveyResults 2016 - Technology. *Stackoverflow* [online]. 2016 [vid. 2019-03-10]. Dostupné z: https://insights.stackoverflow.com/survey/2016#technology

Jwt. Introduction to JSON Web Tokens. *JWT* [online]. 2019 [vid. 2019-03-10]. Dostupné z: https://jwt.io/introduction/

Kodys. EAN 13 a EAN 8 - nejznámější čárový kód pro zboží v obchodní síti. *Kodys* [online]. 2018 [vid. 2019-03-11]. Dostupné z: https://www.kodys.cz/technologie/carovy-kod/ean-13-ean-8

Medium. REST: Good Practices for API Design. *Medium* [online]. 2018 [vid. 2019-03-11]. Dostupné z: https://medium.com/hashmapinc/rest-good-practices-for-api-design-881439796dc9

Nuget. Create .NET apps faster with NuGet. *Nuget* [online]. 2019 [vid. 2019-03-11]. Dostupné z: https://www.nuget.org

Pythonhosted. Flask-Security. *Pythonhosted* [online]. 2012 [vid. 2019-03-11]. Dostupné z: https://pythonhosted.org/Flask-Security/

Stackoverflow. How to design RESTful search/filtering? *Stackoverflow* [online]. 2011 [vid. 2019-03-11]. Dostupné z: https://stackoverflow.com/questions/5020704/how-to-design-restful-search-filtering

Stackoverflow. RESTful URL design for search. *Stackoverflow* [online]. 2008 [vid. 2019-03-11]. Dostupné z: https://stackoverflow.com/questions/207477/restful-url-design-for-search

Wiki.python. Web Frameworksfor Python. *Python* [online]. 2019 [vid. 2019-03-10]. Dostupné z: https://wiki.python.org/moin/WebFrameworks

Zdrojak. REST: architektura pro webové API. *Zdroják* [online]. 2009 [vid. 2019-03-10]. Dostupné z: https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/